Facultad 5

# *"Gestor de bibliotecas de script en Python para el desarrollo de Paseos Virtuales sobre Blender"*

**Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas.**

**Autores: Fernando Javier Mesa Quiles**
**Roger Terrero Hernández.**

**Tutor: Ing. Minardo Gollún González López**

**Ciudad de La Habana**
**Junio 2010**

**DECLARACIÓN DE AUTORÍA**

**Declaramos que somos los únicos autores de este trabajo y autorizamos a la Facultad 5 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.**

**Para que así conste firmo la presente a los 02 días del mes de julio del año 2010.**

_____ _____

Fernando Javier Mesa Quiles Roger Terrero Hernández

_____

Tutor: Minardo Gollún González López

## Datos de Contacto:

**Tutor: Minardo Gollún González López.**
**Institución: Universidad de las Ciencias Informáticas.**
**Título: Ingeniero en Ciencias Informáticas.**
**Correo electrónico: mgonzalezl@uci.cu**

## Opinión del Tutor

**Ing. Minardo Gollún González López**

Firma

Fecha

## Dedicatoria

*A todas las personas que me quieren y confiaron en mí, en especial a mi madre.*

**Fernando Javier Mesa Quiles**

*A mi madre y mi padre que se lo debo todo y a toda mi familia que ha estado al tanto de cada paso que doy en mi carrera.*

**Roger Terrero Hernández**

## Agradecimientos

*Quiero agradecerle a todas las personas que hicieron posible que se realizara esta investigación; particularmente a Yoan González por su ayuda incondicional, a nuestro tutor Minardo González, a mi papá, a Rubén que ha sido como un padre para mi, a mi novia Dalila por apoyarme siempre a pesar de estar separados por cosas de la vida, a mi hermana Chabely por estar siempre conmigo, y un agradecimiento especial a la persona más importante de mi vida, mi madre, por ser incondicional y especial en su amor hacia mí.*

***Fernando Javier Mesa Quiles***

*Agradezco ante todo a mi madre y a mi padre por toda la confianza y el amor que me han brindado siempre, a mi novia por todo su amor, a toda mi familia en general que se sienten muy orgullosos de mi en cada paso triunfante que logro dar en mi vida profesional y personal.*
*Hago llegar también de esta forma mis grandes y sinceros agradecimientos a todas aquellas personas que de una u otra forma estuvieron al tanto de nuestro trabajo y nos brindaron gran parte de su tiempo a apoyarnos en todo lo que fuera necesario en el transcurso de estos 5 años de universitarios y en la realización del nuestro trabajo de diploma como fruto de la amistad que surgió desde el primer año de esta carrera.*

***Roger Terrero Hernández***

## Resumen:

En el trabajo se describe detalladamente el proceso de desarrollo de la herramienta “Gestor de bibliotecas de script en Python para el desarrollo de Paseos Virtuales sobre Blender”. Como parte del proceso investigativo se muestra un estudio de los diferentes tipos de bibliotecas y gestores existentes, además se resume el estado del arte relacionado con las herramientas, metodologías y conceptos asociados al trabajo. Se presenta un estudio realizado para lograr agrupar los diferentes scripts en bibliotecas que a su vez se clasificarán según su fin. Se muestra un análisis de los resultados alcanzados y las aplicaciones de la herramienta propuesta.

**Palabras clave:** Blender, Biblioteca, Scripts, Python, Paseos Virtuales.

## *Abstract:*

*This paper describes in detail the process of developing the tool library manager script in Python to develop walkthrough on Blender. As part of the investigative process shows a study of the different types of libraries and existing managers also shows the state of the art related to tools, methodologies and concepts associated with the work. We present a study group to achieve different script libraries, which in turn are classified according to their purpose. It presents an analysis of the results achieved and the proposed tool applications.*

***Key words**: Blender, library, script, Python, walkthrough.*

# ÍNDICE

# ÍNDICE DE FIGURAS

## ÍNDICE DE TABLAS

# Introducción

La creación por el hombre del primer ordenador ha sido un comienzo importante para el desarrollo de las tecnologías, desde entonces surgió un nuevo término, Informática. Este sector se ha ido desarrollando paulatinamente de un modo considerable, en sus inicios no se imaginaba que todo o casi todo fuera controlado por las computadoras.
Hoy en día se realizan cosas increíbles mediante software creados por personas ingeniosas y por qué no decirlo, futuristas. Muchas de estas herramientas dieron pie al surgimiento hace más de dos década de la ”Realidad Virtual”, que básicamente es un sistema o interfaz informático que genera entornos sintéticos en tiempo real, representación de las cosas a través de medios electrónicos o representaciones de la realidad, una realidad ilusoria, pues se trata de una realidad perceptiva sin soporte objetivo, sin red extensa, ya que existe sólo dentro del ordenador, además es un sistema de computación usado para crear un mundo artificial donde el usuario tiene la impresión de estar en ese mundo y la habilidad de navegar y manipular objetos en él.

*Existen numerosos conceptos asociados con Realidad Virtual, entre ellos:*

*"La realidad virtual es un sistema de computación usado para crear un mundo artificial donde el usuario tiene la impresión de estar en ese mundo y la habilidad de navegar y manipular objetos en él".* [Michael Louka 1998]

*"La realidad virtual te permite explorar un mundo generado por computadoras a través de tu presencia en él".* [Michael Louka 1998]

Un mundo virtual es un modelo matemático que describe un "espacio tridimensional"; dentro de este "espacio" están contenidos objetos que pueden representar desde una simple entidad geométrica, por ejemplo un cubo o una esfera, hasta una forma compleja, como puede ser un desarrollo arquitectónico, un nuevo estado físico de la materia ó el modelo de una estructura genética. Se trata, en definitiva, de un paso más allá de lo que sería la simulación por computador, tratándose realmente de la simulación interactiva, dinámica y en tiempo real de un sistema.

En la Universidad de las Ciencias Informáticas (UCI) se desarrolla el Proyecto Paseos Virtuales, donde para la creación de escenarios y objetos virtuales se utiliza la herramienta Blender, la cual es multiplataforma, dedicada al modelado, animación y creación de gráficos tridimensionales. Cuenta con un intérprete de Python por lo que se puede programar mediante scripts, siendo esto una ventaja impresionante para el Proyecto.

A raíz del surgimiento y la evolución de las nuevas técnicas y simulaciones insertadas en el grupo de trabajo de Paseos Virtuales, se han utilizados diferentes tipos de scripts para interactuar y diseñar escenas en 3D que simulan entornos virtuales diversos y complejos. El grupo de desarrollo necesita para su trabajo cotidiano una gran cantidad de script, contando para esto con un cúmulo de ficheros .blend (formato de los ficheros exportados desde la herramienta Blender) y .py(formato de los ficheros exportados por la herramienta Python) que se encuentran dispersos y sin clasificar, provocando en varias ocasiones la implementación de algunos sin ser necesarios ya sea porque no se tienen y ya están implementados, o porque se cuenta con ellos pero no se está consciente. Haciendo del proceso de selección una tarea que consume tiempo que podría ahorrarse si se contara con una solución que los gestionara eficientemente, así surge la idea de unirlos en una biblioteca, para facilitar el uso de estos y crear una herramienta de gestión de bibliotecas que permita seguir enriqueciendo el arsenal de script disponibles.

Debido a la **situación problémica** antes expuesta se llegó al siguiente **problema científico**: ¿Cómo asegurar la disponibilidad inmediata de los scripts necesarios para el desarrollo de Paseos Virtuales?

Por lo cual el **objeto de estudio** planteado sería: Las herramientas de gestión y edición de scripts para el desarrollo de aplicaciones.

Por lo que se escoge como **campo de acción:** Las herramientas de gestión y edición de scripts desarrollados en Python para la creación de Paseos Virtuales sobre Blender.

Definiendo el siguiente **objetivo general:** Desarrollar una herramienta de gestión para bibliotecas de scripts que permita agilizar el proceso de desarrollo de Paseos Virtuales.

**Tareas**

- Uso de los lenguajes y entornos de desarrollo definidos.

- Clasificación de los scripts con que cuenta el Grupo de Desarrollo.
- Recopilación de los scripts no implementados por el Grupo de Desarrollo.
- Realización del análisis y diseño del software a través de la metodología seleccionada.
- Desarrollo de una biblioteca que contenga de manera organizada todos los scripts seleccionados anteriormente.
- Adición a la herramienta de al menos cinco scripts.

### Métodos Científicos Usados:

El método científico es el conjunto de estrategias que usan los científicos para desarrollar su función, es decir, hacer ciencia, por lo que para darle solución a las tareas asignadas se utilizaron los siguientes métodos de investigación:

### Métodos teóricos

Para logra un mejor desempeño en la realización de la herramienta se utilizó el método teórico **analítico-sintético** para poder guiarse por el análisis de las teorías, documentos, libros, artículos, etc. Permitiendo el procesamiento de la información para arribar a diferentes conclusiones prácticas y teóricas a cerca del trabajo.

Para el estudio de las herramientas relacionadas con el trabajo de diploma se utilizó el método teórico **histórico-lógico** con el objetivo de conocer acerca de la evolución del software Blender y del lenguaje Python, así como de las aplicaciones gestoras y bibliotecas existentes.

### Método Empírico

Como medio de obtener mayor información utilizamos como método empírico **la entrevista** ya que tuvimos que entrevistar a programadores expertos para lograr entender cómo es que funciona el lenguaje de programación definido para la herramienta, además de entrevistar a personas con conocimientos de diseño para lograr una interfaz amigable con el usuario y también a personas con conocimientos a cerca de la realización de documentos de trabajos de diplomas.

También usamos este medio para entrevistar a la mayoría de los integrantes del grupo de desarrollo para ver cuáles son los problemas reales que enfrentan a diario para poder obtener un producto que sea de gran ayuda y optimice el trabajo del grupo.

El presente trabajo está estructurado de la siguiente manera: Introducción, cuatro capítulos de contenido, Conclusiones, Recomendaciones, Bibliografía, Referencias Bibliográficas y Glosario de Términos.

A continuación se hace una breve descripción del contenido de los capítulos.

- Capítulo 1 “Fundamentación Teórica”, en este capítulo se presenta un análisis de las principales herramientas con las cuales trabaja nuestra facultad en la elaboración de los diferentes proyectos de diseño, se hace un estudio de las principales bibliotecas y gestores de bibliotecas existentes hasta este momento. Se caracterizan algunos de los diferentes lenguajes de modelado y de programación existentes.
- Capítulo 2 “Características del sistema”, en este capítulo se decide cuales son las herramientas, lenguajes de modelado y de programación y metodología de desarrollo para la realización de la herramienta. También se definen los requerimientos de entrada que debe presentar la herramienta para su funcionamiento y se mencionan las principales ventajas que tiene la aplicación.
- Capítulo 3 “Análisis y diseño del sistema”, en este capítulo se hace el levantamiento de los requisitos funcionales y no funcionales de la aplicación, también se realiza la descripción de los casos de usos y sus respectivos diagramas de secuencia. Se muestran las principales interfaces que la herramienta posee.
- Capítulo 4 “Implementación y Prueba”, en este capítulo se presentan los diferentes estándares de implementación con los cuales se trabajo en la realización de la aplicación, también se presenta el diagrama de despliegue de nuestro sistema y se muestran las principales pruebas realizadas al software.

# Capítulo 1:

# Fundamentación Teórica

En el presente capítulo se realizará un análisis de las herramientas, tecnologías y metodologías propuestas para desarrollar el trabajo, además se realiza un estudio del arte relacionado con la investigación y se muestran los conceptos fundamentales de bibliotecas, gestores y scripts estudiados. También se presentan las metodologías de desarrollo y lenguajes de programación estudiados para el desarrollo del sistema.

A continuación se presentan el estudio de las principales herramientas de diseño que son utilizadas en nuestra facultad para la realización de los diferentes proyectos que pueden llegar a ser desde un simple diseño en 3D a un complejo proyecto de Realidad Virtual.

