**Universidad de las Ciencias Informáticas**

# OPTIMIZACIÓN DE TEXTURAS PROCEDURALES DE PRE-CÁLCULO

**TRABAJO DE DIPLOMA EN OPCIÓN AL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS**


**AUTORES:** Osdalme Fuentes Colina
Yanisleydis Sampedro Ruiz

**TUTOR:** Lic. Lidiexy Alonso Hernández

**CO-TUTOR:** Ing. Frank Puig Placeres



Ciudad de la Habana

Julio de 2007

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los ___ días del mes de julio del año 2007.

_____
Osdalme Fuentes Colina
Autor

_____
Yanisleydis Sampedro Ruiz
Autor

_____
Lic. Lidiexy Alonso Hernández
Tutor

_____
Ing. Frank Puig Placeres
Co-Tutor

*A nuestros padres…*

*"El batir de las alas de una mariposa produce tifones*
*en el otro extremo del mundo"*
*Teoría del Caos*

## AGRADECIMIENTOS

*A mi mamá por su cariño inmenso, sus consejos y sus incesantes esfuerzos que hicieron posible que llegara a ser quien soy. A Migue por quererme como su hija y ayudarme en todo. A mi abuelita por mimarme tanto. A mi hermano, a mi abuelo, a mis primos, a mis tías, en fin a toda mi familia por depositar en mi ese cariño y esa confianza que tanto necesitaba.*

*A Marielena y Francisco por brindarme su apoyo en todo momento.*

*Y a Frank por ser la luz que ilumina mi camino.*

***Yani***

*A mi familia, por estar presentes en todo momento.*

*A mis amigos, por su apoyo incondicional.*

*A Frank, Lidiexy, Minardo, Yeisnier,*

*Andrés, Nilet, Hardys e Ismael.*

*A los que colaboraron en la*

*realización de la tesis,*

*Muchas gracias.*

***Osdalme***

# RESUMEN

Las texturas han posibilitado representar modelos tridimensionales con alto nivel de detalle, evitando recurrir a la creación de modelos con una complejidad geométrica superior. Entre las que han revolucionado los gráficos por computadora se encuentran las Texturas Procedurales de Pre-cálculo, pues almacenan datos pre-calculados de los modelos tridimensionales que son muy costosos de realizar en tiempo real. Estos datos permiten crear una serie de efectos visuales como relieve, iluminación y sombras, que aumentan significativamente el realismo de los objetos y escenas virtuales.

Esta tesis propone un proceso iterativo para la creación de Texturas Procedurales de Pre-cálculo, donde una vez creados los modelos 3D y las texturas en una herramienta de diseño, se procesan en la aplicación implementada, con el objetivo de mejorar su eficiencia. La aplicación contiene varias optimizaciones cuya función es realizar diferentes análisis en el espacio asignado a cada polígono en la textura, para modificar las coordenadas de textura del modelo tridimensional. De esta forma, al generar en el programa de diseño una Textura Procedural de Pre-cálculo con las nuevas coordenadas, se logra un mejor aprovechamiento del espacio de textura.

Palabras claves: *Textura Procedural de Pre-cálculo, eficiencia*.

# ABSTRACT

The textures have made possible to represent three-dimensional models with a high level of detail, avoiding using complex geometries. Between the ones that have revolutionized the computer graphics there are the Procedurales Lookup Textures, because they store pre-computed data of the three-dimensional models that could be very expensive to make in real time. Those data allows creating a series of visual effects like relief, illumination and shades, which significantly increases the realism of the objects and virtual scenes.

This thesis proposes an iterative process for the creation of Procedurales Lookup Textures, where once created the 3D models and the textures in a design tool, they are analyzed and processed in the application implemented as part of this research with the objective of improving the texture's efficiency. The application contains several optimizations whose purpose is to make different analyses in the texture space assigned to each polygon, and then to modify the texture coordinates of the three-dimensional model. This way, when generating the Procedural Lookup Texture in the design program, by using the new coordinates, it is obtained an improvement in the texture space.

Keywords: *Procedurales Lookup Textures, efficiency.*

# ÍNDICE

# INTRODUCCIÓN

En la actualidad los diseñadores de entornos virtuales se encuentran en una carrera cuya meta es lograr mundos o diseños con un alto nivel de realismo. Para cumplir este objetivo han surgido muchas soluciones, pero indudablemente, la adición de texturas a los objetos tridimensionales es una de las soluciones más significativas hasta ahora alcanzadas; debido a que las texturas representan los detalles de las superficies de los polígonos.

Las texturas se pueden clasificar atendiendo a muchos parámetros. Clasificándolas según su forma de creación, se obtienen dos grandes grupos.

En el primer grupo se encuentran los mapas de bits, que son las imágenes escaneadas, las fotos y dibujos digitales realizados por el hombre mediante una herramienta de edición de imágenes, tal como Photoshop, Corel Draw, entre otras. Este tipo de textura es la más común y es de gran importancia ya que el diseñador puede determinar con exactitud la ubicación de todos los elementos que estarán presentes en la textura.

En el segundo grupo se encuentran las Texturas Procedurales. Estas texturas son generadas mediante algoritmos matemáticos y tienen la característica de que pueden ser creadas en tiempo real o previamente.

Dentro de este último grupo existe un subgrupo de texturas que se generan mediante algoritmos matemáticos y se almacenan para ser aplicadas posteriormente como un mapa de bits. Los texeles que conforman estos tipos de texturas almacenan datos pre-calculados de modelos tridimensionales en forma de colores. A este subgrupo se les denomina Texturas Procedurales de Pre-cálculo.

Estas texturas permiten crear una serie de efectos visuales como son: iluminación, relieve, sombras, entre otros, que permiten añadir gran cantidad de detalles a las superficies de los modelos tridimensionales. Para crear en tiempo real estos efectos visuales se necesitan realizar cálculos muy costosos, sin embargo es posible obtenerlos de forma sencilla si estos datos se pre-calculan, se almacenan y posteriormente se aplican como una textura común, es decir, creando y utilizando una Textura Procedural de Pre-cálculo.

Actualmente existen muchos programas que generan este tipo de textura, entre ellos se encuentran el 3D Studio Max, el Maya y el Lightwave. Sin embargo con estos programas se obtienen texturas ineficientes por diversos motivos, entre ellos se encuentran los siguientes:

- El espacio que le corresponde en la textura a cada polígono del modelo tridimensional es asignado de acuerdo a la geometría del polígono y no se tiene en cuenta la complejidad de ese espacio de textura.
- El espacio de textura perteneciente a cada polígono no se acomoda de la mejor forma, lo cual provoca pérdida de espacio, es decir, que exista un área considerable de la textura sin utilizar.
- Si existen varios polígonos iguales se le asigna un espacio en la textura a cada uno, en lugar de asignarle espacio a uno solo y hacer que los demás utilicen esa misma área.
- Si hay varios polígonos que tienen bordes iguales, se deja espacio entre ellos. Perdiéndose esos texeles que separan los polígonos en lugar de aprovecharlos para representar otros.

Como se puede apreciar, estas texturas son de gran importancia en la representación de mundos virtuales. Eliminando las ineficiencias señaladas, es posible lograr un mayor nivel de detalle en los modelos tridimensionales o disminuir el consumo de memoria.

Esta tesis propone un proceso para la creación de Texturas Procedurales de Pre-cálculo, en el cual se integra una herramienta con el objetivo de mejorar el aprovechamiento del espacio de textura, es decir, elevar la eficiencia de la textura.

**Problema Científico**

¿Cómo elevar la eficiencia de las Texturas Procedurales de Pre-cálculo?

**Objeto de Estudio**

Las Texturas Procedurales de Pre-cálculo y coordenadas de textura vinculadas a un modelo tridimensional.

**Objetivo de la Investigación**

Crear un proceso iterativo que permita aumentar la eficiencia de las Texturas Procedurales de Pre-cálculo.

**Campo de Acción**

Correspondencia entre la asignación de las coordenadas de textura y la eficiencia de las Texturas Procedurales de Pre-cálculo generadas.

**Tareas**

1. Investigar el proceso actual de creación de Texturas Procedurales de Pre-cálculo.
2. Elaborar algoritmos para obtener el porciento de espacio de textura ocupado y catalogar la relevancia de las diferentes áreas de las Texturas Procedurales de Pre-cálculo.
3. Implementar un software que utilice los algoritmos de análisis de la relevancia y el porciento de espacio de textura ocupado para modificar coordenadas de textura, de forma tal que mejore el aprovechamiento del espacio de textura.
4. Integrar el software en el proceso de generación de Texturas Procedurales de Pre-cálculo.
5. Realizar pruebas al software implementado.

**Organización del documento**

El presente documento se estructura en cinco capítulos:

El Capítulo 1 presenta una introducción a las Texturas Procedurales de Pre-cálculo; mostrándose los conceptos básicos relacionados e incluyendo el proceso actual de creación de estas texturas. Además contiene la justificación de las herramientas y lenguajes utilizados.

El Capítulo 2 expone las características del sistema como solución a los problemas planteados, presentando el flujo de trabajo y funcionalidades de la herramienta en general.

El Capítulo 3 contiene el modelo del dominio, los requisitos y los modelos de casos de uso del sistema. Además se muestra un manual de usuario, para facilitar el uso eficiente de la aplicación.

El Capítulo 4 describe el proceso de diseño de clases, utilizando para ello los diagramas de clases de diseño y de secuencia, así como el diagrama de componentes.

El Capítulo 5 muestra los resultados del software implementado, obtenidos a partir de diferentes pruebas realizadas al mismo.

# 1

# CAPÍTULO
# FUNDAMENTACIÓN TEÓRICA

Este capítulo introduce los conceptos básicos relacionados con las texturas. Al mismo tiempo ofrece una visión general del proceso actual de generación de Texturas Procedurales de Pre-cálculo. Además presenta las herramientas y los lenguajes seleccionados para la realización de esta tesis.

## 1.1 ¿Qué es una textura?

Para entender qué es una textura, resulta útil imaginarla como una pared de azulejos donde cada azulejo puede ser de un color diferente, pero cuando se mira la pared desde lejos se percibe una imagen. Esto mismo ocurre con las texturas, ya que desde el punto de vista de su almacenamiento en memoria son representadas mediante arreglos de colores. A cada uno de estos colores se le denomina texel, que es la unidad mínima de una textura aplicada a una superficie (1) y cuando se miran como un conjunto forman la imagen. (Ver Figura 1).

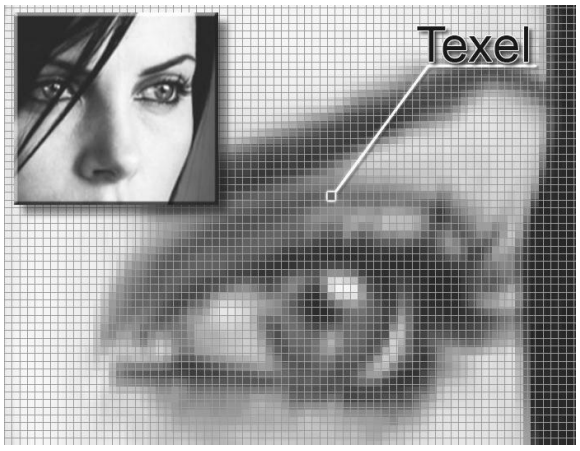



Fig.1 Las texturas están compuestas por texeles

Comúnmente las texturas se representan mediante arreglos unidimensionales, bidimensionales o tridimensionales de texeles.

Las texturas unidimensionales son empleadas en tareas especiales, tales como determinar los colores de un mapa de niveles (por ejemplo, el color del agua, la transición del agua y la arena, los pastos o montañas y la nieve), es decir, no se necesita gran información de los detalles de la textura, solo se necesita conocer el color de cada altura para representarlo.

Las texturas bidimensionales (texturas 2D) son imágenes representadas mediante arreglos de dos dimensiones, en los cuales una dimensión del arreglo corresponde al número de texeles del ancho de la textura y la otra corresponde a la cantidad de texeles del largo de la textura. Es por ello que cuando se habla de una textura de 512 x 512 se refiere a una textura formada por 512 texeles a lo largo y 512 texeles

a lo ancho. Estas texturas son las más comunes y se utilizan principalmente para representar la imagen de la superficie de un objeto.

Las texturas tridimensionales (texturas 3D) son representadas mediante arreglos tridimensionales, donde las dos primeras dimensiones son iguales que las texturas 2D. Al agregar la tercera dimensión los desarrolladores pueden acceder a un componente de profundidad en la información de la textura. A diferencia de las texturas 2D estas texturas no solo describen la superficie del objeto sino que también pueden definir su interior. Por ejemplo, una veta de color que atraviesa una estatua de mármol se puede describir fácilmente mediante una textura 3D, ya que atraviesa el centro de la estatua y sale por el otro lado.

Como se mencionó anteriormente las texturas están compuestas por texeles. Cada texel consta de tres o cuatro componentes en dependencia de si la imagen es de 24 ó 32 bits, es decir, si la imagen posee transparencia o no. Estos componentes son: Red, Green, Blue (RGB) los colores primarios en la computadora y el componente Alfa que es la transparencia.

La Figura 2 muestra los componentes de la textura por separado y el resultado de la combinación de estos para formar una imagen colorida y con posibilidad de transparencia.

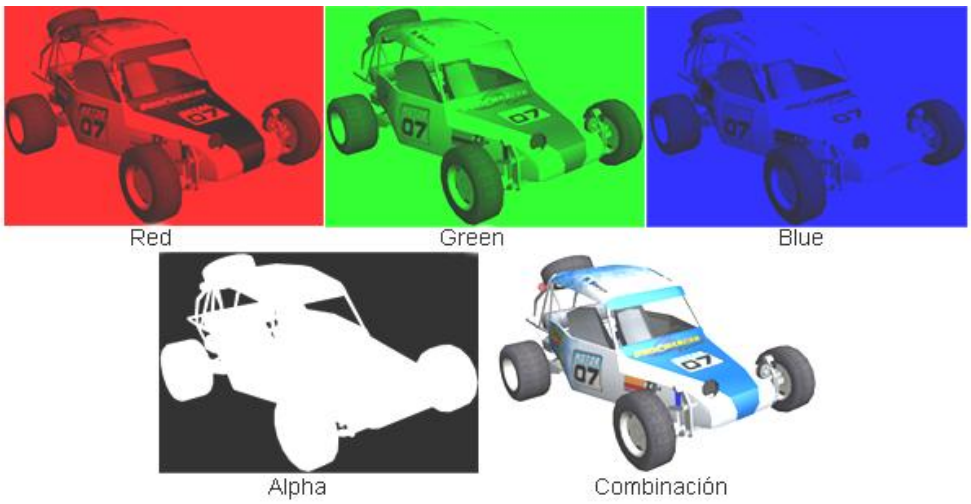



Fig.2 Componentes de un texel

Este documento solo hará referencia a las texturas bidimensionales, pues las Texturas Procedurales de Pre-cálculo son comúnmente almacenadas en este tipo de texturas.

## 1.2 Asignación de coordenadas de textura.

Al proceso comúnmente usado para proyectar una imagen 2D sobre una superficie de un objeto 3D, o sea, al proceso de aplicar una textura a un objeto tridimensional, se le denomina *mapeado*. El mapeado consiste en envolver al objeto con la textura, asignando a cada vértice del objeto una posición en la textura.

Para crear un mundo tridimensional ideal se tendría que modelar cada átomo que lo compone, cosa que es imposible de realizar debido a las limitaciones de procesamiento de las computadoras actuales. Pero gracias a las texturas ha sido posible realizar una aproximación a nivel de detalle visual de ese mundo ideal. Con el empleo de ellas se ha logrado incrementar eficazmente el detalle y el realismo de la superficie de un modelo 3D, sin necesidad de aumentar su complejidad geométrica.

En la Figura 3 se presenta una imagen en la que se demuestra la necesidad de aplicarles texturas a los modelos tridimensionales.


Fig.3 Ejemplo de mapeado

Para visualizar un objeto con textura, es necesario proporcionar para cada vértice tanto las coordenadas geométricas (x, y, z) como las de textura. Siendo los componentes de las coordenadas de texturas nombrados generalmente como ‘u’ y ‘v’, en lugar de ‘x’ y ‘y’ donde ‘u’ es el eje horizontal y ‘v’ el vertical. Este cambio de nombre es realizado para que se tenga en cuenta que las coordenadas de texturas indican posiciones de vértices en la textura, mientras que las clásicas x, y, z se refieren a la escena, es decir, a la posición de los vértices en el mundo virtual.

Las coordenadas de los vértices que conforman la geometría de una escena se encuentran en un rango de –infinito a +infinito; a este rango se le denomina espacio del mundo. A diferencia de estas, las

coordenadas de textura solo proyectan valores únicos en el rango de 0 a 1 [0..1]; llamándosele a este rango, espacio de textura.

En el proceso de mapeado las coordenadas de textura se pueden asignar fuera del rango [0..1], implicando que la textura se aplique de forma repetida. Este proceso se denomina *tiling* y se muestra en la Figura 4.

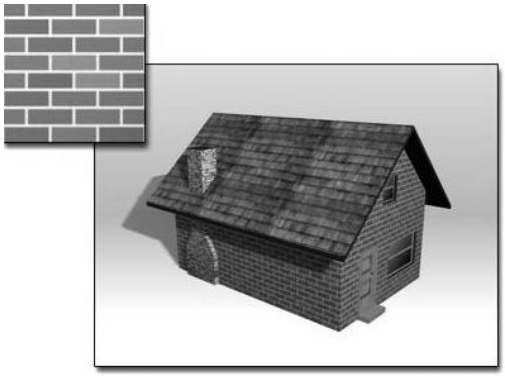


Fig.4 Uso del tiling en el mapeo de textura

## 1.3 Texturas Procedurales.

Las texturas se pueden clasificar en muchos grupos atendiendo a diferentes parámetros. En este trabajo se ha tenido en cuenta su forma de obtención, por lo cual se han dividido en dos grandes grupos.

En el primer grupo se encuentran las Texturas no Procedurales, que no son más que fotografías, imágenes escaneadas o dibujos creados por programas de edición de imágenes, tal como Photoshop y Corel Draw. Las imágenes pintadas manualmente le dan al artista un control total sobre el resultado final, debido a que pueden controlar cada texel sobre la superficie.

El segundo grupo está formado por las Texturas Procedurales. Estas texturas son generadas por la computadora usando algoritmos matemáticos. De esta forma se pueden representar de forma realista elementos naturales tales como la madera, el metal, el mármol, la piedra, entre otros. Con este tipo de textura no resulta necesario utilizar memoria para almacenar la información de cada texel, ya que son generadas algorítmicamente cuando se necesite.

### 1.3.1 Ventajas de las Texturas Procedurales. (2)

- Se obtienen imágenes con un alto nivel de detalle, limitadas solamente por la precisión y el volumen de cálculos necesario para crearlas.
- Pueden abarcar extensas áreas sin necesidad de repetir la textura.
- Se pueden generar variaciones en ellas con gran facilidad.

### 1.3.2 ¿Cuándo usar Texturas Procedurales? (2)

- Cuando se quiere hacer efectos animados como el agua, el fuego, las explosiones, la vegetación, entre otros.
- Cuando se tienen objetos muy grandes donde se nota el tiling.
- Cuando se tiene una colección de objetos iguales y se desea aparentar que los objetos son diferentes.

### 1.3.3 Ejemplos de Texturas Procedurales.

- **Ruido**

La textura de ruido no es más que la perturbación al azar de una superficie basada en la interacción de colores o materiales. También se puede ver como artefactos extraños o ajenos a la imagen.

Este tipo de textura es extremadamente importante, pues es usada como base de muchos efectos. Se utiliza para crear texturas de piedra, metales, concreto, entre otras. Además permite añadirle realismo a una escena, pues agrega imperfecciones.

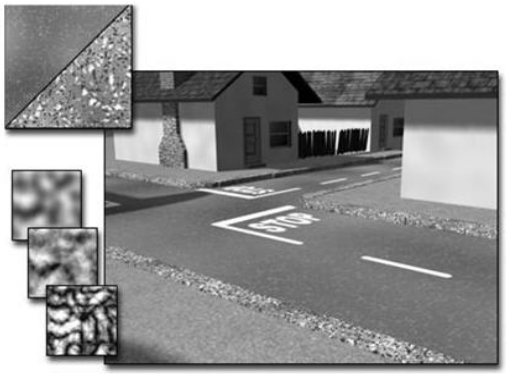



Fig.5 Ejemplo de aplicación del ruido

En Figura 5 se muestra la aplicación de texturas de ruido en la calle, el contén, la acera y en las paredes de las casas. Estas texturas permiten impregnar un mayor realismo en la escena, pues logran que la misma no se vea perfecta; como ocurre en el mundo real, donde la mayoría de las superficies no son perfectamente lisas o no se encuentran uniformemente pintadas.

- **Gradiente**

Un gradiente es una textura compuesta de transiciones suaves de colores o niveles de brillo, es un degradado de la pintura desde un color a otro. Se crea especificando un color en un punto y otro color en

otra posición. De esta forma, los colores van cambiando gradualmente de uno a otro siguiendo una línea recta entre ambos puntos. (3)

Las texturas de gradiente se utilizan fundamentalmente para dar la sensación de profundidad, densidad y compresión de los elementos de una escena. Además se utilizan para hacer cambios de colores en los modelos como se representa en la Figura 6.

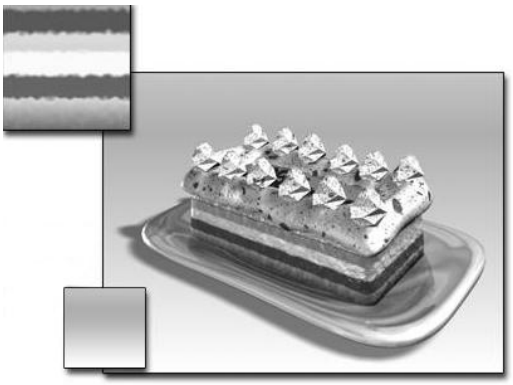


Fig.6 Ejemplo de aplicación del gradiente

### 1.4 Texturas Procedurales de Pre-cálculo.

Existe un grupo de Texturas Procedurales que se generan mediante algoritmos matemáticos y son almacenadas y aplicadas como una textura estándar. Este grupo es conocido como Texturas Procedurales de Pre-cálculo.

Comúnmente este tipo de textura no almacena el color de un objeto, sino que en los valores de cada componente de color guarda información pre-calculada del objeto. Entre los datos que puede almacenar se encuentran la iluminación del modelo, la radiosidad, las normales, entre otros. Para la obtención de estos datos es necesario hacer cálculos muy costosos, que al ser almacenados previamente se evita realizarlos en el momento de la simulación.

Es por ello que la principal ventaja de estas texturas es que aumenta la rapidez en la visualización del modelo en tiempo real. La desventaja fundamental es que se consume mucha memoria, pues para hacer una representación del modelo tridimensional con la calidad requerida se necesita una textura grande, de forma tal que pueda contener el mayor número posible de detalles de la superficie.

Entre las Texturas Procedurales de Pre-cálculo más comunes se encuentran: los Mapas de Luces (LightMaps), los Mapas de Normales (NormalMaps) y los Mapas de Alturas (HeightMaps). A continuación se muestran las principales características de cada uno de esos mapas.

### 1.4.1 Mapas de Luces.

Los Mapas de Luces son texturas que almacenan información referente a la iluminación. Para aplicar un LightMap sobre un modelo primero se pre-calcula la iluminación de la superficie del modelo mediante un algoritmo de iluminación y se almacena en una textura. Estas texturas generalmente se representan en una escala de grises, donde el blanco representa la luz que recibe cada cara y el negro las sombras. Luego mediante un algoritmo de múltiples pasadas, o el uso de la combinación de múltiples texturas en el hardware gráfico, estas se mezclan y se obtiene el objeto iluminado sin necesidad de hacer el costoso cálculo de la luz en tiempo real.

El lightmapping ha sido una técnica muy popular durante los últimos años, ya que permite incluir iluminación en una escena dibujada en tiempo real, con un detalle muy superior a los tradicionales sistemas que efectúan los cálculos en tiempo real (4). El funcionamiento básico de esta técnica queda reflejado en la Figura 7, en la que se aprecia una textura base a la que se le multiplica el LightMap correspondiente.


Fig.7 Ejemplo de aplicación de los Mapas de Luces

### 1.4.2 Mapa de Normales.

Un Mapa de Normales es una textura que almacena información sobre la dirección de las normales de cada una de las caras de un objeto. Siendo una "normal" el vector perpendicular a la superficie con la dirección hacia el exterior del objeto.

Uno de los parámetros que se tiene en cuenta para calcular la iluminación de un objeto son las normales. Por ejemplo, en el componente difuso, si una normal apunta directamente a la fuente de luz esta cara será iluminada con el 100% de potencia (suponiendo que la luz no tenga decaimiento).


Fig.8 Ejemplo de aplicación de los Mapas de Normales

En el mapeado de normales, la normal se almacena directamente en los valores RGB de la textura, valores que se utilizan para calcular la iluminación de los objetos. Como resultado del mapeado, la luz reaccionará bien a los relieves y aunque se observe desde cualquier ángulo, parecerá que realmente hay geometría. (Ver Figura 8).

### 1.4.3 Mapa de Altura.

Un Mapa de Altura o HeightMap es una textura que se utiliza para almacenar valores de la elevación de una superficie. Son aplicados principalmente a los terrenos para obtener detalles extra y realismo, esto se logra agregando perturbaciones a la superficie mediante la distorsión de la luz o la modificación de los vértices de la geometría.

