MoMo: Una Infraestructura basada en Grids para Museos Híbridos

Tesis Doctoral

UNIVERSIDAD
POLITECNICA
DE VALENCIA

Javier Jaén Martínez

fjaen@dsic.upv.es

Departamento de Sistemas Informáticos y Computación

UNIVERSIDAD POLITÉCNICA DE VALENCIA

Director: Dr. José H. Canós Cerdá

Resumen

En el presente trabajo analizamos la situación actual de los museos en el contexto de la aplicación de las tecnologías de la información y abordamos aspectos que tienen que ver con los cuatro usos básicos de la información (documentación, visualización, información e interrelación) en los museos. Veremos cómo la incorporación de las nuevas tecnologías en los últimos años ha hecho evolucionar el concepto de museo hasta la situación actual en la que las tecnologías de carácter ubicuo se traducen en nuevos escenarios que plantean nuevos desafíos.

En particular, el presente trabajo define un nuevo modelo de Museo Híbrido y su correspondencia con el marco ontológico *Conceptual Referente Model* propuesto por el *International Council of Museums*. A continuación, propone una infraestructura de federación semántica para la interoperabilidad a bajo nivel entre repositorios semánticos definidos según el entorno *Resource Description Framework* (RDF) y evalúa sus prestaciones. Seguidamente, define el problema de la orientación en el contexto de los Museos Híbridos y plantea su resolución mediante el uso de algoritmos evolutivos basados en colonias de hormigas y propone un algoritmo distribuido basado en Grids computacionales para la resolución aproximada del problema de la orientación en tiempo polinomial mostrando empíricamente sus buenas prestaciones. Finalmente, proporciona la resolución del problema de la orientación aplicando dos técnicas diferenciadas de aceleración gráfica: una basada en el procesamiento de vértices y otra en el de fragmentos mediante el uso de un procesador gráfico de última generación y se muestra empíricamente sus buenas propiedades.

Con todo ello se demuestran las ventajas de disponer de una infraestructura distribuida basada en Grids, que permite tanto una mejor gestión de datos pertenecientes a diversas organizaciones, como el procesamiento de algoritmos de complejidad no polinomial aplicables al contexto de los museos.

Resum

En el present treball analitzem la situació actual dels museus en el contexte de l'aplicació de les noves tecnologíes i abordem aspectos relacionats amb els quatre usos bàsics de la informació (documentació, visualització, informació i interrelació) en els museus. Veurem com la incorporació de les noves tecnologíes als últims anys ha fet evolucionar el concepte de museu fins la situació actual on les tecnologíes de carácter ubicu es tradueixen en nous escenaris que plantegen nous desafiaments.

En particular, el present treball defineix un nou model de Museu Híbrid i la seua correspondencia amb el marc ontologic *Conceptual Referente Model* proposat pel *International Council of Museums*. A continuació, propasa una infraestructura de federació semántica per a la interoperabilitat a baix nivell entre repositoris semàntics definits en termes de l'entorn *Resource Description Framework* i evalua les seues prestacions. Seguidament, defineix el problema de l'orientació en el contexte dels Museus Híbrids i planteja la seua solució mitjançant l'ús d'algorismes evolutius basats en colonies de formigues i proposa un algorisme distribuit basat en Grids de computació per a la solució aproximada del problema de l'orientació en temps polinomial mostrant empíricament les seues bones prestacions. Finalment, proporiona la solució del problema de l'orientació aplicant dues técniques diferenciades d'acceleració gràfica: una basada en el procesament de vertex i otra en el de fragments mitjançant l'ús d'un procesador gràfic d'última generació i mostrant empíricament les seues bones propietats.

Amb tot això es demostren els aventajes de desposar d'una infraestructura distribuida basada en Grids que permet tant una millor gestió de dades que pertanyen a diverses organitzacions, com el procesament d'algorismes de complexitat no polinomial aplicables al contexte dels museus.

Abstract

In this work we analyze the current situation of museums with respect to information technology deployments. We deal with aspects related to the four basic uses of information (documentation, visualization, information and interrelation) in museums. We will discuss how the latest technological advancements have shaped a new concept of museum, especially those with ubiquitous natures that make it posible to consider new scenarios and challenges.

Particularly, this work defines firstly a new model of Hybrid Museum and its correspondence with an ontological reference framework known as the Conceptual Reference Model proposed by the International Council of Museums. Secondly, a semantic federation infrastructure is proposed to obtain low level interoperable semantic repositories as defined with the Resource Description Framework and its performance is evaluated. Thirdly, the orienteering problem is defined in the context of Hybrid Museums and a solution is presented in terms of evolutionary strategies using ant colonies. The proposed algorithm is based on computational Grids to obtain approximate solutions in polynomial time and its good performance behavior is discussed.

Finally, an improved evolutionary algorithm is defined using two different graphical acceleration techniques, based on vertex and fragments processing respectively. An advanced generation graphical processing unit is used and experimental results demonstrating its enhanced performance are presented.

In summary, all the contributions of this work show the advantages of having a distributed infrastructure based on Grids that allow not only a better management of multi-organizational information, but also the processing of algorithms on non polynomial complexity that are needed in the context of Hybrid Museums.

Palabras clave

Palabras clave:

Museos Híbridos, Grids, Colonias de hormigas, Computación evolutiva, Web Semántica, Federaciones

Keywords:

Hybrid Museums, Grids, Ant Colonies, Evolutionary Computing, Semantic Web, Federations

Paraules clau:

Museus Híbrids, Grids, Colonies de formigues, Computació evolutiva, Web Semàntica, Federacions

"El Beso" por Klimt (1906)

Dedicado,

A Elena por hacerme feliz cada dia con su amor, por su apoyo incondicional, por su interminable paciencia y por su impagable dominio del Word.

A mis padres Joaquín y Encarna por haberme dado la vida y por su constante sacrificio para que yo haya podido llegar hasta aquí.

A mis hermanos Encarna, Angeles y Joaquín por haber creado en mí el hábito del estudio y por su ayuda siempre que los he necesitado.

A mis sobrinos Cristina, Sara y Gonzalo, por lo mucho que disfruto cuando estoy con ellos y por enseñarme cada vez algo nuevo.

Datos de la tesis

Título de la tesis:	MoMo una infraestructura basada en grids para museos híbridos
Presentada por:	Javier Jaén Martínez fjaen@dsic.upv.es
Dirigida por:	Jose H. Canós Cerdá Jhcanos@dsic.upv.es
Universidad:	Universidad Politécnica de Valencia.
Departamento:	Sistemas Informáticos y Computación.
Programa de doctorado:	Programación declarativa e Ingeniería de la Programación. Memoria para optar al grado de Doctor en Informática
Depósito:	Valencia, a 28 de Abril de 2006
Defensa:	

Agradecimientos

Este trabajo no habría sido posible sin el apoyo financiero de los laboratorios de investigación de Microsoft Research en Cambridge. En particular, quiero agradecer la participación de Pierre louis Xech quien consideró interesante este proyecto y apoyó su financiación.

Tampoco habría sido posible la realización del mismo sin los becarios y proyectandos de fin de carrera que han dedicado siempre más horas de las que debían y han mostrado un entusiasmo fuera de lo común. Mi reconocimiento y agradecimiento a Artur Boronat, Vicente Bosch, Alejandro Catalá, Kristian Eide, Jose Miguel Esteve, y Jose Antonio Mocholí. Mi más sincera felicitación a los que ganaron la competición de Imagine Cup España Vicente, Jose Antonio, Jose Miguel y Kristian que demostraron ser unos alumnos excepcionales.

Mi agradecimiento también a Hilario Canós por la dirección de esta tesis y por la impagable tarea de revisión y corrección de este documento.

No quiero olvidar a todos los miembros del grupo ISSI que cada dia me han mostrado su apoyo y a los que animo a seguir trabajando tan bien como han hecho hasta ahora sin desfallecer a pesar de las dificultades.

Tampoco quiero olvidar a otros miembros y amigos del departamento con quien he pasado momentos estupendos. Especialmente quiero agradecer a Isabel Diaz y a Jennifer Pérez las animadas conversaciones de café en la Vella y su estupendo carácter a pesar de las múltiples dificultades que se han cruzado en su camino.

También quiero dar las gracias a mi segunda familia en Albacete y muy especialmente a Elena y a Jose María por su cariño y por acogerme siempre con los brazos abiertos cada vez que voy a la Mancha.

Finalmente, mi agradecimiento a todos mis amigos con los que no he podido estar todo el tiempo que hubiese querido y que siempre me han mostrado su apoyo.

Índice general

1. Introducción *1*

1.1. Motivación **1**

1.2. Museos e Internet **3**

1.3. Planteamiento del Problema: Museos Híbridos **6**

1.3.1. Medios y Sentidos: Hacia los Museos del Futuro 8

1.4. Metodología de trabajo **9**

1.5. Resumen de las Aportaciones **10**

1.6. Estructura de la Tesis **10**

1.6.1. Organización 10

1.6.2. Convenciones tipográficas 11

2. Situación Actual: Museos Híbridos *15*

2.1. Tecnologías para Museos Híbridos **15**

2.1.1. Redes Inalámbricas 16

2.1.2. Tecnologías de Localización en el Espacio 18

2.1.3. Dispositivos Móviles 19

2.2. Proyectos de Museos Híbridos **20**

2.2.1. Urban Tapestries 20

2.2.2. MUSEpad 21

2.2.3. Equator 22

2.2.4. Handscape 23

2.2.5. Tate Modern 24

2.2.6. Science Navigator 25

2.2.7. Getty Guide 25

2.2.8. Otros Proyectos 26

2.3. Conclusiones del Capítulo **26**

3. El Modelo MoMo de Museo Híbrido *29*

3.1. Aprendizaje en Museos Híbridos **29**

3.1.1. Teorías del Aprendizaje en Museos 30

3.1.2. Aprendizaje in situ en Museos 31

3.2. Modelos Existentes de Museos Híbridos **33**

3.2.1. Proyecto iTour 33
3.2.2. Proyecto Multimedia Tour 34
3.2.3. Proyecto GettyGuide 35
3.2.4. Proyecto GuideBook 35
3.2.5. Otros Proyectos 37

3.3. El Modelo de Contenidos de MoMo 37

3.4. El Modelo de Navegación de MoMo 42

3.5. El Modelo de Interacción Social de MoMo 44

3.6. El Modelo Orientado a Servicios de MoMo 48

3.7. La Semántica del Modelo MoMo 51
3.7.1. El Modelo CRM 51
3.7.2. El Modelo MOMO- CRM 54

3.8. Conclusiones del Capítulo 61

4. Federaciones Semánticas en Museos Híbridos 63

4.1. Introducción 63

4.2. La Evolución de la Computación Global 64

4.3. La Evolución de la Web 67

4.4. El Problema de la Integración de Repositorios Semánticos Distribuidos 72

4.5. Una Visión Integradora: Las Federaciones Semánticas 74

4.6. Modelo Arquitectónico de Federación Semántica 80
4.6.1. Nivel de Recurso: Repositorios Semánticos RDF basados en WS 81
El Modelo Redland 81
Acceso Remoto a Repositorios Semánticos Mediante WS 82
4.6.2. Nivel Colectivo: Federaciones Semánticas basadas en WS 84
Gestión de Miembros de la Federación 86
Gestión de Colecciones RDF Federadas 86
Navegación Federada 90

4.7. Evaluación Empírica 90
4.7.1. Implementacion 90
4.7.2. Inserción de aserciones 92
4.7.3. Consulta de aserciones 92

4.8. Federaciones Semánticas en Museos Híbridos 94

4.9. Conclusiones 94

5. Orientación en Museos Híbridos Mediante Colonias de Hormigas 97

5.1. Orientación en Museos Híbridos 97

5.2. Algoritmos para la Resolución del Problema de la Orientación 100

5.3. Optimización Basada en Colonias de Hormigas 102
5.3.1. Problemas Resolubles mediante OCH 103
5.3.2. Hormigas Artificiales 104
5.3.3. Modelos de OCH 105

5.4. Algoritmos de OCH Paralelos 109

5.5. Algoritmo de OCH Basado en Grid 111
5.5.1. Clusterización de Instancias del Problema de la Orientación 112
5.5.2. Estrategia de Integración de Soluciones Parciales 113
5.5.3. Implementación 115
5.5.4. Resultados Experimentales 117

5.6. Conclusiones del capítulo 125

6. Técnicas de Resolución de OCH Mediante Unidades Gráficas de Procesamiento 127

6.1. Introducción a la GPU 127
6.1.1. El "Pipeline" gráfico hardware 128
6.1.2. Revisión histórica de las GPU 134
6.1.3. El "Pipeline" gráfico programable en GPU de 5ª Generación 137
6.1.4. El Procesador de Vértices 138
6.1.5. El Procesador de Fragmentos 138
6.1.6. GPGPU, API 3D y lenguajes de Alto Nivel para GPU 139

6.2. Algoritmo Orientado a Vértices para la Resolución de OCH mediante GPU 142
6.2.1. Aproximación intuitiva al algoritmo gráfico 144
6.2.2. Algoritmo GPU-VERTEX-OCH-OP 150
Estructuras de Datos 150
Proceso Algorítmico 153
6.2.3. Implementación 158
6.2.4. Resultados Experimentales 161

6.3. Algoritmo Orientado a Fragmentos para la Resolución de OCH mediante GPU 168
6.3.1. Aproximación intuitiva al algoritmo GPU-Fragment-OCH-OP 169
6.3.2. Algoritmo GPU-Fragment-OCH-OP 176
Estructuras de Datos 176
Proceso Algorítmico 177
6.3.3. Resultados Experimentales de GPU-Fragment-OCH-OP 180

6.4. Conclusiones del capítulo 189

7. Conclusiones 191

ANEXO A. Especificación de requisitos IEEE 209

ANEXO B. Programas HLSL de GPU-Vertex-OCH-OP 263

ANEXO C. Programas HLSL de GPU-Fragment-OCH-OP 269

Índice de Figuras

Figura 1. Páginas indexadas por Google ____ *4*
Figura 2. Popularidad de Enlaces en Google y MSNSearch ____ *5*
Figura 3. Estructura Jerárquica del proyecto iTour ____ *34*
Figura 4. Navegación lineal en GettyGuide ____ *35*
Figura 5. Navegación basada en páginas Web de GuideBook ____ *37*
Figura 6. Modelo de clases UML básico de contenidos ____ *39*
Figura 7. Modelo de contenidos aumentado ____ *42*
Figura 8. Estructuras de Navegación Horizontal y Vertical ____ *43*
Figura 9. Ejemplo de navegación vertical ____ *44*
Figura 10. Modelo de clases para la Interacción Social. ____ *45*
Figura 11. Modelo de usuario. ____ *46*
Figura 12. Sesión de un visitante. ____ *46*
Figura 13. Modelo de notificaciones. ____ *47*
Figura 14. Datos de un grupo y de sus miembros. ____ *48*
Figura 15. Metamodelo cualitativo de CRM. Fuente: (Doerr,2003) ____ *53*
Figura 16. Razonando sobre información espacial en CRM ____ *53*
Figura 17. Razonando sobre Información temporal en CRM ____ *54*
Figura 18. Modelo de usuario MoMo-CRM ____ *55*
Figura 19. Modelo de Colecciones MoMo-CRM ____ *56*
Figura 20. Modelo de Visitas MoMo-CRM I ____ *57*
Figura 21. Modelo de Visitas MoMo-CRM II ____ *58*
Figura 22. Modelo de Interacción Social MoMo-CRM I ____ *59*
Figura 23. Modelo de Interacción Social MoMo-CRM II ____ *60*
Figura 24. Modelo de Interacción Social MoMo-CRM III ____ *61*
Figura 25 Modelo arquitectónico para aplicaciones de computación global ____ *66*
Figura 26 Modelo en Niveles para la Web Semántica ____ *69*
Figura 27 Modelo Conceptual Simplificado de Federación Semántica ____ *77*
Figura 28 Vistas interna y externa de una aserción RDF distribuida ____ *78*
Figura 29 Nivel de Recurso para acceso remoto a Repositorios RDF ____ *83*
Figura 30 Modelo Conceptual Arquitectónico Federación Semántica ____ *85*
Figura 31 Miembros de una Federación Semántica ____ *86*
Figura 32 Comparativa inserción de una aserción para Redland, RedlandNET, DRedlandNET and DRedlandNET con caché de modelos. ____ *92*
Figura 33 Comparativa número constante de consultas para Redland y RedlandNET. ____ *93*
Figura 34 Comparativa número constante de consultas para Redland, RedlandNET y DRedlandNET. ____ *93*
Figura 35 Modelo conceptual de Grid-OCH-OP ____ *116*

Figura 36. OCH-GRID-OP con 1000 nodos y VNS intensivo. 120
Figura 37. OCH-GRID-OP con 1000 nodos y VNS relajado. 121
Figura 38. Experimento con 1000 nodos y densidad 10%. 122
Figura 39. Experimento con 1000 nodos y densidad 50%. 122
Figura 40. Experimento con 1000 nodos y densidad 90%. 123
Figura 41. Experimento con 5000 nodos y VNS relajada. 123
Figura 42. Experimento con 10000 nodos y VNS relajada. 124
Figura 43. Time and score comparison using 32 colonies. 124
Figura 44. Descripción gráfica del procesamiento del pipeline. 128
Figura 45. El color y sus canales 129
Figura 46. Clipping y Culling 130
Figura 47. Resumen de sistemas de coordenadas y sus transformaciones 134
Figura 48.Descripción lógica de los componenetes del pipeline gráfico 137
Figura 49. Vértices, primitivas, fragmentos generados y mapeo de textura 141
Figura 50. Contenido de la textura de Tour 145
Figura 51. Contenido de la textura Prohibidos 146
Figura 52. Descripción gráfica del algoritmo: componentes 146
Figura 53. Descripción gráfica del algoritmo: Construcción de caminos 147
Figura 54. Descripción gráfica del algoritmo: Marcado de visitados 148
Figura 55. Espacio de recortado con vista paralela ortográfica 149
Figura 56. Selección del siguiente nodo a visitar 150
Figura 57. Contenido de un texel de la textura Grafo 151
Figura 58. Contenido de un texel de la textura Tour 152
Figura 59. Cálculo de coordenadas de textura 159
Figura 60. Problemas de las coordenadas de textura 160
Figura 61. Variantes de GPU-VERTEX-OCH-OP: Tiempo para N=1000 162
Figura 62. Variantes de GPU-VERTEX-OCH-OP: Score para N=1000 163
Figura 63. Variantes de GPU-VERTEX-OCH-OP: Time para N=3000 163
Figura 64. Variantesde GPU-VERTEX-OCH-OP: Score para N=3000 164
Figura 65. GPUvs. GRID: Tiem para N=1000 165
Figura 66. GPU vs. GRID: Score para N=1000 166
Figura 67. GPU vs. GRID: Timer para N=3000 167
Figura 68. GPU vs. GRID: Score para N=3000 167
Figura 69. Componentes del algoritmo GPU-Fragment-OCH-OP 169
Figura 70. Componentes de GPU-Fragment-OCH-OP: Selección de nodos 170
Figura 71. Espacio de recortado y disposición de las líneas 171
Figura 72. Fragmentación de líneas en GPU-Fragment-OCH-OP 172
Figura 73. Selección del siguiente nodo a visitar en GPU-Fragment-OCH-OP 174
Figura 74. Componentes de GPU-Fragment-OCH-OP: Marcado de visitados 174
Figura 75. Vértices de Marcado en GPU-Fragment-OCH-OP 175
Figura 76. GPU-Fragment: Izq.: Sección de la vista. Der.: Fragmentación 175
Figura 77. GPU-Fragment: Izq.: Vértices con coordenadas de textura. Der.: fragmentos con coordenadas de textura interpoladas. 177
Figura 78. Contenido de un texel de la textura de Tour en GPU-Fragment-OCH-OP 177
Figura 79. Variantes de GPU-Fragment-OCH-OP: Score para N=1000. 181
Figura 80. Variantes de GPU-Fragment-OCH-OP: Time para N=1000 181
Figura 81. GPU-Vertex vs. GPU-Fragment: Score para N = 1000. 183
Figura 82. GPU-Vertex vs. GPU-Fragment: Time para N = 1000. 184
Figura 83. GPU-Vertex vs. GPU-Fragment: Score para N = 3000. 185

Figura 84. GPU-Vertex vs. GPU-Fragment: Time para N = 3000. _____185
Figura 85. Comportamiento temporal teórico. N = 1000. _____186
Figura 86. GPU-Vertex vs. GPU-Fragment: Time para N=1000 y H hasta 128. _____187
Figura 87. Time para algoritmos de GPU. N=1000 y H hasta 128 _____187
Figura 88. GPU-Vertex vs. GPU-Fragment: Score para N=3000 y H hasta 128 _____188
Figura 89. GPU-Vertex vs. GPU-Fragment: Time para N=3000 y H hasta 128 _____188

"Pintura Rupestre de Altamira" por Anónimogo (14.000 a.c.)

Capítulo 1

Introducción

1.1. Motivación

Habría que remontarse a los reyes de la antigua India y sus galerías personales de arte (*chitrashalas*), o a las clases gobernantes de mediados del siglo XVI a.c. en China que atesoraban colecciones de objetos valiosos, para encontrar los orígenes del concepto de *Museo*. Sin embargo, no es hasta el siglo XVII cuando se utiliza el término *museo* por Elias Ashmole al presentar una colección de objetos curiosos y artefactos en la Universidad de Oxford. Este término proviene del griego *mouseion* (sitio de las musas); en la antigua Grecia, los mouseions eran lugares sagrados dedicados a las nueve diosas de las artes y las ciencias: Calliope era la musa de la poesía épica, Clio de la historia, Euterpe de la poesía lírica, Melpomene de la tragedia, Terpsichore de las canciones corales y la danza, Erato de la poesía romántica, Polyhymnia de la poesía sacra, Urania de la astronomía, y Thalia de la comedia. Estos lugares se convertirían posteriormente en sitios de acumulación de regalos y ofrendas por parte de los devotos, y de ahí su utilización para definir el concepto moderno de museo como repositorio de objetos.

En la actualidad existen diversas definiciones de *museo* según los museólogos o las asociaciones profesionales implicadas en su definición. Aunque las diferentes definiciones varían en la naturaleza de las colecciones que un museo ha de tener, su estatus legislativo y su estructura interna, todas ellas coinciden en las funciones que un museo ha de desempeñar. En palabras de George Brown Goode (Goode, 1895),

> *"Un museo ha de ser una institución dedicada a la preservación de aquellos objetos que mejor ilustren los fenómenos de la naturaleza y los productos del ser humano, y la utilización de los mismos para el crecimiento del saber y para el enriquecimiento cultural de las personas".*

De acuerdo con definiciones más modernas establecidas por el Consejo Internacional de los Museos (ICOM, 2001),

> *"Un museo es una institución permanente sin ánimo de lucro para servir a la sociedad y a su desarrollo y que colecciona, conserva, investiga e interpreta evidencias materiales de las personas y de su entorno con objetivos de estudio, educativos y de disfrute".*

De acuerdo con las definiciones anteriores y posteriores estudios de Beer (Beer, 1990) y Lord & Lord (Lord, 1997), podemos agrupar las principales funciones o roles de un museo en tres categorías genéricas: de conservación, sociales, y profesionales; y es en torno a estas tres funciones que se desarrollan las principales necesidades o usos de la información en el contexto de los museos.

Según un estudio realizado por Melnika (Melnika, 2004), tras un exhaustivo análisis de los principales informes presentados en las conferencias anuales museísticas, se distinguen cuatro usos básicos de las tecnologías de la información (TI) en el contexto de los museos: documentación, visualización, información e interrelación. De acuerdo con este estudio, la documentación engloba todos aquellos usos de las TI referidas a la naturaleza de los datos, los estándares de descripción y de catalogación y la creación de conjuntos de datos más complejos a partir de la información básica. Las TI usadas en la visualización incluyen todos los mecanismos digitales no textuales tales como imágenes, sonidos y videos mediante los que un museo consigue que el visitante forme una imagen integrada de alguno de sus contenidos o mediante los que se realizan las funciones de conservación de la institución. La información se encarga de cubrir la complejidad técnica de la infraestructura software empleada por el museo en todos sus niveles y, finalmente, la interrelación enmarca todos los tipos de información en red tanto si se trata de redes internas de recursos, como de redes globales poco estructuradas para la comunicación entre el museo y los agentes externos (profesionales y público) o como de redes en forma de *clusters* globales formados a partir de consorcios de profesionales o estructuras regionales intra-industriales.

En el presente trabajo abordaremos aspectos que tienen que ver con los cuatro usos básicos de la información en los museos, pero en un escenario que poco tiene que ver con los tradicionales usos de la información en este contexto.

Veremos cómo la incorporación de las nuevas tecnologías en los últimos años ha hecho evolucionar el concepto de museo hasta la situación actual en la que las tecnologías de carácter ubicuo en el ámbito de los museos se traducen en nuevos escenarios que plantean nuevos desafíos. Todos ellos serán el objetivo del presente trabajo.

1.2. Museos e Internet

Los últimos decenios han sido testigos de revoluciones tecnológicas que se han incorporado a todos los aspectos de nuestra vida diaria, incluido nuestro Patrimonio Cultural. Internet está evolucionando a una velocidad que pocos podían predecir. A mediados de los años 90 existían 5 millones de usuarios. En 2000, había 200 millones, y en 2004, en plena explosión de la burbuja tecnológica, Internet era ya un monstruo de unos 804 millones (Global, 2005).

Uno de los precursores en el uso de Internet en el dominio de los museos fue el museo del *Louvre*, el cual en los años incipientes de la Web creó el recurso *on-line* denominado *WebMuseum* (WebMuseum, 1994). Este proyecto inicial fue de los primeros en afrontar el problema de la descarga de imágenes de gran tamaño mediante el uso de portales alternativos (*mirrors*) en diferentes puntos del planeta. Este proyecto fue entonces merecedor del premio *Best of the Web* en 1994 en la categoría de *Best Use of Multiple Media* y todavía es hoy considerado por muchos un recurso de gran calidad. Sin embargo, hay que decir en su contra que únicamente ofrece una navegación de sus contenidos por índice temático y por artistas, pero no ofrece ningún tipo de mecanismo de interrelación entre artistas, movimientos artísticos ni de visitas guiadas según diferentes criterios. Afortunadamente, el museo del *Louvre* ofrece en la actualidad un recurso (Louvre, 2004) de mayor calidad, multilingüe, y que ofrece un conjunto de galerías visibles con giros de 360 grados, que permiten una percepción cuasi real de las mismas. Adicionalmente, se ofrecen servicios de venta de entradas *on-line*, un recurso de carácter educativo (LouvreEdu, 2004) y hasta una tienda on-line accesibles desde la página principal. También en París, son de especial mención el *Musée d'Orsay* (Orsay, 2003) que ofrece extensos servicios de compra on-line y el portal del *Centre Georges Pompidou* (Pompidou, 2003) característico por sus servicios multimedia entre los que destacan los videos sobre conferencias, retratos de artistas y presentaciones de exposiciones.

Continuando en el ámbito europeo, y centrándonos en Reino Unido, destaca la *National Gallery* en Londres (National, 2003) que ofrece una impresionante colección de alrededor de 2.500 obras de arte de calidad aceptable y con mecanismos de búsqueda por nombre de artista, nombre de la obra y número

de galería. Esta colección es uno de los recursos educativos de arte más vastos, aunque todo el contenido tanto textual como gráfico del portal está protegido por derechos de *copyright*. También en esta ciudad son destacables la *National Portrait Gallery* (Portrait, 2002) y la *Tate Gallery* (Tate, 2004). La primera ofrece servicios de búsqueda más avanzados que la *National Gallery*, pero las imágenes son de menor resolución. A pesar de ello, es posible comprar versiones impresas de gran calidad de dichas obras. La segunda, un consorcio de galerías de arte en el que su máximo exponente es el *Tate Modern*, ofrece una página Web inicial dinámica que cambia de apariencia en cada visita, dando al usuario la sensación de tener experiencias nuevas en sucesivas visitas. Las colecciones catalogadas *on-line* son de gran tamaño y se ofrecen buenos servicios de búsqueda. Sin embargo, gran cantidad de obras no se incluyen en formato gráfico debido a restricciones impuestas por los derechos de propiedad intelectual.

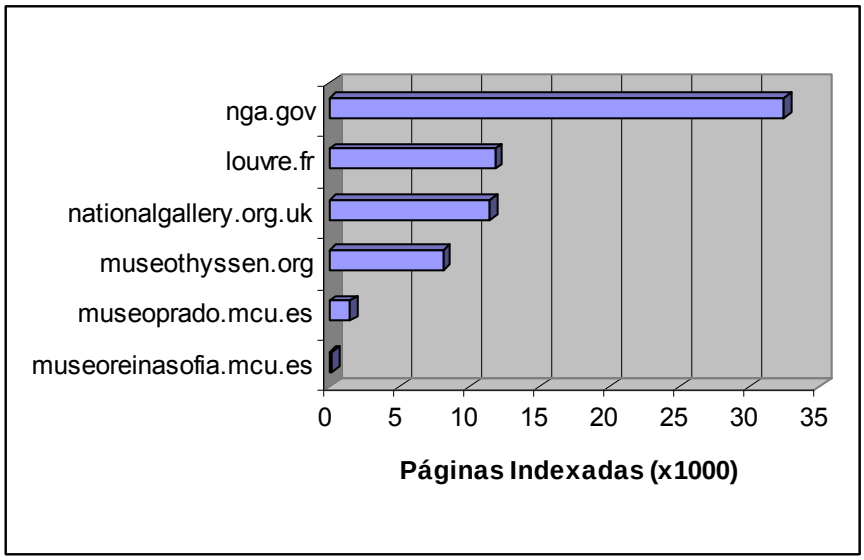


Figura 1. Páginas indexadas por Google

En el ámbito de nuestro país destacan, por orden de importancia en la Web, los museos *Thyssen* (Thyssen, 2003), *El Prado* (Prado, 2004) y el centro de arte *Reina Sofía* (Reina, 2004). El *Thyssen*, además de ofrecer vistas panorámicas de sus salas se caracteriza por ofrecer una amplia oferta educativa bajo la imagen de marca *EducaThyssen* en la que los participantes en programas presenciales o los internautas pueden crear nuevos contenidos que se incorporen a la Web. Para conseguir este objetivo, se ha creado un sistema de comunidades donde los usuarios pueden trabajar con herramientas de generación de contenidos relacionados con las actividades o la colección del propio museo. La oferta *on-line* de este museo, comparada con otros en el contexto internacional, es similar

a los museos europeos en cuanto al número de páginas indexadas por el conocido buscador *Google* (Figura 1), pero presenta un gran déficit con respecto a ellos, y más aún si es comparado con los museos americanos, en cuanto a nivel de popularidad de los enlaces en *Google* y en *MSNSearch* (Figura 2). El Prado, además de no tener aplicaciones educativas interactivas como parte de su oferta *on-line*, ofrece únicamente un paseo por las 50 obras esenciales del museo y carece de mecanismos básicos de búsqueda por título de obra o por autor. Finalmente, el *Reina Sofía* ofrece un portal Web muy moderno en cuanto a su diseño, con posibilidad de búsquedas, pero con sólo siete visitas virtuales y una limitada oferta educativa.

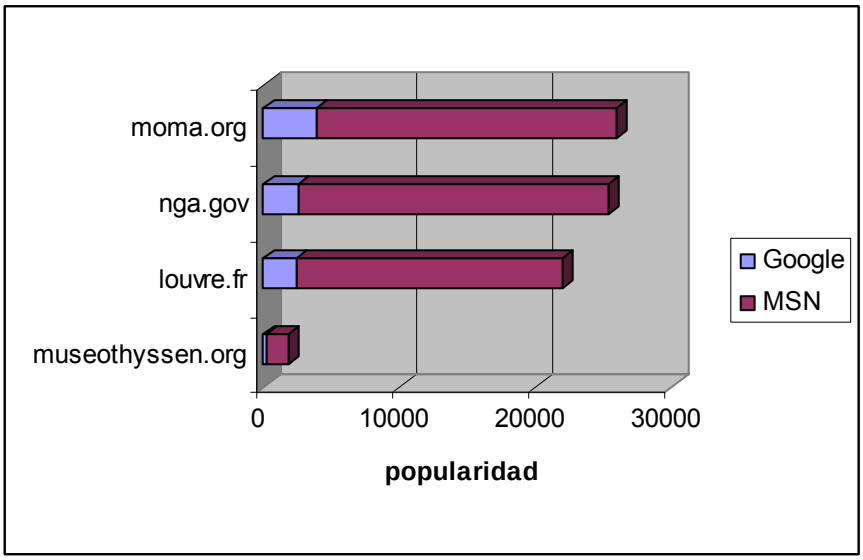


Figura 2. Popularidad de Enlaces en Google y MSNSearch

En el ámbito norteamericano, destaca en Nueva York el *Museum of Modern Art* (Moma, 2004) con una Web con elaborados mecanismos de búsqueda y con programas educativos para niños como el *Destination: Modern Art* y *Red Studio.* En el *San Francisco Museum of Modern Art* (SFMoma, 2004), con un enorme despliegue multimedia como parte de su programa *Making Sense of Modern Art* (MSoMA), se permite gracias a los formatos de imagen empleados hacer *zooms* detallados de las obras de arte, explorar la colección de videos y películas y escuchar los comentarios realizados por artistas, historiadores del arte, críticos y coleccionistas. La *National Gallery of Art* (Nga, 2004) en Washington presenta uno de los portales más pródigos, siendo el que más páginas indexadas tiene en *Google* (ver Figura 1), y cuyos enlaces están entre los más populares (ver Figura 2). Por último, el Art Museum Network (Amn, 2004) también merece ser mencionado porque representa el esfuerzo coordinado de diversas organizaciones museísticas de importancia entre las que destaca el Art Museum

Image Consortium (Amico, 2002), con uno de los mejores catálogos *on-line* consistente en más de 50.000 obras de arte disponibles.

1.3. Planteamiento del Problema: Museos Híbridos

El lector interesado podrá encontrar en (Carreras, 2005) un estudio detallado de los museos *on-line* enumerados anteriormente, junto a otros muchos más, analizados desde diferentes perspectivas como las bases de datos empleadas, los mecanismos de propiedad intelectual, los portales de comercio electrónico y las formas de accesibilidad a los mismos. Sin embargo, desde el punto de vista de este trabajo y de la problemática a abordar, nos interesa conocer cuál es el grado de uso de los museos en Internet más representativos entre los descritos en esta sección con vistas a justificar la necesidad de nuestro planteamiento investigador.

La evaluación de los portales Web de museos ha de ser, según diversos autores (Carreras, 2005), multimodal, esto es, ha de ser una combinación de métodos cualitativos y cuantitativos que de forma conjunta permitan ganar un conocimiento en profundidad sobre los patrones de uso y la efectividad de este tipo de portales. Entre los métodos recomendados se encuentran: el análisis de los *WebLogs,* que proporciona una información cuantitativa valiosa sobre el tipo de contenido mas visitado, los niveles de interacción con el mismo y los perfiles de los usuarios; los cuestionarios y libros de visitantes que recogen la opinión de los mismos, sensaciones, sugerencias y miden su grado de satisfacción; las observaciones in situ de usuarios utilizando el portal Web para evaluar su estructura y diseño; y los estudios estadísticos de visitantes o de población para averiguar el grado de conocimiento, las actitudes, las necesidades y la relación entre las visitas físicas y virtuales. Entre los estudios realizados de acuerdo con los métodos descritos anteriormente destaca el de Loran (Carreras, 2005) en el que se realizó una evaluación del portal del grupo de museos *Tate* sobre un total de 10 secciones con 2.500 páginas de texto y con información sobre 25.000 obras de arte. Este estudio revela que el portal más visitado del consorcio de galerías es el *Tate Modern* con un 29% de los accesos y que, de entre las tres secciones principales (exposiciones temporales, información para la visita, y programas públicos), el 52% de los accesos es a la información general para la visita. Dicho de otro modo, la mayor parte de los accesos son realizados por personas que visitan por primera vez este museo y van en busca de información previa a la visita in situ. También este estudio muestra que, en menor medida, se dan accesos a las secciones de exposiciones temporales por parte de visitantes más habituales al museo. De estos resultados concretos para el *Tate* se deduce que, en este caso particular, el uso de portales Web favorece la diseminación de información que tiene que ver con las futuras

visitas, pero se hace un menor uso de las visitas virtuales con una estructura predeterminada que se ofrecen en los mismos. Sin embargo, aún más interesante es observar el patrón de accesos a la sección *Tate Collection* que permite el acceso a las colecciones a través de una base de datos. Analizando dicho patrón se observa que el 89% de los accesos a dicha colección son resultados de búsquedas y no de seguimiento de visitas preestablecidas. Un 33% de éstas son listas de artistas/obras, 54% son páginas individuales de obras de arte y un 13% son textos interpretativos. De estos datos se deduce que los usuarios muestran un interés mayoritario por obras de arte específicas y no tanto por temáticas artísticas.

Para saber si estos resultados son particulares al consorcio *Tate* o si por el contrario son generalizables, es necesario prestar atención a otros estudios. En particular, los resultados obtenidos por Kravchyna y Hastings (Kravchyna, 2002) a partir de una encuesta internacional realizada en la Web a personal de museos, visitantes, estudiantes y profesores, son consistentes con los datos observados en el consorcio *Tate*. Además, en este estudio en particular se describen las categorías de información más asiduamente buscada en los portales Web de los museos, y se concluye que el 68% de los usuarios busca información sobre exposiciones en curso y que un 64% valora la existencia de información descriptiva suplementaria sobre obras de arte particulares. Parece, pues haber una necesidad de información contextual con narrativas descriptivas que aporten un conocimiento adicional sobre obras de arte específicas.

En el caso de los museos españoles, la situación no es ni mucho menos optimista. La fundación AUNA, en su *Informe Anual sobre el Desarrollo de la Sociedad de la Información en España* del año 2002 (Ballestero, 2002) presenta un estudio detallado en el que se cuantifica el grado de presencia, desarrollo y uso de las tecnologías de la información en los museos españoles en comparación con los museos de mayor relevancia internacional. Dicho estudio analiza desde distintas perspectivas 53 museos de ciencia y arte, y concluye que el nivel de desarrollo global en Internet de nuestros museos es alto únicamente en el 4,5% de los museos analizados, si bien éstos consiguen un grado de fidelización medio-alto, a partir de servicios de acceso a los catálogos digitales, en el 80% de los casos.

En definitiva, de todo lo expuesto anteriormente podemos concluir que en ningún caso los portales Web sustituyen a la riqueza sensorial de las visitas in situ, que los portales Web se utilizan en gran medida como herramientas para la preparación de visitas futuras o como puntos de búsqueda de información contextual adicional sobre obras ya visitadas y que, en cualquier caso, existe una separación espacio-temporal entre la propia visita al museo y la consulta de

los portales Web asociados a los mismos, lo cual dificulta los procesos de aprendizaje que tienen lugar de forma presencial en el propio museo.

1.3.1. Medios y Sentidos: Hacia los Museos del Futuro

En la actualidad, el acceso a la red está pasando de los ordenadores de sobremesa a cualquier tipo de dispositivo móvil, como teléfonos y agendas digitales, y es previsible que, en torno a 2015, exista una convergencia completa entre los sistemas de telefonía, Internet y televisión en lo que se ha empezado a llamar las Tecnologías de la Convergencia Universal (TCU) (Veltman, 2003). Cada medio, como ya propuso McLuhan, usará el medio anterior como su contenido. Los medios digitales y las TCU crearán, en palabras de Veltman

> *"puentes que permiten el intercambio entre medios y sentidos. Esto creará un puente digital, por el que incluso las personas no letradas podrán beneficiarse de los nuevos medios, lo cual merece incluso más atención que la brecha digital. En el dominio cultural, las áreas claves de desarrollo incluyen problemas de depósito, el alcance cambiante del Patrimonio Cultural, nuevos enlaces a nivel nacional, regional y local; entre cultura, conocimiento e investigación académica; aproximaciones europeas a la propiedad intelectual, y necesidad de nuevos modelos de cultura".*

Parece, pues, evidente a la vista de los últimos avances tecnológicos, que no se están estableciendo los puentes digitales adecuados para conseguir una fusión de calidad entre los medios actuales y los sentidos de los visitantes a un museo. Por ello, una infraestructura tecnológica más adecuada que las analizadas en la sección anterior debería tener las siguientes propiedades:

- Ser compatible con la riqueza sensorial de las visitas físicas.
- Enriquecer las visitas con información de carácter virtual para facilitar la comprensión de las obras de arte expuestas.
- No ser un mecanismo estático de producción de información sino tener un carácter dinámico que permita variar los contenidos.
- Ofrecer bajo un único punto de entrada diferentes servicios.
- Proveer información al visitante en cualquier momento durante la realización de una visita.

Es por esta razón que este trabajo presta especial atención a los "puentes digitales" de Veltman que se establecen entre *medios* y *sentidos*, para facilitar los requisitos enumerados con anterioridad. Particularmente, esta tesis analiza los nuevos problemas y escenarios de uso que se plantean en el contexto de lo que llamaremos *Museos Híbridos*, esto es, cuando se establecen puentes digitales

entre los medios digitales de naturaleza móvil y la riqueza sensorial que existe en el disfrute del Patrimonio Cultural de forma presencial. Veremos cómo los usos de las TI planteados por Melnika (documentación, visualización, información e interrelación) plantean nuevos retos cuando consideramos medios de naturaleza ubicua que tendrán la gran ventaja de hacer compatible la riqueza sensorial de las visitas físicas a los museos con la riqueza digital de las infraestructuras de información asociadas.

En esta tesis plantearemos nuevos modelos asociados a la *documentación* tanto básica como compleja del patrimonio cultural para su disfrute en el nuevo concepto de Museo Híbrido; definiremos nuevos mecanismos de *interrelación* que permitan integrar repositorios distribuidos, daremos soporte a todos los medios de *visualización* y plantearemos una infraestructura de *información* que permita las visitas a los museos híbridos dando soluciones algorítmicas a problemas emergentes como el de la orientación.

1.4. Metodología de trabajo

La presente tesis se enmarca dentro de las actividades del grupo de investigación Ingeniería del Software y Sistemas de Información (ISSI) del Departamento de Sistemas Informáticos y Computación. Los inicios de la presente tesis se remontan al año 2001 en el que se planteó el análisis de sistemas de información de naturaleza ubicua. Como consecuencia de la revisión de los sistemas existentes, se detectó la incipiente aplicación de dichos sistemas al dominio del patrimonio cultural y se observaron serias carencias que motivaron la definición del proyecto investigador. La experiencia previa del autor de esta tesis en el campo de los sistemas distribuidos en forma de *Grid* contribuyó al enfoque del problema desde una perspectiva de sistema heterogéneo y no centralizado. Fruto del análisis del problema, se presentó un proyecto de investigación (MoMo) que fue aceptado y financiado por los laboratorios de investigación *Microsoft Research (Cambridge)*.

Durante el desarrollo del proyecto se ha llevado a cabo una metodología experimental activa consistente en la definición de modelos y algoritmos, su implementación, el análisis de sus cualidades y la posterior optimización de los mismos a la vista de los resultados inicialmente obtenidos.

Se ha seguido un desarrollo incremental, de manera que la base teórica y sus correspondientes implementaciones se han ido enriqueciendo a medida que se identificaban nuevos escenarios de uso en los Museos Híbridos.

El estudio de los problemas y requisitos emergentes en este contexto ha permitido, por ejemplo, descubrir relaciones entre la cultura digital y la

algorítmica evolutiva que inicialmente eran insospechadas. Estas relaciones que inicialmente eran de tipo cualitativo han podido ser validadas mediante diferentes iteraciones en el desarrollo de esta tesis.

1.5. Resumen de las Aportaciones

Las principales aportaciones de esta tesis abarcan todas las dimensiones descritas por Melnika particularizadas al desarrollo de una plataforma de Museo Híbrido. Enumeramos a continuación las más relevantes:

1. Describe el estado del arte en la aplicación de las tecnologías móviles al dominio del Patrimonio Cultural (véase capitulo 2).
2. Define un nuevo modelo de Museo Híbrido y su correspondencia con el marco ontológico Conceptual referente Model (CRM) propuesto por el *Internacional Council of Museums* (ICOM) (véase capítulo 3).
3. Define una infraestructura de federación semántica para la interoperabilidad a bajo nivel entre repositorios semánticos definidos según el entorno *Resource Description Framework* (RDF) y evalúa sus prestaciones (véase capítulo 4).
4. Define el problema de la orientación en el contexto de los Museos Híbridos y plantea su resolución mediante el uso de algoritmos evolutivos basados en colonias de hormigas (véase capítulo 5).
5. Define un algoritmo distribuido basado en Grids computacionales para la resolución aproximada del problema de la orientación en tiempo polinomial y se muestra empíricamente sus buenas prestaciones (véase capítulo 5).
6. Define la resolución del problema de la orientación aplicando dos técnicas diferenciadas de aceleración gráfica: una basada en el procesamiento de vértices y otra en el de fragmentos mediante el uso de un procesador gráfico de última generación y se muestra empíricamente sus buenas propiedades (véase capítulo 6)

1.6. Estructura de la Tesis

A continuación pasamos a describir de forma abreviada el contenido de cada capítulo y las convenciones tipográficas empleadas

1.6.1. Organización

- El presente capítulo introduce la motivación, describe el problema y establece las convenciones tipográficas utilizadas en esta tesis.

- El capítulo 2 describe el estado del arte en el ámbito de los museos híbridos y los trabajos previos relevantes con el ámbito genérico del problema.
- El capítulo 3 describe los modelos existentes de Museo Híbrido, plantea las funciones de aprendizaje que un museo ha de cumplir y, en base a las mismas, propone un modelo de Museo Híbrido que permite los procesos de aprendizaje mediante la compatibilidad entre los contenidos digitales y los reales. En este capítulo también se define la semántica del modelo propuesto en función de los estándares ontológicos existentes en el dominio de aplicación
- El capítulo 4 plantea la problemática de la integración a bajo nivel de los repositorios semánticos de naturaleza distribuida. Se propone una infraestructura de naturaleza federada para la integración de repositorios semánticos y se muestra su aplicación al dominio de los museos híbridos
- El capítulo 5 define el problema de la orientación y presenta un algoritmo distribuido en forma de Grid que, utilizando técnicas de descomposición del problema en subproblemas, consigue resolver el problema en tiempo polinomial.
- El capítulo 6 ofrece una perspectiva de solución alternativa al problema de la orientación basándose en el uso de técnicas de aceleración gráfica. En el mismo se define la estructura de un procesador gráfico, su programabilidad y se proponen dos algoritmos que utilizan la programación a nivel de vértices y de fragmentos para resolver de forma altamente eficiente, a la vista de los resultados experimentales, el problema.
- Finalmente, el capítulo 7 resume las conclusiones y las aportaciones del presente trabajo
- Los apéndices proporcionan información adicional sobre el proceso de análisis en forma de casos de uso de los escenarios soportados en un Museo Híbrido. También presentamos con más detalle la formulación semántica del modelo de Museo Híbrido formulado en el capítulo 3 y, finalmente, se detallan los algoritmos implementados en los procesadores gráficos de última generación.
- Este trabajo se cierra con la bibliografía y los índices de figuras y tablas.

1.6.2. Convenciones tipográficas

Las convenciones tipográficas empleadas en este documento son las habituales en los textos científicos y están encaminadas a la mejora de la lectura de esta tesis. Dichas convenciones vienen enumeradas a continuación con un ejemplo

- Los términos provenientes de otros idiomas pero que se utilizan con asiduidad en el lenguaje científico informático aparecen resaltados en cursiva, por ejemplo, el cluster formado.
- Las citas procedentes de terceras fuentes aparecen en cursiva y convenientemente delimitadas, por ejemplo, "operibus credite et non verbis".
- El código fuente proveniente de un lenguaje de especificación en pseudo-código o de un lenguale de programación aparece en bloques sombreados con tipografía no proporcional.
- Las referencias bibliográficas aparecen entre paréntesis siguiendo el convenio estándar indicando un acrónimo y un año de producción. En caso de tener dicho autor varias publicaciones en ese mismo año utilizamos letras del alfabeto para distinguir cada una de ellas, por ejemplo, (Jaén, 2004). La información completa de las mismas se encuentra ordenada alfabéticamente en la sección de Bibliografía.

- Las notas al pie[1] aparecen etiquetadas con un subíndice numérico (único por capítulo) que corresponde con la nota etiquetada con el mismo número en el pie de la página.

[1] Un ejemplo de nota al pie

Arte Egipcio. Escena Funeraria

Capítulo 2

Situación Actual: Museos Híbridos

El presente capítulo realiza una revisión de los trabajos relacionados con la aplicación de las tecnologías móviles de la información en el contexto del patrimonio cultural. Esta revisión del estado del arte es de carácter genérico e irá sucedida en los capítulos posteriores de revisiones del arte específicas a la problemática tratada en los mismos.

2.1. Tecnologías para Museos Híbridos

Tradicionalmente, el acceso a las exposiciones por parte de visitantes y su correspondiente experiencia in situ se ha facilitado mediante el uso de información adicional como etiquetas que aparecen al lado de cada elemento museístico, documentos impresos o, en el mejor de los casos, mediante kioscos multimedia que ofrecen información de todo tipo al visitante. En todos estos casos, la información se encuentra disponible en puntos fijos del museo a los que han de dirigirse los visitantes. Sin embargo, los últimos desarrollos tecnológicos referidos a la movilidad permiten el acceso a la información en cualquier lugar y momento en el que un visitante se encuentre, de manera que éste puede obtener la información que necesite bajo demanda. Por otro lado, motivado por esta ubicuidad en el acceso a la información se ha hecho cada vez más necesario poder ubicar al usuario de manera que se le pueda ofrecer información contextualizada a su posición. Asimismo, la necesidad de movilidad ha provocado el diseño de dispositivos de visualización de tamaño reducido que puedan ser transportados con facilidad.

Haremos, pues, un repaso en esta sección a las tecnologías más relevantes en la actualidad que posibilitan tanto el acceso ubicuo a la información mediante dispositivos móviles como la localización de los usuarios allí donde se encuentren, para más adelante analizar los distintos escenarios en los que se hace uso de ellas en el dominio de los museos híbridos.

2.1.1. Redes Inalámbricas

Los diferentes estándares de redes inalámbricas se basan en especificaciones que definen el mecanismo tanto radioeléctrico como de protocolo de acceso al medio inalámbrico. Entre las especificaciones más importantes podemos destacar: *Bluetooth* (Bluetooth, 2005), *Wireles Fidelity* (WiFi) en sus diferentes versiones (WiFi, 2005), Global System for Mobile Communication (GSM) y su variante orientada a datos GPRS (GSM, 2005), y WiMax (WiMax, 2005).

Bluetooth es el nombre de un estándar propuesto por un consorcio de más de 1000 empresas tecnológicas que permite que distintos tipos de dispositivos puedan conectarse de forma inalámbrica mediante canales que soportan velocidades máximas de 720Kbps. En su especificación física este estándar utiliza el rango de frecuencias alrededor de los 2.45Ghz, que está reservado a nivel internacional para aplicaciones de carácter científico e industrial. Dado que este rango de frecuencias es también utilizado por otros dispositivos como teléfonos sin cables, mecanismos de apertura de puertas a distancia, y microondas, entre otros, Bluetooth evita interferir con ellos mediante el empleo de señales de baja potencia, -lo que limita su uso a un máximo de 10 metros-, y también mediante una técnica denominada *spread-spectrum frequency hopping* que permite cambiar a una velocidad de 1600 veces por segundo la frecuencia concreta en la que dos dispositivos acceden al medio en un instante determinado. Debido al limitado alcance de este estándar, se hace uso del mismo en escenarios de interior para transferir información almacenada en los propios objetos de la exposición al dispositivo receptor del visitante si éste se encuentra a una distancia menor que la especificada por el estándar. La limitada ubicuidad en el acceso a la información ha hecho que Bluetooth no haya sido ampliamente utilizado en escenarios de acceso a información relacionada con el patrimonio cultural.

WiFi es un conjunto de estándares propuesto por la *WiFi Alliance* basados en la especificación IEEE802.11 (IEEE802.11, 2005). Al igual que el anterior, WiFi utiliza el rango de frecuencias de 2.4Ghz con velocidades teóricas de acceso al medio de 11, 54 y 108Mbps según la variante utilizada (802.11b, 802.11g y 802.11n respectivamente). Sin embargo, a diferencia de Bluetooth, WiFi permite el enlace a distancias entre 30 y 100 metros en interiores y de hasta 150 metros en condiciones ideales en exteriores, por lo que se ha convertido en el estándar de facto para las comunicaciones inalámbricas más extendido a nivel

internacional. Se estima que en la actualidad existen alrededor de 50.000 puntos de acceso (*hotspots*) que dan servicio a un número aproximado de 30 millones de usuarios (DataQuest, 2005) y que en 2008 se alcanzará la cifra de 200.000 *hotspots*, lo que da una idea de la relevancia de este estándar. A ello hay que añadir que cada vez más se incrementa el número de *hotspots* que dan acceso gratuito a la red, lo cual está contribuyendo a su popularidad. WiFi es, por tanto, una tecnología clave en el acceso ubicuo a la información en el contexto de los museos híbridos, y no tanto en otros escenarios de visita al patrimonio cultural en espacios abiertos en los que el area de movilidad del usuario se incrementa.

La movilidad en espacios abiertos necesita de tecnologías de gran cobertura y que gestionen de forma transparente el paso de unas regiones a otras sin pérdida de conectividad. Normalmente las tecnologías móviles con estos requisitos han sido utilizadas para proporcionar comunicaciones de telefonía móvil. Para conseguirlo normalmente se divide el espacio en regiones (celdas) de aproximadamente 26 kilometros cuadrados. Cada célula tiene una estación base consistente en una antena y en un rango de frecuencias único de manera que se eviten colisiones en el acceso al medio entre celdas adyacentes. El sistema de mayor éxito en telefonía móvil ha sido GSM, que utiliza mecanismos de acceso al medio de division en el tiempo (*Time Division Multiple Access*). GSM ha estado tradicionalmente orientado a las comunicaciones de voz, pero existen variantes como GPRS y una posterior evolución, UMTS, basada en *Wide Code Division Multiple Access,* que permite el envío y recepción de datos incluso de naturaleza multimedia con anchos de banda que pueden llegar hasta los 2Mbps. Dadas las características de estas tecnologías, es comprensible que hayan sido muy utilizadas en escenarios móviles de visitas a patrimonio cultural en espacios abiertos, como por ejemplo, visitas a monumentos en una ciudad.

Finalmente, una de las tecnologías más prometedoras para el futuro es sin duda WiMax (del inglés *Worldwide Interoperability for Microwave Access*). WiMax es el resultado de los esfuerzos de diseño y estandarización del *WiMax Forum*, compuesto en la actualidad por más de 100 empresas. Esta tecnología, basada en el protocolo de transmisión inalámbrica IEEE 802.16, utiliza la banda de frecuencias 2-11Ghz y utiliza *Ortogonal Frequency Division Multiplexing* como técnica de modulación que permite la cobertura de areas de 48 kilómetros desde la estación base y con capacidad de transferencia de datos de hasta 75Mbps. Su interés radica en que combina su amplia cobertura y gran velocidad de transferencia con costes competitivos si se le compara con otras redes como WiFi. Además, recientemente se ha aprobado el estándar WiMAX MÓVIL (802.16e) que permite utilizar este sistema de comunicaciones inalámbricas con terminales en movimiento.

2.1.2. Tecnologías de Localización en el Espacio

Las tecnologías de ubicación en el espacio pueden analizarse desde diferentes perpectivas, como su precisión, el ámbito de aplicación y el coste de implantación. Las tecnologías más relevantes en este sentido son WiFi, *Ultra Wide Band* (UWB) (UWB, 2005), *Global Positioning System* (GPS) (GPS, 2005), y *Radio Frequency Identification* (RFID) (RFID, 2005).

Las técnicas de posicionamiento basadas en puntos de acceso WiFi tienen la gran ventaja de aprovechar infraestructuras de red ya existentes. Estos sistemas de posicionamiento se basan en la comparación de la intensidad de las señales de varios puntos de acceso, medidas por un receptor WiFi que se desea ubicar, con las intensidades medidas en posiciones conocidas durante un proceso de calibración previo. Esta técnica permite alcanzar medidas de la posición en interiores con una precisión de hasta un metro pero no puede usarse en espacios abiertos sin cobertura WiFi. Uno de los productos comerciales de mayor precisión es *Ekahau Positioning Engine* (Ekahau, 2005), aunque su coste de implantación es elevado si se compara con otros mecanismos de posicionamiento como GPS.

En caso de necesitar una mayor precisión, hemos de considerar los sistemas de posicionamiento basados en UWB que permiten ubicar objetos en el espacio con una precisión de hasta 20 centímetros. UWB transmite señales de radio de muy baja potencia por medio de pulsos electrónicos de corta duración (un picosegundo) en todas las frecuencias simultáneamente. Los receptores UWB han de traducir estos cortos mensajes, que tienen apariencia de ruido en datos, mediante la identificación de secuencias de pulsos familiares. Dado que UWB requiere muy poca energía, las señales son difíciles de detectar y por tanto de regular. Sin embargo, dada la naturaleza de estas señales, tiene la ventaja de poder ser utilizada tanto en interiores como en presencia de un gran número de obstáculos sin una enorme pérdida de precisión como le ocurre a GPS y a WiFi. Por otro lado, UWB también tiene la ventaja de que permite transferir datos a velocidades de entre 40 y 60 Mbps, pudiendo llegar en el futuro hasta 1Gbps. El mayor exponente de esta tecnología para su uso en localización es la empresa Ubisense (Ubisense, 2005) que ha desarrollado un producto de enormes prestaciones pero cuyo coste es en la actualidad prohibitivo.

GPS es quizás la más familiar de las tecnologías que hemos presentado hasta ahora. Se trata de una tecnología desarrollada en sus orígenes con objetivos militares para el seguimiento de tropas y el apoyo a la navegación, pero en los años 80 se puso a disposición del sector público en areas como la navegación aérea, marítima, y terrestre. En todos estos casos, GPS se ha utilizado tradicionalmente para el posicionamiento de vehículos de transporte, pero más recientemente los sistemas GPS están siendo utilizados para el

posicionamiento de personas en movimiento, de forma que se puedan ofrecer servicios localizados. Esta tecnología determina la posición global de los objetos calculando la distancia entre un objeto y al menos tres satélites y aplicando posteriormente nociones de trigonometría. Para ello, los receptores de GPS utilizan señales de alta frecuencia emitidas por una red mundial de 27 satélites. En el contexto de los museos híbridos, esta tecnología no sólo se utiliza para localizar la posición de objetos estáticos, sino también para detectar los caminos seguidos por los usuarios y extraer de esta forma información sobre sus posibles gustos. El gran desarrollo que esta tecnología ha experimentado en los últimos años ha abaratado enormenente los costes de adquisición de receptores GPS. Sin embargo, las características físicas de las señales emitidas por los satélites GPS hacen que esta tecnología pueda ser únicamente utilizada en el exterior de edificios con una adecuada precisión, que nunca es menor, en aplicaciones civiles, a una decena de metros.

Por último, hay que destacar la irrupción de las etiquetas RFIDs que contienen antenas que les permiten responder a peticiones por radiofrecuencia desde un emisor-receptor RFID. Existen dos tipos de etiquetas: las pasivas, que no necesitan alimentación eléctrica interna, y las activas, que sí la requieren. Las etiquetas pasivas utilizan la energía inducida en la antena por la señal de escaneo de radiofrecuencia para poder transmitir una respuesta que es normalmente un número de identificación. Las etiquetas pasivas tienen distancias de lectura que oscilan entre unos 10 milímetros hasta cerca de 6 metros, dependiendo del tamaño de la antena, de la etiqueta, y de la potencia y frecuencia en la que opera el lector. Por el contrario, las etiquetas activas son utilizadas en posicionamiento de objetos dado que permiten un mayor rango de alcance, aunque este se limita a unas pocas decenas de metros. En el contexto de los museos, las etiquetas RFIDs pasivas son utilizadas para identificar la obra de arte ante la que se encuentra el visitante y proveerle sin prácticamente ninguna intervención los contenidos asociados a la misma. La gran ventaja de esta tecnología es su escaso coste, apenas 0,30€ por etiqueta, lo que permite su fácil implantación en cualquier museo.

2.1.3. Dispositivos Móviles

En el pasado ha existido una clara separación entre aquellos dispositivos móviles cuyo objetivo principal era facilitar la comunicación mediante voz, como los teléfonos móviles, y los dispositivos cuya función principal era la gestión de datos personales (contactos, citas, tareas) en escenarios de movilidad, como son las agendas personales digitales (PDA). Esto ha sido así no sólo por el tipo de funcionalidad que estos dispositivos ofrecían, sino también por el mecanismo de acceso al medio inalámbrico, GSM para los primeros y WiFi o Bluetooth para los segundos. Sin embargo, la necesidad de

una completa ubicuidad en el acceso a la información y en la comunicación con otros usuarios está provocando la convergencia de ambos dispositivos en lo que denominaremos genéricamente dispositivos de mano (*handheld devices*). Los dispositivos de mano tienen diferentes tamaños según el uso principal que se les de a los mismos. Así, encontramos en la actualidad a los dispositivos denominados *smartphones* que añaden a los tradicionales teléfonos móviles un gran número de capacidades de gestión de datos como ocurría en las PDA; y las PDA extendidas con capacidad de conexión a redes GSM para la realización de llamadas teléfonicas.

En la actualidad estamos, pues, asistiendo a la convergencia de las redes de acceso a la información (GSM, WiFi y Bluetooth) y de posicionamiento (principalmente GPS) en dispositivos individuales. Además, los avances tecnológicos permiten que estos dispositivos de reducido tamaño puedan reproducir múltiples formatos multimedia, de manera que es posible escuchar música, ver videos e incluso sintonizar cadenas de televisión digital. A nivel de mecanismos de interacción, los dispositivos de tipo smartphone basan su interacción en el uso del teclado y en algunos casos un *joystick*, mientras que los dispositivos PDA lo hacen mediante una pantalla táctil de mayor tamaño que el usuario presiona mediante un lápiz.

Entre las plataformas dominantes en el ámbito de los *smartphones* se encuentra *Symbian* -apoyada por los principales proveedores de telefonía movil- y *Windows Mobile*, de la compañía *Microsoft*. Por su parte, en el ámbito de las PDA predominan la plataforma *Palm OS* de la compañía *3Com* y, de nuevo, la plataforma *Windows Mobile* bajo la imagen de marca *PocketPC*.

En el contexto que nos ocupa, y dadas las características descritas, los dispositivos smartphone son utilizados con mayor asiduidad en museos híbridos en espacios abiertos, mientras que las PDA son el elemento fundamental para el acceso a la información en museos híbridos en el interior de edificios.

2.2. Proyectos de Museos Híbridos

2.2.1. Urban Tapestries

El proyecto *Urban Tapestries* fué concebido por Prosbotics en colaboración con los laboratorios de *Hewlett Packard* en Bristol, la *London School of Economics* y la compañía de telefonía *Orange* en el año 2003. Este proyecto permitía a los usuarios explorar un espacio abierto y anotar comentarios sobre los lugares recorridos. Las anotaciones podían ser de diverso tipo, de forma que la unión de todas ellas constituyera la memoria colectiva de una comunidad sobre un espacio común. El objetivo inicial del proyecto fue poner a prueba tecnologías

de localización (GPS) y acceso inalámbrico a la información (WiFi y GPRS). Los dispositivos empleados fueron PDA de la plataforma PocketPC y *smartphones* Symbian. Los principales problemas técnicos que tuvo el proyecto fueron la imposibilidad de cambiar de forma transparente entre redes inalámbricas de distinta naturaleza y la falta de precisión del sistema GPS, que en esos años era del orden de decenas de metros incluso en exteriores. En palabras del propio autor Giles Lane,

> *¨el proyecto define un equilibrio entre dispositivos móviles, conocimiento y emociones. El prototipo tiene como objetivo crear una oportunidad para que los usuarios puedan incluir lo que ellos consideran como ´conocimiento´ en el seno del entorno urbano mediante tecnologías inalámbricas. Les permite el acceso a contenidos e impresiones tanto propios como de otros mediante un conjunto de técnicas de filtrado y de toma de decisiones que se pueden programar para reflejar su estado emocional, un proceso conocido como ´filtrado emocional´. Cada Urban Tapestry ha de tener su propio estilo, ya sea e un ámbito urbano entorno a un área local o en el seno de un museo para que los visitantes puedan intercambiar reflexiones y comentarios sobre las exposiciones¨(Giles, 2003).*

En comparación con nuestra propuesta, lo interesante de este proyecto es su capacidad para favorecer la interacción social entre los usuarios de una comunidad, y que nosotros también abordaremos -en nuestro caso, en un Museo Híbrido- no sólo de manera *off-line* sino también en tiempo real durante la visita. Entre las carencias que presenta este proyecto destacamos la ausencia de mecanismos que permitan guíar a futuros visitantes en función de los comentarios positivos o negativos que se realicen sobre los elementos urbanos o museísticos de interés. Nosotros abordaremos este problema desde una perspectiva de algoritmos evolutivos resolviendo el problema de la orientación.

2.2.2. MUSEpad

El proyecto MUSEpad, desarrollado en la universidad de Indiana para su museo *Mathers Museum of World Culture*, es un claro ejemplo de uso de las tecnologías para mejorar el disfrute de la cultura en personas con discapacidades físicas, en este caso personas ciegas, sordas y con problemas de movilidad. Uno de los aspectos más innovadores de este proyecto fue la decisión de incorporar al proyecto mecanismos de personalización de la información teniendo en cuenta el contexto personal, social, físico y mental de cada visitante individual. Mediante MUSEpad los usuarios podían seleccionar el formato de los contenidos (texto, audio, imágenes, gráficos) y las herramientas, entre las que destacaban juegos multimedia, para facilitar la interacción con los recursos museísticos. La información podia proporcionarse

de una forma más elaborada si estaba dirigida a adultos o en forma de cuentos en el caso de que se tratase de niños. En este proyecto se prestó especial atención a las necesidades de los distintos tipos de visitantes para determinar qué funcionalidades deberían estar presentes en las PDA de forma que especialmente los usuarios con discapacidades pudieran tener acceso a toda la información. El proyecto, tras un estudio basado en entrevistas y observaciones del uso de las interfaces propuestas, llegó a la conclusión de que era necesario encontrar un equilibrio entre la funcionalidad que estaría destinada a la gran mayoría de los usuarios y el deseo de proporcionar mecanismos que faciliten el acceso a aquellos que puedan necesitar un mayor nivel de ayuda.

En relación a nuestro proyecto, hemos tenido en cuenta las observaciones y conclusiones obtenidas en este proyecto para proponer un modelo conceptual de Museo Híbrido que permita definir contenidos en múltiples formatos y crear diferentes diseminaciones de un mismo objeto museístico según las diferentes categorías de visitantes. En nuestro caso, la organización de la información explicativa en torno a colecciones que pueden ser de cualquier categoría permite personalizar los contenidos diseminados al usuario en función de sus gustos personales.

2.2.3. Equator

Este proyecto, desarrollado en el periodo 2001-2002 como una colaboración entre cinco universidades (Glasgow, Nottingham, Bristol, College London y Southampton), fue pionero al combinar entornos virtuales, tecnología hipermedia, dispositivos de mano, y ultrasonidos para la localización. El sistema realizado para los estudios experimentales se implantó en el *Centro de Interpretación Mackintosh* dedicado a la comprensión del trabajo del arquitecto *Charles Rennie Mackintosh*, y consistía en un sistema de localización por ultrasonidos y una infraestructura de red inalámbrica en lo que se denominaba la *sala Mack*. Este sistema permitía a tres usuarios (uno in situ y dos de forma remota) visitar la sala Mack simultáneamente. El usuario in situ disponía de una PDA dotada de un sistema de posicionamiento y en el que podía visualizar la posición en un mapa de los otros dos visitantes. Uno de los visitantes remotos tenia acceso mediante una aplicación Web con un mapa del museo y el segundo visitante remoto utilizaba un sistema de realidad aumentada en el que podía ver en primera persona los avatares de los otros dos usuarios. Además, los tres visitantes disponían de un canal de audio que les permitía estar en contacto continuamente. La información accesible por los usuarios remotos eran versiones digitales de los elementos reales que podía disfrutar el visitante in situ, de manera que si los tres se encontraban en el mismo punto espacial del museo podían disfrutar del mismo elemento museístico. El interés de este

proyecto radica en que se trata de un primer intento de experimentar con el concepto de visita híbrida en el que representaciones digitales y objetos reales coexisten, aunque en este caso con usuarios diferentes. Parece evidente que uno de los objetivos principales del proyecto fue la experimentación con tecnologías de realidad mixta y la exploración de las interacciones sociales que se pudieran establecer. En este sentido, el proyecto según sus creadores fue todo un éxito dado que los usuarios participantes en la experiencia declaraban tener una sensación inmersiva satisfactoria a pesar del carácter experimental y de prototipo del sistema implementado. Sin embargo, en nuestra opinión, no parece clara la razón que justifique los distintos mecanismos de acceso a la información de forma remota (realidad virtual versus páginas Web), por lo que pensamos que se trató más bien de una decisión basada en las competencias tecnológicas de los distintos grupos de investigación que participaron en el proyecto.