**Herramientas de diseño.**

### 1.1.1 Blender

Herramienta de renderizado y diseño 3D, como Maya o 3ds Max, pero gratuita [sitio oficial Blender 2008]. Es sumamente completa ya que permite modelar, texturizar, animar, renderizar, simular efectos de partículas, editar, sincronizar audio y video en forma no-lineal y crear aplicaciones 3D como videojuegos. Originalmente desarrollado por la compañía 'Not a Number' (NaN), el programa inicialmente fue distribuido de forma gratuita pero sin el código fuente, con un manual disponible para la venta aunque posteriormente pasó a ser software libre. Blender es ahora desarrollado como 'Software Libre', con el código fuente disponible bajo la licencia GNU GPL. [Motores de Juego 2007]

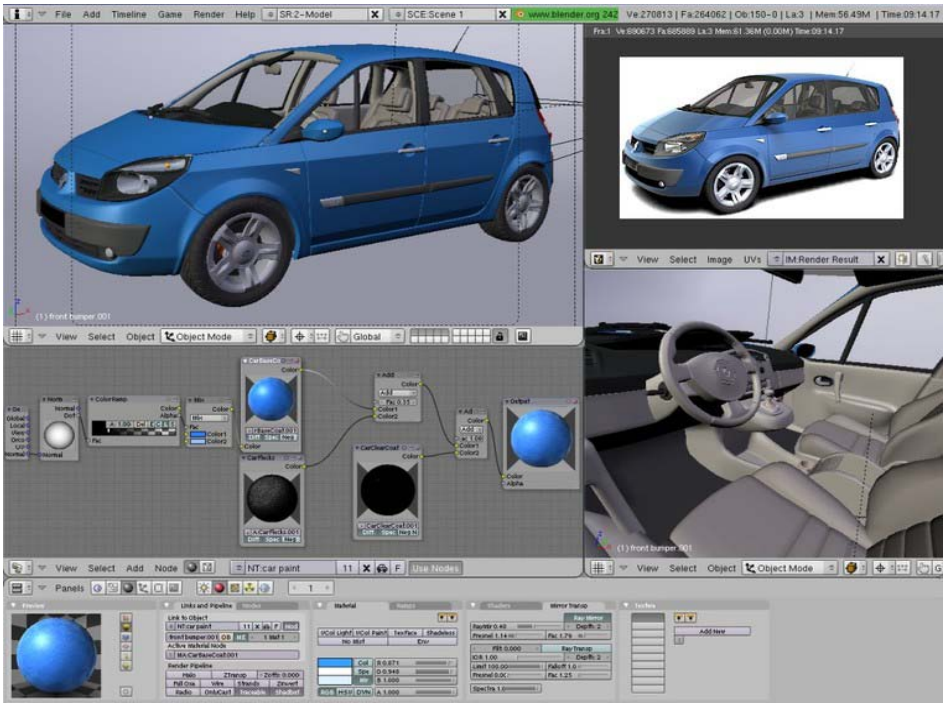


**Figura 1 Interfaz de la herramienta Blender**

Características de Blender.

Tiene una característica muy poderosa que a menudo es pasada por alto. Esta característica es un intérprete de Python totalmente funcional. Esto le permite a cualquier usuario añadir funcionalidades a Blender escribiendo algún script de Python.

Posee un paquete de creación totalmente integrado, ofreciendo un amplio rango de herramientas esenciales para la creación de contenido 3D. Es multiplataforma, con una interfaz unificada para todas las plataformas basada en OpenGL, listo para ser usado en todas las versiones de Windows (98, NT, 2000 y XP), Linux, OSX, FreeBSD, Irix y Sun, y otros sistemas operativos.

- Cuenta con una arquitectura 3D de alta calidad permitiendo un rápido y eficiente desarrollo.
- Ofrece herramientas de simulación avanzada tales como dinámicas de cuerpos rígidos, de cuerpos suaves y de fluidos, herramientas de animación de personajes, un sistema de simulación física avanzado, y un sistema de composición de materiales basado en nodos.

- Posee un motor de 3D en tiempo real que permite la creación de contenido interactivo que puede ser reproducido independientemente.
- Tiene capacidad para una gran variedad de primitivas geométricas, incluyendo curvas, mallas poligonales, vacíos, NURBS (modelo matemático para representar curvas y superficies) y metaballs (nombre de una técnica de gráficos realizada por el ordenador para simular interacción orgánica entre diferentes objetos n-dimensionales).
- Junto a las herramientas de animación se incluyen cinemática inversa, deformaciones por armadura o cuadrícula, vértices de carga y partículas estáticas y dinámicas.
- Posibilita el renderizado interno versátil y la integración externa con el potente trazador de rayos o "raytracer" libre de YafRay (motor de Render que trata de imitar la iluminación real, genera imágenes bastante realistas).
- Presenta modificadores apilables, para la aplicación de transformación no destructiva sobre mallas.
- Cuenta con un sistema de partículas estáticas para simular cabellos y pelajes, al que se han agregado nuevas propiedades entre las opciones de shaders (tecnología escrita en un lenguaje de sombreado que se puede compilar independientemente) para lograr texturas realistas.

### 1.1.2 Maya

Es una herramienta para el trabajo con animaciones y gráficos 3D. Se caracteriza por su potencia, posibilidades de expansión y la fácil personalización de su interfaz. Dispone de una extensa gama de técnicas específicas para la animación de personajes. Con Maya, el animador puede controlar el comportamiento de la piel, incluso en las partes más complicadas del cuerpo, como los hombros. Sus deformadores permiten cambiar la forma de un objeto o personaje, además de integrar sincronización labial, o cualquier otro tipo de movimiento que precise sincronización con el sonido. MEL (Maya Embedded Language) es el código que forma el núcleo de maya, y gracias a él se pueden crear scripts que aumentan la potencia del software, y permiten su personalización. En el 2005 se convirtió en el primer y único software que ha obtenido un Oscar, gracias al enorme impacto que ha tenido en la industria cinematográfica como herramienta de efectos visuales a pesar de ser un software propietario.

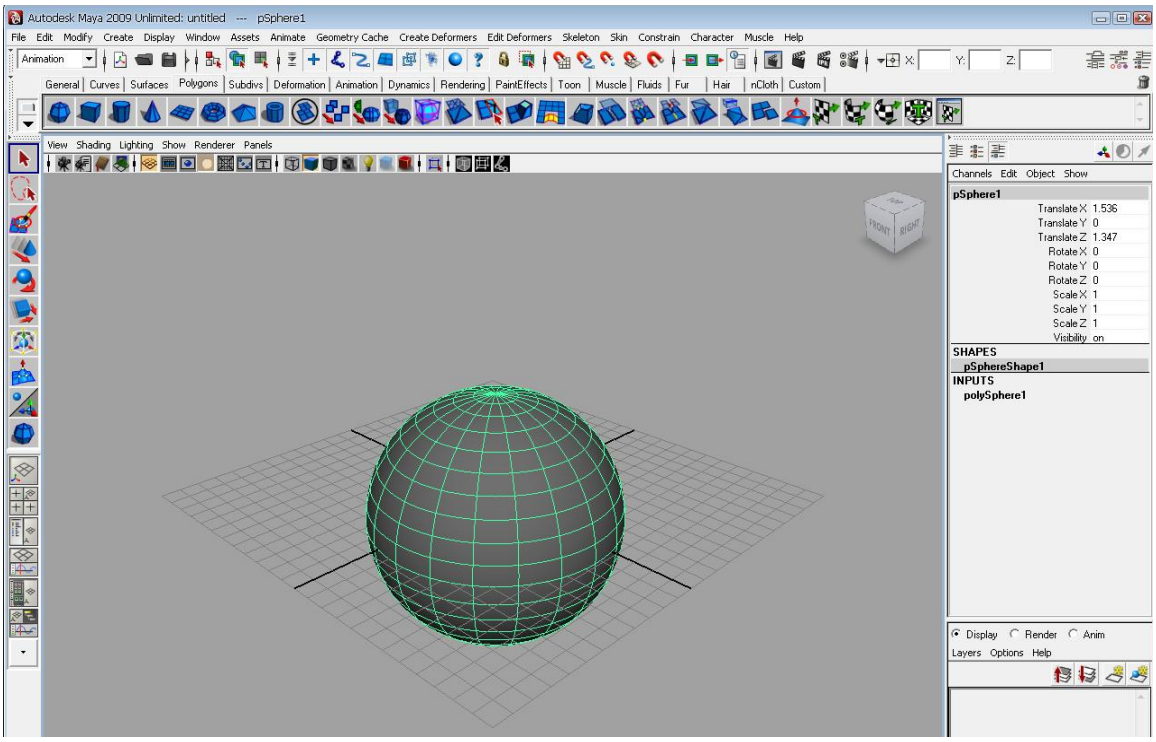



**Figura 2 Interfaz de la herramienta Maya**

Características de Maya

- Hace que sea más fácil para los artistas, diseñadores y entusiastas del 3D crear experiencias de entretenimiento convincente, diseños estilísticos y evocadores de imágenes digitales de la foto real de efectos visuales creíbles a los personajes.
- Posee una gran potencia de diseño y las posibilidades de expansión, personalización de su interfaz y herramientas.
- MEL (Maya Embedded Language) es el código que forma el núcleo de Maya, y gracias al cual se pueden crear scripts y personalizar el paquete.
- Posee diversas herramientas para modelado, animación, Render, simulación de ropa y cabello, dinámicas (simulación de fluidos), etc.

### 1.1.3 3D Studio Max

Algunas veces llamado 3ds Max o simplemente Max, es uno de los programas de animación 3D más utilizado y respetado en todo el mundo. Es un programa de creación de gráficos y animación 3D desarrollado por Autodesk, en concreto la división Autodesk Media & Entertainment (anteriormente Discreet). Es uno de los más sencillos para iniciarse en el mundo de la animación 3D para la creación de video, juegos y multimedia. Su arquitectura abierta, su

baja curva de aprendizaje y sus potentes herramientas lo convierten en uno de los programas líderes del diseño y la animación 3D en infinidad de ámbitos, como: arquitectura, publicidad, televisión y video, cine, artes escénicas, desarrollo de juegos, etc. Dispone de una sólida capacidad de edición y una dinámica arquitectura de plugins. Los profesionales del diseño y la animación trabajan con este programa para crear desde juegos de ordenador hasta escenas cinematográficas con espectaculares efectos especiales, anuncios, simulaciones, etc.

Permite además exportar y salvar a varios formatos, incluso diferentes de los que trae por defecto la herramienta. Este es un software propietario por lo tanto se dificulta su licencia de uso por su elevado costo.

Dentro de los plugins que se le pueden agregar a esta herramienta se encuentra el ogreMax, este brinda una serie de opciones como son: exportar los modelos con sus animaciones para poder cargarlo en el Engine de Ogre. También permite ver en tiempo real como se verá una escena sin tener que abrir el Engine de Ogre.

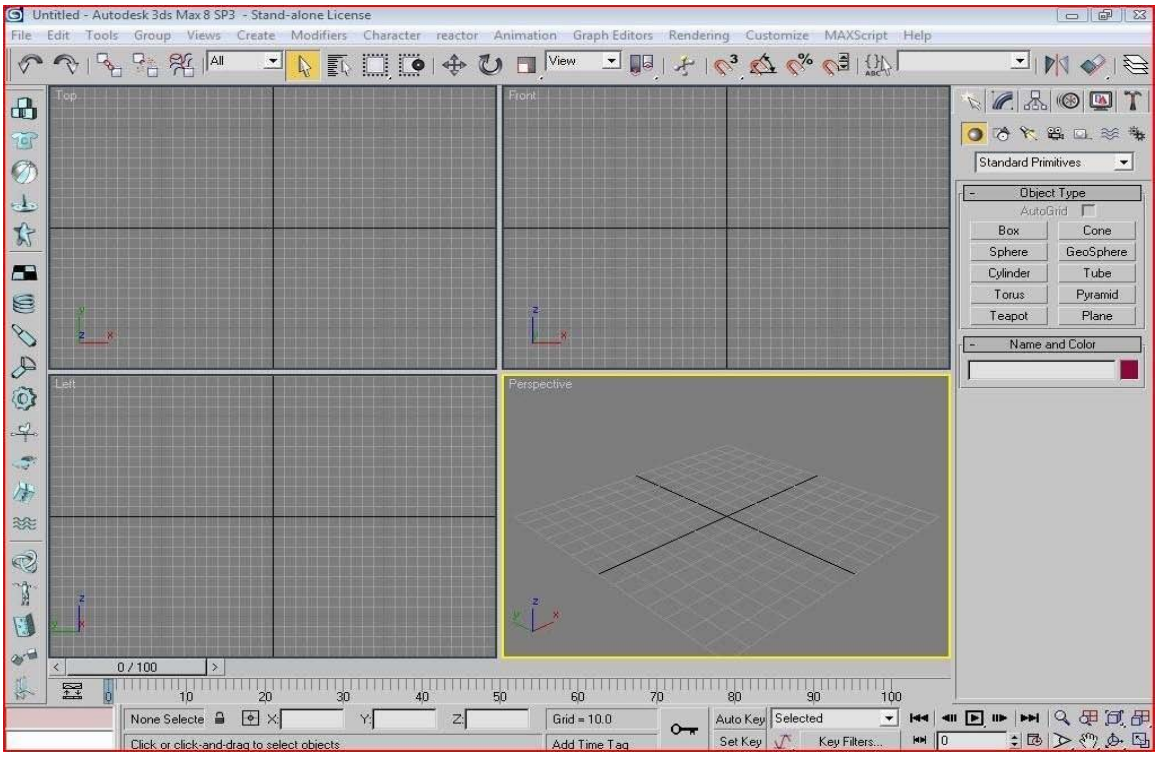



**Figura 3 Interfaz de la herramienta 3D Studio Max**

Características de 3D Studio Max.

- Ofrece a los animadores nuevas y poderosas herramientas de flujo de trabajo creativo incluyendo las herramientas más completas de animación avanzada para personajes


Autores: Fernando Javier Mesa Quiles
Roger Terrero Hernández

actualmente disponibles, los efectos visuales más aclamados y soporte para entornos de producción de próxima generación.