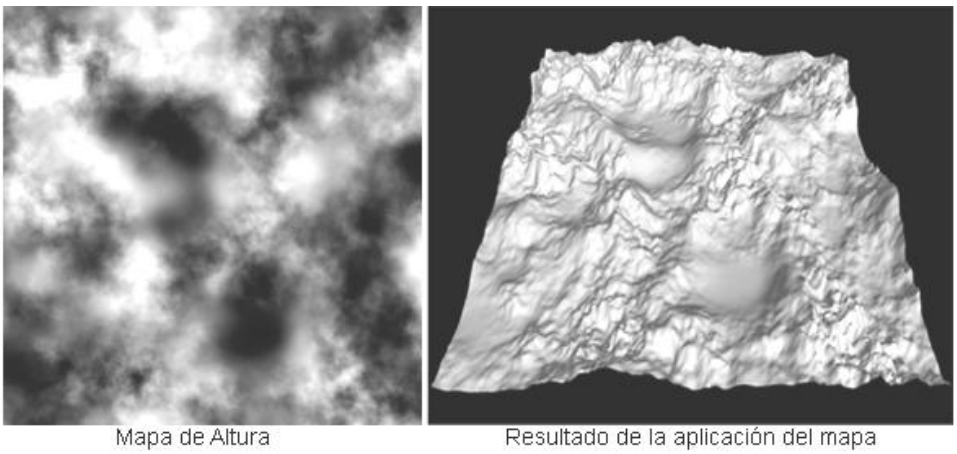



Fig.9 Ejemplo de aplicación de un Mapa de Altura

Un HeightMap almacena en los componentes RGBA de cada texel información acerca de la distancia de desplazamiento o altura del piso de una superficie. Es visualizado como una imagen en escala de grises, donde el negro significa la mínima altura y el blanco la máxima altura. En la Figura 9 se muestra un ejemplo donde aplica a una superficie un Mapa de Altura.

### 1.5 UVAtlas.

UVAtlas es una librería para generar coordenadas de texturas únicas en modelos tridimensionales, o sea, a cada polígono del modelo tridimensional se le asigna un único espacio en la textura. De esta forma, si se altera el texel que le corresponde a un polígono, no será alterado el espacio correspondiente a otro polígono.

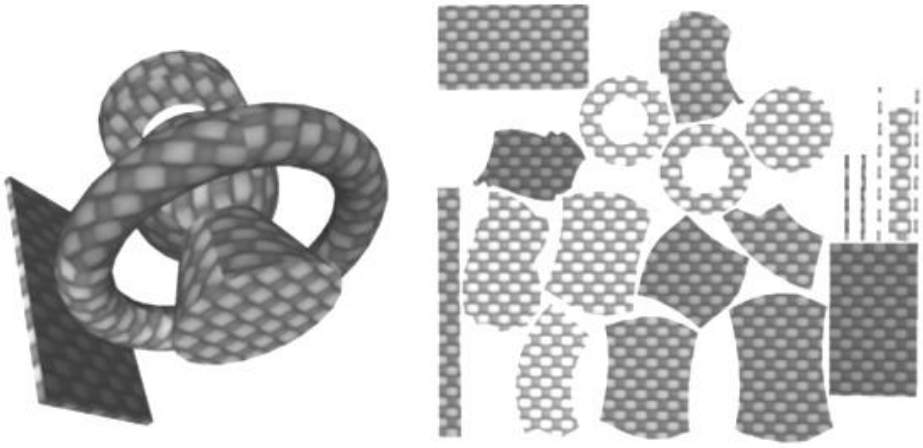


Fig.10 Regiones generadas utilizando UVAtlas

Como se muestra en la Figura 10 las coordenadas de textura se dividen en varios grupos o regiones. Generar coordenadas de texturas únicas de forma manual para cada región es demasiado complicado cuando aumenta la complejidad del modelo.

El API de UVAtlas genera automáticamente un mapa de la textura de forma tal que no haya solapamiento entre las regiones. También permite minimizar la expansión y distorsión de la textura. Además maximiza el empaquetamiento del espacio de la textura para hacer un uso eficiente de la memoria.

## 1.6 Proceso actual de generación de Texturas Procedurales de Pre-cálculo.

Utilizando programas de diseño 3D como el 3D Studio Max y el Maya se generan modelos tridimensionales desde muy simples hasta de gran complejidad geométrica. Estos modelos están compuestos por un conjunto de triángulos y a cada uno de ellos se le asigna un espacio de la textura mediante las coordenadas de textura.

Para generar Texturas Procedurales de Pre-cálculo es necesario que las coordenadas de texturas para cada triángulo del modelo tridimensional se encuentren distribuidas dentro del rango [0..1]. Esto se puede lograr de forma manual, es decir moviendo las coordenadas de textura de forma que todas se encuentren en el espacio definido, o se puede obtener de forma automática, utilizando programas que integran las funcionalidades de la librería UVAtlas o funciones parecidas a esta. Generalmente los programas de diseño 3D tienen integrados estos programas.

La generación de Texturas Procedurales de Pre-cálculo se realiza a partir del modelo tridimensional y sus coordenadas de texturas. En la Figura 11 se representa el proceso actual de generación de este tipo de texturas.

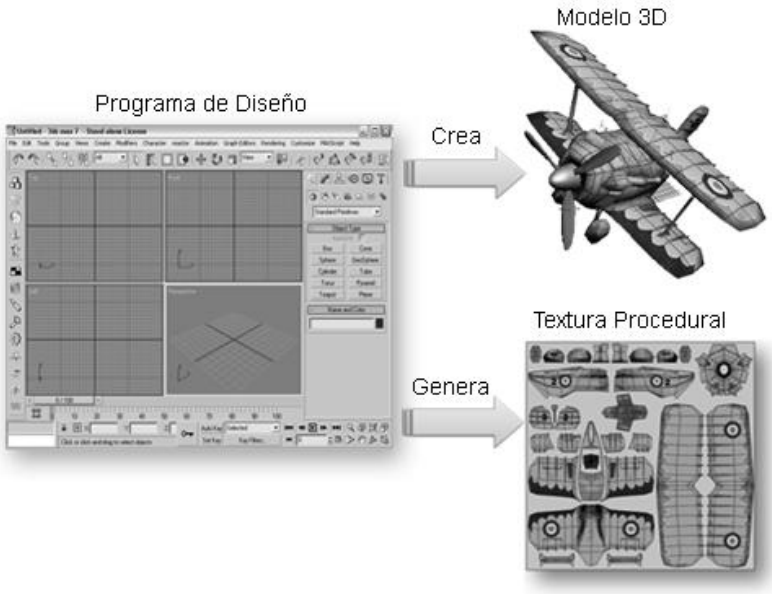



Fig.11 Proceso actual de generación de Texturas Procedurales de Pre-cálculo

## 1.7 Metodologías y herramientas de desarrollo.

Para el desarrollo de la aplicación se realizó un estudio sobre las posibles herramientas y metodologías a utilizar en su construcción. Se tuvo en cuenta la tendencia actual y las características de cada una de ellas y se realizó la selección descrita a continuación.

### 1.7.1 Metodología.

Se escogió RUP (Rational Unified Process, Proceso Unificado de Desarrollo de Software) como una guía para realizar el análisis y diseño de la aplicación debido a que es una metodología que ha probado su efectividad durante muchos años, ya que numerosos proyectos la han utilizado para desarrollar su software. Además esta metodología tiene un gran número de documentos publicados que es posible consultarlos para esclarecer dudas. También porque siguiendo los pasos que propone se obtiene una amplia documentación.

Entre las principales características que influyeron en la selección de esta metodología de desarrollo se encuentran los siguientes aspectos:

- Guiado por casos de uso: Pues constituye la guía fundamental establecida para las actividades a realizar durante todo el proceso de desarrollo incluyendo el diseño, la implementación y las pruebas del sistema.
- Centrado en arquitectura: Porque permite implementar la arquitectura del sistema y luego ir desarrollando cada uno de los módulos en dependencia de las necesidades.
- Iterativo e Incremental: Ya que facilita dividir el proyecto en ciclos y para cada ciclo se establecen fases de referencia. Permite una comprensión creciente de los requerimientos a la vez que crece el sistema.
- Utilización de un único lenguaje de modelado: UML.

### 1.7.2 Herramientas de desarrollo.

- Visual Studio 2005: Se escogió este compilador porque es rápido, tiene buena detección y corrección de errores, posee facilidad de trabajo con los elementos visuales y buena integración de estos con el código. Asimismo contiene muchas librerías con códigos pre-escritos que ayudan en el desarrollo del código de la aplicación. Se vincula con el Enterprise Architect para generar y

mantener actualizados diagramas como los de clases y de secuencia. Además posee numerosas herramientas asociadas que ayudan a escribir, analizar y distribuir el código.

- Enterprise Architec: Como herramienta de modelado con UML se seleccionó Enterprise Architect pues brinda una interfaz sencilla y cómoda para trabajar. Incluye todos los elementos necesarios para modelar el análisis y diseño de una aplicación, es decir, no hay que instalar varios paquetes para obtener todas sus funcionalidades. Se integra automáticamente con el Visual Studio 2005. Además permite combinar diferentes tipos de diagramas para explicar las ideas con mayor facilidad.
- Adobe Photoshop CS2: Para el tratamiento de las imágenes se escogió este programa pues constituye la herramienta estándar de edición de imágenes y líder de la gama de productos de edición de imágenes digitales. Tiene una interfaz amigable haciendo posible la creación y la modificación de imágenes de manera rápida y sencilla.
- 3D Studio Max 7: Este es uno de los programas más utilizados para generar Texturas Procedurales de Pre-cálculo, es por ello que se escogió como una guía para explicar el proceso de creación de estas texturas y realizar las pruebas necesarias para fundamentar la tesis.

## 1.8 Lenguajes.

Para desarrollar la aplicación se seleccionaron dos lenguajes: el C++ como lenguaje de programación y el UML como lenguaje de modelado.

### 1.8.1 Lenguaje de programación.

Se seleccionó el lenguaje C++ debido a que posee características superiores a otros lenguajes. Las más importantes son: programación orientada a objetos, portabilidad, programación modular y velocidad.

Además, se trata de un lenguaje de programación estandarizado (ISO/IEC 14882:1998), ampliamente difundido, y con una biblioteca estándar que lo ha convertido en un lenguaje universal, de propósito general y ampliamente utilizado tanto en el ámbito profesional como en el educativo. También es muy fácil de vincularlo con DirectX; librería que ha facilitado el desarrollo de la aplicación.

### 1.8.2 Lenguaje de modelado.

Para modelar el análisis y el diseño del software se escogió el lenguaje UML (Unified Modeling Language, Lenguaje Unificado de Modelado).

"UML es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software" (5). Se usa para entender, diseñar, configurar, mantener y controlar la información sobre los sistemas a construir. Además permite la modelación de sistemas con tecnología orientada a objetos.

# 2

# CAPÍTULO
# SOLUCIÓN PROPUESTA

En el presente capítulo se describe el flujo de trabajo propuesto para elevar la eficiencia de las Texturas Procedurales de Pre-cálculo o disminuir el consumo de memoria, según la necesidad del usuario. Además, se presentan las optimizaciones propuestas para ser integradas en el software y se determinan sus ventajas.

## 2.1 Flujo de Trabajo Iterativo

Para la creación y optimización de Texturas Procedurales de Pre-cálculo se propone un flujo de trabajo iterativo, mediante el cual una vez obtenida la textura se le adiciona la herramienta Optimizador de Texturas Procedurales de Pre-cálculo (LUTOR), con el propósito de mejorar el aprovechamiento de la textura generada.

El flujo de trabajo no altera la forma de crear Texturas Procedurales de Pre-cálculo, sino que incluye la herramienta LUTOR. Este proceso se representa en la Figura 12.

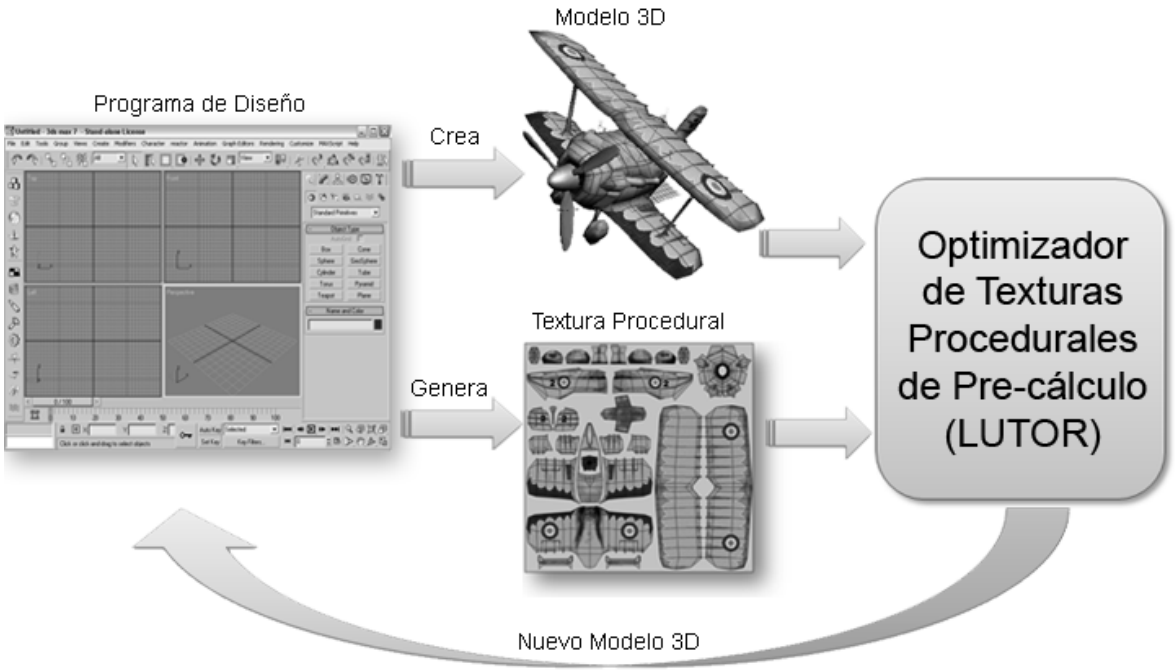



Fig.12 Proceso iterativo propuesto

El proceso comienza con la creación de un modelo tridimensional mediante el empleo de un programa de diseño 3D como el 3DStudio, el Maya, entre otros. Para esto existen diversas técnicas de modelado, que a partir de diferentes primitivas como cajas, planos, líneas, NURBS y aplicando diferentes modificadores, se le va dando forma al objeto hasta que finalmente se obtiene el modelo deseado.

Independientemente de la técnica de modelado que se utilice siempre se va obtener un objeto que va estar compuesto por polígonos. A estos polígonos se les aplican diferentes atributos que definen la superficie del modelo, tales como, la adición de texturas, colores, materiales, entre otros.

Un parámetro necesario para generar Texturas Procedurales de Pre-cálculo son las coordenadas de texturas adecuadas para el modelo, pues van a permitir asignar a cada polígono un espacio en la textura. Estas coordenadas se pueden obtener de forma manual o utilizando funciones como las de la librería UVAtlas de DirectX.

La Figura 13 muestra una de las formas de configurar los parámetros para generar coordenadas de texturas utilizando el 3D Studio Max.

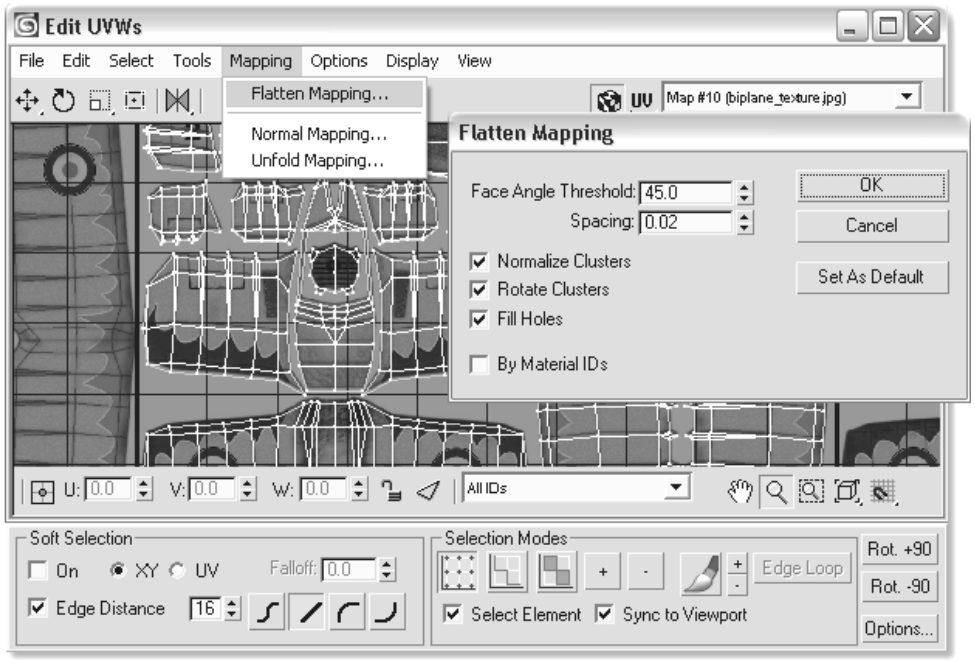



Fig.13 Generación de coordenadas de textura

Como se puede apreciar en la Figura 13, es posible configurar el tipo de mapeado o proyección a usar y de este definir las propiedades que parametrizan el modo en que se esparcirán los polígonos por el área de la textura.

Teniendo el modelo 3D con sus coordenadas de texturas correspondientes, se procede a la creación de la Textura Procedural de Pre-cálculo. Para explicar este proceso se toma como ejemplo el procedimiento que se ejecuta en el 3DStudio para obtener este tipo de textura.

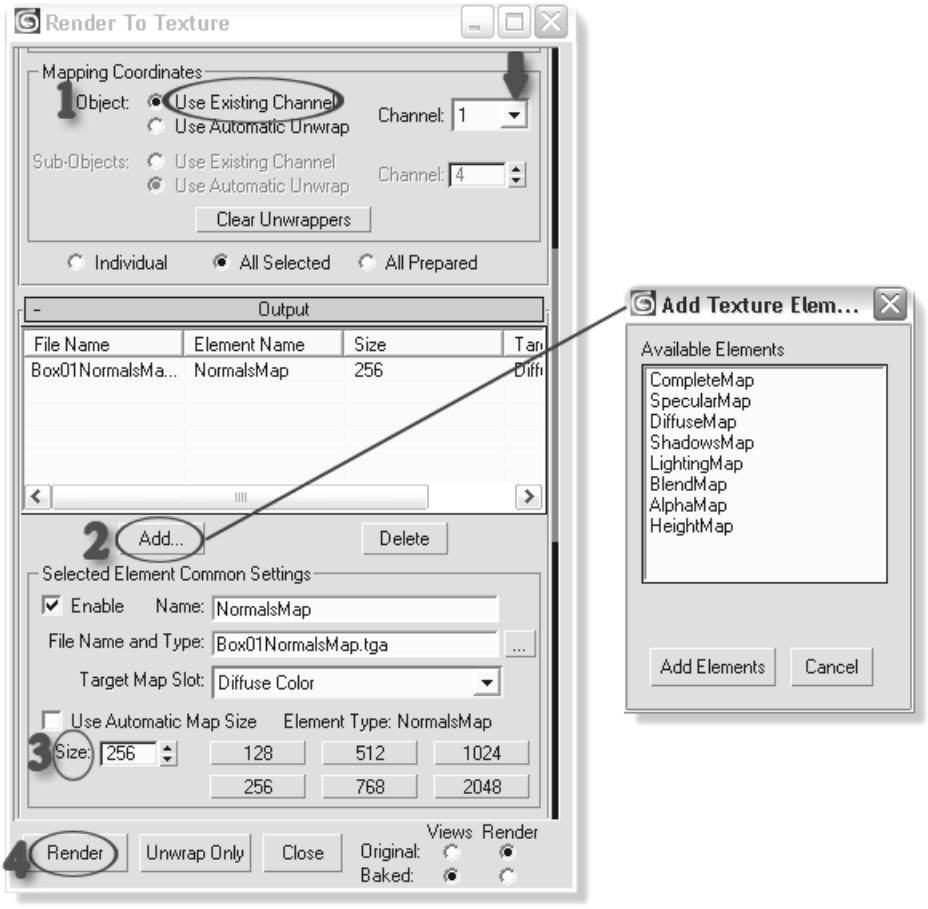



Fig.14 Creación de la Textura Procedural de Pre-cálculo

Como se muestra en la Figura 14 existen 4 pasos fundamentales:

1. Escoger el canal donde se encuentran las coordenadas de texturas generadas.

2. Adicionar el tipo de Textura Procedural de Pre-cálculo a generar.

3. Seleccionar el tamaño con que se quiere generar la Textura Procedural de Pre-cálculo adicionada.

4. Crear y mostrar la Textura Procedural de Pre-cálculo.

Es necesario destacar que la función Render to Texture puede generar de forma automática coordenadas de texturas únicas. Para esto en lugar de indicar un canal existente se debe seleccionar la generación automática de coordenadas de texturas (Use Automatic Unwrap). Sin embargo, esta solución no permite configurar los parámetros para obtener coordenadas de texturas acorde a las necesidades del usuario.

La aplicación propuesta para integrar el proceso de generación de Texturas Procedurales de Pre-cálculo se utiliza para elevar la eficiencia de este tipo de texturas. Para esto, se recomienda que el usuario genere la textura a optimizar con el mayor tamaño posible.

La Textura Procedural de Pre-cálculo obtenida en el proceso descrito se exporta junto con el modelo 3D y ambos se cargan en la herramienta LUTOR. Esta herramienta permite escoger las optimizaciones que se requieran aplicar y exportar el modelo con las modificaciones efectuadas en las coordenadas de texturas.

El proceso finaliza cuando el modelo modificado es cargado nuevamente en el programa que fue creado y con las nuevas coordenadas de texturas se vuelve a generar la Textura Procedural de Pre-cálculo. El usuario determina, según sus necesidades, si reduce el tamaño de la textura optimizada (lo cual es equivalente a una reducción de memoria) o si quiere mantener el tamaño original para abarcar mayor cantidad de detalles.

Si se desea optimizar una vez más la Textura Procedural de Pre-cálculo, se puede repetir este proceso. Siendo posible repetirlo tantas veces como se quiera, hasta obtener una textura con la eficiencia requerida por el artista.

## 2.2 Requerimientos de los datos de entrada al programa.

LUTOR tiene dos entradas fundamentales: un modelo tridimensional o mesh y la Textura Procedural de Pre-cálculo a optimizar. Cada entrada tiene que cumplir ciertos requisitos para que el software pueda proporcionar un resultado válido. A continuación se listan dichos requerimientos.

- El modelo 3D que necesita cargar:
  - ✓ Debe existir.
  - ✓ Debe estar compuesto por más de un polígono.

- ✓ Debe contener al menos un canal de coordenadas de texturas.

- Cada polígono del modelo a cargar:
  - ✓ Debe tener exactamente 3 lados (debe ser un triángulo).
  - ✓ Debe tener asignado coordenadas de textura.
- La Textura Procedural de Pre-cálculo a importar:
  - ✓ Debe existir
  - ✓ Sus dimensiones (ancho y alto) deben ser potencias de 2.
  - ✓ Cada texel de la textura debe estar almacenado en 24 ó 32 bits, coincidiendo con los componentes RGB o RGBA.
  - ✓ Debe estar almacenada en un formato para el cual haya algún exportador en el sistema (tda, bmp, dds, tpg, png, dib, hdr, pfm y ppm).

El diagrama de la Figura 15 representa una forma sencilla para comprender los requerimientos de los que depende el software para su correcto funcionamiento.

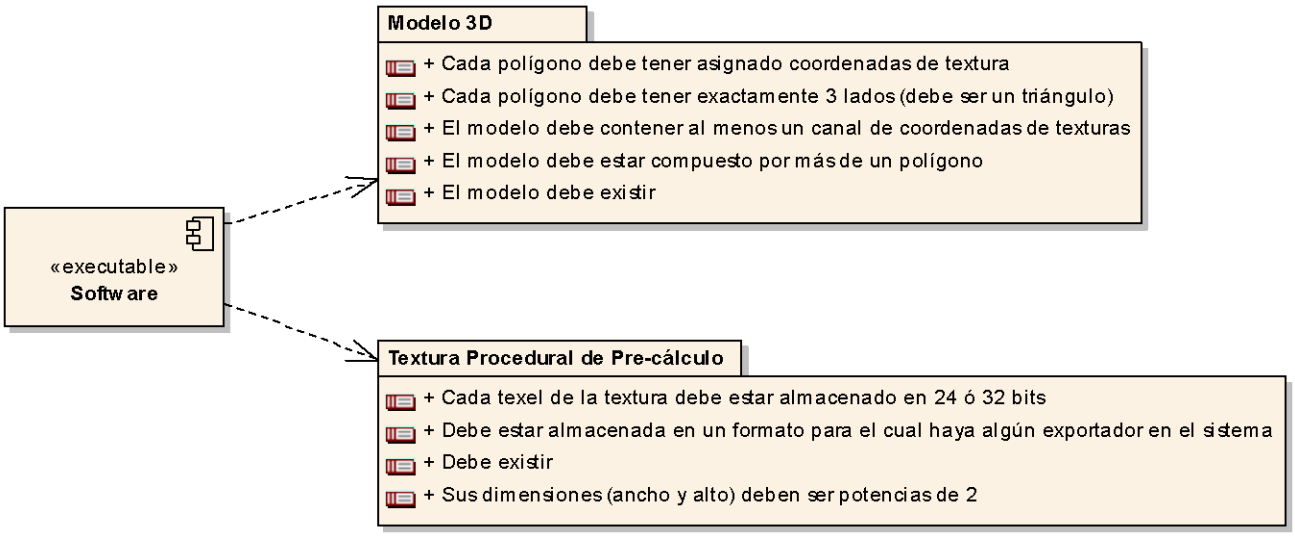



Fig.15 Requerimientos de entrada

## 2.3 Recursos manejados por la aplicación

Para modificar las coordenadas de textura del modelo tridimensional, el Procesador tiene en cuenta la forma en que este es mapeado en la textura. Además utiliza las configuraciones almacenadas en un fichero, las cuales son establecidas por defecto o creadas por el usuario durante la ejecución de la aplicación. El modelo resultante que contiene las modificaciones es exportado a un fichero seleccionado por el usuario. (Ver Figura 16).

