

ón323Planificaciónchapter.3

ón651Implementaciónchapter.6

imágenessection.A.3 ágenesA.363Carga de imágenesfigure.A.3

ño545Diseñochapter.5

ágenesA.363Carga de

ónA.5.267Ratónsubsection.A.5.2





*ugr*

Universidad  
de **Granada**

TRABAJO FIN DE GRADO  
GRADO EN INGENIERÍA INFORMÁTICA

# Desarrollo de Módulos de Visualización para Sistema de Recuperación de Imágenes

---

**Autor**

Alejandro Casado Quijada

**Directores**

Jesús Chamorro Martínez



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE  
TELECOMUNICACIÓN

—  
Granada, September 12, 2016

## Desarrollo de Módulos de Visualización para Sistema de Recuperación de Imágenes

Alejandro Casado Quijada

**Palabras clave:** *visualización imágenes, java 3d, 3d, recuperación imágenes, CBIR*

### Resumen

Debido al avance de la tecnología, cada vez disponemos de más dispositivos con la capacidad de capturar imágenes. Esto ha provocado un importante incremento en las bases de datos de imágenes, lo que a su vez ha propiciado la aparición de metodologías para solventar el problema de la manipulación, gestión y recuperación de dicha información. En este caso cuando hablamos de metodologías nos referimos a los sistemas de recuperación de información, basados fundamentalmente en descriptores de bajo nivel (color, textura, etc.) obtenidos directamente a partir de la imagen. Estos sistemas se denominan CBIR.

El objetivo de este proyecto es desarrollar módulos para visualizar la información obtenida mediante el uso de sistemas de recuperación de información, en concreto Java Multimedia Retrieval©. El entorno en el que se representa la información es un entorno tridimensional, en el cual el usuario puede moverse libremente, modificar ciertos parámetros para mejorar su experiencia y obtener datos sobre lo que se muestra en dicho entorno.

La principal ventaja de estos módulos respecto a otros es el uso de un entorno en tres dimensiones. La mayoría de los CBIR utilizan visualizaciones en dos dimensiones, lo que puede resultar problemático si se quiere observar el resultado de la consulta usando varios descriptores. Este proyecto cuenta con visualizaciones que solo requieren un único descriptor, aunque pueden usarse con mas de uno, y otras que requieren varios descriptores para funcionar correctamente.

Para el desarrollo de los distintos módulos se ha usado Java 3D. Se trata de una API usada para la creación de gráficos 3D mediante el uso del lenguaje de programación Java.

## Visualization Module Development for Image Retrieval System

Alejandro Casado Quijada

**Keywords:** *image visualization, java 3d, 3d, image retrieval, CBIR*

### Abstract

Due to the advancement of technology, more and more devices have the ability to capture images. This has caused a significant increase in image databases, which has led to a significant increase in image databases, which in turn has led to the emergence of methodologies to solve the problem of handling, management and retrieval of such information. In this case when we talk about methodology we refer to the information retrieval systems, based primarily on low-level descriptors (color, texture, etc.) obtained directly from the image. These systems are called CBIR.

The aim of this project is to develop modules to display the information obtained by using information retrieval systems, specifically Java Multimedia Retrieval©. The environment in which information is represented is a three-dimensional environment in which the user can move freely through it, modify certain parameters to improve the experience and get data on what is shown in that environment.

The main advantage of these modules over others is the use of a three-dimensional environment. Most CBIR use two-dimensional views which can be problematic if you want to see the result of the query using various descriptors. This project has visualizations that only require a single descriptor, but can be used with more than one, and others that require various descriptors to function properly.

For the development of the different modules has been used Java 3D. It is an API used for creating 3D graphics using the Java programming language.

---

Yo, **Alejandro Casado Quijada**, alumno de la titulación Grado en Ingeniería informática de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación** con DNI 75928287C, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Alejandro Casado Quijada

Granadaa September 12, 2016.

---

**Jesús Chamorro Martínez**, profesor del Departamento de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Granada.

**Informa**

Que el presente trabajo, titulado ***Desarrollo de Módulos de Visualización para Sistema de Recuperación de Imágenes***, ha sido realizado bajo su supervisión por **Alejandro Casado Quijada**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expide y firma el presente informe en Granadaa September 12, 2016.

**El tutor:**

**Jesús Chamorro Martínez**



# Agradecimientos

Quiero dar las gracias a mi familia, amigos y profesores porque sin ellos nada de esto hubiera sido posible.



# Contents

<b>1</b>	<b>Introducción</b>	<b>15</b>
1.1	Motivación . . . . .	15
1.2	Estado del arte . . . . .	15
1.2.1	CBMIR . . . . .	16
1.2.2	GaZIR . . . . .	17
1.2.3	Similarity-Based Visualization of Large Image Collections . . . . .	18
1.2.4	Visual Clustering of Image Search Results . . . . .	19
1.3	Objetivos . . . . .	20
1.3.1	Objetivos principales . . . . .	20
<b>2</b>	<b>Especificacion de requisitos</b>	<b>21</b>
2.1	Requisitos funcionales . . . . .	21
2.2	Requisitos no funcionales . . . . .	22
2.3	Requisitos de información . . . . .	22
<b>3</b>	<b>Planificación</b>	<b>23</b>
3.1	Primer acercamiento . . . . .	23
3.2	Segundo acercamiento . . . . .	24
3.3	Tercer acercamiento . . . . .	25
3.4	Documentación y detalles . . . . .	25
<b>4</b>	<b>Análisis</b>	<b>27</b>
4.1	Descripción del problema . . . . .	27
4.2	Soluciones propuestas . . . . .	27
4.2.1	Visualización secuencial . . . . .	28
4.2.2	Visualización espiral . . . . .	29
4.2.3	Visualización camino . . . . .	30
4.2.4	Visualización cartesiana 2D . . . . .	32
4.2.5	Visualización polar 2D . . . . .	32
4.2.6	Visualización cartesiana 3D . . . . .	33
4.2.7	Visualización polar 3D . . . . .	34
4.2.8	Visualización n caminos . . . . .	35
4.3	Casos de uso . . . . .	37

4.4	Diagramas de secuencia . . . . .	39
<b>5</b>	<b>Diseño</b>	<b>45</b>
5.1	Introducción al diseño . . . . .	45
5.2	Arquitectura . . . . .	45
5.3	Netbeans . . . . .	46
5.4	Java 3D . . . . .	47
5.5	Diagrama de clases del diseño . . . . .	48
<b>6</b>	<b>Implementación</b>	<b>51</b>
6.1	Descripción de la implementación . . . . .	51
6.2	Abstract3DPanel . . . . .	51
6.2.1	Constructor por defecto . . . . .	51
6.2.2	Método add . . . . .	52
6.2.3	Métodos para la interacción . . . . .	52
6.2.4	Método obtención de información . . . . .	52
6.2.5	Métodos para obtener imágenes . . . . .	52
6.2.6	Método para obtener información . . . . .	52
6.2.7	Métodos abstractos . . . . .	52
6.2.8	Método guía de posición . . . . .	52
6.2.9	Método para pintar imágenes . . . . .	53
6.3	SecuencialPanel . . . . .	53
6.4	SpiralPanel . . . . .	53
6.5	PathPanel . . . . .	53
6.6	Cartesian2DPanel . . . . .	53
6.7	Cartesian3DPanel . . . . .	53
6.8	Polar2DPanel . . . . .	53
6.9	Polar3DPanel . . . . .	54
6.10	NPathPanel . . . . .	54
<b>7</b>	<b>Pruebas</b>	<b>55</b>
<b>8</b>	<b>Conclusiones y trabajos futuros</b>	<b>57</b>
8.1	Conclusiones . . . . .	57
8.2	Trabajos futuros . . . . .	57
	<b>Bibliografía</b>	<b>60</b>
<b>A</b>	<b>Manual de usuario</b>	<b>61</b>
A.1	Instalación de Java 3D . . . . .	61
A.2	Introducción . . . . .	61
A.3	Carga de imágenes . . . . .	62
A.4	Elección de descriptores . . . . .	64
A.5	Interacción . . . . .	64

---

A.5.1	Teclado . . . . .	65
A.5.2	Ratón . . . . .	66
A.6	Visualizaciones . . . . .	67
A.7	Opciones . . . . .	69
A.7.1	Reiniciar posición . . . . .	70
A.7.2	Cambio de primitiva . . . . .	71
A.7.3	Niebla . . . . .	73
A.7.4	Fondo . . . . .	74
<b>B</b>	<b>Licencia</b>	<b>77</b>



# List of Figures

1.1	Salida consulta CBMIR . . . . .	16
1.2	Salida consulta GaZIR . . . . .	17
1.3	Visualización espiral SBVLIC . . . . .	18
1.4	Visualización nube SBVLIC . . . . .	18
1.5	Puntos de atracción VCISR . . . . .	19
1.6	Salida VCISR . . . . .	19
3.1	Diagrama Gantt primer acercamiento . . . . .	23
3.2	Diagrama Gantt segundo acercamiento . . . . .	24
3.3	Diagrama Gantt tercer acercamiento . . . . .	25
3.4	Diagrama Gantt documentación y detalles . . . . .	26
4.1	Visualización secuencial . . . . .	28
4.2	Visualización espiral . . . . .	30
4.3	Visualización camino 1 . . . . .	31
4.4	Visualización camino 2 . . . . .	31
4.5	Visualización cartesiana 2D . . . . .	32
4.6	Visualización polar 2D . . . . .	33
4.7	Visualización cartesiana 3D . . . . .	34
4.8	Visualización polar 3D . . . . .	35
4.9	Visualización n-caminos n=6 . . . . .	36
4.10	Visualización n-caminos n=10 . . . . .	36
4.11	Caso de uso gestión de imágenes . . . . .	37
4.12	Caso de uso gestión de descriptores . . . . .	38
4.13	Caso de uso gestión de interacción . . . . .	38
4.14	Caso de uso gestión de visualización . . . . .	39
4.15	Diagrama de secuencia carga de imagen . . . . .	39
4.16	Diagrama de secuencia carga de imágenes . . . . .	40
4.17	Diagrama de secuencia elección imagen consulta . . . . .	40
4.18	Diagrama de secuencia elección descriptor . . . . .	40
4.19	Diagrama de secuencia elección de descriptores . . . . .	41
4.20	Diagrama de secuencia elección de consulta . . . . .	41
4.21	Diagrama de secuencia rotación escena . . . . .	41

4.22	Diagrama de secuencia desplazamiento . . . . .	42
4.23	Diagrama de secuencia modificación escena . . . . .	42
4.24	Diagrama de secuencia reinicio de posición . . . . .	42
4.25	Diagrama de secuencia elección visualización . . . . .	43
4.26	Diagrama de secuencia obtención de información de la imagen . . . . .	43
5.1	Desarrollo de interfaces usando Netbeans . . . . .	46
5.2	Estructura Java 3D . . . . .	48
5.3	Diagrama de clases . . . . .	49
A.1	Descarga de Java 3D . . . . .	61
A.2	Pantalla principal . . . . .	62
A.3	Diálogo apertura imágenes . . . . .	63
A.4	Imágenes abiertas . . . . .	63
A.5	Elección de descriptores 1 . . . . .	64
A.6	Elección de descriptores 2 . . . . .	64
A.7	Información de la imagen . . . . .	65
A.8	Ejemplo rotación hacia arriba teclado . . . . .	66
A.9	Ejemplo rotación eje x ratón . . . . .	67
A.10	Ejemplo de visualización 1 . . . . .	68
A.11	Ejemplo de visualización 2 . . . . .	69
A.12	Menú desplegable de opciones . . . . .	70
A.13	Reinicio de posición . . . . .	71
A.14	Elección de primitiva . . . . .	72
A.15	Cambio de primitiva . . . . .	72
A.16	Niebla activada . . . . .	73
A.17	Niebla al acercarse a una imagen . . . . .	74
A.18	Elección del color . . . . .	74
A.19	Color cambiado . . . . .	75



# Chapter 1

## Introducción

### 1.1 Motivación

Este proyecto busca permitir al usuario visualizar las imágenes resultantes de una consulta a un sistema de recuperación de información de una manera atractiva, interactiva y amigable. Las distintas visualizaciones que han sido implementadas permiten el uso de varios descriptores simultáneamente lo que permite a investigadores comprobar el funcionamiento de sus descriptores de una manera más sencilla.

Debido al poco uso de entornos de tres dimensiones en los CBIR este proyecto persigue usarlo en cada una de las distintas visualizaciones disponibles. Como se ha mencionado antes, en cada consulta se pueden utilizar varios descriptores, estos se utilizarán para determinar que posición del espacio ocupa cada imagen.

Se busca interactividad por parte del usuario, por ello será capaz de moverse por el entorno con cierta libertad, lo que es esencial si el número de imágenes mostradas es muy elevado. A causa de la navegación por el entorno el usuario puede encontrarse perdido en algunos momentos, para evitar esto se añadirán mecanismos de control y ayuda.

Una vez que las imágenes estén dibujadas y que el usuario pueda moverse, este debe ser capaz de obtener información sobre cada imagen, si no las visualizaciones no servirían. El usuario simplemente posando el ratón encima de cada imagen obtendrá datos sobre ella, concretamente sobre el valor obtenido por el uso de cada descriptor.

Todo lo descrito se va a desarrollar para un CBIR concreto, Java Multimedia Retrieval©.

### 1.2 Estado del arte

El número de CBIR actuales es muy grande, ya que pueden ser usados en diversos ámbitos como la medicina, arquitectura, policial, militar... entre otros muchos. La

mayor parte de los CBIR se centran en como se recupera la información, especificando al detalle las técnicas y herramientas utilizadas, dejando la visualización de dicha información en un segundo plano, incluso algunos artículos omiten esa parte. Estos CBIR usan visualizaciones en dos dimensiones.

Para entender el punto del que parte este proyecto se van a describir varios CBIR junto con sus visualizaciones:

### 1.2.1 CBMIR

Este CBIR es usado en ámbitos médicos, de ahí su nombre Content-based medical image retrieval, ha sido desarrollado por la universidad nacional de Malasia. Se centra en ayudar a diagnosticar irregularidades vertebrales, lo cual es muy importante si se realiza a tiempo para disminuir el riesgo de sufrir fracturas vertebrales. Las fracturas aparecen en radiografías, pero su detección es poco frecuente por parte de médicos y radiólogos. Por esta razón, el objetivo de este CBIR es ayudar al personal sanitario a detectar estas irregularidades como se ha comentado anteriormente.

Una vez definido este CBIR, veamos como visualiza la información de una consulta.