2.2.4. Handscape

Este proyecto surgió en 2003 como una colaboración de 3 años entre el consorcio CIMI, el grupo de interacción hombre-máquina de la universidad de Cornell y los museos *Field Museum* (Chicago), *The Royal Botanic Gardens* (Kew, Inglaterra) y el *American Museum of the Moving Image* (New York). El proyecto investigó dos hipótesis. La primera consistía en conocer si la aplicación de tecnologías móviles ofrecía a los museos la posibilidad de modificar la manera en la que se comunicaban con sus visitantes. La segunda, si la creación de nuevas aplicaciones y servicios para los dispositivos móviles mejoraría la experiencia de visita al museo. En definitiva, uno de los objetivos principales del proyecto consistió en el diseño de sistemas en los que el uso de dispositivos móviles junto a una adecuada provisión de la información pudiera mejorar la experiencia de las diferentes actividades realizadas por los visitantes en el seno de diferentes museos. Sin embargo, a nuestro juicio, el aspecto mas interesante de este proyecto radicó en el estudio etnográfico realizado sobre un total de 100 individuos a los que se les preguntó *¨¿Qué características de diseño o funcionalidades esperaría usted de una aplicación móvil e inalámbrica destinada a su uso en museos?¨*. Los participantes respondieron que ésta debería tener:

- una interfaz fácil de usar, personalizable y multilingüe;
- capacidad de descargar información para podérsela llevar a casa y la posibilidad de crear anotaciones sobre las exposiciones;
- capacidad de comunicación mediante mensajes entre los usuarios y un mecanismo de localización para visualizar información sobre la ubicación de grupos de usuarios;

- imágenes y detalles en texto sobre las obras;
- capacidad de mostrar en qué exposiciones los usuarios emplean mayor tiempo y un mecanismo de conteo para saber el número de visitantes de cada exposición;
- indicaciones sobre otras exposiciones a visitar basándose en las que ya haya visitado el usuario;
- información detallada sobre los productos que se pueden encontrar en la tienda del museo relacionados con cada obra visitada.

Parece, pues, a la vista de los resultados obtenidos por este proyecto que tienen gran importancia la necesidad de comunicarse con otros usuarios, la curiosidad por conocer las exposiciones o las obras que el resto de usuarios visitan y la voracidad por disponer de gran cantidad de información asociada a las obras visitadas. Estos aspectos serán tenidos muy en cuenta en el desarrollo de nuestra solución.

2.2.5. Tate Modern

El museo *Tate Modern* lleva a cabo una intensísima actividad de desarrollo del concepto de Museo Híbrido, y en la actualidad incluye tres tipos de visitas híbridas diferentes: *Highlights Tour*, *British Sign Language Tour*, y *Collections Tour*. El primero está diseñado para jóvenes entre 16 y 25 años y consiste en un total de 19 obras para las que se provee contenido audiovisual tradicional junto con juegos interactivos y mecanismos de comunicación entre visitantes. El segundo está destinado a personas con discapacidad auditiva, y consiste en un conjunto de vídeos en los que mediante lenguaje de signos se presentan explicaciones sobre un conjunto reducido de obras de arte. El tercero ofrece información textual para las más de 300 obras de arte en la colección y permite acceder a una gran cantidad de información adicional sobre las obras presentes en el museo.

A pesar de ser uno de los proyectos más maduros y avanzados en cuanto a los dispositivos empleados y las interfaces de usuario diseñadas, los propios autores de esta infraestructura reconocen el escaso número de obras de arte con información multimedia accesible a través de los dispositivos. Otra crítica a la propuesta presentada por el Tate Modern son los mecanismos de navegación propuestos: selección en un mapa del museo las obras que están digitalizadas y elección mediante un pad numérico en el que poder introducir el número de la obra que se desea visualizar.

El primer mecanismo claramente no es escalable si se digitalizara un gran número de obras, dado el reducido tamaño de la pantalla de un dispositivo móvil; y el segundo, aunque intuitivo, obliga al usuario a estar introduciendo

continuamente obras de arte según su propio criterio y sin poder seguir un orden de visita predefinido siguiendo una estructura narrativa definida por un especialista. Además, esta propuesta presenta carencias a nivel de modelo conceptual que impiden el establecimiento de relaciones entre obras de arte según diferentes criterios.

2.2.6. Science Navigator

La *Royal Institution* de Gran Bretaña es una de las organizaciones científicas más importantes del país; tanto es así que 14 de sus investigadores han recibido el premio Nobel a lo largo de la historia. Como resultado de este éxito científico, esta institución dispone de objetos y obras de arte que cuentan la historia del debate científico. Además, disponen de importantes museos como el museo Faraday, dedicado al descubrimiento de la inducción electromagnética, el efecto magneto-óptico y la teoría de campos, entre otros. Esta institución realiza desde 1820 un enorme esfuerzo de diseminación de la cultura científica, de forma que en la actualidad mas de 32.000 niños visitan la *Royal Institution* cada año para participar en el programa interactivo para jóvenes.

Fruto de este compromiso con el desarrollo científico y con la comunicación de la ciencia al público en general, la *Royal Institution* creó en el 2004 una visita multimedia denominada *Science Navigator* con el apoyo de la *National Endowment for Science, Technology and the Arts*. Science Navigator permite personalizar los contenidos que se visualizan según la ubicación del visitante y su perfil. Science Navigator utiliza PDA inalámbricas con capacidades multimedia y provee a los usuarios con información adicional en función de lo que ya han visto o realizado anteriormente en el museo. La importancia de este proyecto radica en que ha sido el primero en combinar localización y tecnologías de personalización en una visita multimedia. El motor de personalización combina el perfil básico del visitante (niño, jóven y adulto) con información dinámica sobre qué objetos o información ha visitado hasta un momento dado para proponerle de forma proactiva qué nuevos objetos del museo ha de visitar. Si bien este proyecto presenta avanzadas técnicas de personalización, carece de mecanismos de interacción social entre los visitantes y plantea un modelo lineal de visita híbrida.

2.2.7. Getty Guide

Uno de los proyectos de Museo Híbrido más ambicioso es el creado en la primavera de 2004 por el museo J. Paul Getty. El museo ha evolucionado de una infraestructura basada puramente en audioguías con un contenido sonoro de aproximadamente 50 horas a un sistema basado en PDA inalámbricas. El museo tiene planes de tener una infraestructura de más de 500 dispositivos denominados colectivamente la guía Getty (Getty Guide). Es claramente la

implantación de mayores dimensiones aunque, desafortunadamente, en las primeras versiones del sistema no se ofrece ni imágenes ni videos a los visitantes. El sistema carece de mecanismos de personalización, no permite obtener información sobre las obras o exposiciones más visitadas por otros usuarios y no presenta mecanismos de interacción social en su estado actual, por lo que necesitará realizar un gran esfuerzo de desarrollo de su infraestructura de información y de visualización de contenidos.

2.2.8. Otros Proyectos

Existen otros proyectos de museos híbridos como la *Renwick Gallery* del *Smithsonian American Art Museum* que ofrece una visita híbrida con mecanismos de localización similares a los de Science Navigator pero sin mecanismos de personalización; el *Te Papa Tongarewa*, del Museo Nacional de Nueva Zelanda, que permite seguir visitas lineales y aleatorias dando soporte así a dos estilos diferentes de aprendizaje, pero que sólo tiene 24 elementos explicativos en un total de 8 habitaciones; el Technology Museum, en el norte de California, creado en 2003 por una empresa de tecnología y en el que se presentan los principales logros tecnológicos mediante tres visitas diferentes de una hora de duración. Este proyecto, desarrollado en sólo 3 meses, presenta serias carencias en cuanto a los contenidos multimedia ofrecidos, dado que éstos son principalmente extractos de audio y sólo en escasas ocasiones se proponen juegos interactivos o contenido de naturaleza audiovisual; las principales atracciones arqueológicas en Grecia como la Acropolis, el estadio de Olimpia, el Oráculo de Delphi, el Museo Arqueológico Nacional de Atenas, el Palacio de Knossos y las tumbas del Rey Felipe han incorporado con motivo de los juegos olímpicos celebrados en Atenas las últimas tecnologías de dispositivos de mano inalámbricos. Se trata de uno de los primeros proyectos de Museo Híbrido con localizaciones múltiples. Otro proyecto a mencionar es Dubbed TaggedX en el Museo de Historia Natural de Aarhus en Dinamarca, por ser uno de los precursores en el uso de las etiquetas RFID en su exposición sobre aves, de forma que cada animal en la exposición posee un número de serie único que se obtiene mediante un PDA provisto de un lector RFID. De esta forma los visitantes pueden acceder de forma automática a texto, preguntas, videos y sonidos de cada animal almacenados en una base de datos.

2.3. Conclusiones del Capítulo

La llegada de los dispositivos de mano con capacidades inalámbricas a los museos en torno al año 2002 ha sido una de las grandes revoluciones tecnológicas en este dominio, comparable sólo con la llegada de los cassettes compactos de audio en los años 80, que redujo significativamente el tamaño de

los reproductores que llevaban los visitantes, y la transición de los sitemas de audio analógico a los digitales en 1994.

Desde 2002, año en el que se inició este proyecto investigador y en el que sólo existía un prototipo muy primitivo de Museo Híbrido en el *Tate Modern*, se han desarrollado diversas propuestas de aplicación de las tecnologías móviles para la mejora de las experiencias de visitas in situ a diferentes museos.

El análisis de las propuestas existentes en el año 2002 reflejaba una total carencia de sistemas que ofreciesen mecanismos de localización de visitantes, de análisis del comportamiento de los usuarios durante su visita al museo, de personalización de los contenidos en base a la localización y a los perfiles de los usuarios, de soporte a diferentes modelos de aprendizaje, de interacción social entre los visitantes, de navegación de información interrelacionada, de integración de información distribuida y perteneciente a diferentes organizaciones, de creación de contenidos adicionales por parte de los usuarios y de generación de valoraciones y opiniones sobre los elementos de la exposición que puedan ser utilizadas por futuros visitantes.

Los proyectos analizados, desarrollados en su mayoria en paralelo a este proyecto investigador durante el periodo de 2003 a 2005, presentan, como hemos podido observar, soluciones parciales a algunos de estos requisitos. En nuestro caso, planteamos un modelo de Museo Híbrido que da soporte a diferentes estilos de aprendizaje y que permite la interacción social, el acceso a repositorios de carácter distribuido, la valoración de las obras de arte visitadas y el uso de estas valoraciones para proponer visitas dinámicas en tiempo real, ajustándose a la disponibilidad de tiempo y maximizando la importancia de las obras visitadas, medida ésta en términos de su popularidad. Todo ello demostrando que las propuestas realizadas escalan para un número de obras museísticas elevado (miles de obras) de forma que se garantice una gran variedad de posibles experiencias de visitas híbridas alternativas, incluso para un visitante en sucesivas visitas futuras a un mismo museo.

"Creación de Adán" por Miguel Angel (1510)

Capítulo 3

El Modelo MoMo de Museo Híbrido

En este capítulo definiremos un modelo de Museo Híbrido que permita soportar tres modelos de aprendizaje: conductivo, constructivo y social. Para ello, el modelo propuesto está estructurado en tres ámbitos: el modelo de contenidos, el modelo de navegación de los mismos y el modelo de interacción social. Adicionalmente, se define en este capítulo la semántica del modelo propuesto en el marco ontológico *Conceptual Reference Model* (CRM) que ha definido como estándar internacional el *International Council of Museums* (ICOM), de manera que se favorece la interoperabilidad semántica entre el modelo propuesto y los diversos modelos de metadatos existentes en el dominio de la documentación museística.

3.1. Aprendizaje en Museos Híbridos

Los Museos Híbridos, tal y como definimos en el capítulo anterior y como consecuencia de sus características inherentes, tienen un papel fundamental como entornos en los que se dinamiza la dimensión educativa de los museos tradicionales. Por esta razón, y para obtener un modelo conceptual adecuado de Museo Híbrido, es fundamental que analicemos detenidamente consideraciones de tipo teórico, filosófico y, por supuesto, pragmático relativas a las oportunidades educativas o de aprendizaje que se presentan en el contexto de los museos. Esto es, hemos de definir un marco conceptual adecuado de Museo Híbrido para dar soporte a una de sus principales actividades de carácter creativo, cultural e intelectual: el aprendizaje acerca del patrimonio.

3.1.1. Teorías del Aprendizaje en Museos

Existen múltiples teorías generales del aprendizaje, de entre las que en este trabajo destacaremos tres perspectivas interesantes y muy relacionadas con el aprendizaje en los museos: el conductismo, el constructivismo y el aprendizaje sociocultural.

El conductismo (Watson, 1913; Skinner, 1953), ya en los años 60, propone un modelo en el que el aprendizaje se traduce en la modificación del comportamiento externo del estudiante, el cual se convierte en un mero receptor de conocimiento. En el mundo de los museos, esta teoría se traduce en exposiciones en las que los objetos están clasificados según algún criterio (cronológico, estilo artístico, etc.) y el visitante es un mero receptor de información fáctica que ha de memorizar.

El constructivismo (Piaget, 1970), sin embargo, aparece a partir de los años 70 con Piaget como su máximo exponente, y propugna que el aprendizaje ha de provenir del interior del individuo, el cual de forma activa construye su conocimiento en base a sus habilidades y su madurez. Este tipo de aprendizaje se da con gran asiduidad en museos de ciencia, donde se ofrece a los visitantes actividades y experimentos que ilustran diferentes principios científicos, pudiendo adquirir dicho conocimiento sin una gran cantidad de información factual.

Finalmente, en la tercera tradición teórica encontramos la perspectiva sociocultural del aprendizaje en la que se enfatiza que la comprensión emerge a partir de la interacción entre individuos y su contexto sociocultural. Vygotsky habla de la mediación, donde diferentes categorías lingüísticas y la interacción entre individuos mediante objetos físicos permite la comprensión y la interpretación del mundo real (Vygotsky, 1978). El aprendizaje, según esta visión, no es un proceso de absorción de conocimientos genéricos, sino que se sitúa y tiene lugar bajo ciertas condiciones en un entorno determinado. Un ejemplo de esta visión son los estudios de Crowley y Callanan que han analizado cómo contribuyen al aprendizaje las interacciones sociales que tienen lugar entre los miembros de una familia que visitan simultáneamente un museo (Crowley&Callanan, 1998).

En nuestra opinión, estas tres teorías contribuyen cada una desde una perspectiva diferente a la definición de aprendizaje, pero sería incorrecto argumentar que cualquiera de ellas por separado proporciona una visión completa de dicho fenómeno. De hecho, en las teorías más modernas sobre el aprendizaje en el contexto de los museos, Gammon propone una taxonomía consistente en cinco categorías que permite clasificar todas las experiencias museísticas de carácter educativo en cognitivas, afectivas, sociales, de desarrollo de habilidades y personales (Gammon, 2001). De forma similar, en

los trabajos de Hooper se proponen también cinco áreas para clasificar las experiencias educativas en museos: conocimiento y comprensión; habilidades, valores y actitudes; disfrute, inspiración y creatividad y finalmente actividad, comportamiento y progresión (Hooper, 2002).

Esta riqueza en la categorización de las experiencias de aprendizaje en museos tiene sus raíces en la propia naturaleza de los mismos, dado que tienen una larga tradición como repositorios no sólo de objetos sino de todo el conocimiento asociado a los mismos. Son, pues, los objetos y su conocimiento asociado los que permiten desarrollar tanto el aprendizaje de tipo cognitivo (aprender sobre los objetos) como también el aprendizaje constructivo y el socio-cultural a partir de los mismos mediante las innumerables interpretaciones y las múltiples historias que ofrecen, ya se trate de obras de arte, especímenes biológicos, artefactos, documentos o cualquier otro objeto expositivo.

Como consecuencia de esta diversidad, los museos han de proveer mecanismos de aprendizaje a la carta para que los visitantes se sientan motivados por aprender. Las experiencias de aprendizaje han de ser estimulantes, relevantes y apropiadas al visitante. Para favorecer la creación de nuevo conocimiento y el establecimiento de nuevos enlaces cognitivos entre los conceptos previamente adquiridos por los visitantes y los presentados en el museo, es necesario que las experiencias sean los más multidisciplinares posible. Se han de plantear mecanismos en el contexto del museo que permitan el crecimiento intelectual de los visitantes. Por tanto, una de las tareas principales de un museo a la hora de facilitar nuevas experiencias enriquecedoras es crear un contexto adecuado para el aprendizaje, dando estructura y coordinando las distintas alternativas de manera que se provean los elementos esenciales de todo proceso de aprendizaje en un museo: el acceso al conocimiento, el disfrute sensorial de los objetos expuestos y la provisión de un contexto social y cultural que permita apreciar el valor único de la institución y sus contenidos.

Veremos, pues, cómo se pueden conseguir estos objetivos con el uso de las nuevas tecnologías y en concreto con las asociadas al concepto de Museo Híbrido; además, plantearemos un modelo conceptual de sistema de información que permita dar soporte a los modelos teóricos de aprendizaje considerados relevantes en el dominio de los museos.

3.1.2. Aprendizaje in situ en Museos

Una de las características fundamentales del proceso de aprendizaje en un Museo Híbrido es el hecho de que éste ocurre in situ en las propias instalaciones del museo mediante la *coexistencia* de los objetos reales expuestos y

los objetos virtuales interactivos que proporcionan las tecnologías digitales. Para algunos, ambos tipos de objetos son mutuamente excluyentes:

> *"ubicar una pantalla de ordenador en el seno de una exposición de artefactos puede distorsionar en gran medida la experiencia del visitante. La interacción con dichos elementos tiende a sobrepasar las formas de interacción alternativas más lentas, menos controlables y más complejas que tienen lugar cuando los visitantes establecen interacciones mentales con los objetos expuestos a partir de su curiosidad" (Boon, 2000).*

En nuestra opinión, esta visión es demasiado simplista y está basada en un concepto muy primitivo del proceso de aprendizaje que subyace a los visitantes. La existencia de objetos digitales interactivos en el seno del museo es fundamental, entendiendo por interactivo aquel objeto que induce en el visitante un proceso complejo de toma de decisiones y no una simple interacción reducida a acciones simples como "empezar" y "parar" una explicación.

Otra de las características de gran relevancia en el contexto del aprendizaje in situ es la relativa facilidad con la que la *colaboración* se hace posible gracias a la presencia física de múltiples actores en un mismo espacio entregados a una actividad similar. Además, si a esto le unimos las capacidades técnicas ofrecidas por una infraestructura de Museo Híbrido que permiten acceder de forma ubicua a recursos, información, conceptos relacionados e incluso a otros actores que hayan pasado por el museo previamente o se encuentren en otra sala, entonces podemos hacernos una idea de las posibilidades colaboración que emergen en este nuevo contexto. Los aspectos colaborativos del aprendizaje en los museos han estado siempre en una posición preeminente [], sobre todo haciendo énfasis en aquellos aspectos de la colaboración que permiten compartir materiales educativos, impresiones, y opiniones. En nuestro caso, estamos interesados en dar soporte al aspecto social del aprendizaje, de forma que se puedan romper las barreras entre los visitantes locales al museo y los remotos, lo presentes y los ausentes. La intención es que los visitantes a un museo (actuales y pasados) puedan comunicarse para poder establecer asociaciones en el seno de las colecciones de un museo e incluso entre las colecciones mismas, y que dichas asociaciones se puedan convertir en recursos no sólo para visitas subsecuentes del actor que establece las asociaciones, sino también para otros actores de un mismo grupo de interés.

Finalmente, existe un factor importante a tener en cuenta en el aprendizaje in situ, que es el de la *personalización* de los contenidos según el perfil del usuario, y su ajuste a la movilidad en el seno del museo. De esta forma, en un Museo Híbrido se ha de poder sacar provecho del hecho de que los dispositivos

utilizados son: altamente móviles, de forma que están disponibles en cualquier momento en el que el usuario necesite aprender; individuales, de forma que es posible adaptar los contenidos al perfil y la forma de interacción del usuario; y poco intrusivos, dado que el usuario puede detectar contextos y acceder a información relacionada a los mismos sin necesidad de abandonar el contexto.

Nuestra propuesta de modelo conceptual de Museo Híbrido se basa en las ideas anteriormente expuestas. Por un lado, permite el aprendizaje de tipo conductivo para la adquisición de conocimiento básico y memorístico sobre los objetos expuestos in situ. Pero, además, permite un modelo de aprendizaje más avanzado, de tipo constructivo, mediante el establecimiento de relaciones entre las distintas colecciones presentes en el museo. Estas relaciones permiten a los usuarios construir conocimientos personalizados basados en las distintas exploraciones o navegaciones que cada visitante realice del conjunto de relaciones definidas. Plantearemos, para conseguir este objetivo, un modelo genérico de navegación que permita explorar cualquier estructura compleja de objetos y conceptos relacionados. Adicionalmente, la flexibilidad de que dotaremos a nuestro modelo conceptual permitirá el establecimiento de nuevas relaciones (no predefinidas) que puedan ser utilizadas por otros visitantes para construir sus propios modelos de conocimiento durante o tras su visita.

Finalmente, no olvidaremos en nuestro modelo la importancia del factor socio cultural, de forma que plantearemos un modelo de interacción social que permita, a través de la comunicación con visitantes in situ y con visitantes previos de los objetos expuestos, adquirir nuevo conocimiento mediante la discusión, la colaboración y la compartición en el seno de un grupo de los conceptos aprendidos por sus miembros.

3.2. Modelos Existentes de Museos Híbridos

Como paso previo al planteamiento de un nuevo modelo conceptual de Museo Híbrido, hemos de revisar los modelos planteados en los proyectos más relevantes de los mencionados en el capítulo anterior, de forma que podamos tener una idea clara de las principales contribuciones e innovaciones que se proponen en MoMo.

3.2.1. Proyecto iTour

En el proyecto iTour se plantea un modelo de datos jerárquico, tal y como se puede observar en la Figura 3. En este modelo las obras de arte tienen asociadas un conjunto básico de metadatos que las describen, y a partir de los mismos se puede acceder a información catalogada en tres clases denominadas "Learn", "Create/Interact" y "FAQ" (Frequently Asked Questions) La

información contenida en la categoría "Learn" es información relativa a la biografía del autor, o enlaces a videos del mismo y recursos adicionales. En la categoría "Create/Interact" se disponen los elementos de carácter interactivo y en la de FAQ las preguntas más frecuentes y sus respuestas. Si bien es interesante, desde el punto de vista de la usabilidad de la aplicación, la diferenciación en categorías de aquellos elementos interactivos de los que no lo son, el carácter jerárquico del modelo es muy limitado puesto que no es posible establecer relaciones de cualquier tipo entre las obras de arte ni proceder, como consecuencia de ello, a una navegación compleja de los elementos expositivos siguiendo relaciones semánticas que existan entre ellos. Es por tanto, un modelo estático predefinido, además de centralizado, que es difícil de alterar sin afectar de forma importante a la aplicación que lo manipula. Tampoco existen capacidades de interacción social ni mecanismos que permitan definir visitas generadas automáticamente según ciertos criterios.

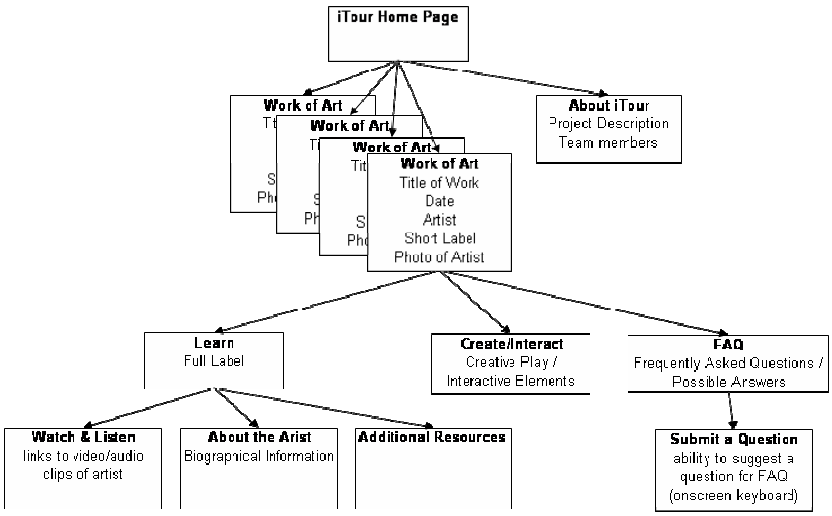


Figura 3. Estructura Jerárquica del proyecto iTour

3.2.2. Proyecto Multimedia Tour

En primer lugar, hay que destacar en este proyecto el pequeño número de obras de arte disponibles (en torno a 30). El modelo subyacente, aunque no se describe con exactitud en ningún documento por tratarse de un proyecto comercial, es muy similar al modelo de iTour (nos basamos para esta afirmación en la experiencia real del autor de esta tesis con este sistema). Es un modelo en el que existen videos informativos sobre cada obra, y comentarios en video por parte del autor durante su proceso creativo. También existen servicios de localización de la obra en las instalaciones del museo. De nuevo, es

un modelo simple de objetos de arte no relacionados donde se plantea sobre todo un modelo educativo de tipo conductivo.

3.2.3. Proyecto GettyGuide

Este proyecto es quizás el más simplista en cuanto a su modelo conceptual, dado que el Museo Híbrido en este caso se concibe como una secuencia de obras de arte navegables de forma lineal y en las que únicamente se puede acceder a una explicación de cada obra de arte en cuestión. En palabras de los propios autores del proyecto

> *"The prototype did very little in terms of architecture design and user functionality" (Marshack, 2004).*

Figura 4. Navegación lineal en GettyGuide

3.2.4. Proyecto GuideBook

Este proyecto sigue un modelo simple, como reconocen los propios autores en:

> *"It was not envisioned to support the implementation of a fully functional system, but rather to point the way for future developments" (Semper, 2002)*

Su objetivo es proporcionar un contexto genérico sobre las exposiciones existentes, y no tanto una explicación detallada sobre cada uno de los objetos que las componen. Es por ello que no existe, a nuestro juicio, un modelo de datos ni un modelo de navegación de contenidos claro en este sistema, sino que se trata de un conjunto de páginas Web poco estructuradas. Existe una página inicial para cada exposición, y hasta 3 niveles de navegación en profundidad a partir de la página inicial. Los contenidos y la organización de los mismos en la página son altamente dependientes de la exposición. Por ejemplo, en la Figura 5 podemos observar que para la exposición "Echo Tube", dedicada a explicar las propiedades del sonido, existe un primer nivel con 5 categorías de información: "I wonder", "articles", "explanations", "echo locations" y "exhibit evolution". La categoría "explanations" contiene un video, unos gráficos y una explicación sobre el uso del sonido para medir el tiempo, pero carece, como puede apreciarse, de todo tipo de enlaces a otros objetos relacionados con el sonido que permitieran una navegación en profundidad. Es por tanto un sistema que calificaríamos como orientado a la exposición, y no orientado a los objetos expositivos como lo son los ejemplos anteriores y también lo es MoMo.

Figura 5. Navegación basada en páginas Web de GuideBook

3.2.5. Otros Proyectos

El resto de proyectos considerados en el capítulo anterior presentan modelos similares a los anteriores, por lo que no los detallaremos. Cabe señalar que SottoVocce facilita una mínima interacción social porque permite que dos dispositivos diferentes se encuentren sincronizados de manera que las acciones que se realizan en uno de ellos sean repetidas en el otro.

3.3. El Modelo de Contenidos de MoMo

Describir la naturaleza de los objetos existentes en un museo ha sido un problema recurrente que se ha abordado tradicionalmente mediante el empleo de metadatos. El término *metadatos* tiene la misma raiz griega que el término *metamorfosis*. *Meta* significa cambio y los *metadatos*, por tanto, son datos que

cumplen el papel principal de describir el origen de otros datos y los cambios que hayan podido sufrido éstos últimos (datos estructurados sobre los datos).

En el contexto de los museos existen docenas de supuestos "estándares" de metadatos, cientos de metadatos no estándar y cientos de sistemas terminológicos diferenciados en función del objetivo que se persiga con los mismos. Por ejemplo, tal y como se describe de forma detallada en (CHIN, 2002) existen metadatos para la descripción de colecciones como los esquemas RSLP, CDWA, VRA Core y RLG REACH; para facilitar la búsqueda de recursos como Dublin Core, Z39.50 y DarwinCore; para la descripción de elementos multimedia como NISO Z39.87, DIG35, Mpeg-7; para la preservación digital como RLG, METS, OAIS; para la propiedad intelectual y el comercio electrónico como INDECS, MPEG-21, DOI, sistemas de identificación ISO y para la descripción de recursos educativos como GEM, IMS, CanCORE, IEEE LTSC-LOM. Gran parte de esta diversidad se debe a que dichos esquemas se han diseñado para la captura y el almacenamiento de datos. Si bien cualquiera de estos mecanismos de representación podría haber sido elegido para representar la visión estática de los contenidos de un Museo Híbrido, no hemos encontrado ningún modelo de entre los anteriormente mencionados que tenga en cuenta los aspectos navegacionales y de interacción social que se dan en el contexto específico de nuestra aplicación. El no ajustarnos a ningún esquema de representación existente en el ámbito de los museos no supone un grave problema porque, como veremos más adelante, hemos definido la interpretación semántica de los conceptos del modelo de MoMo en términos del modelo de referencia conceptual definido por el *Comité Internacional para la Documentación* (CIDOC) del Consejo Internacional de Museos (ICOM). Este modelo de referencia (Doerr, 2003), comúnmente denominado CIDOC-CRM, es un estándar ISO que permite la interoperabilidad entre distintos esquemas de metadatos en el ámbito museístico y por tanto la interpretación semántica de MoMo que hemos definido en función de este modelo permite que cualquier esquema de metadatos expresado en términos de CIDOC-CRM pueda interoperar semánticamente con nuestro modelo, lo cual es cierto para la mayoría de esquemas principales de metadatos de carácter museístico.

Definiciones Previas

Elemento Expositivo: Elemento de naturaleza material, catalogado y susceptible de ser expuesto en una institución museística. Llamaremos Ω al conjunto de elementos expositivos.

Elemento Explicativo: Elemento de naturaleza multimedia consistente en un elemento visual y uno o varios elementos sonoros (tantos como idiomas). Un

elemento explicativo puede proveer información descriptiva particular y propietaria de un elemento expositivo o no. Llamaremos E al conjunto de elementos explicativos

Diseminación: Elemento Explicativo relativo a un único elemento expositivo. Dado un elemento expositivo $w \in \Omega$, denotaremos $Dis(w) \subseteq E$ al conjunto de sus diseminaciones.

Nótese que es necesario realizar una distinción entre el concepto de diseminación y el de elemento explicativo puesto que pueden existir unidades de explicación de carácter genérico que no sean propias o particulares a ningún elemento expositivo concreto. Además, una visita museística, como definiremos más adelante, estará compuesta por diseminaciones relativas a elementos expositivos y no por elementos explicativos genéricos.

Como se puede observar en el modelo de la Figura 6, representamos los conceptos descritos anteriormente realizando una distinción clara entre lo que es un elemento expositivo (*ExhibitObject*) y sus diferentes concreciones según se trate de una obra de arte (*Artwork*), un animal (*Animal*), etc. Asimismo, distinguimos entre el objeto expuesto como tal y sus diferentes diseminaciones (*Dissemination*) que tienen en forma de elementos explicativos (*ExplanatoryItem*) las dos dimensiones definidas anteriormente (*VisualExplanation*) y (*AudioExplanation*).

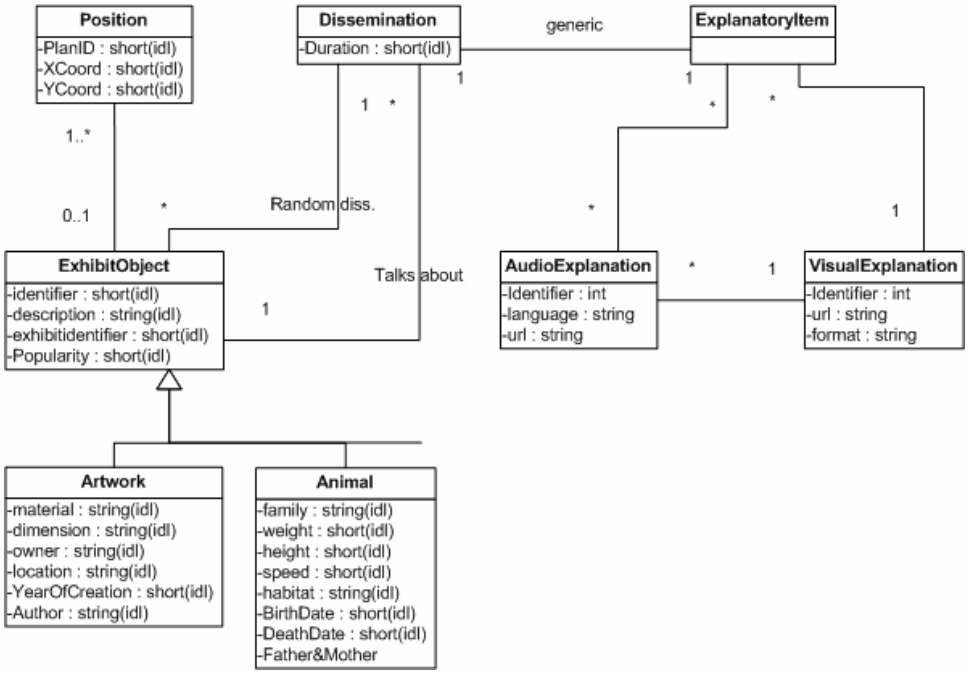


Figura 6. Modelo de clases UML básico de contenidos

Este modelo básico, de permanecer así, sería un modelo no estructurado que si bien sería adecuado para dar soporte a un aprendizaje de tipo conductista, tendría serías limitaciones para soportar formas de aprendizaje constructivo dado que no existirían relaciones entre los distintos elementos expositivos. Más aún, en la realidad museística estas relaciones entre elementos expositivos existen, como por ejemplo: las obras realizadas en un mismo contexto histórico, las obras de un mismo autor, las obras previas a una dada, las obras de un determinado estilo pictórico, los animales de una misma familia, etc. Por tanto, se han de proporcionar mecanismos que permitan a un usuario explorar otros elementos expositivos que pudieran estar relacionados con uno dado. Además, estas relaciones en el dominio de los objetos expositivos han de poder ser definidas de forma dinámica por los creadores de contenidos atendiendo a diferentes criterios o metadatos e incluso por los propios visitantes en el ejercicio de un proceso de aprendizaje constructivo.

Las relaciones entre elementos expositivos inducen relaciones entre sus respectivas diseminaciones. De manera formal, dado un conjunto de relaciones entre elementos expositivos $\Upsilon = \{\Upsilon_i \subseteq \Omega \times \Omega : i = 1,..,n\}$ se induce un conjunto de relaciones $\Upsilon^* = \{\Upsilon_i^* \subseteq E \times E : i = 1,..,n\}$ de forma que

$$w \Upsilon_i w' \longrightarrow d \Upsilon_i^* d' \quad \forall w, w' \in \Omega \ \forall d \in Dis(w), d' \in Dis(w')$$

Definición

Dado un elemento expositivo $w \in \Omega$, una diseminación $e \in Dis(w)$ y una relación $\Upsilon_K^* \in \Upsilon^*$, se define como *proyección de* Υ_K^* sobre *e* al subconjunto $\Upsilon_K^*(e) \subset \Upsilon_K^*$ tal que $\Upsilon_K^*(e) = \{(e', e'') \in \Upsilon_K^* : e'' = e\}$

Definición

Dada la proyección $\Upsilon_K^*(e)$ se define como *proyección unitaria válida* de $\Upsilon_K^*(e)$ a cualquier subconjunto $\Psi \subset \Upsilon_K^*(e)$ tal que $\forall (e, e_1), (e, e_2) \in \Psi \longrightarrow \exists w \in \Omega : e_1, e_2 \in Dis(w)$ y denominaremos $\Psi(e)$ al conjunto de proyecciones unitarias validas para *e*.

El concepto de proyección unitaria es necesario dado que en el transcurso de una visita museística a partir de la diseminación de una obra, si seleccionamos un tipo de relación, no será posible navegar la proyección completa sino que únicamente se podrá explorar una diseminación por cada obra de arte que forme parte de la relación. Permitir proyecciones no unitarias causaría confusión a los visitantes dado que existirían varias diseminaciones por cada

elemento expositivo y el proceso cognitivo para identificar mentalmente la relación original entre los elementos expositivos se vería enormemente dificultado.

Adicionalmente a las relaciones entre elementos expositivos y las relaciones entre diseminaciones pueden existir relaciones entre elementos expositivos y elementos explicativos. Estas últimas representan información contextual que no es específica de un elemento expositivo particular, sino que son elementos explicativos de carácter genérico aplicables a múltiples elementos expositivos.

Definición

Dado un elemento expositivo $w \in \Omega$, definimos su conjunto de *relaciones explicativas contextuales* $\Gamma(w)$ como $\Gamma(w) = \left\{ \Gamma_i \subset \{w\} \times \mathrm{E} : i = 1,..,n \right\}$

Dado un elemento expositivo $w \in \Omega$, para toda diseminación $e \in Dis(w)$ se induce el conjunto de relaciones explicativas contextuales $\Gamma^*(e) = \left\{ \Gamma_i^* \subset \{e\} \times \mathrm{E} : (e,e') \in \Gamma_i^* \longleftrightarrow \exists \Gamma_i \in \Gamma(w) : (w,e') \in \Gamma_i \right\}$

Para permitir este tipo de exploración basada en relaciones entre elementos expositivos y sus correspondientes diseminaciones hemos completado el modelo básico introduciendo el concepto de colección. Una colección es un contenedor de objetos que pueden ser elementos expositivos, diseminaciones, e incluso elementos explicativos en general. Las colecciones de elementos expositivos servirán para modelar el conjunto de relaciones Υ. Las colecciones de diseminaciones se utilizarán para modelar las proyecciones (totales y unitarias). Las colecciones de elementos explicativos servirán para modelar el conjunto de relaciones explicativas conceptuales. El conjunto de requisitos funcionales asociados al modelo de contenidos descrito anteriormente se detalla en el ANEXO A

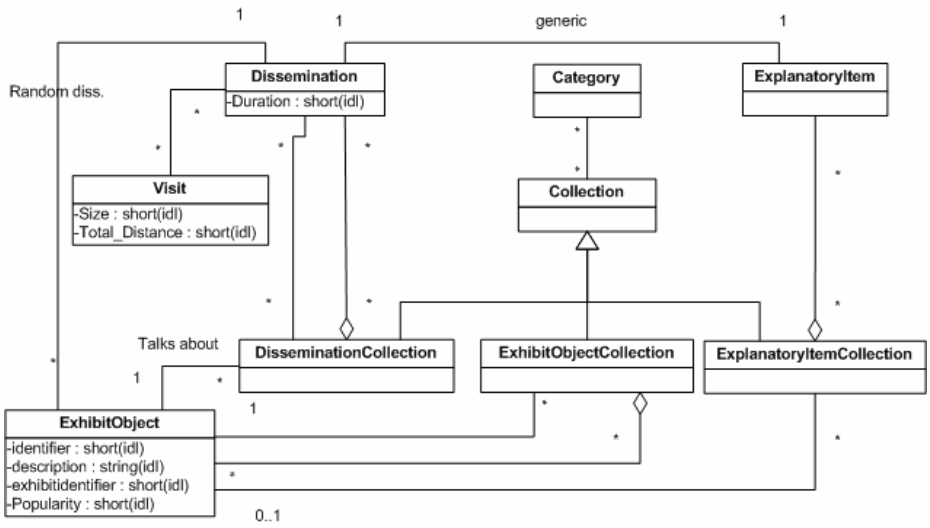


Figura 7. Modelo de contenidos aumentado

3.4. El Modelo de Navegación de MoMo

Como ya hemos mencionado con anterioridad, una visita se concibe como una secuencia ordenada de diseminaciones de elementos expositivos. Recorrer una visita de esta manera, sin acceder a las colecciones que aportan información adicional relacionada, es una forma de navegación que denominaremos *navegación horizontal*. Sin embargo, si a partir de una determinada diseminación se accede a las colecciones asociadas (bien explorando las proyecciones unitarias de relaciones inducidas, o bien explorando las relaciones explicativas contextuales), entonces hablaremos de una navegación en profundidad o *navegación vertical*, dado que este proceso puede repetirse varias veces a partir de cada nueva diseminación. En la Figura 8 podemos observar de forma esquemática cómo a partir de una navegación horizontal podemos acceder a diferentes tipos de colecciones asociadas. Una vez seleccionada una colección nos encontramos ante un conjunto nuevo de diseminaciones que, a su vez, pueden tener asociadas nuevas colecciones que podrán ser exploradas por el usuario.

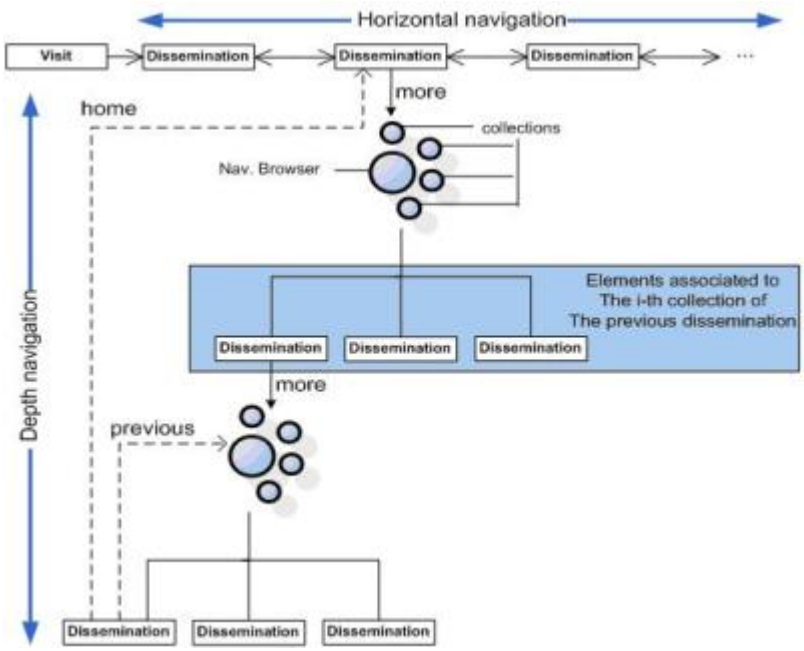


Figura 8. Estructuras de Navegación Horizontal y Vertical

Definición

Dado un conjunto Ω de elementos expositivos y sus correspondientes diseminaciones $Dis(\Omega)$, se define una *navegación horizontal válida* H como una secuencia de diseminaciones

$$H = \left\{u_1, u_2, ..., u_N : \forall u_i, u_j \nexists e \in \Omega : u_i, u_j \in Dis(e) \;\; i \neq j \;\; i, j \in \left\{1, ..N\right\}\right\}$$

Dada una navegación horizontal válida H, y desde una diseminación $u \in \mathrm{H}$ se define una navegación vertical válida Λ relacionada con *u* como una secuencia $\Lambda = \left\{u_1, u_2, ..., u_m, u_{m+1}\right\}$ de forma que

$$u_i \in Dis(\Omega) \;\; i \in \left\{1, .., m\right\} \;\; \wedge \;\; u_{m+1} \in E$$

$$u_1 = u$$

$$\nexists e \in \Omega : u_i, u_j \in Dis(e) \;\; i \neq j \;\; i, j \in \left\{1, .., m+1\right\}$$

$$\forall u_i, u_{i+1} \exists \Psi_K \in \Psi(u_i) : (u_i, u_{i+1}) \in \Psi_K \;\; i \in \left\{1, .., m-1\right\}$$

$$u_{m+1} \notin Dis(\Omega) \rightarrow \; \exists \Gamma^*_K \in \Gamma^*(u_m) : (u_m, u_{m+1}) \in \Gamma^*_K$$

Nótese que en una navegación vertical válida el último elemento puede ser una diseminación o un elemento explicativo genérico. En este último caso, se trata de un elemento terminal en la navegación, puesto que a partir de un elemento explicativo genérico no es posible alcanzar ninguna otra diseminación. En la Figura 9 podemos ver un ejemplo concreto de navegación vertical en la interfaz de usuario propuesta en MoMo siguiendo el modelo presentado hasta ahora. A partir de una diseminación que el usuario esta visualizando se acceden a las colecciones asociadas. En este ejemplo existen dos colecciones ("previous" y "same author") inducidas a partir de relaciones entre elementos expositivos. En el ejemplo de la figura el usuario realiza una navegación vertical seleccionando la colección "same author" y, a continuación, seleccionando una de las diseminaciones pertenecientes a dicha colección. El resultado es que el usuario navega entre dos diseminaciones de dos obras de arte que pertenecen al mismo autor.

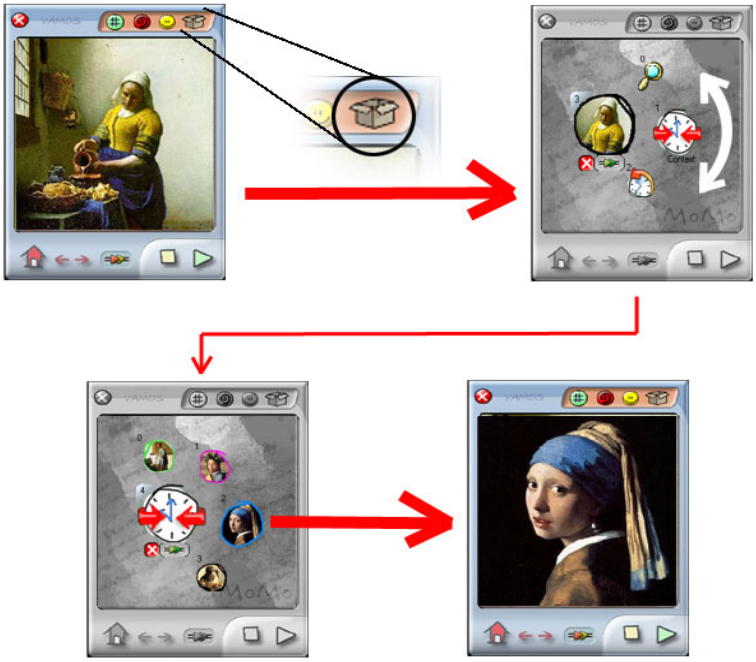


Figura 9. Ejemplo de navegación vertical

3.5. El Modelo de Interacción Social de MoMo

El modelo de interacción social, tal y como describimos anteriormente, ha de permitir la colaboración y el establecimiento de contextos socio culturales que

contribuyan a la realización de procesos de aprendizaje sociales. En el anexo de Especificación de requisitos IEEE se describen todos los requisitos relacionados con la interacción social y en la Figura 10 se muestra el modelo de clases completo obtenido tras el análisis de los mismos.

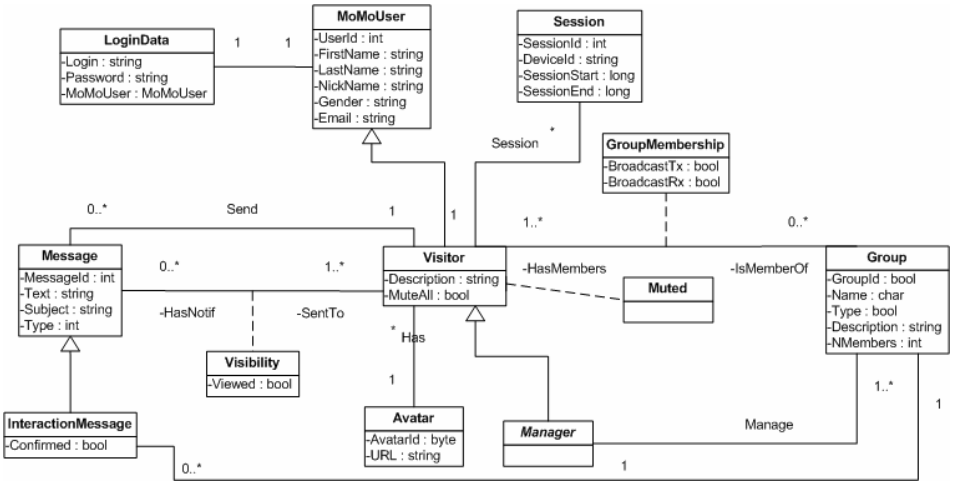


Figura 10. Modelo de clases para la Interacción Social.

El Modelo de interacción social propuesto permite el establecimientos de grupos de interés que puedan estar en contacto permanente mediante el envío de mensajes (individuales o colectivos) durante la visita al museo. Sin embargo, a diferencia de un mecanismo de mensajería tradicional, el usuario puede inhabilitar la recepción de mensajes por parte de otro u otros usuarios, e incluso el usuario que crea el grupo, erigido en moderador del mismo, puede inhabilitar el envío de mensajes en modo *broadcast*. Además, en el modelo propuesto se pueden crear grupos privados en los que la incorporación de nuevos miembros es en todo momento controlada por el gestor del grupo.

En la Figura 11 y Figura 12, podemos observar las clases específicas que definen nuestro modelo de usuario:

- MoMoUser. representa los datos personales de cada usuario del sistema, tanto para los presentes en el museo como los externos al mismo y que pueden personalizar sus visitas con anterioridad.
- Visitor. Se corresponde con aquellos usuarios del sistema que están presentes en el museo utilizando un cliente MoMo y por tanto tienen capacidades de interacción social durante la visita.
- LoginData. Contiene la información confidencial que permite a cada usuario identificarse ante el sistema.

- Avatar. Representa a los distintos avatares que existan en el sistema. Todos los visitantes tendrán un avatar con el que se sientan identificados.
- Session. Contiene la información de las sesiones, actual y pasadas, de cada visitante.

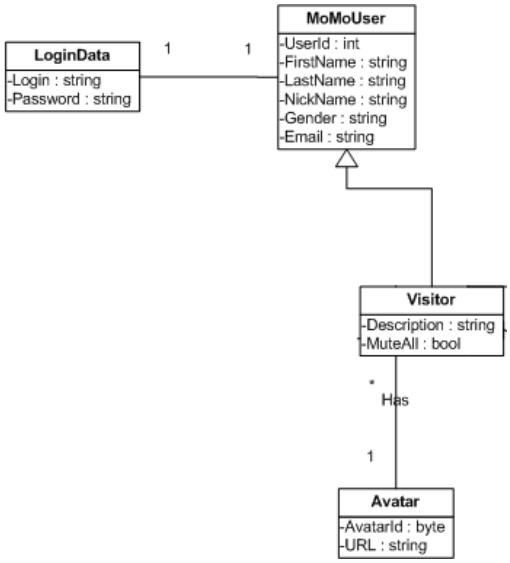


Figura 11. Modelo de usuario.

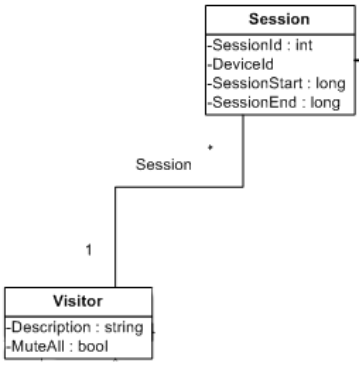


Figura 12. Sesión de un visitante.

Para la gestión de notificaciones (Figura 13) hemos identificado las siguientes clases:

- Message: representa las notificaciones que pueden enviarse los visitantes para comunicarse.

- InteractionMessage. son aquellas notificaciones que implican la realización de una acción por parte del receptor: invitación para unirse a un grupo (el receptor aceptará o rechazará la invitación), y petición de unión a un grupo privado (el gestor del grupo privado aceptará o denegará la solicitud de adhesión).
- Visibility: indica si un mensaje ha sido visualizado o no por su receptor.
- Muted. Cuando un visitante quiere impedir que otro visitante del museo le pueda seguir enviando notificaciones, tan sólo ha de silenciar al visitante.

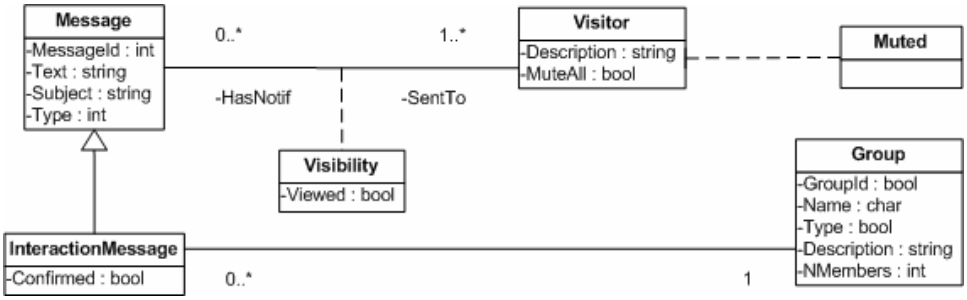


Figura 13. Modelo de notificaciones.

En la Figura 14 mostramos las clases relacionadas con la gestión de grupos y sus miembros:

- Group: representa a un grupo de visitantes.
- GroupMembership: para indicar que visitantes son miembros de un grupo. Un grupo siempre tendrá al menos un miembro: su gestor. También indica el deseo del visitante miembro de recibir, o dejar de recibir, notificaciones dirigidas al grupo (recepción de difusiones, BroadcastRx) y la decisión del gestor para que un miembro no pueda enviar mensajes al grupo (envío de difusiones, BroadcastTx).
- Manager: Se trata del visitante que creó el grupo y sólo él puede realizar la gestión del mismo.

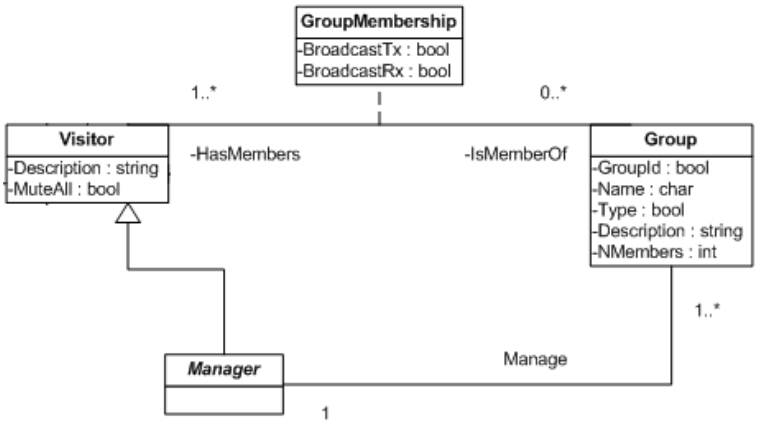


Figura 14. Datos de un grupo y de sus miembros.

3.6. El Modelo Orientado a Servicios de MoMo

Los modelos descritos anteriormente han de ir acompañados de una especificación en forma de servicios que permita a aplicaciones externas (en nuestro caso residentes en dispositivos móviles) ejecutar los casos de uso derivados del análisis. Como consecuencia de estos casos de uso, hemos identificado dos conjuntos diferenciados de servicios: aquellos que facilitarán la navegación de colecciones (ver Tabla 1) y aquellos que permiten las actividades de interacción social (ver Tabla 2). Estos servicios han sido implementados en el contexto de nuestro trabajo como Servicios Web sobre la plataforma .NET (MSDN, 2005). Veremos en el próximo capítulo cómo estos servicios se conjugan con servicios mediadores de carácter distribuido que permitirán tanto la navegación como la interacción social en una infraestructura MoMo completamente distribuida.

Método	Función
getVisits	obtiene un conjunto de objetos *CVisit* para una sesión determinada
getDisseminationSet	obtiene un conjunto de objetos *CDissemination* en un rango para una visita dada
getSupportedCollectionCategories	obtiene un conjunto de objetos *CCategory* para una visita dada
getCollectionMembers	obtiene los miembros de una colección, objetos *CDissemination* o *CExplanatoryItem* según se trate de una colección de diseminaciones o de elementos explicativos
getRandomDissemination	obtiene un objeto *CDissemination* a partir de

	un identificador en la exposición (navegación aleatoria)
getAreaDisseminations	devuelve el conjunto de diseminaciones que componen el área que ocupa el orden indicado en la visita dinámica indicada
traceVisitedItem	conocer qué elemento, y su categoría, visita un visitante
vote	indica el voto favorable o desfavorable, del visitante identificado por la sesión, acerca del elemento explicativo que acaba de ver
getPosition	devuelve la ubicación en el museo de la obra asociada al identificador de diseminación indicado
generateDynVisit	genera una visita dinámica para la sesión indicada, que empieza y acaba en las salas del museo indicadas y dentro del tiempo máximo indicado

Tabla 1. Métodos Servicio Web de Navegación

Método	Función
getNotification	Recupera una notificación específica de un determinado visitante
logIN	Permite que un usuario se identifique al sistema
getNumberOfAvatars	Recupera el número de avatares disponibles en el sistema
muteAll	El visitante no quiere recibir notificaciones de ningún otro visitante
unsilenceGroup	Habilita la recepción de difusiones de un grupo del que se es miembro
getVisitorsEI	Recupera una lista de visitantes que han visto con anterioridad el mismo elemento explicativo
setGroupProfile	Modifica los valores actualizables del perfil de un grupo
getVisitorsIcanWriteTo	Devuelve los visitantes actuales a los que puede enviar notificaciones
fireMember	Elimina una lista de miembros de grupos que gestiona un visitante concreto
getVisitors	Devuelve todos los visitantes del museo (desde una fecha marcada por un parámetro de configuración del servidor)
muteGroupMember	A una serie de visitantes se les incapacita para el envío de notificaciones a un grupo del que son miembros. Esta acción la lleva a cabo el gestor del grupo al que pertenecen
getGroupProfile	Devuelve el perfil de un grupo determinado
getAvatars	Obtiene las URL de los avatares disponibles en el sistema
deleteNotifications	Elimina un conjunto de notificaciones que un

	determinado visitante ha recibido
logOUT	Permite que un usuario acabe su sesión en el sistema
silenceGroup	Deshabilita la recepción de difusiones de un grupo determinado al que el visitante pertenece
getGroupMembers	Obtiene el conjunto de miembros de un grupo determinado
getPublicProfile	Obtiene el perfil público de un determinado visitante
unmuteAll	El visitante desactiva la opción de no recibir notificaciones de nadie
leaveGroup	Elimina un determinado visitante de una lista de grupos a los que pertenece
getManagerGroups	Obtiene el conjunto de grupos que gestiona un determinado visitante
unmuteGroupMember	A una serie de visitantes se les devuelve la capacidad de enviar notificaciones a un grupo del que son miembros. Esta acción la lleva a cabo el gestor del grupo al que pertenecen
getAvailableGroups	Devuelve todos los grupos a los que puede unirse un cierto visitante
joinGroups	Añade un determinado visitante, o solicita su unión, a una lista de grupos
deleteGroup	Elimina una lista de grupos
getGroupNonMembers	Devuelve todos los visitantes que no son miembros de un determinado grupo
muteVisitor	El visitante no quiere recibir notificaciones de una lista de visitantes
getGroupsIcanWriteTo	Devuelve los grupos de los que es miembro el solicitante, y para los que tiene permiso de envío de difusiones
getGroupMembersIcan WriteTo	Obtiene el conjunto de miembros de un grupo determinado, al cual también pertenece el solicitante, a los que puede enviar notificaciones
getVisitorGroups	Obtiene el conjunto de grupos a los que pertenece un determinado visitante
createGroup	Crea un grupo con los datos proporcionados
sendNotification	Envía un mensaje determinado a los visitantes especificados
setProfile	Modifica el perfil del visitante con los nuevos datos
unmuteVisitor	El visitante quiere volver a recibir notificaciones de una lista de visitantes
getNotifications	Recupera todas las notificaciones recibidas para un determinado visitante

Tabla 2. Métodos del Servicio Web de Interacción Social

3.7. La Semántica del Modelo MoMo

Como ya hemos comentado anteriormente, el modelo propuesto en el presente trabajo carecería de validez si no pudiera ser puesto en una perspectiva ontológica común con el resto de formas de descripción de datos en el contexto de los museos. Por ello, es necesario definir los mecanismos adecuados para dotar de interoperabilidad semántica al modelo propuesto. No es de extrañar que cada vez más se haga uso de ontologías formales como marcos conceptuales para la integración de información (Doerr, 2003).

Dotar de un marco semántico común al conjunto tan diverso de metadatos y de formatos usados para representar la información en el ámbito museístico puede parecer una labor imposible. Afortunadamente, la comunidad de este dominio ha hecho el esfuerzo para separar las ontologías de alto nivel, que representan conocimiento extraído de los cuerpos de metadatos, de la mera terminología. El exponente más claro de este esfuerzo de abstracción es, como ya avanzamos con anterioridad, el modelo de referencia conceptual (CRM) propuesto por el comité CIDOC del Consejo Internacional de Museos, que se ha constituido como el marco ontológico de referencia en el dominio que nos ocupa, estando avalado también por la Organización Internacional de Estándares (ISO). Por esta razón, en el presente trabajo hemos abordado la tarea de definir las correspondencias entre todos y cada uno de los elementos definidos en el modelo de MoMo y los conceptos ontológicos del modelo CRM. Pasaremos, pues, a continuación a describir con cierto detalle el modelo CRM para posteriormente enumerar las correspondencias entre éste y nuestro modelo.

3.7.1. El Modelo CRM

El modelo conceptual de referencia CRM puede ser definido como una ontología del dominio del patrimonio cultural. El término ontología se deriva del campo de la filosofía donde se utiliza para referirse a las asunciones sobre la existencia que subyacen a una visión particular del mundo. Dicho de otro modo, describe qué tipos de cosas existen en el mundo y cuáles son las relaciones existentes entre las mismas [5 paper Crofts]. El hecho de que CRM se defina como una ontología específica de un dominio determina que su ámbito no es la descripción de un universo completo, sino que se restringe a definir qué entidades existen y cuáles son sus relaciones en el dominio de la patrimonio cultural. Más aún, su orientación ontológica significa que en su formulación no se han tenido en cuenta aspectos de implementación, ni se han asumido procesos de negocio o reglas institucionales particulares. En CRM no se definen reglas de validación, ni formatos de datos, ni elementos de interfaces de usuario. Por tanto, CRM no garantiza la compatibilidad o interoperabilidad

a bajo nivel sino que asegura la compatibilidad conceptual. Esto no quiere decir que renunciemos a una interoperabilidad de bajo nivel, la cual detallaremos en el próximo capitulo.

El diseño del modelo CRM parte de los siguientes objetivos:

- Cubrir todos los aspectos de las formas de documentación del patrimonio cultural que sean necesarios para el intercambio de información en un contexto global.
- Permitir la documentación de conocimiento parcial e incluso contradictorio.
- Permitir la integración y el intercambio de información sin pérdida semántica entre esquemas de distinto nivel de riqueza descriptiva.
- Proveer un entorno extensible que permita la incorporación de nuevos conceptos ontológicos.

El modelo CRM está definido en la actualidad como un modelo semántico orientado a objetos, dado que en el momento de su definición no existían mecanismos estándares de definición de ontologías. En particular, el modelo consiste en una jerarquía de 81 clases relacionadas mediante 132 propiedades que son heredadas por diversas clases aplicando los mecanismos de herencia del modelo orientado a objetos. Aunque este número de clases y propiedades refleja la inherente complejidad del modelo, hay que hacer notar que el metamodelo que subyace al mismo define un conjunto mínimo de metaentidades y metapropiedades. CRM conceptualiza el dominio de la documentación museística de una manera muy general definiendo (Figura 15) *Actores* (individuales o en forma de grupo) que participan en *Entidades Temporales* (eventos), que se ven afectados por *Entidades Físicas* (cosas materiales) y por *Objetos Conceptuales* (ideas y conceptos) que ocurren en *Lugares* durante ciertos *Intervalos de Tiempo*. Las *Apelaciones* (nombres) se usan para identificar cualquiera de estas entidades, y los *Tipos* se utilizan para clasificarlos mediante un adecuado nivel de detalle. Las propiedades de CRM se agrupan en el siguiente conjunto de metapropiedades: de *Identificación* de elementos del mundo real mediante nombres del mundo real; de *Clasificación* de elementos del mundo real; de *Descomposición* en partes de objetos conceptuales y físicos, periodos, actores, lugares y tiempos; de *Participación* de elementos persistentes en entidades temporales; de *Localización* de periodos en espacio y tiempo y de objetos físicos en el espacio; de *Influencia* de los objetos en las actividades y los productos y viceversa; y de *Referencia* de objetos de información a cualquier elemento del mundo real.

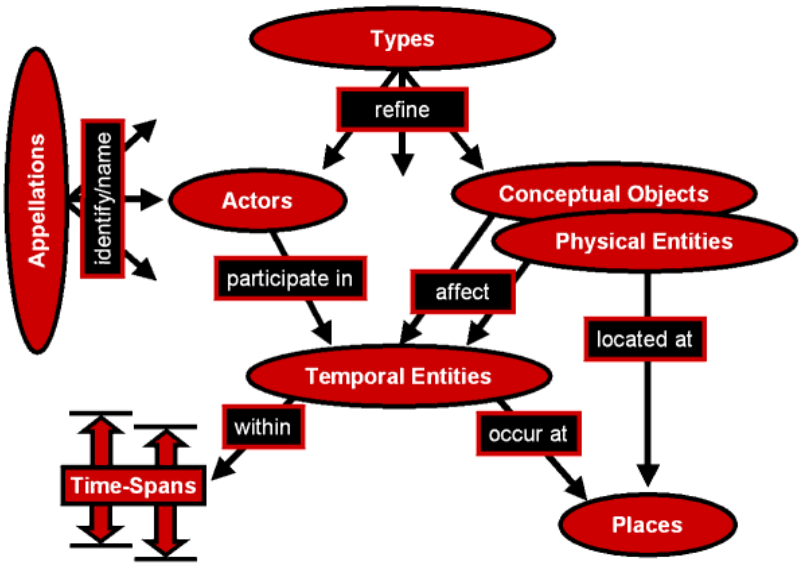


Figura 15. Metamodelo cualitativo de CRM. Fuente: (Doerr,2003)

En concreto, dentro del modelo CIDOC es de especial interés destacar los mecanismos por los que se representan información de carácter espacial y temporal. La información espacial (ver Figura 16) se expresa mediante *Lugares* que pueden tener diversas *Apelaciones de Lugar* que describen tanto las localizaciones actuales como pasadas de los *Objetos Físicos*.

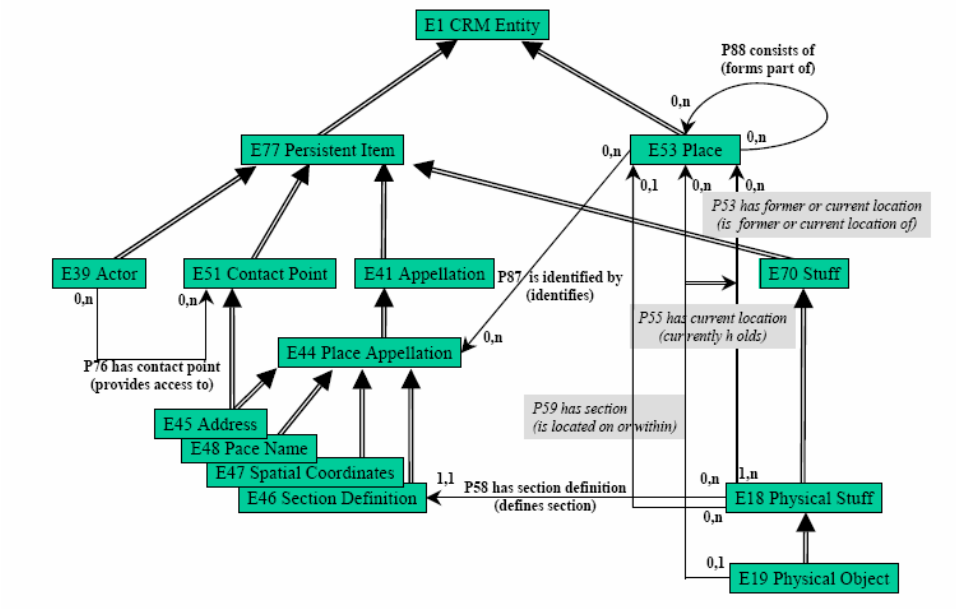


Figura 16. Razonando sobre información espacial en CRM

Por otro lado, las *Entidades Temporales* (bien sean *Eventos*, *Períodos* o *Estados*) están compuestas por *Intervalos de Tiempo* y son, como vimos en el metamodelo cualitativo anterior, el punto de relación entre Actores, *Objetos Conceptuales* y *Físicos*, y *Lugares* (ver Figura 17).