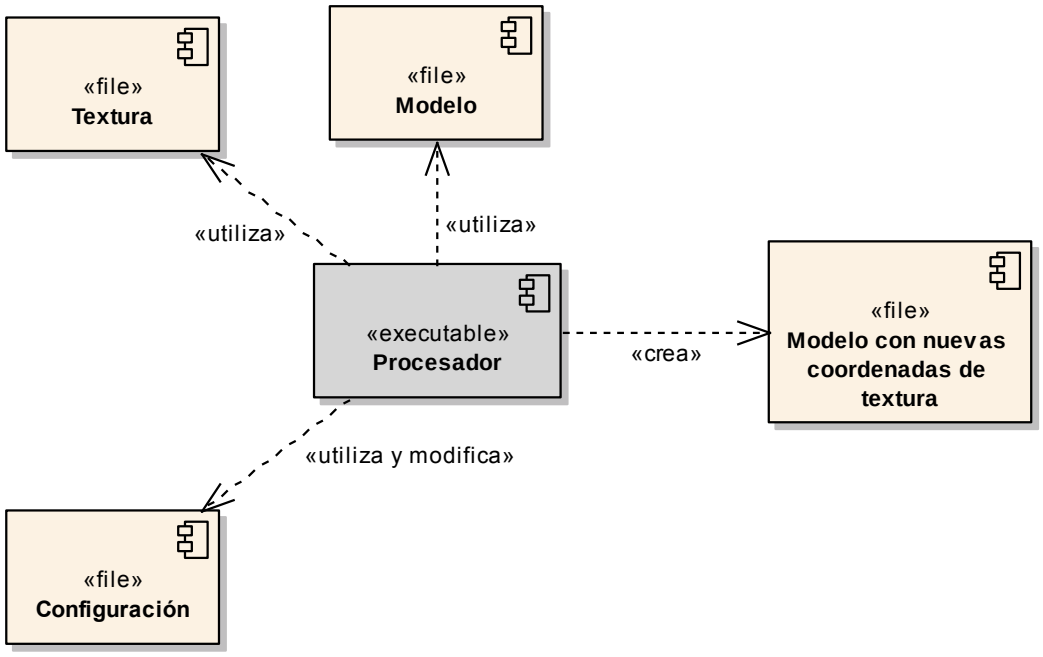



Fig.16 Recursos manejados por la aplicación

## 2.4 Optimizaciones

La solución propuesta incluye cinco optimizaciones a implementar en el software. La primera versión del producto ha integrado tres de estas optimizaciones y en futuras versiones se irán integrando las demás junto con algoritmos que perfeccionen las ya existentes. Para permitir esta integración, el software contiene un sistema de Plugines que permiten adicionar nuevas funcionalidades sin alterar el Framework actual.

En los siguientes epígrafes se explican las optimizaciones propuestas, señalando sus ventajas y ejemplificando cómo debe ser su funcionamiento. Para esto se utilizó un ejemplo sencillo, una pirámide modelada en 3D Studio Max que ha sido mapeada con una textura de diferentes colores. (Ver Figura 17).

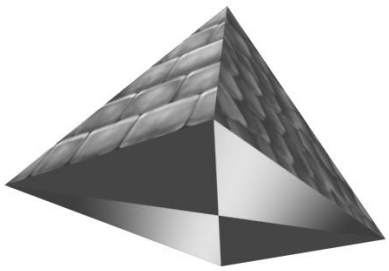


Fig.17 Pirámide texturizada

### 2.4.1 Colapsar polígonos de un solo color

Esta optimización detecta los polígonos que utilizan un solo color, es decir, determina los polígonos que todos sus texeles tienen igual color y los reduce a un solo texel. De esta forma el polígono será representado completamente con el color de este texel.

La optimización es de gran utilidad esta cuando se necesitan representar muchos polígonos que estén compuestos por un único color. Por ejemplo, cuando en un Mapa de Luces se tiene una zona con una luz intensa (que se representa por el color blanco) o una total oscuridad (que se representa con el color negro), se ahorra gran espacio en la textura. Este espacio puede ser utilizado para representar polígonos que presenten una complejidad mayor (que contengan varios colores) y de esta forma lograr una visión más detallada de los mismos.

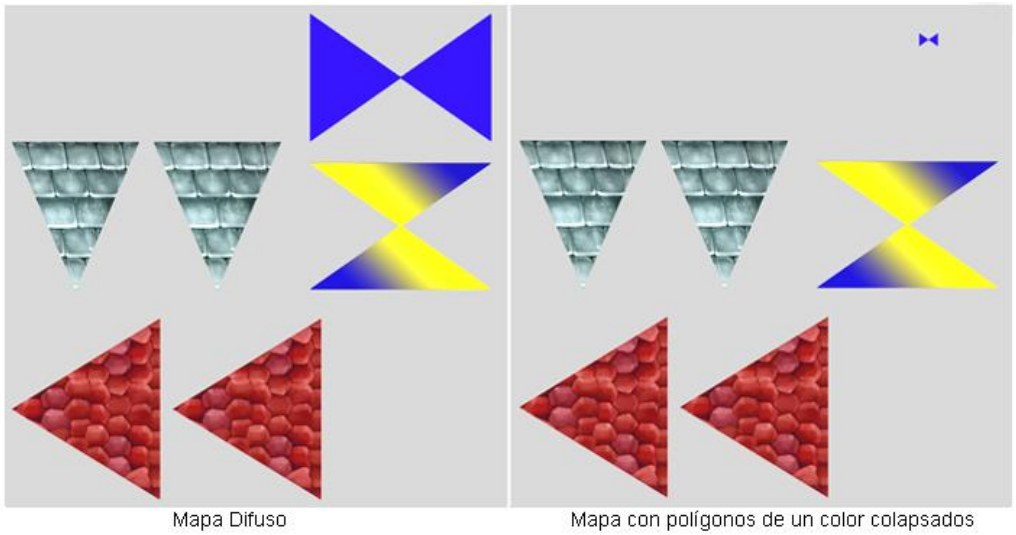



Fig.18 Colapsar polígonos de un solo color

La Figura 18 muestra el resultado del Mapa Difuso al aplicarle esta optimización. Como se puede apreciar se obtiene un espacio significativo sin utilizar, que puede ser empleado para representar polígonos más complejos.

### 2.4.2 Colapsar polígonos iguales

Es posible que existan varios polígonos que contengan igual color, pero como se le asigna a cada polígono un espacio en la textura, entonces se representa la misma información varias veces. Esta optimización detecta dichos polígonos y los representa solamente una vez. Para ello le asigna a cada polígono las mismas coordenadas de textura.

Por ejemplo, en la Figura 19 existen ocho polígonos, de los cuales, dos pares de ellos son iguales, al aplicarle esta optimización le asignó solamente espacio en la textura a un polígono de cada par. Esto se logra tomando un polígono y buscando si hay polígonos iguales a él, en caso de encontrarlos, entonces iguala las coordenadas de texturas de ambos.

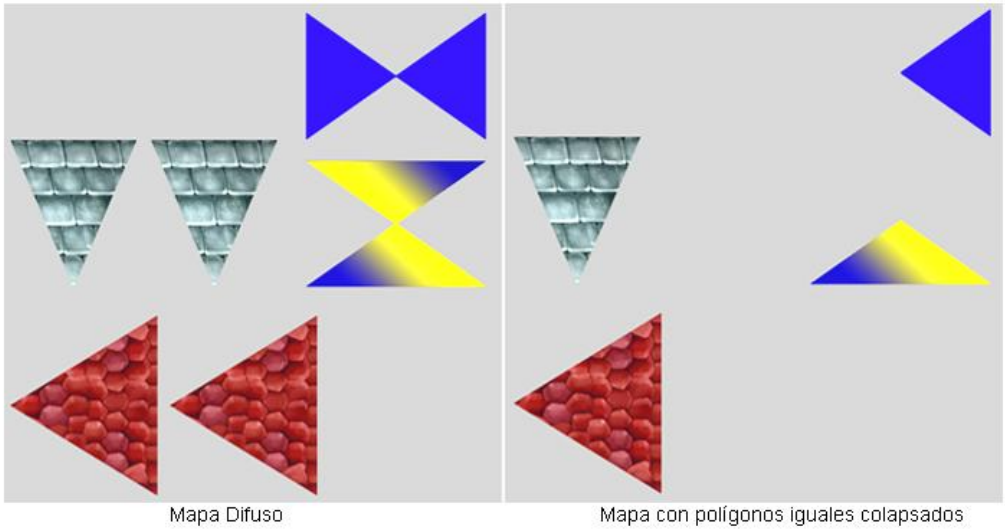



Fig.19 Colapsar polígonos iguales

### 2.4.3 Escalar polígonos de acuerdo a la complejidad

Un polígono es denominado complejo cuando su espacio correspondiente en la textura contiene varios colores, como los ruidos y las imágenes coloridas. Es decir, la complejidad de un polígono viene dada por la cantidad de transiciones de color que se necesite para representarlo.

Constituye un error pensar que el espacio de la textura se debe repartir utilizando como único criterio la complejidad geométrica de los polígonos. Debido a que pueden existir polígonos de grandes dimensiones con un único color, los cuales solo necesitan un texel de la textura para ser representados. A su vez pueden existir polígonos de menor tamaño pero de mayor complejidad y para representarlos de forma detallada se necesita un mayor espacio en la textura.

Esta optimización resuelve la problemática anterior, analiza cuales polígonos deben obtener mayor área en la textura y le asigna el espacio conforme a este análisis. En la Figura 20 se muestra un ejemplo en el que se escalan uniformemente los polígonos representados por varios colores. Es importante tener en cuenta que la optimización cobraría mayor fuerza si se aplicara luego de haber colapsado los polígonos de un solo color y los polígonos iguales.

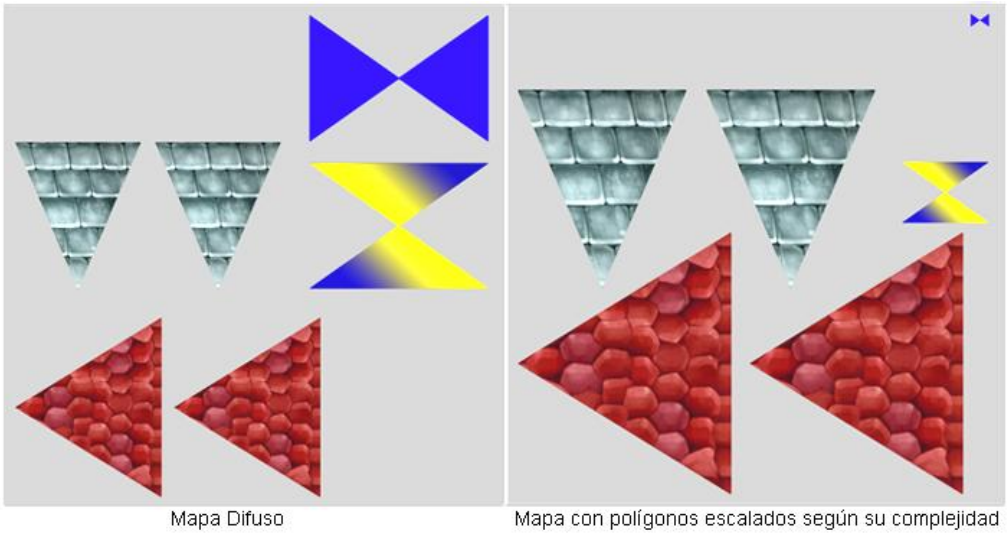



Fig. 20 Escalar polígonos de acuerdo la complejidad

### 2.4.4 Soldar polígonos

Normalmente se acomodan los polígonos sin tener en cuenta la similitud de los colores que los representan; lo cual trae como consecuencia que si están muy unidos, el color que se forma en los bordes no es el deseado. Esto ocurre porque a la hora de tomar un texel de la textura no se selecciona exactamente dicho texel, sino también parte de sus alrededores; debido a las funciones de filtrado de textura que realiza el hardware gráfico para mejorar la calidad visual de las escenas virtuales y disminuir la

texelación al aplicar las texturas. De esta forma se hace un gradiente desde el color del texel hasta el color de sus alrededores y así se evita que una vez rendereado el polígono se vea pixelado.

Para que no suceda lo antes explicado se establece un espacio considerable entre un polígono y otro. Pero, en el caso de que los bordes de los polígonos sean iguales se dejaría un espacio sin ninguna necesidad. La optimización soldar polígonos resuelve esta situación detectando los polígonos de igual borde y uniéndolos.

### 2.4.5 Reacomodar polígonos

Como su nombre lo indica esta optimización reacomoda las regiones en la textura de tal forma que se obtenga una mejor distribución de las coordenadas de textura. Además escala todas las regiones de manera que se aproveche el espacio de textura.

En la Figura 21 se muestra un ejemplo donde se puede apreciar que esta optimización posibilita escalar los polígonos, logrando que en la representación de cada uno se utilicen más texeles. De esta forma se pierde menos espacio en la textura y la calidad visual del objeto texturizado es mayor, ya que puede contener más detalles.

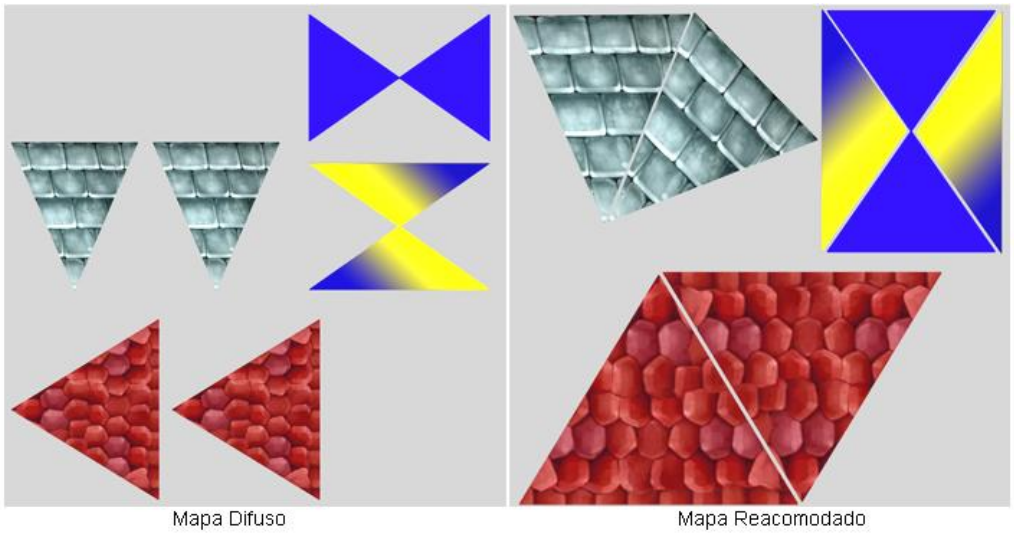



Fig.21 Reacomodar polígonos

Las optimizaciones descritas en los sub-epígrafes anteriores tratan de ganar espacio en la textura, sin embargo la más importante es reacomodar polígonos, pues distribuye las coordenadas de textura de forma tal que se obtenga como resultado una textura con la mayor cantidad de texeles con información utilizable. Es por ello que se recomienda que siempre sea utilizada esta optimización, aplicándola tanto sola como en combinación con el resto de las optimizaciones.

# 3

# CAPÍTULO
# FUNCIONAMIENTO DEL SISTEMA

En este capítulo se realiza una detallada descripción del funcionamiento del sistema de forma conceptual. En él se muestran los conceptos necesarios para el desarrollo de la solución, se describen las restricciones a considerar para modelar la misma, así como las funcionalidades del software. Además se definen los actores que interactúan con el sistema y se presentan los casos de uso con sus correspondientes descripciones.

## 3.1 Captura de Requisitos

La aplicación deberá cumplir con ciertos requisitos para su correcto funcionamiento. A continuación se presentan dichos requisitos clasificados en Funcionales y no Funcionales.

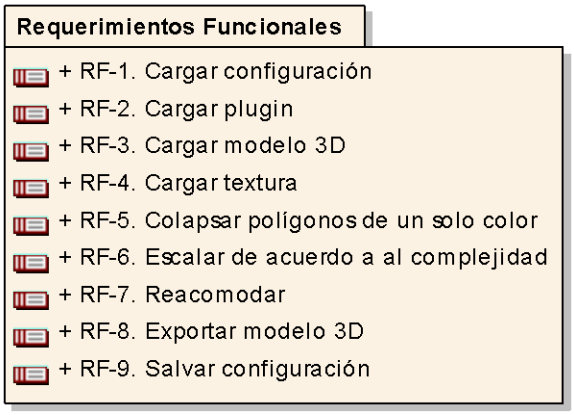



Fig.22 Requerimientos Funcionales

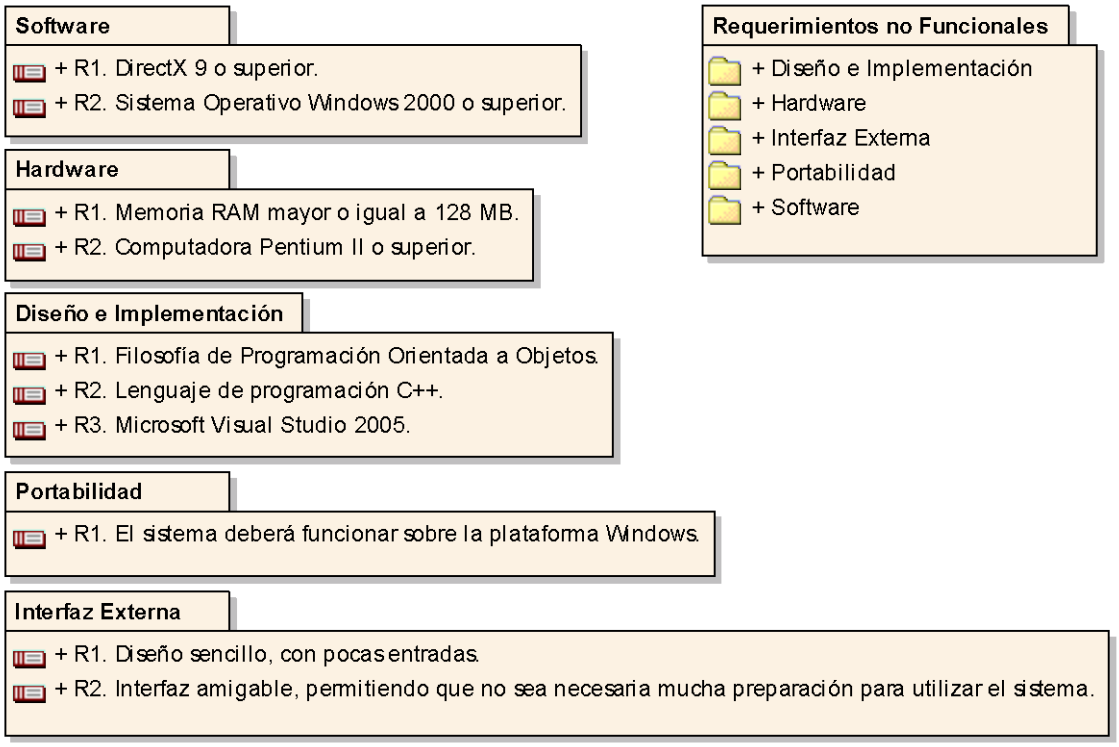



Fig.23 Requerimientos no Funcionales.

## 3.2 Casos de Uso del Sistema

A partir de los requerimientos funcionales definidos en la captura de requisitos se obtienen los casos de uso del sistema. En la figura 24 se muestran agrupados por paquete para facilitar la comprensión de los mismos.

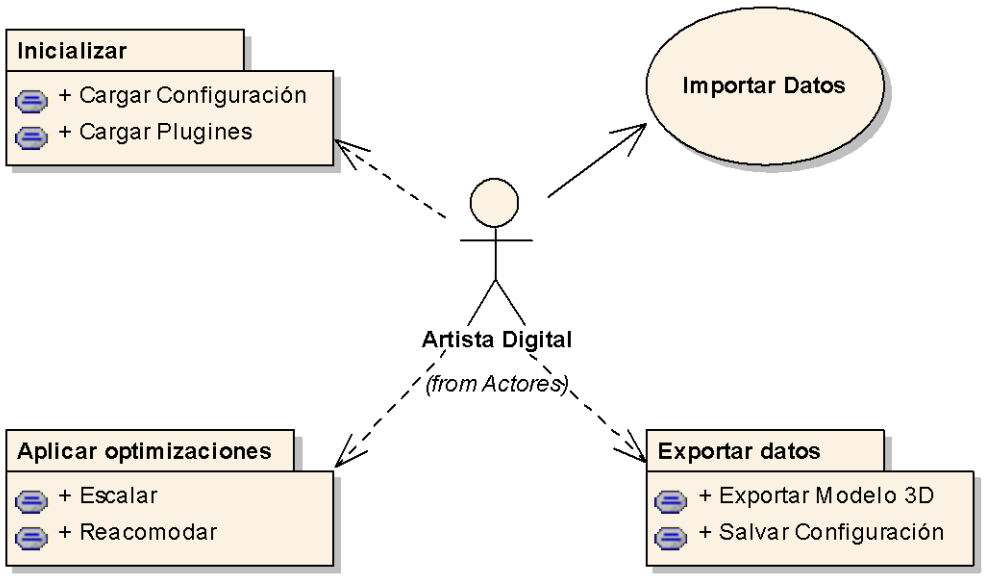



Fig.24 Diagrama de Casos de Uso del Sistema por paquetes

Las Figuras 25, 26 y 27 muestran el diagrama de casos de uso de cada uno de los paquetes especificados.

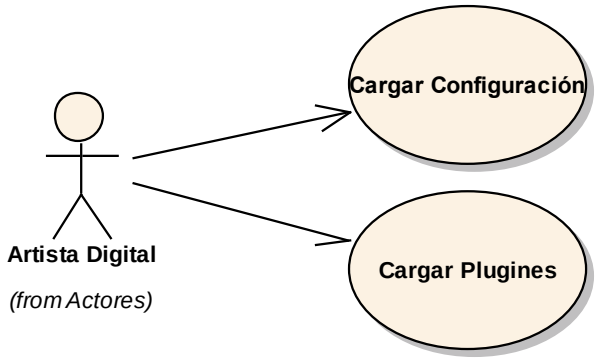



Fig.25 Paquete Inicializar

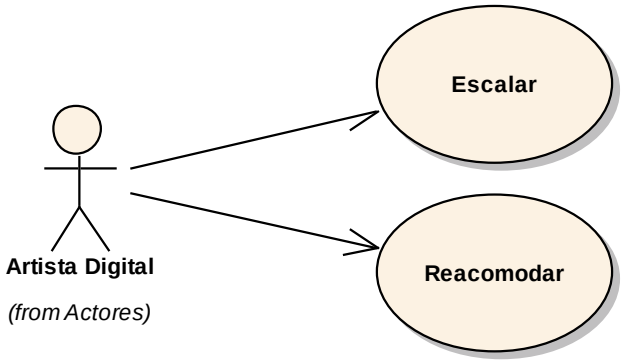



Fig.26 Paquete Aplicar optimización

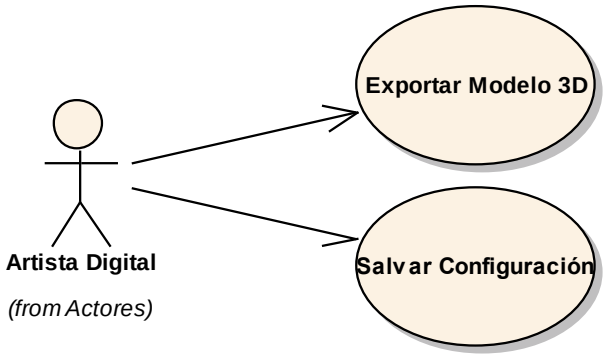



Fig.27 Paquete Exportar Datos

## 3.3 Definición de actores del sistema

| **Actores** | **Justificación** |
|---|---|
| Artista Digital | Es quién interactúa con el sistema con el objetivo de elevar la eficiencia de la Textura Procedural de Pre-cálculo correspondiente a un modelo 3D. |

Tabla 1: Definición de actores del sistema

## 3.4 Descripción textual de los Casos de Uso del Sistema

| **Nombre del Caso de Uso** | Cargar Configuración |
|---|---|
| **Actores** | Artista Digital |
| **Propósito** | El objetivo de este caso de uso es cargar la configuración que el sistema va a utilizar. |
| **Resumen** | El sistema busca y carga el fichero de configuración. |
| **Referencias** | RF-1 |
| **Precondiciones** | --- |
| **Poscondiciones** | --- |
| **Curso Normal de los Eventos** | |
| **Acciones del Actor** | **Respuesta del Sistema** |
| 1. El artista digital ejecuta la aplicación. | 1.1 El sistema busca el fichero de configuración.<br>1.2 El sistema carga el fichero de configuración. |

| **Curso Alterno de los Eventos** | |
|---|---|
| Acción 1 | 1.1 Si el sistema no encuentra el fichero de configuración, carga la configuración que tiene por defecto y crea un nuevo fichero de configuración. |

Tabla 2: Descripción del caso de uso Cargar Configuración

| **Nombre del Caso de Uso** | Cargar Plugines |
|---|---|
| **Actores** | Artista Digital |
| **Propósito** | El objetivo de este caso de uso es cargar los Plugines que contienen las diferentes optimizaciones que se pueden aplicar, así como las extensiones de los ficheros que el sistema reconoce en el momento de importar y exportar. |
| **Resumen** | El sistema busca y carga los diferentes Plugines. Esto se realiza a través de las API con que cuenta la aplicación. |
| **Referencias** | RF-2 |
| **Precondiciones** | --- |
| **Poscondiciones** | Tiene que estar cargado al menos un Plugin que permita cargar y exportar. Además tiene que haberse cargado al menos una optimización. |
| **Curso Normal de los Eventos** | |
| **Acciones del Actor** | **Respuesta del Sistema** |
| 1. El artista digital ejecuta la aplicación. | 1.1 El sistema busca los Plugines internos.<br>1.2 El sistema carga los Plugines. |