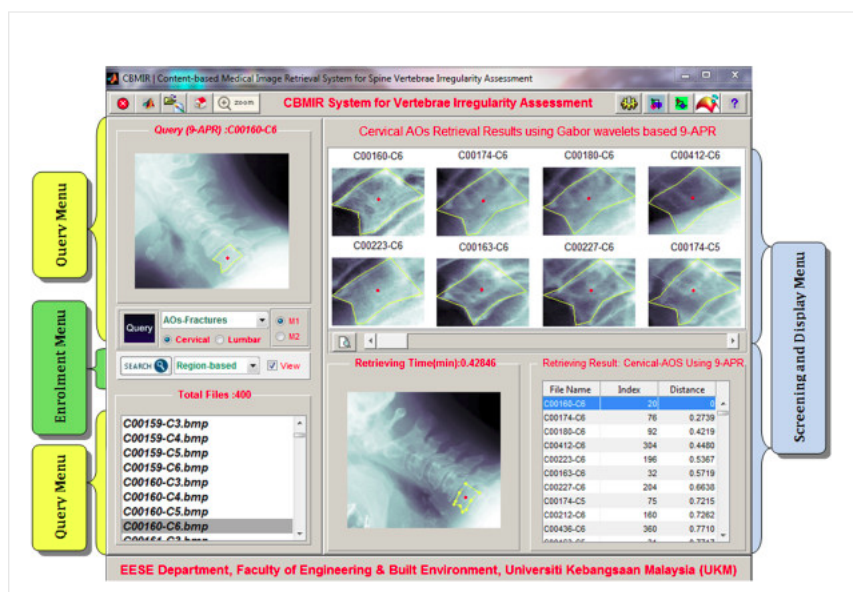


Figure 1.1: Salida consulta CBMIR

La visualización se realiza en la parte derecha de la pantalla. En la parte superior se aprecian todas las imágenes resultantes de la consulta, mientras que en la parte inferior

podemos obtener mas información sobre cada una de esas imágenes.

Mientras que para la consulta se utiliza la parte izquierda. En ella se puede seleccionar el descriptor y la imagen a usar.

### 1.2.2 GaZIR

GaZIR, Gaze-based Zooming Interface for Image Retrieval, es un sistema basado en la observación para la navegación y búsqueda de la imagen desarrollado por László Kozma, Arto Klami y Samuel Kaski.

La novedad fundamental es que la relevancia de la retroalimentación se infiere de las señales implícitas obtenidas en tiempo real desde el patrón de la mirada, el uso de un estimador aprendió durante una fase de entrenamiento por separado. La visualización se realiza disponiendo las imágenes en forma de anillos, en los cuales la retroalimentación obtenida en los anillos externos influye en las mostradas en los anillos interiores.

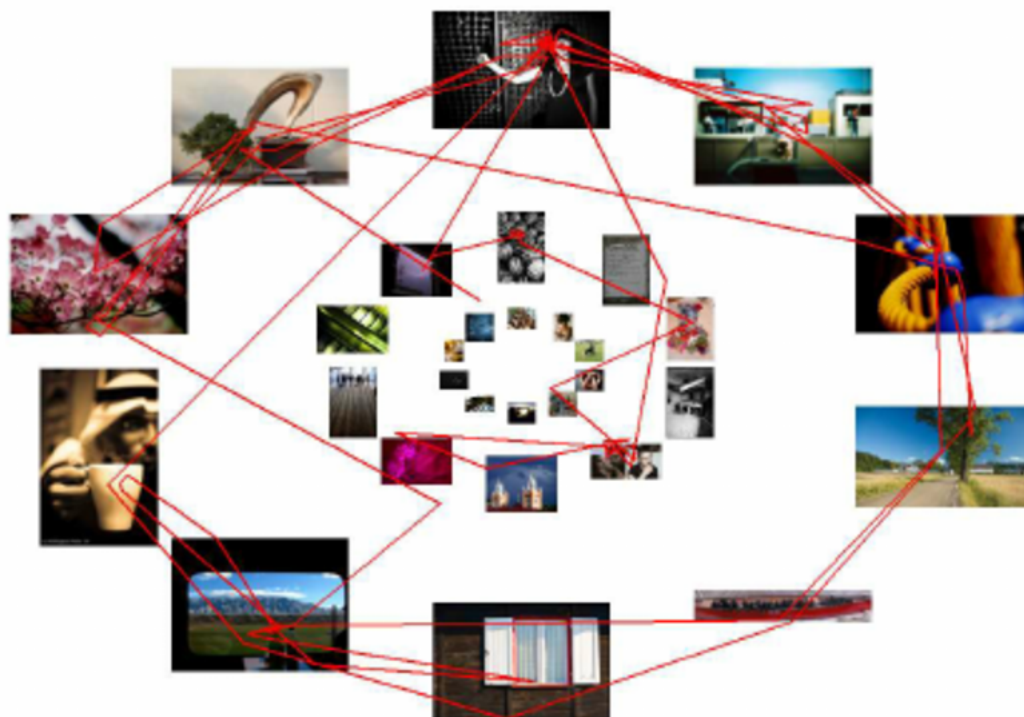


Figure 1.2: Salida consulta GaZIR

### 1.2.3 Similarity-Based Visualization of Large Image Collections

Este sistema trabaja con un número de imágenes mucho mayor que los descritos anteriormente. También cuenta con distintos tipos de visualizaciones, siendo uno de los que mas poseen. Ha sido desarrollado por Chaoli Wang, John P. Reese, Huan Zhang, Jun Tao, Yi Gu, Jun Ma, and Robert J. Nemiroff. Debido a que dispone de varios tipos de visualización ha sido de gran ayuda para este proyecto.

A continuación vamos a ver las distintas visualizaciones de este sistema:

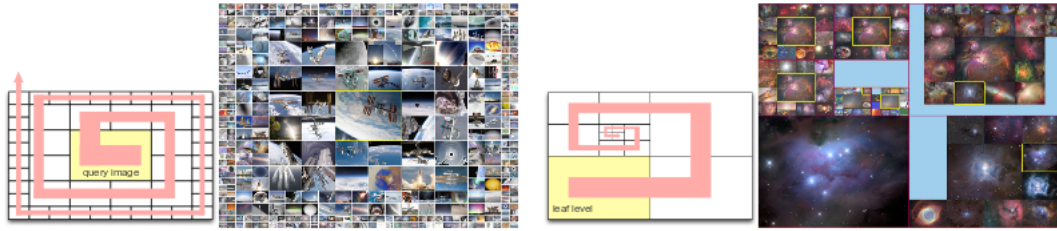


Figure 1.3: Visualización espiral SBVLIC

Estas visualizaciones colocan las imágenes en forma de espiral, siendo la central la imagen consulta. Las imágenes disminuyen de tamaño a la vez que se alejan, así se puede distinguir fácilmente que imágenes son mas similares a la elegida.

Veamos otro tipo:

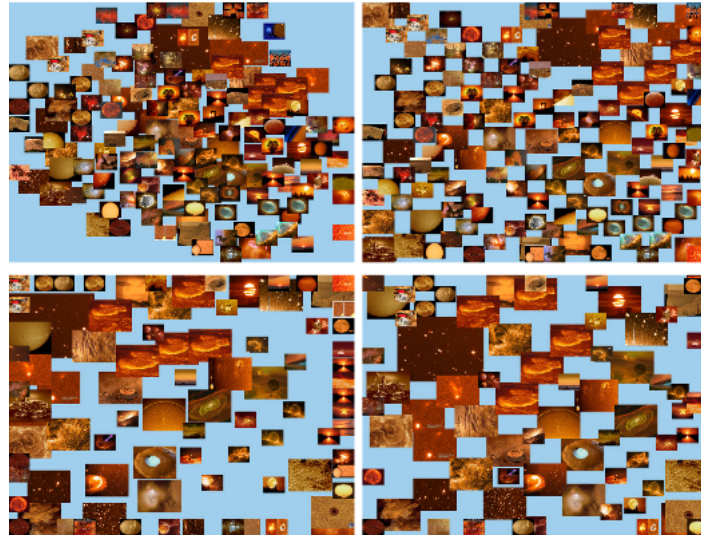


Figure 1.4: Visualización nube SBVLIC

En este tipo la imagen de mayor tamaño es la usada en la consulta. Como se puede apreciar algunas imágenes aparecen encima de otras lo que dificulta su visión y hay cierta incertidumbre a la hora de saber cual es la imagen original, se han propuesto soluciones a estos problemas en este proyecto.

### 1.2.4 Visual Clustering of Image Search Results

La visualización de ese sistema implementado por Trystan G. Upstill, Rajehndra Nagappan y Nick Craswell se basa a su vez en un modelo primavera desarrollado por Olsen y Korfhage. Esta técnica fue adaptada de RadViz. En este sistema, RadViz, los puntos de referencia se distribuyen en torno de un círculo, mientras que los elementos de datos están distribuidos en el círculo según su atracción a los puntos de referencia.

En esta imagen se puede apreciar lo explicado:

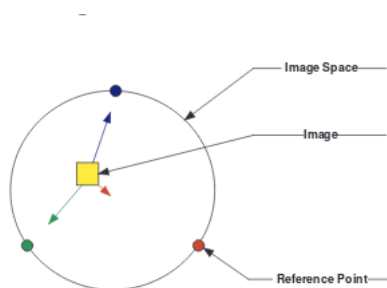


Figure 1.5: Puntos de atracción VCISR

Ahora veamos la representación visual de una salida producida por una consulta.

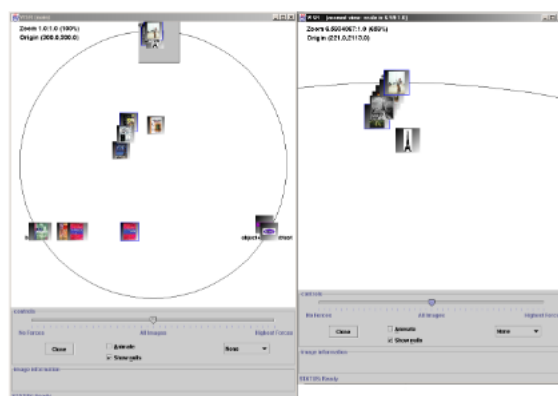


Figure 1.6: Salida VCISR

Podemos apreciar como se distribuyen las imágenes a lo largo del círculo siguiendo el patrón comentado anteriormente.

Por otro lado podemos ver lo que ocurre al hacer zoom sobre una región. Al hacerlo apreciamos nuevas imágenes que permanecían ocultas.

Este tipo de distribución ha servido de inspiración para varias visualizaciones implementadas en este proyecto.

## 1.3 Objetivos

Una vez que sabemos el estado del arte y la motivación de este trabajo, es hora de esclarecer los objetivos a cumplir.

Los objetivos van a ser clasificados en dos subgrupos: principales y secundarios.

### 1.3.1 Objetivos principales

- Desarrollar una serie de módulos que permitan la visualización de imágenes en un entorno 3D. Actualmente la mayoría de los CBIR utilizan visualizaciones bidimensionales por lo que el uso de entornos tridimensionales puede ser considerado como una gran novedad respecto a los demás.
- Estos módulos deben de resultar eficaces para el usuario que los use. En todo momento debe saber la información que se encuentra mostrada y ser capaz de moverse por el entorno con el fin de poder apreciar las imágenes correctamente.

## Chapter 2

# Especificacion de requisitos

Por requisitos podemos entender: Condición o capacidad que necesita el usuario para resolver un problema o conseguir un objetivo determinado.

Como en todo proyecto de software, los requisitos deben ser especificados antes de empezar el desarrollo del mismo.

En esta sección vamos a detallar cada uno de los distintos requisitos necesitados agrupados en requisitos funcionales, no funcionales y de información.

### 2.1 Requisitos funcionales

Los requisitos funcionales se encargan de especificar cómo se realizará la interacción entre el sistema y su entorno, indicando los servicios que ha de tener el sistema o cómo responderá ante ciertos estímulos.

Este proyecto cuenta con los siguientes requisitos funcionales:

**RF-1. Imágenes:** Gestión de imágenes.

RF-1.1. Se debe permitir cargar cualquier imagen.

RF-1.2. Se debe permitir cargar cualquier número de imágenes.

RF-1.3. La elección de la imagen consulta ha de ser sencilla.

**RF-2. Descriptores:** Gestión de descriptores.

RF-2.1. Ha de ser posible la elección de cualquier descriptor disponible.

RF-2.2. Se debe permitir elegir más de un descriptor al mismo tiempo.

RF-2.3. La consulta ha de realizarse con los descriptores seleccionados.

**RF-3. Visualizaciones:** Gestión de visualizaciones.

RF-3.1. El usuario debe poder elegir cualquier visualización activada.

RF-3.2. El usuario debe obtener información de las imágenes mostradas mediante el uso del ratón.

**RF-4 Interacción** Gestión de la interacción.

RF-4.1. El usuario debe poder interactuar con teclado y/o ratón.

RF-4.2. El usuario debe ser capaz de rotar la cámara en los ejes x e y

RF-4.3. El usuario debe ser capaz de desplazarse a lo largo del entorno.

RF-4.4. El usuario debe poder cambiar elementos del entorno para adaptar la visualización.

RF-4.5. El usuario debe ser capaz de volver a la posición inicial de la visualización.

## 2.2 Requisitos no funcionales

Los requisitos no funcionales describen cualidades o restricciones del sistema que no se relacionan de forma directa con el comportamiento funcional del mismo.

Este proyecto cuenta con los siguientes requisitos no funcionales:

**RNF-1** Debe de utilizar la mínima cantidad de memoria posible.

**RNF-2** Ha de ser implementando en Java usando Java 3D.

**RNF-3** Las visualizaciones deben ser lo suficientemente flexibles para que puedan ser adaptadas a sistemas propios.

**RNF-4** Solo se puede elegir una visualización al mismo tiempo.

**RNF-5** Las visualizaciones deben activarse/desactivarse atendiendo al número de descriptores que necesiten.

**RNF-6** El tamaño de las imágenes mostradas ha de ser lo suficientemente grande para poder verse correctamente.

**RNF-7** Se debe indicar el orden en el que fueron dibujadas las imágenes.

## 2.3 Requisitos de información

Este tipo de requisito de detallar la necesidad por parte del sistema del almacenamiento de la información.

**RI-1** Almacenar información de la consulta para evitar nuevas consultas innecesarias.



# Chapter 3

## Planificación

Después de aclarar los objetivos del proyecto y requisitos del mismo, es tiempo de realizar la planificación.

Para realizar la planificación del proyecto disponemos de múltiples alternativas, pero en este caso la técnica a usar son los diagramas de Gantt. Un diagrama de Gantt puede ser considerado una herramienta visual en la cual se ve reflejado el tiempo que va a ser empleado en cada una de las tareas del proyecto.

La planificación de este proyecto ha sido dividida en varios bloques que serán comentados a continuación:

### 3.1 Primer acercamiento

Este bloque ha sido dedicado al estudio de las distintas alternativas existentes para la representación visual de imágenes para disponer de un buen punto de partida para la realización de los módulos de visualización propios. Las alternativas consideradas más interesantes han sido explicadas en el primer capítulo.

También se ha estudiado la tecnología a usar, en este caso Java 3D ya que ha sido la primera vez que he tratado con ella, aunque ya contaba con experiencia previa en Java.