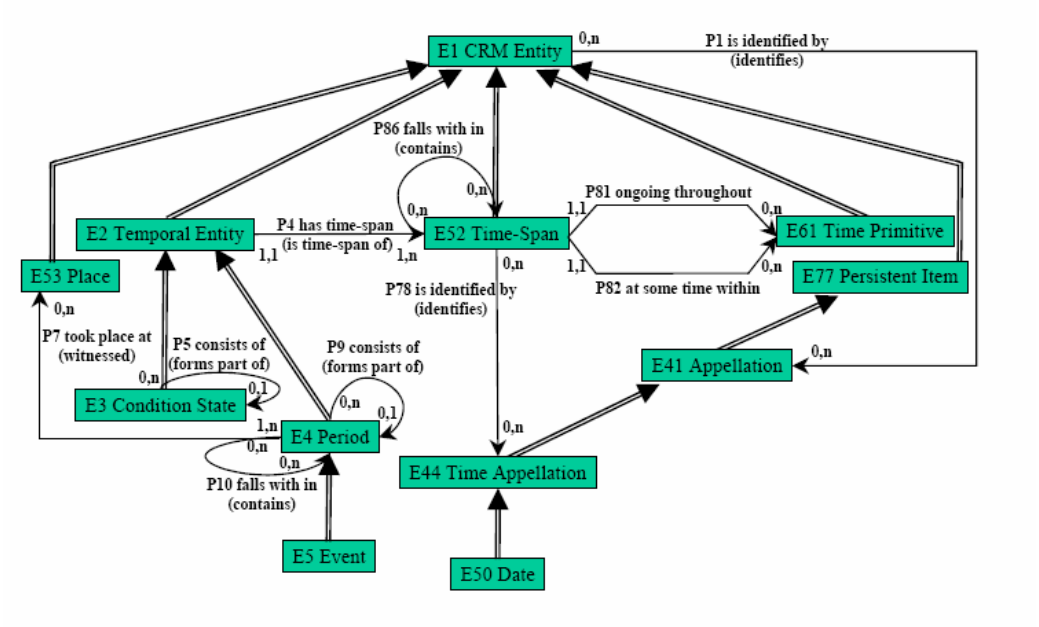


Figura 17. Razonando sobre Información temporal en CRM

3.7.2. El Modelo MOMO- CRM

La expresión del modelo de MoMo en términos de la ontología definida en CRM hace principalmente uso de los elementos ontológicos *Actor, Objeto Información, Apelación, Dimensión y Actividad*. La entidad Actor cumple un papel fundamental en la expresión ontológica de un visitante del museo (Figura 18), mientras que el modelo de colecciones y de visitas viene principalmente expresado en función de la entidad *Objeto Información* (Figura 19 y Figura 20). Finalmente, el modelo de interacción social es el más complejo e incluye entidades CRM de tipo Actor, Grupo y Objeto Información entre otras. La expresión de la correspondencia MoMo-CRM en este trabajo se ha realizado en términos de grafo semántico RDF (W3C, 2005) y no de manera orientada a objetos como jerarquía de clases. Esto es así porque, como veremos en el próximo capítulo, la integración de repositorios distribuidos de patrimonio cultural se plantea en términos de repositorios semánticos expresados como redes RDF que seguirán el modelo MoMo-CRM y entre los que existirán enlaces en términos de propiedades MoMo-CRM distribuidas.

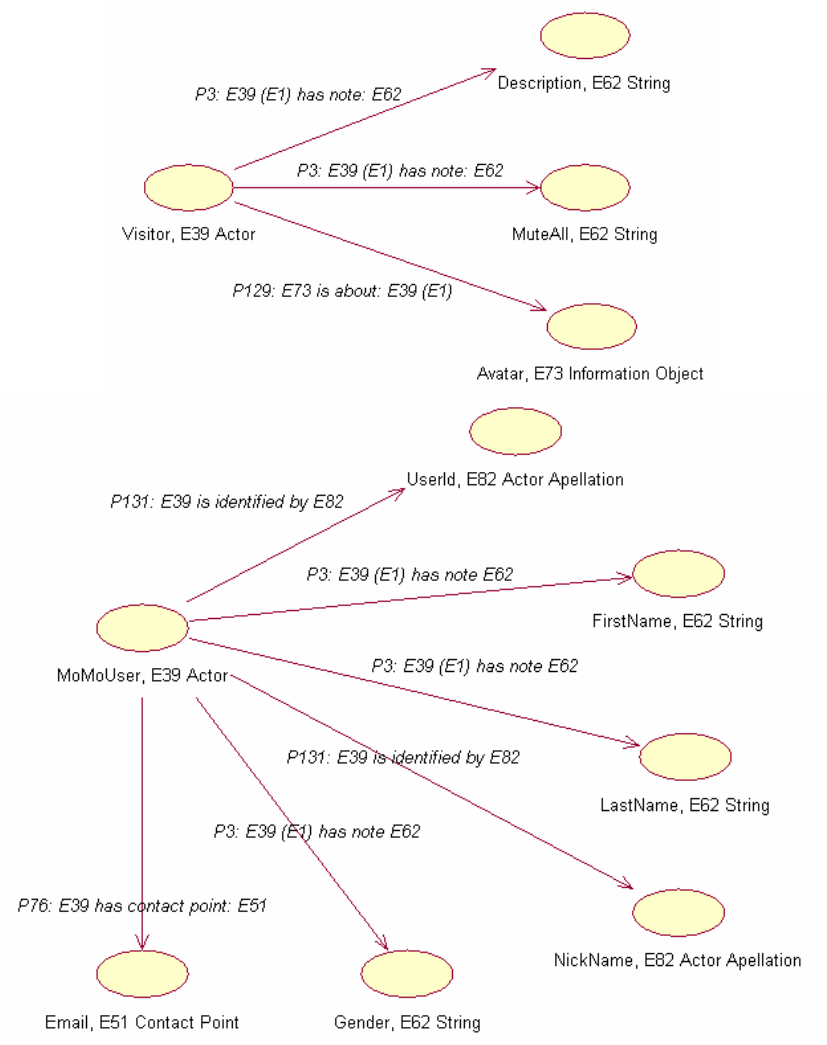


Figura 18. Modelo de usuario MoMo-CRM

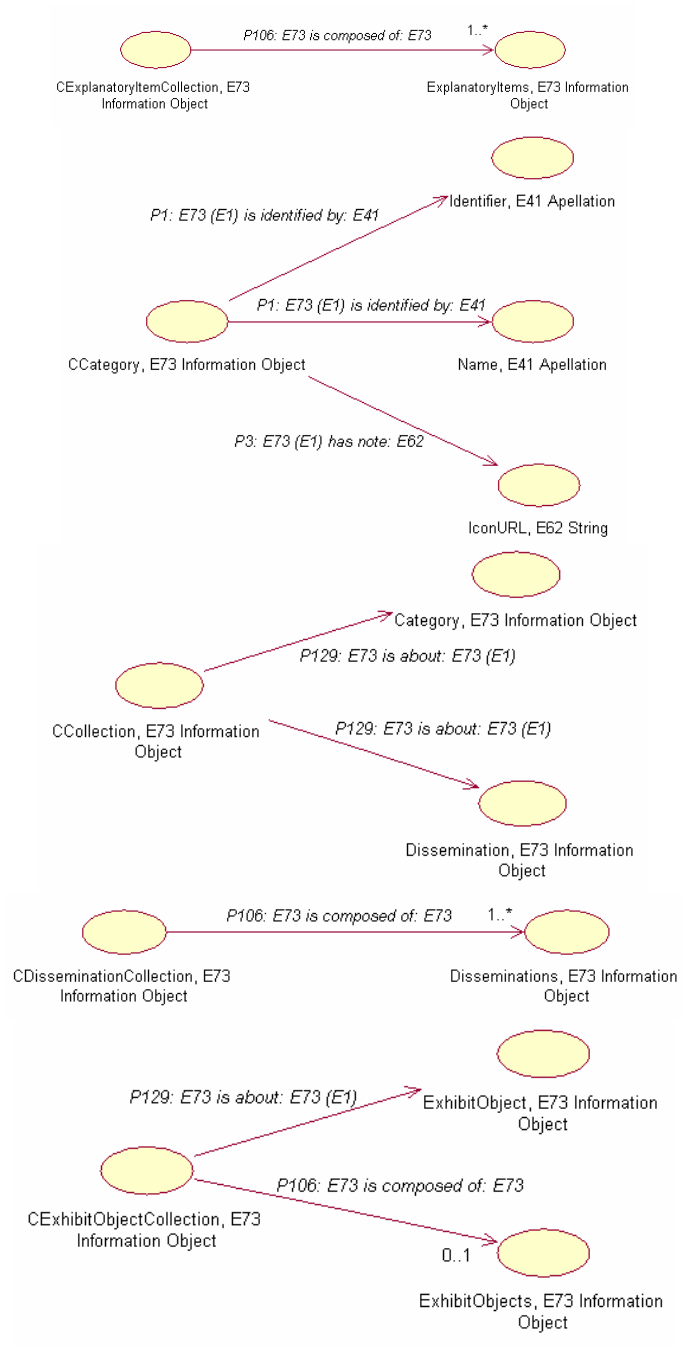


Figura 19. Modelo de Colecciones MoMo-CRM

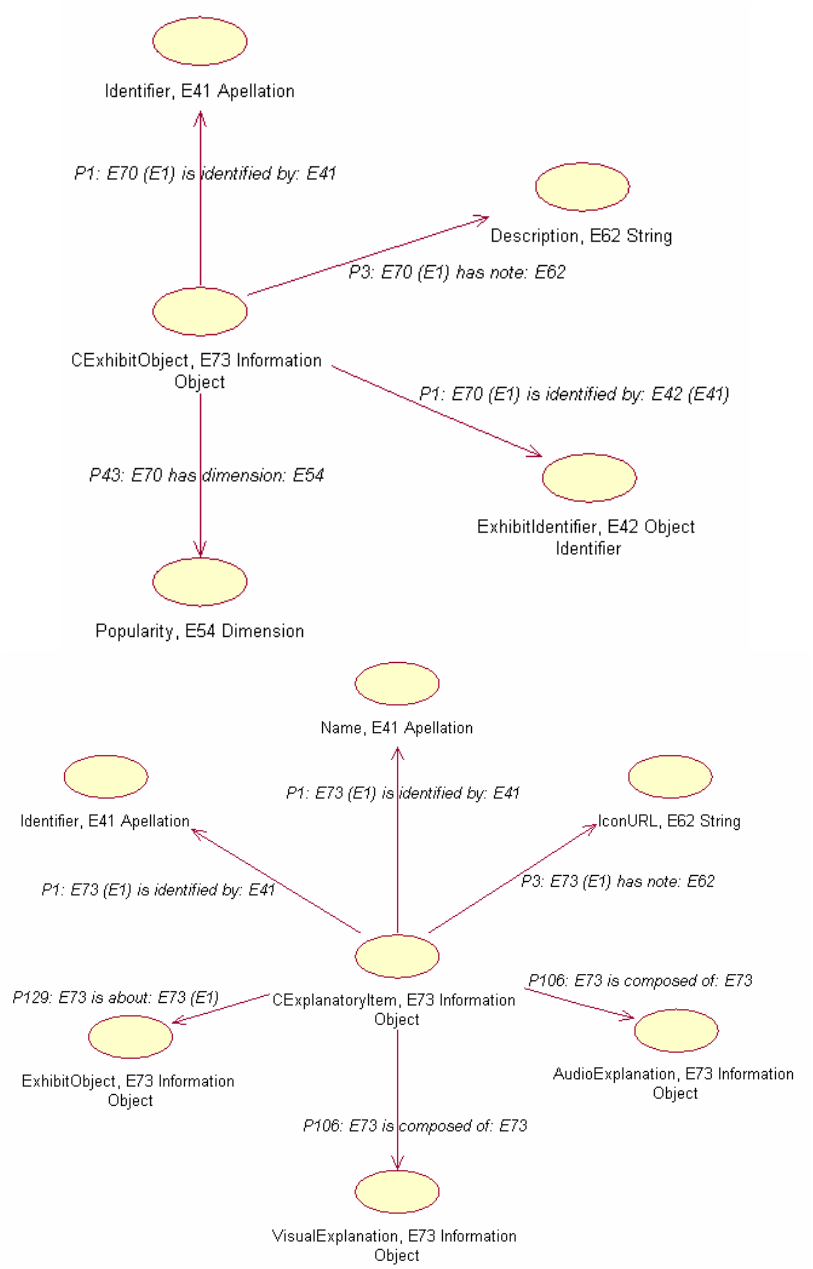


Figura 20. Modelo de Visitas MoMo-CRM I

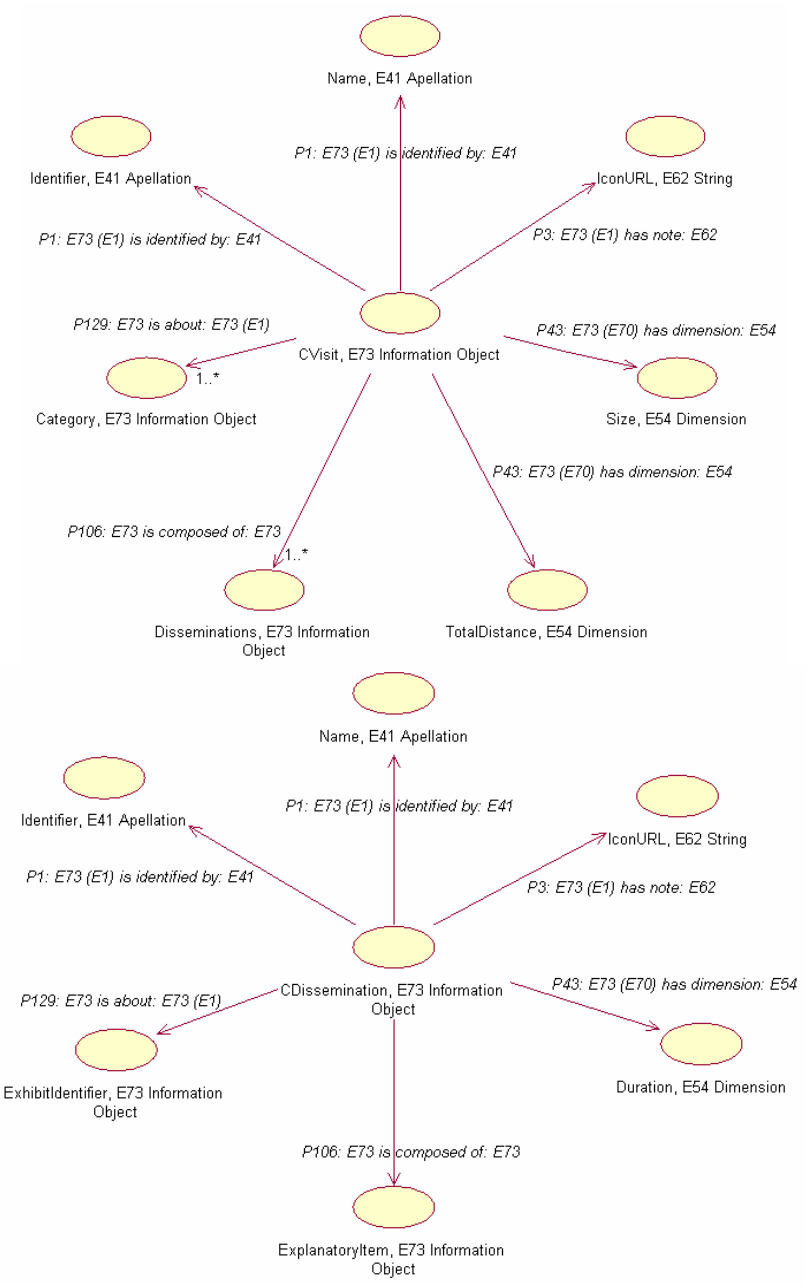


Figura 21. Modelo de Visitas MoMo-CRM II

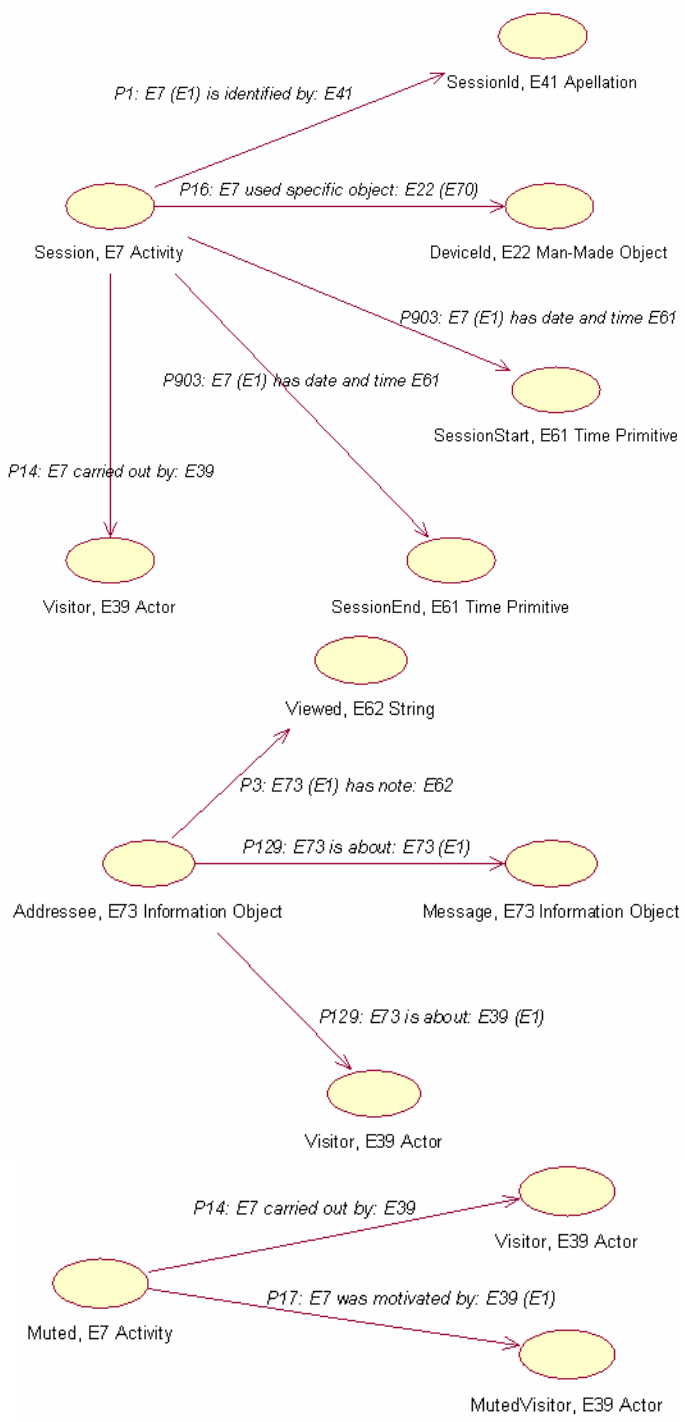


Figura 22. Modelo de Interacción Social MoMo-CRM I

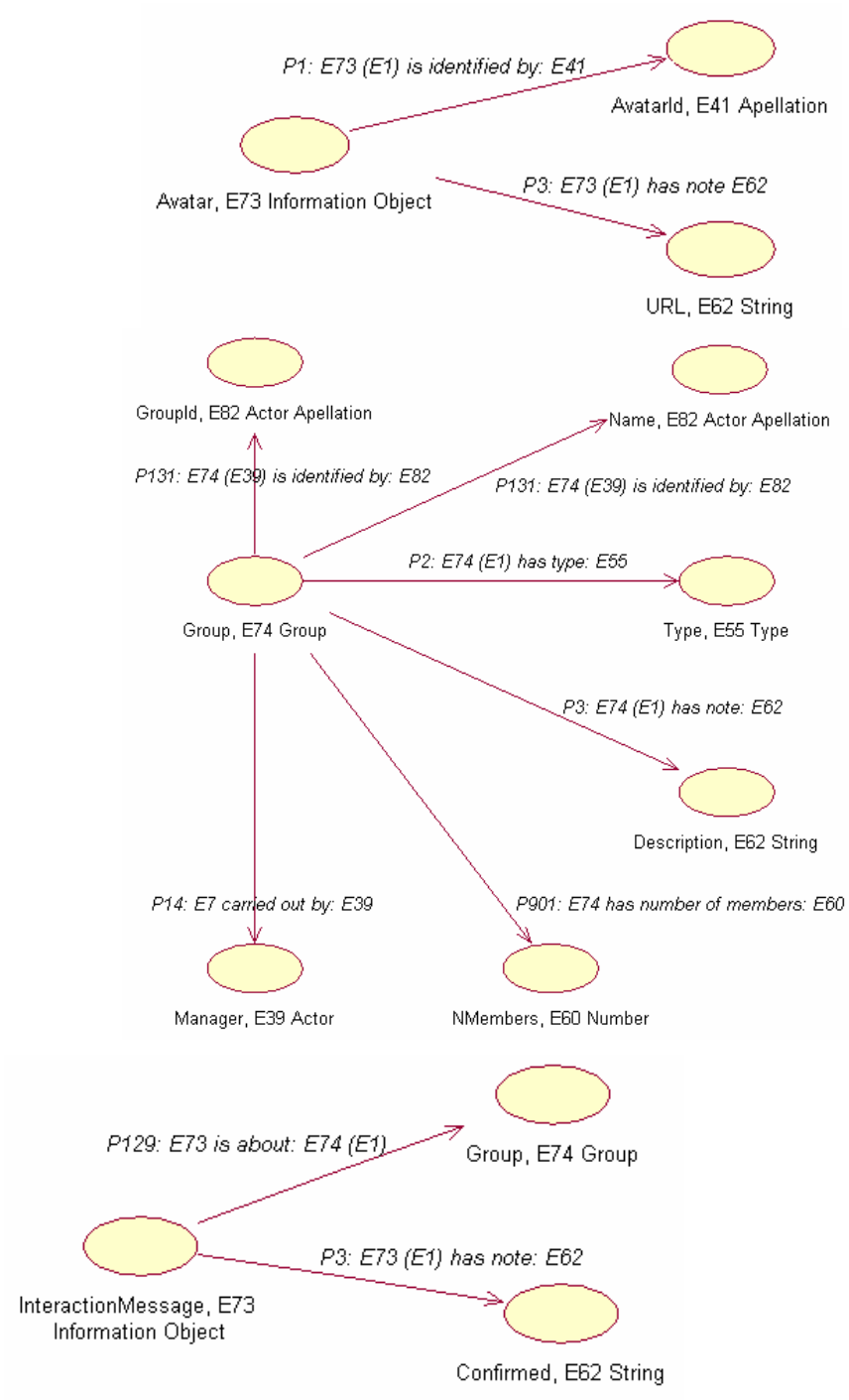


Figura 23. Modelo de Interacción Social MoMo-CRM II

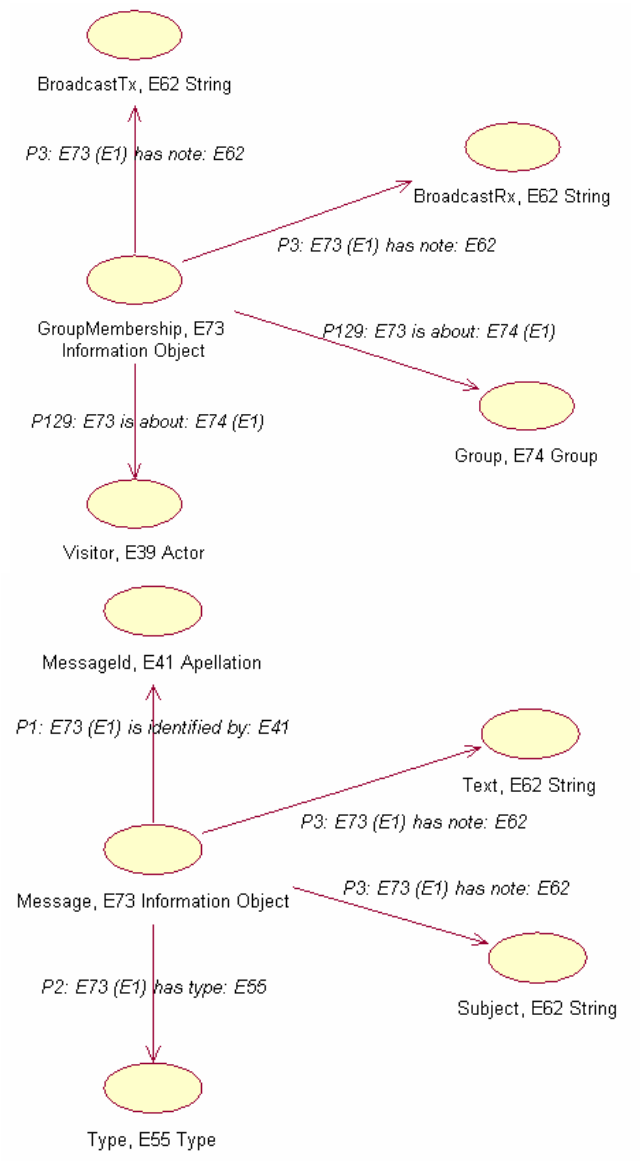


Figura 24. Modelo de Interacción Social MoMo-CRM III

3.8. Conclusiones del Capítulo

En este capítulo hemos propuesto un modelo conceptual de Museo Híbrido que permite definir procesos educativos de aprendizaje no sólo de tipo conductivo, sino también de carácter constructivo y social. Este planteamiento es novedoso con respecto a las propuestas existentes de museos híbridos en las que se plantean modelos simples de exploración de contenidos basados en una

concepción primitiva del aprendizaje. En nuestra propuesta es posible la definición, mantenimiento y exploración personalizada tanto de las colecciones de objetos preservados en el museo como de las relaciones entre las mismas y entre los objetos y conceptos explicativos asociados a los mismos. Nuestro modelo permite procesos de aprendizaje simples basados en navegaciones horizontales, así como procesos altamente estructurados y complejos a partir de las diferentes diseminaciones que se pueden crear de un objeto expositivo, las proyecciones que de dichas diseminaciones se pueden hacer, el establecimiento de relaciones entre dichos elementos y las navegaciones en profundidad que los visitantes pueden realizar como parte de un proceso constructivo de conocimiento en el seno del museo. Adicionalmente, se ha propuesto un modelo de interacción social que permite la creación y gestión de grupos de interés que permiten tanto la comunicación in situ entre individuos presentes en el museo como la comunicación con visitantes previos. De esta forma es posible, mediante un mecanismo de interacción social, poder construir conocimiento adicional sobre los objetos expuestos en el museo de forma cooperativa.

Otro de los aspectos a resaltar en este capítulo ha sido la definición de la semántica del modelo propuesto en términos de la ontología CRM propuesta por CIDOC. Gracias a esta caracterización semántica posibilitamos la interoperabilidad semántica entre nuestro modelo y los modelos existentes en el ámbito de la documentación museística de forma que repositorios con representaciones de datos diferentes a las propuestas en este capítulo puedan ser integrados mediante el uso de mediadores que hagan corresponder de forma semántica conceptos equivalentes en los distintos espacios de información. Por otro lado, al expresar el Modelo semántico de MoMo como un grafo constituido por recursos y propiedades expresados en formato RDF podremos beneficiarnos de los múltiples mecanismos de almacenamiento de información semántica que han sido desarrollados en el campo de la Web semántica.

Nuestro modelo de Museo Híbrido, como veremos en el siguiente capítulo, es altamente susceptible de ser distribuido, lo que contribuirá a la escalabilidad de los contenidos que pueden formar parte de un proceso constructivo de aprendizaje, y al establecimiento de grandes volúmenes de relaciones entre conceptos y objetos pertenecientes a distintos actores, ya sean instituciones museísticas o incluso individuos particulares en un entorno masivamente distribuido.

"Torre de Babel" por Abel Grimmer (1570-1619)

Capítulo 4

Federaciones Semánticas en Museos Híbridos

4.1. Introducción

La idea de disponer de un recurso universal de acceso a la información es un tema recurrente en muchos dominios, que varían desde la ciencia ficción, siendo un claro ejemplo la película "The Matrix", donde la información de forma simbólica es accesible a través de entornos virtuales, hasta los ciberspacios a los que estamos acostumbrados donde podemos navegar y obtener información que está relacionada mediante hiperenlaces.

La investigación fundamental relacionada con este tipo de entornos ha sido el objetivo de diversos programas de investigación, entre los que cabe destacar, en primer lugar, los programas sobre "*Global/Grid Computing*" (Foster, 1999), en el que todo tipo de recursos que son gestionados por diferentes instituciones pueden ser compartidos en un entorno global y abierto para construir entornos de computación más potentes; y en segundo lugar, el esfuerzo internacional sobre Web semántica (Berners-Lee, 2001), en el que se trata de dotar de contenido semántico a la información existente en la red, de forma que las tareas de búsqueda, recuperación y filtrado sean más efectivas.

Ambos esfuerzos investigadores han sido en el pasado ortogonales: en el campo de la Computación Global se incidía en los protocolos y servicios que permitieran compartir recursos de computación, dejando de un lado los aspectos semánticos de la información manipulada, mientras que en el campo de la Web Semántica, la infraestructura de computación sobre la que se facilita

el acceso a la información es la Web tradicional, que no provee de mecanismos para construir entornos de computación global.

A medida que los usuarios se integran en la red y demandan servicios de computación más complejos, se hace inevitable que ambas visiones lleguen a un punto de unión en el que la computación global tenga a su disposición recursos descritos de forma semántica y, a su vez, la Web Semántica disponga de mecanismos que le permitan integrar e interrelacionar recursos semánticos que sean gestionados de forma individual por múltiples usuarios, tal y como abandera el modelo de computación global.

Un paso adelante en la convergencia entre ambos mundos supone la definición de mecanismos que permitan a diferentes repositorios de información semántica constituirse como una organización virtual federada, de forma que se respete la gestión y propiedad individual de los mismos pero al mismo tiempo se ofrezca una visión integral para los posibles miembros de la federación. Las *federaciones semánticas* pueden ser consideradas, desde el punto de vista de la computación global, como un servicio más de nivel colectivo para la gestión de información semántica altamente distribuida, y desde el punto de vista de la Web Semántica, como un mecanismo eficaz para evitar la existencia de islas semánticas que sean de poco provecho para la realización de tareas complejas debido precisamente a su aislamiento.

En este capítulo abordamos la problemática derivada de la carencia de mecanismos para la integración de repositorios semánticos heterogéneos y altamente distribuidos, proponiendo como solución un modelo de federación. Adicionalmente, para certificar la validez de nuestra propuesta validaremos las ideas descritas en este punto en el campo de los museos híbridos para resolver el problema de la integración semántica en este contexto.

Lo que resta del presente capítulo está estructurado como sigue. En primer lugar, describimos el problema que se pretende resolver, revisamos el estado del arte y las condiciones que han contribuido a su existencia. Seguiremos con una propuesta de solución a dichos problemas, la evaluación experimental de la misma y su aplicación a los museos híbridos.

4.2. La Evolución de la Computación Global

A mediados de la última década se introdujo el término "*The Grid*" para hacer referencia a un nuevo modelo de infraestructura distribuida utilizado en la resolución de problemas científicos y de ingeniería. El principal problema subyacente, que justifica la aparición de dichas infraestructuras, es

> *"la puesta en común de recursos de forma coordinada para la resolución de problemas en organizaciones virtuales dinámicas y de carácter multi-institucional." (Foster, 1999)*

En este tipo de organizaciones virtuales emergen un gran número de problemas que han de ser resueltos de forma colaborativa, poniendo en común recursos que van desde simples ficheros hasta software, pasando por nodos de computación, datos e infraestructuras de interconexión.

Los primeros esfuerzos en el campo de la computación global cristalizaron en proyectos cuyo objetivo era la interconexión de centros de supercomputación para proveer de recursos de computación a un gran número de aplicaciones de altas prestaciones. De esta forma se iniciaba lo que se denominó metacomputación, y en la que los principales problemas a resolver eran las comunicaciones, la gestión de recursos y la manipulación de datos remotos. Los dos proyectos que abanderaron estos inicios fueron FAFNER (Cowie, 1996) un proyecto dedicado a la factorización de números en el campo de la encriptación, e I-WAY (Foster, 1997) para la integración de redes de altas prestaciones.

Tras estos primeros proyectos, la segunda generación de Grids se planteó el objetivo más ambicioso de no sólo unir unos pocos centros de computación, sino hacer que el sistema se convirtiera en un entorno ubicuo, de forma que fuera una infraestructura distribuida a escala global que fuera capaz de resolver problemas de heterogeneidad, escalabilidad y adaptabilidad. Para ello, este campo experimentó un fuerte desarrollo de numerosas componentes de *middleware* tales como planificadores (*schedulers)*, gestores de recursos, sistemas de objetos distribuidos, etc. Los grandes exponentes de esta generación son GLOBUS (Foster, 2001) y Legion (Natrajan, 2001), sobre los cuales se desarrollaron una gran cantidad de proyectos de computación global (Baker, 2002) siguiendo un modelo arquitectónico (Figura 25) que distingue entre aquellos elementos de *middleware* encargados de encapsular la funcionalidad o dar acceso a recursos individuales (niveles de Fábrica, Conectividad y Recursos) frente a aquellos que integran colecciones de recursos, también entendidos como elementos de *middleware* de carácter distribuido (nivel Colectivo).

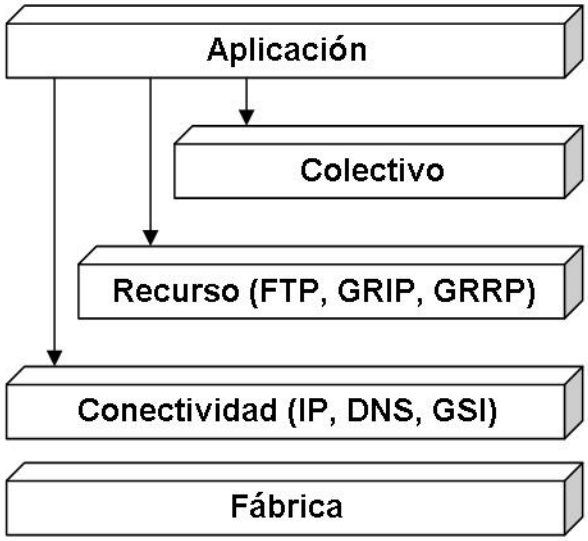


Figura 25 Modelo arquitectónico para aplicaciones de computación global

Sin embargo, pese al gran avance producido, todavía restaban grandes problemas por resolver, principalmente la inexistencia de estándares para la definición, publicación y acceso tanto a los servicios de *middleware* como a los metadatos asociados a recursos, datos, entidades, etc. La adopción de una visión más orientada a los servicios y la creciente importancia en la gestión de los metadatos son el gran caballo de batalla de los entornos de computación global actuales, también llamados de tercera generación. La adopción del concepto de servicio tiene grandes implicaciones en la gestión del conocimiento a nivel distribuido, ya que es necesario proveer información acerca de la funcionalidad, la disponibilidad y las interfaces de las distintas componentes; esta información ha de tener una interpretación que no sea ambigua, para así poder ser procesada automáticamente. El máximo exponente de esta nueva generación es la definición de la Arquitectura para Servicios Grids Abiertos (OGSA) (Foster, 2002) en la que se adopta una visión basada en servicios Web para el descubrimiento, la creación dinámica, la gestión y la invocación de servicios de *middleware* tal y como se habían definido en la segunda generación de entornos Grid. Para ello, es necesario: a) poder describir a los servicios ofertados de forma semántica y no sólo sintáctica como se realiza con WSDL (Christensen, 2001); y b) ser capaces de establecer clasificaciones complejas de los mismos, y no simples jerarquías como ocurre con UDDI (Bellwood, 2003). La descripción de servicios es fundamental para su búsqueda, selección, composición, operación, invocación, ejecución y monitorización. Las alternativas para proveer un determinado servicio dependen del valor de ciertos metadatos que tienen que ver con la

funcionalidad, el coste, la calidad de servicio, la situación geográfica y hasta con la identidad del creador del servicio.

En la actualidad existen diversos proyectos de computación global que persiguen integrar los aspectos semánticos. Entre ellos destacan: *myGrid* (Wroe, 2003), que utiliza ontologías para describir los servicios de bioinformática que ofrece y que orquesta; *Comb-eChen* (Frey, 2003), dedicado a la generación combinatoria de moléculas farmacológicas y en el que se describe semánticamente el conocimiento obtenido sobre la aplicabilidad de ciertas moléculas a determinadas enfermedades; *Geodise* (Chen, 2003) especializado en la optimización de problemas de ingeniería y en el que es posible anotar semánticamente los flujos de trabajo que dan como resultado una determinada optimización para un problema concreto de diseño; *coAKTinG* (Buckingham, 2002) que provee herramientas para la colaboración científica integrando espacios de reuniones inteligentes, involucrando ontologías para el dominio de aplicación, el contexto organizacional, la infraestructura de reuniones, y para los procesos y productos; y grids socio-cognitivas (Bruijn, 2003), en las que se plantea la integración de información semántica distribuida para el apoyo a actividades de carácter social y cognitivo (por ejemplo, para el acceso a la cultura digital).

Por su planteamiento, estos proyectos presentan información semántica aislada que no puede relacionarse entre sí fuera del ámbito de la aplicación para la que fueron diseñados. Esto impide que se provea en último lugar un acceso global y transparente a una rica colección de conocimiento heterogéneo y distribuido para dar apoyo a cualquier tipo de tarea de un usuario. Por ello, una de las principales contribuciones del presente trabajo, como detallaremos más adelante, será la definición de un servicio de nivel colectivo, aplicable a cualquier tipo de entorno de computación global, para la integración de repositorios de información semántica altamente distribuidos y con sistemas de persistencia heterogéneos.

4.3. La Evolución de la Web

La Web que conocemos actualmente tiene sus orígenes en los sistemas de hipertexto que fueron definidos por primera vez por Ted Nelson (Nelson, 1981) en 1960 como un modo de "escritura no secuencial", tomando como referencia las ideas publicadas en 1945 por Vannevar Bush en su conocida obra "As We May Think" (Bush, 1945) donde se describe una maquina conceptual ("memex") capaz de almacenar vastas cantidades de información.

Tomando como base todas estas ideas preliminares, Tim Berners-Lee concibió en el Laboratorio Europeo de Física de Partículas (CERN) a finales de los 80

los primeros prototipos de lo que hoy es comúnmente conocido como la World-Wide-Web (Berners-Lee, 1989). La Web se diseñó como un espacio de información cuyo objetivo era no sólo la comunicación entre personas, sino también la participación de entidades software que pudieran ser de ayuda en la realización de tareas por parte de aquéllas. Sin embargo, uno de los mayores obstáculos en el desarrollo de esta idea ha sido que la mayor parte de la información presente en la Web está diseñada para el consumo humano, prestándose atención principalmente a la visualización y los procesos de formateo de la misma para que sea comprensible y atractiva; sin embargo, está poco o nada estructurada semánticamente, a pesar de que la información proviene de fuentes con estructuras semánticas bien definidas. El resultado es un vasto depósito global de información del que pueden extraer poco provecho otros tipo de consumidores como son los robots o agentes software, dado que no se dispone actualmente de un sistema de información global (Web Universal) en el que la información almacenada esté estructurada de forma semántica y en el que el razonamiento automático sea ubicuo y proporcione apoyo a las tareas que los usuarios deseen realizar. Para resolver este problema, se están desarrollando grandes esfuerzos investigadores en varios campos, todos ellos englobados bajo el término general de Web Semántica y cuyo objetivo final es la formulación de una Web universal de aserciones semánticas. Para ello, y siguiendo el principio de diseño minimalista, se han realizando importantes progresos en la definición de un modelo común de gran generalidad y simplicidad denominado Resource Description Framework (RDF) (Lassila, 1999). Este modelo contiene únicamente los conceptos de aserción (para describir recursos) y de notas (aserciones sobre aserciones), de forma que al no poseer conectivas de negación ni de implicación, es fácil determinar, para cualquier pregunta, si existe una prueba en términos lógicos sin que se den problemas de intratabilidad. El minimalismo de este modelo, sin embargo, no permite definir nuevos tipos de aserciones (vocabularios u ontologías para describir recursos), ni restringir los tipos de aserciones que se pueden realizar sobre un determinado recurso, ni establecer jerarquías de herencia entre aserciones o tipos de recursos, ni tampoco saber si un determinado conjunto de aserciones satisfacen un conjunto de restricciones. Todas estas facilidades están siendo incorporadas a lo que se denominan esquemas RDF y en lenguajes como DAML+OIL (Horrocks, 2002) y OWL (Paetl-Schneider, 2003).

Desde un punto de vista arquitectónico, las componentes de la Web semántica se organizan en un modelo por niveles; como muestra la Figura 26, además de los ya citados existen niveles superiores para dar soporte a la inferencia lógica a partir de información semántica.

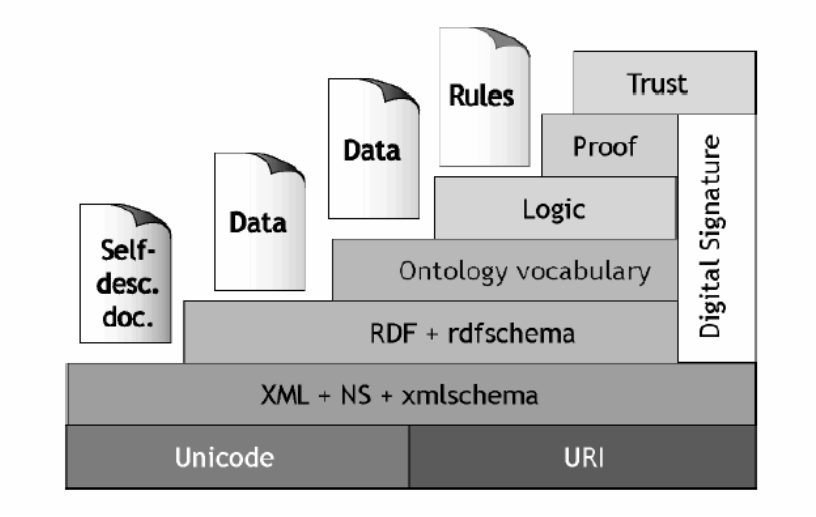


Figura 26 Modelo en Niveles para la Web Semántica

La investigación desarrollada hasta la fecha en relación con la Web semántica se ha centrado principalmente en los niveles inferiores de esta arquitectura, dando como resultado una gran profusión de proyectos relacionados con el acceso a información en la Web Semántica mediante plataformas que sean capaces de almacenar grafos RDF y de proveer dicha información mediante interfaces de aplicación que en muchos casos no son compatibles. Un estudio en profundidad de las distintas infraestructuras que permiten el almacenamiento y recuperación de información semántica (Magkanari, 2002) revela que existe una gran diversidad en los modelos utilizados para la representación de las ontologías, los modelos de datos para representar metadatos, los lenguajes de recuperación de datos, los lenguajes de implementación, las bases de datos sobre las que se almacenan dichos datos, etc. (Tabla 3).

Entorno	Lenguaje Query	Leng Impl	SGBD	API	Formato
ICS-RDFSuite	RQL	Java C++	ORDBMS (SQL3 compliant, e.g., PostgreSQL)	C++/Java/SQL	RDF
SESAME	RQL*	Java	ORDBMS (PostgreSQL)	http/SOAP	RDF
INKLING	SquishQL	Java	En memoria (supporting JDBC, e.g., SQL, Postgres SQL)	Java	Triples in ASCII
RDFDB	SquishQL*	C	Persistence (SleepyCat)	C, Perl	Triples in ASCII
RDFSTORE	SquishQL	C Perl	In-memory / persistence (e.g., file, BerkeleyDB, SDBM)	Perl	N-Triples, RDF
EOR	Triple-matching	Java	Persistence(SQL databases, e.g. MySQL)	HTTP, Java,SQL/JDBC	Triples rendered with XSL
REDLAND	Triple-matching	C	In memory / persistence (SleepyCat / BerkerleyDB)	Java, C, Perl, Python, Tcl	Triples
JENA	RDQL	Java	In-memory / persistence (e.g., BerkeleyDB, Interbase, PostgreSQL)	Java	Triples in ASCII
RDF GATEWAY	RDFQL	?	RDBMS	MicrosoftADO, JDBC	Triples in ASCII
TRIPLE	TRIPLE	Java	In-memory	Java	Lisp, XML for FaCT DTD, DOT, DAML, ASCII

KAON	F-logic	Java, Python	In-memory / persistence (KAON server/ Files, RDBMS)	Java	?
CEREBRA	DL-based	Java	Distributed data (CORBA)		?
Empolis k42	TMQL	Java	Persistence storage(K42 Generic Store, other DBMS)		Tepic Maps (XTM)
Ontopia KS	Tolog	Java	In-memory / RDBMS / OODB		XTM, XML version of ISO 13250

Tabla 3 Sistemas para la persistencia de información semántica RDF

Cabe, pues, plantearse si las tecnologías propuestas en el ámbito de la Web semántica son de aplicabilidad al ámbito de las aplicaciones de nueva generación de la computación global. Como señalan Goble y de Roure en (Goble, 2002), esto es altamente factible, llegando a afirmar que *"las aplicaciones de computación global pueden considerarse como aplicaciones de la Web Semántica"*; es decir, que se pueden utilizar las tecnologías propuestas en el ámbito de la Web semántica para implementar servicios de computación global "semánticos". Esto ha hecho que en los proyectos de computación global anteriormente citados se hayan utilizado alguna de las plataformas mostradas en la tabla anterior para definir y almacenar información semántica. Es de prever, por tanto, que en los próximos años esta tendencia se consolide y nos encontremos con un escenario en el que distintas aplicaciones de computación global hagan uso de distintas plataformas de la Web semántica y sea imposible hacer realidad el sueño de un acceso global y transparente a la información semántica, dada la heterogeneidad de formas en que los repositorios almacenan la información.

4.4. El Problema de la Integración de Repositorios Semánticos Distribuidos

Hemos constatado en las secciones anteriores que:

- En el futuro de la computación global nos encontramos ante el desafío de disponer de mecanismos que permitan la definición de repositorios semánticos siguiendo los principios de simplicidad, descentralización, modularidad e interoperabilidad.
- En el ámbito de la Web Semántica se están desarrollando modelos y lenguajes adecuados para la definición y manipulación de información de carácter semántico, y que dichas tecnologías son de aplicabilidad en el ámbito de la computación global.
- El fuerte crecimiento en el número de plataformas para el almacenamiento de información semántica ha dado como resultado una gran heterogeneidad de soluciones que no siempre son capaces de interoperar.
- Es previsible que dicha heterogeneidad se traslade a las distintas aplicaciones de computación global dando como resultado islas de información semántica que no pueden ser accedidas de forma transparente como si se tratase de un recurso de información semántica universal
- RDF es el estándar "de facto" para la representación de información semántica en la Web.

Por todo ello, en nuestra propuesta planteamos la necesidad de definir modelos e implementar mecanismos que permitan, en primer lugar, la integración de repositorios semánticos RDF heterogéneos y distribuidos, de forma que sea posible acceder a ellos de forma transparente, y en segundo lugar, interrelacionar información semántica que es gestionada descentralizadamente por múltiples propietarios.

El problema de la integración de repositorios de datos heterogéneos no es un problema reciente. Numerosos investigadores en el campo de los sistemas de información han trabajado durante más de 20 años en la búsqueda de soluciones aceptables y eficientes al problema de la integración de fuentes de datos heterogéneas. Una de las primeras soluciones aparece con la noción de federación en el campo de las bases de datos. Así se define en (Sheth, 1990) una base de datos federada como *una colección de componentes de sistemas de bases de datos cooperantes pero autónomos* en el sentido de que dichos componentes pueden diferir en aspectos tales como el modelo de datos y los lenguajes de consulta. Los principales problemas a resolver en estos sistemas son la integración de esquemas, la negociación, la descomposición y optimización de consultas y la gestión global de transacciones. Una revisión detallada de dichos sistemas se puede encontrar en (Sheth, 1990) pero en todos los casos ahí descritos se atiende a los modelos relacionales y orientados a objetos en los mecanismos de integración, los cuales no son adecuados para la integración de repositorios semánticos expresados siguiendo el modelo RDF.

Más recientemente, Arms define a una biblioteca digital federada como "un grupo de organizaciones que trabajan conjuntamente de manera formal o informal y que se ponen de acuerdo para dar soporte a un conjunto común de servicios y estándares para que sus miembros puedan interoperar" (Arms, 2000). De nuevo, en este caso se oculta la heterogeneidad para proveer información de manera integrada. Una discusión interesante sobre cómo resolver el problema de la interoperabilidad en bibliotecas digitales utilizando soluciones que van desde el uso de estándares hasta la participación de mediadores se puede encontrar en (Paepcke, 1998). Finalmente, cabe destacar también en este campo la propuesta de la Open Archives Initiative (OAI) (OAI, 2000) para obtener repositorios de datos interoperables por medio de un protocolo de recolección (*harvesting*) de metadatos.

En la actualidad existe un proyecto denominado EDUTELLA (Nejdl, 2002) que por primera vez plantea la integración de repositorios semánticos basados en descripciones RDF mediante la definición de una arquitectura P2P construida a partir del entorno JXTA (Brendon, 2002)]. Una red EDUTELLA conecta nodos semánticos heterogéneos haciendo la naturaleza de los nodos RDF individuales completamente transparente. Sin embargo, esta solución se

restringe a un determinado lenguaje de programación (Java) lo que hace difícil su integración con plataformas semánticas basadas en otros lenguajes. Además, en esta solución no se presta atención a aspectos de seguridad que son un requisito fundamental en entornos de computación global, ni es posible establecer relaciones semánticas entre recursos pertenecientes a diferentes repositorios semánticos.

4.5. Una Visión Integradora: Las Federaciones Semánticas

Nuestra propuesta de integración se enuncia de la siguiente forma:

Dado el problema de la integración de repositorios semánticos RDF distribuidos, definimos las federaciones semánticas entendidas como colecciones de repositorios RDF heterogéneos (esto es, multiplataforma, multilenguaje, y multipersistente) en los que sea posible la interconexión semántica de recursos definidos en diferentes repositorios, la navegación transparente de propiedades RDF distribuidas, el soporte a la replicación de elementos semánticos definidos en la federación, el mantenimiento de la consistencia frente a modificaciones descentralizadas de los contenidos de dichos repositorios, y la definición de un mecanismo de autorización y autentificación escalable para su uso en aplicaciones de computación global.

Puesto que nuestra propuesta de federación semántica está basada en el modelo RDF, que es el estándar de facto, cualquier esfuerzo para la definición formal del concepto de federación ha necesariamente de pasar por la formalización previa del concepto de repositorio RDF.

Formalización Repositorio Semantico

Un ***alfabeto semántico*** se define como la tupla $\sum=(\Re,\Upsilon,\wp,\Lambda)$ tal que

$\Re$ *es un conjunto de recursos ,*

Υ *es un conjunto de literales,*

$\wp$ *es un conjunto de propiedades y*

Λ es un conjunto de variables

Dado un alfabeto semántico $\sum=(\Re,\Upsilon,\wp,\Lambda)$ se define

el universo de aserciones sobre $\sum$, $C_\sum = \Re \times \Re\cup\Upsilon \times \wp$

el conjunto de términos válidos sobre $\sum$, $\Im_\sum$

$$\Im_{\Sigma} = (\Re \cup \Lambda) \times (\wp \cup \Lambda) \times (\Re \cup \Upsilon \cup \Lambda)$$

Una **colección RDF** se define como una tupla $\Omega = (\sum , C)$ tal que

$\sum$ *es un alfabeto semántico*

$C \subseteq C_{\Sigma}$

Un **Repositorio Semántico** se define como la tupla $I = (\Omega, m , M)$ /

$\Omega = (\sum, C)$ *es una colección RDF*

m es una función de matching $m: \Im_{\Sigma} \rightarrow 2^{C}$

$m(t_m,t_p,t_m) = \{(a,p,b) \in C / \exists$ *sustitución* $\sigma : (t_m,t_p,t_m)_{\sigma} = (a,p,b)\}$

M una colección de funciones de gestión

$M = \{\varphi_k : 2^{C_{\Sigma}} \times C_{\Sigma} \rightarrow 2^{C_{\Sigma}} \; k=1..T\}$

Y el universo de todos los posibles repositorios semánticos se denota como Ψ.

Ejemplos de funciones de gestión en M son

$$\varphi_{insert}(C_o,(n,p,m)) = C_o \cup \{(n,p,m)\} / \; (n ,p,m) \in C_{\Omega}$$

$$\varphi_{delete}(C_o,(n,p,m)) = C_o - \{(n,p,m)\} \; / \; (n,p,m) \in C_o$$

Las definiciones anteriores establecen repositorios RDF individuales. Sin embargo, tal y como mencionábamos en nuestros objetivos iniciales, para un entorno como la Web necesitamos un marco que permita la especificación de repositorios RDF distribuidos. Tal marco requiere, por un lado, mecanismos que permitan el acceso a información semántica interrelacionada independientemente de donde se encuentren los recursos RDF almacenados. De esta forma, estableciendo mecanismos de navegación para poder acceder de forma transparente a recursos semánticos, se evita la proliferación de islas semánticas.

Por otro lado, dado que será probable que las especificaciones RDF distribuidas sean propiedad de diversas instituciones o "*stakeholders*" que utilizarán distintos tipos de repositorios (Tabla 3), será necesario proporcionar mecanismos de interoperabilidad para que la navegación de dichos repositorios sea efectiva.

En nuestra propuesta, damos soporte a estos requisitos introduciendo el concepto de Federación Semántica tal y como definiremos de una manera formal a continuación y de una manera conceptual más adelante.

Formalización Federación Semántica

Dado un conjunto de alfabetos semánticos $\Gamma=\{\sum_i\}_{i=1..N}$ se define

el universo de aserciones sobre Γ , $C_\Gamma = \cup_{i=1..N} C_{\Sigma i}$

el conjunto de términos válidos sobre Γ, $\Im_\Gamma$

$$\Im_\Gamma = (\cup_{i=1..N} \Re_i \cup \Lambda) x (\cup_{i=1..N} \wp_i \cup \Lambda) x (\cup_{i=1..N} \aleph_i \cup \Lambda)$$

Una **Federación Semántica** se define como la tupla $F=(I_F,m_F,M_F)$ donde

I_F *es una colección de repositorios semánticos* $I_F=\{I_i\}_{i=1..N}$

m_F *es una función de matching federada* $m_F: \Im_\Gamma \rightarrow 2^{C_\Gamma}$

M_F *es una colección de funciones de gestión* $M_F = M_M \cup M_S /$

$M_M = \{\varphi^M_k: 2^\Psi \times \Psi \rightarrow 2^\Psi \quad k=1..T\}$

$M_S = \{\varphi^S_k: 2^\Psi \times C_\Phi \rightarrow 2^\Psi \quad k=1..T\}$

Dada una federación semántica *F, una* ***aserción distribuida*** *es una aserción* $(n,p,m) \in C_\Gamma \; / \; \exists\, i, j=1..N / \; n \in \Re_i - \{\iota_{empty}\}$ *and* $m \in \aleph_j \quad i \neq j$ y dicha aserción se representa en los alfabetos de los repositories locales como sigue:

(n,p,m_1), $(m_1,\text{remote},\text{uri}_{ji}(m)) \in C_i$ y

$(m, \text{is\_linked\_by}, \text{uri}_{ij}(n)) \in C_j \; / \; \text{uri}_{km}: \aleph_k \rightarrow \Re_m$.

El conjunto de estas **aserciones locales de distribución** se denota como $C_{ij}^D \subseteq C_i$ y $C_i^D = \cup_{j=1..N} C_{ij}^D$

Dada una federación semántica F el conjunto de aserciones distribuidas en F se denota como D_F

Para un determinado término $(t_n,t_p,t_m) \in \Im_\Gamma$

$m(t_n,t_p,t_m) = m_{local}(t_n,t_p,t_m) \cup m_{dist}(t_n,t_p,t_m) \; /$

$m_{loca:} : \Im_\Gamma \rightarrow 2^{C_\Gamma} /$

$m_{local}(t_n,t_p,t_m) = (\cup_{i=1..N} m_i(t_n,t_p,t_m) \; / \; (t_n,t_p,t_m) \in \Im_{\Omega i}) - \cup_{i=1..N} C_i^D$

$m_{dist}: \Im_\Phi \rightarrow 2^{C_\Phi} /$

$m_{dist}(t_n,t_p,t_m) = \{(a,p,b) \in D_F : \; (t_n,t_p,t_m)_\sigma = (a,p,b)\}$

Ejemplos de funciones de gestión en M_M son:

$\varphi^M_{add\_member}(I_F,I) = I_F \cup \{I\}$

$\varphi^M_{remove\_member}(I_F,I_k) = \cup_{i=1..N} I_i^* - I_k \quad i \neq k \; I_k \in I_F \; /$

$$I_i^* = (\Omega_i^*, m_i, M_i, \Lambda_i),$$

$$\Omega_i^* = (\Re_i, \Upsilon_i, \wp_i, C_i^*, \Lambda_i),$$

$$C_i^* = C_i - \{(a,p,b) \in C_{ik}^D\} \ \forall I_i \in I_F$$

Un ejemplo de funcion en M_S es

$$\varphi_{insert}(I_F,(n,p,m)) = \cup_{i=1..N} I_i^*, I_i^* = (\Omega_i^*, m_i, M_i), \Omega_i^* = (\Re_i, \Upsilon_i, \wp_i, C_i^*)/$$

$$si\ (n,p,m) \in C_{\Sigma i}\ \ C_i^* = \varphi_{insert}(C_i,(n,p,m))$$

$$si\ (n,p,m) \in D_F\ y\ n \in \Re_i\ \ ,m \in \aleph_j\ i \neq j$$

$$C_i^* = Ci \cup \{(n,p,m_{new})(m_{new}, remote, uri_{ji}(m))\},$$

$$si\ (n,p,m) \in D_F\ y\ n \in \Re_j\ \ ,m \in \aleph_i\ i \neq j$$

$$C_i^* = C_i \cup \{\ (m,\ is\_linked\_by,\ uri_{ji}(n))$$

$$en\ otro\ caso\ C_i^* = C_i$$

Modelo Conceptual Federación Semántica

Las Federaciones Semánticas tal y como se han presentado se definen según el siguiente modelo conceptual simplificado (Figura 27, subconjunto de servicios incluidos)

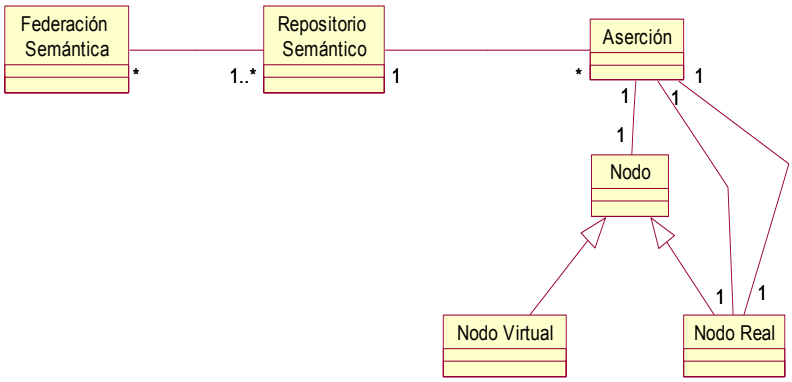


Figura 27 Modelo Conceptual Simplificado de Federación Semántica

y en el que destaca, como hemos definido de manera formal, la introducción de aserciones distribuidas mediante un mecanismo de nodos virtuales que se comportan como proxies y representan a recursos RDF remotos. Un nodo real representa a un recurso web único en la colección federada mientras que un

nodo virtual representa a un nodo real almacenado en un repositorio semántico remoto para poder navegar las colecciones de forma distribuida. Los nodos virtuales son para consumo interno por parte de la federación y un cliente externo no puede acceder a ellos directamente. Un nodo real puede tener muchos nodos virtuales que le apuntan desde diferentes repositorios remotos. Para distinguir entre nodos reales y nodos virtuales proponemos la introducción de dos propiedades RDF especificas denominadas: "*remote*" e "*is_linked_by*" (Figura 28). El sujeto de una aserción que posea una propiedad "remote" es un recurso virtual, y el objeto de dicha aserción indica los repositorios semánticos de la federación donde se ha almacenado el nodo real (o cualquiera de sus réplicas). Por otro lado, el sujeto de una aserción cuya propiedad sea "is_linked_by" es un recurso real y el objeto de dicha aserción apunta al repositorio semántico de la federación donde se almacena el nodo virtual que le apunta. Ambos tipos de aserciones son necesarias para poder proveer navegabilidad bidireccional entre los repositorios y para el mantenimiento de la consistencia de las aserciones distribuidas.

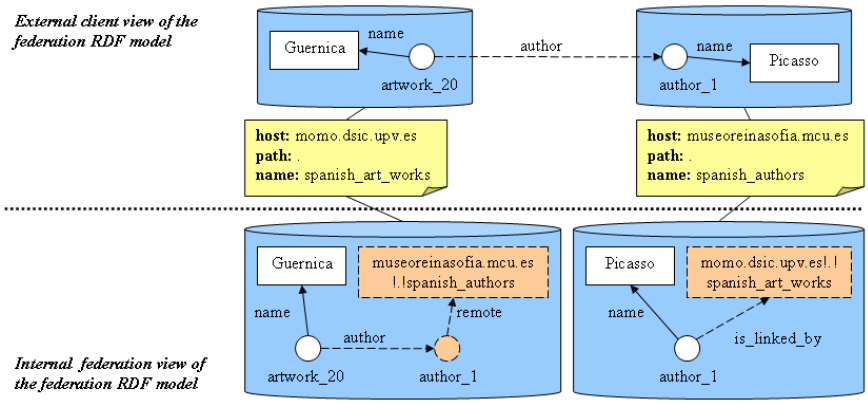


Figura 28 Vistas interna y externa de una aserción RDF distribuida

Para ilustrar nuestra propuesta, supongamos que usamos nuestro modelo de aserciones distribuidas para definir una infraestructura de metadatos sobre obras de arte que dé soporte a una federación de diferentes museos. Dicha infraestructura de metadatos estaría compuesta por varios repositorios semánticos que contendrían información sobre obras de arte y sus autores que están interrelacionados por medio de aserciones distribuidas. En este caso podríamos tener un recurso real denominado "author_1" en el repositorio "spanish_authors" localizado en el nodo "museoreinasofia.mcu.es" y otro recurso denominado "artwork_20" en el repositorio "spanish_art_works" en el servidor "momo.dsic.upv.es" (Figura 4). Si quisieramos definir una propiedad "author" indicando que el recurso "author_1" es el autor del recurso "artwork_20" tendríamos que definir un recurso virtual en el repositorio

"spanish_art_works" con una propiedad "remote" para indicar en qué repositorio remoto esta el recurso real asociado. El nodo objeto de dicha aserción identificará de forma unívoca, pues, a dicho repositorio. Adicionalmente, una propiedad de tipo "is_linked_by" se define en el nodo real para indicar que existe un nodo virtual en el repositorio "Spanish_art_works" apuntándole. Al final de este proceso se ha podido definir una propiedad para el recurso remoto "artwork_20" pero la federación oculta las aserciones internas de forma que una aplicación externa pueda navegar dicha propiedad distribuida sin tener que preocuparse por su ubicación real.

El uso de aserciones distribuidas implica ciertos riesgos, debido a los posibles fallos en el funcionamiento de ciertos repositorios semánticos que los hagan inaccesibles. En estas condiciones, aunque una colección RDF federada debe poder ser recorrida de forma transparente como si se tratase de una colección única, es posible que en un momento dado algún subconjunto de dicha federación no se pueda recorrer. Para dar solución a este grave problema proponemos también en este capítulo un mecanismo de replicación efectivo a partir de los conceptos propuestos anteriormente. En concreto, proponemos el uso de la propiedad "remote" para que un nodo virtual pueda ser sujeto de múltiples aserciones, de manera que cada una indique la existencia de una réplica de un recurso real. Dichas réplicas han de existir en repositorios diferentes, de forma que si durante la navegación de una aserción distribuida se produce un error en el acceso al repositorio donde el nodo real se encuentra ubicado, podemos automáticamente buscar una réplica accesible sin más que inspeccionar los nodos objeto de las propiedades "remote" del nodo virtual que se pretendía inicialmente atravesar.

Desde el punto de vista arquitectónico para aplicaciones de computación global que se mostró en la Figura 25 podemos ver que las federaciones semánticas han de ofrecer servicios de nivel colectivo para: Gestión de miembros de una federación, Gestión de modelos RDF federados, Gestión de consultas federadas con aserciones distribuidas, Gestión de replicas y Gestión de la consistencia de aserciones distribuidas. En cambio, los repositorios semánticos han de ofrecer servicios de nivel de recurso para: Gestión modelos RDF locales y Gestión de consultas locales

Todos estos servicios serán el objeto de investigación del presente capítulo tomando como referencia el modelo descrito anteriormente.

4.6. Modelo Arquitectónico de Federación Semántica

Una vez presentado el modelo conceptual de Federación Semántica, tenemos que tomar en consideración las alternativas de diseño arquitectónico e implementación para el mismo. Los principales objetivos de diseño son, en primer lugar, permitir el acceso remoto a los repositorios semánticos y, en segundo lugar, resolver el problema de la heterogeneidad que surge cuando se utilizan diferentes tecnologías en cada repositorio. Por lo tanto, se necesita un mecanismo de comunicación que permita acceder a colecciones RDF independientemente de su implementación y de su ubicación. Este requisito, afortunadamente, no es específico o exclusivo de las federaciones semánticas, sino que es considerado como fundamental en el contexto de la World Wide Web, donde se ha considerado la incorporación de servicios que puedan ser publicados, encontrados y consumidos por terceras partes de una forma estándar para poder explotar así completamente las capacidades que Internet proporciona para la computación distribuida. Como consecuencia, se han propuesto un gran número de tecnologías clave alrededor del concepto de *Servicio Web* (WS) (Booth, 2003). Éstas están basadas en estándares abiertos, están apoyadas por un gran número de organizaciones y empresas del sector de la computación y son independientes de la plataforma y del lenguaje de programación que se adopte para el desarrollo de una aplicación. Así pues, lo que se pretende como fin último es hacer interoperar a elementos software que hayan sido implementados en diferentes lenguajes de programación y sobre plataformas diversas. Un WS puede ser definido como una entidad software cuya funcionalidad es accesible (invocada) a través de un protocolo estándar denominado Simple Object Access Protocol (SOAP) (Gudgin, 2003), que es descrito mediante un documento escrito en lenguaje Web Service Description Language (WSDL) (Christensen, 2001) y que es publicado en un directorio UDDI (Bellwood, 2003). No adoptar estas tecnologías que son estándares de facto para el acceso remoto e interoperabilidad con los SW sería un grave error por nuestra parte. Además, si consideramos el modelo arquitectónico de computación global en el que las federaciones semánticas son servicios de nivel colectivo, y queremos ser acordes con las directrices marcadas en este campo para la definición de arquitecturas de servicios grid abiertas (OGSA) (Foster, 2002), nos vemos también obligados a utilizar los WS como tecnología que esté en el núcleo de nuestro diseño.

Por todo lo anteriormente expuesto, presentaremos a continuación un modelo arquitectónico para las federaciones semánticas entendido como una infraestructura de WS que soportará el acceso remoto a implementaciones heterogéneas de repositorios RDF. Explicaremos cómo los detalles de implementación se hacen transparentes para la federación semántica, y cómo

una federación semántica tal y como fue definida anteriormente puede proveer un acceso homogéneo a una colección de repositorios RDF ocultando la estructura arquitectónica de los repositorios distribuidos.

4.6.1. Nivel de Recurso: Repositorios Semánticos RDF basados en WS

Las colecciones semánticas basadas en el modelo RDF se almacenan en repositorios que proveen servicios para la gestión de recursos RDF, la persistencia y la navegación. En nuestro contexto hemos de definir no sólo una interfaz adecuada que permita la navegación por parte de entidades software de modelos RDF almacenados en repositorios heterogéneos, sino también un mecanismo que permita la adecuada gestión de los modelos RDF de forma que cualquier entidad externa autorizada pueda manipular dichos modelos sin tener que preocuparse por los detalles de implementación. Esto significa en términos de WS la definición de una interfaz descrita en WSDL y que sea considerada como un estándar por parte del organismo WWW Consortium. Dado que dicha interfaz no ha sido todavía definida a nivel estándar hemos adoptado el modelo propuesto en Redland (Beckett, 2003) como un candidato válido de interfaz para alcanzar nuestros objetivos, dado que este modelo se considera como el estándar *de facto* y porque, como veremos, soporta los mecanismos de navegación adecuados para la exploración de recursos RDF.

El Modelo Redland

Redland es un entorno flexible que proporciona una interfaz de alto nivel para almacenar, manipular y consultar modelos RDF. Implementa todos los niveles inferiores de la arquitectura que se muestra en la Figura 26. Dicha interfaz permite manipular especificaciones RDF sin tener en cuenta detalles de bajo nivel como por ejemplo la representación en XML de una especificación RDF. Las principales características que lo definen son: su diseño modular, que facilita su mantenimiento y optimiza su comportamiento, sus interfaces de alto nivel estables que permiten su integración con otras aplicaciones, y su portabilidad.

Las principales clases que ofrecen soporte a la gestión de modelos son:

- **World:** para la gestión de todas las funciones de inicialización y terminación.
- **Node:** para representar nodos tal y como son definidos en (Lassila, 1999), incluyendo recursos para representar a recursos web identificados por URIs, literales para representar valores, blancos como nodos auxiliares y nodos li para definir colecciones de propiedades.

- **Statement:** para representar arcos según se definen en (Lassila, 1999). Una aserción se compone de tres nodos que representan el sujeto, el predicado y el objeto de una propiedad en una especificación RDF, o usando una terminología diferente, el origen, el arco y el destino respectivamente.
- **Model:** para representar un conjunto de aserciones que normalmente se almacenan en un repositorio. Proporciona la interfaz principal para el desarrollo de aplicaciones en Redland.
- **Storage:** Abstrae los modelos físicos de almacenamiento que pueden ser persistentes en una base de datos o residir simplemente en memoria.