Tabla 3: Descripción del caso de uso Cargar Plugines

| **Nombre del Caso de Uso** | Importar Datos |
|---|---|
| **Actores** | Artista Digital |
| **Propósito** | El objetivo de este caso de uso es cargar el modelo tridimensional y la textura con el que el sistema trabajará. |
| **Resumen** | El caso de uso inicia cuando el artista digital selecciona la ubicación del modelo y la textura a importar y finaliza cuando el sistema importa ambos ficheros. |
| **Referencias** | RF-3, RF-4 |
| **Precondiciones** | --- |
| **Poscondiciones** | Tienen que haberse cargado el modelo tridimensional y la textura. |
| **Curso Normal de los Eventos** | |
| **Acciones del Actor** | **Respuesta del Sistema** |
| 1. El artista digital procede a importar el modelo 3D. | 1.1 El sistema muestra una ventana de búsqueda donde filtra los ficheros según los formatos que puede importar. |
| 2. Selecciona la ubicación del modelo. | 2.1 Muestra la ubicación seleccionada. |
| 3. El artista digital busca la textura a importar. | 3.1 El sistema muestra una ventana de búsqueda donde filtra los ficheros según los formatos que puede importar. |
| 4. Selecciona la ubicación de la textura | 4.1 Muestra la ubicación seleccionada. |
| 5. Ejecuta el próximo paso del asistente. | 5.1 El sistema carga el modelo y la textura seleccionada. |
| **Curso Alterno de los Eventos** | |
| Acción 5 | 5.1 En caso de que no pueda cargar el modelo o la textura le informa al usuario mediante un mensaje. |

Tabla 4: Descripción del caso de uso Cargar Modelo 3D

| **Nombre del Caso de Uso** | Escalar |
|---|---|
| **Actores** | Artista Digital |
| **Propósito** | El objetivo de este caso de uso es asignarle un mayor espacio de textura a los polígonos más complejos y reducírselo a los de menor complejidad. |
| **Resumen** | El sistema determina cuáles son los polígonos más complejos y los escala. |
| **Referencias** | RF-5, RF-6 |
| **Precondiciones** | El sistema debe tener seleccionada la optimización Escalar. |
| **Poscondiciones** | --- |
| **Curso Normal de los Eventos** | |
| **Acciones del Actor** | **Respuesta del Sistema** |
| 1. Manda a ejecutar las optimizaciones. | 1.1 El sistema ejecuta la optimización Escalar<br>a) Para cada polígono del modelo 3D analiza la complejidad.<br>b) En correspondencia con la complejidad del polígono, aumenta uniformemente el espacio de textura asignado a este. |

Tabla 5: Descripción del caso de uso Escalar

| **Nombre del Caso de Uso** | Reacomodar |
|---|---|
| **Actores** | Artista Digital |
| **Propósito** | El objetivo de este caso de uso es reacomodar las coordenadas de texturas de forma eficiente. |
| **Resumen** | El sistema determina cuál es la mejor ubicación para cada una de las regiones y luego escala uniformemente los polígonos para aprovechar el espacio de textura. |

| | |
|---|---|
| **Referencias** | RF-7 |
| **Precondiciones** | El sistema debe tener seleccionada la optimización Reacomodar. |
| **Poscondiciones** | --- |
| **Curso Normal de los Eventos** | |
| **Acciones del Actor** | **Respuesta del Sistema** |
| 1. Manda a ejecutar las optimizaciones. | 1.1 El sistema ejecuta la optimización Reacomodar.<br>a) Analiza en qué posición pueden acomodarse las regiones para que ocupe el menor espacio de textura posible.<br>b) Acomoda las regiones teniendo en cuenta los parámetros establecidos por el Artista Digital con anterioridad. |

Tabla 6: Descripción del caso de uso Reacomodar

| | |
|---|---|
| **Nombre del Caso de Uso** | Exportar Modelo 3D |
| **Actores** | Artista Digital |
| **Propósito** | El objetivo de este caso de uso es exportar el modelo tridimensional resultante. |
| **Resumen** | El caso de uso inicia cuando el artista digital selecciona la ubicación donde desea exportar el modelo resultante y finaliza cuando el sistema exporta el fichero. |
| **Referencias** | RF-8 |
| **Precondiciones** | Debe tener un modelo cargado en el sistema. |
| **Poscondiciones** | --- |
| **Curso Normal de los Eventos** | |

| **Acciones del Actor** | **Respuesta del Sistema** |
|---|---|
| 1. El artista digital procede a exportar el modelo 3D. | 1.1 El sistema muestra una ventana de búsqueda donde filtra los ficheros dependiendo de los formatos que puede exportar. |
| 2. Selecciona la ubicación donde desea exportar el modelo. | 2.1 Muestra la ubicación seleccionada. |
| 3. Ejecuta el próximo paso del asistente. | 3.1 El sistema exporta el modelo en la ubicación establecida. |

Tabla 7: Descripción del caso de uso Exportar Modelo 3D

| **Nombre del Caso de Uso** | Salvar Configuración |
|---|---|
| **Actores** | Artista Digital |
| **Propósito** | El objetivo de este caso de uso es salvar la configuración que el artista digital haya especificado, para ser utilizada posteriormente. |
| **Resumen** | El caso de uso inicia cuando el artista digital decide salvar las configuraciones existentes. El sistema procede a guardar la configuración y finaliza el caso de uso. |
| **Referencias** | RF-9 |
| **Precondiciones** | Debe tener un modelo cargado en el sistema. |
| **Poscondiciones** | --- |
| **Curso Normal de los Eventos** | |
| **Acciones del Actor** | **Respuesta del Sistema** |
| 1. El usuario selecciona la opción Salvar Configuración o ejecuta la acción de salir de la aplicación. | 1.1 Almacena en el fichero “CONFIGURATION.LUTOR” las configuraciones existentes. |

| **Curso Alterno de los Eventos** | |
|---|---|
| Acción 1 | 1.1 En caso de que el fichero no exista, crea un nuevo fichero con el nombre “CONFIGURATION.LUTOR” y almacena en él las configuraciones existentes. |

Tabla 8: Descripción del caso de uso Guardar Configuración

## 3.5 Modelo Conceptual

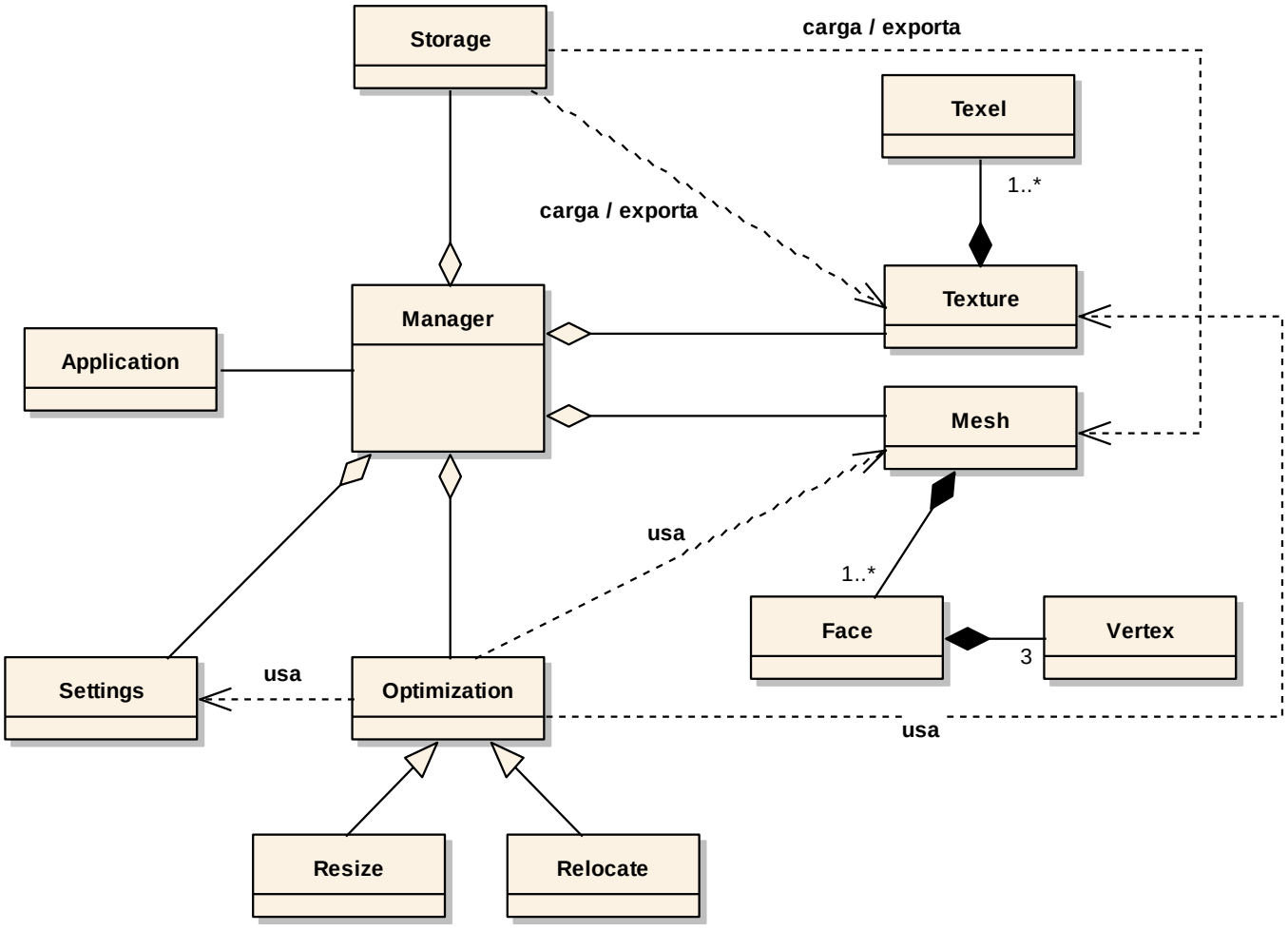



Fig.28 Modelo Conceptual del Sistema

El modelo que se muestra en la Figura 28 refleja la relación entre los conceptos involucrados en el desarrollo de la aplicación. Estos conceptos constituyen la base de la arquitectura del sistema, la cual se encuentra descrita en el siguiente epígrafe.

### 3.6 Arquitectura del Sistema

En la Figura 29 se muestran las clases principales que conforman el sistema, empleando para ello un modelo de análisis. A este conjunto de clases se le denominará “Framework”. El Framework constituye la estructura base del programa, la plataforma sobre la que se implementa el soporte para todas las funcionalidades del software.

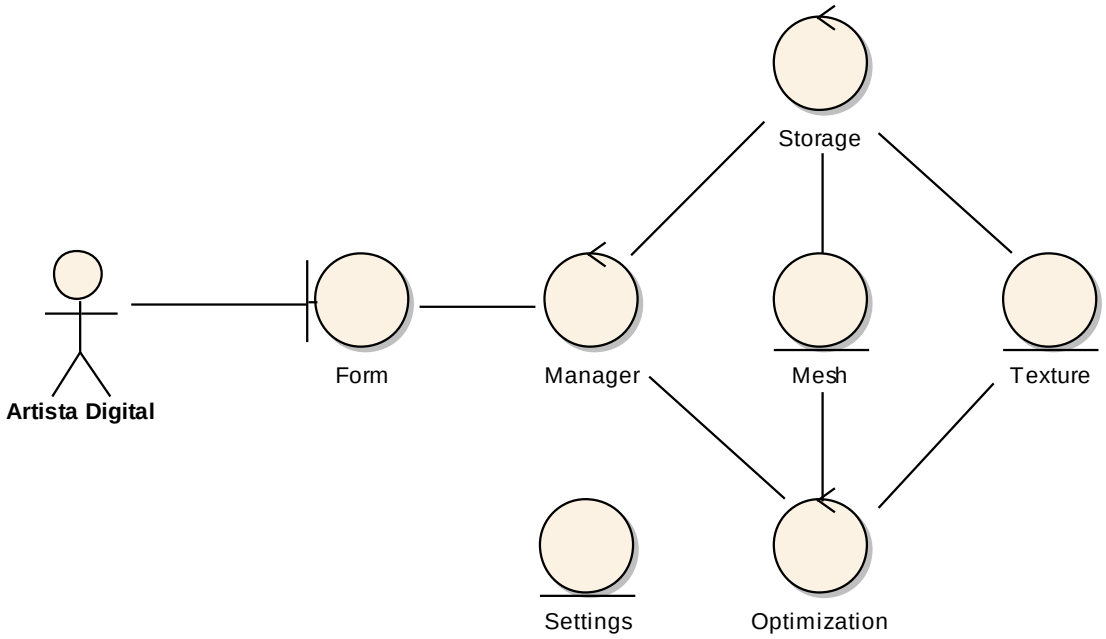



Fig.29 Modelo de análisis

El usuario interacciona con el Framework a través de la clase interfaz “Form”. Esta clase es la encargada de recibir todas las peticiones hechas por el artista y de transmitírselas a la clase controladora principal “Manager”. Esta última clasifica las peticiones y en dependencia de la petición, asigna tareas a las demás clases controladoras.

De esta forma, si el usuario desea cargar una textura o un modelo, el Manager le envía a Storage la petición, pero si lo que quiere es hacer alguna optimización en específico se la envía a Optimization, para que esta a su vez realice los cambios necesarios en Mesh y Texture.

## 3.7 Funcionamiento del Framework

A continuación se presenta el flujo de acciones que genera cada una de las peticiones principales que puede hacer el artista digital a través de la clase interfaz. Para explicar este flujo de acciones se determinó realizar un diagrama de secuencia por cada petición que pueda recibir la clase controladora principal.

### 3.7.1 Petición Inicializar

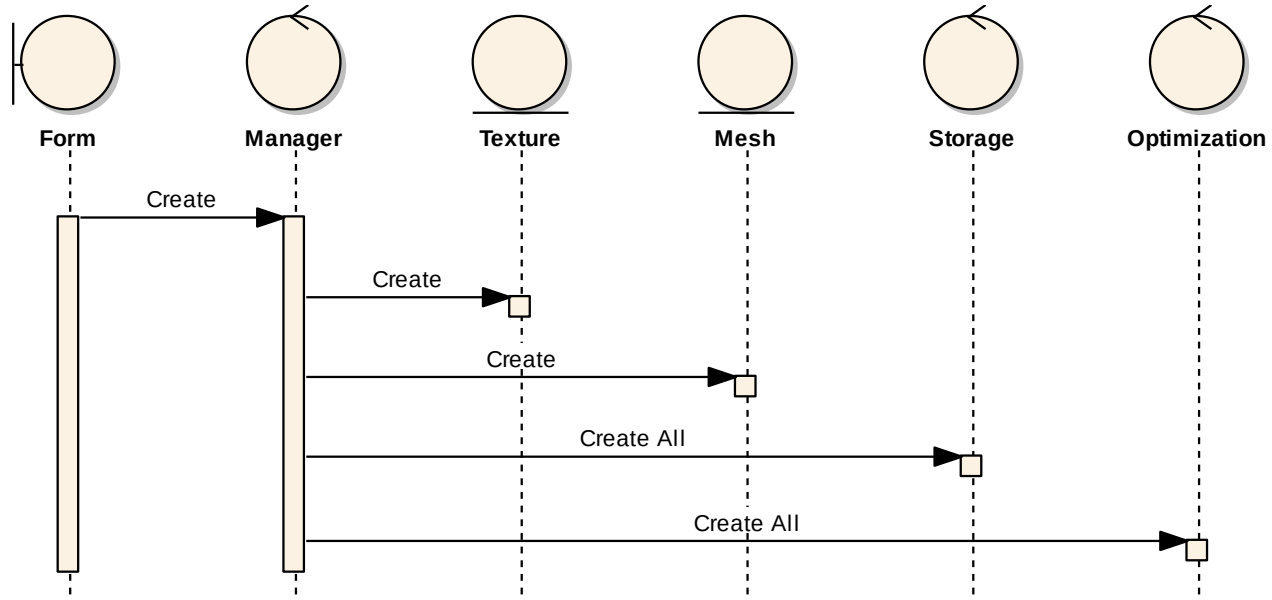



Fig.30 Diagrama de secuencia de la petición Inicializar

Cuando se ejecuta la aplicación se crea el Manager, el cual carga todos los Storages, así como todas las Optimizations, además se crean Mesh y Texture. (Ver Figura 30).

El principal objetivo por el cual son creados Mesh y Texture es para insertarle los datos del modelo y de la textura a importar, es decir, cuando se importa un modelo, se extraen los datos del mismo y se insertan en el Mesh que creó la aplicación cuando se inicializó. Un proceso similar al anterior ocurre con la textura importada.

Los Storages contienen la información necesaria para cargar tanto modelos como texturas y exportar los modelos resultantes. De esta forma el sistema funciona de forma modular, ya que es posible agregar al sistema cualquier otro importador/exportador sin necesidad de modificar el resto de las clases del sistema.

Las Optimizations funcionan de de forma similar a la de los Storages. Se cargan todas las Optimizations que son las encargadas de procesar la Textura Procedural de Pre-cálculo y el modelo con el objetivo de aumentar la eficiencia de la textura importada.

### 3.7.2 Petición Cargar Modelo

Cuando el Manager recibe la petición de Cargar Modelo, este verifica si alguno de sus Storages puede cargar el formato del fichero especificado. En caso de tener algún Storage que pueda cargar ese formato, le asigna la realización de la petición. Este Storage busca los datos que necesita del fichero y los inserta en Mesh. (Ver Figura 31).

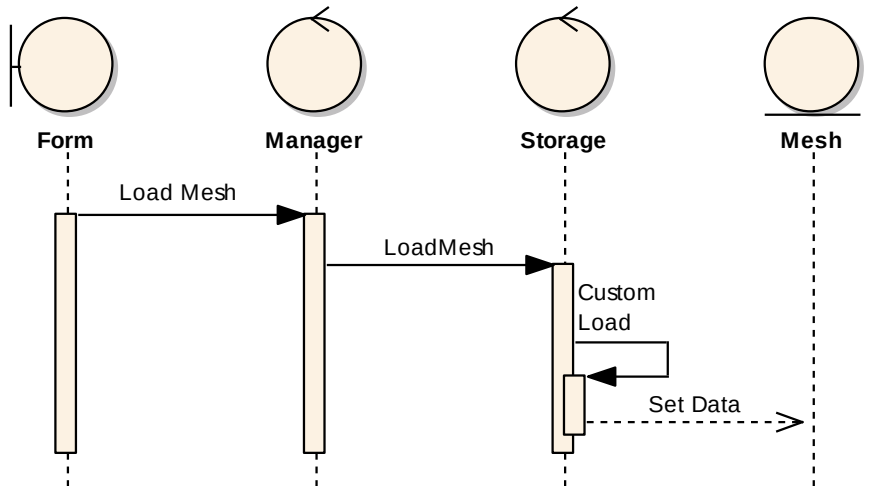



Fig.31 Diagrama de secuencia de la petición Cargar Modelo

### 3.7.3 Petición Cargar Textura

Una vez que al Manager le envían una petición de Cargar Textura, este verifica si alguno de sus Storages puede cargar el formato de la textura especificada. En caso de tener algún Storage que pueda cargar ese formato, le asigna la realización de la petición. Este Storage obtiene los datos que necesita del fichero especificado y los inserta en Texture. (Ver Figura 32).

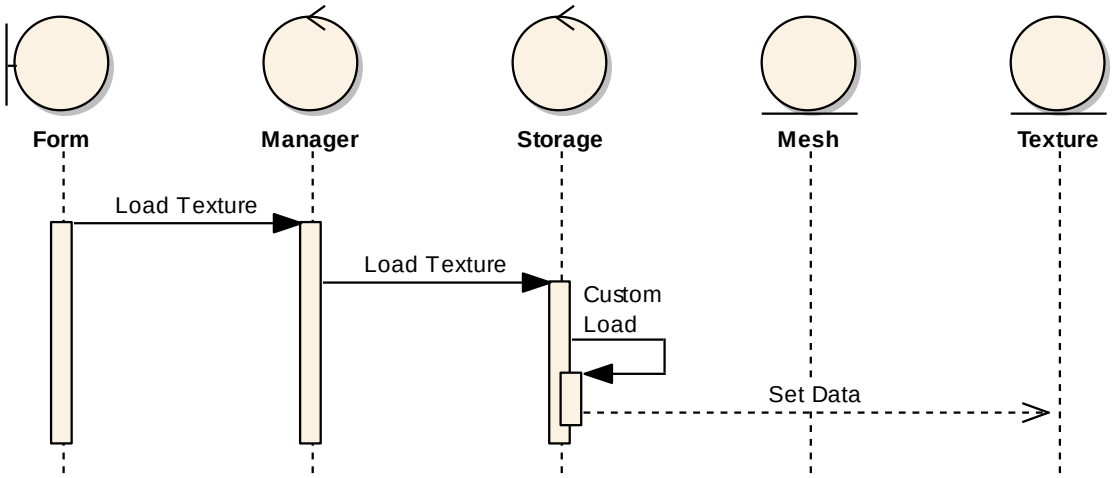



Fig.32 Diagrama de secuencia de la petición Cargar Textura

### 3.7.4 Petición Optimizar

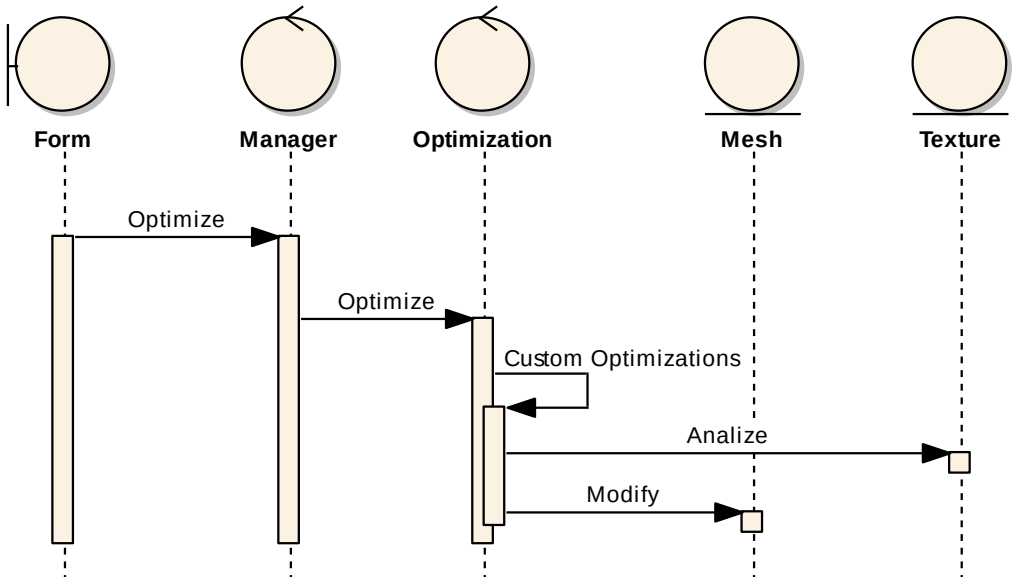



Fig.33 Diagrama de secuencia de la petición Optimizar

Cuando al Manager le envían una petición de Optimizar una Textura Procedural de Pre-cálculo, chequea si tiene la Optimization requerida, en caso positivo procede a su ejecución. Para esto analiza la textura y modifica las coordenadas de textura contenidas en Mesh según los parámetros establecidos en Optimization. (Ver Figura 33).

### 3.7.5 Petición Exportar

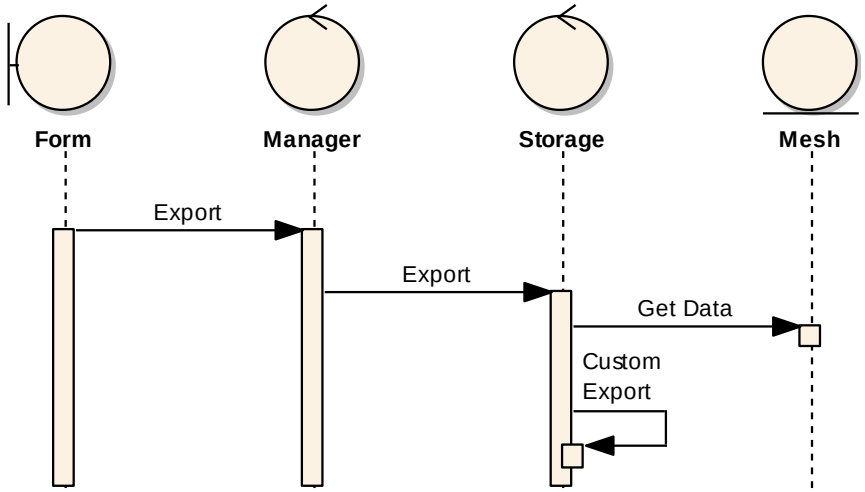



Fig.34 Diagrama de secuencia de la petición Exportar

Cuando al Manager le envían una petición de Exportar un modelo tridimensional busca el Storage que pueda exportar el formato del fichero especificado y le asigna esta tarea. El Storage obtiene los datos de Mesh y los almacena con el formato adecuado. (Ver Figura 34).

### 3.7.6 Petición Cargar/Salvar Configuración

Si al Manager le llega la petición de Cargar Configuración, este abre el fichero predeterminado, extrae los datos y los almacena en Settings. En caso de que la petición que le llegue sea de Salvar Configuración escribe en el fichero los datos almacenados en Settings. (Ver Figura 35).