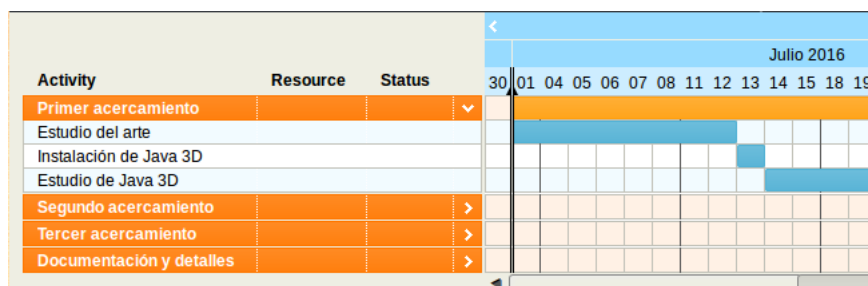


Figure 3.1: Diagrama Gantt primer acercamiento

El proyecto comenzó el 1 de julio. Los primeros 12 días fueron empeñados en el estudio del arte, punto fundamental para empezar el proyecto. El resto de días se usaron para aprender a trabajar con Java 3D y a su correcta instalación cosa que puede complicarse si no se tiene cuidado.

## 3.2 Segundo acercamiento

Una vez establecido el punto de partida e instalado todo lo necesario pasamos al segundo bloque.

En este bloque se realizó un análisis para establecer cuáles eran los requisitos necesarios para el correcto funcionamiento del proyecto. Por otro lado, se decidieron los distintos módulos de visualización a implementar, siendo las siguientes:

- Secuencial
- Espiral
- Camino
- N-caminos
- Cartesiana 2D
- Cartesiana 3D
- Polar 2D
- Polar 3D

Cada una de ellas será explicada más adelante.

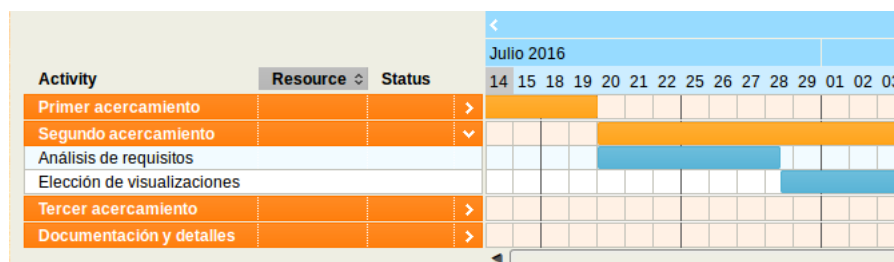


Figure 3.2: Diagrama Gantt segundo acercamiento

En este bloque el tiempo ha sido repartido de manera casi equitativa entre el análisis de requisitos y la elección de las visualizaciones a implementar. Cosa crucial para la correcta realización del proyecto, ya que, aparte de decidir las visualizaciones, todos los requisitos deben estar establecidos antes de avanzar en el proyecto.

### 3.3 Tercer acercamiento

Este es el bloque al cual se le ha dedicado más tiempo en este proyecto, debido a que es en el que se ha procedido a la implementación de todo lo que se ha ido comentando en los dos bloques anteriores.

Como se ha mencionado a lo largo de la memoria, los distintos módulos van a ser desarrollados para Java Multimedia Retrieval©, por lo que se empleó un tiempo en obtener la información necesaria de las consultas para garantizar la correcta actividad de las diferentes visualizaciones. Es decir, en *unir* JMR con los módulos.

Tras esto, se diseñó una estructura de clases para dar soporte a las visualizaciones actuales y a las futuras. Este soporte se encarga de instanciar el mundo 3D, preparar los diferentes eventos y gestionar la interacción con el usuario. De esta forma los distintos módulos de visualización solo se deben preocupar de como colocar las distintas imágenes en el entorno.

Finalmente, después de todo lo anterior se comenzó a implementar cada una de las distintas visualizaciones. Algunas implementaciones fueron simples, pero otras debido a la información que se manejaba fueron más complejas.

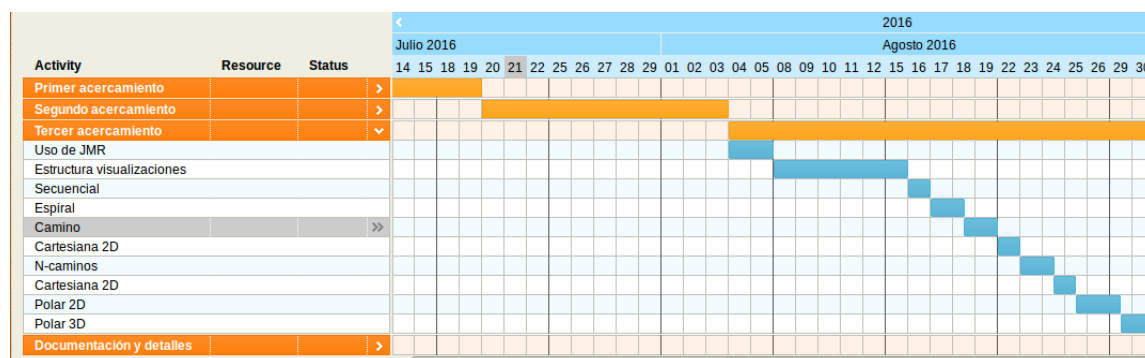


Figure 3.3: Diagrama Gantt tercer acercamiento

### 3.4 Documentación y detalles

En este último bloque se ha prodecido a la realización de esta memoria y a su vez a la revisión de todo el proyecto en busca de erratas y malfuncionamiento.

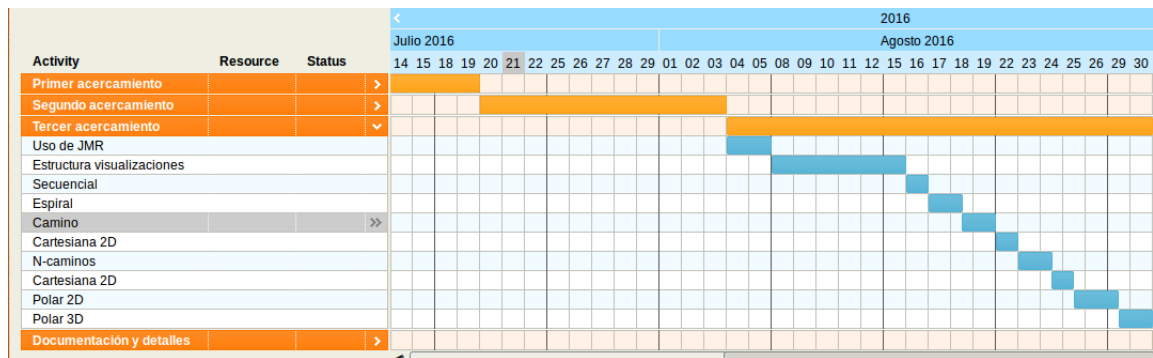


Figure 3.4: Diagrama Gantt documentación y detalles

Tras terminar todo el proyecto es muy importante revisarlo minuciosamente con el objetivo de encontrar fallos o elementos que se encuentren mal explicados. Por eso este bloque es gran importancia a pesar de ser el último.

## Chapter 4

# Análisis

### 4.1 Descripción del problema

Nos encontramos ante la siguiente situación: Debemos representar imágenes en un mundo tridimensional usando la información obtenida de los distintos descriptores. Dicha información proporcionará el punto en el que se dibujarán las imágenes.

Dependiendo del número de descriptores podemos hablar de visualizaciones en 1D, un descriptor, 2D, dos descriptores, 3D, tres descriptores. Esto es así ya que cada descriptor indica una coordenada en el espacio tridimensional.

Los valores proporcionados por cada descriptor se encuentran en el rango 0-1. Los cálculos para determinar la posición se realizarán con esos datos, pero a la hora de dibujar las imágenes esos datos serán multiplicados de manera proporcional para evitar que las figuras se muestren en un rango de puntos muy pequeño, lo que supondría el malfuncionamiento de la visualización ya que no se podrían apreciar correctamente las imágenes.

Dado el número de imágenes y la posición de estas, no todas podrán ser vistas a la vez ni cuando se inicie la visualización, por lo que el usuario debe ser capaz de desplazarse por el escenario y debe contar con guías que le indiquen el orden de las diferentes imágenes mostradas.

También deberá ser posible obtener información de las imágenes mostradas con el fin de comprender mejor la visualización y su vez comprobar el correcto funcionamiento de los descriptores.

### 4.2 Soluciones propuestas

Como solución a este problema se han diseñado una serie de visualizaciones que cumplen lo anteriormente descrito. También cuentan con un menú desplegable con opciones como reiniciar la vista, cambiar la forma en la que se pintan las imágenes, cubos o esferas, activar/desactivar la niebla así como su color. También es posible cambiar el color del fondo del entorno, por si coincide con el color de las imágenes.

Existe una clase abstracta de las que heredan las demás visualizaciones. Esta clase se encarga de inicializar el mundo 3D, controlar los eventos y la interacción, con el fin de que las clases que representan las visualizaciones solo deban dibujar las imágenes y elegir el tipo de interacción que desean.

A continuación vamos a hablar de cada visualización con más detalle.

#### 4.2.1 Visualización secuencial

Esta visualización corresponde al tipo secuencial, es decir, la imagen original el centro y las demás a su derecha ordenadas según su parentesco con la original. Aunque en este caso el orden que se sigue es diferente. Se trata de una visualización 1D porque solo utiliza un descriptor.

- La imagen original se posiciona en el centro.
- Se eligen las dos imágenes más similares a la original. La más similar se coloca a su izquierda, la restante a su derecha, tras esto se vuelven a coger las siguientes dos imágenes más parecidas a la usada en la consulta y se repite el proceso hasta usar todas las imágenes.

La interacción se realiza mediante teclado.

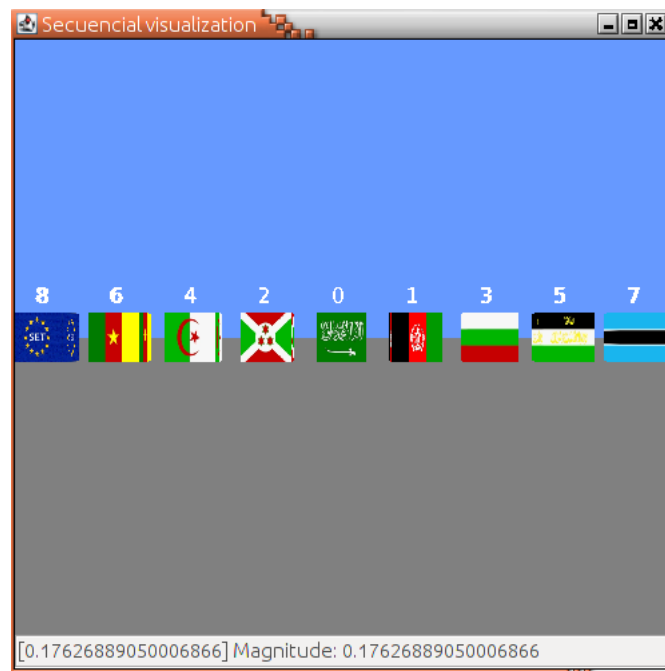


Figure 4.1: Visualización secuencial

### 4.2.2 Visualización espiral

En esta visualización la imagen original se coloca en el centro, las demás se colocan a su alrededor trazando una espiral en sentido antihorario. A medida de que se alejan, su tamaño disminuye, lo que ayuda a apreciar esta visualización con mayor detalle. Se trata de una visualización 1D porque solo utiliza un descriptor.

La interacción se realiza mediante ratón.

El algoritmo que dibuja la espiral se comenta a continuación.

- Es un bucle que se divide en 4 subbucles. Todos ellos disminuyen el tamaño de las imágenes en función del número total de estas.
- El primer subbucle dibuja la imagen original en el centro de coordenadas. Tras esto, pinta las siguientes  $n$  imágenes, 3 la primera vez, en la derecha de la imagen original. Cuando todos los subbucles finalizan el valor de  $n$  en este aumenta 1 la primera vez, aumentando 2 el resto de las ocasiones. Su valor  $n$  inicial es de 3.
- El segundo subbucle pinta  $n$  imágenes debajo de la última imagen mostrada por el primer subbucle. En este caso cuando todos los subbucles terminan el valor de  $n$  en este aumenta en 2. Su valor de  $n$  inicial es de 1.
- En el tercer subbucle se dibujan  $n$  imágenes a la izquierda de la posición de la última imagen mostrada por el segundo subbucle. Cuando todos los subbucles terminan el valor de  $n$  en este aumenta en 2. Su valor de  $n$  inicial es de 3.
- En el último subbucle se muestran  $n$  imágenes encima de la posición de la última imagen dibujada por el tercer bucle. Cuando todos los subbucles terminan el valor de  $n$  en este aumenta en 2. Su valor de  $n$  inicial es de 2.

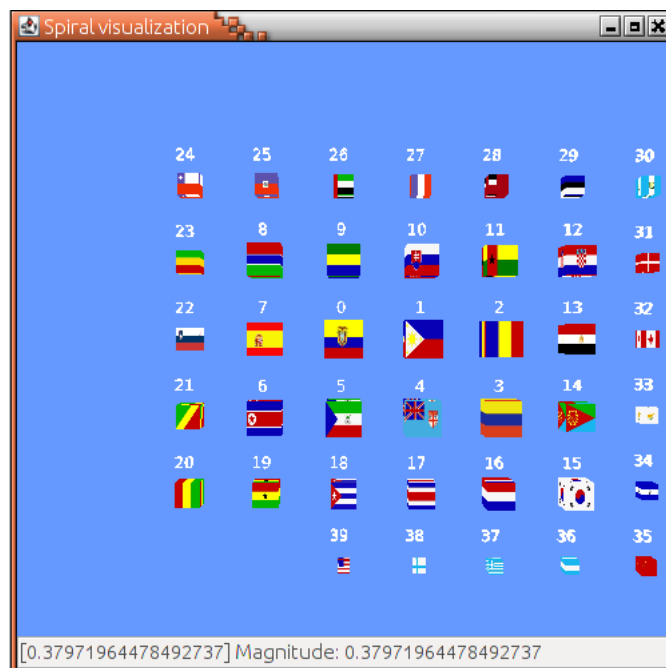


Figure 4.2: Visualización espiral

### 4.2.3 Visualización camino

En esta ocasión la imagen original se encuentra en el centro de coordenadas, pero el resto de imágenes se encuentran detrás de ella, simulando un camino, que al recorrerlo, se consigue ver las imágenes mas parecidas a la imagen consulta. Como las anteriores se trata de una visualización 1 D ya que solo utiliza un descriptor. La distancia entre la posición de las imágenes viene determinada por el descriptor.

La interacción se realiza mediante teclado.

En la siguiente figura podemos apreciar la imagen consulta e intuir gracias a los números indicadores de posición, las demás imágenes.