En nuestro caso, la interfaz que definirá la funcionalidad permitida a nivel de recurso es la especificada en la clase *Model*. Dicha interfaz provee los siguientes mecanismos para la navegación de modelos RDF:

- find_statements(), que devuelve todas las aserciones que satisfacen una determinada sustitución de las variables por términos básicos
- get_sources(), que obtiene todos los nodos origen (sujetos) que poseen un determinado predicado y objeto.
- get_arcs(), que obtiene los predicados de las aserciones para un sujeto y objeto dados.
- get_arcs_out(), especialización del anterior cuando se trata de un nodo sujeto
- get_arcs_in(), especialización cuando se trata de un nodo objeto
- get_targets(), obtiene los nodos objeto para un sujeto y predicado dados
- contains_statement(), valida la existencia de una aserción específica
- has_arc_in() y has_arc_out(), validan la existencia de una propiedad

Por otro lado, para la gestión de modelos RDF dicha interfaz provee los siguientes servicios:

- add_statement(), para insertar una aserción
- remove_statement(), para eliminar una aserción

Acceso Remoto a Repositorios Semánticos Mediante WS

La interfaz que acabamos de presentar es la base de todo servicio de nivel de recurso que encapsule a cualquier tipo de repositorio RDF que forme parte de una federación. Dado que todos los repositorios deben exponer la misma interfaz, los tipos de los argumentos utilizados en los distintos servicios han de ser forzosamente los mismos (compatibilidad de tipos). La colección de tipos

comunes ha sido definida como un conjunto de clases serializables y son un componente esencial cuando hablamos de interoperabilidad entre implementaciones de repositorios semánticos heterogéneos.

En definitiva, la aplicación de WS expone los métodos para gestionar y navegar una colección RDF de forma que cualquier aplicación remota puede acceder a sus contenidos. La Figura 29 ilustra el nivel de recurso de nuestra arquitectura, en el que el WS constituye la principal interfaz y donde las clases *SerializableStatement*, *SerializableNode* y *SerializableUri* son los tipos comunes utilizados para intercambiar información entre el WS y las aplicaciones cliente.

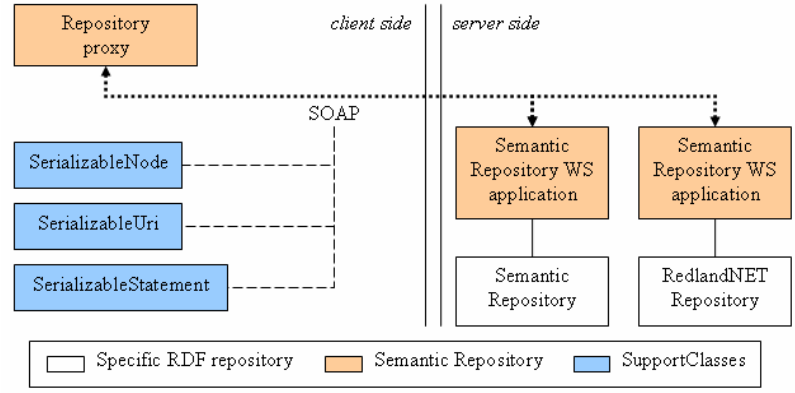


Figura 29 Nivel de Recurso para acceso remoto a Repositorios RDF

Con esta arquitectura de WS facilitamos la adaptación de cualquier repositorio RDF existente para hacerlo accesible mediante protocolos estándares. Algorítmicamente, cada WS realiza las siguientes tareas:

- Abre el repositorio que da persistencia a las aserciones del modelo
- Carga el modelo
- Deserializa los argumentos de entrada y crea las instancias adecuadas para su manipulación por los niveles inferiores
- Invoca el servicio correspondiente del repositorio
- Serializa los valores de retorno de acuerdo a las clases serializables definidas
- Libera los recursos utilizados y cierra el modelo
- Cierra el repositorio,
- Devuelve la instancia serializada.

4.6.2. Nivel Colectivo: Federaciones Semánticas basadas en WS

Una vez definido un mecanismo estándar para acceder a repositorios RDF heterogéneos, estamos en condiciones de considerar la integración de islas de información semántica de forma que constituyan una red distribuida de conocimiento. Por supuesto, abordaremos esta integración mediante el modelo de federación semántica definido anteriormente que permite el uso de aserciones distribuidas para poder recorrer colecciones heterogéneas de repositorios RDF distribuidos como si se tratase de un repositorio único y local.

Como hemos descrito anteriormente, a nivel de recurso cada repositorio es un WS. Cada uno actúa como si fuera un miembro de una federación y es identificado en la misma mediante tres parámetros: la URI donde se encuentra el WS que representa al repositorio, el *path* que determina dónde se encuentra físicamente ubicado dicho repositorio en el servidor donde se ejecuta el WS y el nombre del repositorio que contiene la colección RDF. Para ocultar esta arquitectura interna de repositorios la federación debe proveer la ilusión de que existe un único repositorio RDF. Para conseguir esto se ha de definir, de nuevo, una interfaz estándar que abstraiga los detalles de la infraestructura interna y provea los mismos servicios que ofrecería un repositorio RDF individual. Dado que esta interfaz ya existe y fue definida a nivel de recurso, la hemos reutilizado de forma que podamos utilizar las federaciones semánticas como un mecanismo escalable de acceso a múltiples repositorios RDF individuales. La Figura 30 muestra el esquema conceptual de una federación semántica atendiendo al paralelismo existente entre las interfaces que se proveen a nivel de recurso y las que se proveen a nivel colectivo, y atendiendo también a la posibilidad de que un repositorio semántico RDF pueda ser ubicado en el mismo servidor donde reside la federación o en uno remoto.

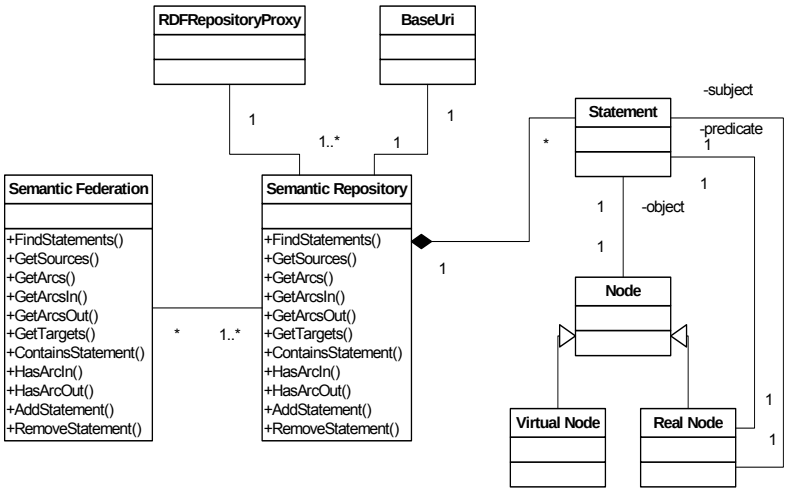


Figura 30 Modelo Conceptual Arquitectónico Federación Semántica

Las clases involucradas en dicho modelo conceptual son:

- RDFRepositoryProxy: oculta la complejidad de la serialización en el acceso a un repositorio semántico remoto.
- BaseUri: encapsula la información que identifica a un repositorio semántico en una federación.
- SemanticRepository: contiene una instancia de RepositoryProxy y abstrae y simplifica la información que es necesaria para invocar de forma correcta los servicios de una instancia de dicha clase. Como resultado, los servicios del proxy se exhiben como si fueran servicios de un repositorio local. Adicionalmente, más de una instancia de la clase SemanticRepository pueden compartir el mismo RepositoryProxy si se da el caso en el que todas ellas interactúan con el mismo WS de la federación semántica.
- Semantic Federation: Contiene varias instancias de la clase SemanticRepository que representan repositorios que pueden existir en el mismo servidor que la federación o en uno remoto.

Sin embargo, pese al paralelismo entre las interfaces de nivel colectivo y nivel de recurso, es obvio que una federación semántica ha de ofrecer servicios adicionales que no existen en un repositorio RDF aislado, como son: Gestión de miembros de la federación, Gestión de colecciones RDF Federadas y Navegación Federada. A continuación pasaremos a describir dichos servicios con mayor nivel de detalle.

Gestión de Miembros de la Federación

Los repositorios semánticos de nivel de recurso pueden ser añadidos o eliminados de una federación semántica. En la Figura 31 ilustramos este proceso con un ejemplo de federación que contiene cuatro miembros, uno de ellos ubicado en el mismo lugar que la federación, otros dos (r2 y r3) ubicados en *issi.dsic.upv.es* y el repositorio restante (r4) en *momo.dsic.upv.es*.

Cuando un repositorio es añadido, se puede hacer en modo escritura o sólo lectura, de forma que varias federaciones puedan compartir un determinado miembro.

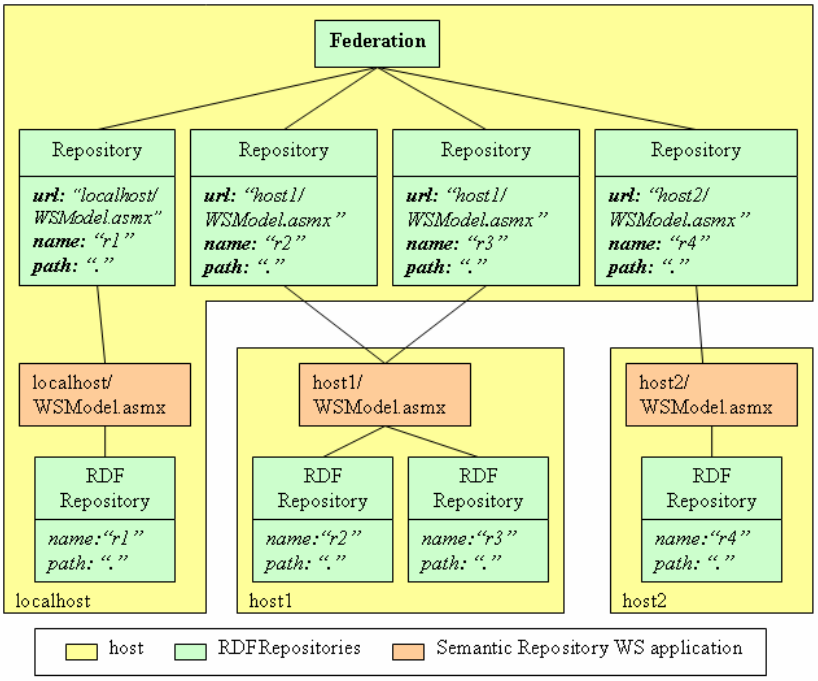


Figura 31 Miembros de una Federación Semántica

Gestión de Colecciones RDF Federadas

Los dos servicios principales considerados para la gestión de colecciones RDF federadas son *AddStatement* y *RemoveStatement,* ubicados en nuestro modelo en la clase *SemanticFederation*. Éstos son ejemplos interesantes de servicios de gestión distribuidos y, por ello, presentaremos sus singularidades a continuación.

Servicio AddStatement

Este servicio permite la creación de aserciones en una colección RDF federada, y requiere como argumentos tres nodos que son el sujeto, el predicado y el

objeto respectivamente identificados por una instancia de la clase *BaseUri*. Desde un punto de vista interno, la nueva aserción puede ser distribuida o local. El cliente de la federación puede indicar la localización de cada nodo mediante las instancias de *BaseUri* que acompaña. Por ello, se ha desarrollado un algoritmo para la inserción de aserciones distribuidas en la federación. Dicho algoritmo hace uso de los siguientes servicios asociados a la clase *SemanticRepository*:

- IsReal(node), que determina si un nodo es real en un determinado repositorio RDF.
- IsVirtual(node, BaseUri), que determina si un nodo es virtual en un repositorio apuntando a otro repositorio que es identificado por la instancia de BaseUri proporcionada.

```
AddStatement(subject, predicate, object)
        source_member = SFederation.GetMember(subject.BaseUri);
        target_member = SFederation.GetMember(object.BaseUri);
        if (source_member==null) or (target_member==null) then
                error;
        end if
        if not(source_member.IsReal(subject)) then
                error;
        end if
        if not(target_member.IsReal(object)) then
                error;
        end if
        if (source_member == target_member) then
          // local statement
          source_member.AddStatement(subject, predicate, object);
        else
          // distributed statement
          if not(target_member.IsReal(object)) then
            target_member.AddStatement(object, “is_linked_by”,
                                        source_member.BaseUri);
          end if
```

```
            if not(source_member.IsVirtual(object, target_member.BaseUri))
          then
              source_member.AddStatement(object, “remote”,
  target_member.BaseUri);
  end if
  source_member.AddStatement(subject, predicate, object);
  end if
end AddStatement
```

Adicionalmente, para definir una replica de un nodo real se opera del mismo modo que cuando se inserta la aserción distribuida anterior, pero en esta caso la instancia de *BaseUri* define la ubicación de la replica en otro repositorio.

Servicio RemoveStatement

Este servicio permite la eliminación de aserciones en una colección RDF federada y requiere los mismos argumentos que en el caso anterior. Sin embargo, el proceso de borrado ha de tener en cuenta el hecho de que cuando se elimina la última propiedad asociada a un determinado nodo o recurso, éste ha de ser también eliminado. Si se trata de una aserción distribuida todas las aserciones auxiliares (*remote* e *is_linked_by*) han de ser también eliminadas para mantener una federación consistente y, además, la aserción que contiene el nodo virtual ha de ser también eliminada si no existen más réplicas.

También se ha diseñado un algoritmo para realizar esta operacion a partir de los servicios:

- CountRealArcsIn(node), que cuenta el numero de aserciones locales que tienen como objeto el nodo especificado
- CountRealArcsOut(node), que cuenta el numero de aserciones locales que tienen como sujeto el nodo especificado

```
RemoveStatement(subject, predicate, object)
        source_member = SFederation.GetMember(subject.BaseUri);
        target_member = SFederation.GetMember(object.BaseUri)
        if (source_member==null) or (target_member==null) then
                error;
        end if
        if not(source_member.IsReal(subject)) then
```

```
                    error;
          end if
          if not(target_member.IsReal(object)) then
           error;
          end if
          if (source_member == target_member) then
            // local statement
            source_member.RemoveStatement(subject, predicate, object);
          else
          // distributed statement we process the source node
if (source_member.CountRealArcsIn(subject)==0)
and (source_member.CountRealArcsOut(subject)==1) then
  local_member = source_member;
  for each remote_member in RemoteRepositories(subject) do
  remote_member.RemoveStatement(subject, “remote”, local.BaseUri);
  if not(remote_member.Contains(subject, “remote”, ?)) then
          // there is no replication for the subject node
   remote_member.RemoveStatement(?, ?, subject);
end if
                      local_member.RemoveStatement(subject, “is_linked_by”,
                                                    remote_member.BaseUri);
                    end for each
                    end if
                    // we process the target node
if (target_member.CountRealArcsIn(object)==1)
and (target_member.CountRealArcsOut(object)==0) then
  local_member = target_member;
  for each remote_member in RemoteRepositories(object) do
          remote_member.RemoveStatement(object, “remote”, local.BaseUri);
          if not(remote_member.Contains(object, “remote”, ?)) then
           // there is no replication for the object node
            remote_member.RemoveStatement(?,?,object);
```

```
        end if
        local_member.RemoveStatement(object, “is_linked_by”,
                                        remote_member.BaseUri);
end for each
end if
source_member.RemoveStatement(subject, predicate, object);
end if
end RemoveStatement
```

Navegación Federada

La federación semántica objeto de nuestro trabajo proporciona también servicios de navegación transparente de la colección RDF subyacente. Así como los servicios de gestión necesitan saber la ubicación de los nodos pasados como argumentos, los servicios de navegación no necesitan esta información puesto que la navegación es ubicua. Aún así, el servicio retorna dicha información junto a las aserciones, de forma que las aplicaciones clientes puedan decidir si dicha información les es de utilidad. Este servicio permite la navegación de aserciones distribuidas. Además, también se ha implementado un servicio de consultas federadas que ejecuta el mismo servicio en todos los miembros de la federación y une los distintos resultados. En este proceso se filtran las aserciones internas que son utilizadas para la gestión de aserciones distribuidas y de nodos replicados de manera que todas las aserciones retornadas contengan únicamente nodos reales.

4.7. Evaluación Empírica

4.7.1. Implementacion

Una vez introducido el modelo de federación tanto en sus dos niveles: recurso y colectivo y los servicios que se ofrecerán mediante WS hemos de elegir la tecnología que sea más adecuada para su implementación. De entre las alternativas existentes hemos tomado el entorno .NET dado que simplifica enormemente las tareas de desarrollo de aplicaciones altamente distribuidas basadas en WS. El entorno .NET (Microsoft, 2003)] tiene dos componentes principales: el Common Language Runtime (CLR) y la biblioteca de clases .NET. El CLR actúa como un mediador o máquina virtual para conseguir una verdadera independencia de la plataforma de forma que cualquier aplicación en este entorno pueda ser ejecutada en cualquier lugar. La biblioteca de clases es

una colección de tipos reutilizables que están altamente integrados con el CLR, es orientada a objetos y, por tanto, permite derivar nuevas funcionalidad a partir de los tipos existentes. De este modo, el entorno .NET provee un sistema de ejecución manejado, simplifica el desarrollo y la implantación y se integra con una gran cantidad de lenguajes de programación. La interoperabilidad entre entidades software escritas en diferentes lenguajes de programación se consigue mediante la creación de ensamblados. Los ensamblados son los bloques básicos a partir de los cuales se generan aplicaciones .NET, se realiza el control de versiones, se reutiliza código multi-lenguaje y se definen mecanismos de seguridad. Simplificando, contienen el código que el CLR es capaz de ejecutar.

Las grandes ventajas que este entorno proporciona de cara a la interoperabilidad entre elementos software nos ha convencido sobre su utilización para obtener una versión ejecutable de nuestro modelo sobre la plataforma .NET. Nuestro objetivo inicial de cara a la implementación ha sido la obtención de un ensamblado .NET que contuviera toda la funcionalidad de nivel de recurso y que hemos denominado RedlandNET. Las clases obtenidas nos proporcionan los beneficios inherentes de .NET de forma que pueden ser utilizada desde cualquier lenguaje de programación compatible tales como C#, visualBasic, F#, C++ manejado, etc.

Adicionalmente, otra ventaja que proporciona .NET es que da soporte a la generación de WS y a todos los protocolos estándares mencionados con anterioridad, para poder finalmente ofrecer dentro del Nivel de Recurso un WS que proporcione todas las funciones de navegación y de gestión de repositorios RDF individuales.

Una vez descritos los niveles de recurso y colectivo y los detalles arquitectónicos de los mismos, hemos evaluado la eficiencia de las operaciones de inserción y de consulta para las tres infraestructuras semánticas descritas anteriormente: el repositorio Redland original (*Redland*), la infraestructura de nivel de recurso compatible con .NET (*RedlandNET*) y finalmente el WS de nivel colectivo que representa a la federación y accede a los WS de los miembros de la federación (*DRedlandNet*).

Para la realización de los experimentos se ha seguido el siguiente esquema genérico:

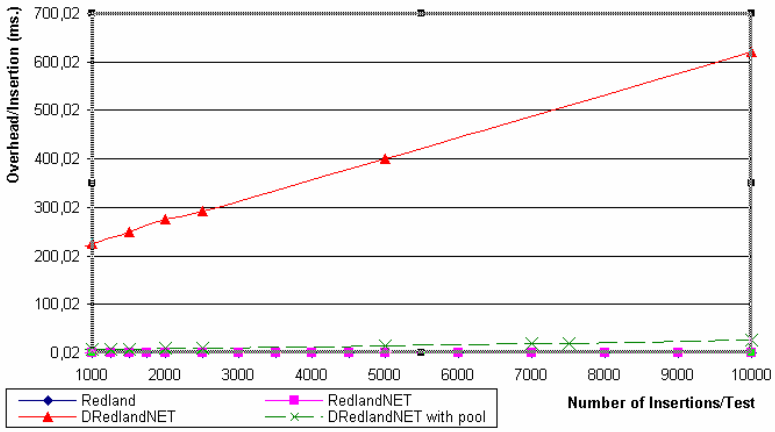
- Operaciones de inicialización
- Apertura del repositorio y del modelo RDF
- Inicialización de los argumentos de la operación
- Registro del tiempo de inicio

- Repetición de la operación
- Registro del tiempo de finalización
- Cálculo del overhead/operación en milisegundos
- Cierre del modelo RDF y la conexión al repositorio
- Liberación de recursos

En el caso del experimento DredlandNET las operaciones 1,2,8 y 9 pueden obviarse ya que éstas son realizadas por el WS contenedor.

4.7.2. Inserción de aserciones

Los resultados (Figura 32) muestran que el coste en todos los casos es muy similar excepto en el caso de la inserción usando el WS. Esto se debe a que los modelos en memoria no se reutilizan en cada invocación de la operación y deben ser cada vez creados, inicializados y finalmente destruidos. Para resolver este problema se simuló una caché de modelos y se observó una reducción del tiempo de inserción significativa.



.

Figura 32 Comparativa inserción de una aserción para Redland, RedlandNET, DRedlandNET and DRedlandNET con caché de modelos.

4.7.3. Consulta de aserciones

Para la consulta de aserciones se realizaron diversos experimentos de los cuales mostramos aquí aquellos en los que se varía el tamaño del repositorio. Como se puede observar, la versión RedlandNET solo incrementa en un pequeño

factor constante el coste de la operación debido a que en la versión de código manejado han de crearse los envoltorios adecuados para las estructuras de código no manejado de Redland. Finalmente, sí que se observa un preocupante incremento lineal del coste en la versión DRedlandNET. Tras un estudio de dicho problema, se detectó que era debido a una operación (*librdf_stream_get_object*) que en la infraestructura de Redland no está correctamente implementada.

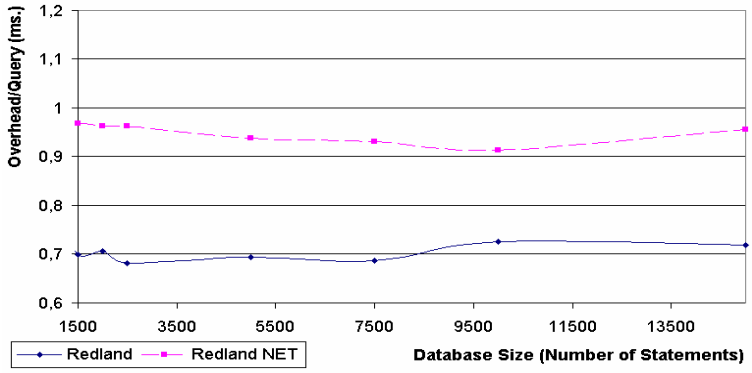


Figura 33 Comparativa número constante de consultas para Redland y RedlandNET.

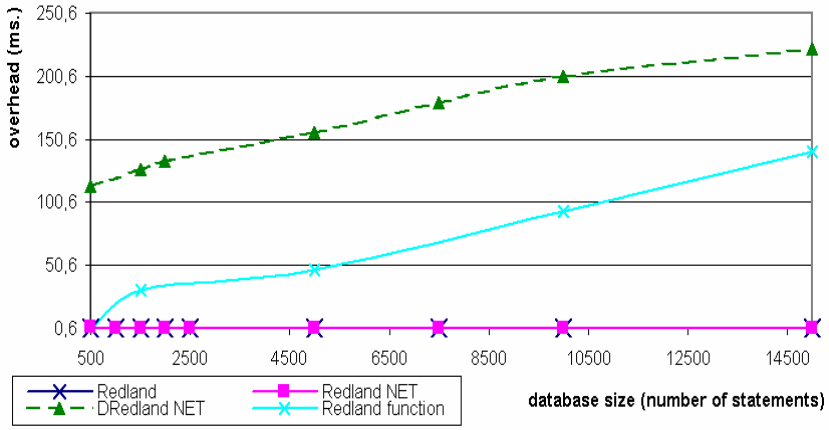


Figura 34 Comparativa número constante de consultas para Redland, RedlandNET y DRedlandNET.

4.8. Federaciones Semánticas en Museos Híbridos

Los museos híbridos son un claro ejemplo de infraestructura de computación global con información de carácter semántico altamente distribuido. En un Museo Híbrido se puede enriquecer la experiencia cultural del visitante integrando elementos explicativos que pertenezcan a diferentes instituciones y que se encuentren almacenados en repositorios gestionados de forma individual por diferentes *stakeholders*.

Las federaciones semánticas, tal y como se han definido anteriormente, son un claro instrumento que contribuye a la realización de esta visión integradora. Sin embargo, a pesar de tener un mecanismo de acceso homogéneo y de navegación transparente que permite el acceso a repositorios semánticos distribuidos, no es posible conseguir una interoperabilidad real si no se definen mecanismos de homogeneización ontológica que permitan que diferentes instituciones cuyos conjuntos de metadatos son normalmente dispares puedan integrar a nivel ontológico la información que tienen en sus repositorios. En nuestro caso, como ya vimos en el capítulo anterior, esta problemática de integración ontológica ha sido ya resuelta gracias al modelo CRM y a la expresión del modelo MoMo en términos de aquél. Tomando en consideración estas correspondencias semánticas, podemos definir un Museo Híbrido Federado en el contexto de MoMo como una Federación semántica de repositorios RDF que almacenan contenidos museísticos expresados según el modelo MoMo-CRM. El acceso a la infraestructura semántica distribuida subyacente se realiza mediante los servicios descritos en el presente capítulo. Dichos servicios de distribución semántica son utilizados por los servicios de mayor nivel de abstracción (el servicio navegacional y el de interacción social, respectivamente) que presentan la infraestructura de información distribuida de forma integrada a las aplicaciones externas y que fueron presentados en el capítulo anterior.

4.9. Conclusiones

En este capítulo abordamos la problemática asociada a la carencia de mecanismos para la integración de repositorios semánticos RDF heterogéneos y altamente distribuidos en el contexto de la computación global, proponiendo como solución un modelo de federación basado en enlaces RDF distribuidos y la navegación transparente de los mismos. Se han presentado los resultados obtenidos tras la implementación de prototipos preliminares para validar la efectividad de las tecnologías que se van a utilizar y los modelos arquitectónicos propuestos, obteniéndose excelentes resultados empíricos. Dichos resultados muestran que el coste en tiempo de ejecución de tener una

infraestructura de mediación semántica distribuida es perfectamente asumible. Adicionalmente, para probar la validez de la propuesta realizada se han aplicado las ideas expuestas en la implementación de un Museo Híbrido distribuido a partir de repositorios RDF siguiendo el modelo semántico MoMo-CRM.

"Moebius Strip" por Escher (1898-1972)

Capítulo 5

Orientación en Museos Híbridos Mediante Colonias de Hormigas

Este capítulo plantea el problema de la orientación en museos híbridos y la solución a dicho problema de optimización mediante un algoritmo de colonias de hormigas. En el capítulo se describe una implementación eficiente de dicho algoritmo mediante una arquitectura de computación siguiendo el modelo de "Grid Computing" y un mecanismo de descomposición del problema basado en información geométrica mediante el que se obtiene un algoritmo escalable y capaz de resolver instancias del problema de la orientación de gran tamaño.

5.1. Orientación en Museos Híbridos

La visita a museos de gran envergadura, bien por el número de salas de que disponen o por el tamaño de su colección de obras de arte plantea serios problemas de orientación a los usuarios. En un gran número de ocasiones, un visitante medio no sabe encontrar con facilidad aquellas obras que son de gran popularidad, ha de perder tiempo en la planificación de la visita, e incluso, cuando ha podido realizar dicha planificación, es altamente probable que no se ajuste al tiempo de que dispone. Este problema se conoce comúnmente como el problema de la orientación aunque en su origen no estriba en el contexto de

los museos híbridos sino en el de las carreras deportivas donde un grupo de participantes han de recorrer un espacio abierto desconocido para ellos con la ayuda de un mapa. Debido al hecho de que no sólo el tiempo necesitado para llegar a la meta sino también el número de puestos de control visitados determinan quién es el ganador, los corredores han de encontrar un balance entre la velocidad y el número estaciones de control que visitan. Este problema también se da en otros campos como el de la logística y el de las aplicaciones industriales y tiene la siguiente formulación.

Descripción

Dado un grafo no dirigido $G(V,A)$ con $|V|=n$ nodos en el espacio Euclídeo donde cada nodo tiene asociado un determinado beneficio S_i, $S_i > 0$ excepto los nodos v_1 y v_n (inicial y final) que tienen asociado un beneficio $S_1 = S_n = 0$; cada nodo puede ser visitado como máximo una vez; y cada arista (v_i, v_j) $v_i, v_j \in V$ tiene asociado un coste $c_{ij} > 0$ el objetivo es encontrar un camino de beneficio máximo donde el coste asociado a dicho camino no supere un determinado valor máximo T_{max}.

Formulación Matemática

$$Max \sum_{i=1}^{n} \sum_{j=1}^{n} S_i \cdot x_{ij}$$

Manteniendo las siguientes restricciones

$$\sum_{j=2}^{n} x_{1j} = \sum_{i=1}^{n-1} x_{in} = 1$$

$$\sum_{i=2}^{n-1} x_{ik} = \sum_{j=2}^{n-1} x_{kj} \leq 1, \qquad k = 2, \ldots, n-1$$

$$\sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} \cdot x_{ij} \leq T_{\max}$$

$$x_{ij} \in \{0,1\}, \qquad i, j = 1, \ldots, n$$

En nuestro caso particular los elementos de interés a visitar son obras de arte, el coste asociado es el tiempo de visita de cada elemento de interés, el beneficio

es el grado de popularidad del elemento de interés y el coste entre dos elementos es el tiempo estimado para llegar de un punto de interés a otro.

La estimación de dicha información en el contexto de los museos híbridos puede realizarse mediante diversos métodos:

- Popularidad: La popularidad de una obra puede ser estimada de forma participativa mediante el voto de los visitantes para señalar su grado de satisfacción con la misma; midiendo el tiempo que un visitante permanece en el entorno del elemento de interés aunque esto requiere la utilización de sistemas de posicionamiento en tiempo real y la cercanía espacial no garantiza que el usuario esté prestando atención a la misma ni que le haya gustado; midiendo el número de solicitudes realizadas para obtener las diseminaciones asociadas al elemento de interés o a elementos adicionales relacionados con el mismo; estableciendo grados de popularidad artificiales mediante la intervención del personal del museo o de críticos de arte. Este último mecanismo es especialmente útil, como veremos más adelante, para la gestión de las sinergias de visitantes de forma que se controlen los flujos de visita en caso de tener un gran volumen de asistentes o si se desea garantizar que éstos exploren nuevas obras de arte que inicialmente tengan poca popularidad. En nuestro caso la función de estimación de popularidad de una obra de arte i se define como la proporción de consultas que recibe cualquier diseminación asociada a dicha obra con respecto al volumen total de consultas a diseminaciones:

$$\overline{P}_i = \frac{\sum_{d \in Dis_i} hits(d)}{\sum_{e \in Dis} hits(e)}$$

- Coste Elemento Interés: La estimación del tiempo medio que un usuario permanece en el elemento de interés v se estima a partir de las duraciones medias de las diseminaciones obtenidas en una navegación en profundidad a partir de dicho elemento:

$$\overline{C}_v = \frac{\sum_{P \in Path(v)} \sum_{node \in P} t(node)}{|\, Path(v) \,|}$$

- Distancia entre Elementos de Interés (u,w): Estimado como el tiempo necesario para ir de un punto a otro a partir de la distancia euclídea entre los elementos y la velocidad media de desplazamiento de una persona.

$$D_{u,w} = \frac{|\, p_u - p_w \,|}{v}$$

5.2. Algoritmos para la Resolución del Problema de la Orientación

Aunque el planteamiento del problema de la orientación puede parecer simple, su resolución no lo es en absoluto dado que dicho problema pertenece a la categoría de problemas NP-Duros (Garey, 1979) lo que en términos prácticos quiere decir que no existe ningún algoritmo conocido que lo resuelva en tiempo polinomial. La resolución de este tipo de problemas entre los que destaca, entre otros, el problema del viajante de comercio,se aborda mediante técnicas que podrían clasificarse en dos categorías principales: los algoritmos exactos y los algoritmos aproximados. Los primeros definen un mecanismo para encontrar la solución óptima y se demuestra que dicha solución es la mejor posible a nivel global. Entre las técnicas más comúnmente conocidas para la resolución exacta de problemas combinatoriales están la ramificación y poda, la programación dinámica y la vuelta atrás o "backtracking". Sin embargo en un gran número de ocasiones dichos algoritmos presentan serios problemas de eficiencia. Por este motivo la mayor parte de algoritmos de resolución de problemas NP-duros abordan la obtención de soluciones aproximadas (no óptimas) pero de gran calidad en un tiempo razonable a nivel computacional.

Los algoritmos de naturaleza aproximada pueden a su vez clasificarse en algoritmos constructivos y algoritmos de búsqueda local. Los primeros generan una solución desde cero añadiendo un elemento a cada paso basándose únicamente en información local a ese elemento para tomar la decisión sobre qué nueva componente ha de añadirse a la solución. Un ejemplo claro de este tipo de algoritmos son los que llamados voraces ("greedy") (Brassard, 1996) que obtienen soluciones muy rápidas con costes computacionales muy bajos pero que desgraciadamente son de muy baja calidad dado que tienden a converger hacia soluciones óptimas locales.

Los algoritmos de búsqueda local parten de una solución inicial que puede haberse obtenido por ejemplo mediante un algoritmo constructivo e intentan de forma repetitiva mejorar dicha solución mediante movimientos a soluciones vecinas. Estos algoritmos por sí solos son incapaces también de obtener soluciones globalmente satisfactorias y su efectividad depende en una gran medida de la calidad de la solución que se les provee inicialmente. En el caso del problema de la orientación veremos que los algoritmos de búsqueda local jugarán un papel fundamental para la mejora de las soluciones obtenidas.

Dado que tanto los algoritmos constructivos como los de búsqueda local parecen no obtener soluciones de gran calidad, la investigación en las últimas décadas en el campo de la algorítmica ha radicado fundamentalmente en la definición de técnicas de propósito general capaces de guiar de forma efectiva

el proceso de selección de componentes en el caso de los algoritmos constructivos o la búsqueda local. Estas técnicas de propósito general son comúnmente conocidas como metaheurísticas (Glover, 2003) y son un marco de propósito general que pueden ser aplicadas a diversos problemas de optimización si ya se ha definido para los mismos algún tipo de método heurístico. Las metaheurísticas suelen incorporar mecanismos o conceptos de disciplinas tan diversas como la biología, las matemáticas, la neurología y la genética. Entre las metaheurísticas que nos serán de interés para el problema de la orientación encontramos la búsqueda tabú (Glover, 1997), la búsqueda local con vecindario variable (Hansen, 1999) y de forma especial las colonias de hormigas (Dorigo,2003) dado que su estructura, como detallaremos mas adelante, permite el diseño de algoritmos paralelos o distribuidos de forma efectiva.

En concreto si nos ceñimos al problema de la orientación, podemos encontrar una gran diversidad de trabajos que se describen con cierto nivel de detalle en (Liang, 2003). Entre las propuestas que tratan de obtener mediante métodos exactos soluciones a este problema destaca (Laporte, 1990) donde se propone una relajación lineal de un modelo de programación entera con variables 0-1 en el marco de un esquema de ramificación y poda. En este algoritmo se relajan las restricciones iniciales de forma que el problema relajado se puede resolver mediante programación lineal y resolviendo de forma gradual las condiciones violadas mediante ramificación y poda. Otros autores como (Fischetti, 1998) (Leifer, 1993) utilizan una técnica similar de resolución de una versión relajada del problema a la que se van añadiendo gradualmente inecuaciones. En cambio en (Hayes, 1984) se utiliza por primera vez una técnica de programación dinámica y en el que los nodos del grafo no tienen asociado ningún beneficio. En este caso la función objetivo a minimizar es el tiempo total de viaje e imponiendo como restricción que los competidores han de visitar ciertos nodos. En (Ramesh, 1992) se usa una relajación de Lagrange conjuntamente con un proceso de mejora de la solución mediante ramificación y poda en el caso de que la relajación no obtenga una solución óptima. Todas estas aproximaciones han demostrado la obtención de soluciones para problemas de orden reducido y como ocurre en otros problemas de carácter NP-duro existen limitaciones temporales que hacen poco recomendable el uso de métodos exactos para la obtención de soluciones a favor de métodos heurísticos.

Los primeros métodos heurísticos se dan en los algoritmos *S* y *D* propuestos por Tsiligirides (Tsiligirides, 1984). En el algoritmo *S* se utiliza un método de Monte Carlo para la construcción de caminos utilizando distribuciones de probabilidad altamente correlacionadas con el ratio beneficio/distancia desde un nodo actual a un nodo destino. En el algoritmo *D* se utiliza el mecanismo

de scheduling de vehículos propuesto en (Wren, 1972). En este caso, el área de búsqueda se divide en sectores definidos mediante dos círculos concéntricos y un arco de longitud conocida. La variación de los sectores se consigue variando el radio de los círculos concéntricos y rotando los arcos. Un camino se construye cuando todos los nodos de un determinado sector han sido visitados o cuando es imposible visitar cualquier otro nodo del mismo circulo sin violar la restricción temporal T_{max}. Existen otras aproximaciones heurísticas, algunas de ellas derivadas a partir de la propuesta inicial de Tsiligirides y otras, en cambio, basadas en algoritmos genéticos (Tasgetiren, 2000), redes neuronales de Hopfield (Wang, 1995), algoritmos de varias fases de optimizaciones locales (Chao, 1996) (Goleen, 1987) y colonias de hormigas (Liang, 2003).

Especial atención merece la utilización de colonias de hormigas dado que muestra su superioridad con respecto al resto de heurísticas obteniendo las soluciones de mayor calidad. Pasaremos, pues, en el siguiente punto a describir con detalle este algoritmo heurístico, discutiremos los problemas asociados al mismo y las contribuciones que realizamos en esta tesis para resolverlos.

5.3. Optimización Basada en Colonias de Hormigas

Las colonias de hormigas, al igual que otros tipos de sistemas biológicos, son sistemas que presentan una organización social enormemente estructurada y en las que existen mecanismos de coordinación que permiten a las mismas realizar conjuntamente tareas que serían difícilmente abordables de manera individual. En muchas especies de hormigas la percepción visual está muy poco desarrollada y por esta razón casi todos sus mecanismos de comunicación se basan en el intercambio de sustancias químicas producidas de forma natural conocidas como feromonas. En particular, en la vida social de las hormigas son de especial importancia las feromonas de rastreo "trail pheromones" que algunas especies como la "Lasius Niger" o la "Iridomyrmex humilis" utilizan para marcar de manera química caminos en el terreno que les conduzcan desde el hormiguero hasta los lugares donde han encontrado alimentos. A medida que las hormigas se desplazan desde el hormiguero hasta una fuente de alimento depositan feromonas en el terreno. Si una hormiga no encuentra ningún rastro de feromona se mueve de forma aleatoria pero si detectan dicha sustancia tiende a seguir con mayor probabilidad el rastro. Existen numerosos experimentos (Pasteels, 1987) (Goss, 1989) que han demostrado que las hormigas escogen aquellos caminos que tienen más concentración de feromonas y que en la práctica cuando varios caminos se bifurcan las hormigas eligen el camino por el que continuar su marcha basándose en la intensidad de feromonas. De esta forma, como las hormigas depositan feromonas en el camino que han elegido, este mecanismo supone un proceso de refuerzo que

concluye con la formación de caminos intensamente marcados. Este proceso tiene una razón de ser dado que como han demostrado los experimentos del puente doble de Deneubourg (Beckers, 1992) el mecanismo de autorefuerzo conduce a la selección de aquellos caminos más cortos. Deneubourg diseñó un puente doble con dos caminos (uno de ellos el doble de largo que el otro) que conectaba un hormiguero de la especie I. humilis con una fuente de alimento. En los experimentos realizados se pudo observar de forma consistente que tras un periodo transitorio donde las hormigas escogían aleatoriamente ambos caminos siguiendo un proceso que podría interpretarse como de exploración, las hormigas terminaban intensificando el camino más corto.

Basándose en este comportamiento natural de las hormigas Dorigo y sus colaboradores (Dorigo, 1993) (Dorigo, 1996) (Dorigo, 1999) introducen por primera vez la optimización basada en colonias de hormigas (OCH) para resolver problemas de optimización combinatoria. La formulación de Dorigo se basa en la existencia de de un modelo de hormiga artificial que son agentes software que trabajan de forma cooperativa comunicándose mediante rastros de feromonas también de naturaleza artificial.

Para entender de una forma sencilla cómo trabajan los algoritmos de OCH hemos de ver el proceso de resolución como un ejercicio constructivo en el que en cada iteración cada hormiga construye una solución recorriendo el grafo subyacente al problema. Este proceso constructivo puede verse como un proceso de decisión en el que cada hormiga debe decidir desde el último nodo del grafo ya visitado a qué nuevo nodo dirigirse. Para poder llevar a cabo esta decisión el algoritmo mantiene dos tipos de información, heurística y de rastros de feromonas respectivamente. La información heurística mide la atracción (o grado de preferencia) de moverse desde un nodo origen a un nodo destino y dicha atracción no se modifica durante la ejecución del algoritmo. En cambio, la información de feromonas artificiales mide la atracción aprendida de moverse entre dos nodos y actúa como una memoria a largo plazo. Esta atracción aprendida imita a las feromonas naturales y por tanto dicha información se modifica a lo largo de la ejecución del algoritmo según unos criterios que detallaremos más adelante.

5.3.1. Problemas Resolubles mediante OCH

Los problemas resolubles mediante OCH pertenecen a la categoría de problemas de camino mínimo que están caracterizados, como se describe en (Dorigo, 2003), por los siguientes aspectos:

Un conjunto de restricciones posiblemente dependientes del tiempo $\Omega(t)$

Un conjunto finito de componentes $C = \left\{c_1, c_2, ..., c_{N_C}\right\}$

Los estados del problema se definen en términos de secuencias $x = \langle c_i, c_j, ..., c_h, ... \rangle$ de longitud finita sobre los elementos de *C*. El conjunto de todos los posibles estados se denota como χ.

Un conjunto de soluciones candidatas $S \subseteq \chi$

Un conjunto de estados posibles $\overline{\chi} \subseteq \chi$ definidos mediante un test que es dependiente del problema que verifica que no es imposible completar una secuencia $x \in \overline{\chi}$ en una solución que satisface las restricciones Ω.

Un conjunto no vacío de soluciones óptimas $S^* \subseteq \overline{\chi}$ y $S^* \subseteq S$

Una función de coste $g(s,t)$ asociada con cada solución candidata $s \in S$

Una función de coste, o estimación del coste $J(x,t)$ asociados a los estados que no son soluciones candidatas

Se puede observar mediante un análisis comparativo que el problema de la orientación descrito anteriormente pertenece a la categoría de problemas resolubles mediante OCH.

Con esta formulación una hormiga artificial construye soluciones realizando caminos de forma probabilista en el grafo completamente conectado $G_C = (V, A)$ (grafo de construcción) donde los vértices son las componentes de *C* y las aristas de *A* conectan completamente las componentes de *C*. Tanto los nodos como las aristas pueden tener asociada los dos tipos de información mencionados anteriormente: un valor heurístico de atracción (η_i, η_{ij} respectivamente) y un nivel de feromona (τ_i, τ_{ij} respectivamente). A continuación veremos una definición más formal del modelo de hormiga artificial y el mecanismo mediante el cual recorre el grafo de construcción y hace uso de la información asociada al mismo para encontrar soluciones.

5.3.2. Hormigas Artificiales

Como hemos mencionado, una hormiga artificial *k* es un proceso estocástico constructivo que presenta las siguientes propiedades:

Hace uso del grafo de construcción $G_C = (V, A)$ en búsqueda de soluciones óptimas $s^* \in S^*$

Tiene una memoria $\mathfrak{M}^k$ para almacenar información sobre el camino seguido hasta un instante dado. Dicha memoria puede ser utilizada para

- Construir soluciones factibles

- Calcular valores heurísticos η_i, η_{ij}
- Evaluar la solución encontrada
- Recorrer el camino seguido hacia atrás

Tiene un estado de partida x_s^k y una o más condiciones de terminación e^k. Normalmente el estado inicial se expresa como una secuencia vacía o como una secuencia con una única componente

Estando en un estado $x_r = \langle x_{r-1}, i \rangle$ y si no se satisface ninguna condición de terminación, se mueve a un nodo *j* en su vecindad $\mathfrak{A}^k(x_r)$, esto es, al estado $\langle x_r, j \rangle \in \chi$. Si al menos alguna de las condiciones de terminación es cierta entonces la hormiga se detiene.

Selecciona un movimiento aplicando una regla de decisión probabilística. Dicha regla es función de

Los valores locales heurísticos η_i, η_{ij} y de feromona τ_i, τ_{ij}

La memoria privada de la hormiga en la que se almacena el estado actual

Las restricciones del problema

Cuando se añade una componente c_j al estado actual se puede actualizar el nivel de feromona asociado a dicha componente o a la arista seleccionada.

Una vez se ha construido una solución puede volver hacia atrás el camino realizado y actualizar los niveles de feromonas de las componentes seleccionadas

Es importante señalar que varias hormigas actúan de forma concurrente e independiente y que aunque cada hormiga puede encontrar una solución (probablemente pobre) las soluciones de mayor calidad emergen a partir de la interacción colectiva de varias hormigas. Esta interacción colectiva se obtiene mediante la comunicación indirecta que se da a través de las variables que almacenan niveles de feromonas y que las hormigas leen/escriben. Es pues, una forma de aprendizaje colectivo en el que cada hormiga modifica de forma adaptativa la manera en la que el resto de hormigas perciben el problema a resolver.

5.3.3. Modelos de OCH

Una vez descrita de forma genérica la metaheurística OCH hay que destacar que existen en la literatura diversos algoritmos que hacen uso de ella para problemas de naturaleza combinatoria. Pasaremos a describir de forma sucinta

el Sistema de Hormigas (Dorigo, 1996) que tiene interés de naturaleza histórica y el Sistema de Colonia de Hormigas (Dorigo, 1997) por ser el utilizado en (Liang, 2003) para la resolución del problema de la orientación, el cual tomaremos como referencia en nuestro trabajo.

Adicionalmente existen otros algoritmos similares como el Sistema de Hormigas con ordenación (Bullnheimer, 1999), el Sistema de la Mejor-Peor Hormiga (Cordón, 1999), y el sistema Max-Min (Stützle, 2000), que alcanzan resultados satisfactorios.

El Sistema de Hormigas

El Sistema de hormigas fue el primer algoritmo de OCH propuesto por Dorigo y sus colaboradores en 1991. Los autores proponen tres variantes denominadas SH-ciclo, SH-cantidad y SH-densidad. En el SH-ciclo las hormigas depositan feromonas una vez han completado una solución ("offline update"). En cambio, SH-cantidad y SH-densidad realizan la deposición de feromona a cada paso de construcción de una solucion ("online update"). En SH-densidad la cantidad de feromona depositada es constante mientras que en SH-cantidad depende de la atracción heurística η_{ij} de la transición seleccionada.

Los estudios empíricos han demostrado que SH-ciclo es superior al resto y por esta razón y dado que su mecanismo de actualización ha servido de referencia para posteriores algoritmos lo detallaremos a continuación.

Como hemos comentado la feromona se deposita una vez que todas las hormigas han completado una solución. En primer lugar los rastros de feromona de cada transición se evaporan mediante una tasa de evaporación de forma que los rastros se reducen en un factor constante

$$\tau_{ij}^{new} \leftarrow (1-\rho)\cdot\tau_{ij}^{old} : \rho \in (0,1]$$

Una vez evaporadas las feromonas cada hormiga *k* recorre su memoria local $\mathfrak{M}^k$ y deposita una cantidad de feromona $\Delta\tau_{ij}^k = f(C(S_k))$ en cada arco transitado del grafo de construcción donde la cantidad de feromona depositada depende de la calidad $C(S_k)$ de la solución S_k obtenida por la hormiga *k*:

$$\tau_{ij}^{new} \leftarrow \tau_{ij}^{old} + \Delta\tau_{ij}^k, \forall a_{ij} \in S_k$$

Por último, el proceso de construcción de la solución sigue el mecanismo descrito en el modelo de hormiga artificial donde la regla de decisión probabilista descrita en el punto 5 de dicho modelo es como sigue

$$p_{ij}^{k} = \begin{cases} \dfrac{\left[\tau_{ij}\right]^{\alpha} \cdot \left[\eta_{ij}\right]^{\beta}}{\sum_{u \in \mathfrak{A}_i^k} \left[\tau_{iu}\right]^{\alpha} \cdot \left[\eta_{iu}\right]^{\beta}}, & \text{si } j \in \mathfrak{A}_i^k \\ 0, & \text{en otro caso} \end{cases}$$

donde $\mathfrak{A}_i^k$ es el vecindario alcanzable $\mathfrak{A}^k(x_r)$ estando en el estado $\left\langle x_r^k, i \right\rangle \in \chi$ y $\alpha, \beta \in \Re$ son dos factores que ponderan la importancia de la información heurística y de los rastros de feromona. Si $\alpha = 0$ aquellos nodos con mejor atracción heurística son los seleccionados y nos encontramos ante un algoritmo probabilista de naturaleza voraz. Si, en cambio, $\beta = 0$ sólo las feromonas dirigen el proceso de construcción de soluciones de manera que se llega a un rápido estancamiento en el que las hormigas construyen siempre las mismas soluciones que suelen ser óptimos locales.

Es importante resaltar que los autores de este trabajo propusieron una mejora final a este procedimiento denominada SH-elitista en la que una vez realizado el proceso anterior la hormiga reina deposita un nivel de feromona adicional en aquellas aristas que pertenecen a la mejor solución encontrada hasta el momento en el proceso de búsqueda. Este incremento de feromona se produce en un factor E que indica el número de hormigas elitistas que se consideran

$$\tau_{ij}^{new} \leftarrow \tau_{ij}^{old} + E \cdot f(C(S_{mejor})), \forall a_{ij} \in S_{mejor}$$

El Sistema de Colonias de Hormigas

El sistema de colonias de Hormigas es una evolución del SH en el que se proponen diversas modificaciones y que detallaremos aquí pues es el algoritmo que se toma como base en (Liang, 2003) para la resolución del problema de la orientación. En primer lugar se añade una regla de selección probabilista distinta denominada regla proporcional pseudo-aleatoria en la que se hace uso de un valor aleatorio $q_0 \in [0,1]$ de forma que el siguiente nodo a visitar por la hormiga *k* se elige según la siguiente distribución de probabilidad:

$$Si\ q \le q_0$$

$$p_{ij}^{k} = \begin{cases} 1, & \text{si } j = \arg\max_{u \in \mathfrak{A}_i^k} \left\{ [\tau_{iu}]^{\alpha} \cdot [\eta_{iu}]^{\beta} \right\} \\ 0, & \text{en otro caso} \end{cases}$$

Si no

$$p_{ij}^{k} = \begin{cases} \dfrac{\left[\tau_{ij}\right]^{\alpha} \cdot \left[\eta_{ij}\right]^{\beta}}{\sum_{u \in \mathfrak{A}_i^k} \left[\tau_{iu}\right]^{\alpha} \cdot \left[\eta_{iu}\right]^{\beta}}, & \text{si } j \in \mathfrak{A}_i^k \\ 0, & \text{en otro caso} \end{cases}$$

Como podemos observar, en el primer caso ($q \leq q_0$) se explota la información disponible tanto de naturaleza heurística como de feromonas tomando como transición aquella que maximiza la atracción. Sin embargo en el segundo caso se aplica una exploración aleatoria siguiendo la distribución de probabilidad descrito en el algoritmo SH tradicional.

La segunda modificación de esta versión del algoritmo con respecto al algoritmo original estriba en el hecho de que únicamente la hormiga reina actualiza los rastros de feromonas cuando todas las hormigas han obtenido una solución. A esta actualización se le denomina *offline* y la solución que se tiene en cuenta para esta única actualización es la mejor solución global evaporando previamente los valores de feromonas de todas las aristas que participan en la mejor solución global. Además, en algunas ocasiones se pueden aplicar técnicas de búsqueda local para mejorar la solución obtenida antes de actualizar las feromonas.

Por último, la tercera variación con respecto al algoritmo SH original deriva de la actualización de las feromonas paso a paso por cada hormiga cada vez que se ha seleccionado una arista a_{ij} como sigue:

$$\tau_{ij}^{new} \leftarrow (1-\varphi) \cdot \tau_{ij}^{old} + \varphi \cdot \tau_0 : \varphi \in (0,1]$$

Donde φ es un parámetro más de decremento de feromona. La regla anterior aplica simultáneamente una evaporación y una aplicación de τ_0 unidades de feromona donde τ_0 es un valor muy pequeño que sirve de límite inferior para los valores de feromona. Al aplicar esta regla aquellas aristas visitadas por muchas hormigas tendrán un valor de feromona menor de manera que su atracción para otras hormigas será cada vez menor y por lo tanto se facilitará que no todas las hormigas sigan el mismo camino. A este mecanismo de actualización se le denomina actualización *online*.

En el caso del algoritmo propuesto en (Liang, 2003) para resolver el problema de la orientación se sigue la estrategia planteada en esta versión y que de forma pseudos-algorítmica podría resumirse como sigue

```
Inicialización de todos los parámetros
Loop
```

Sub Loop (para cada hormiga k)
Construir solución según la regla de decisión
Aplicar la modificación de feromonas online
Hasta que todas las hormigas han sido generadas
Aplicar la búsqueda local (opcional)
Evaluar soluciones y almacenar la mejor global
Aplicar la modificación de feromonas offline
Hasta alcanzar el criterio de parada

En el caso particular del problema de orientación además se particulariza la definición matemática expresada anteriormente de manera que

$$\eta_{ij} = \frac{S_j}{c_{ij}} \qquad \tau_0 = \frac{1}{n \cdot T_{\max}} \qquad S_{kp} = S_k \cdot \frac{T_{\max}}{T_k}$$

Donde, como ya definimos en la formulación matemática del problema de la orientación, S_j representa el beneficio (*score*) asociado a cada nodo del grafo de construcción; c_{ij} la distancia o coste asociado a la arista que une los nodos del grafo *i* y *j*; *n* el numero de nodos en el grafo, y S_{kp} es la penalización del beneficio asociado a una solución S_k no factible obtenida por la hormiga *k*. En esta versión del algoritmo se permiten soluciones no factibles porque, como se sugiere en (Ramalhinho, 1998), se permita la exploración de soluciones no factibles que se encuentren cercanas a la frontera de las soluciones factibles dado que en estas regiones es probable que se encuentren óptimos globales.

5.4. Algoritmos de OCH Paralelos

La obtención de algoritmos paralelos de OCH ha sido recientemente sujeto de intensas labores de investigación. El objetivo básico a conseguir es la aceleración del proceso de construcción de soluciones sin comprometer la calidad de las soluciones finales obtenidas. Los algoritmos propuestos en este campo de investigación se clasifican en dos categorías según el paralelismo sea individualmente a nivel de hormiga o globalmente a nivel de colonia. En el primer caso, los algoritmos propuestos ubican a cada hormiga en un procesador dedicado obteniéndose de esta forma un paralelismo de granularidad fina. En el segundo caso, en cambio, varias hormigas e incluso una colonia entera comparten un mismo procesador o nodo de computación.

El trabajo propuesto por Bolondi (Bolondi, 1993) es un claro ejemplo de paralelismo de grano fino. Sin embargo, dicha propuesta no consigue tener buenas propiedades de escalabilidad debido al sobre coste asociado con las comunicaciones que son necesarias.

Un claro representante de paralelismo de grano grueso es el trabajo de Bullnheimer (Bullnheimer, 1998) donde se propone un mecanismo mediante el que el intercambio de información entre colonias tiene lugar cada *k* generaciones de soluciones. En dicho trabajo se concluye que a medida que se incrementa el valor k se tarda menos tiempo en obtener una solución. Sin embargo, en el trabajo no se discute como evoluciona la calidad de las soluciones obtenidas a medida que varía el factor *k*.

Stützle (Stützle, 1998) estudia la calidad de las soluciones que se obtienen al realizar varias ejecuciones independientes del algoritmo con un pequeño numero de iteraciones y estableciendo una comparación con la solución obtenida mediante una única ejecución del algoritmo donde el número total de iteraciones equivale a la suma total de las iteraciones realizadas por las ejecuciones independientes anteriores. El autor demuestra que en algunas ocasiones las ejecuciones independientes de menor número de iteraciones consiguen obtener soluciones de mejor calidad que la ejecución única de mayor duración.

En (Michels, 1998) se propone un modelo de islas. En este caso cada nodo de computación contiene una colonia de hormigas y las colonias intercambian información sobre la mejor solución obtenida localmente tras un número de iteraciones fijo. Si la solución que se recibe de una isla vecina es mejor que la mejor solución encontrada hasta ese momento, la colonia receptora adopta la solución vecina como mejor solución propia. También hay que resaltar que en un trabajo posterior de los mismos autores (Middendorf, 2000), se investigan diferentes tipos de intercambio de información en algoritmos de hormigas multi-colonia y se demuestra que el intercambio de una pequeña fracción de información heurística puede ser ventajoso a la hora de reducir los tiempos de ejecución.

Finalmente, en la propuesta de Talbi (Talbi, 1999) se utiliza un mecanismo maestro-esclavo en el que cada esclavo contiene una única hormiga que obtiene una única solución. Cada esclavo envía su solución al nodo maestro (o nido) donde se calcula una nueva matriz de feromonas que se envía de nuevo a los nodos esclavos.

En general hay que destacar dos factores importantes que limitan las prestaciones de todas las propuestas discutidas anteriormente. En primer lugar, tanto si se trata de paralelismo de grano fino como grueso, todas las hormigas

o colonias trabajan sobre la totalidad del espacio de búsqueda y esto limita seriamente el número de nodos que se pueden considerar en el grafo de construcción. En segundo lugar, tanto las hormigas como las colonias han de compartir información sobre feromonas que ha de ser comunicada. Los algoritmos paralelos que han de comunicar la matriz de feromonas al completo no alcanzan buenas prestaciones, y aún en el caso de que sólo se comparta la información de la mejor solución global (y no la matriz al completo) tal y como se describe en (Krüger, 1998), es necesario establecer puntos de sincronización entre las colonias para dicha comunicación, lo cual afecta también negativamente a las prestaciones.

Nuestro objetivo, pues, es demostrar que es posible diseñar un algoritmo paralelo de grano grueso y multi-colonia pero en el que las colonias trabajen como islas aisladas e independientes trabajando únicamente en una región parcial del espacio de búsqueda completo. Dichos espacios parciales serán obtenidos mediante técnicas de clusterización de manera que el algoritmo final pueda escalar de una forma simple con el tamaño del grafo de construcción.

5.5. Algoritmo de OCH Basado en Grid

El algoritmo que proponemos está inspirado, por un lado, en trabajos previos relacionados con el campo de la computación distribuida basada en Grids (Foster, 2001) y por otro en los clásicos esquemas algorítmicos de Divide y Vencerás. En el campo de la computación basada en Grids se plantean infraestructuras de computación en las que un conjunto de nodos distribuidos que son gestionados por distintas organizaciones y que trabajan de forma desacoplada son usados para resolver problemas de naturaleza científica de gran complejidad debido al volumen de datos manipulados o al tipo de computación que requieren. Nuestra estrategia general pretende resolver instancias de gran tamaño del problema de la orientación mediante la partición del espacio de búsqueda en subespacios que puedan ser resueltos de forma distribuida e independiente mediante una infraestructura de Grid sin que exista ningún tipo de comunicación heurística o puntos sincronización entre los nodos que participan en la computación.

Nuestra propuesta, de aquí en adelante denominada GRID-OCH-OP, puede ser vista como una infraestructura maestro-esclavo como la que propone Middendorf (Middendorf, 2002) pero en la que no existe propagación de la matriz de feromonas dado que los esclavos trabajan en instancias independientes y parciales con respecto al problema original.

La formulación general de nuestro algoritmo es como sigue:

```
GRID_OCH_OP
  Leer datos del problema
  Aplicar algoritmo de clustering
  Para_cada cluster
Aplicar OCH-OP
  Fin Para
  Integrar caminos parciales de cada cluster
  Devolver camino final, puntuación y coste
Fin GRID_OCH_OP
```

5.5.1. Clusterización de Instancias del Problema de la Orientación

Uno de los pasos claves de nuestro algoritmo es la partición adecuada del grafo de construcción en subregiones. Existen diversos algoritmos en la literatura para clasificar datos en categorías según diversos factores. Entre ellos, el algoritmo de las k-medias introducido por MacQueen (MacQueen, 1967) es uno de los más utilizados para agrupar datos con similares características en clusters de información.

En nuestro caso, dado que las instancias de grafos que manipulamos en nuestro contexto son de naturaleza euclídea, esto es, cada nodo del grafo tiene una posición *(x,y)* en un espacio bidimensional, podemos dividir el grafo de construcción mediante una implementación del algoritmo de las k-medias que tome como factor de categorización los valores *x* e *y* de cada nodo. La estrategia que subyace a esta categorización es asignar a cada colonia independiente de hormigas sólo aquellos nodos que son similares en términos de su posición en el espacio, esto es, aquéllos más cercanos entre sí. Esta es una buena estrategia dado que, de acuerdo con la teoría de grafos, es un hecho que ninguna solución óptima puede tener caminos que se crucen y por este motivo es bastante improbable que existan soluciones óptimas que se construyan realizando saltos de forma arbitraria entre distintos cluster que no se encuentren cercanos en el espacio.

Otro aspecto importante a resolver cuando dividimos el grafo de construcción original en subgrafos es cómo particionar la restricción global *TMax* entre los diferentes subproblemas que se obtienen tras el proceso de clusterización. Una posible forma simple de partición sería la repartición de la restricción de coste global de manera proporcional al coste medio observado en cada cluster u al numero de nodos tal y como sigue:

$$TMax_i = \frac{c_i}{\sum_{j=1}^{K} c_j} \cdot TMax, \quad c_i = N_i \frac{\sum_{(u,v)\in A_i} c_{uv}}{|A_i|}$$

Sin embargo, una forma alternativa de partición más interesante distribuye el valor de TMax impuesto de forma proporcional a la atracción media de los nodos de un cluster determinado, entendida dicha atracción tal y como se define en la heurística de OCH-OP propuesta por Liang:

$$TMax_i = \frac{c_i}{\sum_{j=1}^{K} c_j} \cdot TMax, \quad c_i = \sum_{u\in N_i} \frac{\sum_{v\in N_i} \frac{S_u}{c_{vu}}}{|N_i|-1}$$

Hay que resaltar el hecho de que con esta última forma de partición, si los clusters tienen tamaños similares (cosa que ocurre con enorme frecuencia si la disposición de nodos en el espacio es homogénea) y el coste medio es también equiparable, aquellos clusters con mayor densidad de nodos atractivos obtendrán una porción de la restricción TMax disponible dado que en estas regiones se obtiene un mayor beneficio por unidad de área. Sin embargo, hay que hacer hincapié en el hecho de que, en la fórmula anterior, el cálculo del factor c_i para cada cluster en un grafo de grandes dimensiones puede tener un coste temporal en absoluto despreciable y penalizar las prestaciones del algoritmo propuesto. Para resolver este problema, se propone la estimación $\hat{c}_i$ de dicho factor mediante la selección aleatoria de una muestra del conjunto de nodos/arcos pertenecientes a cada cluster.

5.5.2. Estrategia de Integración de Soluciones Parciales

Otro aspecto fundamental de nuestro algoritmo es cómo integrar las diferentes soluciones parciales obtenidas al aplicar OCH-OP. Para conectar un par de clusters el algoritmo que proponemos utiliza una aproximación heurística. Comenzando con el cluster que contiene el nodo inicial, se selecciona la solución del cluster más cercano, aquel cuyo centroide se encuentra a menor distancia. A continuación, se encuentra el mejor lugar (aquel que produce un menor coste) en el seno de la solución global para insertar la solución parcial y tras insertarla se continúa con el mismo proceso de forma iterativa hasta que todas las soluciones parciales han sido integradas en una única solución global. Hay que resaltar que, aunque las soluciones parciales encontradas son, por definición factibles, el proceso de integración no garantiza que la solucion

integrada tenga la propiedad de factibilidad, esto es, puede violar la restricción de coste global *TMax*. En este caso, procedemos con un proceso de optimización local consistente en concatenar diversos métodos iterativos de búsqueda local decreciente tal y como se propone en la metaheurística de Busqueda por Vencindad Variable de Mladenovic y Hansen (Hansen, 1999). En concreto los métodos de búsqueda local utilizados en nuestro algoritmo son:

Método de Eliminación (*ELM*): Excepto con los nodos inicial y final y empezando con el segundo nodo se elimina un nodo de la solución.

Método de Intercambio (*INT*): Empezando con el segundo nodo de un camino se intercambia éste nodo con un nodo no visitado si existe alguno.

Método de Swapping (*SWAP*): Se intercambian las posiciones de dos nodos pertenecientes al camino (excepto los nodos inicial y final)

Método de Inserción hacia delante (*FI*): Toma un nodo desde su posición actual y se inserta una posición más adelante pero sin sobrepasar al nodo final.

Método de Inserción hacia atrás (*BI*): Igual que *FI* pero hacia atrás.

Método de Adición (ADD): Se añade un nodo no visitado, si existe, en una posición entre el nodo inicial y el nodo final.

Más adelante, en la discusión de los resultados experimentales obtenidos discutiremos la efectividad de aplicar dichos métodos de forma intensiva dichos métodos y su impacto en la calidad de las soluciones obtenidas.

Para finalizar, la expresión algorítmica de todo el proceso heurístico descrito anteriormente es como sigue:

```
INTEGRACION_OCH_OP
Entrada: k solutions de OP desconectadas
Salida: solution global integrada (Sol_Merg)
Encontrar sol con nodo inicial (Sol_0)
Encontrar sol con nodo final (Sol_n)
Calcular centroides de las k solutiones
Sol_Merg=Sol_0
Mientras existan soluciones desconectadas
  Conectar(Sol_Merg, MasCercana(Sol))
  Calcular centroide de Sol_Merg
Fin Mientras
```

```
Conectar(Sol_Merg, Sol_n)
Si Sol_Merg no factible
  Aplicar Búsqueda Local INT, ADD, FI, BI, SWAP, ELM
Hasta que se encuentre solución factible
Devolver Sol_Merg, puntuación, coste
```

De una manera más formal, la heurística de integración de soluciones se expresa como sigue.

Dadas dos soluciones:

$$t = \langle t_1, t_2, ..., t_M \rangle$$
$$s = \langle s_1, s_2, ..., s_N \rangle$$

La integración de *s* en *t* resulta en una solución *m*

$$m = \langle t_1, t_2, ..., t_{k,} s_1, s_2, ..., s_N, t_{k+1,}, ..., t_M \rangle$$

tal que

$$k = \arg\min \{ c(t_k, s_1) + c(s_N, t_{k+1}) \}$$

Es de resaltar que en nuestra heurística encontramos el coste mínimo de integrar *s* de forma completa en *t*, esto es, sin dividir s en secciones. La justificación de la no división de s estriba en el hecho de que los métodos de búsqueda local realizan ya esta tarea sobre la solución global ya integrada. De esta forma conseguimos un proceso de integración rápido y eficiente tal y como se observará en los resultados experimentales.

5.5.3. Implementación

El algoritmo que proponemos en nuestro trabajo ha sido implementado mediante una infraestructura de Grid consistente en un nodo maestro que ejerce las funciones de coordinador y servidor y un conjunto variable y dinámico de colonias esclavas. El servicio maestro se encarga de la división del grafo de construcción, del proceso de registro de colonias dispuestas a trabajar (existe una colonia por cada nodo de computación), del envío de trabajo a las colonias remotas y de la integración de las soluciones parciales y aplicación de los métodos de búsqueda local.

Hemos implementado este servicio como un servicio Web mediante la tecnología .NET (Troelsen, 2005) y las colonias remotas como ensamblados de código manejado en .NET (todo ello utilizando el lenguaje de implementación C#). De esta manera, el código que implementa el comportamiento de una

colonia puede ser ejecutado en diversas plataformas. De hecho, en nuestro contexto de Museos Híbridos, este era un requisito fundamental dado que podemos ubicar colonias no sólo en ordenadores de sobremesa sino también en los ordenadores de bolsillo que los visitantes utilizan durante su visita en el museo. Así conseguimos, de una manera sencilla, construir Grids de dispositivos heterogéneos que puedan acomodar los picos de demanda en el cálculo de soluciones para distintas instancias del problema de la orientación.

Conceptualmente, el modelo que subyace a nuestra implementación se detalla en la Figura 35 donde hemos omitido todos los parámetros relativos a los métodos de las clases por limitaciones de espacio.

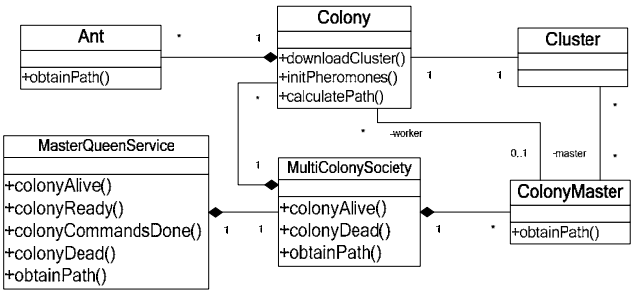


Figura 35 Modelo conceptual de Grid-OCH-OP

La clase *MasterQueenService* es la interfaz al mundo exterior de la parte "maestro" de nuestra arquitectura. Controla el registro de colonias "*colonyAli*, la disponibilidad para realizar trabajo "*colonyReady*", la obtención de resultados "*colonyCommandsDone*", la muerte de una colonia remota "*colonyDead*" y la petición de trabajo por parte de alguna entidad externa para que se obtenga una solución para una instancia determinada del problema de la orientación "*obtainPath*".

En el servidor, cada vez que se recibe una petición de cálculo, se selecciona una instancia de *ColonyMaster* disponible y se le asignan un conjunto de colonias disponibles para realizar el cómputo de forma distribuida. El objeto ColonyMaster toma entonces el control sobre sus múltiples esclavos, asignándoles trabajo (un cluster sobre el que se ha de encontrar una solución) y controlando su evolución. Es importante resaltar que las colonias y sus respectivos maestros son elegidas de forma dinámica de manera que una misma colonia en distintos momentos temporales puede trabajar para diferentes maestros y, por tanto, en diferentes instancias del problema de la orientación. Incluso, se puede dar el caso en el que si el número de clusters es mayor que el de colonias de hormigas libres entonces una colonia deba trabajar de forma secuencial en diferentes clusters. Este mecanismo proporciona la

flexibilidad necesaria para operar en un entorno a la Grid donde nuevas colonias pueden aparecer y desaparecer de una forma muy dinámica.