- Ofrece a los usuarios desempeño y escalabilidad mejorados para manejar grupos de datos extremadamente grandes, las complejas tareas de modelado 3D y de animaciones a las que los estudios se enfrentan hoy en día.

### 1.2 Motor de Juego.

Es un conjunto de librerías que permiten el diseño, creación y representación de un videojuego con el propósito de simular una parte de la realidad, reflejando características propias del mundo como la física, gravedad, colisiones, sonidos, etc. [Motores de Juego 2007]

Habitualmente un motor de juego está compuesto por los siguientes componentes

- Motor de física.
- Motor de audio.
- Motor de Render.
- Inteligencia artificial.

#### 1.2.1 Ejemplos de motores de juegos.

- **Panda 3D:** Panda3D es un motor gráfico de escenas. Esto significa que el mundo virtual es inicialmente un espacio cartesiano vacío en el cual el programador de videojuegos inserta los modelos en 3D. Panda 3D no distingue entre modelos 3D "grandes", como el modelo de una mazmorra o una isla, y modelos 3D "pequeños", como el modelado de una mesa o una espada. Ambos modelados grandes y pequeños son creados usando un programa de modelado estándar como Blender, 3ds Max o Maya, cargados en Panda 3D, y después insertados en el espacio cartesiano.
- **Ogre 3D:** (acrónimo del inglés *Object-Oriented Graphics Rendering Engine*) es un motor de renderizado 3D orientado a escenas, escrito en el lenguaje de programación C++. Sus bibliotecas evitan la dificultad de la utilización de capas inferiores de librerías gráficas como OpenGL y Direct3D, además proveen una interfaz basada en objetos del mundo y otras clases de alto nivel.
- **Irrlicht Engine:** Es un código abierto de motor de 3D escrito en C++. Es multiplataforma, oficialmente se ejecuta en Windows, Mac OS, Linux y Windows CE y

debido a su carácter abierto los puertos a otros sistemas como la Xbox (consola de videojuegos), Play Station Portable (consola portátil de videojuegos) están disponibles.

- **3D GameStudio:** Es un conjunto de herramientas de desarrollo para la realización de juegos de ordenador. Provee de un motor 3D, un motor 2D, un editor de niveles y modelos, compilador de scripts y librerías de modelos, texturas, etc. Manipula con igual rendimiento escenas de interior y de exterior. Tiene un motor de iluminación que soporta sombras verdaderas y fuentes de luz en movimiento. El principal objetivo que esta aplicación persigue es que el creador del juego no necesite ser un programador experimentado. Abogan por reducir al máximo el esfuerzo del creador a costa de perder flexibilidad en el diseño del juego. Sin embargo, admiten que los juegos realizados con poca o ninguna programación serán siempre juegos poco ambiciosos.

- Ofrecen tres posibilidades para crear un juego:
- Juegos diseñados a base únicamente de ratón, para usuarios sin conocimientos de programación
- Juegos o efectos diseñados con algo de programación utilizando C-scripts, para el que quiere algo más.
- Juegos o efectos programados en C++ o Delphi, para programadores con experiencia.

- **Genesis3D:** Es un motor para la visualización de escenas tridimensionales en tiempo real y que permite construir aplicaciones gráficas 3D de altas prestaciones. Ha sido diseñado principalmente para la visualización de escenas de interior logrando un alto 'frame-rate' siempre y cuando estén compuestas por una cantidad moderada de polígonos. También puede ser utilizado para escenas de exterior si el diseño de las escenas se realiza tomando ciertas precauciones. Sus principales características son la detección rápida de colisiones, iluminación pre calculada y chequeo de la visibilidad. Su principal inconveniente es la visualización de escenarios exteriores sin imponer algún tipo de restricción o límite al tamaño de la escena.

### 1.2.2 Motor de Juego de Blender (BGE).

Blender posee desde Agosto del 2000, un motor de juego o “GE”, por sus siglas en inglés, el cual ha evolucionado constantemente con cada versión de esta herramienta, pero fue en el período 2008-2009 que gracias al proyecto “Apricot” se le realizaron importantes aportes al Motor de Juego. Con la realización de este proyecto de software-libre, el primero de su tipo, ya

que fue el primer juego realizado con Blender, demostró las potencialidades de esta herramienta para la realización de este tipo de aplicaciones. Las ventajas presentes en este Motor de Juego son:

Integra rápidamente los modelos realizados, es decir, después del modelado y texturizado de determinado modelo, sin previa declaración de atributos puede ser usado en el Motor de Juego.

Los objetos con multitexturas y materiales pueden ser visto de la misma forma tanto en el espacio 3d, como en el Render o después de ejecutar el Motor de Juego. Gracias al "OpenGL Shading Language" (GLSL).

Posee un Editor de interacciones que permite crear muchas acciones sin necesidad de realizar ningún script de Python, también es una de las fortalezas de este Motor de Juego, pues casi todas las interacciones se dirigen desde este Editor.

Permite observar en tiempo real las transformaciones que sufre el modelo y sus materiales, texturas y animaciones.

Posibilita trabajar con multitexturas, materiales, propiedades y modificaciones a través de nodos. También podemos utilizar características como el mapa de normales y reflexiones, con el cual conseguimos un mayor realismo en los modelos de baja cantidad de polígonos.

Los modificadores pueden ser utilizados igualmente en cuerpos blandos.

También se puede trabajar con luces dinámicas dentro de una escena, permitiendo que los modelos que se mueven o se transformen puedan reflejar una sombra realista.

El uso de la biblioteca de física "Bullet", permite mayor realismo en las interacciones físicas, como choques, caídas de objetos, etc.

Algunas desventajas:

Existen áreas de los shaders que no se encuentran plenamente desarrolladas u optimizadas.

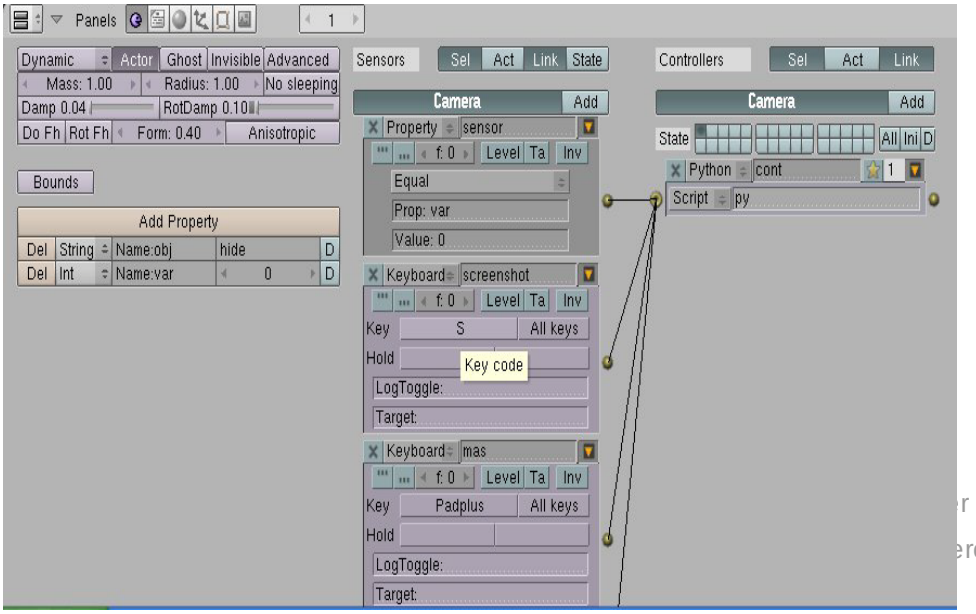


**Figura 4 Ejemplo de métodos que vienen implementados por defecto en el Game Engine**

.

### 1.2.3 Estructura del BGE.

La base de la estructura del BGE radica en lo que se llama Ladrillos de Lógica (Logic Bricks), estos proporcionan una interfaz visual fácil de utilizar para diseñar aplicaciones interactivas sin necesidad de tener conocimiento de lenguajes de programación.

Existen tres tipos de Ladrillos de Lógica:

**Sensores:** Inician toda las acciones de la lógica, los sensores sienten estímulos, como por ejemplo la cercanía de un objeto, presionar el teclado, el movimiento del mouse, etc. Cuando se activa un sensor, el pulso con la información es enviada al controlador vinculado.

**Controladores:** Manejan la lógica, evalúan los pulsos enviados desde los sensores y dirigen la señal de respuesta hacia los actuadores, algunos tipos de controladores son: AND, OR, XOR, XAND.

**Actuadores:** Son los encargados de cambiar el juego de alguna manera, movimiento de objetos, sonido, propiedades, física, etc. Estos cambios pueden ser en otros objetos o pueden causar la activación de otros ladrillos de lógica.

## 1.3 Script.

Es un guión o conjunto de instrucciones. Permiten la automatización de tareas creando pequeñas utilidades. Es muy utilizado para la administración de sistemas UNIX. Son ejecutados por un intérprete de línea de comandos y usualmente son archivos de texto. También Script puede considerarse una alteración o acción a una determinada plataforma, muy parecido a los trucos que se usan para alterar juegos y conseguir resultados extras. Es sumamente utilizado para programar sobre otros programas, es decir, hacer scripts y manipular explícitamente los datos del proyecto activo, o sea, lo que vamos a hacer sobre la herramienta Blender. [Sitio oficial Python 2008]

### 1.3.1 Tipos de Scripts.

Dependiendo de su funcionalidad, los scripts son de estos tipos:

Ejecución manual: sirven para realizar una tarea en el diseño del proyecto. Son los más sencillos, se ejecutan dejando el cursor sobre la ventana de texto y pulsando las teclas ALT-P.

A su vez pueden tener un entorno gráfico, usando instrucciones OpenGL, y funcionando por eventos.

Ejecución automática: se ejecutan cada vez que Blender redibuja el entorno, o cambiamos de frame (fotograma). Así podemos aplicar instrucciones a un objeto, por ejemplo, para darle una localización dependiendo del tiempo.

Ladrillos de Lógica: se emplean en el motor de tiempo real para la toma de decisiones, aplicación de atributos, etc.

### 1.4 Biblioteca Informática.

Es un conjunto de subprogramas utilizados para desarrollar software. Las bibliotecas contienen código y datos, que proporcionan servicios a programas independientes, es decir, pasan a formar parte de éstos. Esto permite que el código y los datos se compartan y puedan modificarse de forma modular. Algunos programas ejecutables pueden ser a la vez programas independientes y bibliotecas, pero la mayoría de éstas no son ejecutables. Ejecutables y bibliotecas hacen referencias (llamadas enlaces) entre sí a través de un proceso conocido como enlace, que por lo general es realizado por un software denominado enlazador.
Las Bibliotecas son además un conjunto de rutinas almacenadas en un archivo. También se utiliza su significado en inglés, Library, en ocasiones traducida erróneamente como 'librería'. Cada conjunto de instrucciones dentro de una biblioteca tiene un nombre, y cada uno realiza una tarea diferente, a menudo muy específica. Por ejemplo, la función printf () es parte de la biblioteca estándar de C, y muestra caracteres en la pantalla. Este tipo de conjuntos de instrucciones simplifica el trabajo y evita duplicar el esfuerzo cada vez que sea necesario realizar una tarea específica. Un programador puede identificar una biblioteca frente a un programa, referirse a rutinas de la biblioteca dentro del programa, y hacer que el programa realice las tareas apropiadas, sin tener que volver a escribir las mismas instrucciones cada vez que sean necesarias. Las bibliotecas pueden incluir rutinas estándar para un lenguaje de programación determinado, extensiones o módulos particulares para facilitar la programación en áreas específicas, como rutinas de uso matemático, de gestión de comunicaciones o de procesamiento de imágenes, o bien contener rutinas personalizadas escritas por el programador.

### 1.4.1 Tipos de Bibliotecas Informáticas.

Biblioteca informática estática.

Una Biblioteca estática es un conjunto de ficheros objeto empaquetados en un único archivo. Consiste en unir durante el enlace el código objeto de las librerías con el código del programa, generando así el ejecutable. El programa ejecutable crece notablemente de tamaño respecto de los archivos objetos, ya que incorpora el código de todas las funciones de las librerías.

Son una colección de ficheros (objetos compilados), los ficheros son agrupados en un solo fichero de extensión **.**lib, **.**a, junto con uno o varios ficheros de cabecera (generalmente **.**h). Con las librerías estáticas que contengan las funciones que se usen más frecuentemente no será necesario escribir el código cada vez. La ventaja de este tipo de enlace es que hace que un programa no dependa de ninguna librería (puesto que las enlazó al compilar), haciendo más fácil su distribución.

Biblioteca informática dinámica.

Los recursos ocupan un fichero independiente del ejecutable, que puede ser utilizado por cualquier aplicación que lo necesite. En algún momento, durante la carga del ejecutable, o posteriormente, en tiempo de ejecución (run-time), el ejecutable deberá integrar este bloque de código en su propio espacio, de forma que pueda acceder a los recursos contenidos en él.