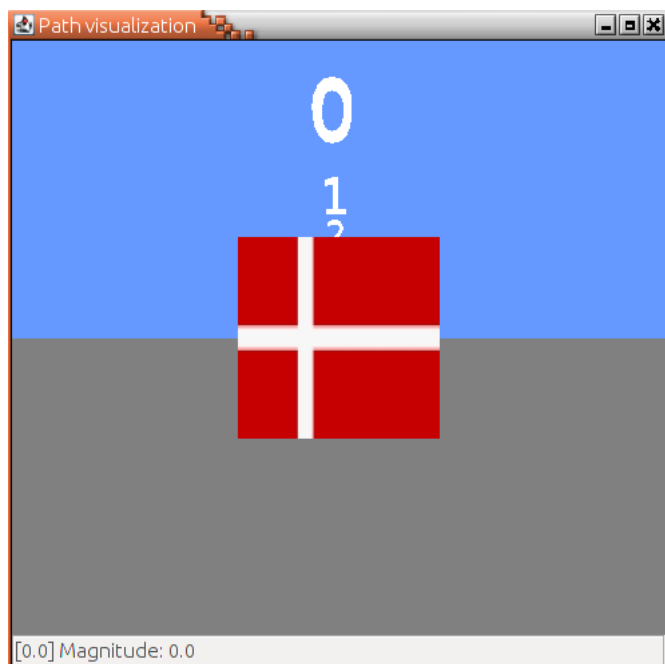


Figure 4.3: Visualización camino 1

Este sería el resultado si avanzásemos

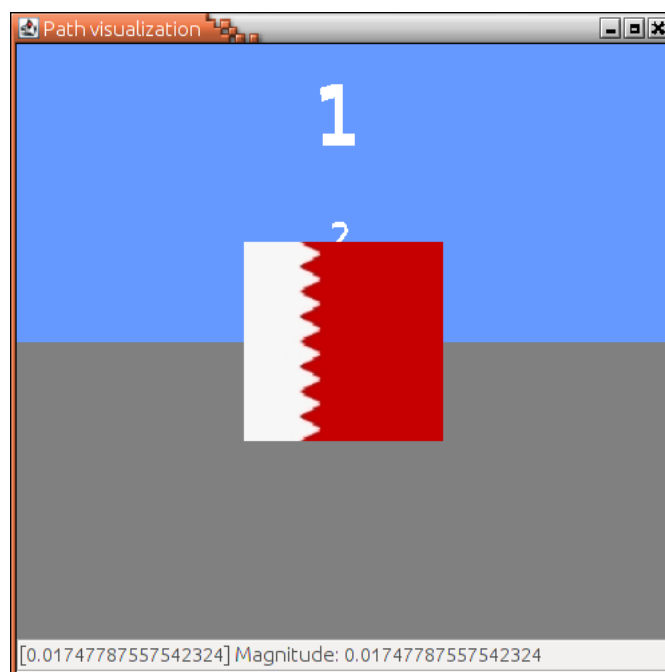


Figure 4.4: Visualización camino 2

#### 4.2.4 Visualización cartesiana 2D

A diferencia de las anteriores, esta visualización puede ser considerada como una visualización 2D ya que utiliza dos descriptores para calcular el punto en el espacio de cada imagen. Se usan las coordenadas cartesianas para posicionar la imagen, siendo el eje  $x$  utilizado por el primer descriptor, mientras que el eje  $y$  es utilizado por el segundo descriptor.

El resultado es la imagen original en el centro, y el resto a su derecha, siendo las más similares las más cercanas tanto en el eje  $x$  e  $y$ , mientras que las más alejadas en ambos ejes serán las menos similares.

La interacción se realiza mediante teclado.

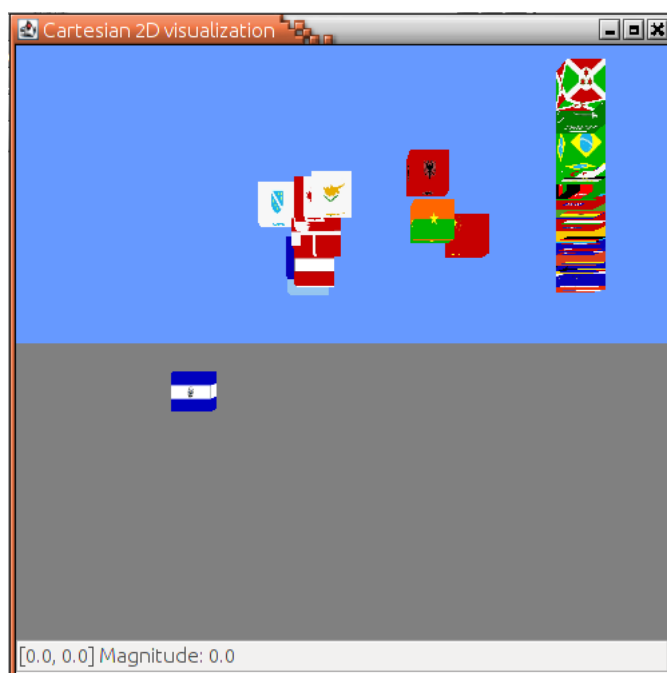


Figure 4.5: Visualización cartesiana 2D

#### 4.2.5 Visualización polar 2D

Se trata de otra visualización 2D. En este caso utilizamos las coordenadas polares para establecer el punto en el que se pintará la imagen. Como sabemos la posición de los puntos usando coordenadas polares viene dada por una distancia y un ángulo. Debemos obtener los valores comentados utilizando la información aportada por los descriptores. Por lo que el cálculo del ángulo quedaría de la siguiente forma:

$$\text{ángulo} = \text{atan}(x/y)$$

donde  $x$ ,  $y$ , son valores aportados por los descriptores en coordenadas cartesianas, mientras que  $\text{atan}$  representa el arcotangente.

El valor de la distancia se calcula así:

$$distancia = \sqrt{x^2 * y^2}$$

donde  $x$ ,  $y$ , son valores aportados por los descriptores.

La interacción se realiza mediante teclado.



Figure 4.6: Visualización polar 2D

#### 4.2.6 Visualización cartesiana 3D

Primera de las dos las visualizaciones 3D implementadas. Utiliza 3 descriptores, cada uno de ellos aporta la posición en los ejes  $x$ ,  $y$  e  $z$  para cada imagen.

El resultado sería que las imágenes menos similares se encontrarían mas alejadas también en el eje  $z$ , a diferencia de la visualización cartesiana 2D ya que esta es solo 2D.

La interacción se realiza mediante teclado.

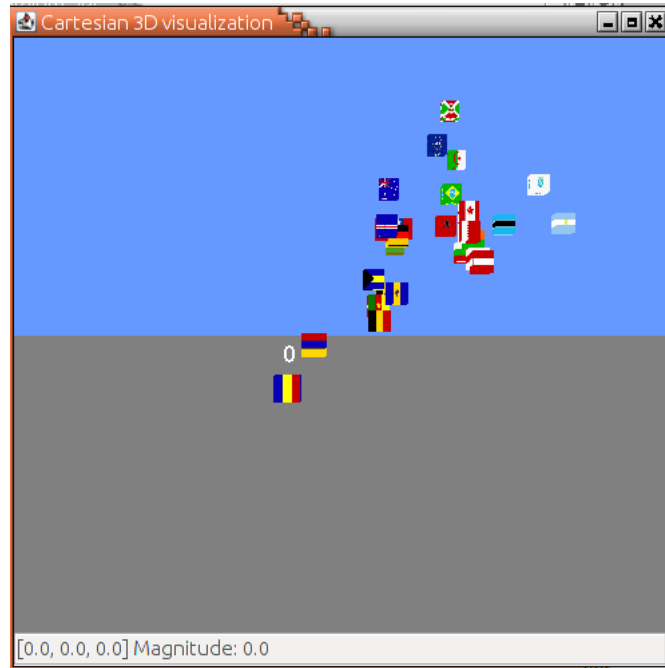


Figure 4.7: Visualización cartesiana 3D

#### 4.2.7 Visualización polar 3D

Aunque se llame polar 3D se utilizan las coordenadas esféricas. Estas vienen dadas por un radio, el ángulo polar o colatitud y el azimut. Como en el caso 2D y a diferencia de las visualizaciones cartesianas, debemos calcular los valores de cada una de las variables descritas.

La ecuación del radio es de la forma

$$radio = \sqrt{x^2 * y^2 * z^2}$$

donde  $x$ ,  $y$ ,  $z$  son valores aportados por los descriptores en coordenadas cartesianas

La ecuación del ángulo se puede representar así

$$\text{ángulo} = \text{acos}(z/radio)$$

donde  $z$  es un valor aportado por un descriptor,  $radio$  es el radio calculado en la ecuación anterior y  $\text{acos}$  representa al arcoseno.

Por último, la ecuación de azimut es de la forma

$$\text{azimut} = \text{atan}(y/x)$$

donde  $x$  e  $y$  son valores aportados por dos descriptores, mientras que  $\text{atan}$  representa al arcotangente.

La interacción se realiza mediante teclado.



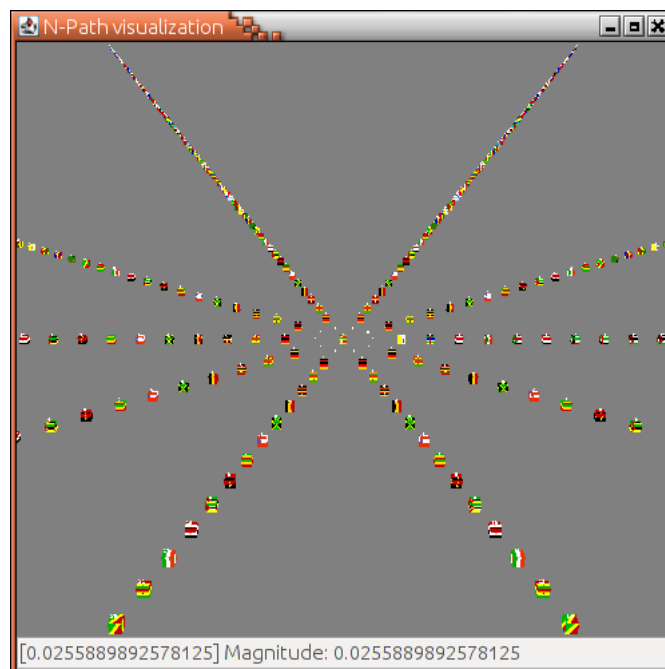
Figure 4.8: Visualización polar 3D

#### 4.2.8 Visualización n caminos

Se trata de una visualización especial, ya que representa  $n$  descriptores utilizando una visualización 1D como es la visualización camino. Esta visualización dispone de un círculo cuyo radio es proporcional al número de descriptores, y alrededor de este se disponen las imágenes habiendo un camino por cada descriptor seleccionado. De esta manera es posible comprobar los descriptores que se quieran a la vez, ya que la distribución varía según los descriptores para adaptarse a ellos. La interacción se realiza mediante teclado. La JMR actualmente cuenta únicamente con pocos descriptores, por lo que en la siguiente figura se encuentran duplicados para simular un mayor número de descriptores con el fin de apreciar correctamente esta visualización.

Para identificar cada descriptor se coloca un número, siendo el 1 el primer descriptor seleccionado. También se colocan números a modo de guía en las imágenes.

Como se ha comentando anteriormente, esta visualización tiene un gran potencial ya que se adapta a cualquier número de descriptores. A continuación veamos un ejemplo para 10 descriptores.

Figure 4.9: Visualización n-caminos  $n=6$ Figure 4.10: Visualización n-caminos  $n=10$

### 4.3 Casos de uso

Los casos de uso podemos entenderlos como una explicación de las actividades que han de ser desarrolladas para llevar a cabo un proceso. En nuestro ámbito, ingeniería del software, un caso de uso es una serie de acciones que se producen entre los actores de un sistema y el mismo, en resultado a un evento iniciado por el actor principal del sistema. Un actor es una entidad externa del sistema que posee un vínculo con él y a su vez, le exige una funcionalidad. Una vez contextualizados vamos a incorporar los diagramas correspondientes a los casos de uso.

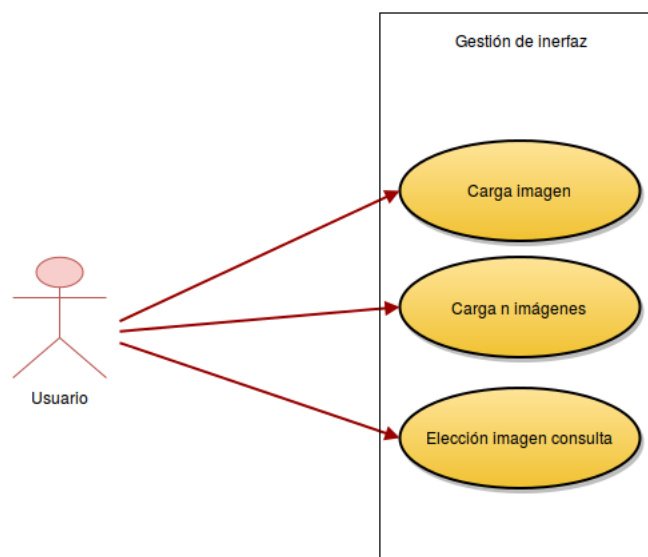


Figure 4.11: Caso de uso gestión de imágenes

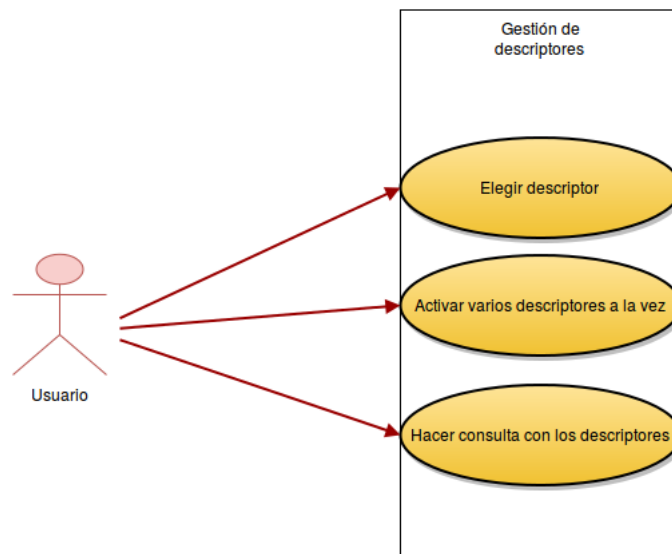


Figure 4.12: Caso de uso gestión de descriptores

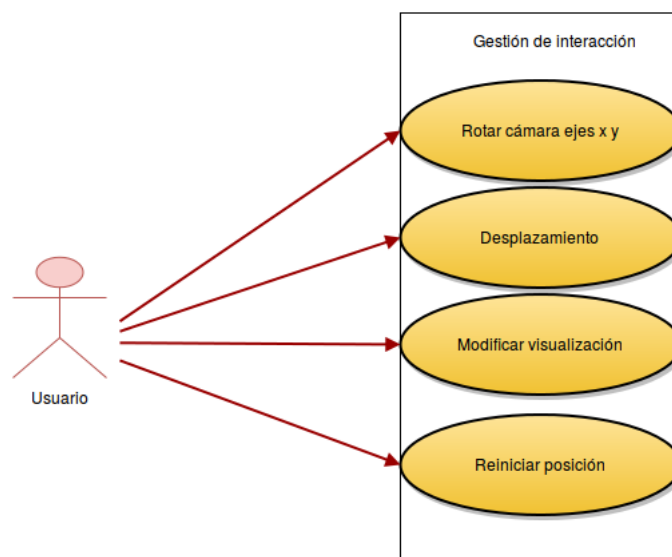


Figure 4.13: Caso de uso gestión de interacción



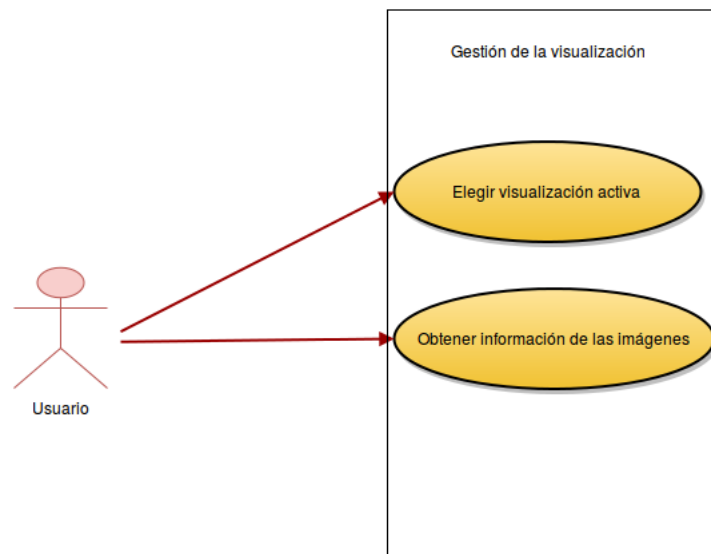


Figure 4.14: Caso de uso gestión de visualización

## 4.4 Diagramas de secuencia

Establecidos los casos de uso, procedamos con los diagramas de secuencia. Estos se usan para establecer el modo en que interaccionan objetos en un sistema atendiendo a UML.