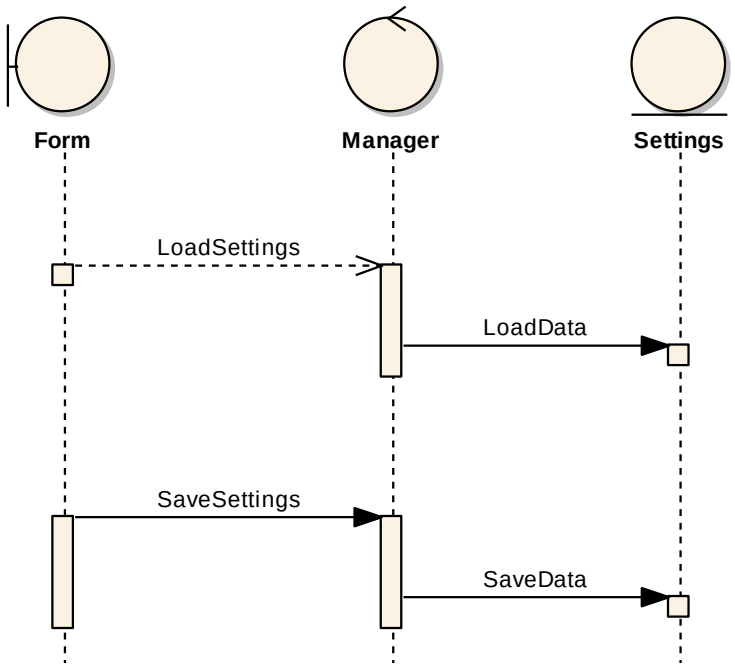



Fig.35 Diagrama de secuencia para la petición Cargar/Salvar Configuración

### 3.8 Interfaz de usuario

La comunicación del usuario con la computadora es un punto clave para determinar qué quiere hacer el usuario en la aplicación y cómo desea hacerlo para obtener un resultado satisfactorio. Para ello, se realizó un estudio con el objetivo determinar la mejor vía para lograr que la interfaz de usuario sea fácil de entender y de utilizar. Este estudio determinó que la mejor opción es utilizar un asistente o wizard.

La elección ha sido basada en el hecho de que un wizard guía al usuario paso a paso todo el tiempo, evitando que se le olvide realizar alguna operación y consiguiendo que se cumplan todas las condiciones que requiere el sistema. Empleando el asistente se logra que el usuario no necesite adiestrarse para utilizar el programa, simplemente puede seguir cada una de las instrucciones mostradas en pantalla y seleccionar o modificar las opciones que desee.

A continuación se presentan cada una de las pantallas del asistente y se explican todas las opciones que contiene cada una.

### 3.8.1 Pantalla Aplicación

Esta pantalla es de información al usuario. En ella se brinda una breve descripción de la funcionalidad del software y se explican brevemente las acciones que realiza el programa al ejecutar las optimizaciones implementadas. (Ver Figura 36).

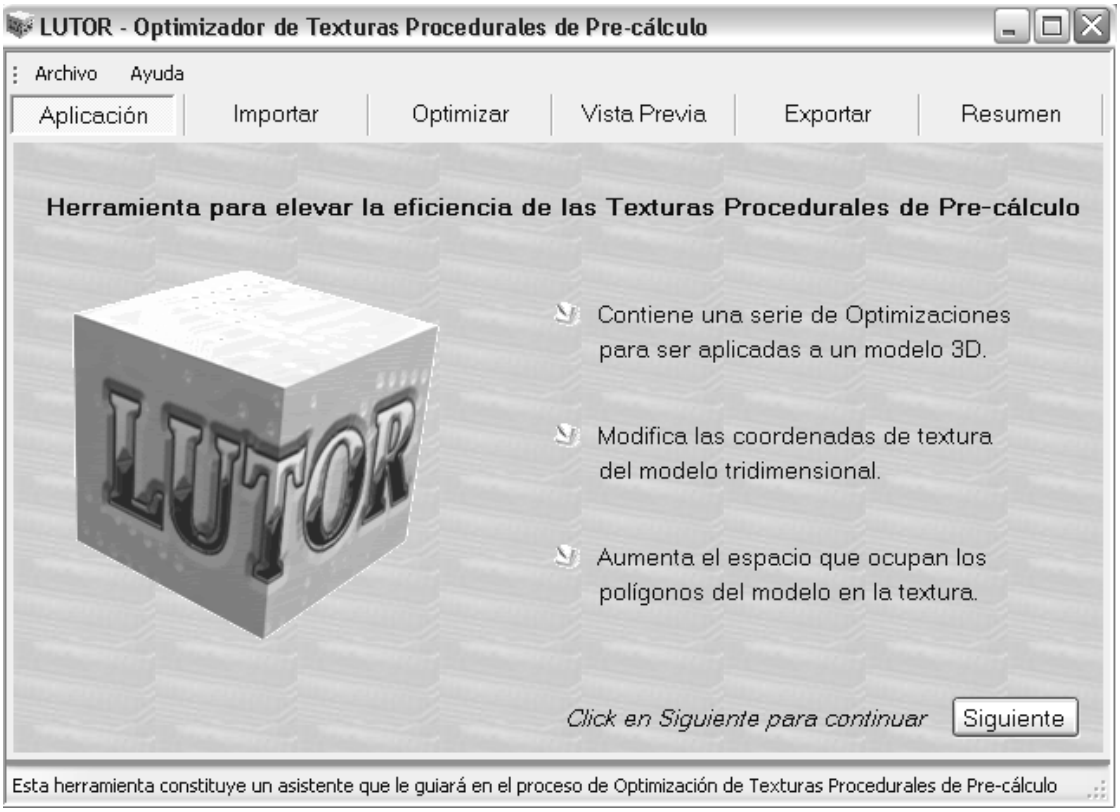



Fig.36 Pantalla Aplicación

### 3.8.2 Pantalla Importar

La Figura 37 presenta la pantalla Importar. Como se puede apreciar tiene tres opciones fundamentales. La primera es la selección de la ubicación del modelo tridimensional a importar. Esto se puede hacer escribiendo la dirección manualmente (**A**) o realizando una búsqueda (**B**).

La segunda opción es la selección de la ubicación de la Textura Procedural de Pre-cálculo a optimizar. Se puede seleccionar al igual que el modelo, es decir, escribiendo manualmente la dirección (**C**) o realizando una búsqueda (**D**).

La tercera opción se utiliza para indicar el canal de textura en el que se encuentra la Textura Procedural a procesar (**E**). Una vez especificados estos ficheros se activa el botón siguiente para pasar a realizar las próximas instrucciones.

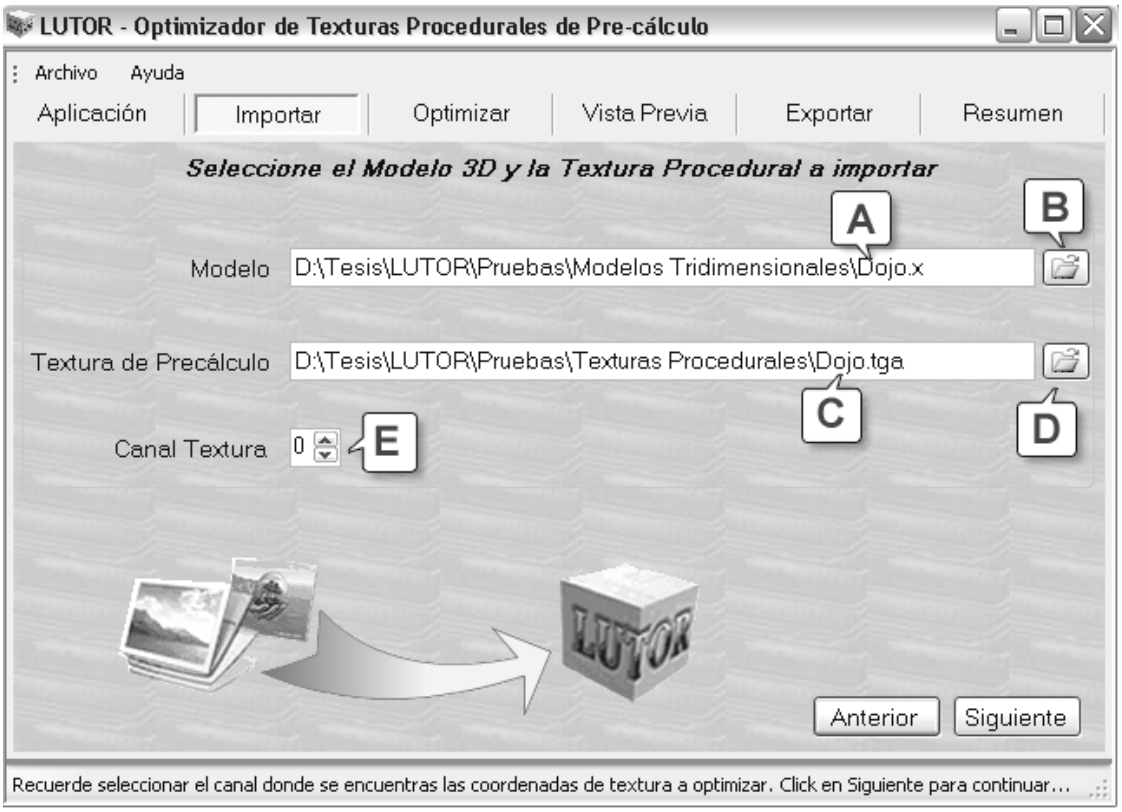



Fig.37 Pantalla Importar

### 3.8.3 Pantalla Optimizar

En esta pantalla se muestra una lista con todas las optimizaciones que contiene el software (**A**). En cada optimización existe un checkbox (**B**) para que el usuario elija la que desee aplicar. Si una optimización es seleccionada y contiene parámetros entonces se activa el panel Parámetros (**C**) con la configuración de dicha optimización, permitiendo que el usuario pueda modificar los parámetros de esta.

Si el usuario no quiere configurar las optimizaciones, se aplica una configuración por defecto. En el caso de que desee utilizar una configuración que creó con anterioridad, en el panel Configuración (**D**) puede seleccionar dicha configuración.

Si el usuario necesita crear una nueva configuración, puede ir al panel Configuración (**E**). Al seleccionar esta opción se activa un nuevo panel, el panel Nueva Configuración. Este último ofrece la posibilidad de indicar el nombre (**F**) con que se desea guardar la configuración, para identificarla fácilmente cuando necesite utilizarla. También si desea eliminar alguna configuración lo puede realizar mediante el botón Eliminar (**G**).

Además de las opciones descritas también ofrece la posibilidad de que el usuario pueda tener una vista previa (**H**) de cómo quedará el espacio de textura resultante. Este proceso se muestra en la Figura 38.

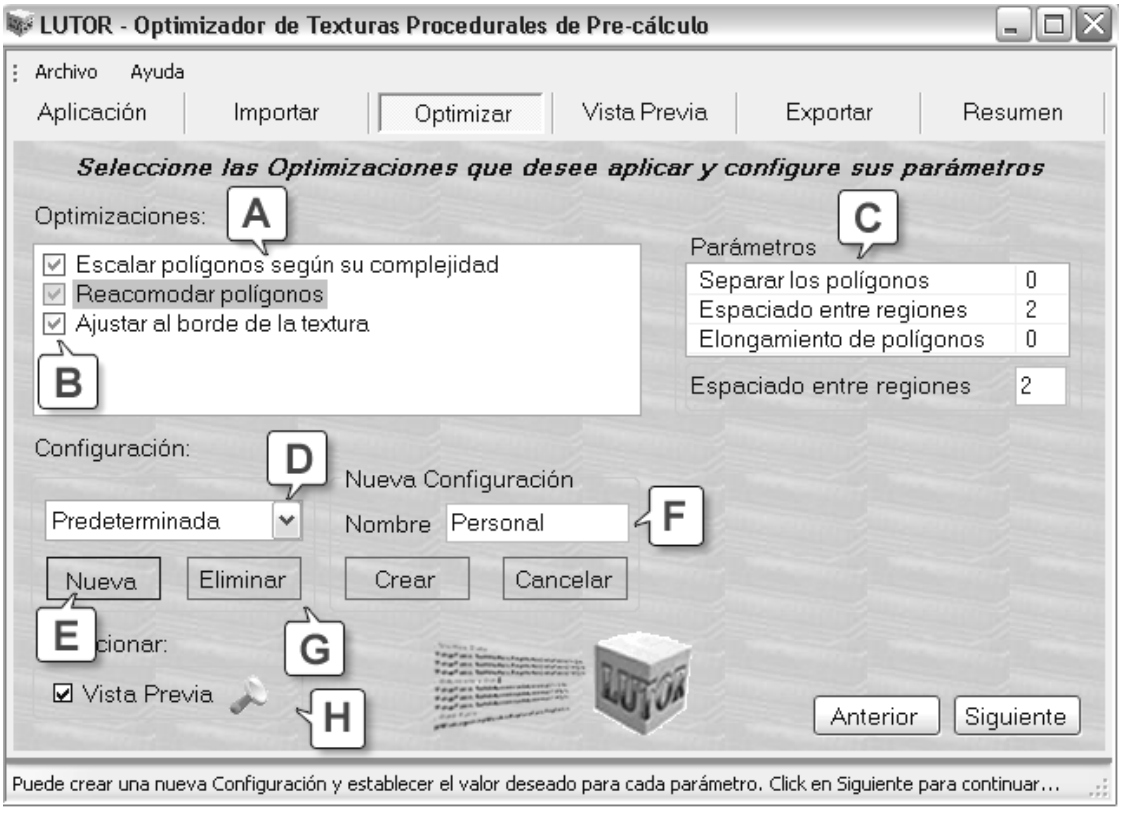



Fig.38 Pantalla Optimizar

### 3.8.4 Pantalla Vista Previa

Si en la pantalla anterior el usuario seleccionó Vista Previa, en esta pantalla se muestra el resultado de la nueva distribución de las coordenadas de textura. En la parte derecha se muestra el modelo original (**A**) y en la parte izquierda el modelo optimizado (**B**), ambos con sus respectivas eficiencias.

Es válido aclarar que para obtener la textura con la calidad visual requerida se necesita volver a generarla con las nuevas coordenadas de textura, en el programa que inicialmente la creó. Sin embargo esta pantalla es útil porque permite conocer como quedaron distribuidas las regiones con las optimizaciones aplicadas. Además, le permite al usuario determinar si necesita escoger otra optimización para aplicar o cambiar algunos parámetros para obtener un mejor resultado.

En la Figura 39 se puede observar un ejemplo de la Vista Previa de un modelo original y otro resultante de las optimizaciones.


Fig.39 Pantalla Vista Previa

### 3.8.5 Pantalla Exportar

El próximo paso del asistente es exportar el modelo cargado con las modificaciones hechas en las coordenadas de textura. La pantalla Exportar brinda la posibilidad de determinar el nombre del modelo resultante y la ubicación del mismo. Esta operación se puede hacer escribiendo la dirección de forma manual (**A**) o realizando una búsqueda (**B**). (Ver Figura 40).

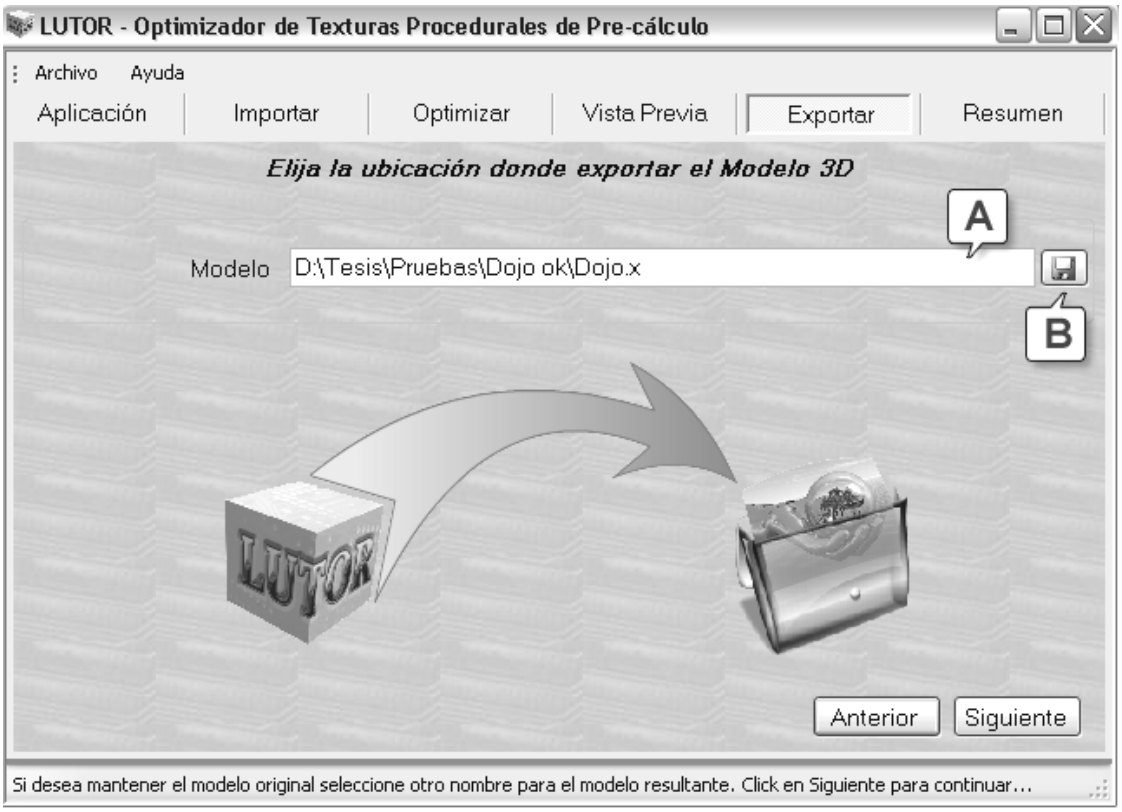



Fig.40 Pantalla Exportar

### 3.8.6 Pantalla Resumen

En esta pantalla se muestran datos generales como la cantidad de optimizaciones escogidas para ser aplicadas (**A**) y la cantidad que realmente se aplicaron (**B**). Además, los nombres de las optimizaciones que se aplicaron se muestran en una lista (**C**).

Asimismo se muestra en el panel derecho (**D**) el porciento del espacio de textura ocupado en el modelo original y el resultante, con el objetivo de que el usuario pueda comparar dichos valores. En la Figura 41 se puede observar un ejemplo de resumen.

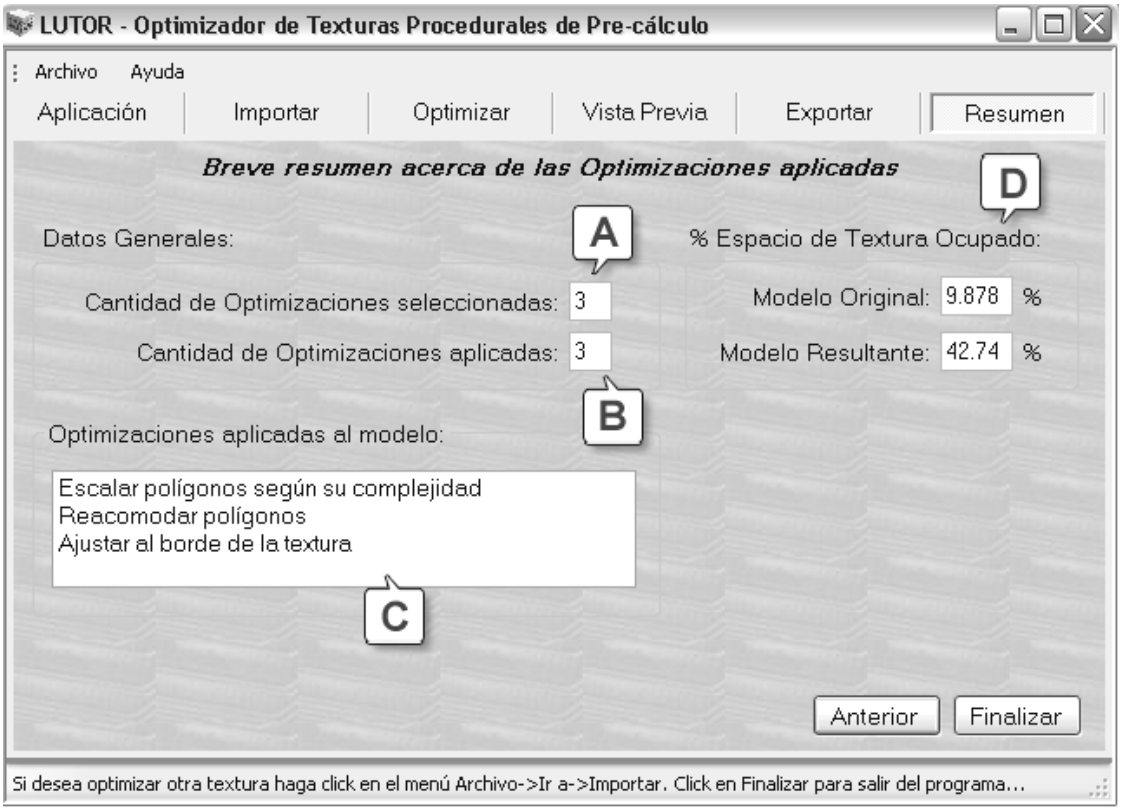



Fig.41 Pantalla Resumen

# 4

# CAPÍTULO
# IMPLEMENTACIÓN DEL SISTEMA

El presente capítulo muestra los diagramas de clases de la aplicación, así como los diagramas de secuencia por casos de uso. Además se explica el funcionamiento del sistema, resaltando la comunicación entre las clases del Framework usado y los métodos y clases implementadas.

## 4.1 Diagrama de clases

En el Anexo 1 se presenta el diagrama de clases general, el cual describe gráficamente las especificaciones de las clases del software y de la interfaz, así como sus relaciones en la aplicación.

Para una mejor comprensión del diseño de las clases el sistema fue dividido en cuatro paquetes. Estos se muestran en la Figura 42, donde los paquetes Storage, Settings y Optimization son controlados por el paquete principal del sistema denominado Manager. Este último es llamado desde la interfaz del software, lo cual evidencia el uso de la implementación de software por capas.

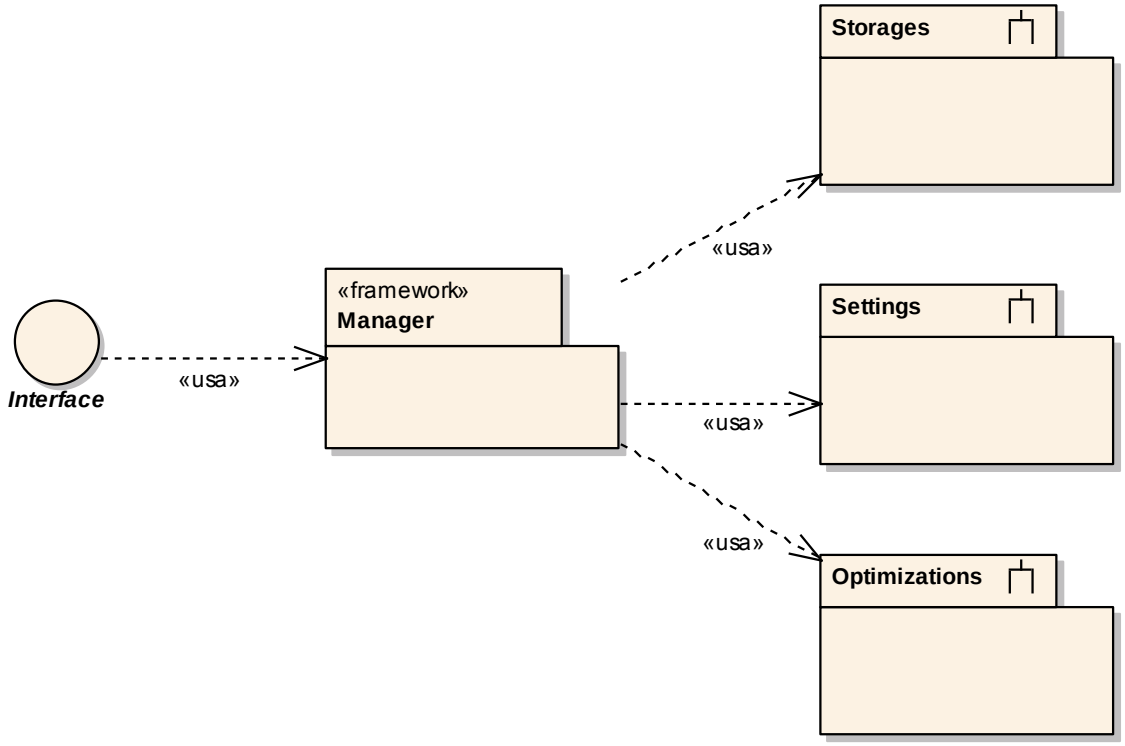



Fig.42 Diagrama de clases del diseño por paquetes

### 4.1.1 Diagrama de clases del paquete Manager

En el paquete Manager se encuentran las clases principales de la aplicación, por tanto constituye la base del sistema. Está compuesto por seis clases, donde una de ellas es la clase cManager, la cual maneja todas las peticiones que realiza el usuario a través de la interfaz. (Ver Figura 43).