5.5.4. Resultados Experimentales

Para la evaluación experimental del algoritmo propuesto hemos seguido una estrategia incremental. En primer lugar, hemos querido corroborar la corrección de nuestra implementación ejecutando el algoritmo propuesto sobre las instancias de referencia propuestas por Tsiligirides (Tsiligirides, 1984), comparando los resultados que obtenemos con los mostrados por Liang. En segundo lugar, hemos realizado diversas ejecuciones sobre grafos Euclídeos de tamaño 1000 generados de forma aleatoria y utilizando hasta 32 colonias distribuidas para discutir sobre los resultados obtenidos aspectos como las prestaciones obtenidas y la degradación de la solución. En tercer lugar, hemos explorado el comportamiento de nuestra implementación al utilizar instancias del problema consistentes en grafos no homogéneos y, finalmente, hemos analizado la escalabilidad del esquema propuesto resolviendo diversas instancias con grafos de tamaño 5.000, 10.000 e incluso 100.000.

En los experimentos realizados, para obtener resultados de speedup comparables, hemos considerado que siempre existen suficientes colonias disponibles para resolver la partición establecida de manera que se asignan tantas colonias remotas como clusters se tengan que resolver. De esta forma todas las colonias trabajan en paralelo realizando en promedio la misma cantidad de trabajo.

Los parámetros α, β, q_0 y ϱ, tal y como se definieron de manera formal en la definición del algoritmo OCH-OP han sido fijados con los siguientes valores $\alpha = 1$, $\beta = 3$, $q_0 = 0.2$ y $\varrho = 0.9$. Adicionalmente, se establecieron criterios de parada en cada colonia de manera que el algoritmo se detiene tras 100 generaciones de soluciones o si en las ultimas 10 generaciones la mejor solución obtenida no mejora en más de un 2%. Hay que hacer notar que este criterio de parada es válido para el cálculo de las soluciones parciales dado que él calculo de precisión de la solución final se realiza en el servidor una vez las soluciones parciales han sido recogidas e integradas.

Corrección

Para asegurar la corrección de nuestra implementación del modelo de colonia inspirado en OCH-OP hemos realizado experimentos con todas las instancias de referencia de Tsiligirides sin división en clusters de forma que una única colonia obtiene soluciones para la totalidad de los grafos de construcción de referencia. Los resultados de dichas ejecuciones han sido comparados con los obtenidos por la implementación de OCH-OP realizada por Liang. La Tabla 4

resume los resultados obtenidos para el problema 3 de Tsiligirides variando la restricción de coste TMax tal y como realiza Liang en sus experimentos. Tras realizar 5 repeticiones sobre cada instancia la tabla muestra los mejores resultados obtenidos por nuestra implementación junto a los mejores resultados obtenidos por Liang. Se puede observar que nuestra implementación obtiene de forma consistente resultados con la misma calidad que los obtenidos por Liang excepto para las instancias con TMax=50,80 donde nuestra implementación obtiene un resultado con puntuación un poco menor. Dada la consistencia de los resultados obtenidos para todas las instancias de problemas propuestas por Tsiligirides podemos llegar a la conclusión de que la implementación de colonia de hormiga realizada en nuestro contexto es correcta y, por tanto, desestiman este factor como posible causa de degradación de la calidad de las soluciones obtenidas.

	OCH-OP			OCH-GRID-OP		
Tmax	Puntua	Coste	Tiempo	Puntua	Coste	Tiempo
15	170	14,573	36	170	14,467	40
20	200	19,792	2194	200	19,792	50
30	320	28,770	465	320	29,469	50
40	430	38,881	711	430	38,881	50
50	520	48,936	1528	500	49,333	60
60	580	59,341	402	580	59,341	70
70	640	69,139	12888	640	69,648	80
80	710	79,710	9406	700	79,946	160
90	770	89,313	12106	770	89,822	120
100	800	97,078	2105	800	97,240	120
110	800	97,078	403	800	97,240	100

Tabla 4. Comparación resultados de la instancia de problema 3

Análisis de los Resultados

Como ya hemos mencionado anteriormente, el principal objetivo de nuestra propuesta es encontrar soluciones de calidad para problemas de la orientación de gran tamaño en un tiempo razonable. Para poder estudiar el speedup de nuestro algoritmo hemos generado grafos Euclideos aleatorios con 1000 nodos distribuidos homogéneamente en un espacio de coordenadas de tamaño 50x50 y cuya puntuación (score) varía entre 0 y 99. Para los grafos generados se han realizado experimentos variando el número de colonias entre 1 y 32 y realizando 5 repeticiones de cada experimento. Los experimentos han sido realizados sobre un conjunto homogéneo de computadores Intel Pentium IV 2.4Ghz 256RAM.

En la Figura 36 se muestran los tiempos medios obtenidos (puntuación, tiempo total de ejecución, tiempo medio empleado por cada colonia y tiempo de integración de las soluciones) para uno de los grafos explorados. Se puede observar que el tiempo total que se requiere para obtener una solución completa decrece exponencialmente a medida que el número de colonias aumenta lo cual es un excelente resultado. Sin embargo, este dato que es meramente informativo sobre el tiempo de ejecución es un dato parcial que debe ser matizado y complementado con la información acerca de cómo evoluciona la calidad de la solución a medida que aumentamos el número de colonias. En este caso, también se observa que la calidad de la solución obtenida aumenta a medida que hacemos participar un mayor número de colonias en el cálculo de la solución hasta un punto en el que la solución se degrada en calidad debido a la excesiva partición del grafo de entrada. Este comportamiento se observa en todos los grafos generados y en todas las repeticiones de los experimentos realizadas. Este comportamiento tiene su explicación en el hecho de que una colonia de hormigas trabajando sobre un espacio de búsqueda reducido es capaz de encontrar mejores soluciones de carácter local que las que serían encontradas con un mismo número de iteraciones sobre un espacio de búsqueda de mayor dimensión. Sin embargo, este buen comportamiento tiene un límite dado que si el espacio de búsqueda es demasiado reducido, las soluciones locales obtenidas en dichos subgrafos son poco representativas de las soluciones obtenidas para el grafo original. Por ello, incluso aplicando intensivamente métodos de búsqueda local tras la integración de la solución completa, es imposible mejorar sustancialmente la solución encontrada dado que esta no se encuentra en la vecindad de una buena solución global.

La calidad de las soluciones obtenidas en el experimento de la Figura 36 viene en gran medida determinada por el uso intensivo de los métodos de búsqueda local descritos con anterioridad. Como consecuencia de ello, se obtienen tiempos de ejecución que son enormemente mejorables. Por ello, se ha diseñado una segunda batería de experimentos en los que sólo se ejecutan los métodos de búsqueda local de forma limitada. El objetivo de estos experimentos es doble, por un lado observar si el algoritmo propuesto sigue siendo capaz de mejorar la calidad de las soluciones cuando se incrementa el número de colonias, y por otro verificar si es posible, aumentando el número de colonias, alcanzar soluciones cuya calidad sea equiparable a las obtenidas mediante la aplicación intensiva de métodos de búsqueda local. Los resultados obtenidos bajo estas nuevas condiciones se observan en la Figura 37 y muestran que si no se realiza ningún tipo de partición en clusters, la calidad de la solución obtenida empeora sensiblemente (en torno a 5000 puntos para el grafo considerado en la Figura 36 en el que se aplicó búsqueda local intensiva).

Como contrapartida se puede observar que el aloritmo es 25 veces más rápido en obtener esta solución de baja calidad. Sin embargo, a medida que se incrementa el número de colonias, el tiempo de ejecución se reduce de forma exponencial como era de esperar pero la calidad de la solución llega a un punto (obsérvese los resultados obtenidos con 16 colonias) en los que la calidad de la solución es comparable con la obtenida con búsqueda local intensiva. Este resultado es realmente importante dado que la solución se obtiene en cerca de 4 segundos en lugar de los varios minutos que se necesitaban en el caso anterior.

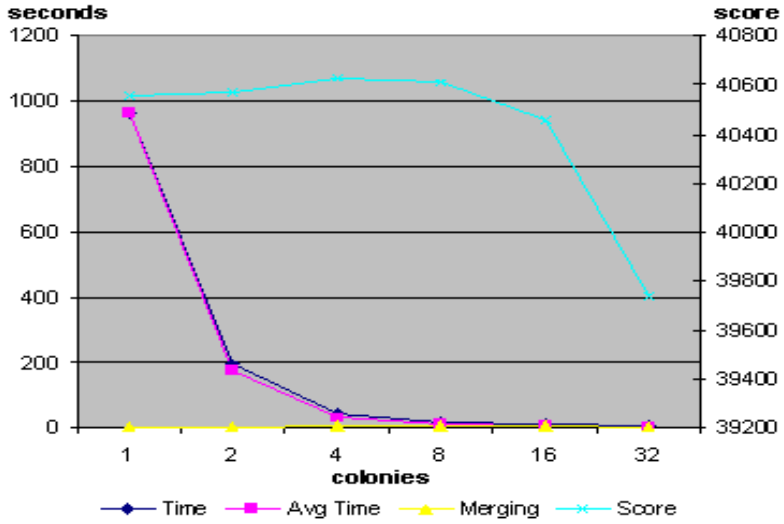


Figura 36. OCH-GRID-OP con 1000 nodos y VNS intensivo.

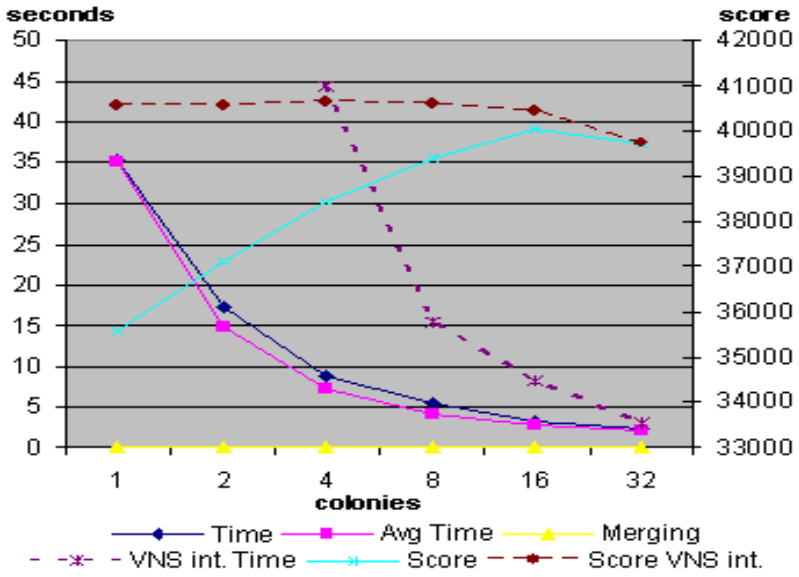


Figura 37. OCH-GRID-OP con 1000 nodos y VNS relajado.

Como hemos podido observar, uno de los resultados más prometedores de nuestra propuesta multi-colonia es su capacidad de mejorar la calidad de las soluciones obtenidas a medida que se incrementa el número de colonias hasta que se llega al punto de degradación. Para verificar que este comportamiento es consistente con distintos tipos de grafos (grafos no homogéneos) hemos realizado experimentos en los que se ha variado el número de nodos que son notablemente más atractivos que el resto (grafos de distinta densidad atractiva). En la Figura 38, la Figura 39 y la Figura 40 mostramos los resultados obtenidos para grafos de tamaño 1000 con densidad (porcentaje de nodos enormemente atractivos) 10%, 50% y 90%. Los resultados muestran de forma consistente que la calidad de la solución se incrementa en casi todos los casos hasta llegar a un punto en el que con 32 colonias la solución obtenida degrada su calidad.

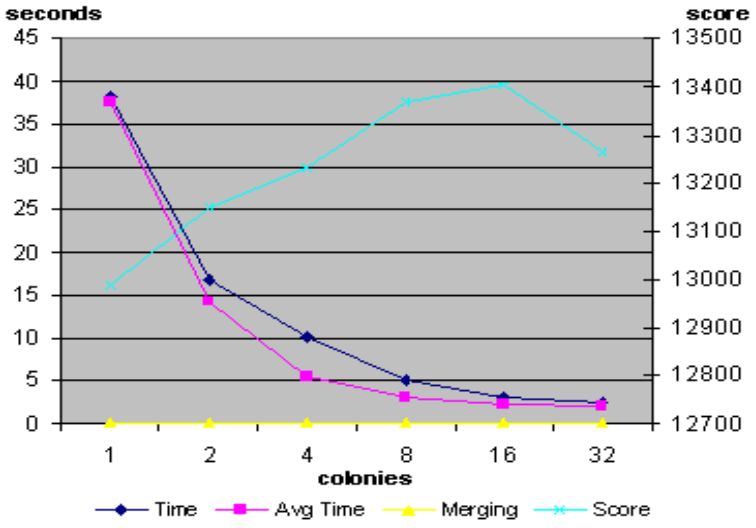


Figura 38. Experimento con 1000 nodos y densidad 10%.

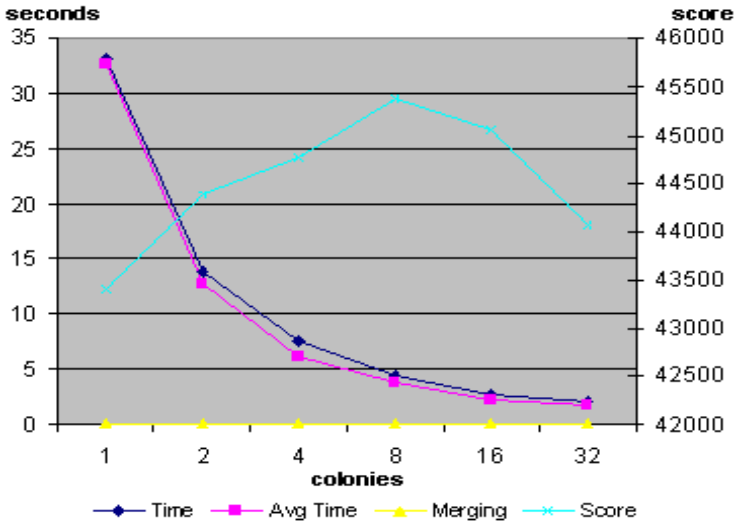


Figura 39. Experimento con 1000 nodos y densidad 50%.

Otro factor importante a analizar es si el comportamiento de mejora de la solución se mantiene cuando incrementamos el tamaño de los grafos considerados y si nuestra implementación es capaz de dar solución a grafos de mayor tamaño que los considerados hasta el momento. Para analizar estas cuestiones, se diseñaron experimentos para instancias de 5000, 10.000 y 100.000 nodos con hasta 32 colonias. En la Figura 41, la Figura 42 y la Figura 43 se muestran dichos resultados donde, por ejemplo, se puede observar que se resuelven instancias de 10.000 nodos con 32 colonias en menos de 10

segundos mientras que son necesarios más de 36 minutos para resolver el mismo problema con sólo dos colonias. Es interesante observar que a medida que el tamaño del grafo se incrementa, y por lo tanto, los clusters obtenidos son de mayor tamaño, el punto de degradación se desplaza hacia la derecha y previsiblemente ocurre haciendo participar a un número de colonias superior a 32. En la Figura 42, por ejemplo, con 32 colonias y un problema con 10000 nodos la solución obtenida no alcanza el punto de degradación.

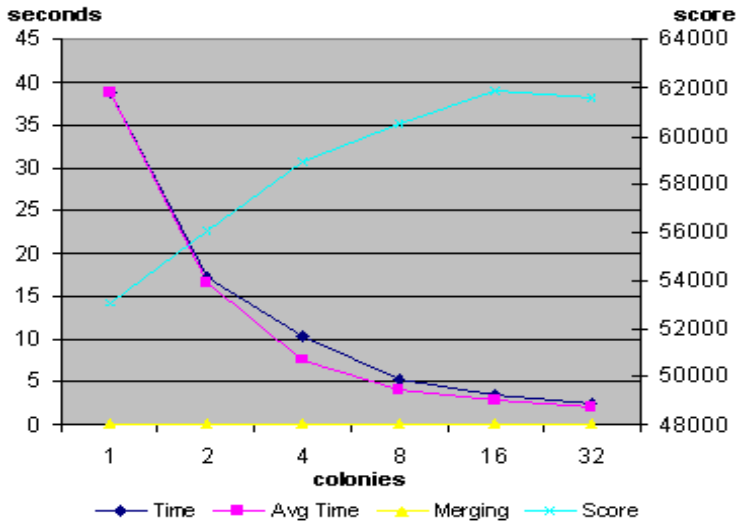


Figura 40. Experimento con 1000 nodos y densidad 90%.

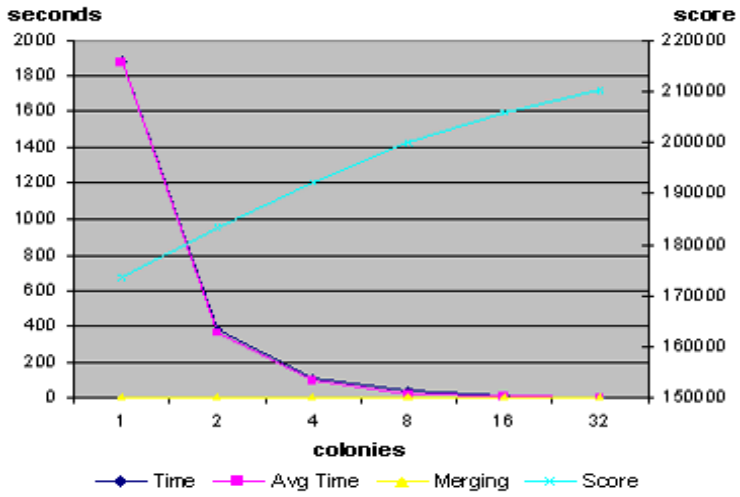


Figura 41. Experimento con 5000 nodos y VNS relajada.

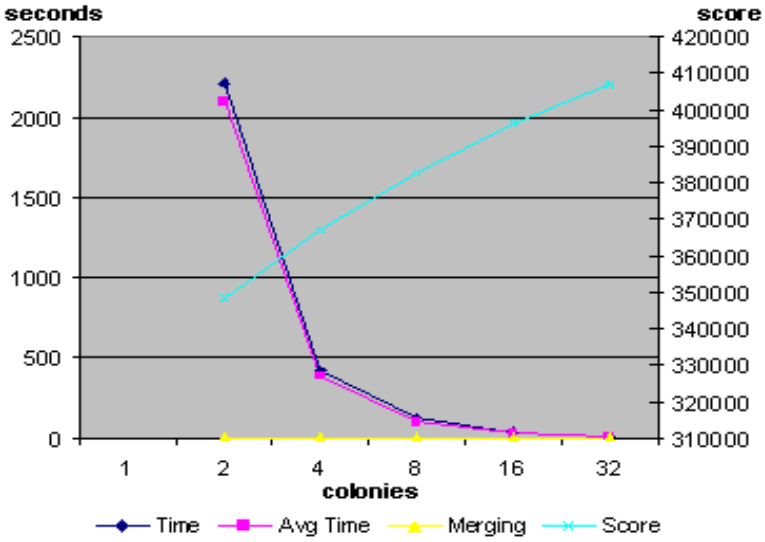


Figura 42. Experimento con 10000 nodos y VNS relajada.

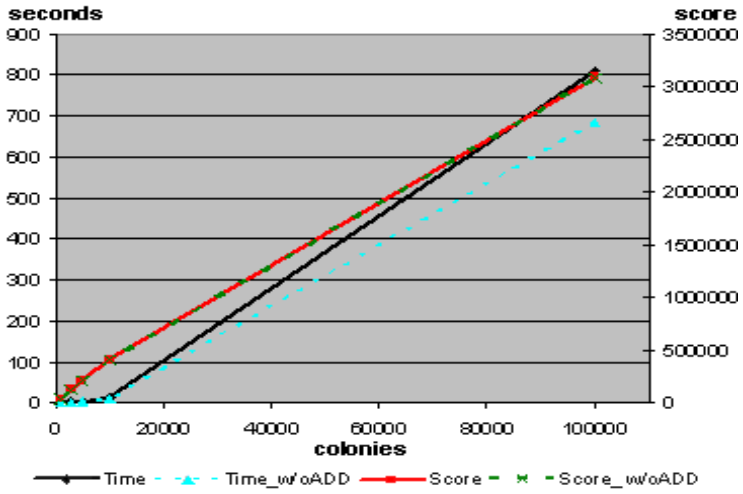


Figura 43. Time and score comparison using 32 colonies.

Finalmente, para analizar la escalabilidad de nuestra implementación podemos observar en la Figura 43 la evolución del tiempo total de ejecución a medida que incrementamos la talla del problema y manteniendo constante el número de colonias en 32. Se observa un crecimiento quasi lineal del coste asociado a la obtención de soluciones aunque hay que hacer notar que para instancias del problema de la orientación de gran talla (100.000) los tiempos obtenidos, pese a ser abordables, son de cierta magnitud debido al uso de mecanismos de búsqueda local. En concreto, observamos que si el método ADD es eliminado obtenemos una mejora cercana al 21% del tiempo de ejecución con una

degradación de la solución de sólo un 0.58%. Dicha mejora no ocurre al eliminar el resto de métodos locales considerados.

5.6. Conclusiones del capítulo

En este capítulo hemos planteado el problema de la orientación en museos híbridos y la solución a dicho problema de optimización mediante un algoritmo de colonias de hormigas. Hemos analizado las implementaciones paralelas existentes sobre algoritmos de colonias de hormigas y hemos constatado que en todos los casos dichas implementaciones no escalan a instancias del problema de gran tamaño. Hemos definido un algoritmo multi-colonia distribuido basado en un mecanismo de descomposición del problema en subproblemas y hemos realizado su implementación con servicios Web mediante una arquitectura de computación siguiendo el modelo de computación Grid. Tras el análisis de los resultados obtenidos podemos concluir que el algoritmo propuesto es capaz de resolver instancias de problemas de hasta 100.000 nodos con sólo 32 colonias. Hemos constatado que nuestra propuesta es capaz de mejorar la calidad de las soluciones obtenidas a medida que aumentamos el número de colonias involucradas en la computación y que este comportamiento se da consistentemente para distintos tamaños de grafos estudiados y diferentes grados de homogeneidad.

"Athene's Jewel Tree" por Garth Thornton (2002)

Capítulo 6

Técnicas de Resolución de OCH Mediante Unidades Gráficas de Procesamiento

En este capítulo se desarrollan dos propuestas y sus correspondientes implementaciones para resolver el problema de la orientación mediante una aproximación basada en colonias de hormigas utilizando para ello un procesador gráfico de última generación. En primer lugar, se hace una revisión histórica de la evolución de dichos procesadores, para a continuación presentar el modelo computacional actual asociado a los mismos: el "pipeline" gráfico. Tras esta introducción al "pipeline" gráfico y a los fundamentos de la GPGPU, *General-Purpose computation on GPUs*, que apuesta por la utilización de los procesadores gráficos o GPUs para realizar cálculos de propósito general, se proponen dos algoritmos, uno orientado a vértices y otro a fragmentos, y se presentan los resultados experimentales y las conclusiones obtenidas.

6.1. Introducción a la GPU

El propósito de este apartado es presentar la evolución de los procesadores gráficos. Se describirá cuál es la forma en la que se organiza el trabajo en estos procesadores, comentando conceptos básicos relacionados y los sistemas de

coordenadas que intervienen. Se comentará cuál es el grado de programabilidad actual de las GPUs; y por último se introducirán brevemente los conceptos de GPGPU.

6.1.1. El "Pipeline" gráfico hardware

La generación de gráficos 3D por computador está dispuesta en una organización de "pipeline". Un "pipeline" es una secuencia de fases o etapas que operan en paralelo en un orden fijo. Cada fase recibe sus entradas desde la etapa previa y envía sus salidas a la etapa siguiente. Este concepto de "pipeline" es el equivalente al de una línea de montaje de automóviles, en el que la línea se segmenta en diferentes fases de montaje dispuestas en un orden determinado. Cada fase recibe un automóvil en proceso de construcción desde su etapa previa, realiza la tarea que le corresponde y pasa el automóvil a su siguiente etapa de construcción. Finalmente se obtiene el automóvil completamente construido, en un proceso en el cual cada etapa ha realizado siempre la misma tarea. Además, cada etapa es capaz de actuar en paralelo respecto a las otras etapas si se aplican a distintos automóviles. En campos más cercanos al que queremos tratar, este concepto de "pipeline" es similar al empleado en otros campos de la informática como en el diseño de procesadores o unidades segmentadas.

Las etapas, de forma simplificada, del "pipeline" gráfico por hardware, son 1) la transformación vértices, 2) el ensamblado de primitivas, 3) Rasterización, 4) interpolación, aplicación de texturas y coloración, 5) Operaciones de Raster (Raster Operations)

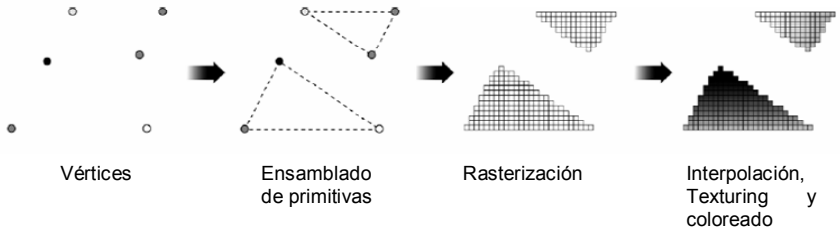


Figura 44. Descripción gráfica del procesamiento del *pipeline*.

Los vértices son los elementos básicos de los gráficos 3D y son los componentes del resto de primitivas que se utilizan para la generación de geometrías: triángulos, líneas y puntos.

Los vértices se definen con información de posición en el espacio, y opcionalmente pueden contener información adicional sobre su color, uno o más conjuntos de coordenadas de texturas, vector de la normal, etc.

Las texturas pueden considerarse como matrices de *texels* que almacena datos de imagen, principalmente en memoria de video. Un texel es un elemento de textura (texture element) que almacena un color en un determinado formato. La combinación de las intensidades de los colores básicos, rojo-verde-azul, forman los colores. De esta forma, en la computadora, un color típicamente es representado por una tripleta RGB o una tupla RGBA. Cada uno de los componentes de la tupla de color es llamado canal y la información que almacene indicará la intensidad del color básico que el canal representa mientras que el canal alpha es utilizado para determinar el grado de transparencia de un texel.

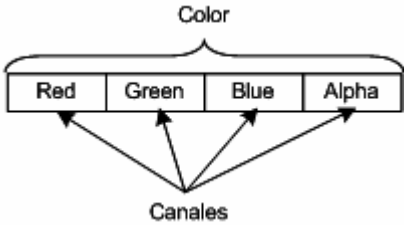


Figura 45. El color y sus canales

Las texturas soportan diferentes formatos para almacenar los colores de los texels. El formato lo que indica es la representación empleada por cada canal: número de bits e interpretación numérica (entero binario natural, coma flotante, etc.). Asimismo, un formato determinado puede contar con menos canales de los presentados en la tupla, pero siempre contará al menos con uno.

La CPU se encarga de suministrar el flujo de vértices definidos a la GPU para que empiece el procesamiento de los mismos. La transformación de vértices es la primera etapa en el "pipeline" gráfico, y lleva a cabo una serie de operaciones matemáticas aplicadas sobre cada vértice, con el propósito de transformar la posición de los vértices en una posición del espacio de pantalla para que sea procesado por el rasterizador y establecer el resto de propiedades, como el color, y coordenadas de textura. Al final del este apartado se introducirán los diferentes sistemas de coordenadas típicos y se expondrá cómo éstos traen la necesidad de transformar las posiciones de los vértices.

Como ya se ha advertido, los vértices forman las primitivas gráficas como los puntos, líneas y triángulos, así que la CPU también se encarga de suministrar la información de cómo la GPU debe ensamblar los vértices para formar las primitivas que se especifiquen. La segunda etapa consiste, pues, en aplicar la

información sobre unir los vértices procesados para obtener las primitivas gráficas que forman las geometrías complejas deseadas.

La CPU también debe aportar cuál es el espacio de *clipping* o recortado para que la GPU pueda aplicar el recortado de polígonos y el descarte de vértices adecuadamente. Típicamente, este espacio se define con información descriptiva del punto de observación (cámara) como son su posición, dirección a la que enfoca, dirección "hacia arriba" en el mundo), y una matriz de proyección acorde con el modelo de vista deseado, obteniendo de esta forma un espacio delimitado de dibujado como el que muestra la figura.

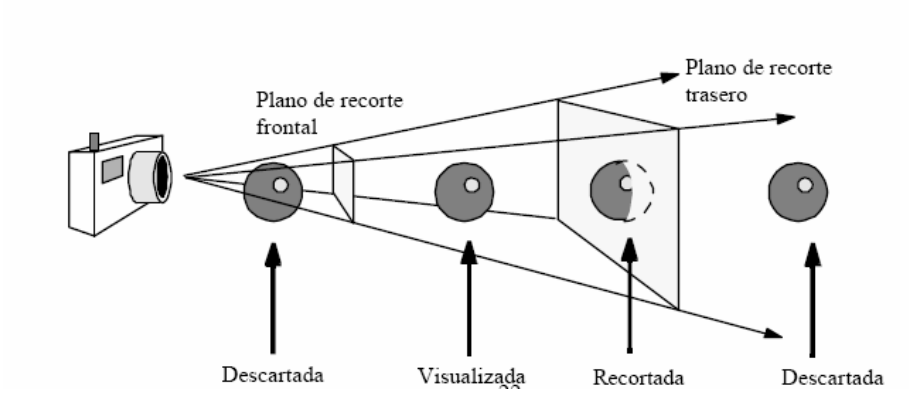


Figura 46. Clipping y Culling

La siguiente etapa, la rasterización, es el proceso que determina u obtiene el conjunto de píxeles que cubre una primitiva geométrica. Los píxeles, en este contexto, también son denominados fragmentos, y de aquí en adelante serán referenciados indistintamente por ambos términos aunque exactamente no son lo mismo. Un píxel en el sentido clásico es un "picture element", es decir, un punto del buffer destino de la imagen resultante 2D con el color como propiedad. Sin embargo, un fragmento es un como un píxel pero no correspondiente todavía al buffer 2D de la imagen resultante, teniendo además información adicional asociada como la profundidad, conjuntos de coordenadas de texturas derivadas de los vértices de la primitiva, etc. Además pueden existir varios fragmentos en la misma posición del espacio, cosa que no puede ocurrir con los píxeles. En definitiva, un fragmento es un "píxel potencial" con información añadida.

En el proceso de rasterización, es posible que se descarten polígonos basándose en el recortado (clipping) o en el descarte (culling), de forma que todos los fragmentos que corresponderían a la región recortada o descartada no se generan y por tanto no pasan a la siguiente etapa (ver Figura 3).

Obviamente, no existe relación entre el número de vértices que tiene una primitiva y el número de fragmentos que se generan tras la rasterización.

Una vez generados todos los fragmentos, éstos pasan a la etapa de interpolación, aplicación de texturas y coloreado. En dicha etapa, se interpolan los parámetros del fragmento como se requiera, llevando a cabo una secuencia de operaciones matemáticas y de textura, y determina el color final de cada fragmento. Además se determina la profundidad relativa de los fragmentos para poder decidir en la siguiente etapa qué fragmentos deben finalmente visualizarse o cuáles deben descartarse pese a haber pasado el recortado y el descarte de la etapa de rasterización.

La etapa de operaciones de raster, llamada Raster Operations, es una etapa en la que se definen operaciones de testing a nivel de fragmento con el propósito principal de descartar fragmentos que no deban salir del *pipeline*. Estos test son principalmente el *scissor test*, el *alpha test*, el *stencil test,* y el *depth test*, siendo este último el más importante y el más utilizado ya que sirve para la eliminación de superficies ocultas.

Como última fase principal, aunque no se ha especificado, sería la de escritura de los fragmentos que han salido con éxito del "pipeline" al buffer de píxeles 2D de la imagen renderizada. Generalmente este buffer es el denominado "frame buffer".

Respecto al funcionamiento del "pipeline" gráfico descrito, no hemos hablado en ningún momento de programabilidad de ninguna de sus etapas. De hecho esta descripción de "pipeline" se corresponde al denominado "pipeline gráfico de función fija", del inglés "fixed function graphics pipeline" o simplemente "fixed graphics pipeline". Esto es así porque, efectivamente, aunque sea totalmente por hardware este "pipeline" no es programable y tan solo podemos especificar determinados comportamientos fijos soportados de forma predefinida.

Los Sistemas de Coordenadas y la Transformación de vértices

Un sistema de coordenadas establece las posiciones de los objetos en el entorno respecto al origen de coordenadas del sistema. La posición de los puntos o vértices dependen del sistema de coordenadas que se emplee, y dos puntos sólo son comparables si sus posiciones vienen expresadas en relación a un sistema de coordenadas común para ambos.

Generalmente, en el modelado de objetos tridimensionales y en la generación de imágenes a partir de los mismos se tienen en cuenta distintos sistemas de coordenadas, ya que cada uno de ellos tiene un propósito concreto.

Para expresar las posiciones de los vértices que están en un determinado sistema de coordenadas en posiciones referentes a otro sistema de coordenadas distinto se aplican las denominadas *transformaciones*. Las transformaciones están fundamentadas matemáticamente, y por tanto su aplicación es automatizable y eficiente, además las API 3D proporcionan rutinas que facilitan su creación y gestión (Luna, 2003).

Las transformaciones matemáticas se expresan en matrices de dimensión 4x4. Con miras a la eficiencia y por la necesidad de un modelo matemático que sea capaz de sintetizar todas las operaciones de transformación en una única matriz las coordenadas tridimensionales <*x,y,z*> vienen a expresarse en coordenadas homogéneas. Las coordenadas homogéneas se caracterizan por ser tuplas de 4 componentes <*x, y, z, w*> donde la *w*, componente homogénea, es el valor por el cual se debería dividir las otras tres para obtener las coordenadas convencionales 3D o coordenadas no homogéneas.

$$\left\langle \frac{x}{w}, \frac{y}{w}, \frac{z}{w}, 1 \right\rangle = \langle x, y, z, w \rangle$$

A continuación se va a exponer brevemente los sistemas de coordenadas y qué transformaciones se emplean para *pasar* de uno a otro (ver Figura 4):

- *Model Space*: es el espacio del modelo o del objeto, y también es conocido como *Object Space*. Generalmente, los vértices de un objeto 3D vienen expresados en el espacio de coordenadas en el cual se define su modelo. El propósito de este sistema es que cada objeto sea independiente del resto de objetos que puedan existir en el mundo o en la escena. Las posiciones de los vértices que componen un objeto son relativas a un sistema propio del objeto, haciendo fácil el cambio de escala, de orientación y de posición del mismo.

- *World Space*: es el espacio del mundo o de la escena. El propósito de este sistema es proveer de alguna referencia absoluta para todos los objetos de la escena. Una escena se compondrá de varios objetos 3D que estarán definidos con vértices en posiciones del *Model Space*. Al definir el *World Space,* todos los objetos pasan a tener coordenadas relativas al origen de coordenadas del mundo, pudiendo ubicarse de forma consistente en la escena aplicando transformaciones de rotación, traslación y escalado. El transformar las coordenadas de un objeto desde su *Model Space* al *World Space* se conoce como *Modeling Transform* o transformación de modelo.

- *Eye Space*: es el espacio del ojo o de vista. Una vez definida la escena, se pretende que sea vista desde un determinado punto de vista, desde la posición de "un ojo". Este sistema de coordenadas *también* es conocido como V*iew Space*.

La posición del "ojo" es establecida como origen del sistema de coordenadas y por tanto todas las posiciones de los objetos serán relativas al mismo.

La transformación de vértices del *World Space* hacia *Eye Space* se denomina *View Transform* o transformación de vista. Esta transformación de vista consta de una transformación de traslación y una transformación de rotación, en virtud a la translación necesaria de los objetos a una posición relativa a la posición del ojo y a la rotación correspondiente a la orientación del punto de vista.

Las transformaciones del modelo y las transformaciones de transformación de vista acostumbran a combinarse en una única matriz para ser más eficientes. Esta matriz que recoge ambas transformaciones se conoce como la *ModelView Matrix*.

- *Clip Space*: Una vez las posiciones están en el *Eye Space*, el siguiente paso es determinar qué posiciones son actualmente visibles en la imagen desde el punto de vista. El sistema de coordenadas que se ocupa de ello es el *Clip Space*. El Clip Space será el espacio de recortado, que será el espacio visible por el ojo, que nos permitirá recortar y descartar superficies.

La transformación que pasa de posiciones del *Eye Space* al *Clip Space* es la denominada transformación de proyección o *Projection Transform*. Al igual que el resto de transformaciones, la transformación de proyección también se puede expresar por una matriz de 4x4, la matriz de proyección o *Projection Matrix*. Y de la misma forma que se puede combinar las transformaciones de modelo y vista para obtener la matriz ModelView, se puede combinar las tres transformaciones para obtener una única matriz denominada ModelView-Projection Matrix.

Los vértices transformados al *Clip Space* son los denominados vértices transformados, es decir, el objetivo de la etapa de transformación de vértices del *pipeline* es obtener vértices en el espacio de recortado.

- *Normalized Device Space*: La coordenadas en el Clip Space están en forma homogénea <*x,y,z,w*>, pero nosotros necesitamos calcular una posición 2D con un valor de profundidad para determinar las superficies que quedarían ocultas por otras, para cumplir con el objetivo del *pipeline*, que es generar imágenes planas.

Estos cálculos consisten en la división de perspectiva o *Perpective Division*, y se lleva a cabo dividiendo *x*, *y* y *z* por *w*.

- *Window Space*: El paso final es tomar las coordenadas de dispositivo normalizado de los vértices para obtener las coordenadas finales medidas en pixeles <*x,y*> del frame buffer. Este paso se conoce como *Viewport Transform*, y alimenta al rasterizador de la GPU. Será entonces cuando el rasterizador forme

puntos, líneas o triángulos a partir de los vértices, y genere fragmentos que determinen la imagen final.

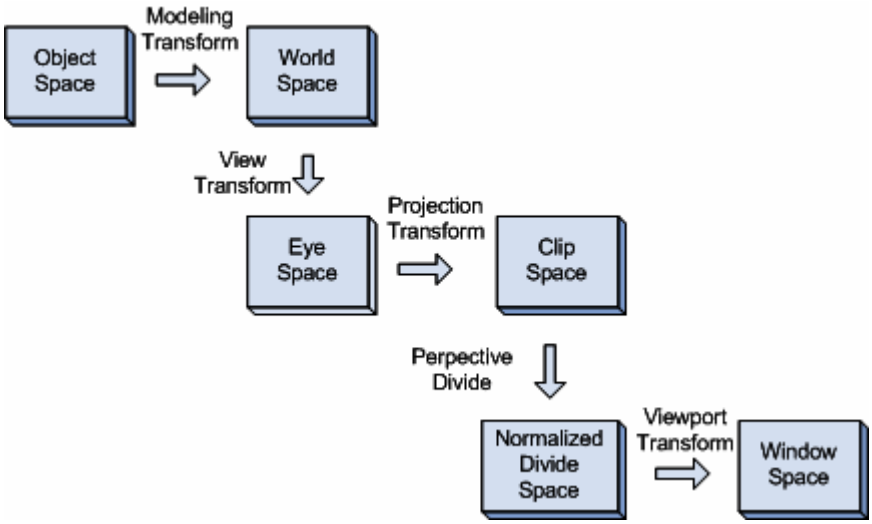


Figura 47. Resumen de sistemas de coordenadas y sus transformaciones

6.1.2. Revisión histórica de las GPU

Durante los últimos años se ha hablado de la Ley de Moore (Stallings, 2003), cuyo enunciado proclama que el número de transistores por centímetro cuadrado de oblea de silicio se duplica cada 18 meses, y en algunas aplicaciones incluso cada 12 meses. Este constante incremento de la capacidad de integración de transistores da lugar a hardware más barato y más rápido.

El hardware de las tarjetas gráficas no es una excepción a la Ley de Moore, e incluso el incremento de su capacidad de integración ha sido aún más espectacular que el de los procesadores de propósito general (Venkatasubramanian, 2004). A lo largo de los últimos años, la programabilidad de las GPU ha ido en aumento, y por ello, los principales fabricantes definen modelos de referencia (Shader Model) que indican capacidades de programabilidad que deberían cumplir las API 3D para un total aprovechamiento de cada generación de GPU (Venkatasubramanian, 2004b).

Aceleración Gráfica Pre-GPU (1982-1996): En esta etapa importantes compañías como Silicon Graphics diseñaron hardware gráfico especializado, pero a la vez muy costoso. Dado que este hardware no llegaba al gran mercado (el correspondiente a los PCs y consolas de video-juegos) este hardware no tuvo mucho éxito. Sin embargo esta etapa fue muy importante ya que este hardware ya aplicaba conceptos importantes como la transformación de vértices,

aplicación de texturas, etc., que servirían como base para el desarrollo posterior de las GPU.

Primera Generación de GPU (1996-1998): Se considera que la primera generación de GPU surgió a partir de 1996 con la tarjeta TNT2 de nVidia, Rage de ATI, y Voodoo3 de 3dfx. Estas GPU eran capaces de realizar la pixelización de triángulos pre-transformados y aplicar una o dos texturas. El logro principal fue que se descargaba a la CPU de la actualización de los píxeles, pero sus carencias más notables fueron la incapacidad de transformación de vértices (que debía hacerse en la CPU); y el reducido número de operaciones matemáticas que se podían llevar a cabo sobre los píxeles a colorear.

Segunda Generación de GPU (1999-2000): La segunda generación incluyó como máximos representantes a las tarjetas GeForce 256 y GeForce2 de nVidia, a Radeon 7500 de ATI, y a Savage3D de S3. Esta generación de GPU descargaba a la CPU de realizar la transformación de vértices 3D, lo que permitía una transformación rápida de los vértices, incrementando considerablemente el rendimiento de las aplicaciones gráficas. Pero pese a que se añadieron más operaciones matemáticas para ser aplicadas sobre los píxeles, éstas aun eran limitadas.

Tercera Generación de GPU (2001): Esta generación surge en el año 2001, e incluye a la GeForce3 y GeForce4 de nVidia, Radeon 8500 de ATI proporcionando programabilidad de los vértices, esto es, ofreciendo la posibilidad de especificar pequeños programas de procesamientos de vértices. Aunque se ofrece una mayor funcionalidad a nivel de píxel, no disponía todavía de programabilidad a nivel de píxel y tratándose de una mera transición hacia las GPU completamente programables.

Cuarta Generación de GPU (2002-2003): La cuarta generación incluía a la familia GeForce FX de nVidia (arquitectura CineFX) y Radeon 9700 de ATI. Esta generación de GPU ya proporcionan programabilidad tanto a nivel de vértice como a nivel de píxel. Esta capacidad de programabilidad dio la posibilidad de realizar complejas transformaciones sobre los vértices y operaciones más complejas de coloreado de píxeles.

Quinta Generación de GPU (2004-2005): Esta es la generación actualmente más difundida, e incluye a GeForce Serie 6000 de nVidia, y X800 de ATI. Esta generación permite la escritura de programas de vértices y píxeles de mayor longitud, soportando mayor precisión en los cálculos de coma flotante, un juego de instrucciones más rico, combinación de un mayor número de texturas, tanto a nivel de vértice como a nivel de píxel, capacidad de saltos condicionales y bucles reales, y renderización a múltiples buffers… En definitiva esta

funcionalidad es la soportada por el Shader Model 3, aunque no siempre completamente.

Sexta Generación de GPU (2006): en el momento de escribir estos párrafos los dos principales fabricantes de GPUs anunciaban el inminente lanzamiento de sus nuevos chips. Por una parte, nVidia anunciaba la serie GT70, incluido en la serie GeForce 7800, mientras que ATI anunciaba el lanzamiento del chip R520. En estas nuevas series de GPU no habrá importantes cambios en cuanto a la programabilidad de los chips, pero sí prometen una mejora sustancial del rendimiento. Estas mejoras pasan por aumentar el número de *vertex-pipelines* y *fragment-pipelines*, así como mejorar la arquitectura y organización del chip para incrementar el rendimiento de la funcionalidad que fue añadida apresuradamente en la anterior generación, especialmente la referente a los saltos condicionales y bucles que suponían una penalización importante.

No todas las fuentes del sector de las GPU distinguen, como nosotros, a la quinta y sexta generación. Esto es así porque con estas dos generaciones no existe un "escalón" tan importante de desarrollo como en las generaciones previas, en las que se aportaba importantes avances en cuanto a la programabilidad del procesador gráfico. Es más, estas dos últimas generaciones se corresponden más con generaciones desde un punto de vista comercial que con generaciones desde un punto de vista exclusivamente técnico.

En las futuras generaciones, se supone que llegará el soporte a modelos de sombreado más avanzados junto con las nuevas versiones de las API 3D. Se empieza a proponer la aparición de un tercer programa de GPU: "The Geometry Shader", que se encargaría de la generación de nuevos vértices en la GPU (Blythe, 2004). También un juego de instrucciones común para los procesadores de la GPU; un mejor soporte a la aritmética en coma flotante IEEE; acceso aleatorio de datos desde los programas. Todo indica que en un futuro se producirá la convergencia de la CPU y GPU, incluyendo la aparición de gestión de memoria virtual en la GPU.

A la vez que las GPU alcanzaban un mayor desarrollo y una mayor capacidad de programabilidad, surgieron algunas soluciones de alto nivel para programar GPU sin necesidad de recurrir al ensamblador. Estas soluciones vienen generalmente ligadas a las API 3D utilizadas (Direct3D, OpenGL) y consisten en un conjunto extendido de rutinas que permiten la compilación, establecimento, y la ejecución de los programas de alto nivel para GPU. Los lenguajes de alto nivel para GPU son principalmente HLSL de Microsoft DirectX, GLSL de OpenGL y Cg de nVidia, siendo todos ellos muy similares.

6.1.3. El "Pipeline" gráfico programable en GPU de 5ª Generación

Se va a introducir con cierto detalle el funcionamiento lógico del "pipeline" gráfico programable que caracteriza a las GPU actuales tomando como base el "pipeline" gráfico descrito en el punto anterior.

Del "pipeline" gráfico descrito hay dos etapas que actualmente son completamente programables y que por tanto permiten aumentar la funcionalidad de la función fija de la que disponíamos (fixed-function). Las etapas programables son la etapa de transformación de vértices, y la etapa de aplicación de textura y coloreado. Para cubrir la programabilidad de estas dos etapas las GPU programables proporcionan el procesador de vértices (programmable *vertex processor*) y el procesador de fragmentos (*programmable fragment processor*) respectivamente. Estas dos etapas ahora son programables, y por tanto se puede especificar operaciones más complejas y operaciones adicionales que antes no eran soportadas.

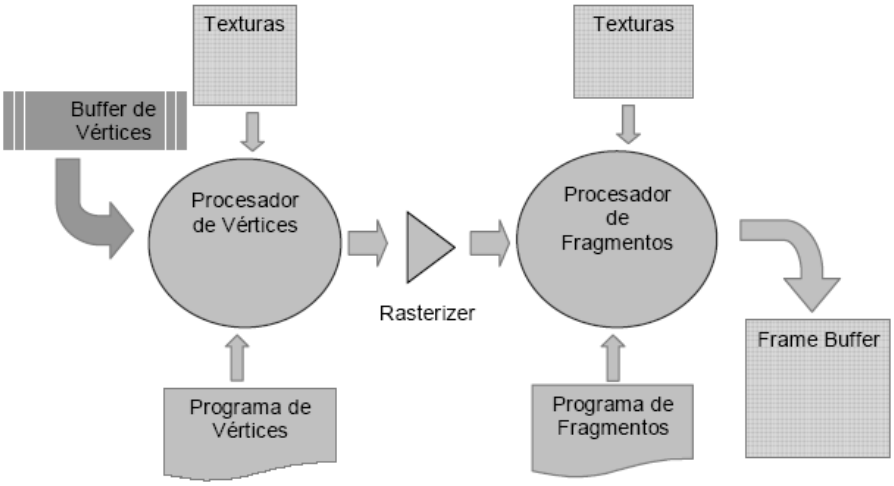


Figura 48.Descripción lógica de los componenetes del *pipeline* gráfico

De esta forma, el funcionamiento del "pipeline" se ve modificado ligeramente. La CPU proporciona el buffer o stream de vértices, la información de qué primitivas componen los vértices y el espacio de recortado, pero además proporciona un programa de vértices y un programa de fragmentos.

Cuando empieza la ejecución del "pipeline", los vértices fluyen hacia el procesador de vértices. En este procesador se ejecutará el programa de vértices especificado a cada uno de los vértices del stream. Una vez se obtienen vértices transformados, con posiciones expresadas en el *Clip Space*, el rasterizador realiza su función tal y como lo hacía en el "pipeline" no programable generando fragmentos. Sobre cada uno de los fragmentos que salen del

rasterizador se ejecuta el programa de fragmentos especificado en el procesador de fragmentos. Finalmente, se obtiene el conjunto de fragmentos procesados que además les corresponda convertirse en los píxeles finales de la imagen obtenida, siendo almacenados en el correspondiente buffer de destino (*frame buffer)*.

6.1.4. El Procesador de Vértices

El procesador de vértices es el encargado de ejecutar el programa de vértices, también conocido como *Vertex Shader* o *Vertex Program*. El programa de vértices se ejecuta para cada vértice que fluye por el "pipeline" obteniendo un vértice transformado acorde con las operaciones especificadas en el programa.

La principal misión del programa de vértices es realizar la transformación de la posición de vértices, la aplicación de texturas, y el descarte temprano, siendo estas acciones potencialmente complejas.

Cuando se procesa un vértice, éste no puede consultar información de ningún otro vértice. Este desconocimiento del resto de vértices proporciona paralelismo y es en parte la propiedad que posibilita el buen rendimiento del *pipeline*.

6.1.5. El Procesador de Fragmentos

El procesador de fragmentos es el encargado de ejecutar el programa de fragmentos, también conocido como *Fragment Program*, o *Píxel Shader*.

El programa de fragmentos se ejecuta para cada fragmento que fluye desde el rasterizador hacia el procesador de fragmentos, obteniendo fragmentos ya coloreados y preparados para que se realice la última etapa de conversión de los fragmentos a píxeles que finalmente formarán la imagen. Su principal misión es colorear el fragmento conforme a las operaciones especificadas en el programa. Este coloreado acostumbra a realizarse a partir del color, interpolado por el rasterizador a partir de los vértices de la primitiva que generó el fragmento en cuestión, y de los diferentes accesos a textura que pueda realizar en la ejecución.

Al igual que el procesador de vértices, en el procesamiento de un fragmento concreto solo se puede leer la información del propio fragmento, dando lugar al fenómeno deseable del paralelismo en el que se refuerza la capacidad del *pipeline*.

Además, actualmente el rendimiento del procesador de fragmentos es superior al del procesador de vértices. Este mayor rendimiento viene justificado por el mayor número de *fragment-pipelines* frente al de *vertex-pipelines* de la GPU, y los accesos a texturas más rápidos y con formatos más "ligeros". También hay que

tener en cuenta que los vértices todavía son generados en la CPU y que hay que hacer una transferencia de todos los vértices desde la CPU a la GPU para que sean procesados. En este sentido, los fragmentos son baratos, ya que no tienen que ser transferidos desde la GPU, y el *pipeline* será más eficiente en tanto en cuanto se tengan pocos vértices, aunque el número de fragmentos que estos generen sea elevado.

6.1.6. GPGPU, API 3D y lenguajes de Alto Nivel para GPU

El término *GPGPU*, "*General-Purpose computation on GPUs*" (GPGPU, 2005), hace referencia a la utilización de una GPU, como coprocesador que extiende la funcionalidad y las capacidades de la CPU, pero ya no solo para procesamiento y generación de gráficos sino para la resolución de problemas de naturaleza no gráfica. En definitiva, consiste en la utilización de la GPU para la programación en general y no específicamente para gráficos.

Utilizar la GPU para este propósito requiere entender cómo funciona el *pipeline* gráfico programable, conocimientos de las API 3D existentes, de los lenguajes de alto nivel para programación de GPU que se proporcionan, y algunos conceptos básicos de cómo hacer corresponder elementos del espacio del problema computacional no gráfico a elementos del espacio de la solución gráfica en GPU (Harris, 2004).

En general los conceptos se relacionan de la siguiente forma:

- Se reconocen colecciones de datos con la misma estructura. Este conjunto se acostumbra a nombrar *stream*.

- Se reconocen secciones del algoritmo que se apliquen a los *streams* identificados. Estas secciones son denominados *kernels* y son núcleos computacionales.

- Los *streams* y arrays de datos se corresponden, según el caso, a búffers de vértices o texturas principalmente.

- Los *kernels* a un programa de vértices y de fragmentos cada uno, que conjuntamente se encarga de hacer su función sobre el *stream* de entrada correspondiente.

- Las lecturas de memoria serán accesos a texturas si el *stream* se ha ajustado a una textura.

- Las lecturas de memoria serán la lectura de la propiedades del vértice o fragmento a procesar si el *stream* se ha ajustado a un búffer de vértices.

- La concatenación de la ejecución de *kernels* que consumen y producen *streams* conduce a la solución del problema. Es decir, el algoritmo que resolvía el

problema en CPU es descompuesto generalmente en varias fases de ejecución de distintos programas de GPU.

- La vista de la cámara y la proyección siguen un modelo de vista y proyección ortográfica. Recordemos que un modelo en perspectiva sería uno como el de la Figura 3, con forma de pirámide truncada, mientras que el modelo paralelo ortográfico tendría la forma de un exaedro, como se podrá ver en las ilustraciones correspondientes al apartado de *Aproximación intuitiva al algoritmo gráfico*.

Las API 3D proporcionan, como ya se adelantó, un conjunto de rutinas extendidas que permiten la programación en alto nivel de la GPU. Para exponer las correspondencias que se establecen y la forma de los programas de alto nivel de la GPU se van a presentar dos ejemplos. Estos ejemplos no son muy útiles en cuanto al trabajo que realizan, pero muestran algunos aspectos que se utilizarán como base en el algoritmo de OCH diseñado. El primer ejemplo consiste en una suma de matrices, mientras que el segundo consiste en obtener el valor mínimo de entre un conjunto de números reales positivos.

Ejemplo 1: La Suma de Matrices

La suma de matrices viene en pseudo-código de CPU expresado como:

```
Para todo i en 0..N-1 hacer
        Para todo j en 0..N-1 hacer
                C[i, j] = A[i,j] + B[i, j]
```

Si queremos aprovechar el rendimiento de la GPU se podría realizar la siguiente aproximación:

- El array bidimensional A pasa a ser una textura de entrada TA.

- El array bidimensional B pasa a ser una textura de entrada TB

- Definimos 4 vértices que formarán un cuadrado y cuya posición será la de las esquinas del área de trazado. Los vértices tendrán una coordenada de textura de forma que la coordenada de textura de cada vértice se ubique en una esquina diferente de las texturas de datos TA (para TB la coordenada de textura es la misma).

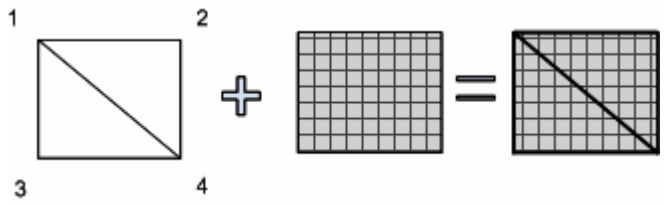


Figura 49. Vértices, primitivas, fragmentos generados y mapeo de textura

- La cámara será definida con una proyección ortográfica, no en perspectiva.

- El *frame buffer* destino de la renderización pasa a ser la textura de resultado TC.

- El programa de vértices no haría mayor trabajo que la sencilla transformación de la posición del vértice, a no ser que los 4 vértices de entrada ya estén transformados.

- El rasterizador generaría tantos fragmentos como la dimensión del área de trazado, es decir, tantos fragmentos como píxeles tenga la textura destino, ya que hemos definido con los 4 vértices un cuadrado de dicha dimensión.

Estos fragmentos pasarán al procesador de fragmentos en el que se ejecutaría el siguiente programa:

```
  sampler2D TA;
  sampler2D TB;
void main( in float4 position : POSITION,
        in float4 color : COLOR0,
        in float2 uv : TEXCOORD0,
        out float4 colorO : COLOR)
{
colorO = tex2D(TA,uv) + tex2D(TB,uv);
}
```

La función tex2D realiza un acceso a textura retornando el valor en la posición uv.

Nótese que no existen los bucles en el programa de fragmentos puesto que dicho programa se ejecuta para cada fragmento. Los bucles que recorrían los arrays en la CPU pasan a ser implícitos. También es importante notar que, a partir de los vértices con sus coordenadas de textura en las esquina de los datos se generan todo el conjunto de fragmentos y las coordenadas de textura de los mismos son obtenidas por interpolación automáticamente.

Los píxeles se almacenan en la textura resultado TC y puede ser leída desde la CPU para utilizar el resultado de la suma de matrices realizada.

Ejemplo 2: Obtención del Mínimo Real

Para la obtención del mínimo valor entre un conjunto de números reales positivos podemos hacer corresponder los elementos del problema de la siguiente forma:

- Existirán tantos vértices como números queramos evaluar. Las primitivas en este caso serán puntos.

- Cada vértice tendrá la misma posición X, Y en el espacio de renderizado, pero diferirán en Z. En Z vendrá el valor del número a evaluar con dicho vértice.

- La cámara será definida con una proyección ortográfica, no en perspectiva. Además la cámara estará ubicada en Z <=0.

- El *frame buffer* puede ser una textura. Realmente necesitamos un valor con lo que podría ser la textura más pequeña posible.

- El programa de vértices se encargaría de obtener los vértices transformados.

- Cada vértice generaría un fragmento.

- Los fragmentos pasarían al procesador de fragmentos que no tendrían ningún tratamiento especial. Sencillamente se "colorea" el fragmento con el valor del número que le corresponde al fragmento.

- En esta situación, de todos los fragmentos solo uno pasa a ser escrito en el *frame buffer*. Será aquel fragmento que más próximo a la cámara se encuentre. O dicho de otra forma, será aquel cuyo número asociado sea menor.

- Esta idea consiste en superponer vértices de forma que sus fragmentos queden ubicados de forma que solo se seleccione el visible.

Ahora se puede acceder al *frame buffer* para leer el número mínimo.

6.2. Algoritmo Orientado a Vértices para la Resolución de OCH mediante GPU

En esta sección se va a presentar un algoritmo diseñado para la resolución del problema de la orientación siguiendo una aproximación de optimización basada en colonias de hormigas.

Recordaremos algunos conceptos básicos, para a continuación presentar de forma intuitiva e ilustrada las ideas sobre las que se construye el algoritmo. Se

procederá a describir la correspondencia de las estructuras de datos, que en cierta medida ya habrán sido introducidas en la explicación del algoritmo. Será entonces cuando se presente el algoritmo en pseudo-código. Finalmente se comentarán algunos detalles sobre la implementación realizada, en especial aquellos que en el pseudo-código no se ve reflejado por simplicidad y claridad, y que ha de tenerse en cuenta en el momento de codificar el algoritmo.

Informalmente, OCH sigue una estrategia iterativa en la que la colonia de hormigas construye caminos que se ajustan a las restricciones definidas en el problema como ya vimos en el capítulo anterior. Recordemos que, de entre todos estos caminos se selecciona como solución aquél que, siguiendo una función de bondad o de puntuación, se considere mejor. Internamente existen dos aspectos que son característicos de la metaheurística de OCH: 1) el cómo cada hormiga construye su camino y 2) cómo se comunican las hormigas para cooperar en la construcción de los caminos de forma que al final se converja a una buena solución.

Una hormiga construye su camino "independientemente" del camino que esté construyendo el resto de hormigas de la colonia. La construcción del camino se puede entender como un proceso también iterativo en el que la hormiga decide, a cada paso de construcción, qué nodo ha de visitar de acuerdo al nodo en el que se encuentre en dicho paso, teniendo en cuenta los nodos que han sido ya visitados, siguiendo una regla probabilística que, como ya vimos, está definida en función de la atracción entre nodos y considerando las restricciones definidas por el problema.

La regla probabilista de transición, viene dividida en dos fenómenos: a) explotación de la máxima atracción; b) exploración pseudo-aleatoria siguiendo una regla probabilista basada en atracciones.

Como ya se introdujo, en OCH no se lleva a cabo una comunicación directa entre las hormigas de una colonia, sino que éstas se comunican indirectamente a través de sustancias químicas que depositan en el entorno (las feromonas). Así, en OCH existe información sobre las pista de feromonas e información heurística entre cada par de nodos, que combinadas forman la atracción. Dicha comunicación se lleva a cabo en forma de reglas de actualización de pistas de feromona, ya sea on-line u off-line, puesto que es la única forma que las hormigas tienen de modificar su entorno.

Por comodidad, de aquí en adelante, al conjunto de hormigas que forman la colonia le llamaremos H, y al número de nodos que forman parte del grafo será N.

6.2.1. Aproximación intuitiva al algoritmo gráfico

En este apartado se explicará con ilustraciones como funciona el algoritmo intuitivamente, dando especial importancia a los resultados que se quieren obtener y a los componentes que intervienen en el proceso.

Estructuras de datos

Se tiene un conjunto de vértices. Este conjunto de vértices representa la posibilidad de que cada nodo pueda ser visitado por cada hormiga de H. Así, un vértice determinado representa el que una hormiga h_k visite un nodo n_w del grafo.

El procesador de vértices, PV, y el procesador de fragmentos, PF, serán vistos como cajas negras que realizan las tareas necesarias para obtener los resultados deseados sobre las texturas de salida.

Habrá varias texturas, de las cuales, la textura de "grafo" y "atracción" serán datos de entrada, mientras que las texturas de "tour" y "prohibidos" serán texturas cuyo contenido se irá completando a medida que se avanza en el proceso de construcción de caminos.

La textura de "grafo" contendrá la información descriptiva del grafo. Esto es, información del coste, puntuación y posición euclídea sobre cada uno de los nodos del grafo. Este grafo es el grafo de construcción G_c. La textura "atracción" tendrá los valores de la atracción de ir desde un nodo *i* cualquiera a otro nodo *j* cualquiera.

La textura de "Tour" almacenará información sobre los caminos que va construyendo cada una de las hormigas de la colonia. Esta textura es, según lo visto en la definición de la hormiga artificial del capítulo anterior, la memoria M^k para las |H| hormigas de la colonia. En el texel (i, j) se almacenará la información sobre el j-ésimo nodo visitado en el camino construido por la hormiga i-ésima.

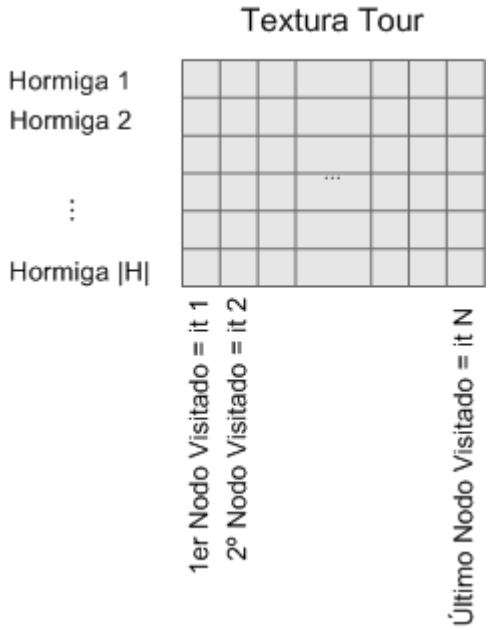


Figura 50. Contenido de la textura de Tour

La textura de "Prohibidos" indicará en cada momento qué nodos han sido visitados por las hormigas, y qué nodos aún no lo han sido en la construcción de sus respectivos caminos. En el texel (i, j) de esta textura se almacenará una marca que indica si el nodo *j* ha sido visitado por la hormiga i-ésima. De esta forma, en un determinado momento de la construcción de los caminos por parte de las |H| hormigas, cada hormiga podrá determinar si un nodo concreto ya ha sido visitado en su camino o si por el contrario puede ser candidato a ser visitado en el siguiente paso de construcción de su camino.

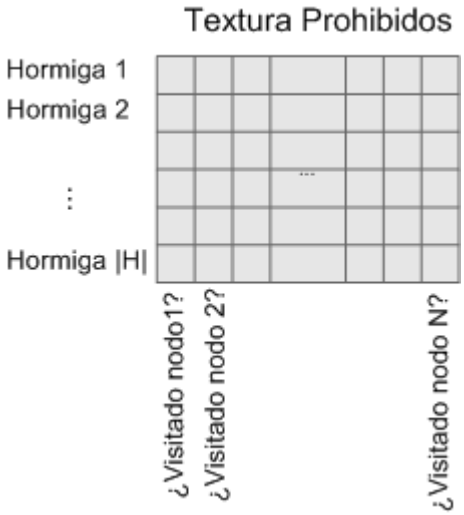


Figura 51. Contenido de la textura Prohibidos

Descritos los componentes básicos de nuestro algoritmo la Figura 9 recoge el esquema definido.

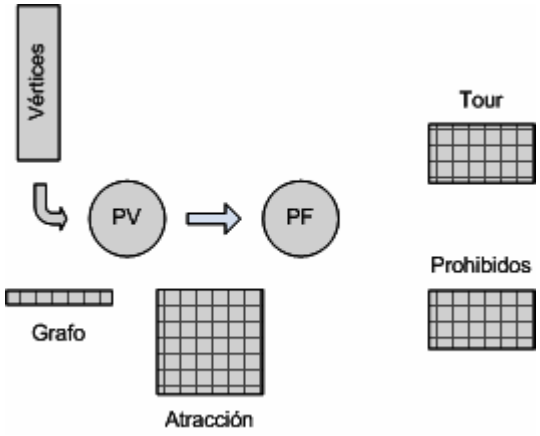


Figura 52. Descripción gráfica del algoritmo: componentes

Inicialmente, la textura de "Tour" no contendrá información relevante, y la textura de "Prohibidos" estará marcada de forma que indique que ninguna hormiga ha visitado ningún nodo.

Sería entonces cuando empezaría nuestro algoritmo, facilitando las texturas de "Grafo" y de "Atracción", y el conjunto de vértices. Las hormigas construirían sus caminos, y en la textura de "Tour" tendríamos la información de los mismos. En ese momento se seleccionará cuál es el mejor camino, procediendo también a la actualización de feromona y por tanto a la actualización de la textura de "Atracción". Si no se cumple el criterio de parada que se haya

considerado, se volverá a repetir el proceso, pero con la ventaja de que se parte con un entorno con caminos más marcados con feromona debido a los caminos obtenidos en iteraciones anteriores.

Construcción de caminos

La construcción de caminos por parte de las hormigas, como ya se ha adelantando, es un proceso iterativo donde cada hormiga debe decidir en cada paso qué nodo debe visitar. Así pues, cada paso de construcción de caminos va a ser concebido como un proceso que consta de dos subfases: a) selección del nodo a visitar; b) marcado del nodo visitado para que no sea considerado en los sucesivos pasos.

Visto así, el esquema que se presenta en la Figura 10 correspondería al de la *subfase a de construcción de caminos*. En esta subfase cada hormiga selecciona el siguiente nodo a visitar en su camino, y lo indica en la textura de *Tour*.

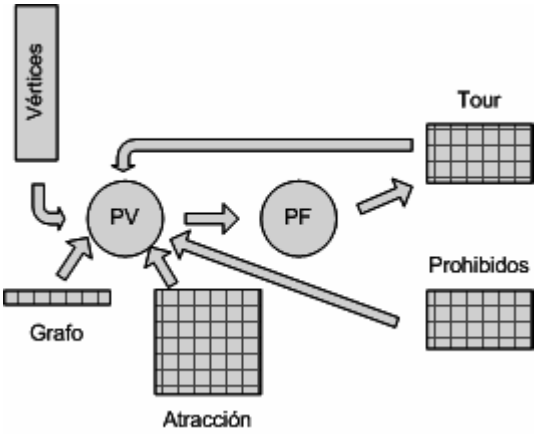


Figura 53. Descripción gráfica del algoritmo: Construcción de caminos

Para ello los vértices empiezan a fluir por el *pipeline gráfico*. Dado que cada vértice representa la intención de visita de un determinado nodo por parte de una determinada hormiga, un programa del procesador de vértices se encargará de descartar o ponderar la posibilidad de visitar el nodo asociado al vértice, en función de la información del paso anterior contenida en la textura de *Tour*, de la información de *Prohibidos*, de *Grafo* y de *Atracción*. Además de esta información, en esta subfase, como ya se recordó más arriba, se requiere la regla probabilista del proceso de construcción que consistía en *la explotación* y *la exploración*. Un programa de fragmentos se encargará de almacenar la información en la textura de *Tour* que servirá de reentrada para el siguiente paso de construcción.

La subfase b, de marcado de los nodos visitados por las hormigas en el paso de construcción de sus caminos presenta el esquema mostrado en la Figura 11.