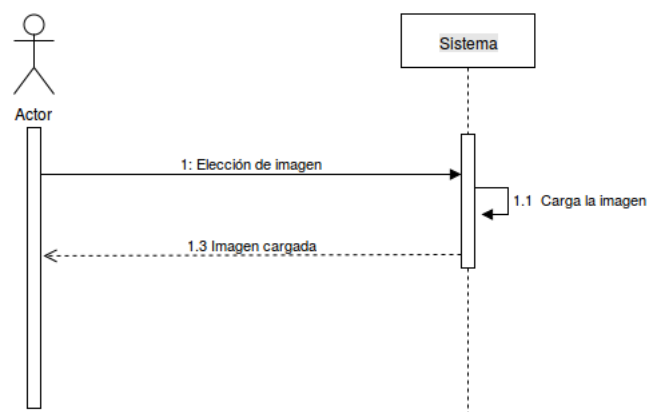


Figure 4.15: Diagrama de secuencia carga de imagen

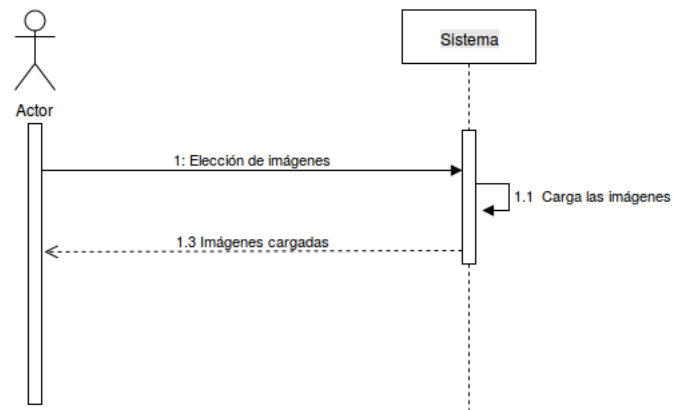


Figure 4.16: Diagrama de secuencia carga de imágenes

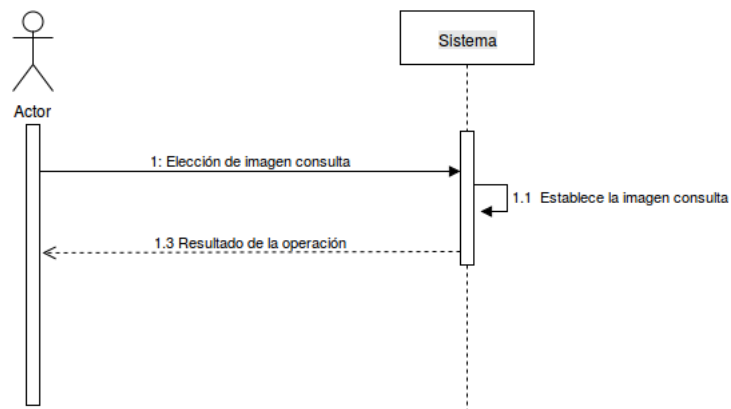


Figure 4.17: Diagrama de secuencia elección imagen consulta

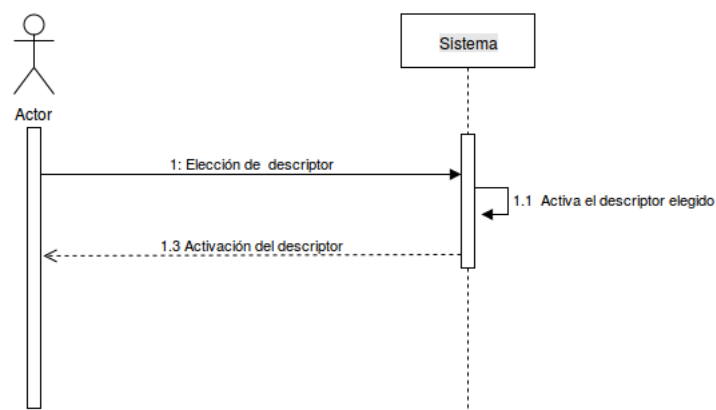


Figure 4.18: Diagrama de secuencia elección descriptor

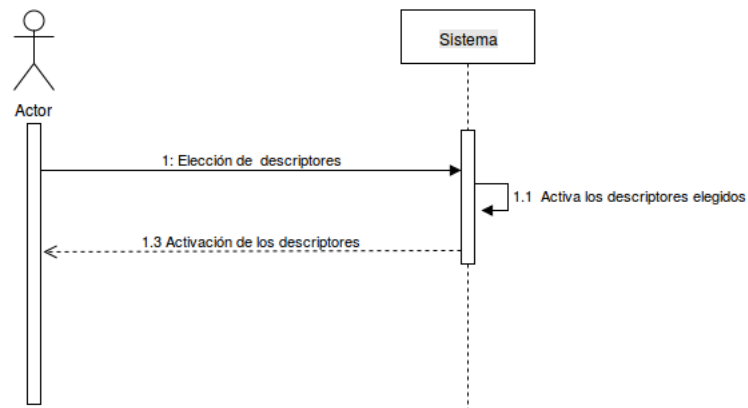


Figure 4.19: Diagrama de secuencia elección de descriptores

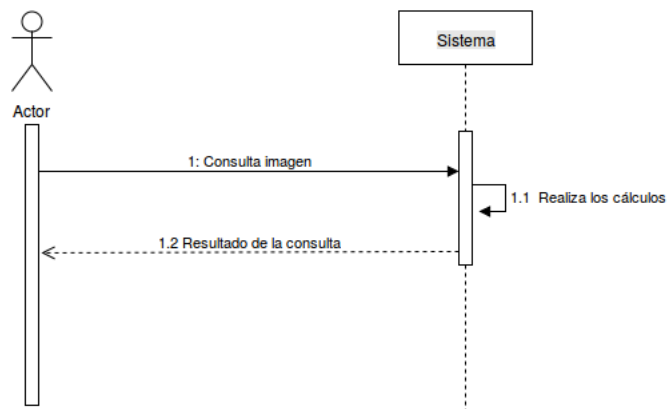


Figure 4.20: Diagrama de secuencia elección de consulta

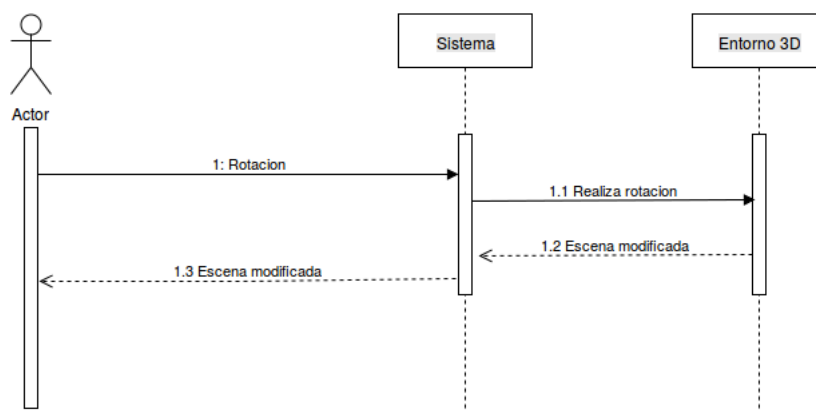


Figure 4.21: Diagrama de secuencia rotación escena

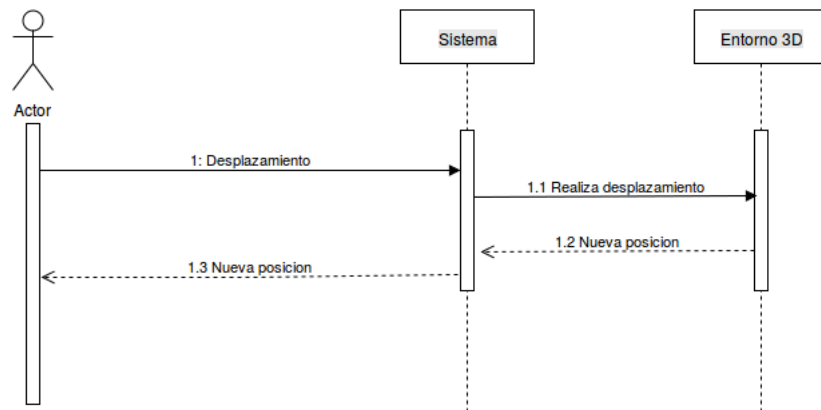


Figure 4.22: Diagrama de secuencia desplazamiento

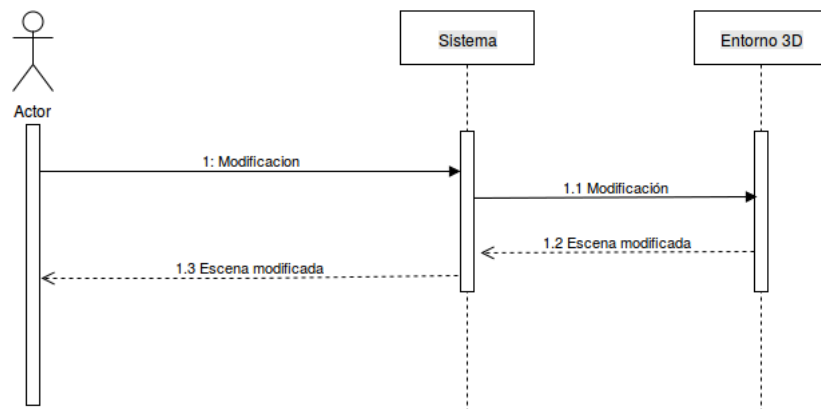


Figure 4.23: Diagrama de secuencia modificación escena

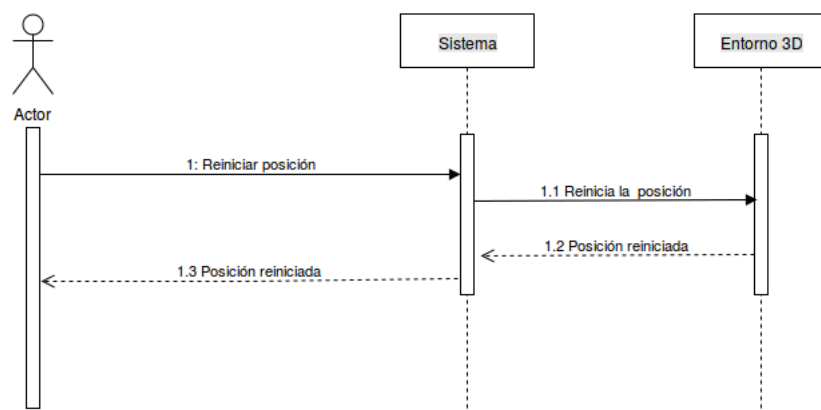


Figure 4.24: Diagrama de secuencia reinicio de posición

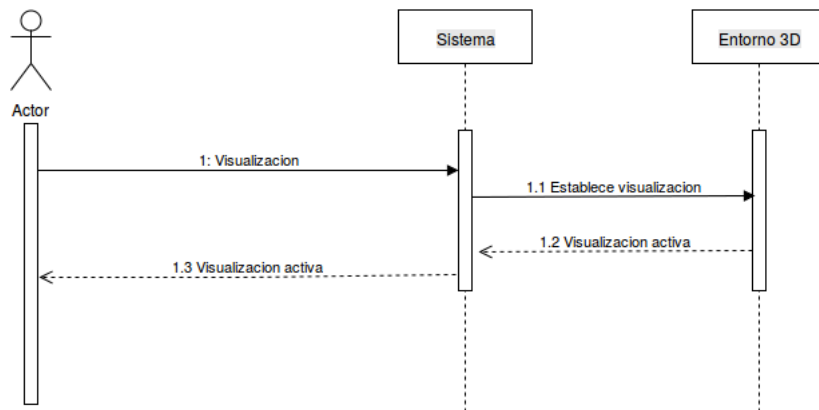


Figure 4.25: Diagrama de secuencia elección visualización

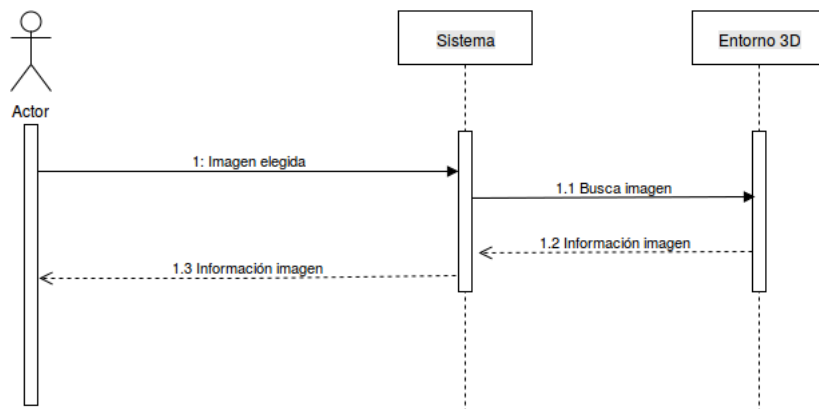


Figure 4.26: Diagrama de secuencia obtención de información de la imagen



## Chapter 5

# Diseño

### 5.1 Introducción al diseño

En este capítulo se va a hablar sobre el diseño del proyecto, así como de la arquitectura y herramienta elegida para el desarrollo del mismo.