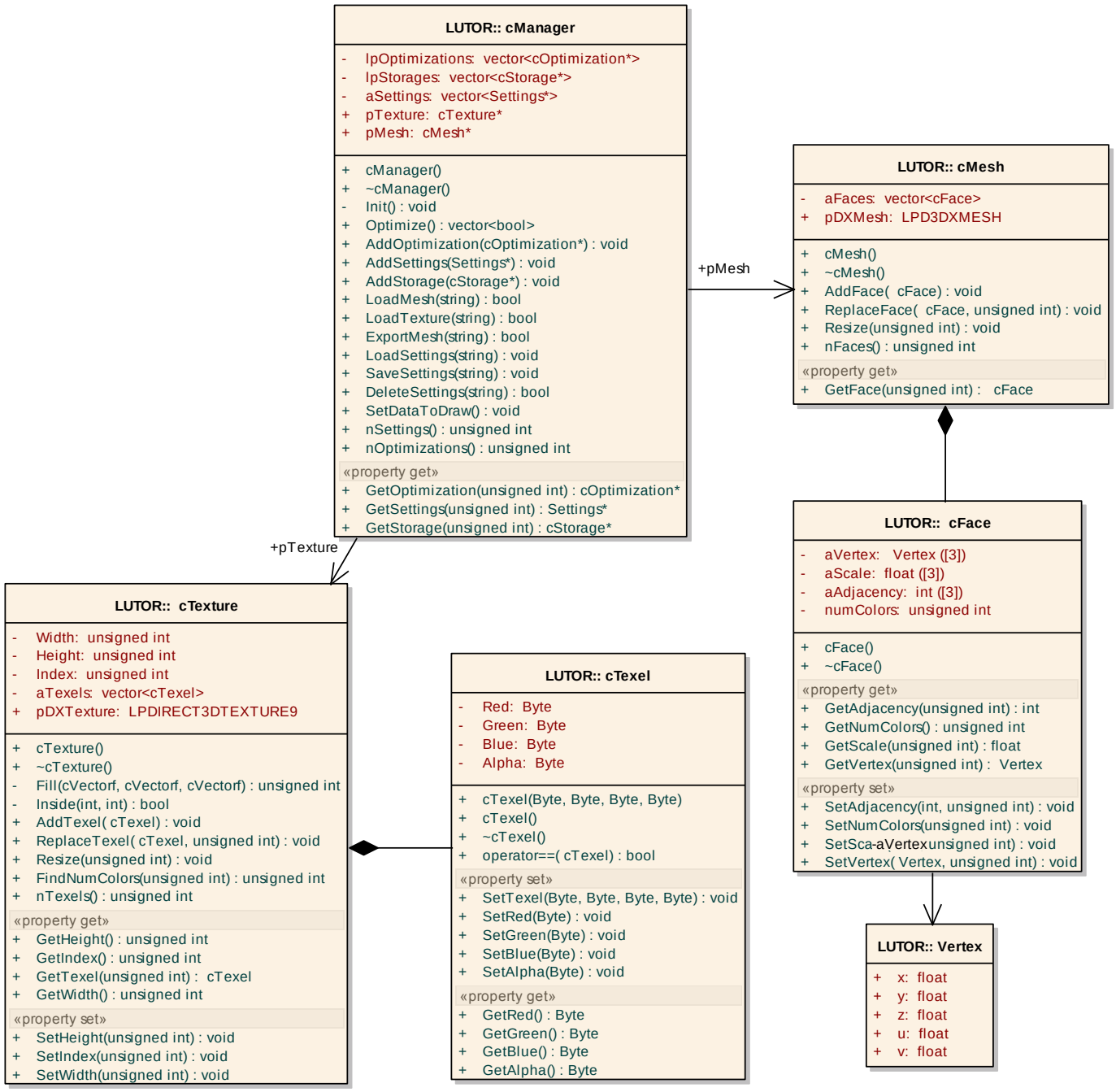



Fig.43 Diagrama de clases del paquete Manager

### 4.1.2 Diagrama de clases del paquete Storage

El paquete Storage contiene las clases encargadas de importar y exportar los ficheros que maneja la aplicación. Debido a la gran variedad de formatos existentes actualmente tanto para los modelos tridimensionales como para las texturas, el paquete contiene una clase abstracta que permite redefinir nuevas clases para incorporar otros formatos sin afectar el diseño actual del sistema. (Ver Figura 44).

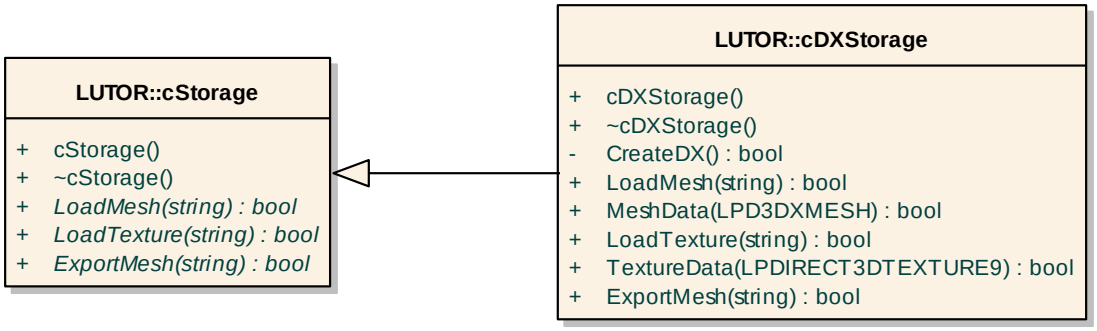



Fig.44 Diagrama de clases del paquete Storage

### 4.1.3 Diagrama de clases del paquete Settings

El paquete Settings agrupa las clases encargadas de guardar las configuraciones del software. Las clases contenidas en este paquete almacenan si una optimización está seleccionada o no, además de los valores de los parámetros establecidos por el usuario al crear nuevas configuraciones. En dependencia de la configuración establecida por el usuario se seleccionan las optimizaciones y se actualizan los parámetros con los valores almacenados. (Ver Figura 45).

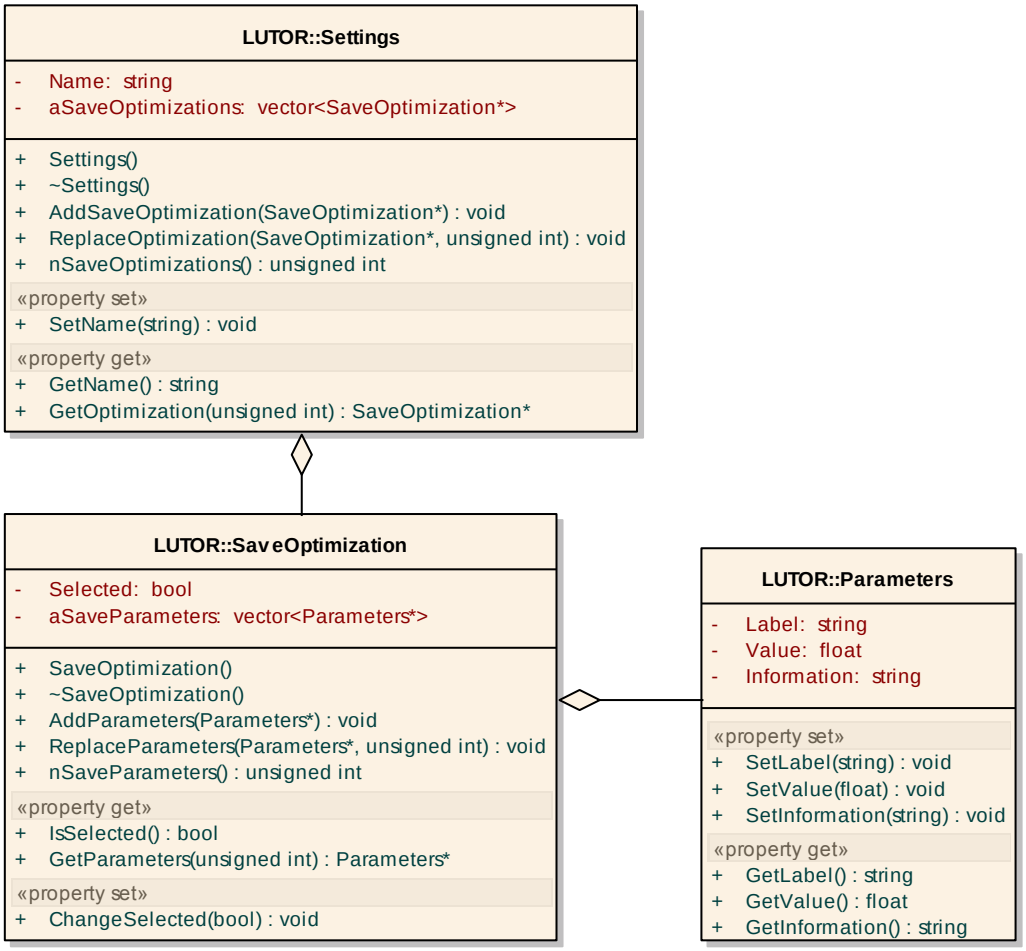



Fig.45 Diagrama de clases del paquete Settings

### 4.1.4 Diagrama de clases del paquete Optimization

Este paquete contiene las clases encargadas de realizar las diferentes modificaciones a las coordenadas de textura del modelo tridimensional, permitiendo elevar la eficiencia de las Texturas Procedurales de Pre-cálculo. Para manejar dichas modificaciones fue creada una clase abstracta de la cual heredan todas las optimizaciones implementadas. Esto permite que si se desea adicionar una nueva optimización a la aplicación no es necesario cambiar el diseño actual, además de enriquecer el software con nuevas prestaciones. (Ver Figura 45).

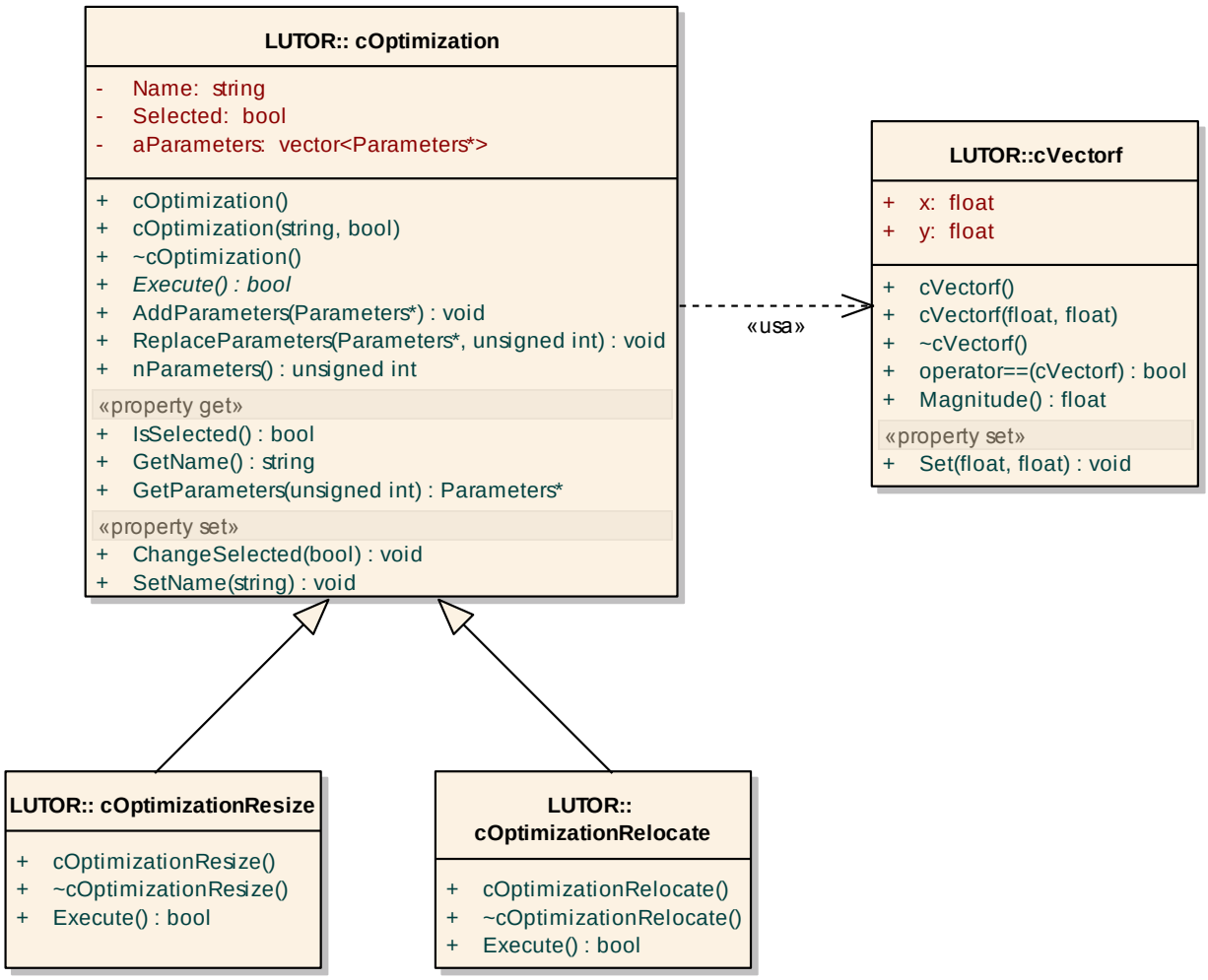



Fig.46 Diagrama de clases del paquete Optimization

## 4.2 Descripción de las clases

A continuación se presentan una serie de Tablas que contienen la descripción detallada de cada una de las clases del diseño del sistema, incluyendo los atributos y métodos para lograr una mejor comprensión del mismo.

| **Nombre:** | cManager |
|---|---|
| **Tipo de clase:** | Controladora |
| **Atributo** | **Tipo** |
| lpOptimizations | vector<cOptimization*> |
| lpStorages | vector<cStorage*> |
| aSettings | vector<Settings*> |
| **Responsabilidades**: | |
| **Nombre** | **Descripción** |
| Init () | Inicializa los atributos de la clase, crea y adiciona las Optimizaciones y los Storages. |
| LoadMesh (mesh: string) | Importa el modelo cuya dirección es especificada en la variable mesh. Utiliza un Storage creado por el programa y especificado por el usuario. |
| ExportMesh (mesh: string) | Exporta el modelo tridimensional en la dirección especificada en la variable mesh. Utiliza un Storage creado por el programa y especificado por el usuario. |
| LoadTexture (texture: string) | Importa la textura cuya dirección es especificada en la variable texture. Utiliza un Storage creado por el programa y especificado por el usuario. |
| LoadSettings (fileName: string) | Carga las configuraciones del sistema desde un fichero externo, su dirección es especificada en la variable fileName. En caso de que no exista el fichero crea una nueva configuración con valores predeterminados. |
| AddSettings (pSettings: cSettings*) | Adiciona una nueva configuración del sistema. |
| DeleteSettings (settingsName: string) | Elimina la configuración del sistema cuyo nombre se especifica en el parámetro settingsName. |

| | |
|---|---|
| SaveSettings (fileName: string) | Salva las configuraciones del sistema a un fichero externo, su dirección es especificada en la variable fileName. Si el fichero existe, lo reemplaza con los nuevos valores de las configuraciones del sistema. |
| SetDataToDraw () | Extrae los valores del modelo optimizado y los almacena, para luego ser utilizados en el dibujado del modelo. |
| nOptimizations () | Retorna la cantidad de Optimizaciones que fueron cargadas. |
| nSettings () | Retorna el número de configuraciones existentes. |
| Optimize () | Ejecuta las optimizaciones seleccionadas por el usuario. Retorna un arreglo booleano que determina cuales optimizaciones fueron ejecutadas satisfactoriamente. |

Tabla 9: Descripción de la clase cManager

| **Nombre:** | cTexture |
|---|---|
| **Tipo de clase:** | Entidad |
| **Atributo** | **Tipo** |
| Width | unsigned int |
| Height | unsigned int |
| Index | unsigned int |
| aTexels | vector<cTexel> |
| **Responsabilidades**: | |
| **Nombre** | **Descripción** |
| Fill (vector1: cVectorf, vector2:cVectorf, vector3: cVectorf) | Recorre el espacio de textura determinado por el polígono y devuelve la cantidad de texeles contenidos en este. |

| | |
|---|---|
| FindNumColors (i: unsigned int) | Devuelve la cantidad de texeles diferentes que ocupa en la textura la i-ésimo polígono del mesh. |
| Inside(x:int, y:int) | Retorna verdadero si un texel determinado se encuentra dentro de la textura. |
| AddTexel (texel: cTexel) | Adiciona un objeto de la clase cTexel al arreglo de texeles (aTexels). |
| ReplaceTexel (texel: cTexel, i:unsigned int) | Reemplaza el texel de la i-ésima posición del arreglo de texeles por la variable texel que recibe como parámetro. |
| Resize (n: unsigned int) | Reserva memoria para n texeles, los crea y adiciona al arreglo de texeles. |
| nTexels () | Retorna el número de texeles que contiene la textura. |

Tabla 10: Descripción de la clase cTexture

| **Nombre:** | cTexel |
|---|---|
| **Tipo de clase:** | Entidad |
| **Atributo** | **Tipo** |
| Alpha | Byte |
| Red | Byte |
| Green | Byte |
| Blue | Byte |
| **Responsabilidades**: | |
| **Nombre** | **Descripción** |
| operator== (texel: cTexel) | Sobrecarga del operador ==. Retorna verdadero si los valores de los atributos del objeto de la clase son iguales a los del texel que se le pasa como parámetro. |

Tabla 11: Descripción de la clase cTexel

| **Nombre:** | cMesh |
|---|---|
| **Tipo de clase:** | Entidad |
| **Atributo** | **Tipo** |
| aFaces | vector<cFace> |
| pDXMesh | LPD3DXMESH |
| pDXSaveMesh | LPD3DXMESH |
| **Responsabilidades**: | |
| **Nombre** | **Descripción** |
| AddFace (face: cFace) | Adiciona un objeto de la clase cFace al arreglo de caras (aFaces). |
| ReplaceFace (face: cFace, i:unsigned int) | Reemplaza la cara de la i-ésima posición del arreglo de caras por la variable face que recibe como argumento. |
| Resize (n: unsigned int) | Reserva memoria para n caras, las crea y adiciona al arreglo de caras. |
| nFaces () | Retorna el número de caras que contiene el modelo. |

Tabla 12: Descripción de la clase cMesh

| **Nombre:** | cFace |
|---|---|
| **Tipo de clase:** | Entidad |
| **Atributo** | **Tipo** |
| aVertex[3] | Vertex |
| aScale[3] | float |
| aAdjacency[3] | int |
| numColors | unsigned int |

Tabla 13: Descripción de la clase cFace

| **Nombre:** | Vertex |
|---|---|
| **Tipo de clase:** | Entidad |
| **Atributo** | **Tipo** |
| u | float |
| v | float |
| x | float |
| y | float |
| z | float |

Tabla 14: Descripción de la clase Vertex

| **Nombre:** | cStorage |
|---|---|
| **Tipo de clase:** | Controladora |
| **Responsabilidades**: | |
| **Nombre** | **Descripción** |
| LoadMesh (mesh: string) | Método virtual. Las clases que heredan de cStorage deben implementar este método para que puedan importar modelos utilizando diferentes formatos. |
| ExportMesh (mesh: string) | Método virtual. Las clases que heredan de cStorage deben implementar este método para que puedan exportar modelos utilizando diferentes formatos. |
| LoadTexture (texture: string) | Método virtual. Las clases que heredan de cStorage deben implementar este método para que puedan importar texturas utilizando diferentes formatos. |

Tabla 15: Descripción de la clase cStorage

| **Nombre:** | cDXStorage |
|---|---|
| **Tipo de clase:** | Controladora |
| **Responsabilidades**: | |
| **Nombre** | **Descripción** |
| CreateDX () | Inicializa DirectX y crea el dispositivo del mismo. Retorna falso si no se inicializa satisfactoriamente. |
| LoadMesh (mesh: string) | Carga el modelo de la dirección especificada en el parámetro mesh. Retorna falso en caso de que no sea posible importar el modelo o extraer los datos del mismo de forma satisfactoria. |
| ExportMesh (mesh: string) | Exporta en modelo en la dirección especificada. Retorna falso si no es posible exportar el modelo satisfactoriamente. |
| LoadTexture (texture: string) | Carga la textura de la dirección especificada en el parámetro texture. Retorna falso en caso de que no sea posible importar la textura o extraer los datos de la misma de forma satisfactoria. |
| MeshData (pDXMesh: LPD3DXMESH) | Extrae los datos necesarios a la aplicación que se encuentran almacenados en la variable pDXMesh de DirectX. |
| TextureData (pDXTexture: LPDIRECT3DTEXTURE9) | Extrae los datos necesarios a la aplicación que se encuentran almacenados en la variable pDXTexture de DirectX. |

Tabla 16: Descripción de la clase cDXStorage

| **Nombre:** | cOptimization |
|---|---|
| **Tipo de clase:** | Controladora |
| **Atributo** | **Tipo** |
| Name | string |
| Selected | bool |
| aParameters | vector<Parameters*> |
| **Responsabilidades**: | |
| **Nombre** | **Descripción** |
| Execute () | Método virtual. Las clases que heredan de cOptimization deben implementar este método para que puedan ejecutar las optimizaciones que contengan. |
| AddParameters (pParameters: Parameters*) | Adiciona un nuevo parámetro al arreglo de parámetros (aParameters). |
| ReplaceParameters (pParameters: Parameters*, i: unsigned int) | Reemplaza el puntero a parámetro de la i-ésima posición del arreglo de parámetros por el puntero pParameters que recibe como argumento. |

Tabla 17: Descripción de la clase cOptimization

| **Nombre:** | Parameters |
|---|---|
| **Tipo de clase:** | Entidad |
| **Atributo** | **Tipo** |
| Label | string |
| Value | float |
| Information | string |

Tabla 18: Descripción de la clase Parameters

| **Nombre:** | cOptimizationResize |
|---|---|
| **Tipo de clase:** | Controladora |
| **Responsabilidades**: | |
| **Nombre** | **Descripción** |
| Execute () | Ejecuta la optimización “Escalar polígonos según su complejidad”. Retorna verdadero si la optimización se ejecutó satisfactoriamente. |

Tabla 19: Descripción de la clase cOptimizationResize

| **Nombre:** | cOptimizationRelocate |
|---|---|
| **Tipo de clase:** | Controladora |
| **Responsabilidades**: | |
| **Nombre** | **Descripción** |
| Execute () | Ejecuta la optimización “Reacomodar polígonos”. Retorna verdadero si la optimización se ejecutó satisfactoriamente. |

Tabla 20: Descripción de la clase cOptimizationRelocate

## 4.3 Descripción de la implementación por casos de uso

Para describir la implementación de los casos de uso, así como cada uno de los métodos y flujos que se utilizan en el software, se crearon diferentes diagramas de secuencia. En el Anexo 2 se muestran estos diagramas, los cuales contienen todos los métodos que conforman el caso de uso desde que el usuario interactúa con la interfaz hasta que obtiene un resultado.

## 4.4 Diagramas de componentes

Los ficheros que implementan las clases, así como sus relaciones, se muestran en el siguiente diagrama de componentes agrupados en paquetes. Para mayor información en el Anexo 3 se describe el diagrama de componente de cada uno de estos paquetes.

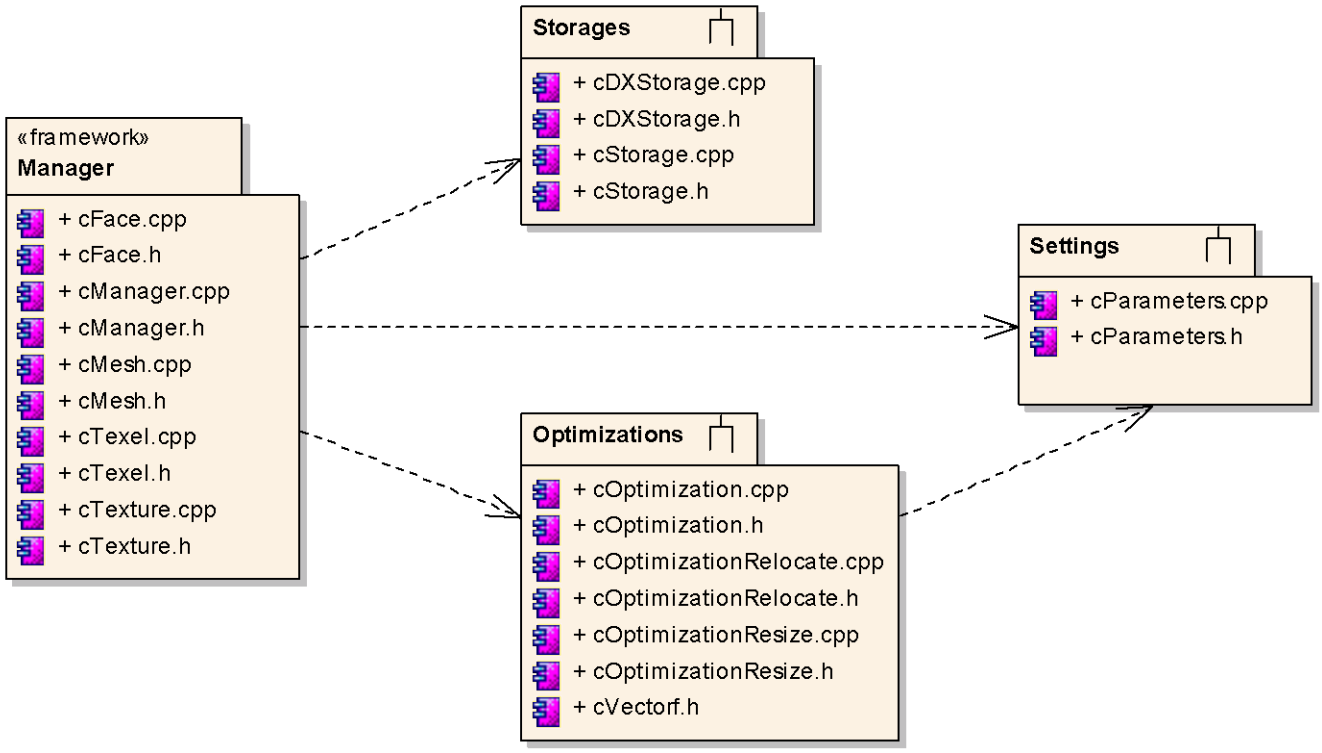



Fig. 47 Diagrama de componentes por paquetes

# 5

# CAPÍTULO
# RESULTADOS

El presente capítulo muestra una serie de resultados obtenidos luego de realizar diferentes pruebas al software implementado. Para esto se seleccionó una muestra de tres modelos tridimensionales con Texturas Procedurales de Pre-cálculo.

Para la evaluación del software se seleccionó una muestra aleatoria de tres modelos tridimensionales con las correspondientes Texturas Procedurales de Pre-cálculo que le fueron aplicadas. El criterio que se tuvo en cuenta para esta evaluación fue el porciento de espacio de textura ocupado del modelo resultante, luego de la aplicación de las optimizaciones implementadas.

En cada una de las muestras se establecieron determinados valores para los parámetros que configuran las optimizaciones. Las pruebas realizadas a las muestras se agruparon en dos grupos fundamentales. Un grupo de pruebas con los parámetros recomendados o por defecto y otro grupo al que se le varían estos parámetros para apreciar cuanto mejoran los resultados según las variaciones.

## 5.1 Pruebas con los parámetros por defecto

Los parámetros que se establecieron por defecto en la optimización “Reacomodar Polígonos” fueron valores con los cuales se obtuvieron los mejores resultados para la mayoría de las pruebas.

Se estableció valor “0” en separar los polígonos, pues permite que los polígonos no se separen, es decir, mantiene la adyacencia de cada polígono del modelo original. También se fijó valor “2” en espacio entre regiones porque logra que no se pierda mucho espacio en la textura e impide que el filtro de textura que aplica el hardware gráfico cambie el color de la textura. Además se estableció valor “0” en elongamiento de polígonos porque permite que los polígonos no tengan ninguna deformación.