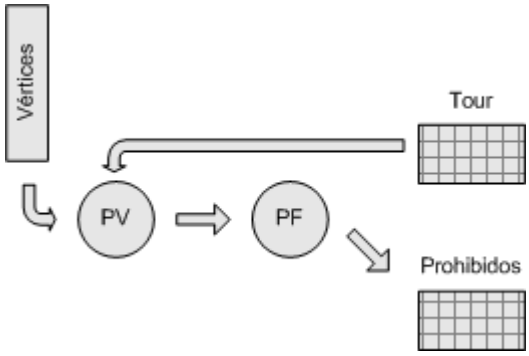


Figura 54. Descripción gráfica del algoritmo: Marcado de visitados

Para cada hormiga, se consulta en la textura de *Tour* qué nodo ha visitado en el paso de construcción anterior y se marca como prohibido en la textura de *Prohibidos* para evitar que vuelva a ser considerado.

Selección de nodos a visitar: Explotación y Exploración

La regla probabilista que determina la selección de nodos del grafo para la construcción de caminos consta de dos partes: explotación y exploración.

Si la hormiga realiza explotación, seleccionará aquel nodo que no haya sido todavía visitado para el cual la atracción desde el último nodo visitado sea máxima.

Si la hormiga realiza exploración, seleccionará aquel nodo que no haya sido todavía visitado, pero sin seguir estrictamente la regla probabilista dictada para el caso de exploración en el modelo de Sistemas de Colonias de Hormigas. Debido al coste y a la dificultad de mantener las probabilidades de la regla en cada paso de construcción, ya que dependen de los nodos visitados en cada momento, del último nodo visitado, y además debería ser considerado para cada hormiga de la colonia, se ha buscado una aproximación más laxa, que aún siendo de base aleatoria trate de tener en cuenta la información de atracción. Esta aproximación consiste en generar un número *r* pseudo-aleatorio, distribuido uniformemente en el intervalo continuo [min_atracción..max_atracción]. La probabilidad de que un nodo no visitado fuera seleccionado vendría ponderada por la siguiente expresión:

$$s = \left| r - \left([\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta} \right) \right|$$

De forma que será seleccionado aquel nodo que no haya sido visitado y para el cual su *s* calculada sea mínima, es decir, la diferencia entre el número *r* generado y la atracción de visitar el nodo sea mínima.

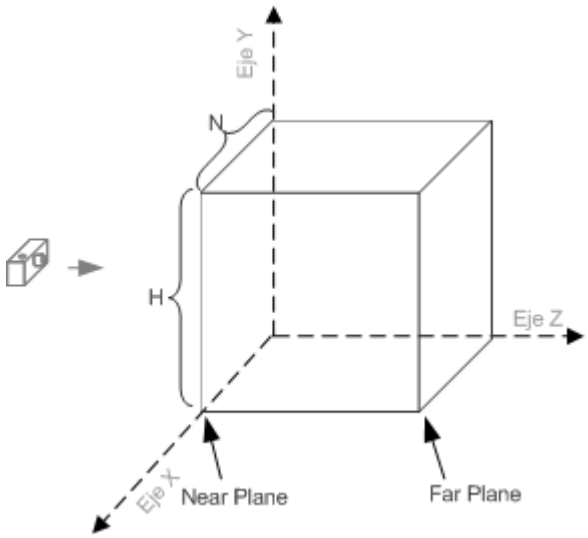


Figura 55. Espacio de recortado con vista paralela ortográfica

El proceso de selección, ya sea por explotación o exploración se basa en la idea de la superposición de vértices que se introdujo en el *pipeline gráfico*. Dicho proceso de selección se basa en la ubicación de un conjunto de vértices en la misma coordenada *x, y,* haciendo diferir esos vértices en la coordenada *z*, de forma que se seleccionará el vértice que esté más cercano a la cámara, puesto que será el único vértice visible al estar tapando al resto. Este objetivo se alcanza utilizando una cámara con proyección ortográfica y ubicando adecuadamente los vértices en el volumen de rendering.

En la figura 13 se pretende sintetizar el proceso de selección. Suponiendo que estamos en el paso *i* de construcción del camino de la hormiga h_k, todos los vértices correspondientes a la hormiga se dispondrían en la misma posición *(x, y);* dicho de otra forma, en el volumen de rendering se dispondrían en la *coordenada x* correspondiente al paso de construcción *i*, y en la *coordenada y* correspondiente a la hormiga h_k. Estos vértices diferirán en *z*, siendo asignada de acuerdo con la regla de selección propuesta. Si el vértice hace referencia a un nodo que la hormiga ya ha visitado, o si la hormiga está inactiva por haber encontrado ya una solución, entonces se ubicaría más allá del *far plane* (Figura 13) para forzar su descarte cuanto antes.

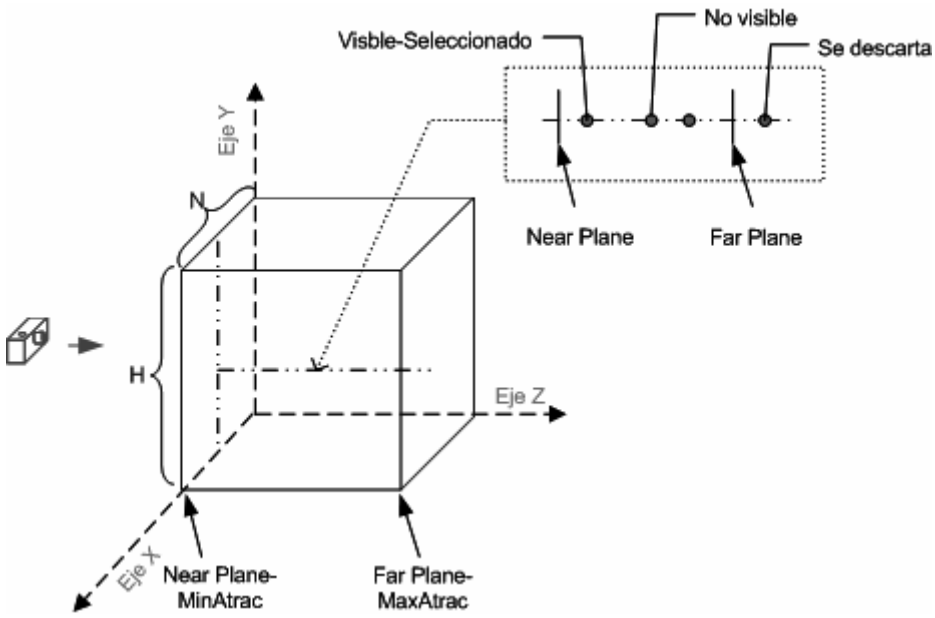


Figura 56. Selección del siguiente nodo a visitar

Si la hormiga tiene que hacer explotación, las z se asignan de forma que, a mayor atracción, más cerca de la cámara se sitúa cada vértice, mientras que si realiza exploración la z se establece en función de la atracción y del número pseudo-aleatorio *r*.

6.2.2. Algoritmo GPU-VERTEX-OCH-OP

En este apartado, se presentará con mayor detalle las estructuras de datos que se manejan en el algoritmo, para finalmente presentar cuál es el proceso algorítmico detallado que se lleva a cabo.

Estructuras de Datos

Buffer de Vértices

Los vértices serán los mismos durante toda la ejecución. Habrá N·H vértices, de forma que cada hormiga tendrá N vértices. Cada vértice de una hormiga representa la posibilidad de que dicha hormiga visite el nodo que codifica el vértice.

Un vértice viene caracterizado por codificar: en la *coordenada x* el número de nodo del grafo al cual se asocia el vértice; en la *coordenada y,* el número de la hormiga a la que está asociado el vértice; y en la *coordenada z* no se almacena información relevante.

Por supuesto, esta información se ha codificado como posición de los vértices solo para aprovechar la existencia de estas propiedades del vértice, pero en los programas de vértices, cuando se procede a realizar su transformación, se establecen los valores adecuados para que el vértice "caiga" en el punto del espacio que le corresponde.

Las primitivas que se utilizan son puntos, obteniendo una correspondencia entre vértices y fragmentos.

Textura de Grafo

Se trata de una textura de dimensión Nx1 con un formato A32B32G32R32F, es decir, color compuesto de cuatro canales, con 32 bits por canal con representación en coma flotante.

Cada texel de esta textura almacena la información sobre un nodo del grafo de construcción G_c. Cada canal almacena un elemento de esta información, siendo esta la siguiente (Figura 14):

- posX: posición X del nodo dentro del espacio euclídeo.

- posY: posición Y del nodo dentro del espacio euclídeo.

- S_i : Puntuación, beneficio o *Score* por visitar el nodo.

- W_i: Coste o *Weight* de visitar el nodo.

Concretamente, el primer texel almacenará la información del nodo 0, el segundo texel almacenará la información del nodo 1,… y el último texel almacenará la información del nodo N-1.

Figura 57. Contenido de un texel de la textura Grafo

Textura de Tour

Se trata de una textura de dimensión HxN, también con formato A32B32G32R32F.

La fila "i" almacenará el tour de la hormiga "i". De esta forma el texel (i,j) almacenará la información necesaria para la construcción del camino de la hormiga "i" en el paso "j".

Cada canal almacena uno de los siguientes elementos (Figura 15):

- nodo visitado y q: el nodo visitado es el número de nodo que ha sido visitado, mientras que q, es una marca binaria codificada en el signo del canal que indica si la hormiga debe realizar explotación o exploración.
- semilla: es una semilla para la hormiga que utilizará como entrada a un generador de números pseudo-aleatorios para la obtención de la siguiente q, determinar el número aleatorio a utilizar en la exploración, y determinar cuál es la semilla para el paso siguiente.
- Puntuación acumulada y hormiga inactiva: la puntuación acumulada es la puntuación o beneficio obtenido con el camino que se está construyendo. En el signo se codifica también si la hormiga está activa o inactiva, ya que es posible que ya haya encontrado una solución y que no necesite, por tanto, seguir realizando cálculos.
- Coste acumulado: es el coste del camino que la hormiga está construyendo.

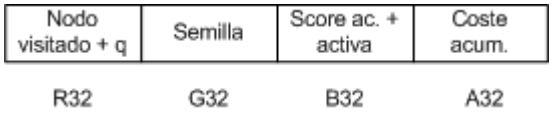


Figura 58. Contenido de un texel de la textura Tour

Textura de Pista de Feromona

Esta textura tiene una dimensión NxN de formato R32F, es decir con un único canal de 32 bits representación en coma flotante. El texel (i, j) almacena la cantidad de feromona en el arco entre el nodo *i* y el nodo *j*.

Inicialmente cada texel de esta textura se inicializa a τ_o, donde $\tau_o = \frac{1}{n \cdot T_{\max}}$. Tras cada iteración completa de la colonia de hormigas se realizará una actualización de pistas de feromona, la llamada actualización off-line, modificando consecuentemente esta textura.

Esta textura permanecerá en RAM, ya que no se utiliza directamente en el algoritmo de GPU.

Textura de Heurística

Esta textura tiene una dimensión NxN también de formato R32F. El texel (i,j) almacena la atracción heurística correspondiente a la atracción que tiene el nodo *j* estando actualmente en el nodo *i*.

La información de atracción heurística es constante durante todo el algoritmo. Esta textura se almacena en RAM y cada texel será inicializado de acuerdo a la expresión $\eta_{ij} = \frac{S_j}{c_{ij}}$.

Textura de Atracción

La textura de *Atracción* es la textura que almacena las atracciones entre nodos del grafo según la expresión $\left[\tau_{ij}\right]^{\alpha} \cdot \left[\eta_{ij}\right]^{\beta}$ con α=1 y β=3 como parámetros típicos. Esta textura, al igual que las texturas de *pistas de feromona* y *de heurística de las que derivan sus valores*, tiene una dimensión NxN y se encuentra en memoria de video, ya que va a ser utilizada directamente por los programas de GPU.

Textura de Prohibidos

La textura de nodos prohibidos tiene dimensión HxN con formato R32F. El texel (i, j) indica si la hormiga *i* ha visitado el nodo *j* en el camino que construye.

El hecho de utilizar un formato de coma flotante para sencillamente marcar si un nodo ha sido visitado o no, viene impuesto por las capacidades y limitaciones de las GPU actuales, como se comentará más adelante en el apartado de *Implementación*.

Proceso Algorítmico

Se ha visto como el "*algoritmo"* hace uso de capacidades de la GPU para resolver el problema. Así que el algoritmo constará de secciones que se ejecutan en la CPU y de secciones que se ejecutan en la GPU. Se puede considerar la sección de CPU como una sección conductora y cargadora de datos, mientras que las secciones de GPU se pueden considerar como secciones de cálculo. Asimismo la sección de CPU es como una sección *maestra* y las secciones de GPU son como secciones *esclavas*. Bajo esta óptica se enfoca la descripción algorítmica, en la que el programa de CPU coordina las operaciones a llevar a cabo en la GPU para aprovechar las capacidades ofrecidas por ésta.

Para la sección de CPU se han distinguido 3 fases que vienen expresadas en la siguiente descripción:

```
Algoritmo GPU-VERTEX-OCH-OP (N, H, G, q0, α, β, ρ, Tmax, Nodoini, Nodofin)
Principio
[Fase 1: Inicialización]
Textura de Grafo
```

```
Textura de Atracción, de Feromona y Heurístico
Textura de Prohibidos
Búffer de Vértices
Loop
[ Fase2: Obtención de caminos ]
Para it = 1 hasta N
        [ Fase 2a: Paso de construcción de caminos ]
        Establecer Parámetros Requeridos por los programas de GPU.
        Establecer Texturas de Lectura: Textura de Grafo, de Atracción, de Prohibidos y de Tour
        Establecer la Textura de Rendering: Textura de Tour
        Ejecutar Programas GPU Fase 2a
        [ Fase 2b: Actualizar la lista de nodos prohibidos ]
        Establecer Parámetros Requeridos por los programas de GPU.
        Establecer Texturas de Lectura: Textura de Tour
        Establecer la Textura de Rendering: Textura de Prohibidos
        Ejecutar Programas GPU Fase 2b
FPara
[Fase 3: Obtener el mejor camino y actualización de Feromona]
Determinar el mejor camino hasta el momento
Actualización off-line de Feromona
Reinicializar Texturas para la siguiente tanda de H hormigas
Continuar Loop hasta alcanzar criterio de parada;
FAlgoritmo
```

Del pseudo-código anterior se desprende que la fase 2, de obtención de caminos por parte de la hormigas de la colonia, se ejecuta en la GPU, mientras que la fase 1, de inicialización, y la fase 3, de reinicialización y obtención de la mejor solución, se ejecutan en la CPU.

Dado que en el apartado anterior se han sentado las bases de los resultados que deberían obtenerse en las texturas para la fase 2, y que en la descripción del algoritmo anterior se ha establecido dónde se enmarcan estos programas de GPU, se va a proceder a la descripción algorítmica de los mismos.

Programas de GPU

Como ya se explicó, la GPU dispone de dos procesadores, el procesador de vértices y el procesador de fragmentos. Así que cada fase de GPU (las correspondientes a la fase 2a y la fase 2b) tendrá sus programas de GPU que vienen descritos a continuación. Con miras a la comprensión, y con el objetivo de reducir la complejidad de la descripción algorítmica, se ha tratado de realizar una descripción de tan alto nivel como se ha podido. No obstante, en el anexo B están disponibles los listados correspondientes a los programas implementados en HLSL.

El programa de vértices de la fase 2a se encarga de transformar los vértices, ubicándolos en el espacio según la regla probabilista. El programa de fragmentos de la fase 2a se encarga de procesar los fragmentos provenientes de los vértices transformados que no han sido descartados, y asignar información que se requerirá en la textura de Tour.

El programa de vértices de la fase 2b transforma los vértices que corresponden a los nodos visitados por las hormigas en el paso de construcción de caminos y descarta el resto. Por su parte, el programa de fragmentos de la fase 2b procesa los fragmentos que no han sido descartados para marcar en la textura *Prohibidos* la visita de los nodos.

```
Programa Vértices Fase2a
Entrada:
-Vértice: codifica el nodo del grafo y a la hormiga que representa
Salidas:
-BeneficioAc: score ac. del camino en construcción
-CosteAc: coste ac. del camino en construcción
-Semilla: tendrá la semilla a propagar
-Desactivar: si la hormiga debe ser desactivada
-NodoVisitado: nodo del vértice transformado
Algoritmo:
Obtener información de la it. anterior para la hormiga que representa el vértice
        desde textura de Tour: ultimoNodo, q, Semilla, BeneficioAc, CosteAc,
Activa
Obtener si el nodo representado por el vértice está marcado como visitado en
textura Prohibidos
Si (la hormiga está activa y el nodo no ha sido visitado)
 Entonces
        Si (visitando el nodo excedo) Tmax Entonces        Cerrar el camino
por restricción
        Sino
                Si q es SI Entonces
                        Ajustar Z según explotación
                Sino
                        Ajustar Z según exploración
Sino
        Si nodo del vértice es nodo inicial Entonces        Propagación del
camino ya construido
        Sino
                Ajustar Z fuera del far plane
Fin Programa
Programa de Fragmentos Fase 2a
Entradas:
-BeneficioAc: score ac. del camino en construcción
```

-CosteAc: coste ac. del camino en construcción

-Semilla: tendrá la semilla a propagar

-Desactivar: si la hormiga debe ser desactivada

-NodoVisitado: nodo que codifica el fragmento

Salidas:

- Fragmento con Color codificando: nuevoBeneficioAc, nuevoCosteAc, nuevaSemilla, nuevoNodoVisitado, q, Activa

Algoritmo:

```
Si Desactivar es NO Entonces
        Si NodoVisitado es el Nodo Final Entonces
                Activa = NO;
        Sino
                Activa = SI;
q = Determinar si para el próximo paso de construcción de camino hay que realizar explotación o exploración;
Sino
        Activa = NO;
Si CosteAc > Tmax Y no es solución propagada Entonces
        Aplicar el score de Ramalinho y Serra
Calcular nuevaSemilla para el generador de números aleatorios;
Codificar en el color de salida: nuevoBeneficioAc, nuevoCosteAc, nuevaSemilla, nuevoNodoVisitado, q, Activa
Fin Programa
```

Programa Vértices Fase 2b

Entradas:

-Vértice: codificando el nodo del grafo y a la hormiga que representa

Salidas:

-Vértice Transformado: codificando el vértice ultimoNodo en caso de ser visible

Algoritmo:

Obtener la información de la fase anterior de la hormiga que representa el vértice desde la textura de Tour

Si el ultimoNodo es el nodo del vértice Entonces

```
        Ajustar Z para que el vértice sea visible
Sino
        Ajustar Z fuera del far plane
Fin Programa
Programa Fragmentos Fase 2b
Entrada:
-Fragmento correspondiente al último nodo visitado
Salida:
-Fragmento con color que indica que el nodo ha sido visitado
Algoritmo:
Color = color de visitados;
Fin Programa
```

6.2.3. Implementación

La implementación realizada se ha escrito en Microsoft Visual C++ 7.0 utilizando Microsoft DirectX 9 como API 3D. Los programas de GPU implementados han sido escritos en HLSL, High-Level Shader Language, que es un lenguaje de alto nivel para GPU que incorpora Microsoft DirectX.

En la implementación del algoritmo presentado deben tenerse en cuenta algunas consideraciones importantes para hacer que éste funcione adecuadamente. Las principales consideraciones vienen resumidas a continuación:

- Una textura no puede ser objetivo de la renderización y leída en la misma fase de ejecución de programas de GPU.

Por ello, la textura de *Tour* no está implementada como una única textura sino que ha sido dividida en dos texturas: *Even Texture* y *Odd Texture*. Estas dos texturas tienen dimensión $H \times \frac{N}{2}$. La textura *Even* almacenará las columnas originales pares de la textura de Tour, mientras que la textura *Odd* almacenará las columnas originales impares de la textura de Tour. De esta forma se podrá leer la textura Even mientras se escribe en la textura Odd, y viceversa, superando la limitación hardware existente.

- El formato de las texturas del procesador de vértices es de coma flotante.

Esta restricción justifica que la textura de *Prohibidos*, pese a ser una textura que se utiliza para marcar lógicamente los nodos que han sido ya visitados en los

tours de las hormigas, no se puede utilizar un formato que consuma menos memoria.

- Cálculo de coordenadas de textura

Típicamente, en nuestro caso, un punto en una textura se referencia por coordenadas definidas en el rango continuo [0.0, 1.0]. En ningún momento se realiza una indexación por coordenadas del texel al que se quiere acceder, así que será necesario realizar una conversión entre las supuestas coordenadas del texel y su posicionamiento en el espacio de textura 2D.

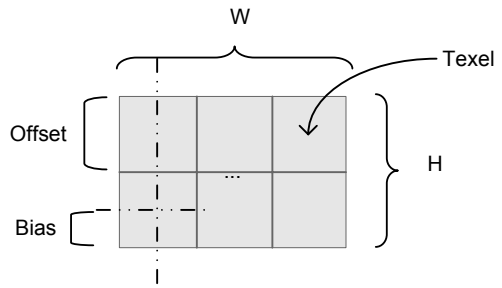


Figura 59. Cálculo de coordenadas de textura

Supongamos el caso de la figura 16. Se dispone de una textura de dimensión HxW texels. Concretamente, H=2 y W=3.

Si se quisiera acceder al texel señalado, (0, 2), entonces se requiere transformar esa coordenada de texel en coordenadas de textura en el dominio real [0, 1]x[0, 1]. Para ello se emplea además del desplazamiento por texel, indicado por *offset*, un desplazamiento dentro del texel que marca la disposición del centro del texel, indicado por *bias*. Por tanto, las expresiones asociadas son las siguientes:

$$offset_H = \frac{1}{H}; bias_H = \frac{offset_H}{2}; offset_W = \frac{1}{W}; bias_W = \frac{offset_W}{2}$$

Estas expresiones nos permiten construir la función que obtiene las coordenadas de textura a partir de las coordenadas lógicas del *texel*:

$$ctexel2ctexture(i, j) = (i \cdot offset_H + bias_H, j \cdot offset_W + bias_W)$$

La expresión es algo diferente a la típica expresión que se utilizaría para acceder a componentes de una array tipo C.

$$coordC(i, j) = \left(i \cdot \frac{1}{H}, j \cdot \frac{1}{W} \right)$$

Nótese, por tanto, que el desplazamiento marcado por *bias*, que permite que se referencie en el centro del texel deseado, es necesario. Para poner esta

necesidad de manifiesto, supongamos que se desea acceder al texel (1,1). La figura 17 muestra cuál sería el efecto de utilizar una expresión u otra para el cálculo de coordenadas de textura.

$$coordC(1,1) = (0.5,0.33)$$

$$ctexel2ctexture(1,1) = (0.75,0.49)$$

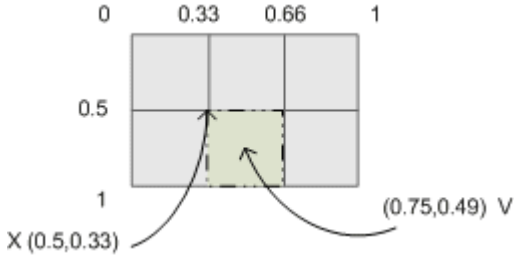


Figura 60. Problemas de las coordenadas de textura

Si utilizáramos coordC, debido a problemas de precisión, y de la API 3D, podríamos estar accediendo de forma aleatoria a cualquiera de los cuatro texels que envuelven a la coordenada de textura calculada, que evidentemente no es el efecto que buscamos.

- La GPU no dispone de un generador de números pseudo-aleatorios.

Como la GPU no dispone de un generador de número aleatorios se implementó un generador en HLSL y en C para utilizarlos desde la GPU y desde la CPU respectivamente.

Para ello se implementó un generador congruencial lineal que se caracteriza por tener la siguiente estructura (Fishman, 1974):

$$X' = a \cdot X + c \bmod m$$

y que es capaz de generar secuencias de números pseudo-aleatorios a partir de una semilla inicial distinta de 0.

En nuestro caso, c será 0, con lo que este tipo de generadores se conocen como congruenciales lineales multiplicativos. La calidad del generador dependerá de la elección del parámetro multiplicador *a*, y del parámetro modulo *m*, de forma que, además de un largo periodo de la secuencia, se consiga independencia y uniformidad entre los números obtenidos.

Existen propiedades matemáticas (Fishman, 1974) que de forma teórica se propone que han de cumplir los parámetros, *a*, *c* y *m*, para que el generador congruencial asociado tenga buenas cualidades. Dado que nuestro objetivo principal no es el diseño de un generador de números aleatorios, y dado que encontrar una combinación de parámetros que cumplan con las propiedades

matemáticas propuestas en la bibliografía no es una tarea trivial, se procedió a tratar de asignar valores siguiendo en la medida de lo posible las condiciones impuestas por las propiedades y a realizar entonces varias pruebas experimentales diseñadas para aceptar/rechazar la hipótesis de uniformidad y aleatoriedad de muestras generadas con el generador implementado.

La GPU implementa números reales de simple precisión. Recordemos que había que almacenar la secuencia de números aleatorios generados en un canal de 32 bits de coma flotante para poder generar el siguiente número de la secuencia. Dado que el número aleatorio es un número entero, solo podremos considerar números comprendidos entre 0 y 2^{24}, ya que la mantisa del número real es de 24 bits y no todos los números enteros más grandes serán representables. Siguiendo algunas de las propiedades matemáticas de los generadores, establecemos *m* al número entero más grande representable que sea primo, 2^{24}-3.

El generador implementado es $X' = 16807 \cdot X \bmod\left(2^{24} - 3\right)$ que es similar al generador congruencial minimal estándar, utilizado generalmente en máquinas de 32 bits, pero con el cambio del módulo ya que solo se dispone de 24 bits para representar enteros.

La expresión del generador congruencial puede tener problemas si no se puede asegurar que los resultados intermedios estén acotados por m. Debido a este problema, el generador se implementó siguiendo el método de Schrage, explicado en (Ríos, 1997), que consiste en obtener una expresión equivalente que no presenta problemas de representación.

Este generador fue evaluado con el test de rachas de monotonía, el test de Chi-Cuadrado, y el test de Kolmogorov-Smirnov, todos ellos de gran tradición y popularidad en la bibliografía (Banks, 2001). Para las sucesivas muestras diferentes, de 10000 números generados cada una, fue satisfactoria la aceptación de la hipótesis de uniformidad e independencia con α=5%. Este resultado lo consideramos suficientemente válido por no tener nuestro generador fines criptográficos, y pese a la existencia de otros tests más potentes que probablemente podrían rechazar la hipótesis de independencia y uniformidad considerada.

6.2.4. Resultados Experimentales

Al igual que el algoritmo presentado en el capítulo anterior, GRID-OCH-OP, se habilitó al algoritmo GPU-VERTEX-OCH-OP para poder adoptar opcionalmente la aplicación de la búsqueda local relajada o intensiva, y los parámetros α, β, q_0 y ϱ, han sido fijados con los siguientes valores $\alpha = 1$, $\beta = 3$, $q_0 = 0.2$ y $\varrho = 0.9$.

En una primera generación de experimentos se observó cómo el hecho de aplicar búsqueda local intensiva no mejoraba considerablemente la calidad de las soluciones obtenidas, y sin embargo el tiempo de ejecución se disparaba notablemente. Así que, atendiendo a este resultado, se procedió a comparar exclusivamente el algoritmo con dos variantes: sin búsqueda local intensiva, y con búsqueda local relajada.

Para ello se utilizaron las instancias de los grafos de densidad aleatoria utilizados en los experimentos de GRID-OCH-OP de tallas 1000 y 3000 nodos. Todos los experimentos corrieron en un Pentium IV a 2.4 GHz, 512MB de RAM y con una GPU nVidia GeForce 6600GT. Los experimentos definitivos se realizaron 20 veces y los resultados presentados son fruto del promediado de los mismos.

En primer lugar se comparan las dos variantes del algoritmo, el que se aplica la búsqueda local relajada frente al que no aplica búsqueda local. Las gráficas muestran que la diferencia entre el tiempo de resolución entre una y otra variante no es significativo. Pero el hecho de aplicar búsqueda local relajada permite obtener soluciones de mayor calidad, aunque en algunos casos no sea una mejora muy importante.

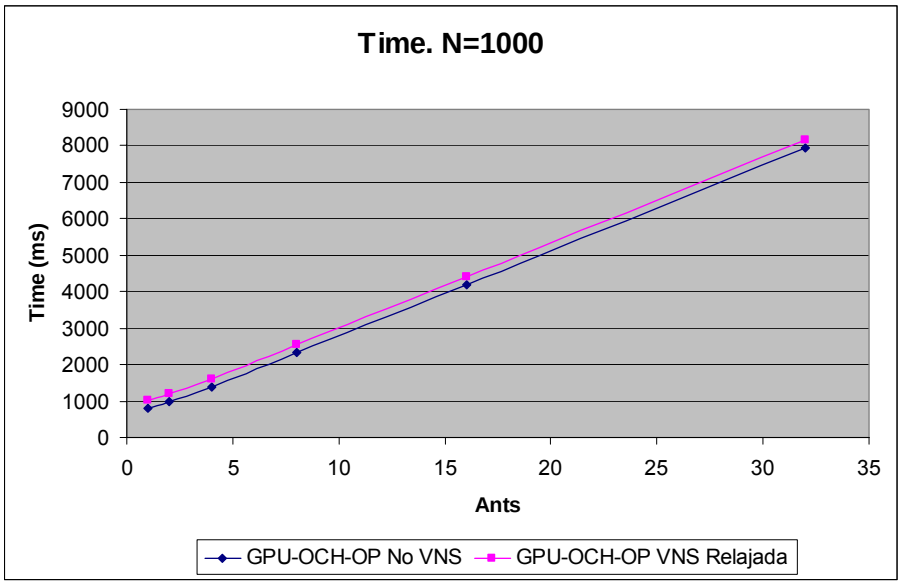


Figura 61. Variantes de GPU-VERTEX-OCH-OP: Tiempo para N=1000

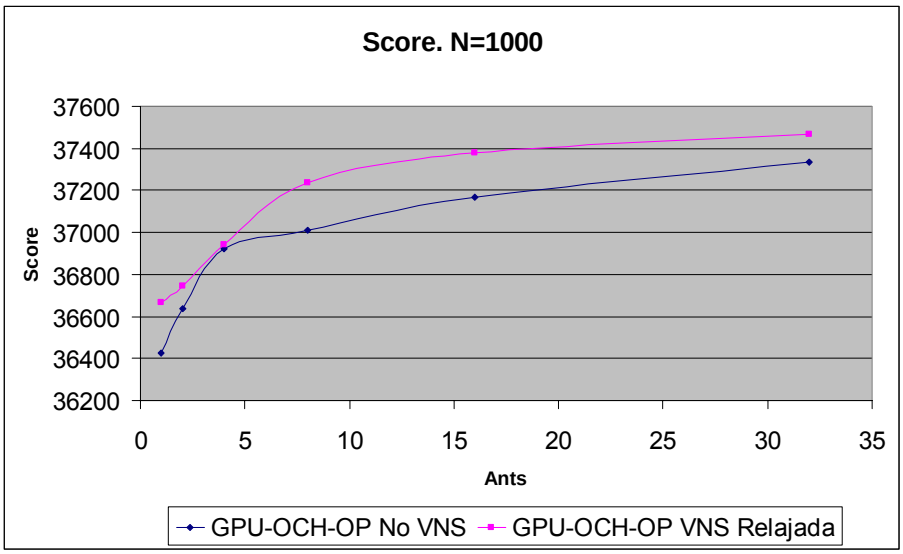


Figura 62. Variantes de GPU-VERTEX-OCH-OP: Score para N=1000

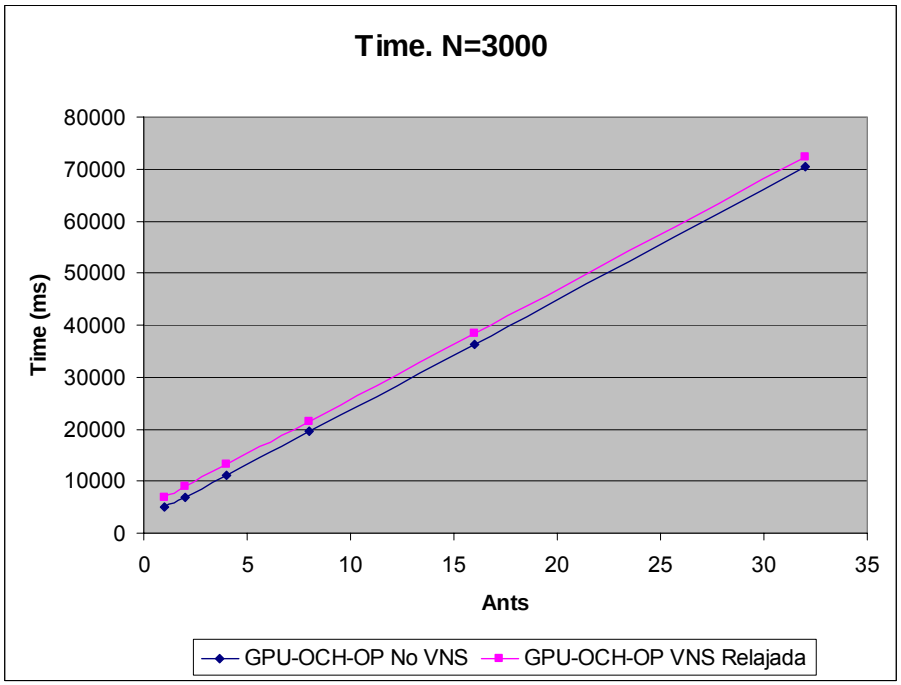


Figura 63. Variantes de GPU-VERTEX-OCH-OP: Time para N=3000

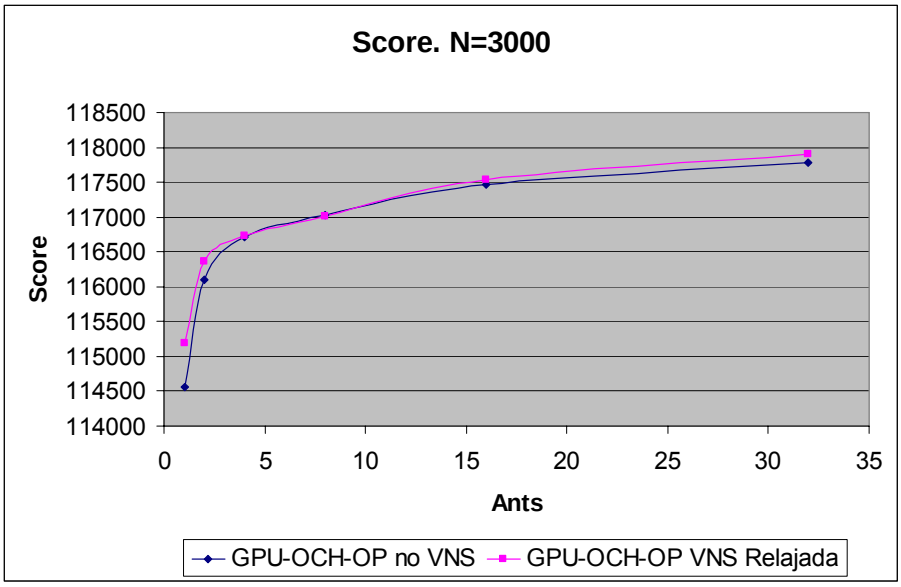


Figura 64. Variantesde GPU-VERTEX-OCH-OP: Score para N=3000

Se observa cómo el coste temporal se comporta O (H·N), y cómo al variar H manteniendo N constante efectivamente se comporta de forma lineal con respecto a H.

En cuanto a la calidad de la solución, el hecho de intervenir varias hormigas en la resolución deja patente cómo la calidad de la solución mejora notablemente hasta con 16 hormigas. Sin embargo, cuanto mayor sea la población de hormigas de la colonia, mayor será el tiempo de resolución.

A continuación se compara el algoritmo propuesto en este capítulo frente al algoritmo GRID-OCH-OP, aplicando para ambos búsqueda local relajada.

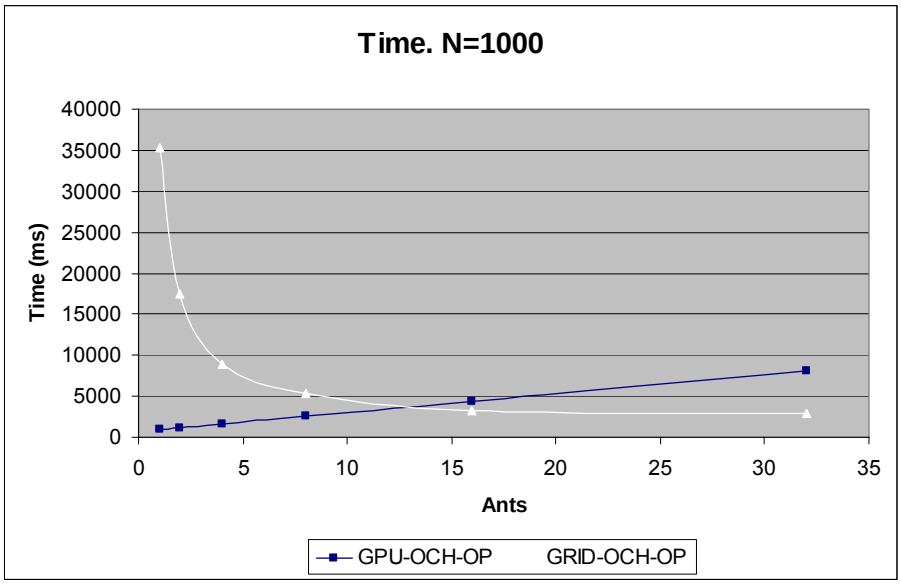


Figura 65. GPUvs. GRID: Tiem para N=1000

El tiempo de ejecución de GRID-OCH-OP, a medida que se incrementa el número de hormigas que participan en el experimento disminuye considerablemente, como ya se explicó en el capítulo anterior.

Sin embargo, los tiempos de ejecución de GPU-VERTEX-OCH-OP son más estables y más predecibles, teniendo un crecimiento controlado. Con 1, 2, 4, 8 y 16 hormigas estamos obteniendo un tiempo de resolución inferior a los 5 segundos, mientras que en GRID-OCH-OP se requieren un mayor número máquinas dedicadas. Es destacable cómo con una única hormiga, GPU-VERTEX-OCH-OP consume en torno a 1 segundo, mientras que GRID-OCH-OP consume aproximadamente 35 segundos.

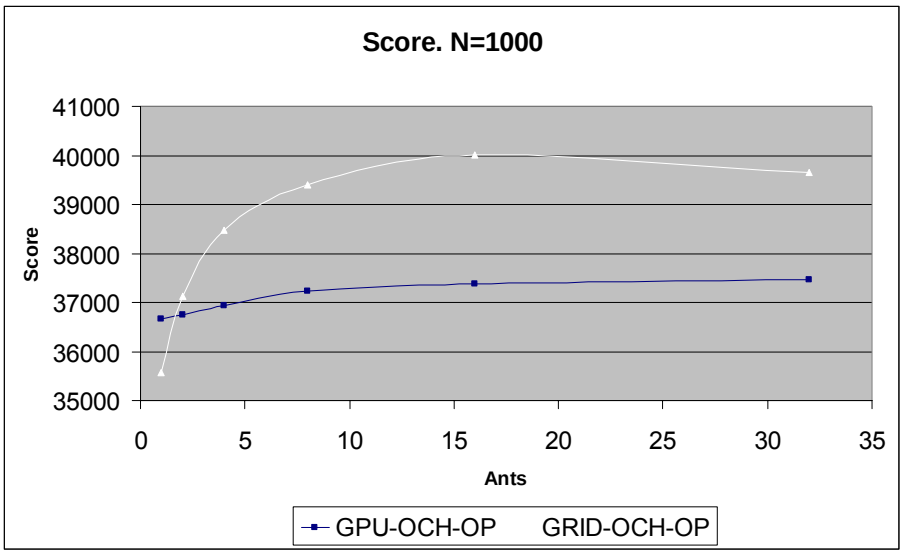


Figura 66. GPU vs. GRID: Score para N=1000

En cuanto al *score* de las soluciones obtenidas para las instancias de 1000 nodos, se observa como GPU-VERTEX-OCH-OP obtiene un cierto crecimiento moderado al incrementar el número de hormigas. El algoritmo GRID-OCH-OP obtiene soluciones con hasta un máximo del 7% mejor *score* que las obtenidas por GPU-VERTEX-OCH-OP en algunos casos para el grafo de 1000 nodos.

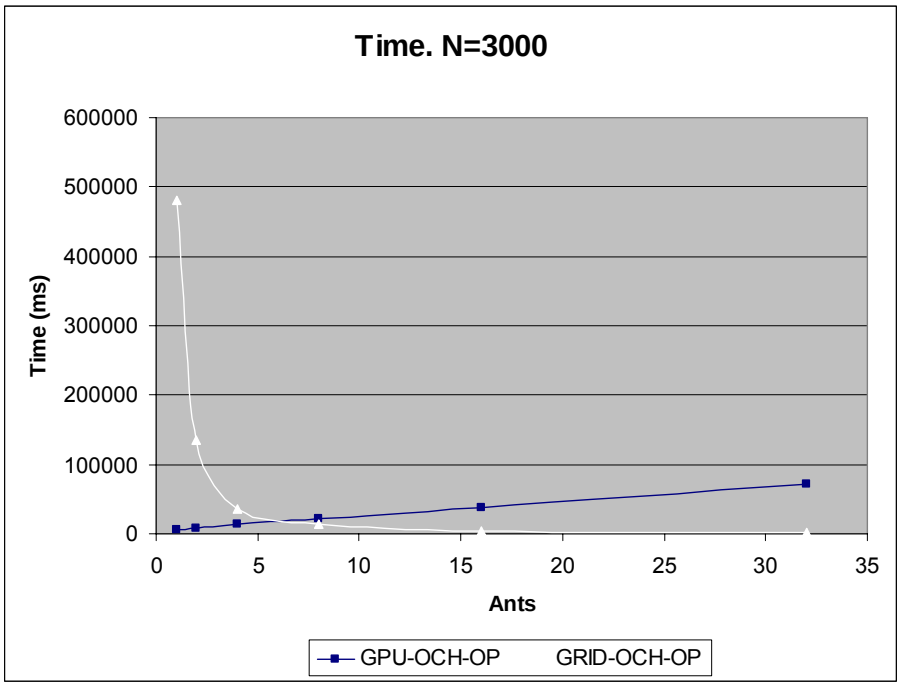


Figura 67. GPU vs. GRID: Timer para N=3000

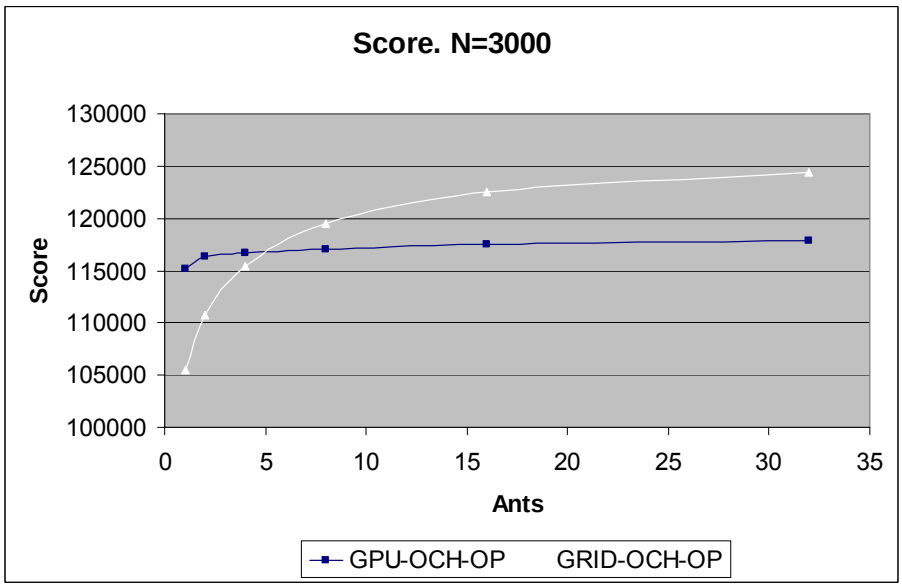


Figura 68. GPU vs. GRID: Score para N=3000

En esta comparación de los resultados de las instancias de 3000 nodos, se observa claramente como el algoritmo GRID-OCH-OP deja de nuevo patente sus buenas propiedades. Sin embargo, el hecho de que este algoritmo necesite

hacer uso de más hormigas para decrementar el espacio de búsqueda y así obtener soluciones de mayor calidad en un tiempo inferior, deja entrever las buenas propiedades del algoritmo GPU-VERTEX-OCH-OP.

El algoritmo GRID-OCH-OP corre sobre un grid de PCs, mientras que el algoritmo GPU-VERTEX-OCH-OP está corriendo sobre una única máquina. Además se observa que con 1, 2 o incluso con 4 hormigas el algoritmo GPU-VERTEX-OCH-OP tiene un coste temporal inferior o muy inferior en relación al que tiene GRID-OCH-OP. Es más, yendo más lejos, hasta con 8 y 16 hormigas, según el caso, las soluciones y los tiempos de ejecución del algoritmo GPU-VERTEX-OCH-OP es totalmente competitivo en relación al algoritmo GRID-OCH-OP.

Concretamente, mientras que GPU-VERTEX-OCH-OP con 1 hormiga obtiene aproximadamente un tiempo de ejecución de 6 segundos y un score de 115200, GRID-OCH-OP requiere algo más de 8 minutos para obtener aproximadamente 105500 de score.

Bajo el supuesto en el que se dispusiera de un GRID de 32 PCs equipados con una GPU como la que hemos utilizado en la experimentación, y que fuera implementado el algoritmo GRID-GPU-VERTEX-OCH-OP que aunaría la capacidad de GPU-VERTEX-OCH-OP con la de GRID-OCH-OP, se propone la resolución del problema de la orientación para una instancia de 100000 nodos.

Bajo este supuesto, dado que la generación de clusters de la instancia del problema es, en cierta medida, homogénea, podemos considerar que cada *nodo del grid* debe resolver una instancia de aproximadamente 3000 nodos. Así, el tiempo de ejecución orientativo estimado sería, de forma aproximada, de 9 segundos para el hipotético algoritmo GRID-GPU-VERTEX-OCH-OP frente a los 809 segundos del GRID-OCH-OP actual.

6.3. Algoritmo Orientado a Fragmentos para la Resolución de OCH mediante GPU

El objetivo que se persigue en el diseño de este segundo algoritmo de GPU no es otro que el de conseguir reducir el coste temporal de ejecución. Esta mejora se pretende obtener a costa de aprovechar de forma más eficaz las capacidades del procesador gráfico. Ello, tras ver como el comportamiento de GPU-Vertex-OCH-OP es muy dependiente del número de vértices, pasa por reducir el procesamiento de vértices y orientar el cálculo en la GPU al procesamiento de fragmentos. Este tipo de procesamiento se consigue utilizando primitivas gráficas que generen un mayor número de fragmentos en relación al número de

vértices que las componen. Además este procesamiento permite hacer uso de la interpolación que el rasterizador puede aplicar y que no estaba siendo aprovechada, y en definitiva aprovechar la mayor capacidad de cómputo del procesador de fragmentos ya que en la GPU se disponen de más fragment-pipelines que de vertex-pipelines.

6.3.1. Aproximación intuitiva al algoritmo GPU-Fragment-OCH-OP

Al igual que se hizo con el algoritmo GPU-Vertex-OCH-OP, se presentará de forma aproximada las ideas principales de funcionamiento del nuevo algoritmo, GPU-Fragment-OCH-OP, haciendo especial hincapié en sus novedades y sus diferencias más importantes.

Estructuras de datos

En esencia, el algoritmo GPU-Fragment-OCH-OP dispone de las mismas componentes de las que disponía el algoritmo GPU-Vertex-OCH-OP, salvo el conjunto de vértices, que se ve modificado debido a que en este nuevo algoritmo se cambian las primitivas gráficas a emplear.

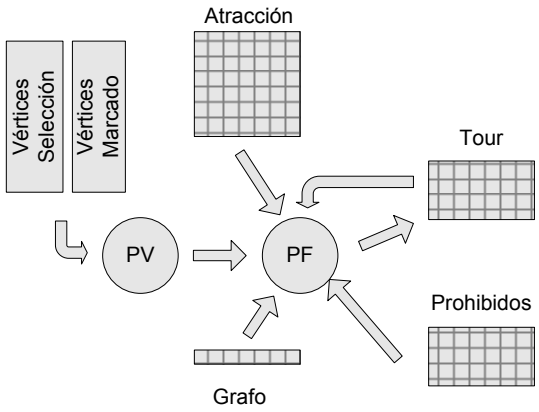


Figura 69. Componentes del algoritmo GPU-Fragment-OCH-OP

En la Figura 69 se muestran los componentes del algoritmo GPU-Fragment-OCH-OP. Si se compara con la **¡Error! No se encuentra el origen de la referencia.**, en la que se mostraban los componentes del algoritmo GPU-Vertex-OCH-OP, se observa como las principales diferencias son la existencia de dos conjuntos de vértices y que el consumo de datos almacenados en texturas se realiza en el procesador de fragmentos (PF).

Al igual que en el algoritmo GPU-Vertex-OCH-OP, el nuevo algoritmo también diferenciará las dos fases o subfases, la de selección de nodos y la de marcado como visitados de los mismos, dentro de la construcción de caminos. En virtud de este hecho y dado que para llevar a cabo estas dos fases no se

emplean las mismas primitivas gráficas surge la necesidad de definir dos conjuntos de vértices.

El conjunto de vértices, referenciado como "Vértices de Selección" se emplearán en la fase de selección de los nodos a visitar, mientras que el conjunto referenciado como "Vértices de Marcado" se emplearán en la fase de marcado de nodos visitados.

En cuanto al resto de estructuras de datos no existen novedades importantes que deban ser expuestas aquí. Esencialmente, la información y su significado será el mismo que en el algoritmo anterior, y por tanto, las diferencias existentes serán expuestas en el apartado de Estructuras de Datos del punto 6.3.2.

Construcción de caminos

Como ya se ha adelantado la fase de construcción de caminos también se concibe como un proceso iterativo que alterna la ejecución de una subfase de selección de nodos a visitar y de una subfase de marcado de los nodos visitados.

A continuación se expone cómo se llevan a cabo estas subfases.

Selección de nodos a visitar: Explotación y Exploración

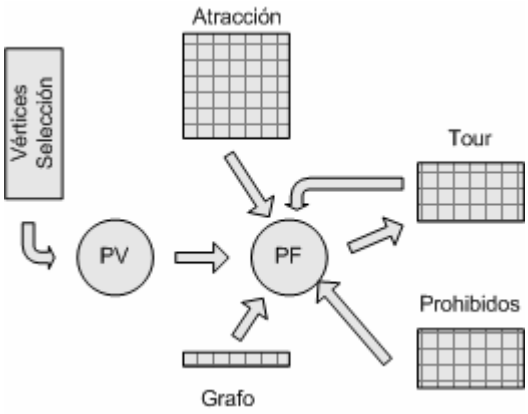


Figura 70. Componentes de GPU-Fragment-OCH-OP: Selección de nodos

En la Figura 70 se muestra el esquema que representa la fase de selección de nodos. A cada ejecución de esta fase, cada hormiga seleccionará el siguiente nodo a visitar. Para ello, cada hormiga deberá consultar la información del último nodo visitado y del camino construido desde la textura de Tour y ponderar la posibilidad de visitar cada nodo que aún no haya sido visitado, obteniendo finalmente el nodo seleccionado.

En esta fase se utiliza un modelo de cámara paralelo-ortográfico que define un volumen HxNxN. El conjunto de vértices "Vértices de Selección" está compuesto de 2·N vértices. Cada par de vértices se utilizará para formar una línea, trazándose un total de N líneas. Cada línea representará la intención de visitar el nodo del grafo asociado a la línea por parte de cualquiera de las hormigas.

La disposición de las líneas durante la ejecución de esta fase en el espacio es como muestra la Figura 71. Los vértices que forman las líneas se ubicarán en la coordenada *x* que corresponda al número de iteración de la fase de construcción de caminos. Las líneas deberán ser paralelas al *eje y*, y su profundidad puede ser cualquiera siempre y cuando quede dentro del volumen visible por la cámara. La ubicación de los vértices se realizará por aplicación de la transformación correspondiente mediante un programa de vértices que se ejecuta en el procesador de vértices.

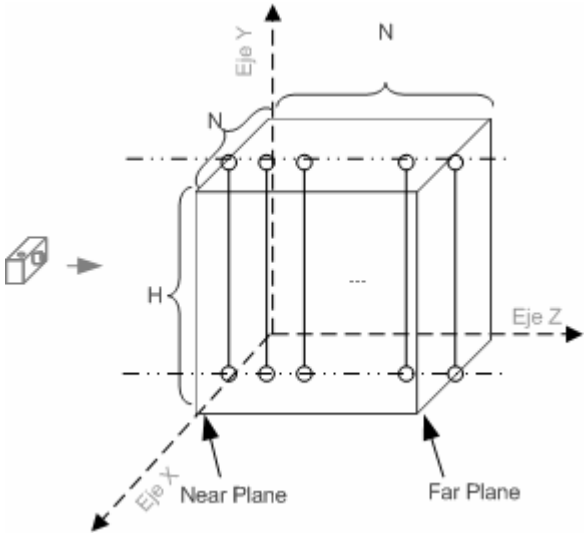


Figura 71. Espacio de recortado y disposición de las líneas

Tras la ubicación de los vértices y el ensamblado de las líneas a partir de los mismos, se producirá su fragmentación, tal y como muestra la Figura 72. Cada fragmento representa la posibilidad de que una hormiga determinada visite un determinado nodo del grafo. Estos fragmentos serán los que entren al procesador de fragmentos y sobre los que se ejecutará el programa de fragmentos que se encargue de aplicar la regla probabilista de selección de nodos.

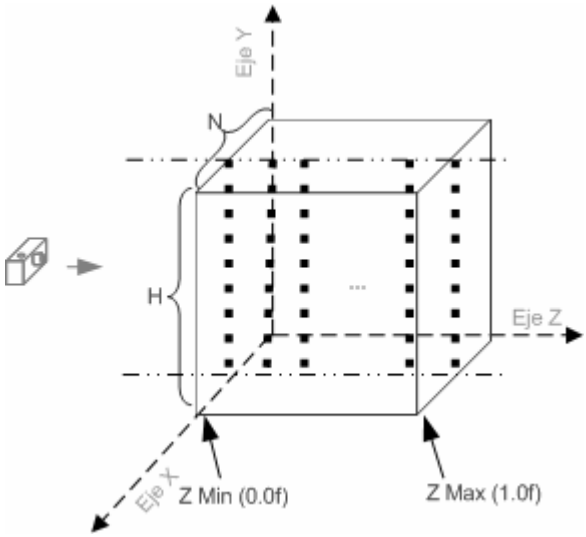


Figura 72. Fragmentación de líneas en GPU-Fragment-OCH-OP

Se aplicará la misma regla probabilista de selección de nodos que se aplicó en el algoritmo GPU-Vertex-OCH-OP. Lo único que hay que tener en cuenta es que el rango de la regla debe ser convenientemente escalada al rango real continuo [0,1].

Para el caso en que una hormiga deba aplicar explotación, la expresión a utilizar sería la siguiente:

$$z = 1.0 - \frac{[\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta} - MinAtrac}{MaxAtrac - MinAtrac}$$

La expresión anterior sencillamente contiene el escalado al intervalo unitario, y posteriormente se calcula su complemento ya que se pretende que a mayor atracción, menor sea el resultado de la función que pondera la explotación.

Para en el caso que una hormiga deba aplicar exploración, la expresión a utilizar quedaría como sigue:

$z = \frac{\left| r - \left([\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta} \right) \right|}{MaxAtrac - MinAtrac}$, siendo r un número pseudo-aleatorio distribuido uniformemente en el intervalo continuo [MinAtrac, MaxAtrac]. Esta función pondera la exploración de forma que a mayor distancia entre r y la atracción de visitar el nodo candidato, mayor será el valor de z resultante.

Como cada fragmento representa un par hormiga-nodo, cada fragmento debe aplicar la regla de selección. La estrategia utilizada para realizar la selección se fundamenta en que sólo un fragmento de entre todos los fragmentos ubicados

en la misma posición *x*, *y* del espacio será escrito finalmente en la textura de salida, basándose en la técnica de *Z-buffer* o *Depth Testing*. Dicha técnica consiste en utilizar un *buffer de profundidad* donde se va guardando la profundidad relativa del fragmento que en cada momento es visible y por tanto ese es el fragmento que se escribe en el *frame buffer*. Cuando se procesa un nuevo fragmento, se compara la profundidad del mismo con la profundidad del fragmento que actualmente está ocupando la misma posición *x,* e *y* en el *frame buffer*. Si la profundidad del nuevo fragmento es inferior al del fragmento que hasta el momento ocupaba el *frame buffer* entonces el nuevo fragmento se convierte en el que se escribirá en el *frame buffer*.

Si nos centramos en una iteración de construcción de caminos concreta, y de todos los fragmentos que se producen en la Figura 72, y sólo prestamos atención a los fragmentos de una determinada hormiga, podemos sintetizar el proceso de selección tal y como lo muestra la Figura 73. Sobre los fragmentos de una misma hormiga, que como ya hemos visto difieren en *z*, se ejecuta el programa de fragmentos que aplica la regla probabilista. Como resultado la profundidad relativa de cada fragmento quedará modificada en consecuencia.

Si el nodo al que representa el fragmento ya ha sido visitado, o si la hormiga está inactiva por haber construido ya una solución entonces el fragmento se descarta forzosamente utilizando el denominado concepto de *fragment kill*. Este concepto consiste en eliminar el fragmento de forma rápida gracias a instrucciones de GPU que existen para tal efecto sin necesidad de utilizar *Deph Testing*.

Si el nodo al que representa el fragmento no ha sido visitado todavía entonces se aplica la regla probabilista expuesta. Finalmente de entre todos los fragmentos de la misma hormiga, solo aquel que tenga la profundidad más próxima a 0 será visible y seleccionado, mientras que el resto de fragmentos habrán sido descartados al haber fallado la aplicación del *Deph Test* con respecto al fragmento que sale seleccionado.

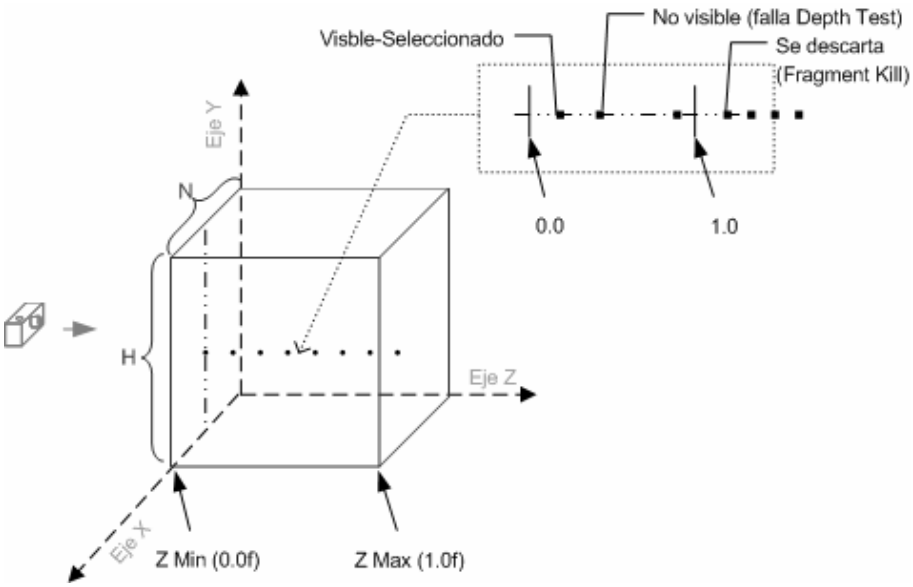


Figura 73. Selección del siguiente nodo a visitar en GPU-Fragment-OCH-OP

Marcado de nodos visitados

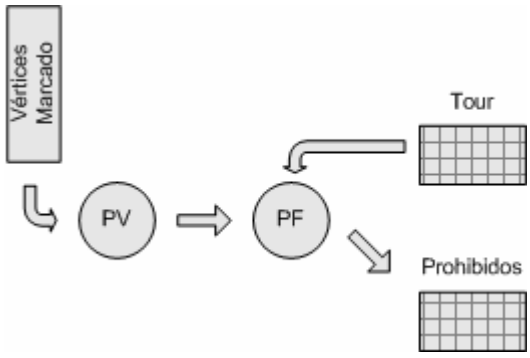


Figura 74. Componentes de GPU-Fragment-OCH-OP: Marcado de visitados

El esquema de la fase de marcado de nodos visitados se muestra en la Figura 74. Esta fase utiliza también un modelo de cámara paralelo-ortográfica que define un volumen de vista HxN donde la profundidad no es relevante y puede ser establecida de forma totalmente arbitraria. El conjunto de vértices "Vértices de Marcado" lo componen tres vértices que se dispondrán en el espacio de la forma mostrada en Figura 75 para formar un triángulo, siendo la relación de los catetos 2H y 2N. La razón de utilizar un triángulo de estas características viene justificada por el hecho de que esta primitiva es la más simple que puede utilizarse para generar los HxN fragmentos que se requieren para esta fase de marcado.

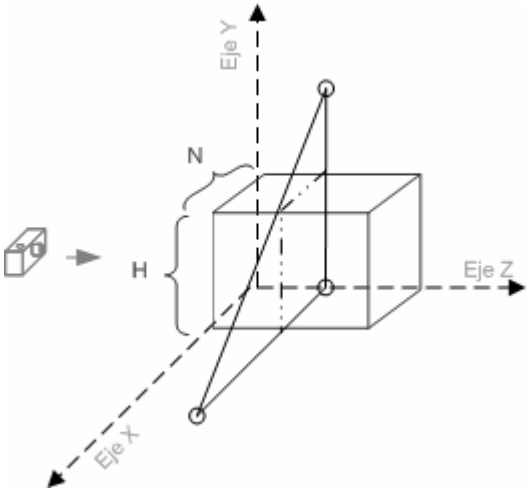


Figura 75. Vértices de Marcado en GPU-Fragment-OCH-OP

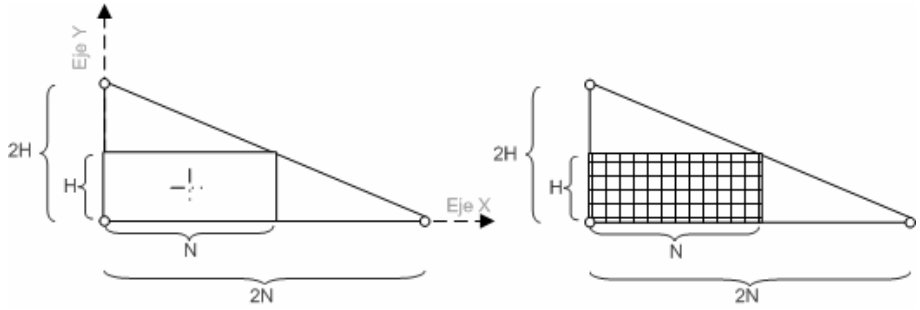


Figura 76. GPU-Fragment: Izq.: Sección de la vista. Der.: Fragmentación

En la ilustración izquierda de la Figura 76 se muestra la sección resultante. La cruz representa el centro del enfoque de la cámara y el rectángulo inscrito en el interior del triángulo marca la zona que será visible por la cámara. Esta zona visible será la que finalmente será fragmentada en $H \cdot N$ fragmentos, como muestra la ilustración derecha de la Figura 76.

Cada uno de los fragmentos generados representa la posibilidad de que cada una de las hormigas haya visitado el nodo que representa el fragmento. Entonces sobre dicho fragmento, que viene con información sobre a qué hormiga se refiere y a qué nodo representa, se ejecuta un programa de fragmentos que se encarga de aceptar o rechazar el fragmento. El fragmento es aceptado, y por tanto se escribe en la textura de visitados, en el caso de que el nodo se corresponda con el nodo seleccionado para la construcción del camino en la textura de tour; en caso contrario, el fragmento es descartado forzando que no se marque como visitado en la textura de visitados.

6.3.2. Algoritmo GPU-Fragment-OCH-OP

En este apartado se presentan con detalle las estructuras de datos que se manejan y se presenta una descripción del proceso algorítmico de GPU-Fragment-OCH-OP.

Estructuras de Datos

Buffer de Vértices

Habrá dos conjuntos de vértices. El conjunto de vértices de selección y el conjunto de vértices de marcado.

El conjunto de *vértices de selección* contiene 2N vértices. Las primitivas gráficas que ensamblan los vértices son líneas rectas. Cada vértice viene caracterizado por codificar: en la *coordenada x*, el número de nodo del grafo al cual se asocia la línea que formará el vértice; en la *coordenada y*, codifica si se trata del vértice "superior" o vértice "inferior" de la línea que debe formar; y en la *coordenada z* no se almacena información relevante. Este conjunto de vértices aprovecha las coordenadas de su posición para codificar información útil, y será un programa de vértices el que se encargue de ubicar adecuadamente los vértices de acuerdo a lo expuesto en la aproximación intuitiva al algoritmo.

El conjunto de *vértices de marcado* contiene tres vértices. Los tres vértices ensamblan una única primitiva: un triángulo. Cada vértice viene caracterizado por codificar: en la *coordenada x*, el valor de la abscisa que debe tener el vértice en el espacio de coordenadas; en la *coordenada y*, el valor de la ordenada que debe tener el vértice en el espacio de coordenadas; en la *coordenada z*, el valor de la profundidad del vértice, establecida de forma arbitraria a 0.1f; Además, estos tres vértices tienen adjunta información en forma de coordenada de textura necesaria para que el rasterizador pueda calcular por interpolación lineal la hormiga y el nodo al que representa cada fragmento.

En la ilustración izquierda de la Figura 77 se muestra cual debería ser valor de las coordenadas de textura que codifican la hormiga y el nodo para que sea interpolado correctamente. En la ilustración de la derecha se muestra cual es el resultado de la interpolación lineal realizada por el rasterizador cuando se produce la fragmentación.

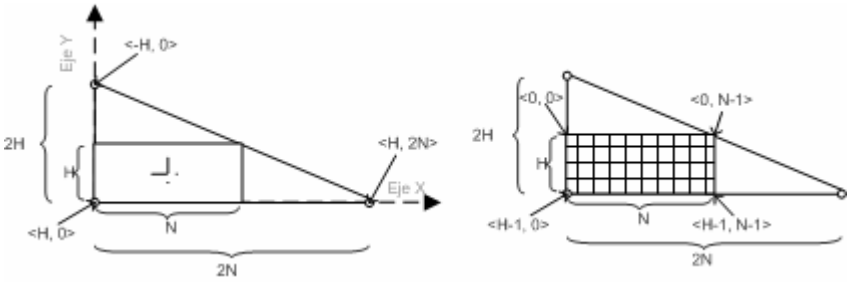


Figura 77. GPU-Fragment: Izq.: Vértices con coordenadas de textura. Der.: fragmentos con coordenadas de textura interpoladas.

Texturas

Las texturas son exactamente las mismas que se utilizaron en el algoritmo GPU-Vertex-OCH-OP, a excepción de la textura de *Tour*, que varía muy ligeramente la información que almacena en los canales de sus texels. La información de cada texel de la textura de tour será (ver Figura 78):

- nodo visitado: el nodo visitado es el número de nodo que ha sido visitado. La diferencia es que no se codifica la q que indicaba si debía hacerse explotación o exploración ya que no es necesario propagar este valor debido a que todos los cálculos relevantes para la hormiga se realizan en el procesador de fragmentos.
- semilla: es una semilla para la hormiga que utilizará como entrada a un generador de números pseudo-aleatorios para determinar si debe hacerse explotación o exploración y calcular cuál es la semilla para el paso siguiente.
- Puntuación acumulada y hormiga inactiva: la puntuación acumulada es la puntuación o beneficio obtenido con el camino que se está construyendo. En el signo se codifica también si la hormiga está activa o inactiva, ya que es posible que ya haya encontrado una solución y que no necesite, por tanto, seguir realizando cálculos.
- Coste acumulado: es el coste del camino que la hormiga está construyendo.

Nodo visitado	Semilla	Score ac. + activa	Coste acum.
R32	G32	B32	A32

Figura 78. Contenido de un texel de la textura de Tour en GPU-Fragment-OCH-OP

Proceso Algorítmico

Al igual que el algoritmo GPU-Vertex-OCH-OP, el algoritmo GPU-Fragment-OCH-OP consiste en un algoritmo iterativo que hace uso de la GPU para realizar determinadas tareas. En realidad, los dos algoritmos de GPU,

genéricamente GPU-OCH-OP, hacen lo mismo y varía el cómo lo hacen. Mientras que GPU-Vertex-OCH-OP utiliza puntos como primitivas gráficas orientando la computación al procesamiento de vértices, el algoritmo GPU-Fragment-OCH-OP utiliza líneas como primitivas gráficas orientando la computación al procesamiento de fragmentos.

La sección que se encarga de conducir el algoritmo y de cargar los datos a la GPU que como ya se discutió en la exposición de GPU-Vertex-OCH-OP se ejecuta en CPU sigue el mismo esquema que se presentó entonces. Allí se distinguieron tres fases: Fase 1, de inicialización; Fase 2, de obtención de caminos; Fase 3, de obtener el mejor camino y actualizar feromonas. De la Fase 1, solo varía la inicialización del buffer de vértices, que en este caso será necesario inicializar los dos buffer de vértices requeridos en el proceso. La Fase 2, realiza la invocación a los programas de GPU que realizarían la construcción de caminos, y la Fase 3 permanece inalterada.