El diseño es el desarrollo de aplicar una serie de métodos, técnicas y principios de diseño para traducir el modelo del análisis a una representación la cual sea posible ser codificada.

El método de diseño elegido para este proyecto ha sido el orientado a objetos. Debido a que cada clase representa a un módulo de visualización, lo que como es natural, es muy modular.

### 5.2 Arquitectura

El estilo arquitectónico utilizado para la realización de este proyecto ha sido el modelo vista controlador, ya que se adapta correctamente a las necesidades del sistema, MVC. Los elementos de una arquitectura MVC son:

- Modelo: Es el encargado del conocimiento del dominio de la aplicación.
- Vista: Responsable de mostrar las diferentes características del modelo al usuario.
- Controlador: Elemento que responde a la interacción del usuario, realizando las peticiones necesarias al modelo y la vista.

Como se puede intuir, este proyecto se centra en la vista, es decir, en el desarrollo de las distintas visualizaciones. Dejando en un segundo plano al modelo, que se trabajará en trabajos futuros, y al controlador.

Para lo demás se cuenta con una aplicación que sirve como controlador, permitiendo la carga de imágenes, su consulta mediante descriptores y la visualización de los resultados utilizando los módulos de visualización implementados en este proyecto. Como esta aplicación no es del ámbito principal del proyecto, se va a obviar durante el diseño.

Una vez definino el estilo arquitectónico y tras barajar diversas opciones, la herramienta seleccionada para desarrollar el proyecto fue Netbeans

## 5.3 Netbeans

Se trata de un entorno de desarrollo, IDE, libre utilizado mayormente para Java, aunque puede ser usado para otros lenguajes con C++. Una de las principales características de Netbeans es su modularidad, es decir, las aplicaciones que han sido implementadas usando este IDE, son fácilmente extensibles por otros desarrolladores.

Permite la elaboración de la documentación del proyecto mediante el uso de Javadoc. También ayuda al diseño de las interfaces, permitiendo al usuario elegir el elemento y su posición dentro de la ventana de manera gráfica.

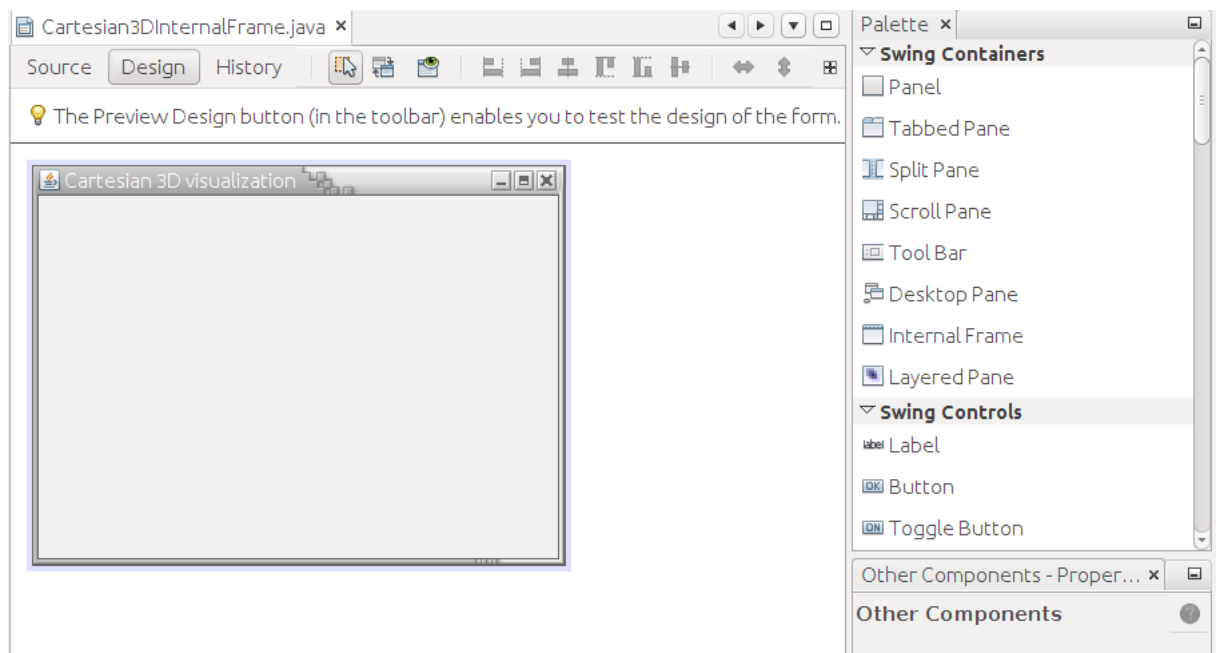


Figure 5.1: Desarrollo de interfaces usando Netbeans

Si bien a primera vista parece que será de gran ayuda en este proyecto esto no es así. Las biblioteca gráfica actual de Java se denomina Swing y podemos entender sus



elementos como ligeros. El elemento de visualización de Java 3D se denomina Canvas 3D y se considera pesado por lo que no se puede utilizar la ayuda proporcionada por Netbeans para crear un elemento de la Swing que contenga un Canvas 3D, de hecho si se intenta, Netbeans se cerrará inmediatamente debido a un error producido por lo comentado.

Como alternativa, si se describen las interfaces a mano, sin la ayuda mostrada en la figura anterior, se podrán combinar ambos elementos para lograr, por ejemplo, un `jpanel` en el que se visualice un entorno 3D. A pesar de esto los problemas persisten.

Un ejemplo de uno de los problemas que persisten es que al redimensionar una ventana de Java 3D el contenido de esta puede desaparecer, debajo un fondo gris, sin nada. Este problema viene relacionado a la tarjeta gráfica del sistema. Por ejemplo, se ha comprobado que este problema es mas evidente en un sistema Windows con una tarjeta gráfica nvidia GTX que en un sistema Ubuntu usando la gráfica integrada.

Otro problema está relacionado con los menús desplegables. Estos menús cuando se utilizan a la vez que un Canvas3D aparecen por detrás del propio canvas, haciendo imposible la visualización y el uso del menú desplegable.

La solución a estos problemas ha sido descrita en el apartado anterior.

## 5.4 Java 3D

Como se ha comentado, la tecnología principal del proyecto es Java 3D. Su estructura es de tipo grafo, cada elemento puede ser tratado como un nodo lo que ayuda a que sea flexible y modulable, ya que se pueden añadir nodos con un determinado propósito y eliminarlos si no se desean sin poner en riesgo el resto de la estructura. En este proyecto se explota dicha posibilidad, ya que se permite al usuario una mayor interacción, y por otro lado, facilita la labor a los desarrolladores a la hora de entender y extender las visualizaciones creadas.

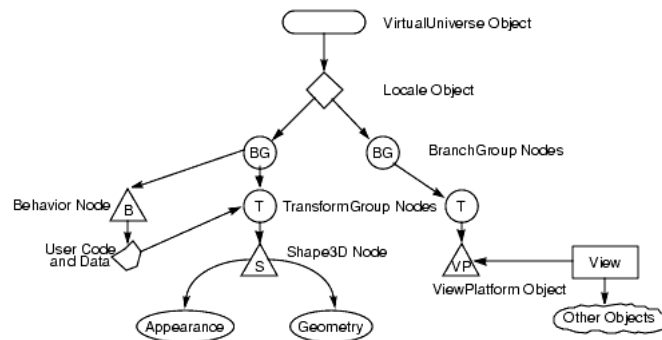


Figure 5.2: Estructura Java 3D

Como puede observarse en la figura, cada nodo representa algo. La parte derecha representa a la vista del usuario dentro del entorno tridimensional, mientras que la izquierda define una figura y el comportamiento que va a poseer

## 5.5 Diagrama de clases del diseño

Tras todo lo anterior es hora de definir el diagrama de clases de diseño de este proyecto. Esto describe de manera gráfica las características de las clases e interfaces software así de las relaciones entre ellas dentro de una aplicación.

La herramienta utilizada para el la creación del diagrama ha sido easyUML, un plugin de Netbeans.





## Chapter 6

# Implementación

### 6.1 Descripción de la implementación

La implementación del proyecto ha sido realizada en Java, concretamente utilizando Java 3D.

El elemento utilizado por Java 3D se llama Canvas 3D. Es un elemento pesado, por lo que al combinarlo con elementos ligeros como los jpanel, paneles de visualización de Java, surgen problemas. Estos problemas han sido solucionados en este proyecto.

Cada clase es un jpanel, de tal manera que cada una de ellas, a excepción de la primera, representa una visualización de las descritas en el proyecto.

Este proyecto es amplio, por lo que se comentarán los elementos más relevantes de las distintas clases implementadas.

### 6.2 Abstract3DPanel

Es una clase abstracta que puede considerarse la clase padre, de ella heredan todas las demás, esto puede verse en el diagrama de clases del capítulo anterior. Se trata de una clase extensa, ya que se encarga de la creación del mundo 3D. Vamos a verla con mas detalle.

#### 6.2.1 Constructor por defecto

Se trata de uno de los métodos principales. Es el encargado de iniciar todas las variables y elementos necesarios para la creación del mundo tridimensional. También se encarga de la gestión de los eventos de dicho mundo.

Como se ha comentado anteriormente, Java 3D puede presentar problemas por lo que es aquí donde dichos problemas son solucionados.

### 6.2.2 Método add

Este método también es importante ya que es llamado con los valores obtenidos por los distintos descriptores seleccionados. Dado que obtiene los valores, llama a otro método encargado de posicionar las distintas imágenes y activa ciertas configuraciones sobre la escena.

### 6.2.3 Métodos para la interacción

Esta clase cuenta con dos métodos para la interacción del usuario con la visualización. Uno de ellos corresponde a la interacción mediante ratón y el otro, mediante teclado. Ambas interacciones pueden ser utilizadas a la vez para una mejor experiencia del usuario.

### 6.2.4 Método obtención de información

Como se ha comentado a lo largo de esta memoria, no sirve de nada visualizar las imágenes si no podemos obtener información sobre ellas. Por ello este método se encarga de, dada una imagen elegida por el usuario posando el ratón sobre ella, obtener la información asociada a esta y mostrarla en la esquina inferior izquierda de la visualización. Esta información es la proporcionada por los descriptores

### 6.2.5 Métodos para obtener imágenes

Dado que esta clase almacena toda la información, las clases hijas deben poder acceder a esta. Por esta razón se incorporan métodos que, dado un índice o un objeto ResultMetadata, propio de la JMR, devuelven la imagen asociada a ellos.

### 6.2.6 Método para obtener información

La información de los descriptores viene dada por un vector n-dimensional, siendo n el número de descriptores. Como en el caso anterior, existe un método que dado un índice devuelve el vector asociado a él. También existen otras formas de representar información propias de la JMR que cuentan con métodos para que las clases hijas puedan acceder a ellas.

### 6.2.7 Métodos abstractos

Estos métodos deben ser implementados por las clases hijas. En estos métodos se debe definir la distribución de las imágenes en el mundo tridimensional, y la interacción a la cual responderá la visualización.

### 6.2.8 Método guía de posición

Si el número de imágenes es elevado, el usuario puede sentirse perdido en el entorno, por lo que esta función se encarga de dibujar junto a las imágenes, un número indicando el orden en el que estas fueron dibujadas con el fin de ayudar al usuario.

### 6.2.9 Método para pintar imágenes

En este método nos encargamos del dibujado de las imágenes en el mundo tridimensional que disponemos. Para ello, a parte de la imagen propiamente dicha, obtenida mediante un ResultMetadata, necesitamos un vector tridimensional.

## 6.3 SecuencialPanel

Esta clase hereda de Abstract3DPanel, cosa que todas las siguientes harán. Representa a la visualización secuencial por lo que gracias a un buen diseño de clases el código de esta es muy reducido. Simplemente necesitamos implementar los métodos abstractos, y dado de que se trata de una visualización secuencial, no es ningún problema.

## 6.4 SpiralPanel

Representa a la visualización en espiral. En esta ocasión lo interesante es el método en el cual se establecen las posiciones de las distintas imágenes en el espacio. Pero para lograrlo basta con seguir lo explicado en la parte de análisis. No olvidar elegir un tipo de interacción.

## 6.5 PathPanel

Representa a la visualización camino y su mención en este capítulo es meramente anecdótica, ya que su implementación es trivial

## 6.6 Cartesian2DPanel

En esta visualización utilizamos las ecuaciones cartesianas para dibujar las imágenes. Por lo que simplemente debemos crear un vector tridimensional, usando los valores obtenidos por los descriptores como coordenadas x e y. La coordenada z se deja a 0 ya que se trata de una visualización 2D.

## 6.7 Cartesian3DPanel

Debido al gran parecido con la clase anterior, hereda de ella. Usa los mismos métodos, salvo que cuenta con un tercer descriptor del cual obtiene la información para el eje z, logrando ser así una representación 3D.

## 6.8 Polar2DPanel

Una vez usadas las ecuaciones cartesianas es hora de usar las polares. Al igual que en la visualización cartesiana en 2D, esta utiliza los valores de dos descriptores, pero los

transforma en coordenadas polares utilizando las fórmulas descritas en el capítulo de análisis.

## 6.9 Polar3DPanel

Como puede apreciarse, presenta grandes similitudes con Polar2DPanel, por lo que hereda de ella pero aun así debe implementar el cálculo de las coordenadas polares ya que es diferente al tratarse de una visualización 3D.

## 6.10 NPathPanel

Esta visualización representa de manera 1D el número deseado de descriptores. Para ello debe almacenar la información de cada uno de los descriptores así de sus imágenes asociadas de manera correcta para asegurar que los distintos caminos corresponden al mismo descriptor. Sin duda muy interesante debido a que puede representar un gran número de descriptores.



## Chapter 7

# Pruebas

Una opción para realizar pruebas y tests son los sistemas de integración continua. Esto consiste en realizar la compilación de un proyecto así como la ejecución de los tests descritos. Algunos ejemplos de estos son Travis y Snap-ci.

Lamentablemente, ninguno de estos sistemas admite Java 3D ya que se trata de una API pesada y que se encuentra de manera separada del Java Development Kit, JDK.

Lo que si se ha logrado es evitar la aparición de cualquier excepción durante la ejecución mediante el correcto uso de la secuencia try catch.



## Chapter 8

# Conclusiones y trabajos futuros

En este último capítulo del proyecto se va a comentar las conclusiones fruto de su realización. Por otro lado se explicarán trabajos futuros para este proyecto que no han podido ser adaptados por falta de tiempo.