En las siguientes Tablas se muestran tres modelos a los cuales se les aplicaron las diferentes optimizaciones. Además presentan los resultados obtenidos al aplicarle combinaciones de las optimizaciones implementadas con los valores de los parámetros por defecto. (Ver Tablas 21, 22 y 23).

| **Modelo:** Dojo.x | **Textura:** Dojo.tga | | | |
|---|---|---|---|---|
| ***% de Espacio de Textura ocupado inicial*** | 9.878 | | | |
| **Optimizaciones** | **Seleccionadas** | | | |
| Escalar polígonos según su complejidad | | X | | X |
| Reacomodar Polígonos | X | X | X | X |
| Ajustar al borde de la textura | | | X | X |
| ***% de Espacio de Textura ocupado*** | 39.02 | 40.12 | 39.07 | 42.74 |

Tabla 21: Resultados de las pruebas realizadas al modelo Dojo.x

| **Modelo:** Librero.x | **Textura:** Librero.tga | | | |
|---|---|---|---|---|
| ***% de Espacio de Textura ocupado inicial*** | 13.59 | | | |
| **Optimizaciones** | **Seleccionadas** | | | |
| Escalar polígonos según su complejidad | | X | | X |
| Reacomodar Polígonos | X | X | X | X |
| Ajustar al borde de la textura | | | X | X |
| ***% de Espacio de Textura ocupado*** | 34.67 | 36.35 | 34.81 | 36.44 |

Tabla 22: Resultados de las pruebas realizadas al modelo Librero.x

| **Modelo:** LS_Sector2.x | **Textura:** LS_Sector2_LM.tga | | | |
|---|---|---|---|---|
| ***% de Espacio de Textura ocupado inicial*** | 22.42 | | | |
| **Optimizaciones** | **Seleccionadas** | | | |
| Escalar polígonos según su complejidad | | X | | X |
| Reacomodar Polígonos | X | X | X | X |
| Ajustar al borde de la textura | | | X | X |
| ***% de Espacio de Textura ocupado*** | 64.45 | 78.19 | 64.66 | 78.33 |

Tabla 23: Resultados de la prueba realizada al modelo LS_Sector2.x

Analizando los resultados anteriores, es posible apreciar que en todos los casos se logra un mejor aprovechamiento del espacio de la textura, al aplicarle las optimizaciones implementadas y estableciendo los parámetros recomendados.

## 5.2 Pruebas con parámetros variables

Los modelos 3D poseen características diferentes, es decir, no tienen igual números de polígonos y/o igual distribución de los mismos, entre otros aspectos. Por tal motivo en algunos casos se obtienen mejores resultados al establecer determinados parámetros.

Las Tablas a continuación muestran los resultados obtenidos al aplicar las optimizaciones con diferentes valores en los parámetros de la optimización “Reacomodar Polígonos”.

| **Modelo:** Dojo.x<br>**Textura:** Dojo.tga | | | ***% de Espacio de Textura ocupado inicial*** | 9.878 |
|---|---|---|---|---|
| **Optimizaciones** | **Seleccionadas** | | *Optimización Reacomodar Polígonos* | |
| Escalar polígonos según su complejidad | X | X | *Parámetros* | *Valor* |
| Reacomodar Polígonos | X | X | *Separar los polígonos* | *0* |
| Ajustar al borde de la textura | | X | *Espaciado entre regiones* | *1* |
| ***% de Espacio de Textura ocupado*** | 55.51 | 55.83 | *Elongamiento de polígonos* | *0* |
| **Optimizaciones** | **Seleccionadas** | | *Optimización Reacomodar Polígonos* | |
| Escalar polígonos según su complejidad | X | X | *Parámetros* | *Valor* |
| Reacomodar Polígonos | X | X | *Separar los polígonos* | *0* |
| Ajustar al borde de la textura | | X | *Espaciado entre regiones* | *2* |
| ***% de Espacio de Textura ocupado*** | 50.57 | 51.04 | *Elongamiento de polígonos* | *0.5* |

Tabla 24: Resultados de las pruebas realizadas al modelo Dojo.x (con parámetros variables)

| **Modelo:** Librero.x<br>**Textura:** Librero.tga | | | ***% de Espacio de Textura ocupado inicial*** | 13.59 |
|---|---|---|---|---|
| **Optimizaciones** | **Seleccionadas** | | *Optimización Reacomodar Polígonos* | |
| Escalar polígonos según su complejidad | X | X | *Parámetros* | *Valor* |
| Reacomodar Polígonos | X | X | *Separar los polígonos* | *0* |
| Ajustar al borde de la textura | | X | *Espaciado entre regiones* | *1* |
| ***% de Espacio de Textura ocupado*** | 44.64 | 44.81 | *Elongamiento de polígonos* | *0* |

| **Optimizaciones** | **Seleccionadas** | | *Optimización Reacomodar Polígonos* | |
|---|---|---|---|---|
| Escalar polígonos según su complejidad | X | X | *Parámetros* | *Valor* |
| Reacomodar Polígonos | X | X | *Separar los polígonos* | *1* |
| Ajustar al borde de la textura | | X | *Espaciado entre regiones* | *2* |
| ***% de Espacio de Textura ocupado*** | 27.44 | 35.65 | *Elongamiento de polígonos* | *0* |

Tabla 25: Resultados de las pruebas realizadas al modelo Librero.x (con parámetros variables)

| **Modelo:** LS_Sector2.x<br>**Textura:** LS_Sector2_LM.tga | ***% de Espacio de Textura ocupado inicial*** | 22.42 | | |
|---|---|---|---|---|
| **Optimizaciones** | **Seleccionadas** | | *Optimización Reacomodar Polígonos* | |
| Escalar polígonos según su complejidad | X | X | *Parámetros* | *Valor* |
| Reacomodar Polígonos | X | X | *Separar los polígonos* | *0* |
| Ajustar al borde de la textura | | X | *Espaciado entre regiones* | *1* |
| ***% de Espacio de Textura ocupado*** | 83.31 | 84.72 | *Elongamiento de polígonos* | *0* |
| **Optimizaciones** | **Seleccionadas** | | *Optimización Reacomodar Polígonos* | |
| Escalar polígonos según su complejidad | X | X | *Parámetros* | *Valor* |
| Reacomodar Polígonos | X | X | *Separar los polígonos* | *1* |
| Ajustar al borde de la textura | | X | *Espaciado entre regiones* | *2* |
| ***% de Espacio de Textura ocupado*** | 33.69 | 47.68 | *Elongamiento de polígonos* | *0.9* |

Tabla 26: Resultados de las pruebas realizadas al modelo LS_Sector2.x (con parámetros variables)

Observando los resultados obtenidos se percibe una variación en el aprovechamiento del espacio de textura al cambiar los parámetros de “Reacomodar Polígonos”. Si se utilizan los valores establecidos en la primera prueba de cada modelo (0,1,0) se obtienen los mejores resultados. Estos valores son recomendados principalmente cuando las regiones tienen bordes iguales, ya que permiten mantener el color de la textura al ser aplicado el filtro por hardware gráfico.

## 5.3 Resumen de las pruebas

A continuación se muestra un resumen de los resultados obtenidos en las pruebas realizadas al software.

| Modelo | Textura | Porciento de espacio de textura ocupado | | |
|---|---|---|---|---|
| | | % Inicial | % Final | |
| | | | Valor Mínimo | Valor Máximo |
| Dojo.x | Dojo_LM.tga | 9.878 | 39.02 | 55.83 |
| Librero.x | Librero_LM.tga | 13.59 | 27.44 | 44.81 |
| LS_Sector2.x | LS_Sector2_LM.tga | 22.42 | 33.69 | 84.72 |

Tabla 27: Resumen de resultados obtenidos en las pruebas realizadas

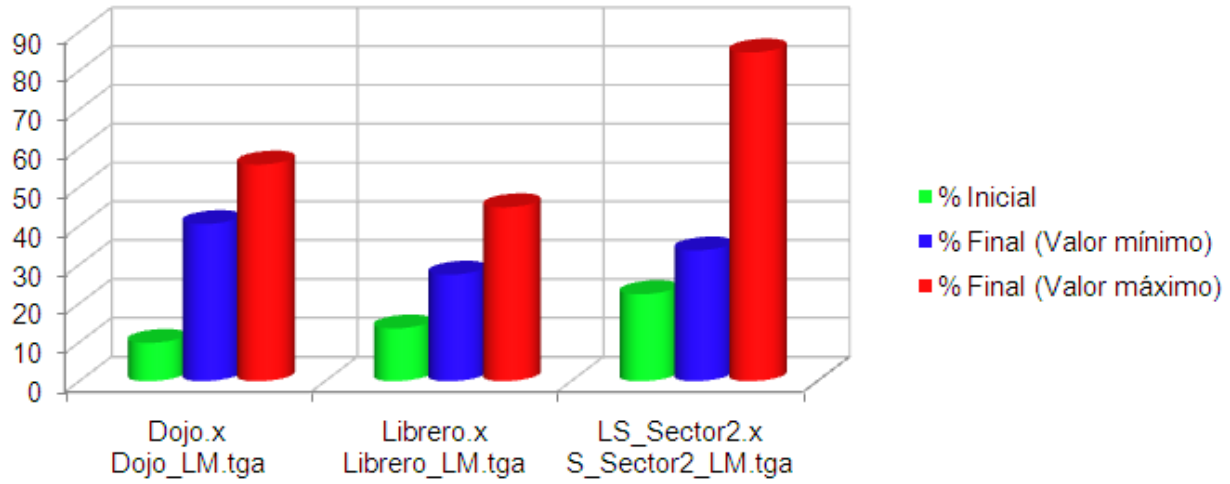



Fig. 48 Gráfico resumen con los resultados obtenidos en las pruebas realizadas

Las pruebas realizadas corresponden a la primera iteración del proceso propuesto para generar Texturas Procedurales de Pre-cálculo. Como se puede apreciar se alcanzaron mejoras significativas llevando a cabo solo una iteración, probando con diferentes combinaciones tanto de optimizaciones como de parámetros. Por lo expuesto anteriormente no fue necesario realizar otra iteración.

LUTOR no solo permite elevar la eficiencia de las Textura Procedurales de Pre-cálculo sino que además posibilita aumentar el detalle de la superficie de los modelos. Con el objetivo de mostrar el aumento de calidad visual se presenta la Figura 49 que muestra fotos tomadas del modelo LS_Sector2.x. A la izquierda se encuentra el modelo con la textura original y a la derecha el modelo con la textura generada luego de obtener nuevas coordenadas de texturas en la aplicación implementada.

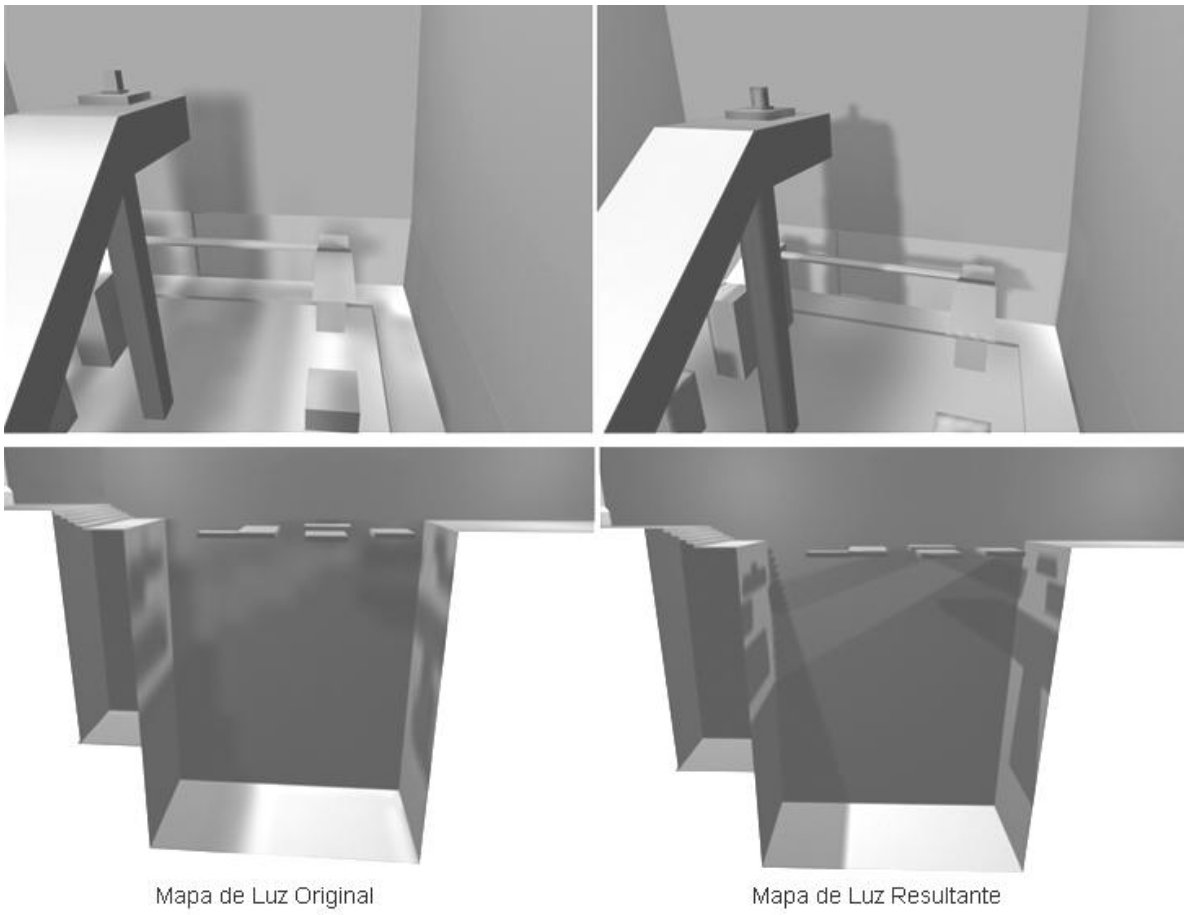



Fig. 49 Comparación de los mapas de luces aplicados al modelo LS_Sector2.x

Comparando estas imágenes se evidencia un aumento considerable del detalle en la superficie de los polígonos al tener aplicada la nueva textura. Esto se puede apreciar en la definición de las sombras proyectadas en las paredes del modelo.

# CONCLUSIONES

Como solución al problema existente en el proceso de creación de Texturas Procedurales de Pre-cálculo, fueron desarrolladas satisfactoriamente las siguientes tareas:

1. Se realizó un estudio del proceso actual de creación de Texturas Procedurales de Pre-cálculo, donde se comprobó que presentaban problemas en la forma que se distribuían las coordenadas de textura del modelo tridimensional, es decir, en el aprovechamiento del espacio de textura.

2. Fue creado un algoritmo para calcular el porciento de espacio de textura ocupado, así como otro algoritmo que permitió catalogar la relevancia de las regiones de la textura.

3. Se implementó un software que utiliza los algoritmos para calcular el porciento de espacio de textura ocupado y catalogar la relevancia de las regiones de la textura con el objetivo de modificar las coordenadas de textura del modelo, lo cual permite aumentar el espacio de textura utilizado.

4. Se propuso un proceso iterativo para la creación de Texturas Procedurales de Pre-cálculo, en el cual fue integrada la herramienta implementada.

5. El software fue extensamente probado y en el Capítulo 5 de esta tesis se documentan algunas de las pruebas realizadas, así como una descripción de las mejoras logradas al finalizar el proceso propuesto.

Por tanto se considera que quedó resuelto el objetivo planteado y se culminaron todas las tareas propuestas, sirviendo de un paso de avance en el desarrollo de los temas que forman parte de uno de los perfiles de la Facultad 5 de la Universidad de las Ciencias Informáticas.

# RECOMENDACIONES

1. Continuar el desarrollo del software con la implementación de otras optimizaciones que permitan mejorar el aprovechamiento del espacio de textura, como son las optimizaciones colapsar polígonos iguales y soldar polígonos.

2. Continuar la investigación y el desarrollo de algoritmos que permitan mejorar las optimizaciones actuales.

3. Buscar otras librerías que provean un servicio similar o mejor que la librería UVAtlas.

4. Realizar una versión de la aplicación que se integre como un Plugin del 3D Studio Max u otro software de diseño.

5. Adicionar una pantalla donde el usuario pueda editar las regiones, es decir, que el usuario pueda seleccionar las regiones y acomodarlas manualmente donde desee o escalarlas según sus necesidades.

# REFERENCIAS BIBLIOGRÁFICAS


(1) Texel. [En línea] [2007] http://www.answers.com/texel.

(2) **Spitzer, John. 2003.** Real-Time Procedural Effects. [Online] [2007] www.gamasutra.com/features/gdcarchive/2003E/RealTime-Procedural-Effects.pdf.

(3) Java2D: Gradients & Textures. [En línea] [2007] http://www.particle.kth.se/~fmi/kurs/PhysicsSimulation/Lectures/07B/gradientsTextures.html.

(4) **AYUCAR, IÑAKI, GARCÍA-ALONSO, ALEJANDRO y MATEY, LUIS. 2004.** Aplicaciones avanzadas del uso de texturas precomputadas en entornos de simulación en tiempo real. [En línea] [2007] www.ici.ubiobio.cl/revista/revista%202/45-58.pdf.

(5) **Rumbaugh, James, Jacobson, Ivar y Booch, Grady. 2000.** *El Lenguaje Unificado de Modelado.* Addison Wesley. pág. 526.

# BIBLIOGRAFÍA


1. *3ds max 7 User Reference.* 2004.
2. **Baylis, Alan. 2001.** Lightmapping Tutorial. [En línea] [2007] http://www.alsprogrammingresource.com/lightmapping_tutorial.html.
3. **Byl, Leigh Van der. 2004.** Lightwave 3D 8 Texturing. [En línea] [2007] http://books.google.com/.
4. **Castro, Silvia. 1997.** *Mapping Techniques.* SIGGRAPH 97. pág. 34.
5. **Corral, Olmo del.** Texturas Procedurales. [En línea] [2007] http://pgrafica.webideas4all.com/texturas_procedurales.html.
6. DirectX9 SDK Documentation. [En línea] Microsotf Corporation. [2007] http://msdn.microsoft.com.
7. **Ebert, David S, y otros. 1994.** Texturing and Modeling: A Procedural Approach. [En línea] [2007] http://books.google.com/.
8. **García Lorenzo, Marcos, Miraut Andrés, David y Robles Sánchez, Óscar David. 2007.** *Técnicas avanzadas de gráficos en 3D.* pág. 64.
9. **Garside, Ryan. 2006.** Procedural Textures: Gaming's Future. [En línea] [2007] http://www.bit-tech.net/gaming/2006/11/09/Procedural_Textures_Future_Gam/.
10. **Larman, Craig. 1999.** *UML y Patrones.* México : Prentice Hall. pág. 536.
11. **León, Rolando Hernández y Coello González, Sayda. 2002.** *El Paradigma Cuatitativo de la Investigación Científica.* Ciudad de la Habana : Editorial Universitaria. pág. 114.
12. **Lewis, Garry Q. y Lammers, Jim. 2004.** Maya 5 Fundamentals. [En línea] [2007] http://books.google.com/.
13. Lightmap. [En línea] [2007] http://en.wikipedia.org/wiki/Lightmap.

14. **Martínez, Alexander Subirós. 2006.** Mapeado. [En línea] [2007] http://www.mailxmail.com/curso/informatica/autocad3d/capitulo106.htm.

15. Normal Mapping. [En línea] [2007] http://www.answers.com/topic/normal-mapping.

16. Procedural Texture. [En línea] [2007] http://www.answers.com/%20procedural%20texture%201.

17. **Sierra, Francisco Javier Ceballos. 2003.** *Enciclopedia del lenguaje C++.* Ra-Ma. pág. 1087.

18. **Smith, Grant.** VGA Mode-X. [En línea] [2007] http://www.gamedev.net/reference/articles/article356.asp.

19. Texturas. [En línea] [2007] http://graficos.uji.es/grafica/Tutorial/Texturas.htm.

20. Texture mapping. [En línea] [2007] http://www.techweb.com/encyclopedia/defineterm.jhtml?term=texture+mapping&x=17&y=11.

21. Using UVAtlas (Direct3D 9). [En línea] [2007] http://msdn2.microsoft.com/en-us/library/bb206321.aspx.

22. **Webb, Jolyon. 2005.** Normal mapping. [En línea] [2007.] http://www.computerarts.co.uk/tutorials/3d__and__animation/normal_mapping.

23. What is a texture gradient? [En línea] [2007] http://ahsmail.uwaterloo.ca/kin356/texture/what.htm.

24. **Wikipedia, Colectivo de.** Bresenham's line algorithm. [En línea] [2007] http://en.wikipedia.org/wiki/Bresenham's_line_algorithm.

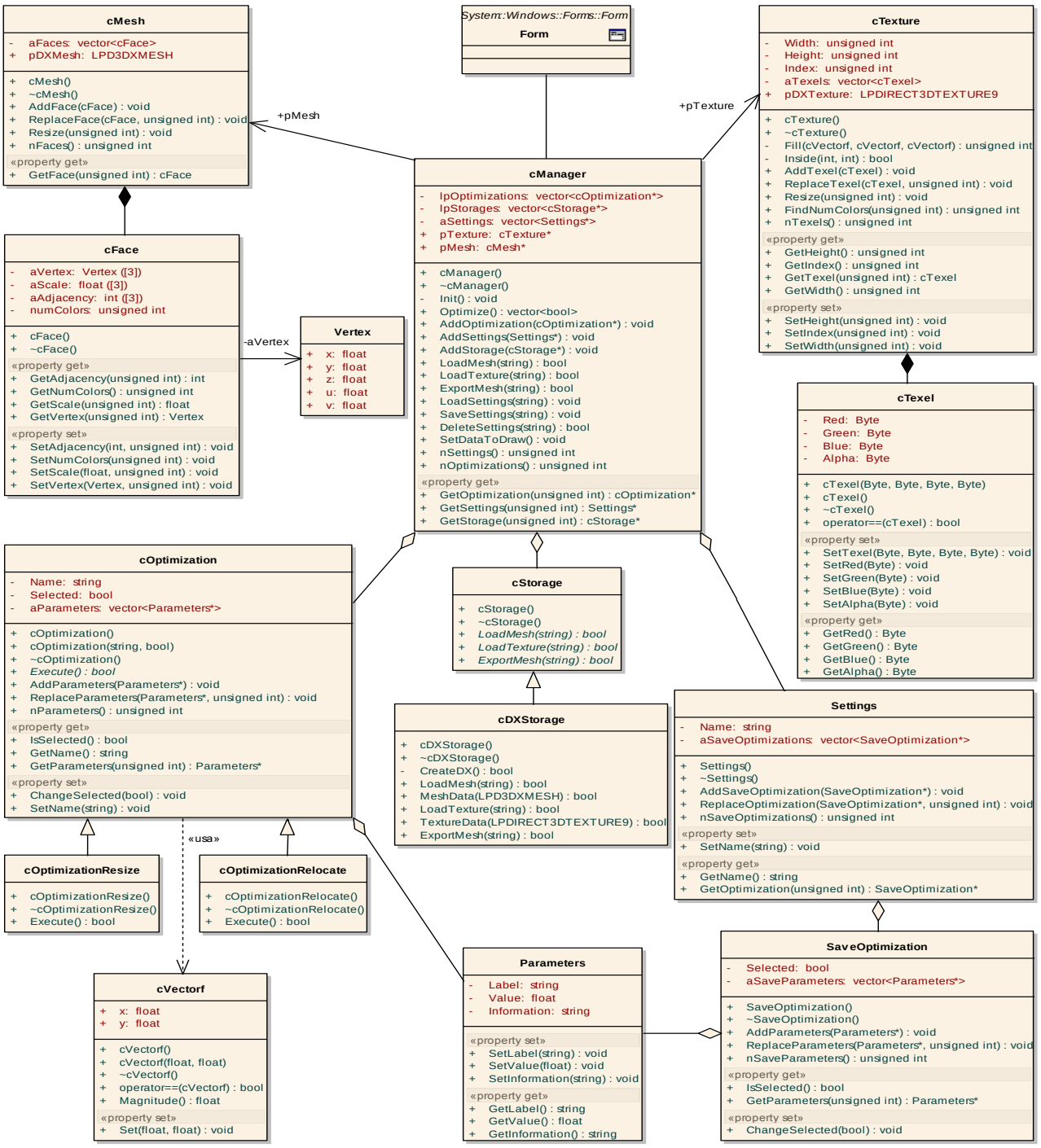
25. **Wikipedia, Colectivo de.** Heightmap. [En línea] [2007] http://en.wikipedia.org/wiki/Heightmap.

26. **Wikipedia, Colectivo de.** Mapas de Relieve y de Normales. [En línea] [2007] http://wiki.blender.org/index.php/Manual.es/PartIV/Bump_and_Normal_Maps.

27. **Wikipedia, Colectivo de.** Texture mapping. [En línea] [2007]. http://en.wikipedia.org/wiki/Texture_mapping.