Programas de GPU

Se va a proceder a presentar una descripción de los programas de GPU correspondientes a la fase 2. A la subfase de selección de nodos se referencia como Fase 2a, mientras que a la subfase de marcado de nodos visitados se referencia como Fase 2b.

Programa Vértices Fase 2a

Entrada:

- Vértice: codifica el nodo que representa y si es el vértice “superior” o “inferior” de la línea

Salidas:

Vértice: posición transformada (tal y como se comentó en la aproximación intuitiva)

Coordenada de Textura: codifica el nodo al que está asociado el vértice y contiene información necesaria para obtener por interpolación la hormiga

Algoritmo:

    Aplicar la transformación del vértice

Fin Programa

Programa Fragmentos Fase2a

Entrada:

-Fragmento: codifica el nodo del grafo y a la hormiga que representa

Salidas:

-BeneficioAc: score ac. del camino en construcción

-CosteAc: coste ac. del camino en construcción

-Semilla: tendrá la semilla a propagar

-Desactivar: si la hormiga debe ser desactivada

-NodoVisitado: nodo del vértice transformado

- Z: profundidad del fragmento ponderando su atracción

Algoritmo:

Obtener información de la it. anterior para la hormiga que representa el vértice desde textura de Tour: ultimoNodo, Semilla, BeneficioAc, CosteAc, Activa

Obtener si el nodo representado por el vértice está marcado como visitado en textura Prohibidos

Si (la hormiga está activa y el nodo no ha sido visitado)

Entonces

Si (visitando el nodo excedo) Tmax Entonces Cerrar el camino por restricción

Sino

Si hormiga hace explotación Entonces

Ajustar Z según explotación

Sino

Ajustar Z según exploración

Sino

Si nodo del vértice es nodo inicial Entonces Propagación del camino ya construido

Sino

Matar el fragmento

Fin Programa

Programa Vértices Fase 2b

Entrada:

- Vértice: codificando su posición espacial. Como coordenada de textura viene codificada la información necesaria para la interpolación de nodo y hormiga.

Salidas:

- Vértice: con posición transformada (tal y como se comentó en la aproximación intuitiva) y como coordenada de textura codifica información para la interpolación del nodo y hormiga

Algoritmo:

Aplicar la transformación del vértice

```
Fin Programa
Programa Fragmentos Fase 2b
Entradas:
-Fragmento: codificando el nodo del grafo y a la hormiga que representa
Salidas:
-Fragmento: codificando el color de "marcado"
-Z: profundidad del fragmento
Algoritmo:
Obtener la información del último nodo visitado (ultimoNodo) de la fase anterior de
la textura de Tour para la hormiga asociada al fragmento
Si el ultimoNodo es el nodo del fragmento Entonces
        Ajustar Z para que el fragmento sea visible
                y establecer su color al de marcado
Sino
        Matar fragmento
Fin Programa
```

6.3.3. Resultados Experimentales de GPU-Fragment-OCH-OP

El algoritmo GPU-Fragment-OCH-OP también ha sido habilitado para que pueda aplicar métodos de búsqueda local, al igual de GRID-OCH-OP y GPU-Vertex-OCH-OP.

Para todos los experimentos llevados a cabo los parámetros de OCH α, β, q_0 y ϱ, han sido fijados con los siguientes valores típicos $\alpha = 1$, $\beta = 3$, $q_0 = 0.2$ y $\varrho = 0.9$. Se empleó el mismo PC equipado con la GPU nVidia GeForce 6600GT, realizando 20 repeticiones de los experimentos y extrayendo resultados promediados para construir las gráficas de rendimiento.

El algoritmo GPU-Fragment-OCH-OP está diseñado de tal forma que trata de superar el rendimiento de GPU-Vertex-OCH-OP. El diseño está orientado a la computación de fragmentos, y no a la computación de vértices, por ello se cambiaron las primitivas gráficas del proceso de resolución con el objetivo añadido de aprovechar la capacidad de trabajo del rasterizador y de la mayor capacidad de procesamiento de fragmentos por parte de la GPU y reducir el procesamiento de vértices.

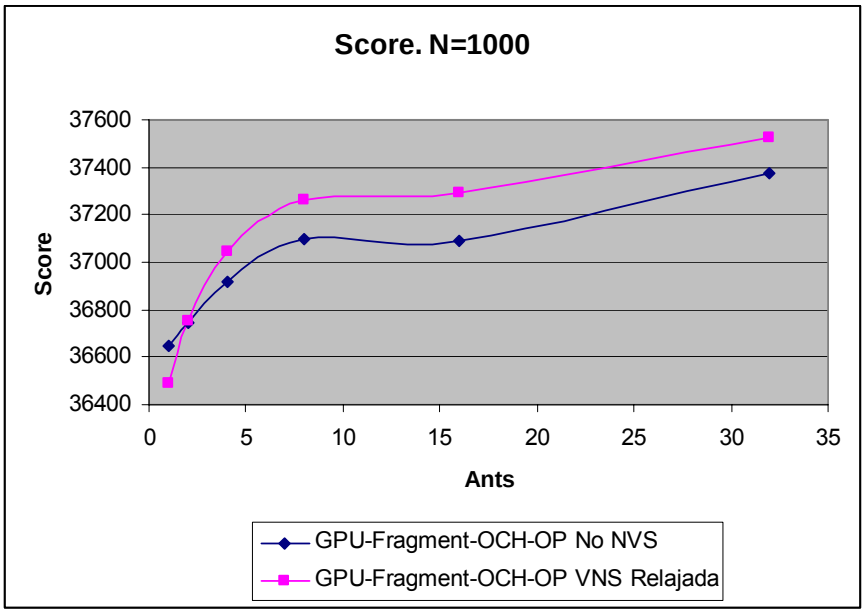


Figura 79. Variantes de GPU-Fragment-OCH-OP: Score para N=1000.

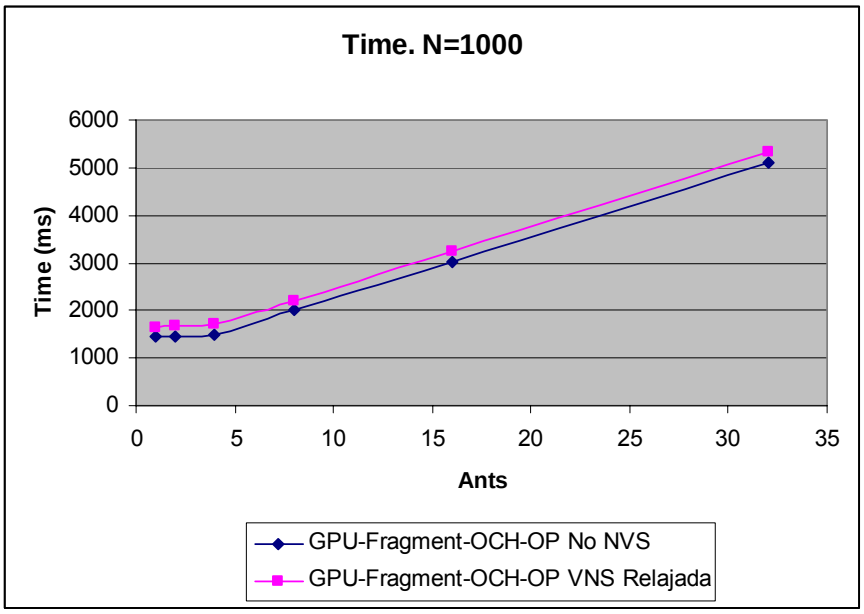


Figura 80. Variantes de GPU-Fragment-OCH-OP: Time para N=1000

En la Figura 79 y en la Figura 80 se observa, como cabía esperar, que la utilización de métodos de búsqueda local mejora el *score* de los caminos obtenidos. La mejora del *score* al utilizar búsqueda local no es muy importante

pero el coste temporal adicional tampoco lo es. Para los grafos de talla 3000 nodos se produce el mismo comportamiento.

Se observa como con 1, 2 y 4 hormigas el tiempo de resolución es prácticamente idéntico mientras que el *score* aumenta de forma bastante notable con ese número de hormigas. El hecho de que con esa combinación de hormigas el tiempo de cómputo sea muy similar y sostenido se explica por el propio diseño del algoritmo. Recordemos que la Fase 2a, la de *selección de nodos*, tiene un coste O(2·N) en el procesador de vértices, y un coste O(H·N) en el procesador de fragmentos. Si realizamos una tabla (Tabla 5) en la que variamos H se observa suponiendo que se resuelve el grafo de N=1000 nodos:

H	Vértices O(2N)	Fragmentos O(HN)
1	2000	1000
2	2000	2000
4	2000	4000
8	2000	8000
16	2000	16000
32	2000	32000

Tabla 5. Comparativa complejidad temporal

Lo que ocurre es que mientras el número de vértices a procesar sea similar al número de fragmentos entonces el tiempo depende más del coste del procesamiento de vértices que no de fragmentos. Es decir, bajo esa condición el tiempo de cómputo depende más de N que no de H. En tanto en cuanto el número de fragmentos a procesar se incrementa de forma notable con respecto al número de vértices ($2 \cdot N << H \cdot N$) el tiempo de cómputo pasa a depender en mayor grado del número de fragmentos a procesar, teniendo un comportamiento $H \cdot N$, y por tanto el tiempo se ve claramente afectado al variar H. Esto significa que el cuello de botella con pocas hormigas se localiza en el procesador de vértices y a medida que aumentan el número de hormigas el cuello de botella se desplaza al procesador de fragmentos. Con las instancias de grafos de 3000 nodos se observa el mismo fenómeno. Con 1, 2 y 4 hormigas el tiempo de cómputo es muy similar y el *score* se incrementa notablemente.

A la vista de estos primeros resultados se desprende que lo sensato es utilizar una colonia de 4 hormigas ya que el tiempo no se ve apenas incrementado y el *score* obtenido mejora bastante. Notar que la mejora del *score* no es excesiva pero siempre se produce a medida que participa un mayor número de

hormigas, y los incrementos más notables se producen con un número de hormigas pequeño.

A continuación se muestran las comparaciones de rendimiento entre los dos algoritmos de GPU. Para estas gráficas se ha tomado los resultados de los experimentos en los que no se utilizaba búsqueda local. El hecho de tomar los experimentos que realizaban búsqueda local o los que no la aplicaban no es relevante para el propósito de esta comparación pero se ha comprobado que los resultados obtenidos eran los esperados.

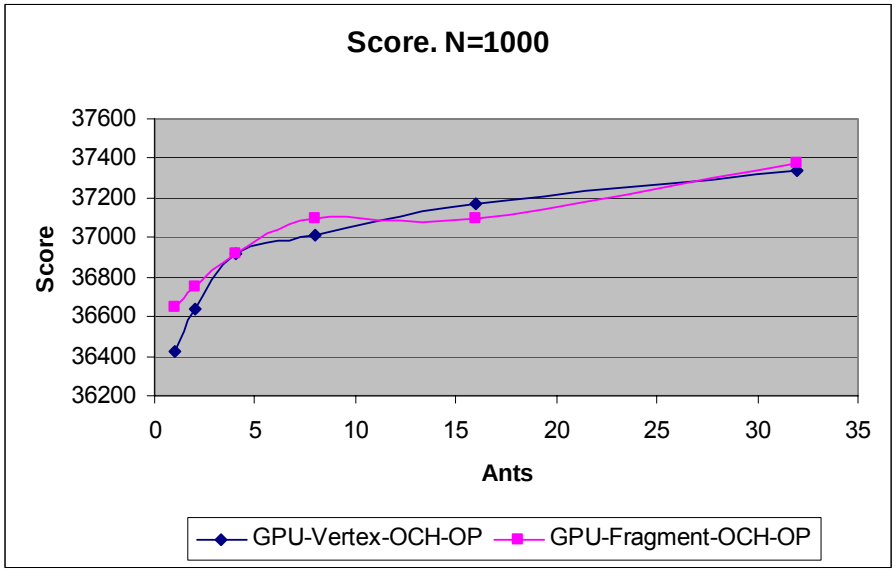


Figura 81. GPU-Vertex vs. GPU-Fragment: Score para N = 1000.

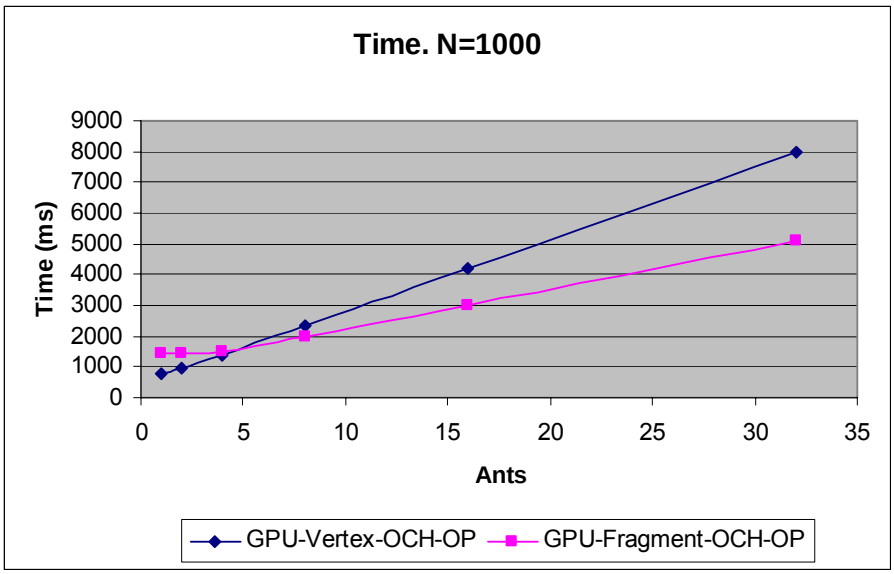


Figura 82. GPU-Vertex vs. GPU-Fragment: Time para N = 1000.

Lo que se observa en las figuras de comparación de los algoritmos de GPU tanto para grafos de 1000 nodos (Figura 81 y Figura 82) como para grafos de 3000 nodos (Figura 83 y Figura 84) es que el *score* obtenido de las soluciones con uno y otro algoritmo de GPU es bastante similar. Este hecho es razonable dado que la regla probabilista que rige el proceso en ambos algoritmos es la misma y por tanto es de esperar soluciones de muy similar calidad obtenidas por los dos algoritmos.

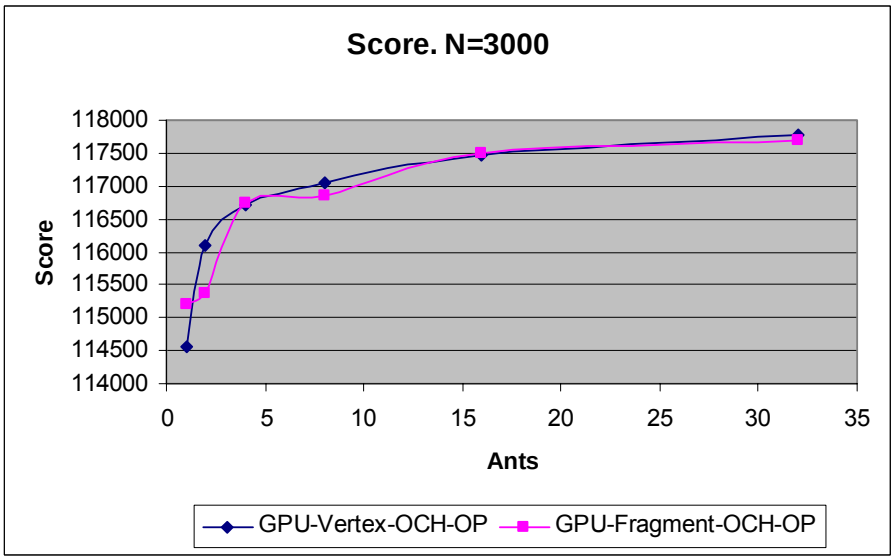


Figura 83. GPU-Vertex vs. GPU-Fragment: Score para N = 3000.

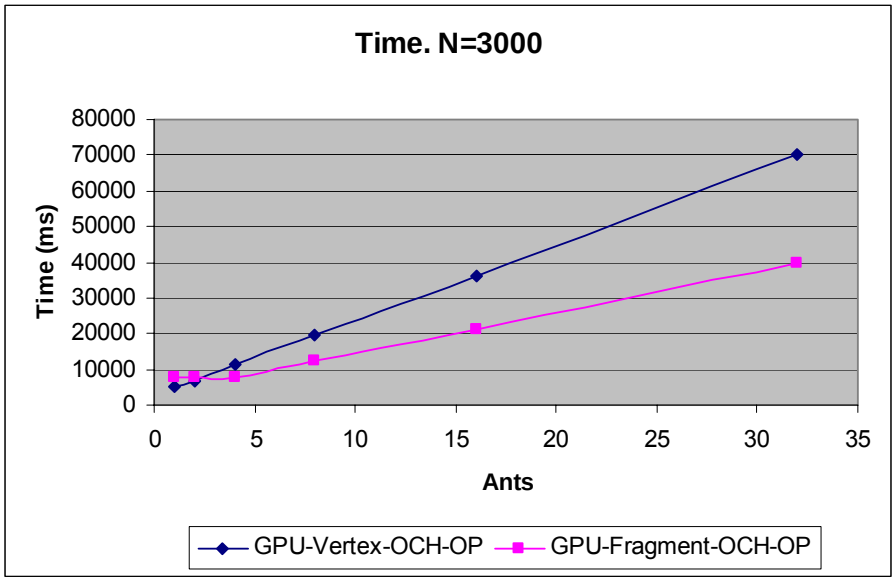


Figura 84. GPU-Vertex vs. GPU-Fragment: Time para N = 3000.

En cuanto al tiempo de ejecución de los algoritmos de GPU se observa como hay mejoras en el coste temporal por parte de GPU-Fragment-OCH-OP. De nuevo hay que recurrir a las expresiones asintóticas sobre los costes temporales teóricos de ambos algoritmos para poder explicar su comportamiento experimental. Para sintetizar los costes teóricos vamos a construir una tabla en la que se hace varíar la H y se supone N = 1000.

	GPU-Vertex-OCH-OP			GPU-Fragment-OCH-OP		
H	Vértices O(H ·N)	Fragmentos O(H ·N)	Total	Vértices O(2 ·N)	Fragmentos O(H ·N)	Total
1	1000	1000	2000	2000	1000	3000
2	2000	2000	4000	2000	2000	4000
4	4000	4000	8000	2000	4000	6000
8	8000	8000	16000	2000	8000	10000
16	16000	16000	32000	2000	16000	18000
32	32000	32000	64000	2000	32000	34000

Evidentemente los totales de la tabla no son completamente exactos ya que no podemos sumar directamente costes de procesar vértices y costes de procesar fragmentos, ya que en principio no tiene el mismo coste, pero nos servirá para explicar el comportamiento, y más tras observar la gráfica asociada (ver Figura 85) que se parece mucho a las gráficas experimentales.

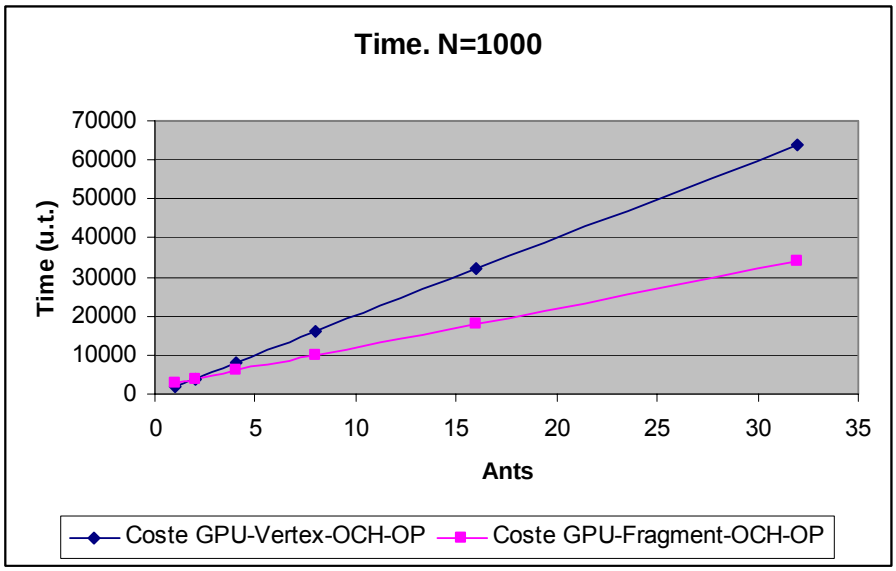


Figura 85. Comportamiento temporal teórico. N = 1000.

Para H pequeño, 1, 2 y 4 hormigas, cuesta más el procesamiento por parte de GPU-Fragment-OCH-OP que por parte de GPU-Vertex-OCH-OP. Este comportamiento se justifica al tener una mayor carga de vértices, mayor carga de trabajo por parte del rasterizador y por no realizar descarte temprano de vértices tal y como GPU-Vertex-OCH-OP puede hacer.

Sin embargo, para H mayor que cuatro, el tiempo de GPU-Fragment-OCH-OP es inferior al que requiere GPU-Vertex-OCH-OP. Además, se observa como la reducción del tiempo llega a estar en torno al 45%, lo que deja patente que GPU-Fragment-OCH-OP mejora, tal y como sugería la estrategia del algoritmo, el tiempo de ejecución frente a GPU-Vertex-OCH-OP.

A continuación se muestran gráficas que permiten ver cual es la evolución del *score* y la evolución del tiempo de ejecución de los algoritmos de GPU cuando se trabaja con un número superior de hormigas. En particular se ha ensayado con 1, 2, 4, 8, 16, 32, 64 y 128 hormigas.

Lo que se observa es que a medida que se incrementa el número de hormigas el *score* siempre mejora. Para un número de hormigas inferior a 16 los incrementos de *score* son algo más significativos que para una mayor cantidad

de hormigas. Los tiempos de ejecución también se incrementan, pero se mantienen sus tendencias de crecimiento. De hecho, podemos decir que los tiempos se comportan de forma lineal con respecto a H si mantenemos N constante.

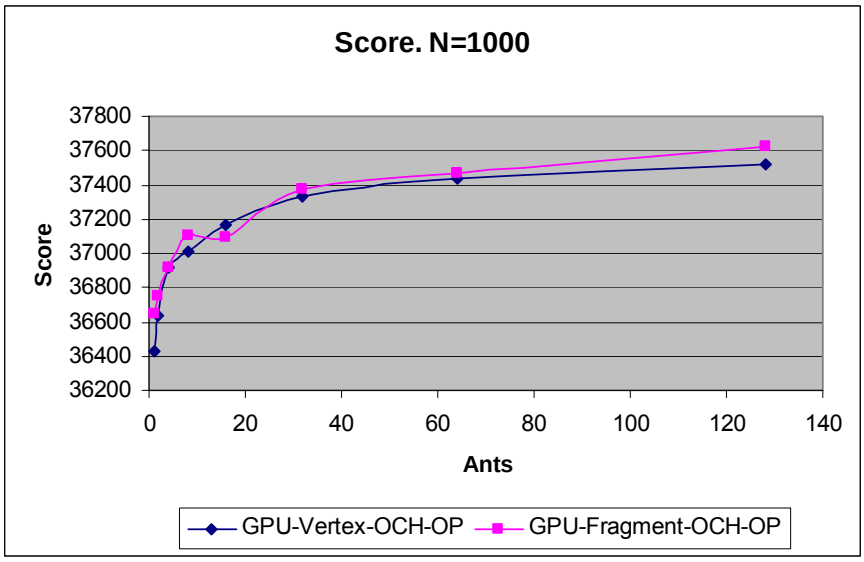


Figura 86. GPU-Vertex vs. GPU-Fragment: Time para N=1000 y H hasta 128.

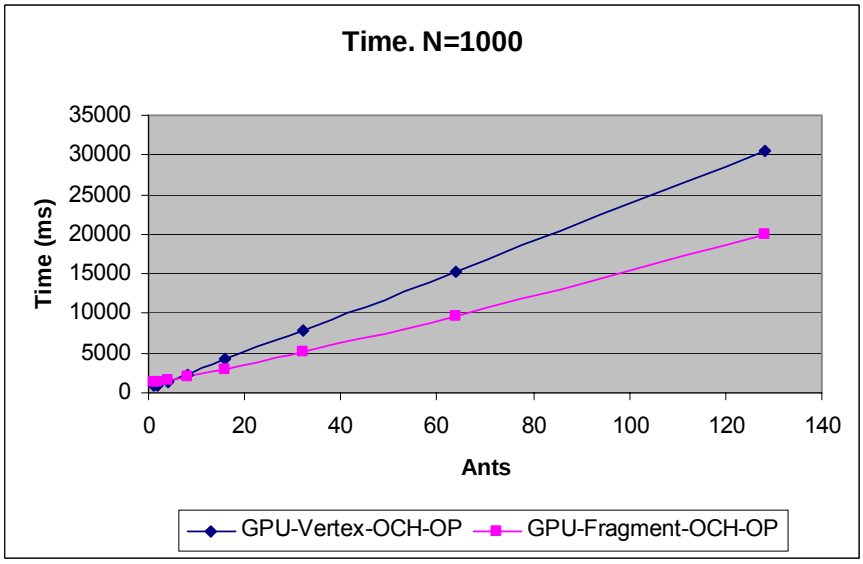


Figura 87. Time para algoritmos de GPU. N=1000 y H hasta 128

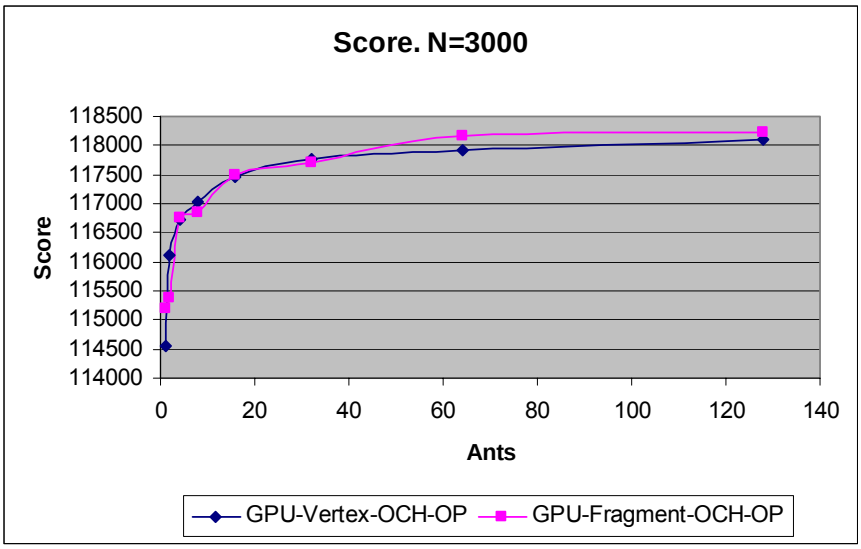


Figura 88. GPU-Vertex vs. GPU-Fragment: Score para N=3000 y H hasta 128

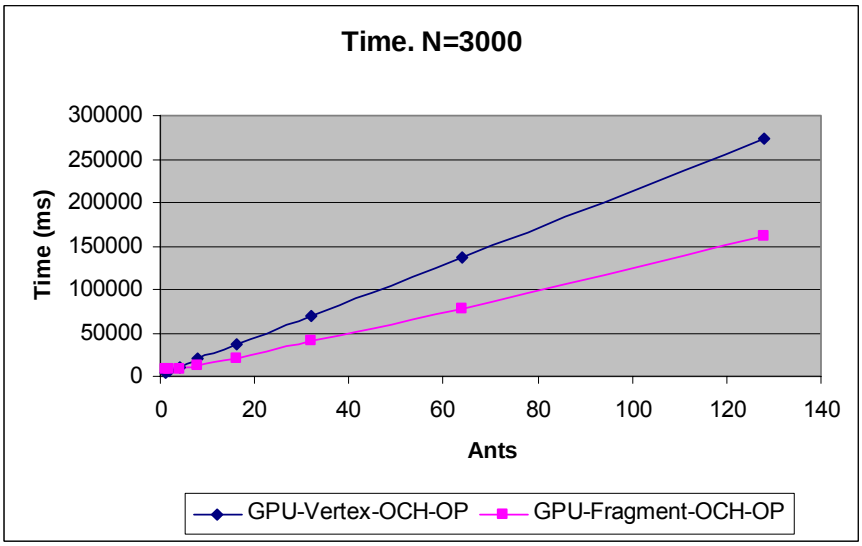


Figura 89. GPU-Vertex vs. GPU-Fragment: Time para N=3000 y H hasta 128

En la siguiente tabla se recogen las expresiones capaces de estimar el coste temporal de los algoritmos cuando se ejecutan sobre grafos de 1000 y 3000 nodos. Estas expresiones han sido obtenidas por el método de ajuste de mínimos cuadrados atendiendo a un modelo lineal.

	GPU-Vertex-OCH-OP	GPU-Fragment-OCH-OP

N	Expresión	R^2	Expresión	R^2
1000	T(h) = 233,45h + 489,63	1	T(h)=145,97h+ 851,01	0,9964
3000	T(h)=2108,9h + 2729,6	1	T(h)=1218,5h + 3057,6	0,9979

6.4. Conclusiones del capítulo

En el presente capítulo hemos repasado cuál es la programabilidad actual de los procesadores gráficos, GPUs, y cómo organizan sus cálculos en el denominado *pipeline gráfico*. Se han presentado los algoritmos diseñados, tanto algorítmicamente como mediante una descripción ilustrada de las ideas más importantes e innovadoras. Se expusieron los detalles más importantes a tener en cuenta en su implementación, como la necesidad de programar un generador de números pseudo-aleatorios para GPU, o algunos detalles relacionados con restricciones del hardware actual. La implementación realizada se ha comparado con los resultados obtenidos con el algoritmo del capítulo anterior, GRID-OCH-OP. En el caso de GPU-VERTEX-OCH-OP se ha observado como para un número reducido de hormigas, las soluciones son de mayor calidad y los tiempos de ejecución son muy inferiores a los obtenidos con el algoritmo en GRID. Por ejemplo, con una única hormiga, GPU-VERTEX-OCH-OP es 80 veces más rápido que GRID-OCH-OP, obteniendo además solución con un 9% más de *score* para las instancias de grafo de 3000 nodos. También hemos estimado, de forma aproximada, lo que sería previsible que ocurriera si dispusiéramos de un grid de 32 GPUs y se implementara un hipotético algoritmo que aunase las propiedades del de GPU y el de GRID. En ese supuesto, y considerando una instancia para el problema de 100000 nodos, se estima que el nuevo algoritmo, GRID-GPU-VERTEX-OCH-OP, sería capaz de resolver el problema en aproximadamente 9 segundos, frente a los 809 segundos que requiere actualmente GRID-OCH-OP.

El algoritmo GPU-Vertex-OCH-OP está orientado al procesamiento masivo de vértices y la gran mayoría de cálculos los llevan a cabo los programas de GPU del procesador de vértices. Para tratar de mejorar los resultados de este algoritmo, se diseñó otro algoritmo de GPU, GPU-Fragment-OCH-OP, que está orientado al procesamiento masivo de fragmentos, con miras de aprovechar con mayor eficiencia las capacidades de la etapa de rasterización del *pipeline* y la mayor potencia de cálculo del procesador de fragmentos.

Se observó en la experimentación con el algoritmo GPU-Fragment-OCH-OP como el tiempo de resolución requerido se mantenía cuando se empleaban 1, 2 y 4 hormigas en la colonia para resolver grafos de 1000 y 3000 nodos, a la vez que se observó que las mejoras más apreciables de *score* se producían con ese mismo número de hormigas. Con más de 4 hormigas el tiempo se comportaba linealmente con respecto a H si hacíamos permanecer a N constante. Este comportamiento se debía al desplazamiento del cuello de botella desde el procesador de vértices hacia el procesador fragmentos al pasar de 4 a más hormigas. Ello deja claro que con este algoritmo es muy recomendable trabajar con colonias de 4 hormigas.

Posteriormente se compararon los dos algoritmos de GPU, con grafos de 1000 y 3000 nodos, y haciendo variar el número de hormigas de la colonia a 1, 2, 4, 8, 16, 32, 64 y 128. Para estos experimentos se observó como a medida que se aumentaban el número de hormigas el *score* también lo hacía pero de una forma moderada. La evolución del *score* para ambos algoritmos era muy parecida ya que la regla probabilista en la que basan la construcción de caminos es la misma. En cuanto al tiempo de ejecución hay que hacer dos distinciones. Cuando se resuelven problemas utilizando 1, 2 y 4 hormigas se observa como GPU-Fragment-OCH-OP tarda algo más que GPU-Vertex-OCH-OP. Sin embargo, cuando se incrementa el número de hormigas por encima de cuatro, GPU-Fragment-OCH-OP emplea menos tiempo en realizar la resolución. Este comportamiento, como se expuso, venía justificado y se explicaba con el estudio de la carga total de vértices y fragmentos al hacer variar el número de hormigas.

Los tiempos de ambos algoritmos de GPU se comportan proporcionalmente a $O(H \cdot N)$, pero los tiempos de GPU-Fragment-OCH-OP llegan a ser en torno a un 45% menores, y en algunos casos con mayores reducciones, que los obtenidos por parte de GPU-Vertex-OCH-OP.

Finalmente, cabe destacar el importante carácter de innovación de los algoritmos diseñados, debido a la utilización de una GPU para la resolución del problema de la orientación. Dicho problema pertenece al ámbito de la teoría de grafos, y hasta el momento, no se conocen otros trabajos similares que hayan tratado de abordar problemas de este dominio para su resolución con GPU. Siendo la mayoría de publicaciones de computación de propósito general en GPU sobre álgebra lineal y simulación de fenómenos físicos, problemas que se prestan más al modelo computacional subyacente.

"The Scream" por Edgard Munich (1863-1944)

Capítulo 7

Conclusiones

Los últimos decenios han sido testigos de revoluciones tecnológicas que se han incorporado a todos los aspectos de nuestra vida diaria, incluido nuestro Patrimonio Cultural. Internet está evolucionando a una velocidad que pocos podían predecir. A mediados de los años 90 existían 5 millones de usuarios. En 2000, había 200 millones, y en 2004, en plena explosión de la burbuja tecnológica, Internet era ya un monstruo de unos 804 millones. Estas revoluciones tecnológicas han permitido que multitud de organizaciones museísticas proporcionen portales cuyo principal objetivo es proveer todo tipo de servicios adicionales y contenidos artísticos de carácter virtual.

Sin embargo, tras un estudio del uso de dichas tecnologías en los principales museos a nivel internacional hemos podido concluir que en ningún caso los portales Web sustituyen a la riqueza sensorial de las visitas in situ, que los portales Web se utilizan en gran medida como herramientas para la preparación de visitas futuras o como puntos de búsqueda de información contextual adicional sobre obras ya visitadas y que, en cualquier caso, existe una separación espacio-temporal entre la propia visita al museo y la consulta de los portales Web asociados a los mismos, lo cual dificulta los procesos de aprendizaje que tienen lugar de forma presencial en el propio museo.

Afortunadamente, la llegada de los dispositivos de mano con capacidades inalámbricas a los museos en torno al año 2002 ha sido una de las grandes revoluciones tecnológicas en este dominio, comparable sólo con la llegada de los cassettes compactos de audio en los años 80, que redujo significativamente el tamaño de los reproductores que llevaban los visitantes, y la transición de los sitemas de audio analógico a los digitales en 1994.

Desde 2002, año en el que se inició este proyecto investigador y en el que sólo existía un prototipo muy primitivo de Museo Híbrido en el *Tate Modern*, se han desarrollado diversas propuestas de aplicación de las tecnologías móviles para la mejora de las experiencias de visitas in situ a diferentes museos.

El análisis de las propuestas existentes en el año 2002 reflejaba una total carencia de sistemas que ofreciesen mecanismos de localización de visitantes, de análisis del comportamiento de los usuarios durante su visita al museo, de personalización de los contenidos en base a la localización y a los perfiles de los usuarios, de soporte a diferentes modelos de aprendizaje, de interacción social entre los visitantes, de navegación de información interrelacionada, de integración de información distribuida y perteneciente a diferentes organizaciones, de creación de contenidos adicionales por parte de los usuarios y de generación de valoraciones y opiniones sobre los elementos de la exposición que puedan ser utilizadas por futuros visitantes.

Los proyectos analizados en el capítulo 2, desarrollados en su mayoria en paralelo a este proyecto investigador durante el periodo de 2003 a 2005, presentan, como hemos podido observar, soluciones parciales a algunos de estos requisitos. Por ello, hemos planteado en esta tesis un modelo de Museo Híbrido que da soporte a diferentes estilos de aprendizaje y que permite la interacción social, el acceso a repositorios de carácter distribuido, la valoración de las obras de arte visitadas y el uso de estas valoraciones para proponer visitas dinámicas en tiempo real, ajustándose a la disponibilidad de tiempo y maximizando la importancia de las obras visitadas, medida ésta en términos de su popularidad. Todo ello demostrando que las propuestas realizadas escalan para un número de obras museísticas elevado (miles de obras) de forma que se garantice una gran variedad de posibles experiencias de visitas híbridas alternativas, incluso para un visitante en sucesivas visitas futuras a un mismo museo.

En el capítulo 3 hemos propuesto un modelo conceptual de Museo Híbrido que permite definir procesos educativos de aprendizaje no sólo de tipo conductivo, sino también de carácter constructivo y social. Este planteamiento es novedoso con respecto a las propuestas existentes de museos híbridos en las que se plantean modelos simples de exploración de contenidos basados en una concepción primitiva del aprendizaje. En nuestra propuesta es posible la

definición, mantenimiento y exploración personalizada tanto de las colecciones de objetos preservados en el museo como de las relaciones entre las mismas y entre los objetos y conceptos explicativos asociados a los mismos. Nuestro modelo permite procesos de aprendizaje simples basados en navegaciones horizontales, así como procesos altamente estructurados y complejos a partir de las diferentes diseminaciones que se pueden crear de un objeto expositivo, las proyecciones que de dichas diseminaciones se pueden hacer, el establecimiento de relaciones entre dichos elementos y las navegaciones en profundidad que los visitantes pueden realizar como parte de un proceso constructivo de conocimiento en el seno del museo. Adicionalmente, se ha propuesto un modelo de interacción social que permite la creación y gestión de grupos de interés que permiten tanto la comunicación in situ entre individuos presentes en el museo como la comunicación con visitantes previos. De esta forma es posible, mediante un mecanismo de interacción social, poder construir conocimiento adicional sobre los objetos expuestos en el museo de forma cooperativa.

Otro de los aspectos a resaltar en el capítulo 3 ha sido la definición de la semántica del modelo propuesto en términos de la ontología CRM propuesta por CIDOC. Gracias a esta caracterización semántica posibilitamos la interoperabilidad semántica entre nuestro modelo y los modelos existentes en el ámbito de la documentación museística de forma que repositorios con representaciones de datos diferentes a las propuestas en este capítulo puedan ser integrados mediante el uso de mediadores que hagan corresponder de forma semántica conceptos equivalentes en los distintos espacios de información. Por otro lado, al expresar el Modelo semántico de MoMo como un grafo constituido por recursos y propiedades expresados en formato RDF podremos beneficiarnos de los múltiples mecanismos de almacenamiento de información semántica que han sido desarrollados en el campo de la Web semántica.

Nuestro modelo de Museo Híbrido, como veremos en el siguiente capítulo, es altamente susceptible de ser distribuido, lo que contribuirá a la escalabilidad de los contenidos que pueden formar parte de un proceso constructivo de aprendizaje, y al establecimiento de grandes volúmenes de relaciones entre conceptos y objetos pertenecientes a distintos actores, ya sean instituciones museísticas o incluso individuos particulares en un entorno masivamente distribuido.

En el capítulo 4 abordamos la problemática asociada a la carencia de mecanismos para la integración de repositorios semánticos RDF heterogéneos y altamente distribuidos en el contexto de la computación global, proponiendo como solución un modelo de federación basado en enlaces RDF distribuidos y

la navegación transparente de los mismos. Se han presentado los resultados obtenidos tras la implementación de prototipos preliminares para validar la efectividad de las tecnologías que se van a utilizar y los modelos arquitectónicos propuestos, obteniéndose excelentes resultados empíricos. Dichos resultados muestran que el coste en tiempo de ejecución de tener una infraestructura de mediación semántica distribuida es perfectamente asumible. Adicionalmente, para probar la validez de la propuesta realizada se han aplicado las ideas expuestas en la implementación de un Museo Híbrido distribuido a partir de repositorios RDF siguiendo el modelo semántico MoMo-CRM.

La naturaleza distribuida de la infraestructura de datos diseñada ha revelado tener enormes ventajas en términos de escalabilidad y ha servido como infraestructura de partida para tener no sólo datos, sino también procesamiento masivamente distribuido que ha permitido implementar algoritmos complejos de gran utilidad en el contexto museístico. Este ha sido el caso del problema de la orientación en museos híbridos y la solución a dicho problema de optimización mediante un algoritmo de colonias de hormigas que hemos presentado en el capítulo 5. Hemos analizado las implementaciones paralelas existentes sobre algoritmos de colonias de hormigas y hemos constatado que en todos los casos dichas implementaciones no escalan a instancias del problema de gran tamaño. Hemos definido un algoritmo multi-colonia distribuido basado en un mecanismo de descomposición del problema en subproblemas y hemos realizado su implementación con servicios Web mediante una arquitectura de computación siguiendo el modelo de computación Grid. Tras el análisis de los resultados obtenidos podemos concluir que el algoritmo propuesto es capaz de resolver instancias de problemas de hasta 100.000 nodos con sólo 32 colonias. Hemos constatado que nuestra propuesta es capaz de mejorar la calidad de las soluciones obtenidas a medida que aumentamos el número de colonias involucradas en la computación y que este comportamiento se da consistentemente para distintos tamaños de grafos estudiados y diferentes grados de homogeneidad.

Finalmente, en el capítulo 6 hemos mejorado los algoritmos definidos en el capítulo anterior mediante el uso de los procesadores gráficos presentes en la mayoría de computadores actuales. Hemos repasado cuál es la programabilidad actual de dichos procesadores, GPUs, y cómo organizan sus cálculos en el denominado *pipeline gráfico*. Se han presentado los algoritmos diseñados, tanto algorítmicamente como mediante una descripción ilustrada de las ideas más importantes e innovadoras. Se expusieron los detalles más importantes a tener en cuenta en su implementación, como la necesidad de programar un generador de números pseudo-aleatorios para GPU, o algunos detalles relacionados con restricciones del hardware actual. La implementación

realizada se ha comparado con los resultados obtenidos con el algoritmo del capítulo anterior, GRID-OCH-OP. En el caso de GPU-VERTEX-OCH-OP se ha observado como para un número reducido de hormigas, las soluciones son de mayor calidad y los tiempos de ejecución son muy inferiores a los obtenidos con el algoritmo en GRID. Por ejemplo, con una única hormiga, GPU-VERTEX-OCH-OP es 80 veces más rápido que GRID-OCH-OP, obteniendo además solución con un 9% más de *score* para las instancias de grafo de 3000 nodos. También hemos estimado, de forma aproximada, lo que sería previsible que ocurriera si dispusiéramos de un grid de 32 GPUs y se implementara un hipotético algoritmo que aunase las propiedades del de GPU y el de GRID. En ese supuesto, y considerando una instancia para el problema de 100000 nodos, se estima que el nuevo algoritmo, GRID-GPU-VERTEX-OCH-OP, sería capaz de resolver el problema en aproximadamente 9 segundos, frente a los 809 segundos que requiere actualmente GRID-OCH-OP. Cabe destacar el importante carácter de innovación de los algoritmos diseñados, debido a la utilización de una GPU para la resolución del problema de la orientación. Dicho problema pertenece al ámbito de la teoría de grafos, y hasta el momento, no se conocen otros trabajos similares que hayan tratado de abordar problemas de este dominio para su resolución con GPU. Siendo la mayoría de publicaciones de computación de propósito general en GPU sobre álgebra lineal y simulación de fenómenos físicos, problemas que se prestan más al modelo computacional subyacente.

Finalmente, como conclusión final, habría que destacar el impacto que el presente trabajo ha tenido en la comunidad que investiga la aplicación de las nuevas tecnologías en el contexto de los museos. En su conferencia más relevante, *Museums and The Web*, y en la edición en la que se cumplía el X aniversario de la misma, uno de sus conferenciantes principales citó en la conferencia de inauguración al proyecto de investigación MoMo como modelo de Museo a seguir en el futuro. En sus propias palabras:

> *"To bring about this conceptual synthesis, future systems will ultimately need to customise content using sophisticated algorithms that bring together more erudite visitor, curator and educator models. In essence they will need to bring to life the MoMo model so passionately espoused by Javier Jaen and the team from the Polytechnic University of Valencia at the 2005 Museums and the Web conference in Vancouver" (Sumption, 2006)*

Bibliografía

(Amico, 2002) Véase: http://www.amico.org

(Amn, 2004) Véase: http://www.amn.org

(Arms, 2000) Arms, W. Digital Libraries. The MIT Press, Cambridge, Massachusetts, 2000.

(Baker, 2002) Baker, M., Rajkumar, B., Laforenza, D. Grids and Grid technologies for wide-area distributed computing. In software- Practice and Experience 2002. John Wiley & Sons

(Ballestero, 2002) Ballestero, F. Et al, 2002. Fundación AUNA. eEspaña 2002. Informe Anual sobre el Desarrollo de la Sociedad de la Información en España. Fundación AUNA, Madrid, SPAIN.

(Banks, 2001) Banks, J., et al. Discrete-event system simulation. Prentice-Hall. 2001

(Beckers, 1992) Beckers R., Deneubourg J.L. and S. Goss. Trails and U-turns in the selection of the shortest path by the ant Lasius niger. Journal of theoretical biology, 159, 397-415. 1992

(Beckett, 2003) D. Beckett The Redland RDF Application Framework Véase: http://www.redland.opensource.ac.uk/

(Beer, 1990) Beer, V. The Problem and Promise of Museums Goals. Curator 33 (1999:5-18)

(Bellwood, 2003) Bellwood, T., Clément, L., von Riegen, C. (eds.). UDDI Version 3.0.1. UDDI Spec Technical Committee Specification. 14 October 2003. http://uddi.org/pubs/uddi-v3.0.1-20031014.htm.

(Berners-Lee, 2001) Berners-Lee, T., Hendler, J., Lassila, O. The Semantic Web. Scientific American. 2001

(Berners-Lee, 1989) Berners-Lee, T. Information Management: A Proposal. European Laboratory for Particle Physics (CERN). http://www.w3.org/History/1989/proposal.html

(Bluetooth, 2005) Véase: http://www.bluetooth.org

(Blythe, 2004) Blythe ,D. Windows Graphics Foundation. WinHEC2004. Microsoft Corporation. 2004

(Bolondi, 2003) Bolondi, M., Bondanza, M. Parallelizzazione di un algoritmo per la risoluzione del problema del commesso viaggiatore. Master's thesis, Dipartimento di Elettronica, Politecnico di Milano. 1993.

(Boon, 2000) Boon, T. The opportunities of hybridity: making the modern world, a new historical gallery in a diverse institution. Paper presented at Science Communication, Education, and the History of Science conference, British Society for the History of Science. 2000

(Booth, 2003) D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, D. Orchard (eds.). Web Services Architecture. W3C Working Draft 8 August 2003. http://www.w3.org/TR/2003/WD-ws-arch-20030808/

(Brassard, 1996) Brassard, G., Bratley, P. Fundamentals of Algorithmics. Prentice Hall, Englewood Cliffs, NJ, 1996

(Brendon, 2002) Brendon, W., JXTA. New Riders ed. June 2002. http://www.jxta.org

(Bruijn, 2003) Bruijn, O., Stathis K. Socio-cognitive Grids: The NET as a Universal Human Resource.Proceedings of Tales of the disappearing computer. CTI Press, Santorini 2003

(Buckingham, 2002) Buckingham, S., De Roure, D., Eisenstadt, M., Shadbolt, N., Tate, A. CoAKTinG: Collaborative Advanced knowledge Technologies in the Grid. Proceedings Second Workshop on Advanced Collaborative Environments, 11th IWWW International Symposium on High Performance Distributed Computing (HPDC-11), 2002, Edinburgh

(Bullnheimer, 1998) Bullnheimer, B., R. F. Hartl, and C. Strauss, "Parallelization Strategies for the Ant System," in R. De Leone, A. Murli, P. Pardalos, and G. Toraldo (eds.), High Performance Algorithms and Software in Nonlinear Optimization; Series: Applied Optimization, vol. 24, 1998

(Bullnheimer, 1999) Bullnheimer, B., Hartl, R. F., Strauss, C. A new rank-based version of the Ant System: A computational study. Central European Journal for Operations Research and Economics, 7:1, 1999

(Bush, 1945) Vannevar, B. "As We May Think." Atlantic Monthly July 1945: 101-108. Rpt. in Nyce and Kahn. 85-110

(Carreras, 2005) Carreras Monfort, C. Patrimonio Cultural y Tecnologias de la Información y la Comunicación. Archivo Municipal de Cartagena. Tendencias (2). 2005

(Chao, 1996) Chao, I. M., B. L. Golden, and E. A. Wasil, "A Fast and Effective Heuristic for the Orienteering," European Journal of Operational Research, vol. 88, 1996

(Chen, 2003) Chen L., Shadbolt N.R., Tao F., Puleston C., Goble C., Cox S.J. Exploiting Semantics for e-Science on the Semantic Grid. Web Intelligence (WI2003) workshop on Knowledge Grid and Grid Intelligence, 2003

(CHIN, 2002) Metadata Standards for Museum Cataloguing. The Canadian Heritage Information Network. 2002

Véase:http://www.chin.gc.ca/English/Standards/metadata_intro.html

(Christensen, 2001) Christensen, E. , Curbera, F., Meredith, G., Weerawarana, S. Web Services Description Language (WSDL) 1.1. W3C Note 15 March 2001. http://www.w3.org/TR/2001/NOTE-wsdl-20010315.

(Cordon, 2000) Cordón, O., Fernández de Viana, I., Herrera, F., Moreno, L. A new ACO model integrating evolutionary computation concepts: The Best-Worst Ant System. En M. Dorigo, M. Middendorf, y T. Stützle, editores, Abstract proceedings of ANTS2000 - From Ant Colonies to Artificial Ants: A series of International Workshops on Ant Algorithms, 2000

(Cowie, 1996) Cowie, J., Dodson, B. , Elkenbrach-Huizing, R., Lenstra, A., Montgomery, P. and Zayer, J.. A World Wide Number Field Sieve Factoring Record: On to 512 Bits . Advances in Cryptology, pages 382--394, 1996. Volume 1163 of LNCS

(Crowley&Callanan, Crowley, K., Callanan, M.A. Identifying and supporting

1998) shared scientific reasoning in parent-child interactions. Journal of Museum Education 23. 1998

(DataQuest, 2005) Véase: http://www.dqindia.com/content/datamine/2005/105020501.asp

(Doerr, 2003) Doerr, M. The CIDOC CRM – An Ontological Approach to Semantic Interoperability of Metadata, AI Magazine, Volume 24, No. 3 2003.

(Dorigo, 1992) Dorigo, M., Optimization, Learning and Natural Algorithms, Ph.D. Thesis, 1992

(Dorigo, 1996) Dorigo, M., V. Maniezzo, and A. Colorni, "The Ant System: Optimization by a Colony of Cooperating Agents," IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics, vol. 26, no. 1, 1996

(Dorigo, 1999) Dorigo, M. and G. Di Caro, "The Ant Colony Optimization Meta-Heuristic," in D. Corne, M. Dorigo and F. Glover (eds.), New Ideas in Optimization, 1999

(Dorigo, 2003) Dorigo, M., Stützle, T. The ant colony optimization metaheuristic: Algorithms, applications and advances. En F. Glover and G. Kochenberger, editores, Handbook of Metaheuristics, páginas 251-285. Kluwer Academic Publishers, 2003

(Ekahau, 2005) Véase: http:// www.ekahau.com

(Fischetti, 1998) Fischetti, M., J. J. S. Gonzalez, and P. Toth, "Solving the Orienteering Problem through Branch-and-Cut," INFORMS Journal on Computing, vol. 10, no. 2, 1998

(Fishman, 1978) Fishman, G. Conceptos y métodos en la simulación digital de eventos discretos. Limusa. México 1978

(Foster, 1997) Foster, I., Geisler, I, Nickless, W., Smith, W., Tuecke, S. Software Infrastructure for the I-Way High Performance Distributed Computing Experiment, in Proc 5th IEEE Symposium on High Performance Distributed Computing pp 562-571, 1997.

(Foster, 1999) Foster, I., Kesselman, C. (eds.) The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, 1999

(Foster, 2002) Foster, I., Kesselman, C., Nick, J., Tuecke, S. The Physiology of the Grid

(Frey, 2003) Frey, J. The Combechen Project.

Véase: http://www.combechem.org/

(Gammon, 2001) Gammon, B. Assessing Learning in Museum Environments: a Practical Guide for Museums Evaluators. London: Science Museum. 2001

(Garey, 1979) Garey, M. R., Jonson, D. Computers and Intractability: A Guide to the Theory of NPCompleteness. Freeman, San Francisco, CA, 1979

(Giles, 2003) Véase: http://urbantapestries.net/weblog/

(Global, 2005) Global Reach. Global Internet Statistics. 2005. Véase:http://www.glreach.com/globstats

(Glover, 1997) Glover, F., Laguna, M. Tabu Search. Kluwer Academic Publishers, Boston, MA, 1997

(Glover, 2003) Glover, F., Kochenberger, G. Handbook of Metaheuristics. Kluwer Academic Publishers, 2003

(Goble, 2002) Goble, C., de Roure, D., The Grid: An Application of the Semantic Web. ACM SIGMOD Record Volume 31 Issue 4. December 2002

(Golden, 1987) Golden, B. L., L. Levy, and R. Vohra, "The Orienteering Problem," Naval Research 19 Logistics, vol. 34, 1987

(Goode,1895) Goode, G.B. The Principles of Museum Administration. Proceedings of the 6th Annual Meeting of the Museums Association, 1895.

(Goss, 1989) Goss, S., Aron, S., Deneubourg, J. L., Pasteels, J. M. Self-organized shortcuts in the Argentine ant. Naturwissenschaften, 76, 1989

(GPGPU, 2005) GPGPU Website: http://www.gpgpu.org. 2005

(GPS, 2005) Véase: http://www.gpsworld.com/gpsworld/

(GSM, 2005) http://www.gsmworld.com

(Gudgin, 2003) M. Gudgin, M. Hadley, N. Mendelsohn, J.J. Moreau, H. Frystyk Nielsen (eds.). SOAP Version 1.2 Part 1:

Messaging Framework. W3C Recommendation 24 June 2003. http://www.w3.org/TR/2003/REC-soap12-part1-20030624/

(Hansen, 1999) Hansen, P. Mladenovic, N. An introduction to variable neighborhood search. En S. Voss, S.Martello, I. H. Osman, and C. Roucairol, MetaHeuristics - Advances and Trends inLocal Search Paradigms for Optimization, Kluver Academic Publishers, Dordrecht, Holanda, 1999

(Harris, 2004) Harris, M. Mapping computational concepts to the GPU. SIGGRAPH 2004.

(Hayes, 1984) Hayes, M. and J. M. Norman, "Dynamic Programming in Orienteering: Route Choice and the Siting of Controls," Journal of the Operational Research Society, vol. 35, no. 9, 1984

(Hooper, 2002) Hooper-Greenhill, E, Moussouri, T. Researching learning in Museums and Galleries 1990-1999: A Bibliografic Review.

Véase: http://www.le.ac.uk/museumstudies/rcmg

(Horrocks, 2002) Horrocks, I. DAML+OIL: a Reason-able Web Ontology Language, in Proceedings of EDBT 2002

(ICOM, 2001) International Council of Museums. ICOM Statutes of 2001, Article 2. Véase: http://icom.museum/statutes.html

(IEEE802.11, 2005) Véase: http://www.ieee802.org/11/

(Kravchyna, 2002) Kravchyna, V., Hastings, S. K. Informational Value of Museum Web Sites. First Monday, Vol. 7, No. 2, 2002

Véase: http://www.firstmonday.org/issues/issue7_2/kravchyna/#k6

(Krüger, 1998) Krüger, F., Merkle, D., Middendorf, M. Studies on parallel ant system for the BSP model. Unpublished manuscript. 1998

(Laporte, 1990) Laporte, G. and S. Martello, "The Selective Traveling Salesman Problem," Discrete Applied Mathematics, vol. 26, 1990

(Lassila, 1999) O. Lassila, R. R. Swick (eds.). Resource Description Framework. (RDF) Model and Syntax Specification. W3C Recommendation 22 February 1999. http://www.w3.org/TR/1999/REC-rdf-syntax-19990222.

(Leifer, 1993) Leifer, A. C. and M. B. Rosenwein, "Strong Linear Programming Relaxations for the Orienteering Problem," European Journal of Operational Research, vol. 73, 1993

(Liang, 2003) Liang, Y. C., Smith, A. E. An Ant Colony Approach to the Orienteering Problem. Journal of the Chinese Institute of Industrial Engineers. 2003

(Lord, 1997) Lord, B., Lord, G.D. The Manual of Museum Management. London: The Stationery Office, 1999

(Louvre, 2004) Véase: http://www.louvre.fr

(LouvreEdu,2004) Véase: http://www.louvre.edu

(Luna, 2003) Luna, F. Introduction to 3D Game Programming with DirectX 9.0. Wordware Publishing, Inc. 2003.

(MacQueen, 1967) MacQueen, J. B. Some Methods for classification and Analysis of Multivariate Observations, Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability", Berkeley, University of California Press, 1967

(Magkanari, 2002) Magkanari, A. et al. Ontology Storage and Querying. Technical Report 308. Foundation or Research and Technology Hellas Institute of Computer Science. April 2002

(Marshak, 2004) Marshak, D. J. Paul Getty Museum Re-Architects Technology to Enhance Visitors' Experience Sun Consultants and Java Technology Keys to Next-Generation Architecture. 2004

Véase:http://emea.cobalt.com/products-n-solutions/articles/article_digital_entertainment.html

(Melnika, 2004) Melnika, J., Mihailovs, Vasilijs, M. Information Technology Strategy and Its Implementation in

Museums. Stockholm School of Economics in Riga Working Papers Vol. 12, No. 67, 2004

(Michel, 1998) Michel, R., Middendorf, M. An island model based ant system with lookahead for the shortest supersequence problem. In A. E. Eiben, T. Back, M. Schoenauer, and H.-P. Schwefel, editors, Proceedings of PPSN-V, Fifth International Conference on Parallel Problem Solving from Nature. Springer-Verlag, 1998.

(Microsoft, 2003) The .NET Framework Home Site http://www.microsoft.com/net/

(Middendorf, 2000) Middendorf, M., F. Reischle, and H. Schmeck, "Information Exchange in Multi Colony Ant Algorithms," Parallel and Distributed Computing: Proceedings of the 15th IPDPS 2000 Workshops, the 3rd Workshop on Biologically Inspired Solutions to Parallel Processing Problems (BioSP3), LNCS 1800, 2000

(Middendorf, 2002) M. Middendorf, F. Reischle, y H. Schmeck. Multi colony ant algorithms. Journal of Heuristics, 8: 3, 2002.

(Moma, 2004) Véase: http://www.moma.org

(MSDN, 2005) Véase: http://msdn.microsoft.com/webservices/

(National, 2003) Véase: http://www.nationalgallery.org.uk

(Natrajan, 2001) Natrajan, A., Humphrey, M., Grimshaw, A. Capacity and Capability Computing in Legion. Proceedings of the 2001 International Conference on Computational Science, May 28-30 2001.

(Nejdl, 2002) Nejdl, W. et al. EDUTELLA: A P2P networking Infrastructure Based on RDF. WWW2002, May 7-11, 2002, Honolulu, Hawaii

(Nelson, 1981) Nelson, T. Literary Machines. Swarthmore, PA: Self-published, 1981.

(Nga, 2004) Véase: http://www.nga.gov

(OAI, 2000) The Open Archives Initiative. http://www.openarchives.org/

(Orsay, 2003) Véase: http://www.musee-orsay.fr

(Paepcke, 1998) Paepcke, A., Chen-Chuan K. Chang, García-Molina, H. and Winograd, T., Interoperability for Digital Libraries Worldwide. Communications of the ACM, Vol. 41, No. 4, April 1998.

(Paetl-Schneider, 2003) Paetl-Schneider, P., Hayes, P., Horocks, I. OWL Web Ontology Langauge: Semantics and Abstract Syntax. W3C Proposed recommendation, December 2003. http://www.w3.org/TR/2003/PR-owl-semantics-20031215/

(Pasteels, 1987) Pasteels, J. M., Deneubourg, J.-L., Goss, S. Self-organization mechanisms in ant societes (I): Trail recruitment to newly discovered food sources. Experientia Supplementum, 54, 1987

(Piaget, 1970) Piaget's theory. In P. Mussen (ed) Handbook of child psychology, Vol.1. 1970. New York: Wiley, 1983

(Pompidou,2003) Véase: http://www.centrepompidou.fr

(Portrait, 2002) Véase: http://www.npg.org.uk

(Prado, 2004) Véase: http://museoprado.mcu.es

(Ramalhinho, 1998) Ramalhinho, H. and D. Serra, "Adaptive Approach Heuristics for the Generalized Assignment Problem," Economic Working Paper 288, 1998

(Ramesh, 1992) Ramesh, R., Y. S. Yoon, and M. H. Karwan, "An Optimal Algorithm for the Orienteering Tour Problem," ORSA Journal on Computing, vol. 4, no. 2, 1992

(Reina, 2004) Véase: http://museoreinasofia.mcu.es

(RFID, 2005) Véase: http://www.rfidjournal.com/

(Ríos, 1997) Ríos, D., Ríos S., Martín ,J. Simulación: Métodos y Aplicaciones. Ra-ma. 1997

(Semper, 2002) Semper, R., Spasojevic, M. The Electronic Guidebook: Using Portable Devices and a Wireless Web-based Network to Extend the Museum Experience. Museums and the Web'02. David Bearman eds. 2002

(SFMoma, 2004) Véase: http//www.sfmoma.org

(Sheth, 1990) Sheth, A., larson J. Federated database systems for

managing distributed heterogeneous, and autonomous databases. ACM Comput. Sur. 22, 3: 183-236, 1990

(Skinner, 1953) Skinner, B. F. Science and Human Behavior. Macmillan Free Press. 1953

(Stallings, 2003) Stallings W. Computer Organization & Architecture. Prentice-Hall 2003.

(Stützle, 1998) Stützle, T. Parallelization strategies for ant colony optimization. In A. E. Eiben, T. Back, M. Schoenauer, and H.-P. Schwefel, editors, Proceedings of PPSN-V, Fifth International Conference on Parallel Problem Solving from Nature, Springer-Verlag, 1998.

(Stützle, 2000) Stützle, T., Hoos, H. H. MAX-MIN Ant System. Future Generation Computer Systems, 16: 8, 2000

(Sumption, 2006) Sumption, K., In Search Of The Ubiquitous Museum: Reflections Of Ten Years Of Museums And The Web, in J. Trant and D. Bearman (eds.). Museums and the Web 2006: Proceedings, Toronto: Archives & Museum Informatics, 2006

Véase:http://www.archimuse.com/mw2006/papers/sumption/sumption.html

(Talbi, 1999) Talbi, E.G., Roux, O., Fonlupt, C., Robilliard, D.,: Parallel Ant Colonies for Combinatorial Optimization Problems. IPPS/SPDP Workshops 1999

(Tasgetiren, 2000) Tasgetiren, M. F. and A. E. Smith, "A Genetic Algorithm for the Orienteering Problem," Proceedings of the 2000 Congress on Evolutionary Computation, San Diego, CA, July 2000

(Tate, 2004) Véase: http://www.tate.org.uk

(Troelsen, 2005) Troelsen, A. Pro C# 2005 and the .NET 2.0 Platform, Third Edition. Apress, 2005

(Tsiligirides, 1984) Tsiligirides, T., "Heuristic Methods Applied to Orienteering," Journal of Operational Research Society, vol. 35, no. 9, 1984

(Thyssen, 2003) Véase: http://www.museothyssen.org/

(Ubisense, 2005) Véase: http://www.ubisense.net/

(UWB, 2005) Véase: http://www.intel.com/technology/comms/uwb/

(Veltman, 2003) Europe's Cultural Heritage in the Digital Age. Digital Resources in the Humanities Conference. 2003

(Venkatasubramanian, 2004) Venkatasubramanian, S. GPU Programming and Architecture: Intro to the course: Why study the GPU. 2004

(Venkatasubramanian, 2004b) Venkatasubramanian, S. GPU Programming and Architecture: Understanding the Fixed Function Pipeline. 2004

(Vygotsky, 1978) Mind in Society. The development of Higher Psychological Processes. Harvard University Press. 1978

(W3C, 2005) Véase: http://www.w3.org/RDF/

(Wang, 1995) Wang, Q., X. Sun, B. L. Golden, and J. Jia, "Using Artificial Neural Networks to Solve the Orienteering Problem," Annals of Operations Research, vol. 61, 1995

(Watson, 1913) Watson, J. B. Psychology as the behaviorist views it. Psychological Review, 20, 158-177.

(WebMuseum, 1994) Véase: http://www.ibiblio.org/wm

(WiMax, 2005) Véase: http://www.wimaxforum.org/home/

(WiFi, 2005) Véase: http://www.wi-fi.org/

(Wren, 1972) Wren, A. and A. Holiday, "Computer Scheduling of Vehicles from One or More Depots to a Number of Delivery Points," Operations Research Quarterly, vol. 23, 1972

(Wroe, 2003) Wroe, C., Stevens, R., Goble, C., Roberts, A., Greenwood, M. "A suite of DAML+OIL Ontologies to Describe Bioinformatics Web Services and Data". To appear in IJCAI Special issue on Bioinformatics Data and Data modelling.

ANEXO A. Especificación de requisitos IEEE

A.1. Funciones del producto

El desarrollo de este proyecto tiene dos funcionalidades claramente diferenciadas. A saber, las que tienen que ver con la actividad que puede realizar cualquier visitante del museo que utilice el sistema, y aquellas que tienen que ver con las tareas que se pueden realizar desde la aplicación de administración del sistema. A su vez, las actividades que puede realizar un visitante del museo se pueden diferenciar entre las que tienen que ver con la Navegación de Visitas y aquéllas relacionas con la Interacción Social.

En primer lugar se muestran las funcionalidades relacionadas con la Interacción Social de los visitantes:

A.1.1. Identificación

- Iniciar una sesión.
- Finalizar una sesión.

A.1.2. Gestión de grupos

- Crear un grupo.
- Eliminar un grupo.
- Recuperar el perfil de un grupo.
- Modificar el perfil de un grupo.
- Listar los grupos disponibles.
- Listar los grupos a los que se pertenece.
- Listar grupos gestionados.

A.1.3. Gestión de participación en grupo

- Unirse a un grupo.
- Abandonar un grupo.
- Expulsar un miembro del grupo.
- Silenciar a un miembro del grupo.
- Habilitar a un miembro del grupo.

A.1.4. Gestión de perfil

- Recuperar el perfil de un visitante.
- Modificar un perfil de visitante.

- Recuperar avatares.

A.1.5. Listado de visitantes

- Listar los miembros de un grupo.
- Listar los visitantes no miembros de un grupo.
- Listar los visitantes del museo.

A.1.6. Gestión de notificaciones

- Enviar una notificación.
- Listar notificaciones.
- Borrar notificaciones.
- Recuperar una notificación.
- Responder a una notificación interactiva.
- Listar grupos accesibles.
- Listar miembros accesibles de un grupo.
- Listar visitantes accesibles.
- Silenciar un grupo.
- Habilitar un grupo.
- Silenciar a un visitante.
- Habilitar a un visitante.
- Silenciar a todos los visitantes.
- Habilitar a todos los visitantes.

En segundo lugar, las funcionalidades correspondientes a la Navegación de Visitas:

- Recuperar visitas disponibles.
- Recuperar categorías soportadas.
- Recuperar tamaño de la colección.
- Recuperar miembros de la colección.
- Recuperar diseminaciones.
- Recuperar diseminación por defecto.
- Recuperar diseminaciones de un área.

- Localizar una obra.
- Marcar elementos visitados.
- Votar elementos visitados.
- Generar una visita dinámica.
- Confirmar fin de una visita.

Por último, las funcionalidades correspondientes a la administración del sistema:

A.1.7. Identificación

- Iniciar una sesión.
- Finalizar una sesión.

A.1.8. Gestión de visitas

- Crear una visita.
- Eliminar una visita.
- Modificar una visita.
- Listar visitas disponibles.

A.1.9. Gestión de colecciones

- Crear una colección.
- Eliminar una colección.
- Modificar una colección.
- Listar colecciones disponibles.

A.1.10. Gestión de diseminaciones

- Crear una diseminación.
- Eliminar una diseminación.
- Modificar una diseminación.
- Listar diseminaciones disponibles.

A.1.11. Gestión de obras expuestas

- Dar de alta una obra.
- Dar de baja una obra.
- Modificar datos de una obra.
- Listar obras disponibles.

A.1.12. Gestión de elementos explicativos

- Crear un elemento explicativo.
- Eliminar un elemento explicativo.
- Modificar un elemento explicativo.
- Listar elementos explicativos.

A.1.13. Gestión de contenido multimedia

- Crear un elemento explicativo visual.
- Eliminar un elemento explicativo visual.
- Modificar un elemento explicativo visual.
- Listar elementos explicativos visuales.
- Crear un elemento explicativo sonoro.
- Eliminar un elemento explicativo sonoro.
- Modificar un elemento explicativo sonoro.
- Listar elementos explicativos sonoros.