### 8.1 Conclusiones

El mundo de los CBIR es extenso, y cubre un gran amplio rango de sectores. Aunque bien es cierto que las visualizaciones suelen ser secuenciales o bien en un entorno bidimensional. Es una razón por la cual creo que este proyecto puede resultar ser una novedad en este ámbito. Por otro lado me gustaría comentar algo sobre Java 3D. El entorno tridimensional creado para este trabajo puede resultar simple a priori, pero conlleva una gran cantidad de trabajo y entendimiento realizar de una manera en la que pueda ser adaptado los sistemas que así lo deseen. Dado que no poseía ningún conocimiento sobre Java 3D, pero si sobre gráficos 3D OpenGL y Java, el aprendizaje y el correcto uso de Java 3D ha supuesto una gran carga de tiempo y trabajo.

### 8.2 Trabajos futuros

Si bien el objetivo principal del proyecto ha sido la visualización de las imágenes, se buscaba una gestión de estas mediante algún tipo de sistema gestor de bases de datos que almacenase la información de estas con el fin de reducir el tiempo de cómputo realizado por cada una de las distintas visualizaciones implementadas.

También se desea continuar mejorando la interacción del usuario con las visualizaciones para mejorar el control de este sobre ella así de mejorar las indicaciones que se muestran en las visualizaciones.



# Bibliography

- [1] Dennis J Bouvier. Getting Started with the Java 3D API - Chapter 4 Interaction. [http://www.cs.stir.ac.uk/courses/CSC9N6/Java3D/Tutorial/j3d\\_tutorial\\_ch4.pdf](http://www.cs.stir.ac.uk/courses/CSC9N6/Java3D/Tutorial/j3d_tutorial_ch4.pdf).
- [2] Oracle Corporation. Instalación Java 3D. <http://download.java.net/media/java3d/builds/release/1.5.2/README-download.html>.
- [3] Oracle Corporation. Netbeans. <https://netbeans.org/>.
- [4] Freepik de flaticon.com. Icono n-caminos. [http://www.flaticon.com/free-icon/directions\\_169295#term=road&page=3&position=47](http://www.flaticon.com/free-icon/directions_169295#term=road&page=3&position=47).
- [5] Madebyoliver de flaticon.com. Icono ecuaciones cartesianas 3D. [http://www.flaticon.com/free-icon/coordinates\\_136810#term=coordinate&page=1&position=2](http://www.flaticon.com/free-icon/coordinates_136810#term=coordinate&page=1&position=2).
- [6] Madebyoliver de flaticon.com. Icono espiral. [http://www.flaticon.com/free-icon/spiral\\_137099](http://www.flaticon.com/free-icon/spiral_137099).
- [7] E.T.S.I.I.T. Apuntes de fundamentos de ingeniería del software de la E.T.S.I.I.T. de la Universidad de Granada.
- [8] Free Software Foundation. Licencia gpl-3.0. <https://www.gnu.org/licenses/gpl-3.0.html>.
- [9] BrainC from markmail.org. MouseOver - ToolTip for Shape3D. <http://markmail.org/message/bddcbgkfpnc6uen3#query:+page:1+mid:de5o6ppps3fv6dmp+state:results>.
- [10] Greg Hopkins. The Joy of Java 3D. <http://www.java3d.org/introduction.html>.
- [11] László Kozma Arto Klami and Samuel Kaski. GaZIR: Gaze-based Zooming Interface for Image Retrieval. <http://www.lkozma.net/mlmi09gazir.pdf>.
- [12] Chaoli Wang John P. Reese Huan Zhang Jun Tao Yi Gu Jun Ma and Robert J. Nemiroff. Similarity-Based Visualization of Large Image Collections. <http://www3.nd.edu/~cwang11/research/iv15-imap.pdf>.

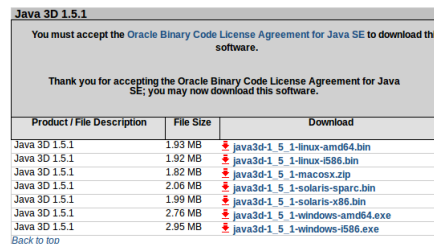
- [13] Sun Microsystems. ShadowApp creates a single plane. <http://www.java2s.com/Code/Java/3D/ShadowAppcreatesasingleplane.htm>.
- [14] Trystan G. Upstill Rajehndra Na-gappan and Nick Craswellb. Visual Clustering of Image Search Results. [http://es.csiro.au/pubs/upstill\\_spie01.pdf](http://es.csiro.au/pubs/upstill_spie01.pdf).
- [15] Henry A. Sowizral David R. Nadeau. An Introduction to Programming AR and VR Applications in Java3D. <http://viz.aset.psu.edu/jack/java3d/java3d.htm>.
- [16] Tom's Planner N.V. Toms planner software de planificación. <https://www.tomsplanner.es/>.
- [17] Oracle. Java 3D API. <http://www.oracle.com/technetwork/articles/javase/index-jsp-138252.html>.
- [18] Oracle. Javadoc Java 3D. <http://download.java.net/media/java3d/javadoc/1.5.2/allclasses-noframe.html>.
- [19] QCAD. Icono ecuaciones polares 2D. [http://www.qcad.org/doc/qcad/3.1.0/reference/en/scripts/Snap/SnapCoordinatePolar/doc/SnapCoordinatePolar\\_en.html](http://www.qcad.org/doc/qcad/3.1.0/reference/en/scripts/Snap/SnapCoordinatePolar/doc/SnapCoordinatePolar_en.html).
- [20] Francisco Velasco Anguita Francisco Javier Melero Rus. Apuntes de Sistemas Gráficos de la E.T.S.I.I.T. de la Universidad de Granada.
- [21] Ismael H. F. Santos. Setting geometry appearances. [http://webserver2.tecgraf.puc-rio.br/~ismael/Cursos/Cidade\\_CG/labs/Java3D/Java3D\\_onlinebook\\_selman/Htmls/3DJava\\_Ch09.htm#9-1](http://webserver2.tecgraf.puc-rio.br/~ismael/Cursos/Cidade_CG/labs/Java3D/Java3D_onlinebook_selman/Htmls/3DJava_Ch09.htm#9-1).
- [22] Sevarac. easyUML. <http://plugins.netbeans.org/plugin/55435/easyuml>.
- [23] Sukirgenk. Icono ecuaciones polares 3D. <http://sukirgenk.dvrlists.com/3d-cartesian-coordinate-system.html>.
- [24] Aouache Mustapha Aini Hussain Salina Abdul Samad Mohd Asyraf Zulkifley Wan Mimi Diyana Wan Zaki and Hamzaini Abdul Hamid. Design and development of a content-based medical image retrieval system for spine vertebrae irregularity. <http://biomedical-engineering-online.biomedcentral.com/articles/10.1186/1475-925X-14-6>.

## Appendix A

# Manual de usuario

### A.1 Instalación de Java 3D

La forma recomendada de instalar Java 3D es descargar los ficheros necesarios de la web de Java 3D e instalar manualmente los archivos necesarios en sus respectivos directorios. Para ello basta con seguir las instrucciones contenidas en el archivo README.txt descargado junto con Java 3D. Aunque es posible instalar Java 3D usando instaladores proporcionados por Oracle.



Java 3D 1.5.1		
You must accept the Oracle Binary Code License Agreement for Java SE to download this software.		
Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.		
Product / File Description	File Size	Download
Java 3D 1.5.1	1.93 MB	<a href="#">java3d-1_5_1-linux-amd64.bin</a>
Java 3D 1.5.1	1.92 MB	<a href="#">java3d-1_5_1-linux-i586.bin</a>
Java 3D 1.5.1	1.82 MB	<a href="#">java3d-1_5_1-macosx.zip</a>
Java 3D 1.5.1	2.06 MB	<a href="#">java3d-1_5_1-solaris-sparc.bin</a>
Java 3D 1.5.1	1.99 MB	<a href="#">java3d-1_5_1-solaris-x86.bin</a>
Java 3D 1.5.1	2.76 MB	<a href="#">java3d-1_5_1-windows-amd64.exe</a>
Java 3D 1.5.1	2.95 MB	<a href="#">java3d-1_5_1-windows-i586.exe</a>

[Back to top](#)

Figure A.1: Descarga de Java 3D

### A.2 Introducción

En esta sección vamos a elaborar un pequeño manual de usuario sobre este proyecto con el objetivo de ayudar a los usuarios que lo utilizan por primera vez.

Al iniciar el programa nos encontraremos con la siguiente figura:

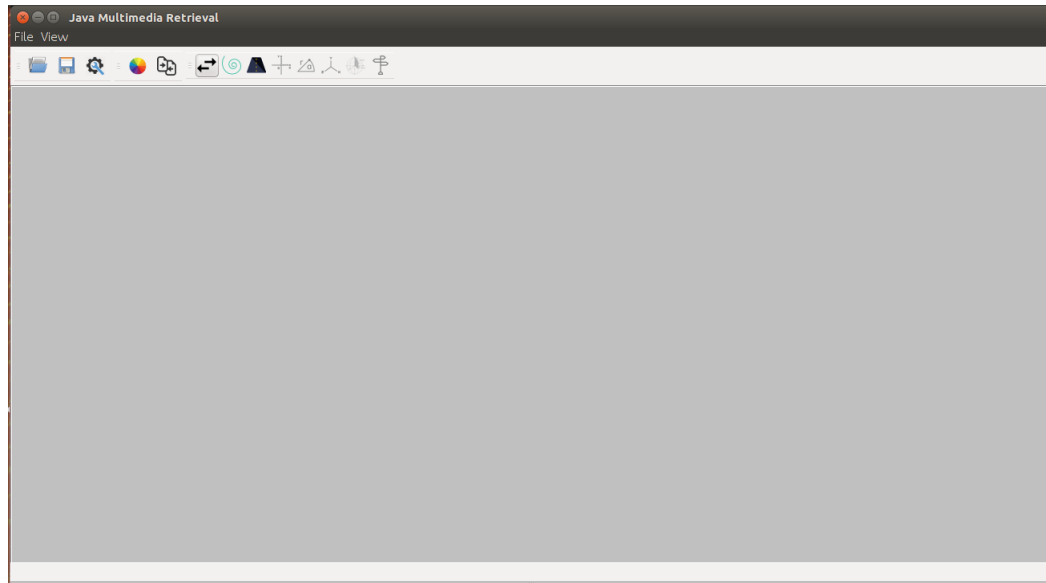


Figure A.2: Pantalla principal

En ella se pueden ver todas las opciones disponibles como abrir una o varias imágenes, elegir los descriptores a utilizar, la visualización deseada, así de como diferentes características menos importantes.

### A.3 Carga de imágenes

Para cargar imágenes basta con pulsar sobre el botón *abrir*. Al hacer esto nos encontraremos ante un diálogo que nos permitirá seleccionar la imagen o imágenes deseadas. Nota: Este diálogo varía según el sistema operativo utilizado.



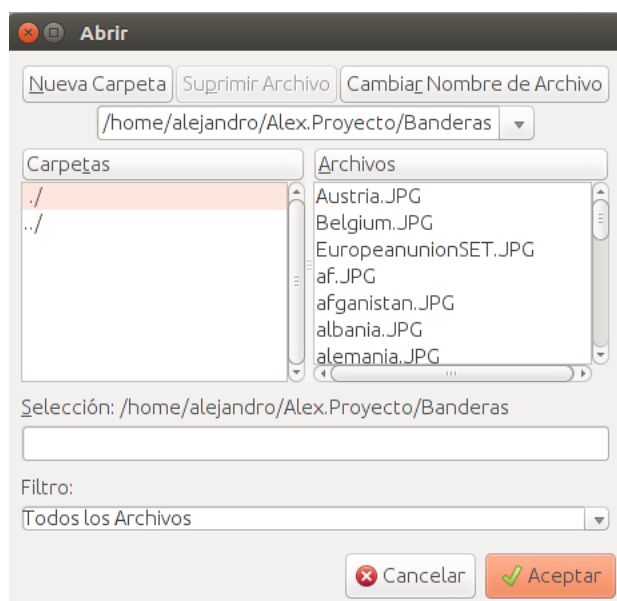


Figure A.3: Diálogo apertura imágenes

Elegidas las imágenes el resultado es el siguiente.

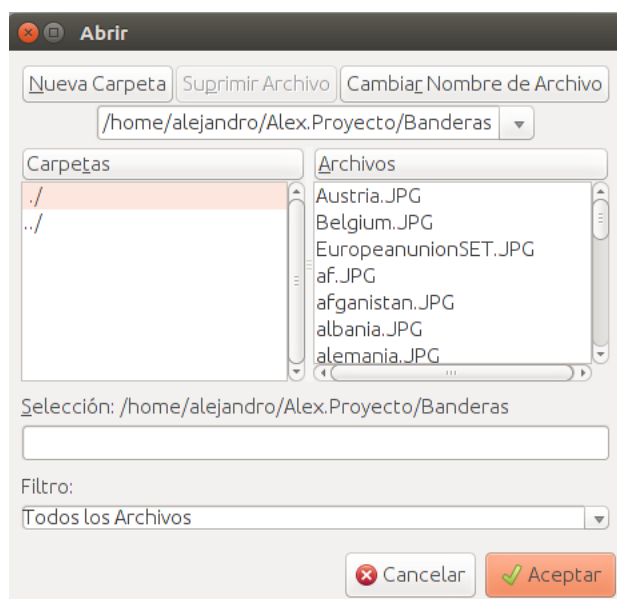


Figure A.4: Imágenes abiertas

## A.4 Elección de descriptores

Una vez decididas y cargadas las imágenes es hora de elegir los descriptores que queremos usar. Para ello basta con pulsar el botón de descriptores para que aparezca un menú, el cual nos permite seleccionarlos. Algunas visualizaciones requieren un mínimo de descriptores, por lo que si se seleccionan menos, estas permanecerán desactivadas.

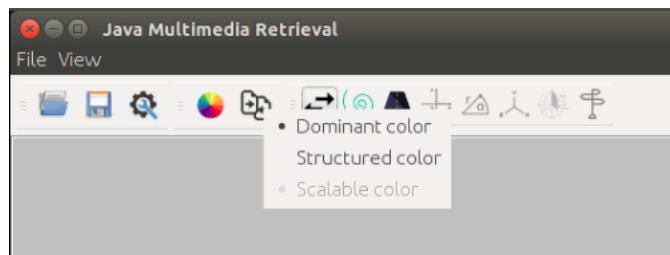


Figure A.5: Elección de descriptores 1

Como se ha comentado anteriormente, algunas visualizaciones se encuentran desactivadas ya que solo hemos seleccionado un descriptor. Si eligiéramos dos o más estas se activarían.

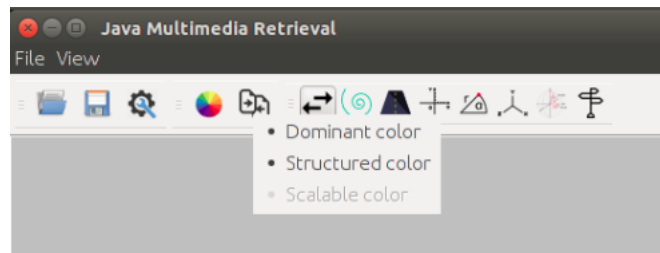


Figure A.6: Elección de descriptores 2

## A.5 Interacción

Antes de utilizar alguna de las visualizaciones disponibles es necesario saber como interactuar con ellas, ya que la interacción puede ser diferente dependiendo de la vista elegida. Independientemente de la interacción activa, al posar el ratón sobre una imagen obtendremos su información, la cual será mostrada en la esquina inferior izquierda.