El enlace dinámico produce ejecuciones más lentas, ya que cada vez que se use una función de librería dinámica es necesario buscar el archivo en el que se encuentra y ejecutar su código. Además, pueden producirse errores de enlace durante la ejecución del programa

La ventaja de este tipo de enlace es que el programa es más liviano, reduce el tamaño del archivo ejecutable, permite la compartición de librerías entre diferentes aplicaciones y evita la duplicación de código (por ejemplo, cuando dos programas enlazan con la misma librería, se necesita sólo una copia).

Estos tipos de librerías son las utilizadas por el Sistema Operativo Windows, el mismo contiene una serie de librerías del tipo DLL donde se encapsulan el conjunto de objetos que controlan todo el funcionamiento del sistema.

### 1.4.2 Biblioteca de script.

Es un directorio de scripts o programas listos para usar, ordenados por las tecnologías con las que se han desarrollado y por una clasificación según su contenido, estos contienen código y datos que proporcionan servicios a programas independientes como Python, Java Script y otros

pasando a formar parte de ellos, esto permite que los datos se compartan y puedan modificarse de forma modular. Este tipo de servicios proporcionan a los programadores un conjunto de instrucciones que simplifica el trabajo y evita duplicar el esfuerzo cada vez que sea necesario realizar una tarea específica. [neoteo 2007]

Ejemplos de bibliotecas de script.

- **PyGame** es una librería implementada en la herramienta Python la cual se conoce como una librería no estándar en la que se pueden crear videojuegos 2D y 3D. Está orientado al manejo de sprites (videojuegos). Gracias al lenguaje, se puede prototipar y desarrollar rápidamente. Esto se puede comprobar en las competiciones que se disputan online, donde es cada vez más usado. Los resultados pueden llegar a ser profesionales. También puede utilizarse para crear otros programas multimedia o interfaces gráficas de usuario.
- La herramienta Ogre presenta una biblioteca implementada en Python denominada con el nombre de **PyOgre** que es una librería no estándar en la cual se pueden crear juegos y efectos 3D para el ordenador.
- Otro ejemplo de bibliotecas encontradas la podemos ver en la herramienta **ToolKit** que es muy sencilla e implementada en Python, actualmente se podría decir que es la estándar, se ha utilizado también en TCL, PERL, Ruby (lenguajes de programación).

### 1.5 Herramienta de gestión.

Son todos los sistemas, aplicaciones, controles, soluciones de cálculo, metodología, etc., que ayudan a la gestión de una empresa, organización o grupo de trabajo. Las herramientas de gestión pueden ser de varios tipos ya que son utilizadas para diferentes fines por ejemplo en un proyecto puede existir una herramienta para gestionar los avances que el proyecto pueda tener o archivos que se deseen controlar.

Ejemplos de herramientas de gestión:

1- **Total Commander:** es un administrador de archivos, un programa como el Explorador de Windows para copiar, mover o eliminar archivos. Sin embargo, puede hacer mucho más que Explorer, por ejemplo, empacar y desempacar archivos, servidores de acceso ftp, comparar archivos por contenido, etc.

2- **Dolphin** es un gestor de archivos del proyecto KDE (proyecto de software libre para la creación de un entorno de escritorio e infraestructura de desarrollo para diversos sistemas operativos como GNU/Linux, Mac OS, Windows, etc.).
3- **STDU Explorer:** es un gestor de archivos para la pre visualización y la gestión de PDF, Comic Book Archive (CBR o CBZ), XPS y formatos de archivo de imagen como BMP, GIF, JPEG, PNG, PSD y WMF. Funciona bajo Microsoft Windows, y es libre para uso no comercial. Admite las operaciones estándar, tales como cortar, copiar, pegar, mover, borrar, renombrar, del menú contextual de integración y vistas de árbol de carpetas.

### 1.5.1 Herramienta de gestión de script.

Un gestor de script es una herramienta para gestionar todos los script que se encuentren en una dirección determinada y cumple con una serie de funciones generales para lograr la gestión de estos archivos como son:

- Identificar y localizar un archivo
- Controlar el acceso de varios usuarios a un archivo.
- Bloquear el uso de archivos.
- Eliminar archivos.
- Adicionar archivos.

Ejemplos de herramientas gestoras de scripts:

1- **Amarok** es un reproductor de música que cuenta con la opción Gestor de script-Amarok, esta tiene como características específicas que ejecuta el script hasta que este termine o lo puede detener manualmente, también permite instalar scripts descargándolos directamente de internet.
2- **Distributable Centralized Scripts Manager (DCSM)** gestiona script de servidores en gran cantidad de redes de forma centralizada en entornos de tipo Unix (Sistema Operativo).
3- **Blank Canvas Script Handler** es una extensión del navegador web **Chrome** que permite gestionar los userscript (vínculos a la ejecución) que instalamos en dicho navegador.

## 1.6 Metodologías de desarrollo

En la actualidad se cuenta con una buena cantidad de metodologías ágiles como Extreme Programming, Scrum, Cristal Methods y Feauture Driven Development y con metodologías

tradicionales o robustas como Rational Unified Process (RUP), Microsoft Solutions Framework (MSF) y Métrica.

Una de las más destacadas dentro de las ágiles es la Programación Extrema (Extreme Programming, XP). XP se basa en la realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. Además se define para proyectos con requisitos imprecisos y muy cambiantes, donde existe un alto riesgo técnico. Craig Larman señala como factores negativos de XP la ausencia de énfasis en la arquitectura durante las primeras iteraciones (no hay arquitectos en XP) y la consiguiente falta de métodos de diseño arquitectónico. Dentro de las metodologías fuertes la más destacada es el Proceso Unificado de Modelado (RUP). RUP sugiere su uso para proyectos nuevos o actualizaciones de sistemas existentes, y se recomienda adoptarlo en forma gradual. Es un proceso de desarrollo de software configurable que se adapta a variados proyectos teniendo en cuenta tamaños y complejidades. Por sus significativas características y ventajas expuestas a continuación, es que se decide tomar RUP como metodología de desarrollo para el sistema que se propone.

### 1.6.1 Proceso unificado de desarrollo de software (RUP)

Para la realización del trabajo de diploma se hizo un estudio de la metodología (RUP) *Rational Unified Process* ya que esta proporciona la guía para poder conocer todo el camino a recorrer desde antes de empezar la implementación, con lo cual se asegura la calidad del producto final al captar las mejores prácticas que el estado actual de la tecnología permite.

RUP sirve de 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, tiene un gran número de documentos publicados que se pueden consultar para esclarecer dudas. También porque siguiendo sus pasos propuestos se obtiene una buena documentación de la investigación.

A continuación se muestran las características que más influyeron en la selección de esta metodología:

- Guiado por casos de uso: Los casos de uso reflejan lo que los usuarios futuros necesitan y desean**,** constituyen la guía fundamental establecida para las actividades a realizar durante todo el proceso de desarrollo del sistema.

- Centrado en arquitectura: La arquitectura muestra la visión común del sistema completo. Permite además, implementar el Framework (plataforma sobre la que se implementa el soporte para todas las funcionalidades del sistema) y luego ir desarrollando cada uno de los módulos según se van necesitando.

- Iterativo e Incremental: RUP divide el proyecto en fases de desarrollo, propone además que cada una de ellas se desarrolle en iteraciones, las cuales aportan un incremento en el proceso de desarrollo y terminan con el cumplimiento del punto de control trazado en la fase.

- Utilización de un único lenguaje de modelado: UML.

**1.7 Lenguaje de Modelado.**

Es necesario modelar Software debido a la dimensión y complejidad de los sistemas, a que no son hechos por una sola persona, para un mayor entendimiento del desarrollo y para lograr representaciones simples como medio de manejar la complejidad.

Los modelos ayudan a visualizar como es o como queremos que sea un sistema, además permiten especificar su estructura o comportamiento, proporcionan plantillas que nos guían en la construcción y documentan las decisiones que se adoptan en el transcurso del desarrollo del software. UML (Lenguaje Unificado de Modelación) puede considerarse un lenguaje grafico que estandariza la forma de crear diagramas, el significado preciso de los mismos y las relaciones existentes entre ellos. Este lenguaje facilita la tarea de dibujar diagramas correctos y coherentes. UML se ha convertido en un estándar que tiene las siguientes características:

- Permite modelar sistemas utilizando técnicas orientadas a objetos (OO).
- Permite especificar todas las decisiones de análisis y diseño, construyéndose así modelos precisos, no ambiguos y completos.
- Puede conectarse con lenguajes de programación (Ingeniería directa e inversa).
- Permite documentar todos los artefactos de un proceso de desarrollo (requisitos, arquitectura, pruebas, versiones, etc.).
- Es un lenguaje muy expresivo que cubre todas las vistas necesarias para desarrollar y luego desplegar los sistemas.
- Existe un equilibrio entre expresividad y simplicidad, pues no es difícil de aprender ni de utilizar.

- UML es independiente del proceso, aunque para utilizarlo óptimamente se debería usar en un proceso que fuese dirigido por los casos de uso, centrado en la arquitectura, iterativo e incremental.

## 1.9 Lenguajes de programación.

### 1.9.1 Python.

El lenguaje de Python fue creada por Guido van Rossum a principios de los años 90 cuyo nombre está inspirado en el grupo de cómicos ingleses “Monty Python”. Es un lenguaje similar a Perl, pero con una sintaxis muy limpia y que favorece un código legible. Se trata de un lenguaje interpretado o de script, con tipiado dinámico, multiplataforma y orientado a objetos. Python es considerado como la “oposición leal” a Perl, lenguaje con el cual mantiene una rivalidad amistosa. Los usuarios de Python consideran a este mucho más limpio y elegante para programar. Este permite dividir el programa en módulos reutilizables desde otros programas. Viene con una gran colección de módulos estándar que se pueden utilizar como base de los programas (o como ejemplos para empezar a aprender Python). También hay módulos incluidos que proporcionan E/S de ficheros, llamadas al sistema, sockets y hasta interfaces a GUI (interfaz gráfica con el usuario) como Tk, GTK, Qt entre otros.

Python se utiliza como lenguaje de programación interpretado, lo que ahorra un tiempo considerable en el desarrollo del programa, pues no es necesario compilar ni enlazar. El intérprete se puede utilizar de modo interactivo, lo que facilita experimentar con características del lenguaje, escribir programas desechables o probar funciones durante el desarrollo del programa. [Desarrolloweb 2008]

Características de Python.

Es un lenguaje de programación fácil de aprender y potente. Tiene eficaces estructuras de datos de alto nivel y una solución de programación orientada a objetos simple pero eficaz. [Python 2008]. La elegante sintaxis de Python y su naturaleza interpretada hacen de él el lenguaje ideal para guiones (scripts) y desarrollo rápido de aplicaciones; en muchas áreas y en la mayoría de las plataformas, o sea, está expresamente diseñado para ser usado como una extensión para las aplicaciones que necesiten una interfaz programable, y esto es por lo que Blender lo utiliza. De las dos maneras que tiene de extender Blender, siendo la otra los plugins (complemento) binarios; los script de Python, es la más poderosa, versátil, más sencilla de

comprender y robusta. Proporciona un equilibrio muy bueno entre lo práctico y lo conceptual puesto que es un lenguaje interpretado, o sea, suele ocurrir que se vuelve algo lento en la ejecución de los programas, ya que en lugar de compilar nuestros programas escritos en este lenguaje para obtener ficheros binarios y librerías, los ejecutamos sobre la marcha. Esta característica nos trae ventajas como por ejemplo:

- No hay que hacer pasos intermedios entre que tocamos el código y lo ejecutamos.
- Es más fácil encontrar errores y probar el programa, ya que el programa corre hasta que encuentra uno, en lugar de ser un compilador el que se detiene, sin haber ejecutado ninguna instrucción. (Esto puede ser también una desventaja, porque tales errores podrían ser críticos y un compilador los filtraría, pero el hecho de que sea de alto nivel hace que no sea fácil provocarlos).

Contiene una gran biblioteca de módulos que se pueden usar para hacer toda clase de tareas que abarcan desde programación para Web hasta gráficos.

### 1.9.2 C++

Una de las razones de programar en C++ es además su increíble versatilidad. Con C++ pueden programarse desde los programas más simples, a los programas más complicados como incluso sistemas operativos. En cuanto a la portabilidad en C++, podemos decir que un programa con el código escrito, se podrá compilar en cualquier sistema operativo o sistema informático sin necesidad de cambiar casi el código fuente. Este es por ejemplo uno de los grandes secretos de Linux, al estar el código escrito en este lenguaje (al menos en su concepción original), es más fácil portarlo a diferentes ordenadores como PC´s, Macintosh, incluso superordenadores. Además otras de las grandes ventajas de C++, es que es un lenguaje multi-nivel, es decir, puedes usarlo tanto para programar directamente el hardware (dependiendo del sistema operativo), como para crear aplicaciones tipo Windows, definidas todas por poseer una misma interfaz. Es un hecho de que la demanda de programadores de C/C++ en un país vanguardista es muy elevado porque C/C++ es el lenguaje base para la programación de sistemas operativos, compiladores, intérpretes, servidores, juegos, herramientas de oficina, herramientas de sistema, drivers, etc. Sólo basta recordar que el compilador java, su máquina virtual, los intérpretes de *php*, *perl*, *ruby*, *python*, entre otros, están escritos en C/C++.