A.1.14. Gestión de visitantes

- Dar de alta un visitante.
- Dar de baja un visitante.
- Consultar datos de un visitante.
- Consultar datos de un grupo.
- Listar visitantes registrados.
- Listar visitantes en museo.
- Listar grupos.

A.1.15. Envío de notificaciones

- Enviar una difusión.
- Enviar una notificación.

A.2. Características de los usuarios

El sistema será utilizado por tres tipos de usuarios, que se diferencian entre sí por el rol que desempeñan en el sistema/museo. Así nos encontramos con:

- Visitantes: como su nombre indica, se trata de cualquier visitante del museo que esté utilizando el sistema para realizar la visita.

- Gestores de grupo: se trata de aquellos visitantes que han creado al menos un grupo. Además de poder realizar las mismas acciones que cualquier Visitante, puede realizar las acciones propias de la gestión de un grupo de usuarios: dar de alta a nuevos miembros, expulsar a miembros, enviar invitaciones para unirse al grupo, etc.
- Administradores: son los miembros del personal del museo que hacen uso de la aplicación de administración para controlar y gestionar el sistema.

A.3. Restricciones para los desarrolladores

Los desarrolladores del proyecto deberán tener en cuenta las siguientes restricciones:

- La capa de persistencia del sistema estará constituida por un único servidor. La capa intermedia podrá estar formada por varios servidores.
- En la medida de lo posible deberá intentarse minimizar el número de comunicaciones, para así permitir que el acceso a los contenidos multimedia sea lo más rápido posible.

A.4. Suposiciones y dependencias

En lo que respecta a las dependencias, destacar que se pretende realizar el desarrollo de este proyecto haciendo uso de la tecnología .Net Framework, y empleando como lenguaje de programación C#. Esto permite que la codificación del lado del cliente (en los *PocketPCs*), se realice casi de la misma forma que si se tratase de un PC normal.

El uso combinado de esta tecnología con el entorno de desarrollo Visual Studio .Net facilita significativamente el desarrollo, ya que guía al desarrollador en su trabajo y automatiza ciertas tareas de la codificación. Como contrapartida, se generan una serie de dependencias; a saber: el sistema operativo a utilizar deberá ser de la familia Windows y la instalación del .Net Framework (la versión adecuada a cada plataforma) será necesaria en cualquier equipo en el que se vaya a ejecutar cualquiera de las partes del sistema.

A.5. Requisitos futuros

Aunque en esta iteración del proyecto *MoMo* no se han incluido, sí que existen una serie de funcionalidades que en el futuro deberán estar soportadas por esta plataforma, a destacar:

- Los visitantes del museo han de poder preparar su visita. Esto quiere decir que han de poder preparar la visita que seguirán cuando ya estén en el museo, pudiendo añadir al sistema sus propios contenidos multimedia.
- La creación de un portal en el que los visitantes del museo, o futuros visitantes, formen comunidades de intereses comunes en las que puedan

compartir sus experiencias y en las que compartir también visitas creadas y contenidos multimedia para usar en las visitas.

- Ya que una de las componentes inteligentes del sistema necesita conocer qué salas están accesibles, se deberán incluir en la plataforma los mecanismos necesarios para que dicha componente pueda recuperar tal información.
- Las personas que realicen su visita en grupo podrán sincronizar sus clientes para hacer el recorrido contemplando las mismas obras y los mismos contenidos multimedia.

A.6. Requisitos específicos

A.7. Requisitos funcionales

A continuación se especifican detalladamente todas las funcionalidades del sistema citadas anteriormente, empezando por las relacionadas con la Interacción Social.

A.7.1. Identificación

Iniciar una sesión

Descripción:

Este es el proceso por el que un visitante se identifica para poder utilizar el sistema.

Entrada:

Nombre de usuario y contraseña.

Salida:

Devolución del perfil del visitante que ha iniciado sesión.

Funcionamiento:

El visitante introducirá su nombre de usuario y su contraseña, y, si la información suministrada por el visitante es correcta, se creará para ese usuario una nueva sesión en la base de datos.

En caso de existir una sesión previa no finalizada, se procederá a finalizarla antes de que el visitante inicie su visita.

Finalizar una sesión.

Descripción:

Este es el proceso por el que un visitante finaliza la sesión iniciada en el sistema.

Entrada:

Identificador de la sesión asociada al visitante.

Funcionamiento:

La sesión asociada a ese identificador es finalizada.

Los grupos gestionados por ese visitante, y las notificaciones asociadas a esos grupos, también son eliminados.

A.7.2. Gestión de grupos

Crear un grupo.

Descripción:

Este es el proceso por el que un visitante puede crear un grupo de visita en el sistema.

Entrada:

Identificador de la sesión asociada al visitante.

Información asociada al grupo: nombre del grupo, tipo de grupo (público o privado) y descripción.

Salida:

Información sobre el éxito o el fallo en la creación del grupo.

Funcionamiento:

Se comprueba la validez del identificador de sesión.

Se comprueba que el nombre de grupo elegido sea único.

Se genera un identificador único para el grupo.

El nuevo grupo de visita es creado en la base de datos; el visitante asociado al identificador de sesión será su gestor y su primer miembro.

Eliminar un grupo.

Descripción:

Con este proceso un visitante puede destruir un grupo que hubiese creado previamente.

Entrada:

Identificador de la sesión asociada al visitante.

Identificador del grupo a eliminar.

Funcionamiento:

Se comprueba tanto la validez del identificador de sesión como la del identificador del grupo.

El grupo y sus notificaciones asociadas son eliminados de la base de datos.

Restricciones de usuario:

Sólo el visitante que sea el gestor del grupo afectado podrá realizar esta operación.

Recuperar el perfil de un grupo.

Descripción:

Esta funcionalidad permite que un visitante vea la información asociada a un grupo.

Entrada:

Identificador de la sesión asociada al visitante.

Identificador del grupo requerido.

Salida:

Devolución del perfil del grupo que incluirá la siguiente información: identificador del grupo, su nombre, su descripción, su tipo, su número de miembros y el identificador del gestor del grupo.

Funcionamiento:

Se comprueba tanto la validez del identificador de sesión como la del identificador del grupo.

El perfil del grupo solicitado es devuelto.

Modificar el perfil de un grupo.

Descripción:

Con esta funcionalidad, un visitante puede modificar algunos datos del perfil de un grupo.

Entrada:

Identificador de la sesión asociada al visitante.

Perfil modificado del grupo.

Funcionamiento:

Se comprueba la validez del identificador de sesión del visitante.

Los datos del perfil que son actualizables (nombre, descripción y tipo) son sustituidos por los nuevos.

El nuevo nombre del grupo ha de ser único, de lo contrario no se actualizará el perfil.

Restricciones de usuario:

Sólo el visitante que sea el gestor del grupo afectado podrá realizar esta operación.

Listar los grupos disponibles.

Descripción:

Esta operación permite a un visitante conocer los grupos a los que se puede unir.

Entrada:

Identificador de la sesión asociada al visitante.

Salida:

Listado de los grupos disponibles con la siguiente información: identificador de grupo, su nombre, el tipo de grupo y el identificador de la sesión del gestor del grupo.

Funcionamiento:

Se comprueba la validez del identificador de sesión.

Algunos datos de los grupos a los que no pertenece el visitante asociado a esa sesión son recuperados y devueltos.

Listar los grupos a los que se pertenece.

Descripción:

Devuelve todos los grupos de los que es miembro un visitante determinado.

Entrada:

Identificador de la sesión asociada al visitante.

Salida:

Listado de los grupos a los cuales pertenece con la siguiente información: identificador de grupo, su nombre, el tipo de grupo y el identificador de la sesión del gestor del grupo.

Funcionamiento:

Se comprueba la validez del identificador de sesión.

Algunos datos de los grupos a los que pertenece el visitante asociado a esa sesión son recuperados y devueltos.

Listar grupos gestionados.

Descripción:

Con esta operación un visitante puede listar todos los grupos que gestiona.

Entrada:

Identificador de la sesión asociada al visitante.

Salida:

Listado con algunos datos de cada grupo que gestiona el visitante: identificador de grupo, su nombre, el tipo de grupo y el identificador de la sesión del gestor del grupo.

Funcionamiento:

Se comprueba la validez del identificador de sesión.

Parte de los datos disponibles sobre los grupos que gestiona el visitante son recuperados y devueltos.

A.7.3. Gestión de participación en grupo

Unirse a un grupo.

Descripción:

Con esta funcionalidad, un visitante puede unirse a un grupo de visita.

Entrada:

Identificador de la sesión asociada al visitante.

Identificador del grupo a unirse.

Funcionamiento:

Se comprueba la validez del identificador de sesión.

Se crea una nueva entrada en los miembros del grupo para ese visitante.

Restricciones de uso:

El visitante sólo podrá unirse directamente a los grupos declarados como públicos por sus gestores.

Abandonar un grupo.

Descripción:

Esta operación permite a un visitante dejar de ser miembro de un grupo.

Entrada:

Identificador de la sesión asociada al visitante. Identificador del grupo.

Funcionamiento:

Se comprueba la validez del identificador de sesión.

Se comprueba que el visitante asociado a esa sesión pertenece al grupo.

La entrada que asociaba al visitante con el grupo es eliminada.

Restricciones de uso:

Los gestores de grupos no pueden abandonar los grupos que gestionan[2].

Expulsar un miembro del grupo.

Descripción:

El gestor de un grupo puede hacer que un visitante abandone su grupo con este proceso.

---

[2] Para dejar de ser gestor de un grupo han de eliminar el grupo.

Entrada:

Identificador de la sesión asociada al gestor. Identificador de la sesión del miembro a expulsar. Identificador del grupo del que será expulsado.

Funcionamiento:

Se comprueba la validez del identificador de sesión.

Se comprueba que el visitante asociado a esa sesión es el gestor del grupo del cual se quiere echar al otro visitante.

La entrada que le asociaba con el grupo es eliminada.

Silenciar a un miembro del grupo.

Descripción:

El gestor de un grupo decide impedir que un miembro de un grupo que él gestiona pueda enviar notificaciones a todo el grupo (difusiones).

Entrada:

Identificador de la sesión asociada al gestor.

Identificador de la sesión del miembro a silenciar.

Identificador del grupo.

Funcionamiento:

Se comprueba la validez del identificador de sesión.

Se comprueba que el visitante asociado a esa sesión es el gestor del grupo al que pertenece el miembro a silenciar.

La entrada que asociaba al miembro con el grupo es marcada como silenciada.

Habilitar a un miembro del grupo.

Descripción:

El gestor de un grupo decide permitir que un miembro de un grupo que él gestiona pueda enviar notificaciones a todo el grupo (difusiones).

Entrada:

Identificador de la sesión asociada al gestor.

Identificador de la sesión del miembro a habilitar.

Identificador del grupo.

Funcionamiento:

Se comprueba la validez del identificador de sesión.

Se comprueba que el visitante asociado a esa sesión es el gestor del grupo al que pertenece el miembro a habilitar.

La marca de silencio es borrada de la entrada que asociaba al miembro con el grupo.

A.7.4. Gestión de perfil

Recuperar el perfil de un visitante.

Descripción:

Esta funcionalidad recupera el perfil público de un determinado visitante.

Entrada:

Identificador de la sesión asociada al visitante.

Identificador de la sesión asociada al visitante de quién se quiere recuperar el perfil.

Salida:

El perfil público del visitante que contendrá los siguientes datos: identificador de sesión, *nickname* (alias), su avatar, su género, su correo, su descripción y si el solicitante le ha silenciado.

Funcionamiento:

Se comprueba la validez del identificador de sesión de ambos visitantes.

Los datos del perfil público del visitante asociado a la sesión especificada son recuperados y devueltos.

Modificar un perfil de visitante.

Descripción:

Esta operación permite que un visitante pueda modificar los datos de su perfil.

Entrada:

Identificador de la sesión asociada al visitante.

Perfil modificado del visitante con estos datos: los datos del perfil público más el nombre y los apellidos.

Salida:

Información sobre el éxito o el fallo de la modificación.

Funcionamiento:

Se comprueba la validez del identificador de sesión.

Los datos del perfil del visitante que son actualizables (nombre, apellidos, alias, sexo, correo, avatar y descripción) son almacenados.

La modificación no se llevará a cabo si el nuevo alias no es único.

Recuperar avatares.

Descripción:

Devuelve las URL de todos los avatares del sistema.

Entrada:

Identificador de la sesión asociada al visitante.

Salida:

Listado con las URL de todos los avatares.

Funcionamiento:

Se comprueba la validez del identificador de sesión.

Las URL de todos los avatares existentes en el sistema, así como sus identificadores, son recuperadas y devueltas.

A.7.5. Listado de visitantes

Listar los miembros de un grupo.

Descripción:

Recupera todos los miembros de un grupo determinado.

Entrada:

Identificador de la sesión asociada al visitante.

Identificador del grupo deseado.

Salida:

Listado con los datos de cada miembro del grupo indicado: identificador de sesión, identificador de grupo, alias del miembro, indicadores de si puede enviar mensajes al grupo y si quiere recibir mensajes del grupo.

Funcionamiento:

Se comprueba la validez del identificador de sesión.

Los datos de todos los miembros del grupo (excepto los del solicitante) se recuperan y devuelven.

Listar los visitantes no miembros de un grupo.

Descripción:

Recupera todos los visitantes que no son miembros de un grupo determinado.

Entrada:

Identificador de la sesión asociada al visitante.

Identificador del grupo deseado.

Salida:

Listado con algunos datos de cada visitante que no forma parte del grupo indicado.

Funcionamiento:

Se comprueba la validez del identificador de sesión.

Parte de los datos disponibles sobre cada visitante son recuperados y devueltos.

Listar los visitantes del museo.

Descripción:

Con esta operación un visitante puede listar todos los visitantes del museo.

Entrada:

Identificador de la sesión asociada al visitante.

Salida:

Listado con algunos datos de cada visitante actual del museo: identificador de sesión, alias, avatar y si el solicitante le ha silenciado.

Funcionamiento:

Se comprueba la validez del identificador de sesión.

Parte de los datos disponibles sobre cada visitante son recuperados y devueltos.

A.7.6. Gestión de notificaciones

Enviar una notificación.

Descripción:

Esta funcionalidad habilita al visitante para que envíe una notificación a una serie de visitantes.

Entrada:

Identificador de la sesión del visitante.

Los datos de la notificación a enviar: asunto, texto, tipo de notificación y, si es de tipo interactiva, el identificador del grupo y la lista de identificadores de sesión de los visitantes destinatarios.

Funcionamiento:

Se comprueba la validez del identificador de sesión.

Se genera un identificador de notificación único y se crea una nueva notificación en el sistema con el visitante como remitente.

Se añade una entrada asociando la notificación a cada destinatario de la misma.

Listar notificaciones.

Descripción:

Con esta operación un visitante puede recuperar todas las notificaciones que ha recibido.

Entrada:

Identificador de la sesión asociada al visitante.

Salida:

Listado con algunos datos de cada notificación recibida por el visitante: identificador de la notificación, asunto, indicador de si ya ha sido leída con anterioridad, identificador y alias del remitente.

Funcionamiento:

Se comprueba la validez del identificador de sesión.

Parte de los datos disponibles sobre cada notificación son recuperados y devueltos.

Borrar notificaciones.

Descripción:

Esta funcionalidad permite al visitante eliminar aquellas notificaciones que no desea mantener.

Entrada:

Identificador de la sesión del visitante.

Identificador de la notificación a eliminar.

Funcionamiento:

Se comprueba la validez del identificador de sesión.

La entrada que asociaba la notificación con el visitante es eliminada.

Si la notificación se queda sin destinatarios, también es eliminada del sistema.

Recuperar una notificación.

Descripción:

Con esta operación un visitante puede recuperar una notificación en concreto.

Entrada:

Identificador de la sesión asociada al visitante. Identificador de la notificación a recuperar.

Salida:

Notificación seleccionada con los siguientes datos: identificador de la notificación, asunto, texto, identificador del remitente, su alias, tipo de notificación y, si es de tipo interactiva, el identificador de grupo.

Funcionamiento:

Se comprueba la validez del identificador de sesión.

Los datos de la notificación seleccionada son recuperados y devueltos.

Si se trata de una notificación interactiva ya respondida, se devuelve convertida en una notificación genérica.

Responder a una notificación interactiva.

Descripción:

Este proceso permite que un visitante responda a una notificación interactiva.

Entrada:

Identificador de la sesión del visitante.

Identificador de la notificación.

Tipo de notificación interactiva.

Identificador del grupo relacionado con la interacción.

Decisión tomada.

Funcionamiento:

Se comprueba la validez del identificador de sesión.

La notificación se marca como respondida.

Dependiendo de la decisión tomada, se realiza la acción indicada por la notificación interactiva.

Listar grupos accesibles.

Descripción:

Devuelve todos los grupos a los que puede enviar notificaciones un visitante.

Entrada:

Identificador de la sesión asociada al visitante.

Salida:

Listado con algunos datos de cada grupo al que el visitante puede enviar una notificación: identificador del grupo, su nombre, el número de miembros, el tipo de grupo, identificador del gestor y si ha silenciado al grupo.

Funcionamiento:

Se comprueba la validez del identificador de sesión.

Parte de los datos disponibles sobre cada grupo, del que es miembro el visitante y para el que no ha sido silenciado, son recuperados y devueltos.

Listar miembros accesibles de un grupo.

Descripción:

Devuelve todos los miembros de un grupo, del cual el visitante también es miembro, y a los que puede enviar notificaciones.

Entrada:

Identificador de la sesión asociada al visitante.

Identificador del grupo deseado.

Salida:

Listado con los datos de pertenencia de cada miembro del grupo al que puede enviar una notificación: identificador de sesión, su alias, identificador del grupo.

Funcionamiento:

Se comprueba la validez del identificador de sesión.

Se recuperan y devuelven los datos de aquellos miembros del grupo indicado que no hayan silenciado al visitante.

Listar visitantes accesibles.

Descripción:

Devuelve todos los visitantes a los que puede enviar notificaciones un visitante.

Entrada:

Identificador de la sesión asociada al visitante.

Salida:

Listado con algunos datos de cada visitante al que puede enviar una notificación: identificador de sesión, alias, avatar y si ha sido silenciado por el solicitante.

Funcionamiento:

Se comprueba la validez del identificador de sesión.

Parte de los datos disponibles de cada visitante que no le hubiese silenciado son recuperados y devueltos.

Silenciar un grupo.

Descripción:

Esta funcionalidad permite que un miembro de un grupo pueda indicar que no quiere recibir notificaciones dirigidas a todo el grupo.

Entrada:

Identificador de la sesión asociada al visitante.

Identificador del grupo a silenciar.

Funcionamiento:

Se comprueba la validez del identificador de sesión.

La entrada que asocia al grupo con el miembro es marcada como silenciada.

Habilitar un grupo.

Descripción:

Esta funcionalidad permite que un miembro de un grupo pueda indicar que quiere volver a recibir notificaciones dirigidas a todo el grupo.

Entrada:

Identificador de la sesión asociada al visitante.

Identificador del grupo a habilitar.

Funcionamiento:

Se comprueba la validez del identificador de sesión.

La entrada que asocia al grupo con el miembro se señala como no silenciada.

Silenciar a un visitante.

Descripción:

Esta funcionalidad permite a un visitante indicar que no quiere recibir notificaciones enviadas por cierto visitante.

Entrada:

Identificador de la sesión asociada al visitante.

Identificador de la sesión del visitante a silenciar.

Funcionamiento:

Se comprueba la validez de ambos identificadores de sesión.

Se almacena en la base de datos una entrada que indica que el visitante indicado ha sido silenciado por el visitante solicitante.

Habilitar a un visitante.

Descripción:

Esta funcionalidad permite a un visitante indicar que quiere volver a recibir las notificaciones que pueda enviar cierto visitante.

Entrada:

Identificador de la sesión asociada al visitante.

Identificador de la sesión del visitante a habilitar.

Funcionamiento:

Se comprueba la validez de ambos identificadores de sesión.

Se elimina de la base de datos la entrada que indica que el visitante indicado ha sido silenciado por el visitante solicitante.

Silenciar a todos los visitantes.

Descripción:

Esta funcionalidad permite a un visitante indicar que no quiere recibir notificaciones.

Entrada:

Identificador de la sesión asociada al visitante.

Funcionamiento:

Se comprueba la validez del identificador de sesión.

Se almacena en la base de datos una entrada que indica que el visitante no quiere recibir más notificaciones.

Habilitar a todos los visitantes.

Descripción:

Esta funcionalidad permite a un visitante indicar que quiere volver a recibir notificaciones.

Entrada:

Identificador de la sesión asociada al visitante.

Funcionamiento:

Se comprueba la validez del identificador de sesión.

Se elimina de la base de datos las entradas que indiquen que el visitante no quiere recibir notificaciones.

A.7.7. Navegación de Visitas

Recuperar visitas disponibles.

Descripción:

Devuelve todas las visitas disponibles.

Entrada:

Identificador de la sesión asociada al visitante.

Salida:

Listado con los datos de todas las visitas: identificador de la visita, el nombre, la URL donde está su icono y el número de diseminaciones que forman la visita.

Funcionamiento:

Se comprueba la validez del identificador de sesión.

Se recuperan y devuelven los datos de todas las visitas que existan en el sistema.

Recuperar categorías soportadas.

Descripción:

Devuelve las categorías de las colecciones que forman parte de una visita.

Entrada:

Identificador de la sesión asociada al visitante.

Identificador de la visita deseada.

Salida:

Listado con los datos de todas las categorías soportadas en esa visita: identificador, nombre y URL del icono.

Funcionamiento:

Se comprueba la validez del identificador de sesión.

Se recuperan y devuelven los datos de todas las categorías soportadas.

Se crea una entrada en la base de datos que indica que el usuario ha iniciado esa visita.

Recuperar tamaño de la colección.

Descripción:

Recupera el tamaño de las colecciones asociadas a una cierta diseminación.

Entrada:

Identificador de la sesión asociada al visitante.

Identificador de la diseminación deseada.

Identificador de la categoría de las colecciones.

Salida:

Tamaño total de las colecciones asociadas a la diseminación indicada.

Funcionamiento:

Se comprueba la validez del identificador de sesión.

Se calcula y se devuelve el número de elementos que componen las colecciones de la categoría especificada y que están asociadas a la diseminación indicada.

Recuperar miembros de la colección.

Descripción:

Recupera una serie de miembros, dentro de un rango, de una colección de diseminaciones.

Entrada:

Identificador de la sesión asociada al visitante.

Identificador de la diseminación.

Identificador de la categoría.

Límite inferior del rango.

Límite superior del rango.

Salida:

Listado con los miembros (elementos explicativos o las diseminaciones) de las colecciones asociadas a la diseminación indicada. Para una diseminación se devuelve: su identificador, su nombre, la URL de su icono, la lista con los identificadores de las categorías de las colecciones asociadas a ella (en el caso de tenerlas) y su elemento explicativo genérico. Para un elemento explicativo: su identificador, su nombre, la URL de su icono, las URL de sus archivos explicativos visuales y sonoros, así como sus formatos.

Funcionamiento:

Se comprueba la validez del identificador de sesión.

Se recupera el identificador de la visita que está realizando el visitante.

Se invoca al predictor, se calcula el número de elementos a devolver para la categoría indicada y se recuperan.

Recuperar diseminaciones.

Descripción:

Recupera una serie de diseminaciones, dentro de un rango, asociadas a una visita determinada.

Entrada:

Identificador de la sesión asociada al visitante.

Identificador de la visita.

Límite inferior del rango.

Límite superior del rango.

Salida:

Listado con las diseminaciones asociadas a la visita indicada y que están dentro del rango; se devolverán: su identificador, su nombre, la URL de su icono, la lista con los identificadores de las categorías de las colecciones asociadas a ella (en el caso de tenerlas) y su elemento explicativo genérico.

Funcionamiento:

Se comprueba la validez del identificador de sesión.

Se recuperan las diseminaciones asociadas a la visita indicada que estén dentro del rango especificado.

Recuperar diseminación por defecto.

Descripción:

Con esta operación se devuelve la diseminación que está asociada a una obra de la exposición.

Entrada:

Identificador de la sesión asociada al visitante.

Identificador de la obra expuesta.

Salida:

La diseminación asociada a la obra indicada con los siguientes datos: su identificador, su nombre, la URL de su icono, la lista con los identificadores de las categorías de las colecciones asociadas a ella (en el caso de tenerlas) y su elemento explicativo genérico.

Funcionamiento:

Se comprueba la validez del identificador de sesión.

Se recupera la diseminación asociada a la obra que tiene ese identificador de exposición.

Generar una visita dinámica.

Descripción:

Genera una visita dinámica para cierto visitante y a partir de los parámetros que éste indique.

Entrada:

Identificador de la sesión asociada al visitante.

Identificador del área inicial.

Identificador del área final.

Tiempo disponible.

Salida:

Identificador de la visita dinámica generada.

Funcionamiento:

Se comprueba la validez del identificador de sesión.

Se crea una nueva visita de tipo dinámica en la base de datos.

Se guardan las áreas de inicio y final para esa visita.

Se invoca al sistema de búsqueda para que calcule el recorrido a realizar durante la visita.

Se almacena en la base de datos que el visitante ha entrado en la visita.

Cuando el sistema de búsqueda devuelve el camino hallado, se almacena en la base de datos calculando los tiempos estimados de entrada en cada área del recorrido.

Recuperar diseminaciones de un área.

Descripción:

Devuelve el conjunto de diseminaciones por defecto de las obras expuestas en un área del museo.

Entrada:

Identificador de la sesión asociada al visitante.

Identificador de la visita. Índice del área a la que se va a entrar.

Salida:

Listado con las diseminaciones asociadas por defecto a las obras expuestas en el área indicada; para cada diseminación se recuperará su identificador, su nombre, la URL de su icono, la lista con los identificadores de las categorías de las colecciones asociadas a ella (en el caso de tenerlas) y su elemento explicativo genérico.

Funcionamiento:

Se comprueba la validez del identificador de sesión.

Se recupera los datos de tiempos de la visita dinámica, el de duración estimada del área indicada y el identificador del área final de la visita.

Con estos datos se calcula la posible desviación respecto a las previsiones de tiempo.

Se obtienen las diseminaciones más populares del área indicada que puedan verse en el tiempo disponible.

Se invoca al sistema de búsqueda para que recalcule la visita dinámica.

Se marca el área como visitada.

Restricciones de uso:

Tan sólo se puede invocar esta operación cuando el visitante esté realizando una visita dinámica.

Localizar una obra.

Descripción:

Esta funcionalidad devuelve al visitante la posición de una obra en el museo.

Entrada:

Identificador de la sesión asociada al visitante.

Identificador de la diseminación asociada por defecto a la obra a localizar.

Salida:

Posición de la obra sobre el plano de la planta del museo en la que se exhiba.

Funcionamiento:

Se comprueba la validez del identificador de sesión.

Con el identificador de la diseminación se recupera la localización de la obra expuesta que tiene esa diseminación como su diseminación por defecto.

Marcar elementos visitados.

Descripción:

Esta funcionalidad permite que el sistema conozca que diseminaciones ve el visitante y a que categoría pertenece.

Entrada:

Identificador de la sesión asociada al visitante.

Identificador de la categoría en la que se ve la diseminación.

Identificador de la diseminación contemplada.

Funcionamiento:

Se comprueba la validez del identificador de sesión.

Se almacena la categoría visitada en la base de datos.

Se aumenta la popularidad de la obra que tiene como diseminación por defecto a la especificada.

Votar elementos visitados.

Descripción:

Con esta operación los visitantes pueden votar por la obra asociada al elemento explicativo que el visitante acaba de ver.

Entrada:

Identificador de la sesión asociada al visitante.

Identificador del elemento explicativo.

Voto del visitante.

Funcionamiento:

Se comprueba la validez del identificador de sesión.

Se incrementa la popularidad de la obra asociada al elemento explicativo según el valor del voto.

Confirmar fin de una visita.

Descripción:

Esta funcionalidad devuelve al visitante la posición de una obra en el museo.

Entrada:

Identificador de la sesión asociada al visitante.

Identificador de la visita que está siguiendo.

Índice del área de la visita en la que se encuentra.

Salida:

Indicador de si se ha alcanzado el final de la visita.

Funcionamiento:

Se comprueba la validez del identificador de sesión.

Se comprueba si el área que ocupa el índice especificado es el área final de la visita.

Restricciones de uso:

Tan sólo se puede invocar esta operación cuando el visitante esté realizando una visita dinámica.

A.8. Administración del sistema

Por último, las funcionalidades correspondientes a la administración del sistema:

A.8.1. Identificación

Iniciar una sesión.

Descripción:

Mediante esta operación el personal correspondiente del museo se identifica para poder utilizar la aplicación.

Entrada:

Información asociada a la identificación del usuario: nombre de usuario y contraseña.

Salida:

En su caso, información del error en pantalla.

Funcionamiento:

Si los datos introducidos son correctos, se activan las funcionalidades de la aplicación.

Finalizar una sesión.

Descripción:

Con este proceso un usuario termina con la utilización de la aplicación.

Funcionamiento:

El usuario elige la opción del menú para terminar la sesión.

Se deshabilitan todas las funcionalidades de la aplicación.

Se muestra en pantalla el diálogo para el inicio de sesión.

A.8.2. Gestión de visitas

Crear una visita.

Descripción:

Esta funcionalidad permite crear una visita que más tarde puedan seguir los visitantes.

Entrada:

Datos de la visita a crear: nombre, URL donde está su icono, diseminaciones que formarán la visita y el orden en el que se seguirán.

Salida:

Información del éxito o del error por pantalla.

Funcionamiento:

El usuario introducirá los datos de la visita y elegirá las diseminaciones que formarán parte de la misma.

La aplicación generará un identificador único para la visita.

Acto seguido almacenará la visita en la base de datos, y mostrará un mensaje por pantalla notificando el éxito o el fallo de la operación.

Eliminar una visita.

Descripción:

Con este proceso se puede eliminar del sistema una visita.

Entrada:

Identificador de la visita a borrar de la base de datos.

Salida:

Información del éxito o del error por pantalla.

Funcionamiento:

La aplicación mostrará una lista con todas las visitas existentes en la base de datos, y ofrecerá la posibilidad de buscar para refinar la lista mostrada.

El usuario elegirá una visita de la lista.

La aplicación mostrará los datos de la visita solamente para que el usuario pueda asegurarse que se trata de la visita correcta.

El usuario ordenará a la aplicación que elimine la visita seleccionada y la aplicación borrará la visita de la base de datos.

La aplicación informará tanto del éxito como de cualquier error en la operación.

Modificar una visita.

Descripción:

El usuario podrá cambiar los datos almacenados en la base de datos para una visita determinada.

Entrada:

Identificador de la visita a modificar, nombre, URL del icono, diseminaciones de la visita.

Salida:

Información del éxito o del error por pantalla.

Funcionamiento:

La aplicación mostrará una lista con todas las visitas existentes en la base de datos, y ofrecerá la posibilidad de buscar para refinar la lista mostrada.

El usuario elegirá una visita de la lista.

La aplicación mostrará los datos de la visita para que el usuario pueda cambiar cualquiera de los datos que conforman la visita.

Cuando el usuario haya acabado las modificaciones, indicará a la aplicación que almacene la visita con los cambios realizados, y se enseñará por pantalla si la operación tuvo éxito o no.

Listar visitas disponibles.

Descripción:

Con esta operación el usuario podrá ver todas las visitas que se encuentran almacenadas en el sistema que se ajusten a un criterio de búsqueda basado en el nombre de la visita.

Entrada:

Nombre (total o parcial) de las visitas a recuperar.

Salida:

Listado de todas las visitas encontradas mostrando estos datos: identificador de la visita, nombre y tamaño.

Funcionamiento:

Si el usuario no introduce una cadena a buscar, la aplicación mostrará por defecto todas las visitas.

A.8.3. Gestión de colecciones

Crear una colección.

Descripción:

El usuario podrá crear colecciones (de diseminaciones o de elementos explicativos) mediante esta funcionalidad.

Entrada:

Datos de la colección a crear: nombre, identificadores de las diseminaciones (o de los elementos explicativos en su caso) y el orden que ocuparán en la colección.

Salida:

Información del éxito o del error por pantalla.

Funcionamiento:

El usuario introducirá un nombre para la colección.

Dependiendo del tipo de colección que el usuario elija crear, se mostrará una lista con todas las diseminaciones o bien con todos los elementos explicativos.

El usuario elegirá de la lista los componentes de la nueva colección.

Se generará un identificador de colección único y se almacenará la nueva colección.

Eliminar una colección.

Descripción:

Permite que el usuario pueda eliminar una colección.

Entrada:

Identificador de la colección a eliminar.

Salida:

Información del éxito o del error por pantalla.

Funcionamiento:

La aplicación mostrará una lista con todas las colecciones existentes en la base de datos, y ofrecerá la posibilidad de buscar para refinar la lista mostrada.

El usuario elegirá una colección de la lista.

La aplicación mostrará los datos de la colección solamente para que el usuario pueda asegurarse que se trata de la colección correcta.

El usuario ordenará a la aplicación que elimine la colección seleccionada y la aplicación borrará la colección de la base de datos.

Modificar una colección.

Descripción:

El usuario podrá cambiar los datos almacenados en la base de datos para una colección determinada.

Entrada:

Identificador de la colección a modificar, nombre, identificadores de las diseminaciones (o de los elementos explicativos en su caso) y el orden que ocuparán en la colección.

Salida:

Información del éxito o del error por pantalla.

Funcionamiento:

La aplicación mostrará una lista con todas las colecciones existentes en la base de datos, y ofrecerá la posibilidad de buscar para refinar la lista mostrada.

El usuario elegirá una colección de la lista.

La aplicación mostrará los datos de la colección para que el usuario pueda cambiar cualquiera de los datos que conforman la colección.

Cuando el usuario haya acabado las modificaciones, indicará a la aplicación que almacene la colección con los cambios realizados.

Listar colecciones disponibles.

Descripción:

Esta funcionalidad permitirá que el usuario pueda ver todas las colecciones almacenadas en la base de datos, y que se ajusten a un criterio de búsqueda basado en el nombre de la colección.

Entrada:

Nombre (total o parcial) de las colecciones a recuperar.

Salida:

Listado de todas las colecciones encontradas mostrando estos datos: identificador de la colección, nombre, tipo y tamaño.

Funcionamiento:

Si el usuario no introduce una cadena a buscar, la aplicación mostrará por defecto todas las colecciones.

A.8.4. Gestión de diseminaciones

Crear una diseminación.

Descripción:

El usuario de la aplicación podrá crear una diseminación utilizando esta funcionalidad.

Entrada:

Datos de la diseminación a crear: nombre, URL del icono, identificador de la obra sobre la que versa la diseminación, duración aproximada de la diseminación, identificador del elemento explicativo genérico de la diseminación, y en caso de ser una diseminación de colecciones, identificadores de categorías más los identificadores de las colecciones que se asociarán a cada categoría.

Salida:

Información del éxito o del error por pantalla.

Funcionamiento:

Al usuario se le mostrará una lista con las obras guardadas en el sistema para que escoja una.

Se le mostrará otra lista para que elija el elemento explicativo por defecto de la diseminación.

Y se le mostrará una tercera para que pueda escoger las colecciones que asociará a cada categoría; en ese caso se trataría de una diseminación de colecciones.

Se generará un identificador de diseminación único y se procederá a guardar la nueva diseminación en la base de datos.

Eliminar una diseminación.

Descripción:

Esta funcionalidad permite que el usuario pueda eliminar una diseminación.

Entrada:

Identificador de la diseminación a eliminar.

Salida:

Información del éxito o del error por pantalla.

Funcionamiento:

La aplicación mostrará una lista con todas las diseminaciones existentes en la base de datos, y ofrecerá la posibilidad de buscar para refinar la lista mostrada.

El usuario elegirá una diseminación de la lista.

La aplicación mostrará los datos de la diseminación solamente para que el usuario pueda asegurarse que se trata de la diseminación correcta.

El usuario ordenará a la aplicación que elimine la diseminación seleccionada y la aplicación la borrará de la base de datos.

Si se trata de la diseminación por defecto de una obra, la duración de la diseminación se restará a la duración estimada del área en la que se encuentra la obra.

Modificar una diseminación.

Descripción:

Con esta funcionalidad el usuario podrá cambiar los datos almacenados en la base de datos para una diseminación determinada.

Entrada:

Datos de la diseminación modificada: identificador de la diseminación, nombre, URL del icono, duración aproximada, identificador de la obra sobre la que versa la diseminación, identificador del elemento explicativo genérico de la diseminación, y en caso de ser una diseminación de colecciones, identificadores de categorías más los identificadores de las colecciones que se asociarán a cada categoría.

Salida:

Información del éxito o del error por pantalla.

Funcionamiento:

La aplicación mostrará una lista con todas las diseminaciones existentes en la base de datos, y ofrecerá la posibilidad de buscar para refinar la lista mostrada.

El usuario elegirá una diseminación de la lista.

La aplicación mostrará los datos de la diseminación para que el usuario pueda cambiar cualquiera de los datos que la conforman.

Cuando el usuario haya acabado las modificaciones, indicará a la aplicación que almacene la diseminación con los cambios realizados.

Si se trata de la diseminación por defecto de una obra, la duración de la diseminación se sumará a la duración estimada del área en la que se encuentra la obra.

Listar diseminaciones disponibles.

Descripción:

Esta funcionalidad permitirá que el usuario pueda ver todas las diseminaciones almacenadas en la base de datos, y que se ajusten a un criterio de búsqueda basado en el nombre de la diseminación.

Entrada:

Nombre (total o parcial) de las diseminaciones a recuperar.

Salida:

Listado de todas las diseminaciones encontradas mostrando estos datos: identificador de la diseminación, nombre y nombre de la obra sobre la que trata.

Funcionamiento:

Si el usuario no introduce una cadena a buscar, la aplicación mostrará por defecto todas las diseminaciones.

A.8.5. Gestión de obras expuestas

Dar de alta una obra.

Descripción:

Esta funcionalidad permite dar de alta una obra en el sistema.

Entrada:

Datos de la nueva obra: identificador de la obra en el museo, título, el material de qué está hecha, sus dimensiones, propietario, origen de la obra, su autor, el año de creación, popularidad inicial, la diseminación por defecto que explicará la obra, identificador del área en la que se exhibirá y las coordenadas de su localización sobre un plano de la planta del museo.

Salida:

Información del éxito o del error por pantalla.

Funcionamiento:

El usuario introducirá los datos de la obra y elegirá su diseminación por defecto.

Elegirá el área en la que se exhibirá la obra y su posición en el plano.

Se generará un identificador de obra único.

La nueva obra se almacena en la base de datos.

Se recalcula la duración estimada del área en la que se exhibirá la obra.

Dar de baja una obra.

Descripción:

Esta funcionalidad permite borrar una obra del sistema.

Entrada:

Identificador de la obra a dar de baja.

Salida:

Información del éxito o del error por pantalla.

Funcionamiento:

La aplicación mostrará una lista con todas las obras existentes en la base de datos, y ofrecerá la posibilidad de buscar para refinar la lista mostrada.

El usuario elegirá una obra de la lista.

La aplicación mostrará sus datos solamente para que el usuario pueda asegurarse que se trata de la obra correcta.

El usuario ordenará a la aplicación que elimine la obra seleccionada y la aplicación la borrará de la base de datos.

Se recalcula la duración estimada del área en la que se exhibirá la obra.

Modificar datos de una obra.

Descripción:

Con esta funcionalidad el usuario podrá cambiar los datos almacenados para una obra determinada.

Entrada:

Identificador de la obra en el museo, título, el material de qué está hecha, dimensiones, propietario, origen de la obra, autor, año de creación, la diseminación por defecto que explicará la obra, identificador del área en la que se exhibirá y las coordenadas de su localización sobre un plano de la planta del museo.

Salida:

Información del éxito o del error por pantalla.

Funcionamiento:

La aplicación mostrará una lista con todas las obras existentes en la base de datos, y ofrecerá la posibilidad de buscar para refinar la lista mostrada.

El usuario elegirá una obra de la lista.

La aplicación mostrará los datos de esa obra para que el usuario pueda cambiarlos.

Cuando el usuario haya acabado las modificaciones, indicará a la aplicación que almacene la obra con los cambios realizados.

Se recalcula la duración estimada del área en la que se exhibirá la obra.

Listar obras disponibles.

Descripción:

Esta funcionalidad permitirá que el usuario pueda ver todas las obras almacenadas en la base de datos, y que se ajusten a un criterio de búsqueda basado en el nombre de la obra.

Entrada:

Nombre (total o parcial) de las obras a recuperar.

Salida:

El listado de todas las obras encontradas mostrando estos datos: identificador de la obra, título, autor, año de creación y el identificador de la obra en el museo.

Funcionamiento:

Si el usuario no introduce una cadena a buscar, la aplicación mostrará por defecto todas las obras.

A.8.6. Gestión de elementos explicativos

Crear un elemento explicativo.

Descripción:

El usuario de la aplicación podrá crear un elemento explicativo con esta funcionalidad.

Entrada:

Datos del elemento explicativo a crear: nombre, URL del icono, identificador del elemento explicativo visual, identificador del elemento explicativo sonoro y el identificador de la obra sobre la que trata.

Salida:

Información del éxito o del error por pantalla.

Funcionamiento:

Al usuario se le mostrará una lista con los elementos explicativos visuales del sistema para que escoja uno.

Se le mostrará otra lista para que elija un elemento explicativo sonoro.

Y se le mostrará una tercera para que pueda escoger la obra sobre la que tratará.

Se generará un identificador de elemento explicativo único y se procederá a guardarlo en la base de datos.

Eliminar un elemento explicativo.

Descripción:

Esta funcionalidad permite que el usuario pueda eliminar un elemento explicativo.

Entrada:

Identificador del elemento explicativo a eliminar.

Salida:

Información del éxito o del error por pantalla.

Funcionamiento:

La aplicación mostrará una lista con todos los elementos explicativos existentes en la base de datos, y ofrecerá la posibilidad de buscar para refinar la lista mostrada.

El usuario elegirá un elemento explicativo de la lista.

La aplicación mostrará los datos del elemento explicativo solamente para que el usuario pueda asegurarse que es el correcto.

El usuario ordenará a la aplicación que lo elimine y la aplicación lo borrará de la base de datos.

Modificar un elemento explicativo.

Descripción:

Con esta funcionalidad el usuario podrá cambiar los datos almacenados en la base de datos para un elemento explicativo determinado.

Entrada:

Identificador del elemento explicativo, nombre, URL del icono, identificador del elemento explicativo visual, identificador del elemento explicativo sonoro y el identificador de la obra sobre la que trata.

Salida:

Información del éxito o del error por pantalla.

Funcionamiento:

La aplicación mostrará una lista con todos los elementos explicativos existentes en la base de datos, y ofrecerá la posibilidad de buscar para refinar la lista mostrada.

El usuario elegirá un elemento explicativo de la lista.

La aplicación mostrará los datos de ese elemento explicativo para que el usuario pueda cambiarlos.

Cuando el usuario haya acabado las modificaciones, indicará a la aplicación que almacene el elemento explicativo con los cambios realizados.

Listar elementos explicativos.

Descripción:

Esta funcionalidad permitirá que el usuario pueda ver todos los elementos explicativos almacenados en la base de datos, y que se ajusten a un criterio de búsqueda basado en el nombre del elemento explicativo.

Entrada:

Nombre (total o parcial) de los elementos explicativos a recuperar.

Salida:

El listado de todos los elementos explicativos encontrados mostrando estos datos: identificador del elemento explicativo, nombre, título y autor de la obra que explica.

Funcionamiento:

Si el usuario no introduce una cadena a buscar, la aplicación mostrará por defecto todos los elementos explicativos.

A.8.7. Gestión de contenido multimedia

Crear un elemento explicativo visual.

Descripción:

El usuario de la aplicación podrá crear un elemento explicativo visual con esta funcionalidad.

Entrada:

Datos del elemento explicativo visual a crear: URL del archivo y su formato.

Salida:

Información del éxito o del error por pantalla.

Funcionamiento:

Se generará un identificador único para el elemento explicativo visual y se procederá a guardarlo en la base de datos.

Eliminar un elemento explicativo visual.

Descripción:

Esta funcionalidad permite que el usuario pueda eliminar un elemento explicativo visual.

Entrada:

Identificador del elemento explicativo visual a eliminar.

Salida:

Información del éxito o del error por pantalla.

Funcionamiento:

La aplicación mostrará una lista con todos los elementos explicativos visuales existentes en la base de datos, y ofrecerá la posibilidad de buscar para refinar la lista mostrada.

El usuario elegirá un elemento explicativo visual de la lista.

La aplicación mostrará los datos del elemento explicativo visual solamente para que el usuario pueda asegurarse que es el correcto.

El usuario ordenará a la aplicación que lo elimine y la aplicación lo borrará de la base de datos.

Modificar un elemento explicativo visual.

Descripción:

Con esta funcionalidad el usuario podrá cambiar los datos almacenados de un elemento explicativo visual determinado.

Entrada:

Identificador del elemento explicativo visual, URL del fichero y su formato.

Salida:

Información del éxito o del error por pantalla.

Funcionamiento:

La aplicación mostrará una lista con todos los elementos explicativos visuales existentes en la base de datos, y ofrecerá la posibilidad de buscar para refinar la lista mostrada.

El usuario elegirá un elemento explicativo visual de la lista.

La aplicación mostrará los datos de ese elemento explicativo visual para que el usuario pueda cambiarlos.

Cuando el usuario haya acabado las modificaciones, indicará a la aplicación que almacene el elemento explicativo visual con los cambios realizados.

Listar elementos explicativos visuales.

Descripción:

Esta funcionalidad permitirá que el usuario pueda ver todos los elementos explicativos visuales almacenados en la base de datos, y que se ajusten a un criterio de búsqueda basado en el formato o el nombre del fichero del elemento explicativo.

Entrada:

Formato o URL (total o parcial) de los ficheros de los elementos explicativos visuales a recuperar.

Salida:

El listado de todos los elementos explicativos visuales encontrados mostrando estos datos: identificador del elemento explicativo visual, URL y formato.

Funcionamiento:

Si el usuario no introduce ningún parámetro de búsqueda, la aplicación mostrará por defecto todos los elementos explicativos visuales.

Crear un elemento explicativo sonoro.

Descripción:

El usuario de la aplicación podrá crear un elemento explicativo sonoro con esta funcionalidad.

Entrada:

Datos del elemento explicativo sonoro a crear: URL del archivo, su formato, el idioma de la explicación y el elemento explicativo visual con el que está relacionado.

Salida:

Información del éxito o del error por pantalla.

Funcionamiento:

La aplicación mostrará una lista con todos los elementos explicativos visuales de la base de datos para que escoja uno.

Se generará un identificador único para el elemento explicativo sonoro y se procederá a guardarlo en la base de datos.

Eliminar un elemento explicativo sonoro.

Descripción:

Esta funcionalidad permite que el usuario pueda eliminar un elemento explicativo sonoro.

Entrada:

Identificador del elemento explicativo sonoro a eliminar.

Salida:

Información del éxito o del error por pantalla.

Funcionamiento:

La aplicación mostrará una lista con todos los elementos explicativos sonoros existentes en la base de datos, y ofrecerá la posibilidad de buscar para refinar la lista mostrada.

El usuario elegirá un elemento explicativo sonoro de la lista.

La aplicación mostrará los datos del elemento explicativo sonoro solamente para que el usuario pueda asegurarse que es el correcto.

El usuario ordenará a la aplicación que lo elimine y la aplicación lo borrará de la base de datos.

Modificar un elemento explicativo sonoro.

Descripción:

Con esta funcionalidad el usuario podrá cambiar los datos almacenados de un elemento explicativo sonoro.

Entrada:

Identificador del elemento explicativo sonoro, URL del fichero, su formato, el idioma de la explicación y el elemento explicativo visual con el que está relacionado.

Salida:

Información del éxito o del error por pantalla.

Funcionamiento:

La aplicación mostrará una lista con todos los elementos explicativos sonoros existentes en la base de datos, y ofrecerá la posibilidad de buscar para refinar la lista mostrada.

El usuario elegirá un elemento explicativo sonoro de la lista.

La aplicación mostrará los datos de ese elemento explicativo sonoro para que el usuario pueda cambiarlos.

Cuando el usuario haya acabado las modificaciones, indicará a la aplicación que almacene el elemento explicativo sonoro con los cambios realizados.

Listar elementos explicativos sonoros.

Descripción:

Esta funcionalidad permitirá que el usuario pueda ver todos los elementos explicativos sonoros almacenados en la base de datos, y que se ajusten a un criterio de búsqueda basado en el formato, el idioma o el nombre del fichero del elemento explicativo.

Entrada:

Formato, idioma o URL (total o parcial) de los ficheros de los elementos explicativos sonoros a recuperar.

Salida:

El listado de todos los elementos explicativos sonoros encontrados mostrando estos datos: identificador del elemento explicativo sonoro, URL, idioma y formato.

Funcionamiento:

Si el usuario no introduce ningún parámetro de búsqueda, la aplicación mostrará por defecto todos los elementos explicativos sonoros.

A.8.8. Gestión de visitantes

Dar de alta un visitante.

Descripción:

Esta funcionalidad permite dar de alta un visitante en la base de datos.

Entrada:

Nombre, apellidos, alias, sexo, correo electrónico, nombre de usuario, contraseña, avatar y descripción.

Salida:

Información del éxito o del error por pantalla.

Funcionamiento:

El usuario introducirá los datos del visitante que se quiere registrar.

Se generará un identificador único de usuario de *MoMo* y se procederá a guardarlo en la base de datos.

Dar de baja un visitante.

Descripción:

Esta funcionalidad permite eliminar un visitante de la base de datos.

Entrada:

Identificador del visitante a dar de baja.

Salida:

Información del éxito o del error por pantalla.

Funcionamiento:

La aplicación mostrará una lista con todos los visitantes existentes en la base de datos, y ofrecerá la posibilidad de buscar para refinar la lista mostrada.

El usuario elegirá un visitante de la lista.

La aplicación mostrará sus datos solamente para que el usuario pueda asegurarse.

El usuario ordenará a la aplicación que borre los datos del visitante seleccionado y la aplicación lo borrará de la base de datos.

Consultar datos de un visitante

Descripción:

El usuario podrá examinar los datos de cualquier visitante dado de alta en el sistema.

Entrada:

Identificador del visitante del cual se quiere recuperar su información.

Salida:

Nombre, apellidos, alias, sexo, correo electrónico, avatar, descripción, nombre de usuario, contraseña, indicador de si ha silenciado a todo el mundo, las fechas de las tres últimas sesiones en el sistema y la lista de grupos a los que pertenece.

Funcionamiento:

La aplicación mostrará una lista con todos los visitantes existentes en la base de datos, y ofrecerá la posibilidad de buscar para refinar la lista mostrada.

El usuario elegirá un visitante de la lista y la aplicación mostrará sus datos.

Consultar datos de un grupo

Descripción:

El usuario podrá examinar los datos de cualquier grupo existente en la base de datos.

Entrada:

Identificador del grupo del cual se quiere recuperar su información.

Salida:

Nombre del grupo, descripción, tipo de grupo, número de miembros, alias del gestor y la lista de los miembros del grupo, con

los alias y los indicadores de si han silenciado al grupo y si han sido silenciados por el gestor.

Funcionamiento:

La aplicación mostrará una lista con todos los grupos existentes en la base de datos, y ofrecerá la posibilidad de buscar para refinar la lista mostrada.

El usuario elegirá un grupo de la lista y la aplicación mostrará sus datos.

Listar visitantes registrados.

Descripción:

Esta funcionalidad permitirá que el usuario pueda ver todos los visitantes dados de alta en el sistema, y que se ajusten a un criterio de búsqueda basado en el alias, el nombre y apellidos o en la fecha de última visita.

Entrada:

Nombre, apellidos o alias (total o parcial) o la fecha de la última visita al museo de los visitantes a recuperar.

Salida:

El listado de todos los visitantes encontrados mostrando estos datos: identificador del visitante, alias, nombre, apellidos y fecha de última visita.

Funcionamiento:

Si el usuario no introduce ningún parámetro de búsqueda, la aplicación mostrará por defecto todos los visitantes.

Listar visitantes en museo.

Descripción:

Esta funcionalidad permitirá que el usuario pueda ver todos los visitantes dados de alta en el sistema que estén realizando una visita en ese momento, y que además se ajusten a un criterio de búsqueda basado en el alias, el nombre o los apellidos.

Entrada:

Nombre, apellidos o alias (total o parcial) de los visitantes a recuperar.

Salida:

El listado de todos los visitantes encontrados mostrando estos datos: identificador del visitante, alias, nombre, apellidos y última área visitada.

Funcionamiento:

Si el usuario no introduce ningún parámetro de búsqueda, la aplicación mostrará por defecto todos los visitantes.

Listar grupos

Descripción:

Esta funcionalidad permitirá que el usuario pueda ver todos los grupos que existan en ese momento, y que además se ajusten a un criterio de búsqueda basado en el alias del gestor o el nombre del grupo.

Entrada:

Nombre del grupo o alias (total o parcial).

Salida:

El listado de todos los grupos encontrados mostrando estos datos: identificador del grupo, nombre, alias del gestor y el número de miembros.

Funcionamiento:

Si el usuario no introduce ningún parámetro de búsqueda, la aplicación mostrará por defecto todos los grupos.

A.8.9. Envío de notificaciones

Enviar una difusión.

Descripción:

El usuario podrá enviar una notificación a todos los visitantes que estén en el museo.

Entrada:

Los datos de la notificación a enviar, que serán el asunto y el texto.

Salida:

Información del éxito o del error por pantalla.

Funcionamiento:

Se genera un identificador de notificación único y se crea una nueva notificación en el sistema. Se añade una entrada asociando la notificación a cada visitante.

Enviar una notificación.

Descripción:

Con esta operación el usuario podrá enviar una notificación a una serie de visitantes que estén en el museo.

Entrada:

Los datos de la notificación a enviar: el asunto, el texto y la lista de los identificadores de los visitantes destinatarios de la notificación.

Salida:

Información del éxito o del error por pantalla.

Funcionamiento:

El usuario escribirá el asunto y el texto del mensaje a enviar.

Al usuario se le muestra en pantalla una lista con todos los visitantes actuales del museo.

De esa lista, seleccionará los visitantes que recibirán la notificación.

Rutinas HLSL para números aleatorios

```
int urandi (int seed)
{
/*
implementa el generador congruencial lineal
multiplicativo según el método de Schrage
(16807 * seed) mod 16777213
*/
 int k, calc, q, r;
 q = 3161;
 r = 1786;
 k = (int) (seed / q);
 calc = 5307 * (seed - k*q) - r*k;
 if (calc < 0)
   calc += 16777213;
 return calc;
}
float urandf (int urndi) {
      return  ((1.0f*urndi)/(16777213+1.0f));
}
float urandfi (int urndi, float minf, float maxf) {
      return  (minf + (maxf*urndi)/(16777213.0f+1.0f));
}
```

Programa HLSL de vértices de la Fase 2A

```
#include "randoms.hlsl"
matrix ViewProjMatrix;
/* textures */
sampler AtractivenessTex;
sampler GraphTex;
sampler ForbiddenTex;
sampler TourTex;
/* uniform parameters */
uniform int n, h, col, prev_col;
uniform int starting_node, ending_node;
uniform float max_atrac, min_atrac, Tmax, s_end, w_end;
uniform float posX_end, posY_end, node_offset;
uniform float node_bias, ant_offset;
uniform float ant_bias, tour_offset, tour_bias;
void main
(
in float3 pos: POSITION,
out float4 posO: POSITION,
```

```
// (BeneficioAc, CosteAc)
out float2 acumulado: TEXCOORD0,
// (Semilla [y en el signo va el desactivar: + es 1; -
// es 0]    , NodoVisitado)
out float2 outputs1: TEXCOORD1
)
{
float4 posicion, info, coord, visitado, desc_nodo_ult;
flota4 desc_nodo, atrac;
float distancia, distanciaFin, distri, s_ac, w_ac;
int nodo_ult, q, desactivar, seed, old_seed;
float2 pos_end;
posicion.x = col;
posicion.y = (float) h - pos.y;
posicion.w = 1.0f;

/* información de la iteración anterior */
coord.x = prev_col*tour_offset + tour_bias;
coord.y = ant_offset*pos.y + ant_bias;
coord.z = coord.w = 0.0f;
info = tex2Dlod (TourTex, coord);

desactivar = -1;

/* ver si el nodo ha sido visitado */
coord.x = pos.x*node_offset + node_bias;
visitado = tex2Dlod (ForbiddenTex, coord);

nodo_ult = abs (info.x) - 1;
q = sign (info.x);
w_ac = info.w;
s_ac = abs (info.z);
seed = info.y;

if (info.z >=0 && visitado.x == 0)
/* si la hormiga está activa y el nodo no ha sido
visitado */
{
  coord.x = nodo_ult*node_offset + node_bias;
  coord.y = 0.0f;
  desc_nodo_ult = tex1Dlod (GraphTex, coord);

  coord.x = pos.x*node_offset + node_bias;
  coord.y = 0.0f;
  desc_nodo = tex1Dlod (GraphTex, coord);
  distancia = distance (desc_nodo_ult.xy,desc_nodo.xy);

  pos_end.x = posX_end;
  pos_end.y = posY_end;
```

```
  distanciaFin = distance (desc_nodo.xy, pos_end.xy);

  if (w_ac + distancia + desc_nodo.w + distanciaFin +
w_end >= Tmax)
    {
      distancia=distance (desc_nodo_ult.xy, pos_end.xy);
      /* cerrar camino por restricción */
      outputs1.y = ending_node;
      desactivar = 1;
      acumulado.y = w_ac + distancia + w_end;
      acumulado.x = s_ac + s_end;
      posicion.z = max_atrac;
    }
  else
    {
      /* se sigue construyendo el camino */
      coord.x = pos.x*node_offset + node_bias;
      coord.y = nodo_ult*node_offset + node_bias;
      coord.z = coord.w = 0.0f;
      atrac = tex2Dlod (AtractivenessTex, coord);
      if (q > 0)
      /* explotación */
      {
        posicion.z = min_atrac + (max_atrac - atrac.x);
      }
      else
    /* exploración */
    {
      seed = urandi (seed);
      distri = urandfi ( seed, min_atrac, max_atrac);
      posicion.z = clamp (min_atrac+abs(atrac.x-distri),
                          min_atrac, max_atrac);
    }
    outputs1.y = pos.x;
    acumulado.y = w_ac + distancia + desc_nodo.w;
    acumulado.x = s_ac + desc_nodo.z;
  }
}
else
{
  if (pos.x == starting_node)
  {
  /* propagación del final del camino */
  /*
- si está inactiva la propagación es obvia ya que ningún
vértice propagaría la solución hacía la última columna
- si está activa la propagación se debe a la posibilidad
de que es posible que estén todos los nodos visitados a
excepción del final y que no se haya cerrado camino por
```

```
violación de Tmax (el buffer de vértices no tiene los
vértices correspondientes al nodo final para evitar el
prematuro cierre del camino y consumir tanto como se
pueda Tmax).*/
    acumulado.x = s_ac;
    acumulado.y = w_ac;
    desactivar = 1;
    posicion.z = max_atrac;
    outputs1.y = -1*nodo_ult;
  }
  else
  /*descarte del vértice, fuera del farplane*/
                 posicion.z = max_atrac + 10000.0f;
  }
  outputs1.x = desactivar * seed;
  posicion.x = col;
  /* hacer la transformación del vértice efectiva */
  posO = mul (ViewProjMatrix, posicion);
}
```

Programas HLSL de fragmentos de la Fase 2A

```
#include "randoms.hlsl"
uniform int ending_node;
uniform float q0, Tmax;
void main
(
 in float4 position: POSITION,
 // (BeneficioAc, CosteAc)
 in float2 acumulado: TEXCOORD0,
 // (Semilla [y en el signo va el desactivar: + es 1;
 // - es 0] , NodoVisitado)
 in float2 inputs1: TEXCOORD1,
 out float4 colorO: COLOR
)
{
int activa, seed, q;
float w_ac, s_ac, qn;
q = 1;
/* Desactivar es NO */
if (inputs1.x < 0)
{
  /* NodoVisitado es el Nodo Final */
  if (inputs1.y == ending_node)
    activa = -1;
  else
    activa = 1;
```

```
  seed = urandi (abs (inputs1.x));
  qn = seed/16777213.0f;
  /* determinar explotación o exploración para sig. */
  if (qn <= q0)
    q = 1;
  else
    q = -1;
}
else
  activa = -1;
w_ac = acumulado.y;
s_ac = acumulado.x;
if (w_ac > Tmax && inputs1.y >= 0)
  /* score penalized de Ramalinho y Serra */
  s_ac = (acumulado.x*(Tmax/w_ac));
seed = urandi (seed);
/*
codificar en el color de salida: nuevoBeneficioAc,
nuevoCosteAc, nuevaSemilla, nuevoNodoVisitado, q, Activa
*/
colorO.x = q*(abs (inputs1.y) + 1);
colorO.y = seed;
colorO.z = s_ac;
colorO.z *= activa;
colorO.w =  w_ac;
}
```

Programa HLSL de vertices de la Fase 2B

```
matrix ViewProjMatrix;
sampler TourTex;
/* uniform parameters */
uniform float max_atrac, min_atrac;
uniform float tour_offset, ant_offset;
uniform float tour_bias, ant_bias;
uniform int h, prev_col;
void main
(
in float3 pos: POSITION,
out float4 posO: POSITION
)
{
  float4 info, coord;
  int nodo_ult, q;
  float s_ac, w_ac;
```

```
  posO.x = pos.x;
  posO.y = (float) h - pos.y;
  posO.w = 1.0f;

  /* información del paso de la fase anterior */
  coord.x = prev_col*tour_offset + tour_bias;
  coord.y = ant_offset*pos.y + ant_bias;
  coord.z = coord.w = 0.0f;
  info = tex2Dlod (TourTex, coord);

  q = sign (info.x);
  nodo_ult = abs (info.x) - 1;
  s_ac = abs (info.z);
  w_ac = info.w;
  // Si mi nodo es el ultimo visitado
  if (nodo_ult == pos.x)
    // hay que hacer visible el vértice
    posO.z = min_atrac;
  else
    // hay que descartar el vértice: fuera del far plane
    posO.z = max_atrac + 10000.0f;

  posO = mul (ViewProjMatrix, posO);
}
```

Programa HLSL de fragmentos de la Fase 2B

```
void main
(
  in float4 position: POSITION,
  out float4 colorO: COLOR
)
{
  colorO = float4 (1.0f, 0.0f, 0.0f, 0.0f);
}
```

ANEXO C. Programas HLSL de GPU-Fragment-OCH-OP

Programa HLSL de Vértices de la Fase 2A

```
/* uniform parameters */
uniform int h;
uniform int col;
void main
(
in float3 pos: POSITION,
out float4 posO: POSITION,
/* nodo_propuesto, hormiga (ésta es interpolada por el
rasterizador) */
out float2 outputData: TEXCOORD0
)
{
float4 posicion;
outputData.x = pos.x;
posicion.x = col;
if (pos.y == -1)
{
      posicion.y = (float) h;
      outputData.y = 0.0f;
}
else
{
      posicion.y = 0.0f;
      outputData.y = (float)h;
}

posicion.z = pos.x;
posicion.w = 1.0f;
posO = mul (ViewProjMatrix, posicion);
}
```

Programa de Fragmentos Fase 2A

```
#include "randoms.hlsl"
/* textures */
sampler AtractivenessTex;
sampler GraphTex;
sampler ForbiddenTex;
sampler TourTex;
/* uniform parameters */
uniform float max_atrac, min_atrac, Tmax;
```

```
uniform int prev_col, starting_node, ending_node;
uniform float posX_end, posY_end, s_end, w_end;
uniform float node_offset, node_bias;
uniform float ant_offset, ant_bias;
uniform float tour_offset, tour_bias, q0;
void main
(
      in float4 position: POSITION,
      /* TEXCOORD0-> X: nodo, Y: hormiga */
      in float2 inputData: TEXCOORD0,

out float4 colorO: COLOR,
      out float atractiveness: DEPTH
)
{
float4 posicion, info, coord, visitado;
float4 desc_nodo_ult, desc_nodo, atrac;
float distancia ,distanciaFin;
float distri, s_ac, w_ac;
int desactivar, seed, old_seed, activa, nodo_ult;
float2 pos_end;
float new_s_ac, new_c_ac, selected_node, qn;

/* información de la iteración anterior */
coord.x = prev_col*tour_offset + tour_bias;
coord.y = ant_offset*inputData.y + ant_bias;
coord.z = coord.w = 0.0f;
info = tex2D (TourTex, coord);

/* saber si el nodo ha sido visitado */
coord.x = inputData.x*node_offset + node_bias;
visitado = tex2D (ForbiddenTex, coord);

desactivar = -1;
/* obtener la información de la anterior iteración */
nodo_ult = info.x;
w_ac = info.w;
s_ac = abs (info.z);
seed = info.y;
activa = step (0, info.z);

if (activa==1 && visitado.x == 0)
/* si la hormiga está activa y el nodo no ha sido
visitado */
```

```
{
coord.x = nodo_ult*node_offset + node_bias;
coord.y = 0.0f;
desc_nodo_ult = tex1D (GraphTex, coord);
coord.x = inputData.x*node_offset + node_bias;
coord.y = 0.0f;
desc_nodo = tex1D (GraphTex, coord);

distancia=distance(desc_nodo_ult.xy,desc_nodo.xy);

pos_end.x = posX_end;
pos_end.y = posY_end;
distanciaFin=distance (desc_nodo.xy, pos_end.xy);

if (w_ac + distancia + desc_nodo.w + distanciaFin +
w_end >= Tmax)
{
      distancia = distance (desc_nodo_ult.xy,
pos_end.xy);

      /* cerrar camino por restricción */
      selected_node = ending_node;
      desactivar = 1;
      new_c_ac = w_ac + distancia + w_end;
      new_s_ac = s_ac + s_end;
      atractiveness = 1.0f;
}
else
{
/* se sigue construyendo el camino */
coord.x = inputData.x*node_offset + node_bias;
      coord.y = nodo_ult*node_offset + node_bias;
      coord.z = coord.w = 0.0f;
      atrac = tex2D (AtractivenessTex, coord);
      seed = urandi (seed);
      qn = seed/16777213.0f;
      if (qn <= q0)
      /* explotación */
      {
            atractiveness = 1.0f - ((atrac.x -
min_atrac)/(max_atrac-min_atrac));
      }
      else
```

```
      /* exploración */
      {
            seed = urandi (seed);
            distri = urandfi ( seed, min_atrac,
max_atrac);
            atractiveness = abs (atrac.x -
distri)/(max_atrac-min_atrac);
      }

      selected_node = inputData.x;
      new_c_ac = w_ac + distancia + desc_nodo.w;
      new_s_ac = s_ac + desc_nodo.z;

}
}
else
{
/* si se trata del nodo inicial */
if (inputData.x == starting_node)
{
  /* propagación del final del camino */
  /*
- si está inactiva la propagación es obvia ya que ningún
vértice propagaría la solución hacía la última columna
- si está activa la propagación se debe a la posibilidad
de que es posible que estén todos los nodos visitados a
excepción del final y que no se haya cerrado camino por
violación de Tmax (el buffer de vértices no tiene los
vértices correspondientes al nodo final para evitar el
prematuro cierre del camino y consumir tanto como se
pueda Tmax).*/
      new_s_ac = s_ac;
      new_c_ac = w_ac;
      desactivar = 1;
      atractiveness = 1.0f;
      selected_node = nodo_ult;
}
else
/* forzar el descarte del vértice */
{
      clip (-1.0f);
      atractiveness = 2.0f;
}
}
```

```
if (atractiveness != 2.0f)
{
/* si superamos la restricción y no sea el fragmento de
propagación */
if (new_c_ac > Tmax && activa == 1 && desactivar == 1)
      new_s_ac = (new_s_ac*(Tmax/new_c_ac));

/* si desactivar es no */
if (desactivar == -1)
      activa = 1;
else
      activa = -1;

seed = urandi (seed);
colorO.x = selected_node;
colorO.y = seed;
colorO.z = new_s_ac;
colorO.z *= activa;
colorO.w =  new_c_ac;
}
}
```

Programa HLSL de Vértices de la Fase 2B

```
matrix ViewProjMatrix;
void main
(
in float3 pos: POSITION,
in float2 inputData: TEXCOORD0,
out float4 posO: POSITION,
out float2 outputData: TEXCOORD0
)
{
posO.xyz = pos.xyz;
posO.w = 1.0f;
posO = mul (ViewProjMatrix, posO);
outputData = inputData;
}
```

Programa HLSL de Fragmentos de la Fase 2B

```
sampler TourTex;
int prev_col;
float tour_offset, tour_bias, ant_offset, ant_bias;
void main
(
/* interpolados la hormiga y el nodo (h, n)*/
in float2 inputData: TEXCOORD0,
out float4 colorO: COLOR,
out float z: DEPTH
)
{
float4 coord, info;
float node, ant, goTo;

      ant = round (inputData.x);
coord.x = prev_col*tour_offset + tour_bias;
coord.y = ant * ant_offset + ant_bias;
coord.z = coord.w = 0.0f;
info = tex2D (TourTex, coord);
node = round (info.x);
goTo = round (inputData.y);
if ( node == goTo)
{
      colorO = float4 (node, 0.0f, 0.0f, 0.0f);
      z = 0.0f;
}
else
{
      colorO = float4 (0.5f, 0.0f, 0.0f, 0.0f);
      clip (-1.0f);
}
}
```