Figure A.7: Información de la imagen

### A.5.1 Teclado

Los diferentes controles utilizando teclado son:

**Rotación izquierda:** Flecha izquierda.

**Rotación derecha:** Flecha derecha.

**Acercar zoom:** Flecha arriba.

**Alejar zoom:** Flecha abajo.

**Rotación arriba:** PgUp.

**Rotación abajo:** PgDn.

**Desplazamiento izquierda:** Alt + flecha izquierda.

**Desplazamiento derecha:** Alt + flecha derecha.

**Desplazamiento arriba:** Alt + PgUp.

**Desplazamiento abajo:** Alt + PgDn.

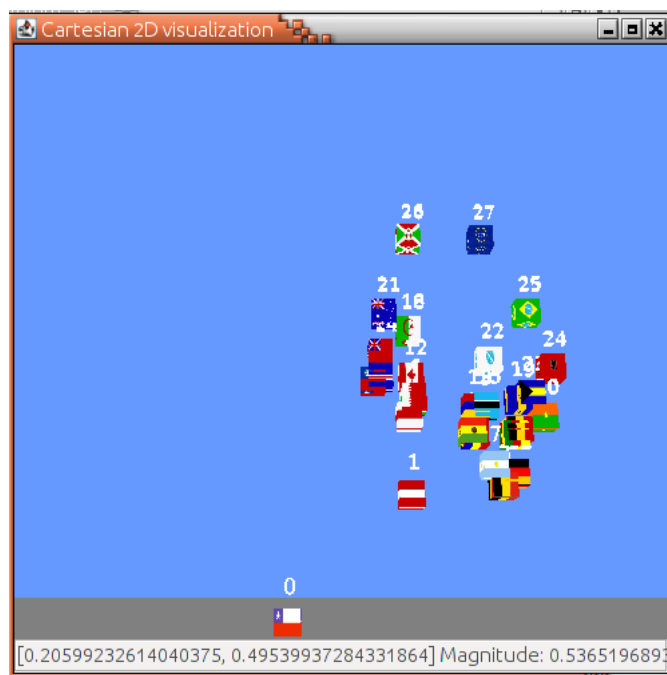


Figure A.8: Ejemplo rotación hacia arriba teclado

### A.5.2 Ratón

Los diferentes controles utilizando ratón son:

**Rotación:** Arrastrar el ratón con el botón izquierdo pulsado.

**Desplazamiento:** Arrastrar el ratón con el botón derecho pulsado.

**Zoom:** Utilizar la rueda del ratón.

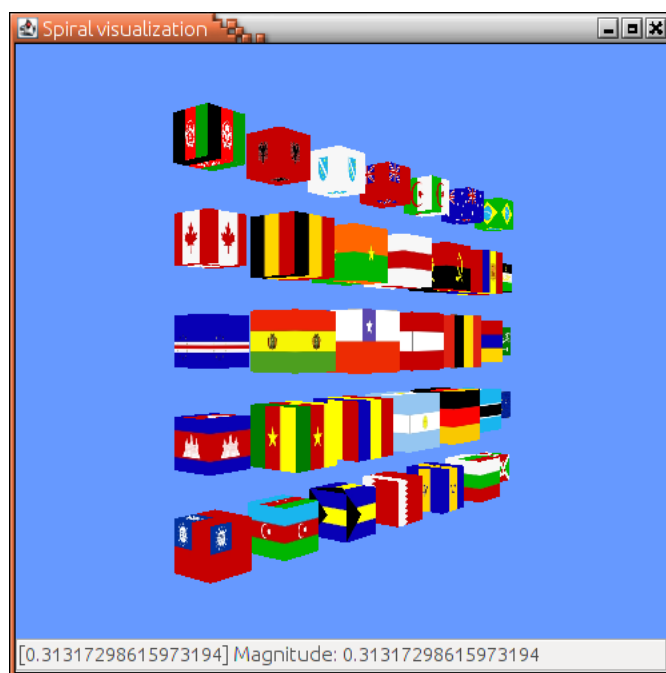


Figure A.9: Ejemplo rotación eje x ratón

## A.6 Visualizaciones

Ahora que ya disponemos de las imágenes, descriptores y sabemos como realizar la interacción, procedamos a seleccionar una vista.



Figure A.10: Ejemplo de visualización 1

Como podemos ver, encima de cada imagen aparece un número, indica el orden en el que fueron dibujadas. Lo que es de gran ayuda ya que si no estuviera, no sabríamos movernos por la visualización correctamente.

Esto obtendríamos si nos desplazásemos a la izquierda.



Figure A.11: Ejemplo de visualización 2

## A.7 Opciones

Cada visualización cuenta con un menú desplegable con una serie de opciones comunes a todas ellas. Veamos cada una en detalle.

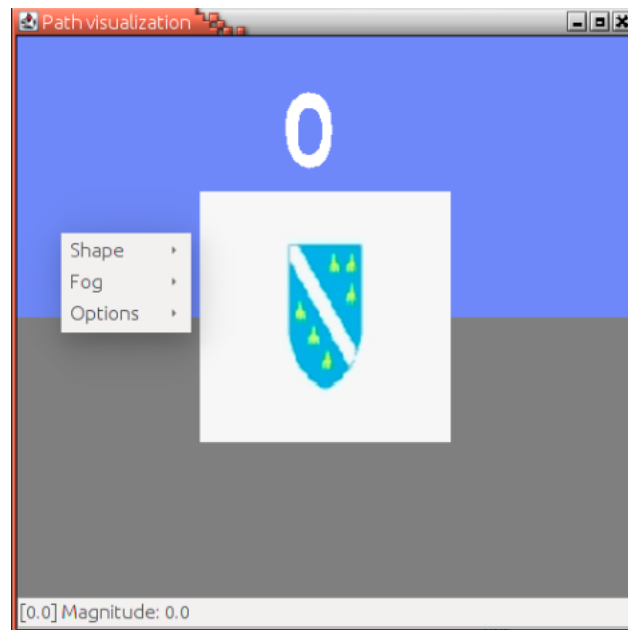


Figure A.12: Menú desplegable de opciones

### A.7.1 Reiniciar posición

Esta opción lleva al usuario a la posición de partida de la visualización. Útil si el usuario se encuentra perdido o quiere volver a la posición inicial



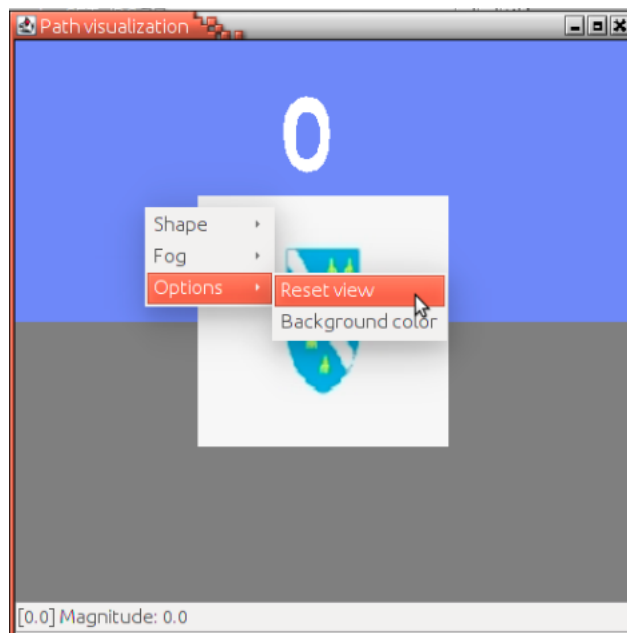


Figure A.13: Reinicio de posición

### A.7.2 Cambio de primitiva

Entendemos por primitiva la forma geométrica en la cual se van a dibujar las diferentes imágenes, pudiendo ser un cubo o una esfera. Para cambiar esto basta con elegir la forma deseada en el apartado correspondiente del menú desplegable.

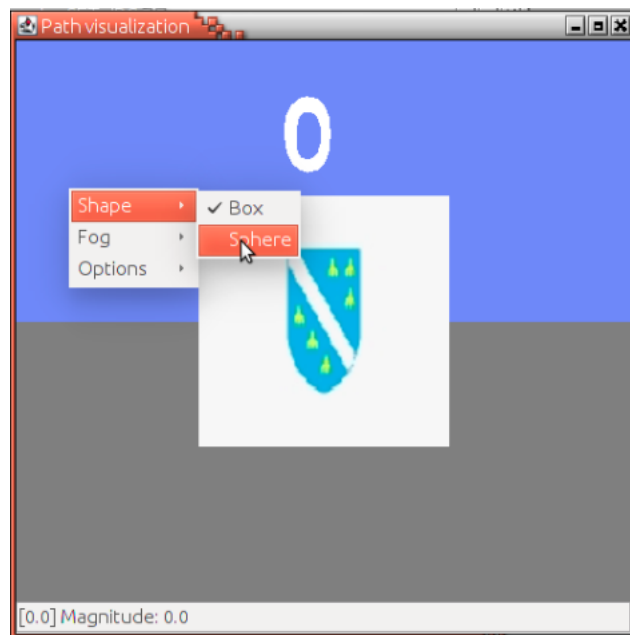


Figure A.14: Elección de primitiva



Figure A.15: Cambio de primitiva

### A.7.3 Niebla

Como añadido se encuentra la posibilidad de añadir un tipo de niebla al escenario. Su uso está desactivado por defecto y es completamente opcional, ya que no añade ninguna información importante.

Para seleccionarla basta con elegirla en el menú desplegable, también se puede cambiar su color desde el menú.

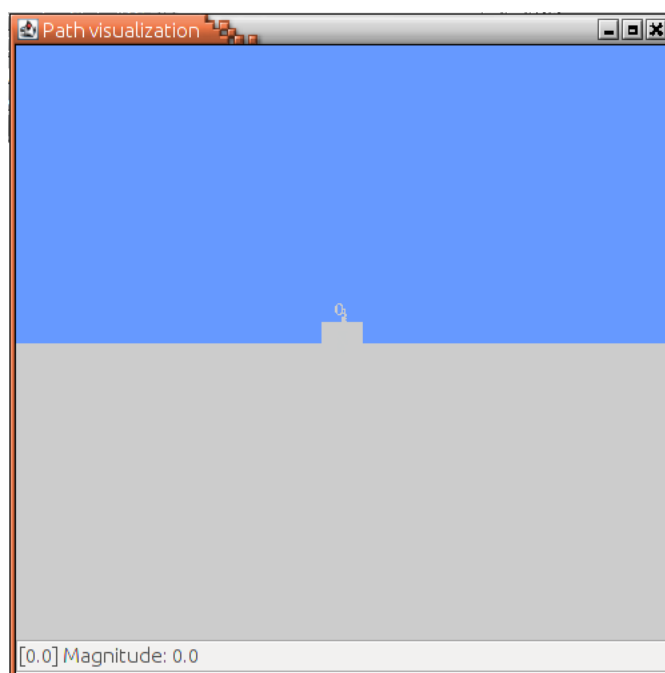


Figure A.16: Niebla activada



Figure A.17: Niebla al acercarse a una imagen

#### A.7.4 Fondo

El color del fondo de la visualización puede ser cambiado para poder ver con mayor claridad las imágenes en caso de que el color de estas coincida con el del fondo.

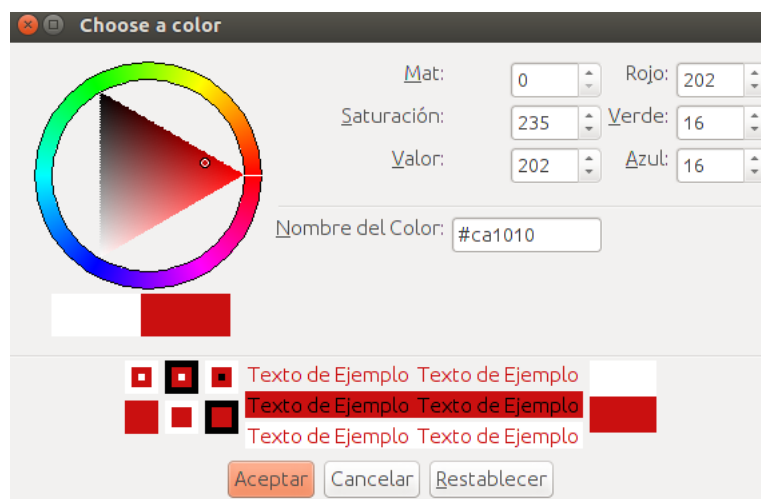


Figure A.18: Elección del color



Figure A.19: Color cambiado



## Appendix B

# Licencia

La licencia de este proyecto es GNU General Public License versión 3. A continuación va a detallar dicha licencia.





# GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Copyright © 2007 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program—to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose com-

puters, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

## TERMS AND CONDITIONS

### 0. Definitions.

“This License” refers to version 3 of the GNU General Public License.

“Copyright” also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

“The Program” refers to any copyrightable work licensed under this License. Each licensee is addressed as “you”. “Licensees” and “recipients” may be individuals or organizations.

To “modify” a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a “modified version” of the earlier work or a work “based on” the earlier work.

A “covered work” means either the unmodified Program or a work based on the Program.

To “propagate” a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To “convey” a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays “Appropriate Legal Notices” to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

### 1. Source Code.

The “source code” for a work means the preferred form of the work for making modifications to it. “Object code” means any non-source form of a work.

A “Standard Interface” means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular

programming language, one that is widely used among developers working in that language.

The “System Libraries” of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component”, in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work’s System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

## 2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- (a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- (b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".
- (c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

- (d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an “aggregate” if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation’s users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

#### 6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- (a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- (b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.
- (c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- (d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

- (e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A “User Product” is either (1) a “consumer product”, which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, “normally used” refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

“Installation Information” for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

## 7. Additional Terms.

“Additional permissions” are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- (a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- (b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- (c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- (d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- (e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- (f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered “further restrictions” within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.



If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

#### 8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

#### 9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

#### 10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party’s predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

#### 11. Patents.

A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor’s “contributor version”.

A contributor’s “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor’s essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a “patent license” is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To “grant” such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream

recipients. “Knowingly relying” means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient’s use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is “discriminatory” if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

#### 12. No Surrender of Others’ Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

#### 13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public

License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License “or any later version” applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy’s public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR

A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

## END OF TERMS AND CONDITIONS

### How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.>

Copyright (C) <textyear> <name of author>

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <<http://www.gnu.org/licenses/>>.

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

<program> Copyright (C) <year> <name of author>

This program comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.  
This is free software, and you are welcome to redistribute it  
under certain conditions; type 'show c' for details.

The hypothetical commands `show w` and `show c` should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <http://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <http://www.gnu.org/philosophy/why-not-lgpl.html>.