Características de C++

- Es un lenguaje multiplataforma, orientado a objetos e imperativo. Usa tipos de datos fuertes y estáticos.

- La eficiencia en tiempo de ejecución de C/C++ está por encima de todos los lenguajes de programación de alto nivel.

- Manejo de memoria por parte del programador, lo que permite un mejor control de la misma y una buena administración de recursos de la computadora.

- Posibilidad de redefinir los operadores (sobrecarga de operadores), y de poder crear nuevos tipos que se comporten como tipos fundamentales.

### 1.10 Consideraciones del capítulo.

En este capítulo se ha mostrado la fundamentación teórica del trabajo de diploma donde se fue bastante amplio en los conceptos básicos y necesarios para el dominio del tema. Se explica que es una herramienta de gestión, se describen sus características y se presentan ejemplos de diferentes herramientas de gestión de archivos existentes. Se explica detalladamente que es una biblioteca informática, los tipos de bibliotecas existentes y principalmente que es una biblioteca de script, donde se presentan ejemplos de bibliotecas de script existentes en diversas herramientas También se aborda sobre los lenguajes a utilizar en el desarrollo del trabajo de diploma y sobre las diferentes bibliotecas.

# Capítulo 2:

# Características del sistema

En este capítulo se realizará un análisis de las características del sistema a desarrollar, haciendo hincapié en la situación problémica que da origen al mismo, también se va a definir la metodología y las herramientas a trabajar en la realización de la aplicación. Se muestran las principales ventajas, la funcionalidad y los requerimientos de entradas de la herramienta de gestión.

## 2.1 Propuesta del sistema.

Se ha decidido darle solución al problema planteado al inicio de esta investigación mediante el desarrollo de un gestor de bibliotecas de script en Python para los desarrollos de Paseos Virtuales sobre Blender, luego de haberlos seleccionado y almacenados en bibliotecas según su clasificación, para facilitar el manejo de estos y reducir el tiempo de trabajo de los diseñadores.


**Figura 5 Propuesta de interfaz del sistema**

.

### 2.1.1 Interfaz de Usuario

- Buscar script por clasificación: el diseñador puede seleccionar el tipo de clasificación con la que quiere trabajar y el sistema le va a facilitar los scripts que se encuentran dentro de la clasificación especificada.

- Adicionar script a la biblioteca: el diseñador puede adicionar un nuevo script a la biblioteca de script.

- Edición de script: el diseñador podrá modificar el script que seleccione.

- Eliminar script: el diseñador puede eliminar el script de la biblioteca si considera que no es necesario en el diseño.

- Adicionar Clasificación: el diseñador puede adicionar una nueva clasificación a la biblioteca.

-Listar script: el diseñador podrá ver todos los script que se encuentran almacenados en la biblioteca.

- Eliminar Clasificación: el diseñador podrá eliminar una clasificación si cree que no es necesaria.

### 2.2 Ventajas de la aplicación

Según el estudio realizado anteriormente, existen varias herramientas para gestionar diferentes tipos de scripts, pero no existe una herramienta capaz de gestionar script de Python sobre Blender. El hecho que la aplicación se ejecute sobre Blender posibilita al diseñador ejecutar el script de la biblioteca si desea directamente al buscarlo mediante el gestor. Con el desarrollo del gestor de bibliotecas de scripts se reduce el tiempo de trabajo con los ficheros .blend y .py que existían acumulados. Esta herramienta presenta entre sus principales ventajas:

- ✓ Ocupa poco espacio en disco.
- ✓ Se conoce el código fuente.
- ✓ Permite integrarse a la herramienta Blender.
- ✓ Uso propio del grupo de desarrollo de Paseos Virtuales.
- ✓ Reduce el tiempo de ejecución de un proyecto de diseño.
- ✓ Permite tener organizados y clasificados los scripts del grupo de desarrollo.

### 2.3 Lenguaje de programación a utilizar.

Se estudió la propuesta por solicitud del cliente y se escogió para la elaboración de la aplicación el lenguaje de programación de Python ya que este es un lenguaje de scripting independiente de plataforma y multiparadigma. Es un lenguaje interpretado, lo que significa que no se necesita compilar el código fuente para poder ejecutarlo, lo que ofrece ventajas como la rapidez de desarrollo.

Contiene una gran cantidad de librerías, tipos de datos y funciones incorporadas en el propio lenguaje, que ayudan a realizar muchas tareas habituales sin necesidad de tener que programarlas desde cero. Una ventaja de este lenguaje es la sencillez y velocidad con la que se crean los programas; un programa en Python puede tener de 3 a 5 líneas de código menos que su equivalente en Java o C. A demás permite integrarse fácilmente con la herramienta Blender ya que este presenta un intérprete de Python que permite a cualquier usuario añadirle

funcionalidades a Blender implementando un simple script. Además el grupo de desarrollo de Paseos Virtuales utiliza este lenguaje para el desarrollo de las tareas.

### 2.4 Proceso Unificado de Desarrollo de Software (RUP).

Para la realización de la herramienta se escogió la metodología de desarrollo de software (RUP) ya que esta es uno de los procesos más generales de los existentes actualmente, está pensado para adaptarse a cualquier proyecto y contiene un conjunto de actividades necesarias para transformar los requisitos de un usuario en un software. RUP se caracteriza por estar dirigido por casos de uso, centrado en la arquitectura y por ser iterativo e incremental. Estas cualidades son válidas para el desarrollo del sistema que se propone en esta investigación. Además incluye artefactos (que son los productos tangibles del proceso como por ejemplo, el modelo de casos de uso, el código fuente, etc.) y roles (papel que desempeña una persona en un determinado momento a lo largo del proceso). Aunque RUP es un proceso de desarrollo de software genérico, se concibió en gran medida para el desarrollo de sistemas basados en programación orientada a objetos.

### 2.5 Herramientas de trabajo.

#### 2.5.1 Blender 2.49

Se utilizó la herramienta Blender 2.49 ya que esta es multiplataforma, libre, gratuita y con un tamaño de origen realmente pequeño comparado con otros paquetes de 3d, dependiendo del Sistema Operativo en el que se ejecute. Tiene posibilidades de renderizado interno-versátil e integración externa con potentes trazadores de rayos, presenta una característica muy peculiar que es un intérprete de Python, garantizando que se pueden programar scripts dentro de la herramienta Blender además es la utilizada por el grupo de desarrollo de Paseos Virtuales.

#### 2.5.2 Visual Paradigm.

Se utilizó Visual Paradigm para UML porque es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una rápida construcción de aplicaciones de buena calidad y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación.

Presenta grandes características como soporte de UML versión 2.1, diagramas de Procesos de Negocio-Proceso, decisión, actor de negocio, documento, modelado colaborativo con CVS y subversión, permite la ingeniería inversa, código a modelo y código a diagrama.

### 2.6 Requerimientos de entrada.

La herramienta destinada para la solución propuesta del trabajo de diploma se nombra *Gestor de Scripts* y para su utilización se deben cumplir los siguientes requerimientos de entrada de datos.

Cada script a adicionar.

1- Debe existir.

2- Debe estar salvado en el formato: .py.

Cada script a eliminar.

1- Debe existir en la biblioteca.

Cada script a buscar

1- Debe existir en la biblioteca

Cada script a editar

1- Debe existir en la biblioteca

### 2.7 Funcionalidades

El *Gestor de Scripts* permite a los usuarios gestionar scripts para minimizar el tiempo de diseño y animación, además permite tener una gran cantidad de script almacenados en una biblioteca organizados por clasificación(ejemplo: Movimiento, Generar, Posición) para proporcionar una mayor eficiencia a la hora de buscar un script determinado.

### 2.8 Consideraciones del capítulo

Este capítulo permite tener una idea clara de la solución que se propone lograr en el desarrollo del Gestor de Script, además de mostrar las diferentes funcionalidades de la herramienta donde se explicó de forma general sus características y seleccionar la metodología y las herramientas con las que se trabajaran en la elaboración de la aplicación.

# Capítulo 3:

# Análisis y Diseño del Sistema

En el presente capítulo se realizará una detallada descripción del funcionamiento del sistema en el cual se definen los Requerimientos Funcionales y No Funcionales así como los actores que interactúan con el mismo, se presentan los distintos Casos de Usos que contiene el sistema con sus correspondientes descripciones y por último se define el prototipo de interfaz de la herramienta.

### 3.1 Requisitos funcionales del sistema.

Una de las primeras etapas de cualquier proceso de desarrollo comienza con la tarea de identificación de los requerimientos tantos funcionales como no funcionales para lograr un buen diseño que conlleve a cumplir todas las peticiones del cliente.

**RF1**- Instalar paquete

**RF2-** Inicializar Blender.

**RF2.1-** Inicializar la aplicación.

**RF3-** Listar Script de la biblioteca por clasificación.

**RF4**- Buscar script.

**RF4**.1- Mostrar script.

**RF5**- Adicionar script a la biblioteca.

**RF5**.1- Cargar script.

**RF5**.2- Agregar script a la clasificación seleccionada.

**RF**5.3- Mostrar listado.

**RF6**- Editar script.

**RF6**.1- Mostar la lista de script para seleccionar.

**RF6**.2- Abrir ventana para edición de script.

**RF6**.3- Guardar el script editado en la biblioteca.

**RF7**- Eliminar script.

**RF7**.1- Mostar la lista de script para seleccionar.

**RF7**.2- Eliminar script de la biblioteca.

**RF7**.3- Eliminar de la lista de script.

**RF8**- Adicionar clasificación.

**RF8.1**-Mostrar ventana de texto para nombrar clasificación.

**RF9**- Eliminar Clasificación.

### 3.2 Requisitos no funcionales del sistema

Requerimientos de apariencia o interfaz externa.

La aplicación propuesta será usada por personas que tengan conocimientos básicos de informática y más sobre el trabajo con la herramienta Blender. La interfaz del sistema debe propiciar el máximo aprovechamiento de la herramienta.

Requerimientos de usabilidad.

A los administradores del sistema se les dará un adiestramiento básico en la utilización de la herramienta. Estas personas tendrán un nivel de acceso amplio para poder darle cumplimiento o respuesta a cualquier edición o diseño puesto en práctica.

Requerimiento de portabilidad.

La biblioteca será utilizada en cualquier sistema operativo Windows donde pueda ser instalada las herramientas Blender y Python.

Requerimientos de seguridad.

**Confiabilidad** - La información manejada por la herramienta debe ser protegida de acceso no autorizado.

**Integridad** - La información manejada por la herramienta debe ser objeto de cuidadosa protección contra la corrupción y estados de inconsciencia.

**Disponibilidad** - La herramienta debe estar disponible en todo momento para aquellas personas con acceso a la información de los scripts y los mecanismos de seguridad no deben ser obstáculos a los usuarios para el manejo de la biblioteca de script.

Requerimientos de software

En las computadoras de los usuarios solo se requiere las herramientas Blender y Python instaladas y listas para su utilización.

Requerimientos de hardware

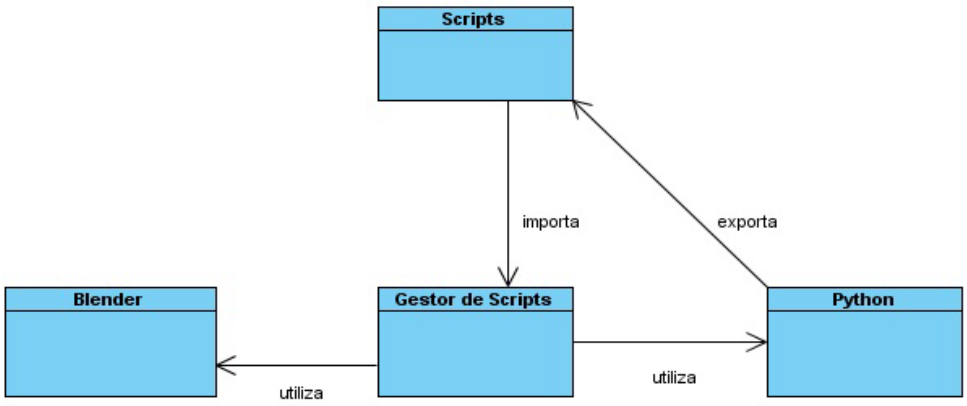
Se requiere una computadora con al menos 512 MB de memoria RAM, un procesador Pentium 4 a una velocidad de 2.0 GHz, 40 GB de capacidad en disco duro y tiene que tener los drivers de la motherboard, si la computadora presenta una tarjeta gráfica también es necesario los drivers de la tarjeta para que la herramienta pueda correr sin problemas.

Restricción en el diseño y la implementación

Se debe realizar una herramienta que permita dar respuesta en el menor tiempo posible garantizando la calidad del sistema. Para garantizar el desarrollo de la herramienta se utilizara la metodología RUP. Se utilizará para realizar los modelos UML la herramienta de apoyo Visual Paradigm 6.4.