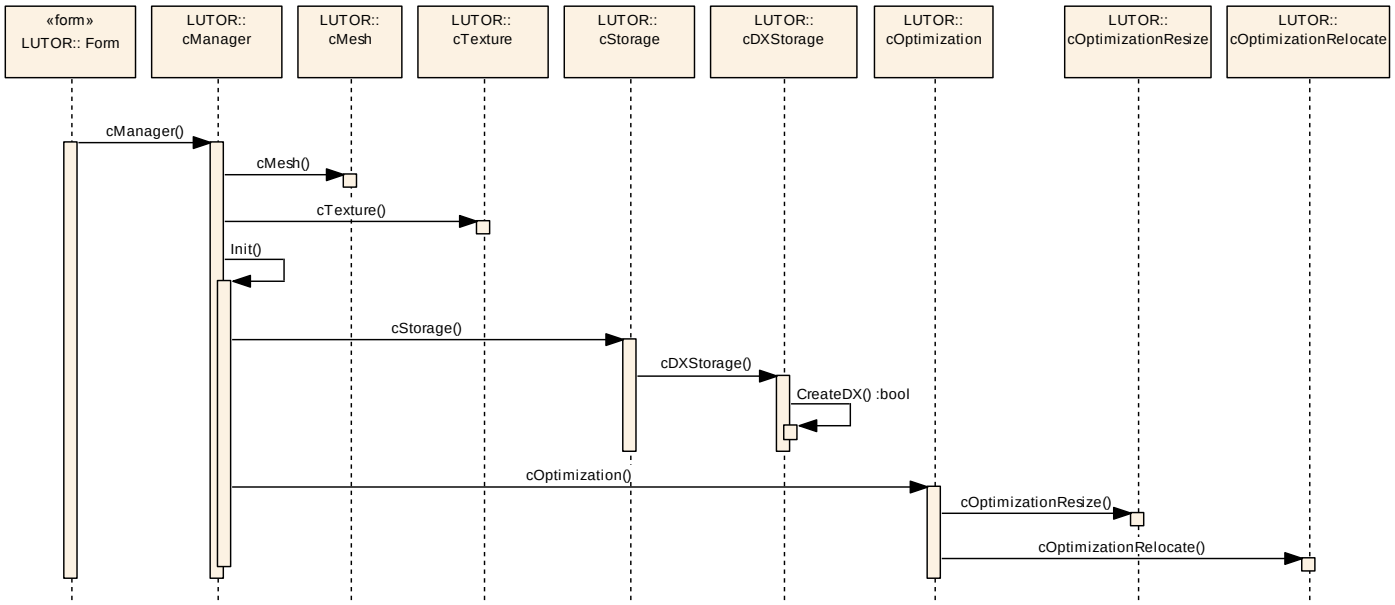
# ANEXOS

## Anexo1: Diagrama de clases

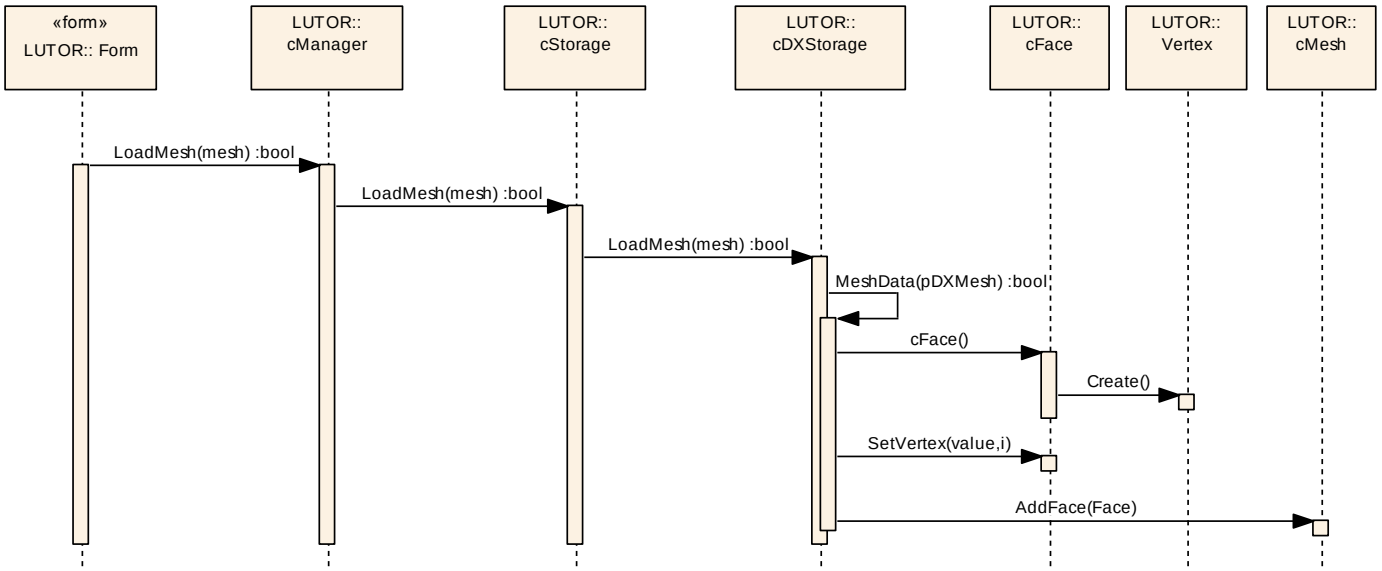


# Anexo 2: Diagramas de secuencia

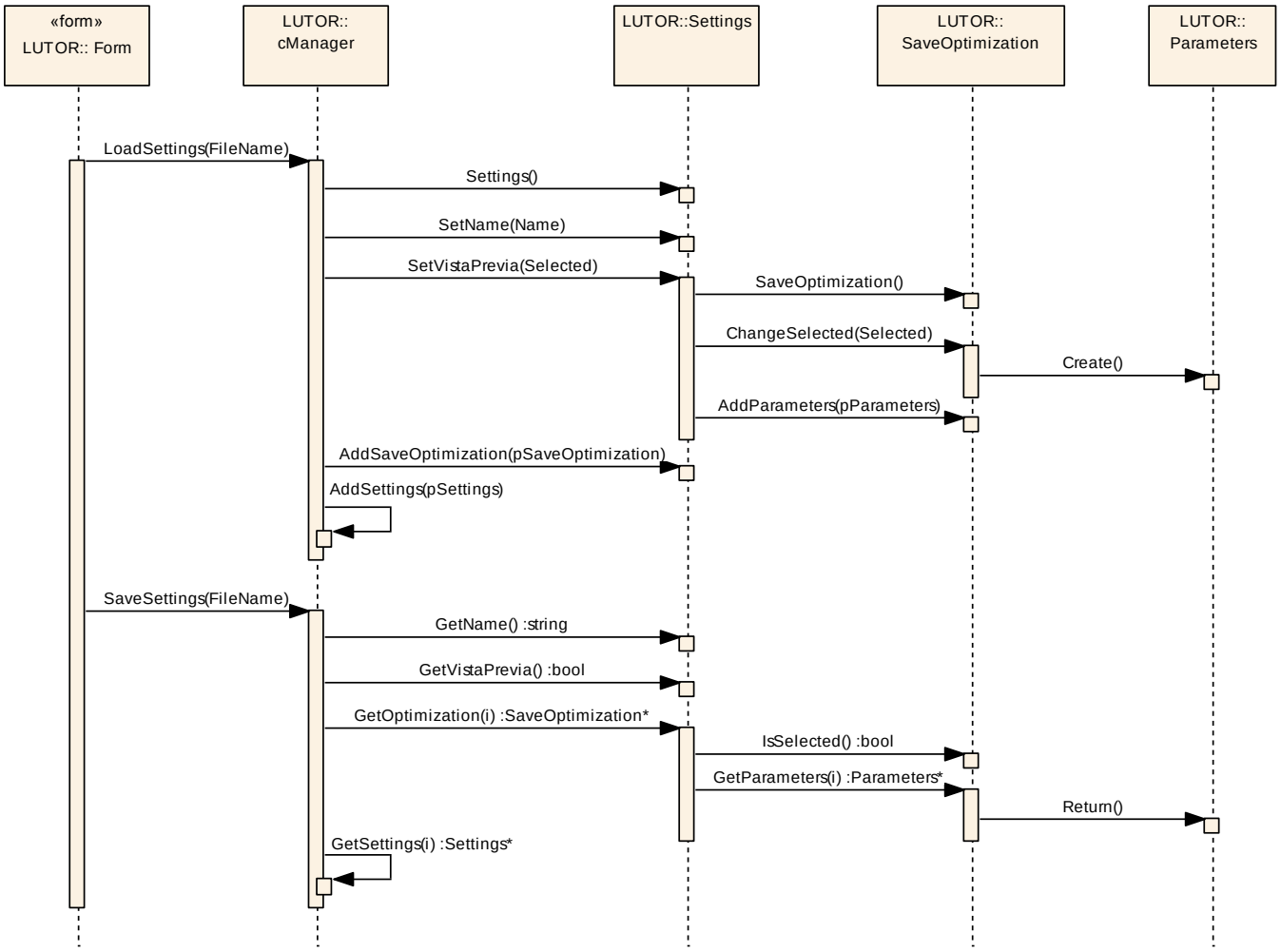
## Diagrama de secuencia de Inicializar



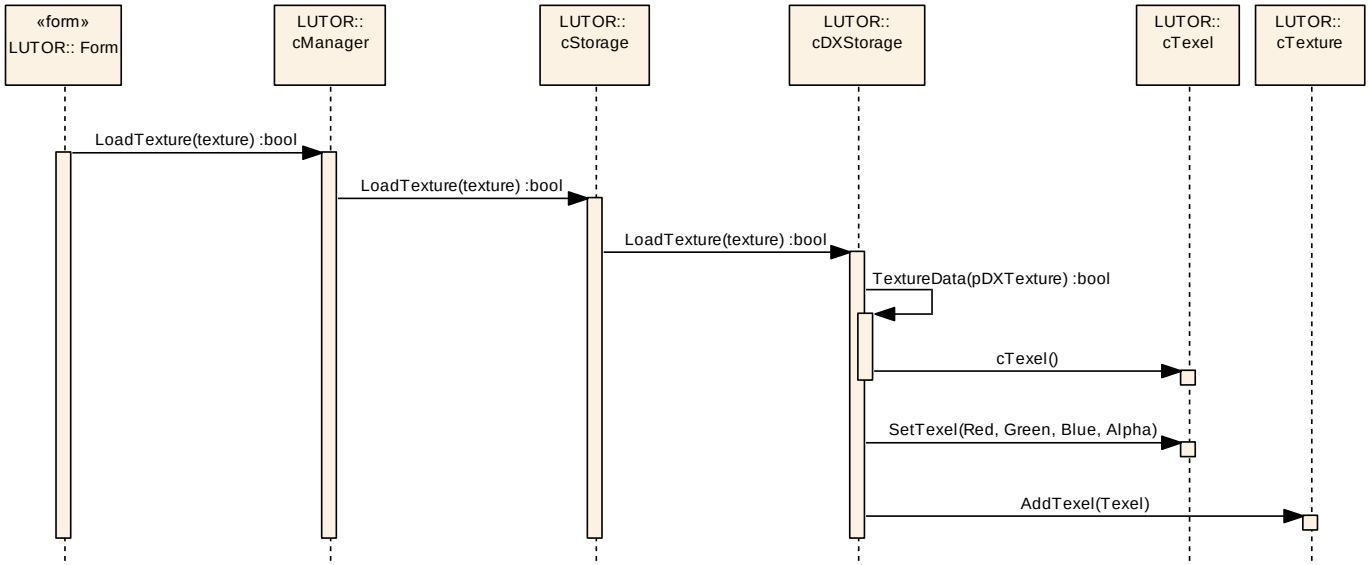

## Diagrama de secuencia de Cargar Modelo



## Diagrama de secuencia de Configuración



## Diagrama de secuencia de Cargar Textura




## Diagrama de secuencia Exportar

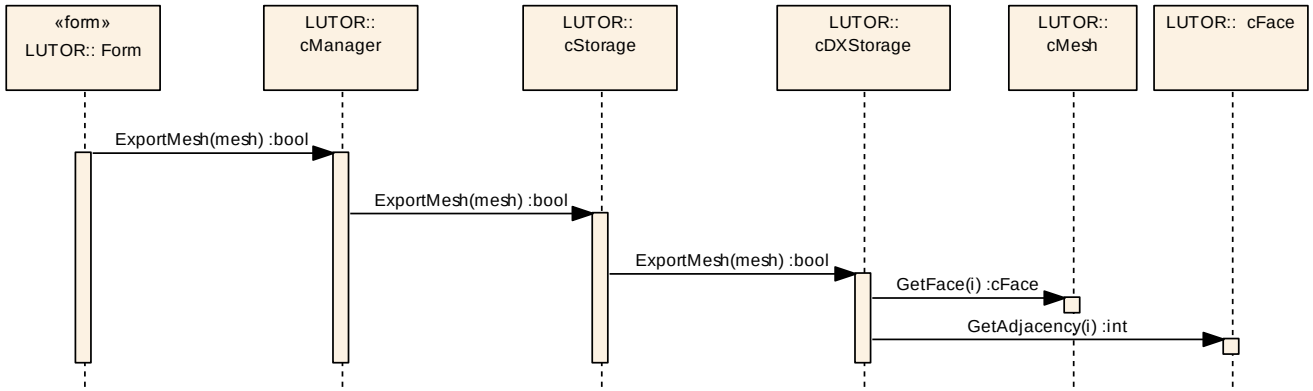
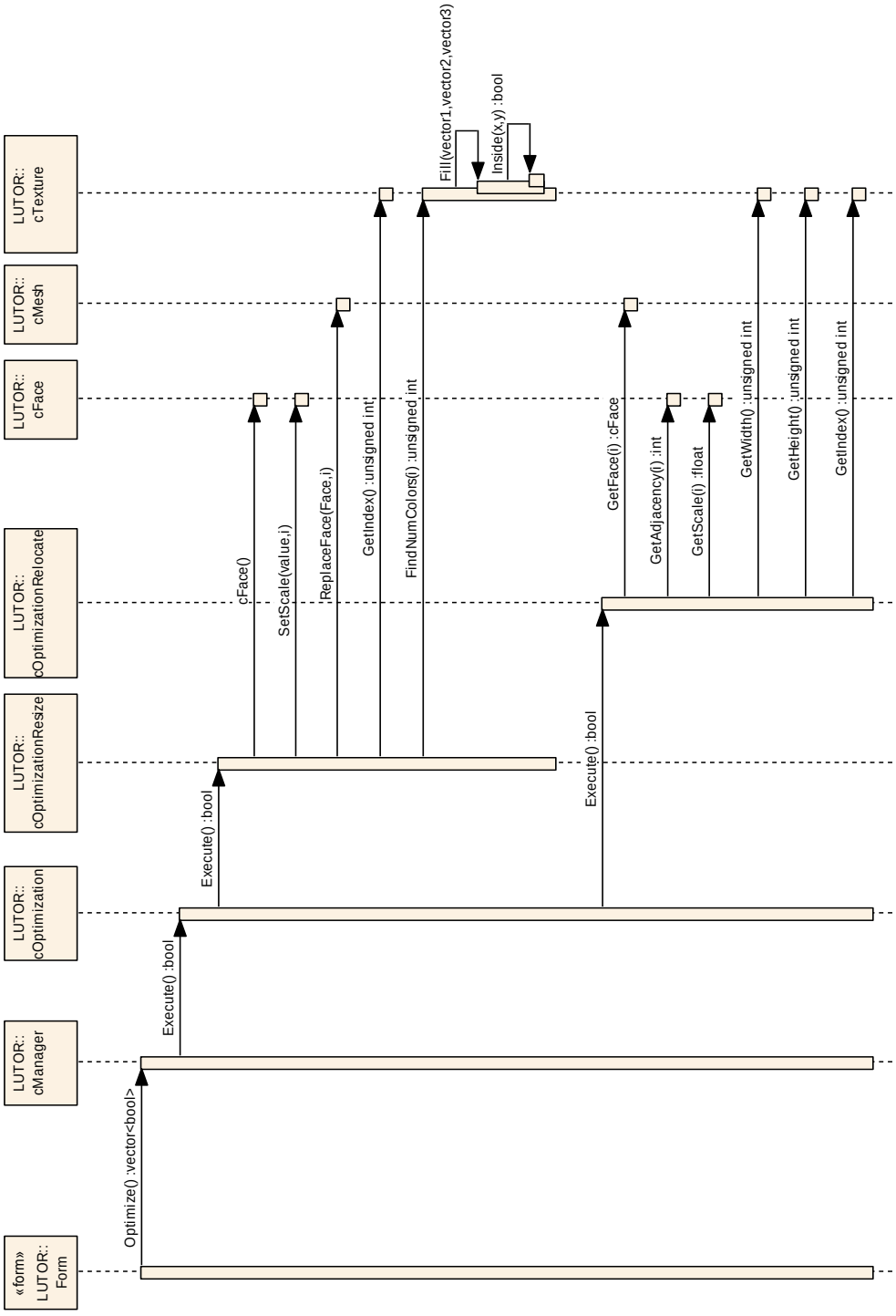
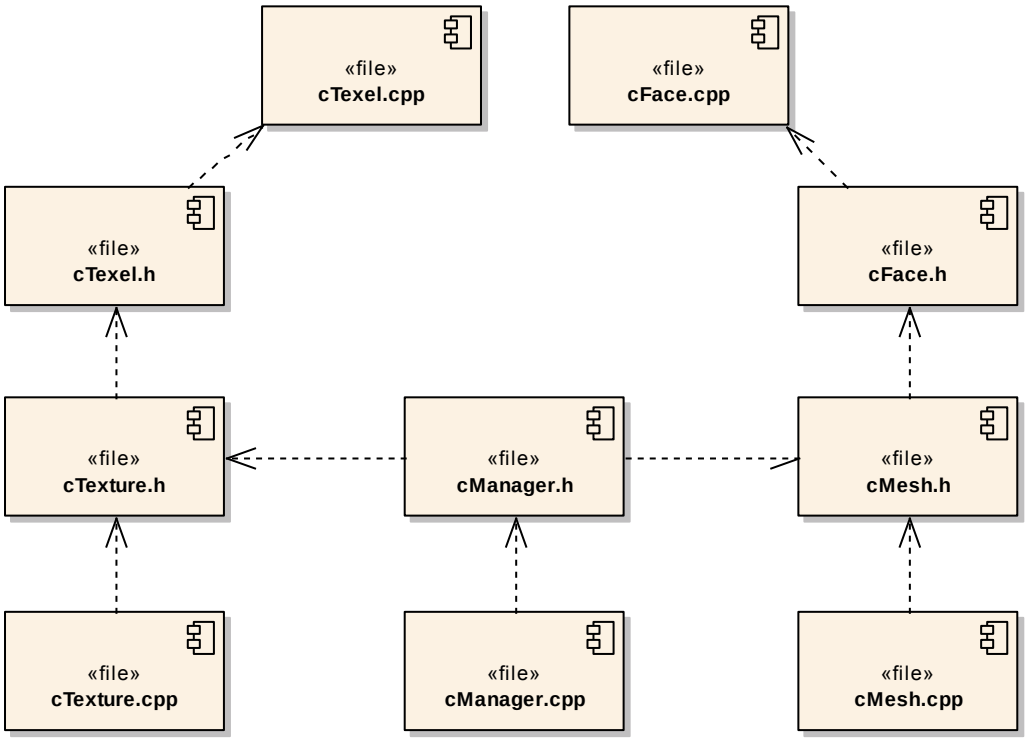


**Diagrama de secuencia de Optimizar**

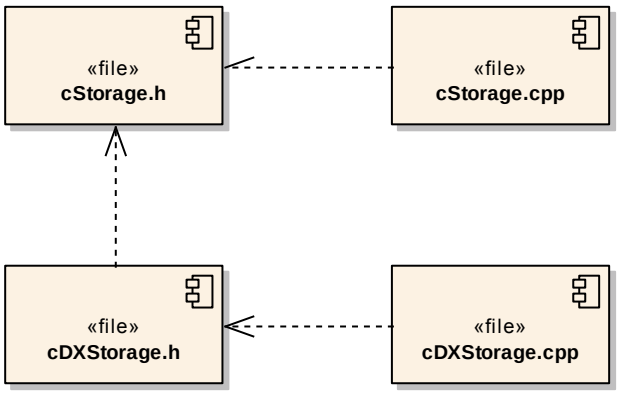


## Anexo 3: Diagramas de componentes

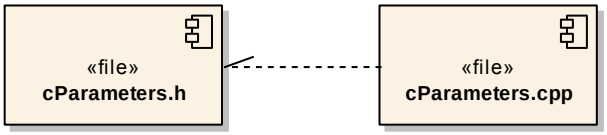
### Diagrama de componentes del paquete Manager



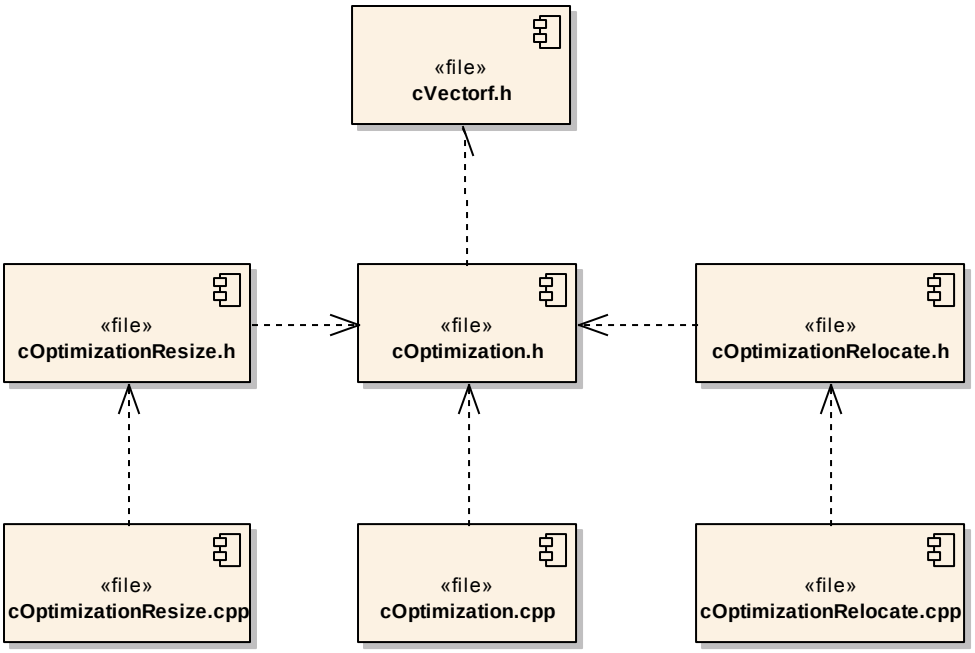

### Diagrama de componentes del paquete Storage



### Diagrama de componentes del paquete Settings




### Diagrama de componentes del paquete Optimizations



# GLOSARIO DE TÉRMINOS

***API* (Application Program Interface):** Conjunto de métodos, clases y funciones que permiten acceder a un modulo determinado. Por ejemplo, el API de UVAtlas son varias funciones que permiten acceder a todas las funcionalidades de la librería UVAtlas.

***Aplicación:*** Cualquier programa que sea ejecutado en un sistema operativo y que realice una función específica para un usuario u otro programa.

***Checkbox:*** es un control que cuando es seleccionada por el usuario, demuestra que se ha permitido una característica particular.

***Coordenadas de texturas:*** Son los valores que definen cual punto de la textura se relaciona con el vértice del modelo tridimensional. Estos valores son usados para rasterizar la textura en el polígono y definir la relación que existe entre cada triángulo y el espacio correspondiente en la textura.

***Eficiencia de la textura:*** Aprovechamiento del espacio de textura, es decir, porciento de espacio de textura ocupado.

***Espacio de textura:*** Región de la textura delimitada por un conjunto de coordenadas de texturas. De esta forma un triángulo está relacionado con la región de la textura definida por las coordenadas de textura de cada vértice del triangulo.

***Filtrado de textura:*** Proceso que a la hora de visualizar un modelo 3D asigna a cada píxel de la pantalla un valor de color en la textura. Existen diversos tipos de filtrado para disminuir la texelación.

***Framework:*** Conjunto de clases que constituyen la base de una aplicación.

***Interfaz de usuario***: Son las ventanas, cuadros de diálogos y otros componentes visuales que permiten al usuario comunicarse con el programa.

***Mapa de Altura:*** Textura Procedural de Pre-cálculo en la que cada texel contiene información sobre la distancia a la que se encuentra el modelo de alta resolución en comparación con el de baja resolución en

el polígono asignado a ese espacio de la textura. Son usadas para definir distorsiones en la superficie que dan sensación de grietas o elevaciones.

***Mapa de Luces:*** Textura Procedural de Pre-cálculo en la que cada texel contiene información sobre cuan iluminado se encuentra el segmento del polígono asignado a ese espacio de la textura.

***Mapa de Normales:*** Textura Procedural de Pre-cálculo en la que cada texel contiene información sobre la normal de la superficie polígono asignado a ese espacio de la textura.

***Mapeado:*** Asignación de coordenadas de textura a cada vértice del modelo.

***Mesh o modelo 3D:*** Objeto tridimensional compuesto por un conjunto de triángulos, que a su vez se encuentran compuestos por vértices y aristas.

***Plugin:*** Aplicación que interactúa con otra para aportarle una función o utilidad específica. Se utilizan como una vía de expandir programas de forma modular, de manera que se puedan añadir nuevas funcionalidades sin afectar a las ya existentes, ni complicar el desarrollo del programa principal.

***Polígono:*** Figura geométrica definida por un conjunto de al menos 3 vértices.

***Programa****:* Es la unión de una secuencia de instrucciones que una computadora puede interpretar y ejecutar.

***Rasterizar:*** convertir un modelo o imagen vectorial en una foto computarizada.

***Renderizar:*** proceso de generar una imagen en 3D o una animación en 3D a partir de un modelo, usando una aplicación de computadora.

***Ruido:*** Perturbación al azar de una superficie basada en la interacción de colores o materiales.

***Software:*** Conjunto de programas y procedimientos necesarios para hacer posible la realización de tareas específicas.

***Texel* (Tex**ture **El**ement**):** Unidad mínima de una textura aplicada a una superficie.

***Textura:*** Colección o arreglo n-dimensional de texeles.

***Texturas Procedurales de Pre-cálculo:*** Texturas creadas mediante algoritmos matemáticos que almacenan en los componentes RGBA valores pre-calculados de los modelos 3D. Ejemplo de estos valores son la iluminación, las normales, la radiosidad, entre otros.

***Texturas Procedurales:*** Texturas creadas mediante algoritmos matemáticos.

***Texturizar:*** Proceso de creación y aplicación de texturas a un objeto.

***UVAtlas:*** Librería para generar automáticamente coordenadas de texturas únicas en modelos tridimensionales, maximizando el uso del espacio de textura y minimizando la elongación.

***Vértice:*** Punto común entre dos aristas de la malla.

***Wizard o asistente:*** Cualquier programa intuitivo que guía paso a paso al usuario para realizar una tarea.