### 3.3 Modelo Conceptual

Para alcanzar una mejor comprensión del sistema a desarrollar se introduce la figura del modelo conceptual (Fig. 4) donde se presentan todos los conceptos y relaciones que están presentes en el dominio. De los principales conceptos mostrados en el modelo se encuentra el de **Gestor de Scripts,** que representa una interfaz externa con la cual interactúa el usuario para ejecutar las diferentes operaciones del sistema que se realizan sobre los **Scripts** de la biblioteca. Los conceptos de **Python** y **Blender** están relacionados con la biblioteca de script ya que mediantes estos se pueden cargar los diferente ficheros que se manipulan en la herramienta.





Autores: Fernando Javier Mesa Quiles
Roger Terrero Hernández

**Figura 6 Modelo Conceptual**

.

### 3.4 Definición de los Casos de Uso.

Para definir las funcionalidades que debe de cumplir la herramienta se elaboraron 4 casos de usos. Buscar script por clasificación, Adicionar script a la lista de script clasificados, Eliminar script, Editar script, Adicionar clasificación a la biblioteca, Eliminar clasificación de la biblioteca y Listar los script de la biblioteca. Cada uno de ellos ayuda a tener una mejor comprensión del sistema a cualquier interesado a desarrollar una herramienta, independiente del tipo de software que decida emplear. En la descripción textual de cada uno de los Casos de Usos se representa además, las interacciones entre el sistema y el actor, lo que elimina cualquier duda posible en el funcionamiento de la herramienta.

### 3.5 Definición de actores

| **Actores** | **Justificación** |
|---|---|
| Diseñador | Es la persona que interactúa con la herramienta y al final del proceso obtiene beneficio de ella. |

**Tabla 1 Definición de actores**

| **CU_1** | Buscar script. |
|---|---|
| Actor | Diseñador |
| Descripción | Buscar todos los script que presenta la biblioteca por su clasificación |
| Referencias | Requisito funcional 4 |

**Tabla 2 Definición de actores del Caso de Uso Buscar Script**

| **CU_2** | Adicionar script. |
|---|---|
| Actor | Diseñador |

| | |
|---|---|
| Descripción | Adicionar un nuevo script a la lista de script clasificados. |
| Referencias | Requisito funcional 5 |

**Tabla 3 Definición de actores del Caso de Uso Adicionar Script**

| | |
|---|---|
| **CU_3** | Editar script. |
| Actor | Diseñador |
| Descripción | Editar un script de la biblioteca |
| Referencias | Requisito funcional 6 |

**Tabla 4 Definición de actores del Caso de Uso Editar Script**

| | |
|---|---|
| **CU_4** | Eliminar script |
| Actor | Diseñador |
| Descripción | Eliminar un script que no sea necesario para la biblioteca. |
| Referencias | Requisito funcional 7 |

**Tabla 5 Definición de actores del Caso de Uso Eliminar Script**

| | |
|---|---|
| **CU_5** | Listar script de la biblioteca. |
| Actor | Diseñador |
| Descripción | Listar los script que se encuentran en la biblioteca. |

| | |
|---|---|
| Referencias | Requisito funcional 3 |

**Tabla 6 Definición de actores del Caso de Uso Lista de Script de la biblioteca**

| | |
|---|---|
| **CU_6** | Adicionar clasificación. |
| Actor | Diseñador |
| Descripción | Adicionar una nueva clasificación a la biblioteca de script. |
| Referencias | Requisito funcional 8 |

**Tabla 7 Definición de actores del Caso de Uso Adicionar Clasificación**

| | |
|---|---|
| **CU_7** | Eliminar clasificación. |
| Actor | Diseñador |
| Descripción | Elimina una clasificación de la biblioteca |
| Referencias | Requisito funcional 9 |

**Tabla 8 Definición de actores del Caso de Uso Eliminar Clasificación**

### 3.6 Diagrama de Casos de Usos

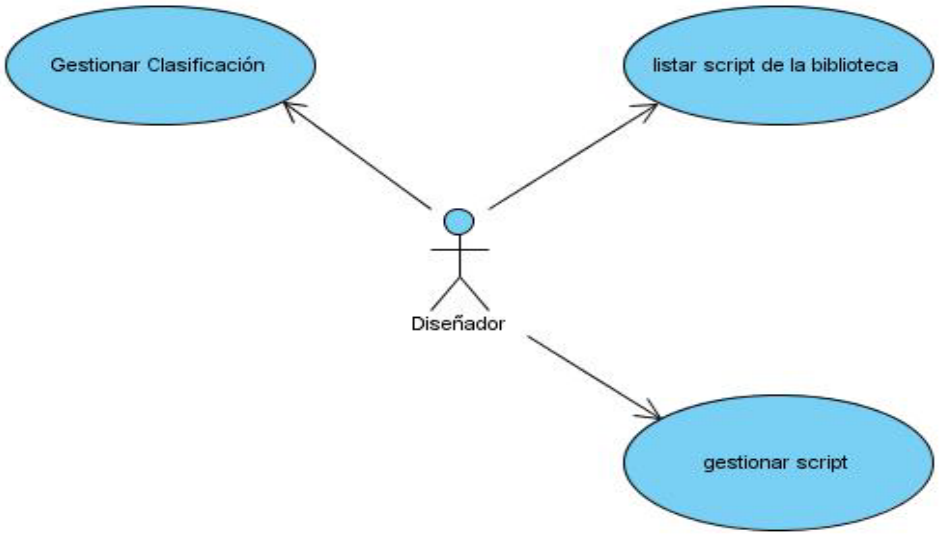



**Figura 7 Diagrama de Caso de Uso del Sistema**

.

En la figura 5 se muestra el diagrama de Casos de Usos del sistema, donde representa de forma general las principales funcionalidades que debe cumplir la herramienta que serían: listar los script de la biblioteca, gestionar clasificación (eliminar clasificación, adicionar clasificación) y gestionar script (eliminar script, buscar script, adicionar script, modificar o editar script).

### 3.7 Definición textual de los Casos de Usos.

| | |
|---|---|
| Caso de uso | Listar Script |
| Autores | Diseñador |
| Resumen | Este Caso de Uso empieza cuando el diseñador presiona el botón de Lista Script. |
| Precondiciones | Que el diseñador haya ejecutado la aplicación. |
| Referencias | Requisito Funcional 3 |
| Prioridad | normal |
| Acción del autor | Respuesta del sistema. |

| | |
|---|---|
| | 1- El sistema muestra la opción de listar todos los script de la biblioteca. |
| 2- El diseñador da clic sobre el botón Lista Script. | 3- El sistema muestra el listado de los script de la biblioteca. |
| Flujos | Alternos |
| Acción del actor. | Respuesta del sistema. |
| | |

**Tabla 9 Descripción del Caso de Uso Listar script de la biblioteca**

| | |
|---|---|
| Caso de Uso | Gestionar Clasificación |
| Autores | Diseñador |
| Resumen | El caso de uso tiene lugar cuando el diseñador gestiona la clasificación con que quiere trabajar. |
| Precondiciones | Que el diseñador haya ejecutado la herramienta. |
| Prioridad | normal |
| Acción del actor | Respuesta del sistema |
| | 1- La herramienta muestra las opciones de eliminar y adicionar clasificación. |
| 2- El diseñador escoge una de las opciones.<br>a- Eliminar clasificación.<br>b- Adicionar clasificación. | |
| Adicionar | Clasificación |
| Acción del actor | Respuesta del sistema |
| | 1- El sistema de la opción de adicionar una nueva clasificación. |
| 2- El diseñador da clic sobre el botón Clasificación de la ventana Adicionar. | |
| | 3- El sistema muestra una ventana de texto para escribir el nombre |

| | |
|---|---|
| | de la nueva clasificación. |
| 4- El diseñador escribe el nombre de la clasificación y presiona **ok**. | |
| | 5- El sistema adiciona la nueva clasificación y muestra el mensaje "La clasificación ha sido adicionada con éxito" |
| Flujos | Alternos |
| Acción del actor | Respuesta del sistema |
| | 3.1- El sistema muestra un mensaje "la cosificación ya existe" y vuelve al flujo normal de eventos. |
| Eliminar | Clasificación |
| Acción del actor | Respuesta del sistema. |
| | 1- El sistema muestra la opción de eliminar una clasificación. |
| 2- El diseñador da clic sobre el botón Clasificación de la ventana Eliminar | |
| | 3- El sistema muestra un listado con las clasificaciones existentes en la biblioteca. |
| 4- El diseñador da clic sobre la clasificación que desea eliminar. | |
| | 5- El sistema elimina la clasificación y muestra un mensaje "La clasificación se ha eliminado". |
| Flujos | Alternos |
| Acción del actor | Respuesta del sistema |
| | 5.1- El sistema muestra el mensaje "Debe seleccionar una clasificación" y vuelve al flujo |

| | |
|---|---|
| | normal de eventos. |

**Tabla 10 Descripción del Caso de Uso Gestionar Clasificación**

| | |
|---|---|
| Caso de uso | Gestionar script |
| Autores | Diseñador |
| Resumen | Este caso de uso tiene lugar cuando el diseñador gestiona el script con que quiere trabajar. |
| Precondiciones | Que el diseñador haya ejecutado la herramienta. |
| Prioridad | normal |
| Acción del actor. | Respuesta del sistema. |
| | 1-La biblioteca muestra las opciones de adicionar script, editar script, eliminar script y buscar script. |
| 1- El diseñador escoge una de las opciones con la que desee trabajar.<br>a- Adicionar script pulsando el botón de Script de la ventana Adicionar.<br>b- Editar script, pulsando el botón Editar Script.<br>c- Eliminar script pulsando el botón Script de la ventana Eliminar.<br>d- Buscar script pulsando el botón Clasificación.<br>e- Buscar script por nombre dando clic sobre el botón Por Nombre. | |
| Adicionar script | |
| Acción del actor | Respuesta del sistema |

| | |
|---|---|
| | 1-El sistema da la opción de adicionar un nuevo script a la biblioteca. |
| 2-El diseñador carga el script que desea adicionar a la biblioteca | |
| | 3-El sistema muestra las clasificaciones que existen para que el usuario elija en cuál de las clasificaciones desea guardar el script. |
| 4-El usuario elige en que clasificación desea guardar el script cargado. | |
| | 5-El sistema adiciona el script en la clasificación elegida por el usuario. |
| | |
| Flujos alternos | |
| Acción del actor | Respuesta del sistema |
| | 3.1- El sistema muestra el mensaje “Debe seleccionar una clasificación” y pasa a la acción 1 del flujo normal de eventos.<br>4.1- El sistema muestra el mensaje “El script debe estar en formato .py” y pasa a la acción 1 del flujo normal de eventos. |
| Editar script | |
| Acción del actor | Respuesta del sistema |
| | 1- El sistema muestra en la interfaz un botón para editar un script. |
| 2- El diseñador da clic sobre ese botón. | |
| | 3- El sistema muestra las clasificaciones existentes. |
| 4- El diseñador selecciona una clasificación. | |


Autores: Fernando Javier Mesa Quiles
Roger Terrero Hernández

| | |
|---|---|
| | 5- El sistema muestra el listado de script de esa clasificación. |
| 6- El diseñador selecciona el script que desea editar. | |
| | 7- El sistema muestra el script en una ventana del Editor de Texto de Blender. |
| | 8- El sistema da la opción de salvar el script. |
| 9- El diseñador da clic sobre el botón Salvar. | |
| | 10- El sistema elimina el script que estaba anteriormente y adiciona el nuevo script que el diseñador editó y muestra el mensaje “El script se salvó con éxito”. |
| Flujos alternos | |
| Acción del actor | Respuesta del sistema |
| | 10.1- El sistema muestra un mensaje “Seleccione un script para editar” y pasa al flujo normal de eventos. |
| Eliminar script | |
| Acción del actor | Respuesta del sistema |
| | 1- El sistema muestra la opción de eliminar un script. |
| 2- El diseñador da clic sobre el botón Script de la ventana Eliminar. | |
| | 3- El sistema muestra el listado de las clasificaciones existentes en la biblioteca. |
| 4- El diseñador selecciona una clasificación en la cual está el script que desea eliminar. | |
| | 5- El sistema muestra el listado de los script de esa clasificación. |

| | |
|---|---|
| 6- El diseñador selecciona el script a eliminar y lo elimina. | |
| | 7- El sistema elimina el script de la biblioteca y muestra el mensaje ¨El script se ha eliminado de la biblioteca¨. |
| Flujos alternos | |
| Acción del actor | Respuesta del sistema |
| | 3.1- El sistema muestra el mensaje “Debe seleccionar un script” y pasa al flujo normal de eventos. |
| Buscar script por clasificación | |
| Acción del actor | Respuesta del sistema |
| | 1- El sistema da la opción de buscar un script por clasificación. |
| 2- El diseñador selecciona el botón clasificaciones de la ventana Buscar Script. | |
| | 3- El sistema muestra las clasificaciones existentes en la biblioteca. |
| 4- El diseñador selecciona la clasificación. | |
| | 5- El sistema muestra la lista de script de la clasificación seleccionada. |
| 6- El diseñador selecciona el script que desea buscar. | |
| | 7- El sistema muestra el script en el Editor de Texto de Blender. |
| Flujos alternos | |
| | 7.1- El sistema muestra un mensaje “El script no se encuentra en la biblioteca” y pasa al flujo normal de los eventos. |
| Buscar script | Por nombre |


Autores: Fernando Javier Mesa Quiles
Roger Terrero Hernández

| Acción del actor | Respuesta del sistema. |
|---|---|
| | 1- El sistema da la opción de buscar un script por el nombre. |
| 2- El diseñador selecciona el botón Por Nombre de la ventana Buscar script. | |
| | 3- El sistema muestra una ventana para introducir un texto (letra o nombre de un script). |
| 4- El diseñador introduce el nombre o la letra de un script. | |
| | 5- El sistema muestra el script en el Editor de Texto si el diseñador introdujo en nombre, sino muestra una lista con los script que empiezan con la inicial introducida. |
| 4.1- El diseñador escoge el script que desee de la lista mostrada. | |
| | 5.1- El sistema muestra el script seleccionado en el Editor de Texto. |
| Flujos alternos | |
| Acción del actor | Respuesta del sistema |
| | 5.2 – el sistema muestra el mensaje ¨El script no se encuentra en la biblioteca¨ y pasa al paso 1 del flujo normal de eventos. |

**Tabla 11 Descripción del Caso de Gestionar Script**

### 3.8 Arquitectura del sistema.

A continuación en la siguiente figura se presenta la aplicación que constituye la estructura base del programa siendo la plataforma sobre la que se implementa el soporte para todas las funcionalidades del sistema. Esta representación se hará a través de un diagrama de clases de análisis que muestra un conjunto de las principales clases que conforman el sistema. La estructura de la aplicación que se verá a continuación nos permitirá además tener una clara

visión y comprensión de cómo funcionará el sistema luego de la fase de implementación y evitara de esta forma que los usuarios que van a interactuar con el sistema no cometan errores en el momento de efectuar una operación haciendo uso del sistema.

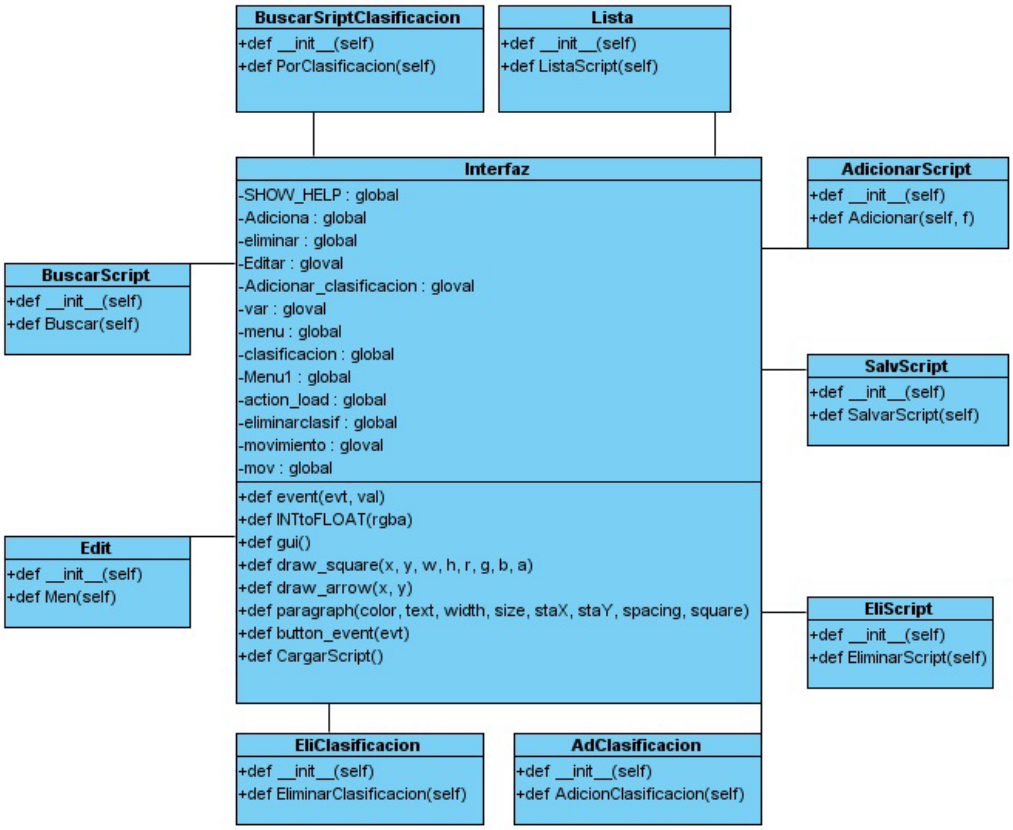



**Figura 8 Diagrama de clases de la aplicación**

.

En la figura 6 se muestra el diagrama de clases del sistema que va a estar conformado por una clase principal la cual tiene el nombre de Interfaz a la cual se asocian las clases que van a conformar las principales funcionalidades de la aplicación.

### 3.9 Diagramas de clases del diseño por paquetes.

#### 3.9.1 Caso de Uso Buscar Script

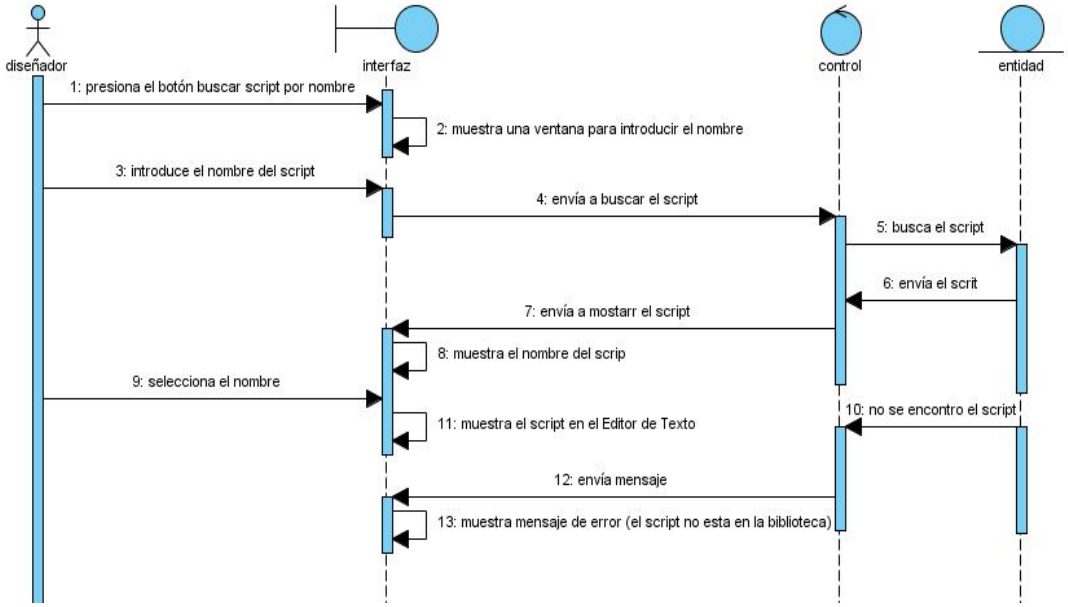



**Figura 9 Diagrama de secuencia para el Caso de Uso Buscar Script por nombre**

Como se muestra en el diagrama de secuencia de la figura 7 el Caso de Uso Buscar Script comienza cuando el diseñador presiona el botón **Por Nombre** de la ventana buscar de la interfaz, luego se le muestra una ventana de texto donde puede poner el nombre de un script dar al botón **ok**, la herramienta pasa a buscar el nombre en la clase controladora y luego esta busca en la clase entidad de la aplicación, en caso de que el nombre no aparezca se le muestra al diseñador un mensaje que le informa que el script no se encuentra en la biblioteca y en caso contrario se muestra el nombre del script, luego el diseñador debe dar clic sobre el nombre del script para poder ver el cuerpo de texto que conforma el mismo.

### 3.9.2 Caso de Uso Buscar Script por clasificación

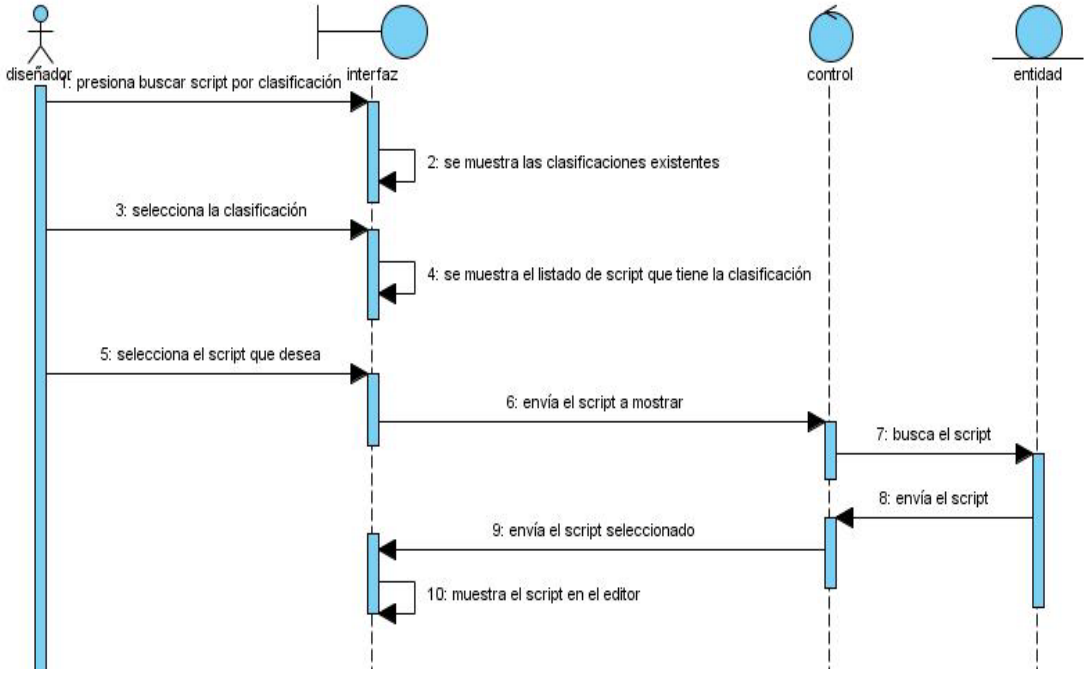



**Figura 10 Diagrama de secuencia para el Caso de Uso Buscar Script por Clasificación**

Como se muestra en el diagrama de secuencia de la figura 8 el Caso de Uso Buscar Script por clasificación comienza cuando el diseñador presiona el botón **Por Clasificación** de la ventana **Buscar** de la interfaz, luego se muestra en la interfaz el listado de las clasificaciones existentes en la biblioteca, el diseñador selecciona la clasificación por donde quiere empezar su búsqueda, al seleccionarla se muestra el listado de script de esa clasificación y al dar clic sobre algún script este se muestra en el Editor de Texto de Blender.

### 3.9.3 Caso de Uso Listar Script

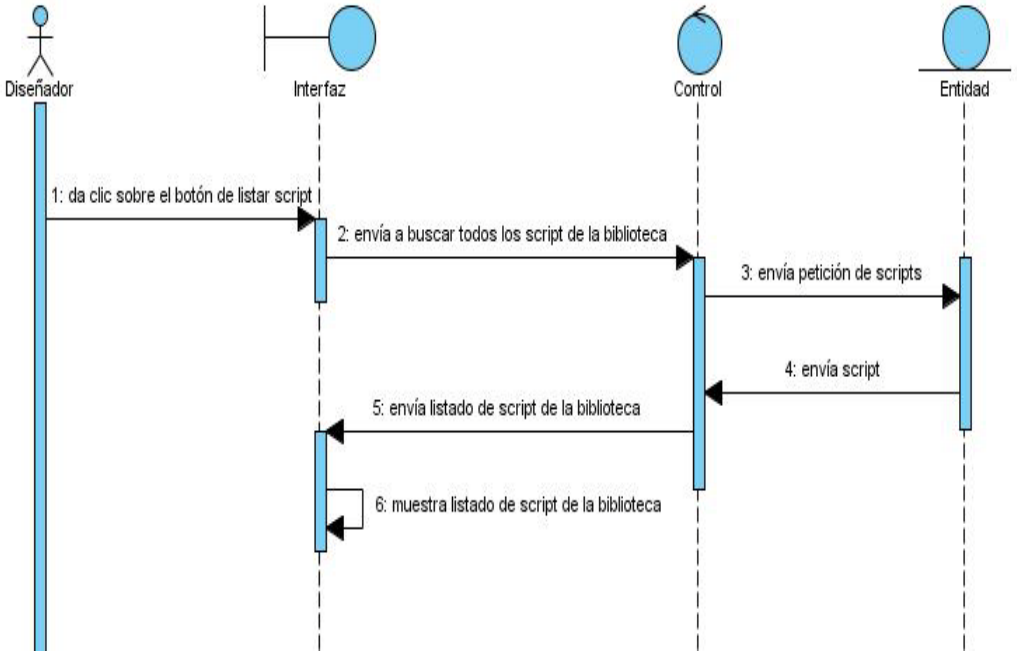



**Figura 11 Diagrama de secuencia para el Caso de Uso Listar Script**

.

Como se muestra en el diagrama de secuencia de la figura 9 el Caso de Uso Listar Script comienza cuando el diseñador da clic sobre el botón **Listar** S**cript**, luego la aplicación muestra una lista de los nombres de los scripts de todas las clasificaciones.

### 3.9.4 Caso de Uso Adicionar Script

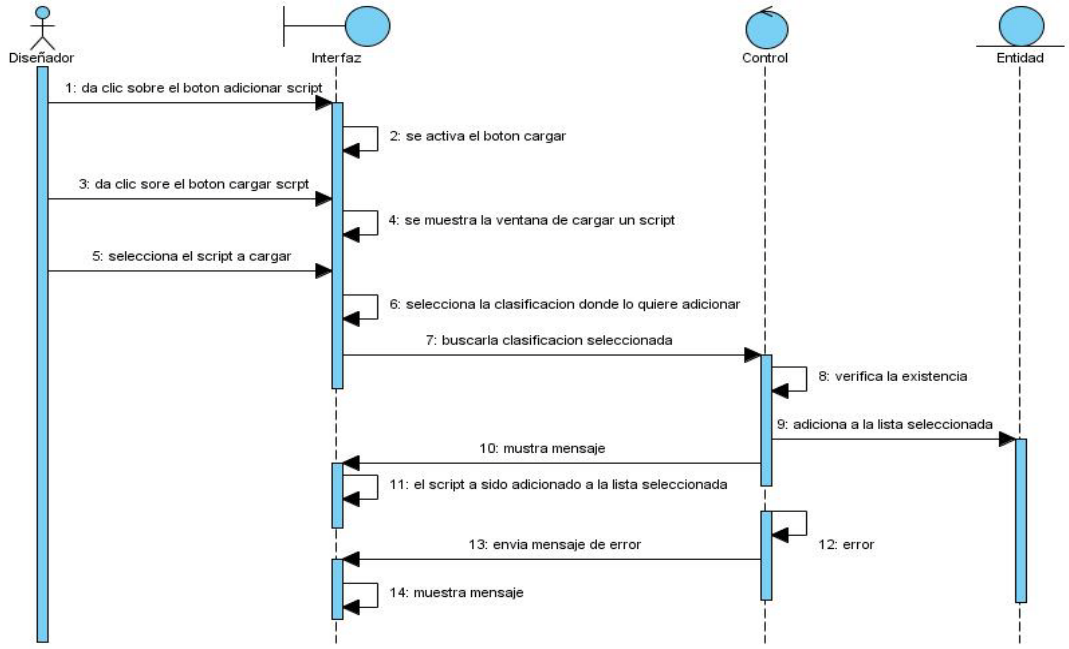



**Figura 12 Diagrama de secuencia para el Caso de Uso Adicionar Script**

.

Como se muestra en el diagrama de secuencia de la figura 10 el Caso de Uso **Adicionar Script** comienza cuando el diseñador da clic sobre el botón **Script** de la ventana **Adicionar**, luego pasa a una ventana que le da la opción al diseñador de buscar el nuevo script que desee adicionar a la biblioteca, una vez que tenga seleccionado el script presiona el botón **Clasificaciones** y se despliega una lista de nombres de las diferentes clasificaciones con que cuenta la biblioteca, el diseñador debe dar clic sobre la clasificación donde desea adicionar el script.

### 3.9.4 Caso de Uso Eliminar Script

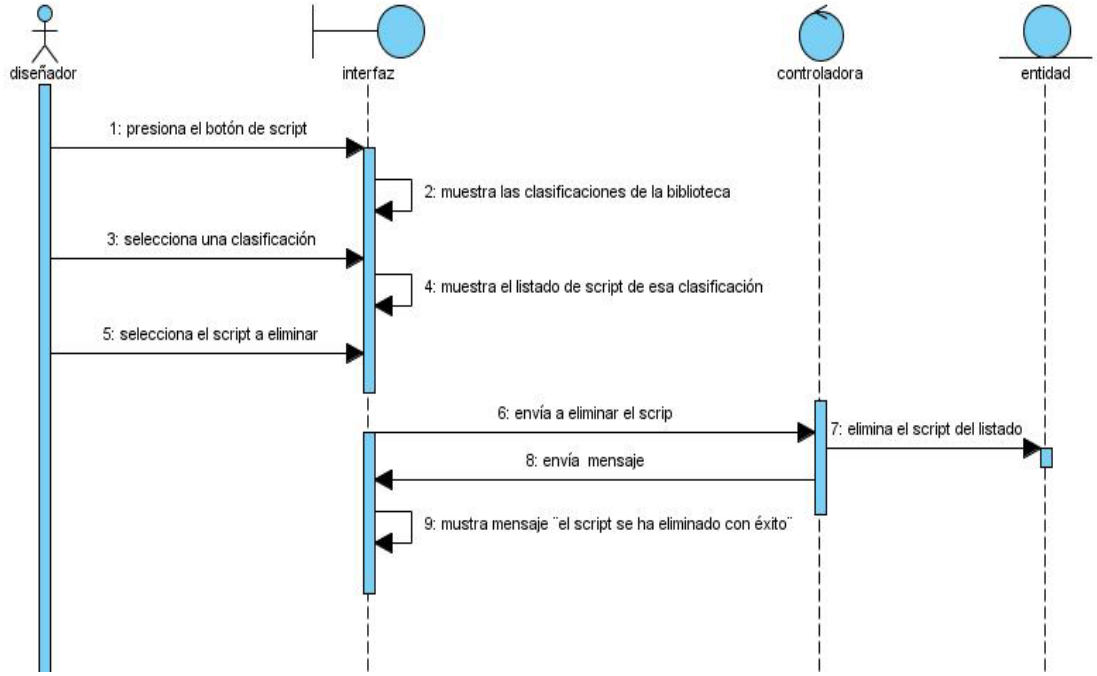



**Figura 13 Diagrama de secuencia para el Caso de Uso Adicionar Script**

.

Como se muestra en la figura 11 el diagrama de secuencia para el Caso de Uso Eliminar Script de la Biblioteca comienza cuando el diseñador da clic sobre el botón **Script** de la ventana **Eliminar**, luego se muestran las clasificaciones existentes en la biblioteca y el diseñador selecciona la clasificación en la que se encuentra el script que busca, debe dar clic sobre él y este se elimina de la biblioteca.

### 3.9.6 Caso de Uso Eliminar Clasificación

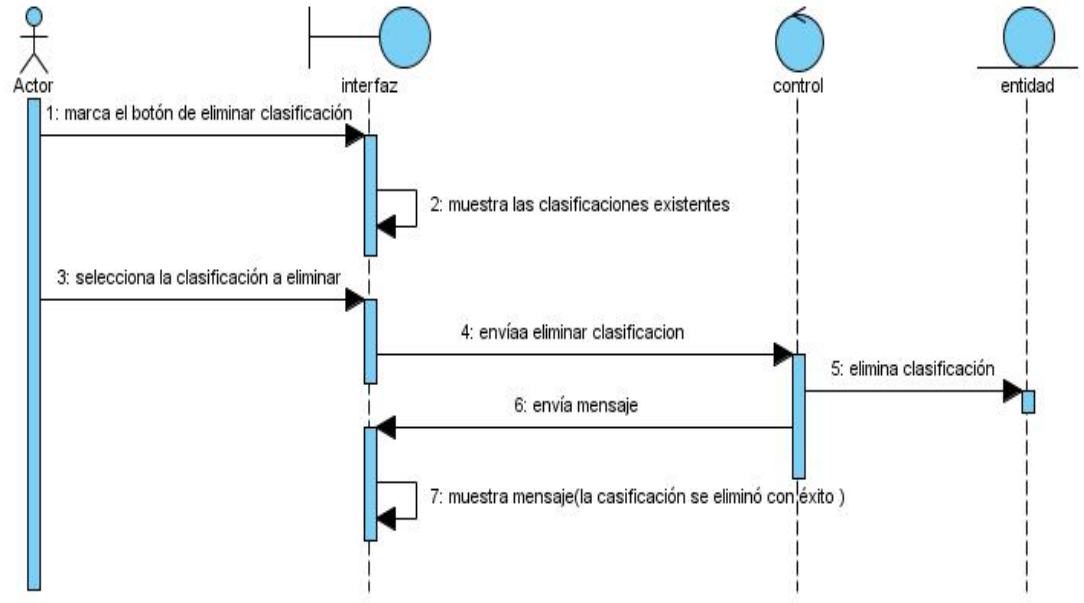



**Figura 14 Diagrama de secuencia para el Caso de Uso Eliminar Clasificación.**

Como se muestra en la figura 12 el diagrama de secuencia del Caso de Uso Eliminar Clasificación comienza cuando el diseñador da clic sobre el botón **Clasificación** de la ventana **Eliminar,** luego se muestra las clasificaciones existentes en la biblioteca y el diseñador da clic sobre la clasificación que desea eliminar.

### 3.9.7 Caso de Uso Editar Script

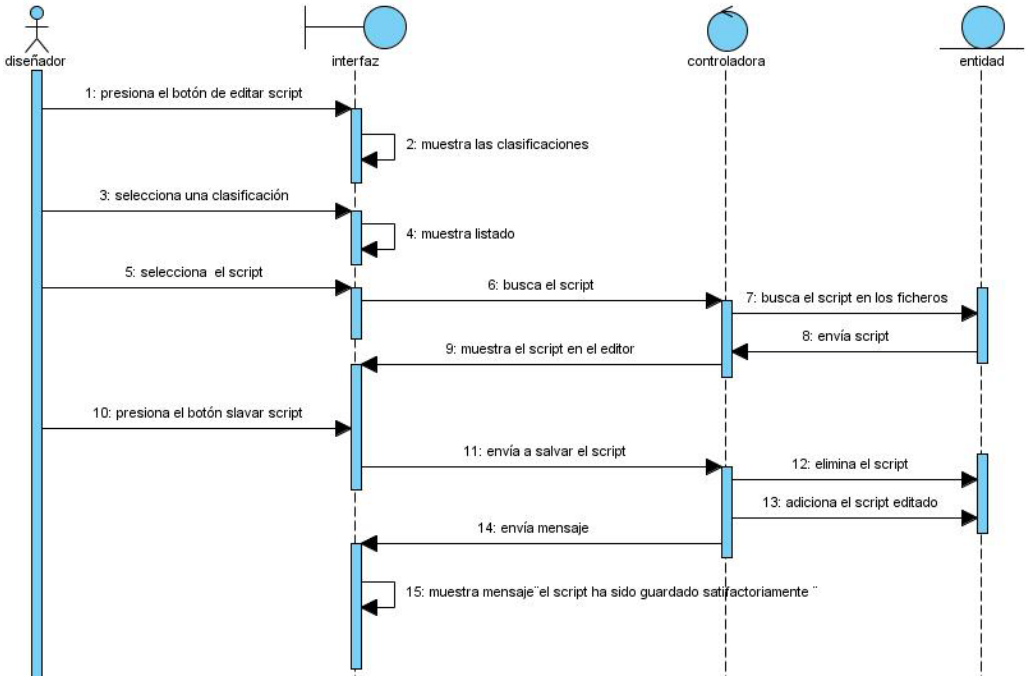



**Figura 15 Diagrama de secuencia para el Caso de Uso Eliminar Clasificación**

.

Como se muestra en la figura 13 el diagrama de secuencia para el Caso de Uso Editar Script comienza cuando el diseñador da clic sobre el botón **Editar Script**, luego se muestra una lista de las clasificaciones existentes en la biblioteca, el diseñador al seleccionar una clasificación se muestra el listado de script de la misma, al dar clic sobre el script que desea editar este se muestra en el Editor de Texto de Blender para poder modificarlo. Al terminar con el script el diseñador tiene la opción de guardar los cambios presionando el botón **Salvar**, este botón tiene la funcionalidad de eliminar el script seleccionado anteriormente y automáticamente adiciona al final del fichero el script modificado.

### 3.10 Prototipo de interfaz de usuario

La interacción usuario-computadora conforma el punto básico para el buen funcionamiento de las aplicaciones. La interfaz de usuario es el vínculo entre el usuario y el programa de computadora y es una de las partes más importantes de cualquier programa. Un programa muy poderoso con una interfaz pobremente elaborada tiene poco valor para un usuario no experto.

La elaboración de una interfaz de usuario, bien diseñada, exige una gran dedicación pues generalmente las interfaces son grandes, complejas y difíciles de implementar, depurar y modificar. Sin embargo un punto crucial para el éxito de esta interacción es la sencillez del diseño de la interfaz, de forma tal que el usuario que vaya a interactuar con la aplicación pueda hacerlo de una manera fácil y comprensible para alcanzar su objetivo.

### 3.10.1 Interfaz de la herramienta de gestión

Esta es la interfaz general de la herramienta, es decir, esta es la primera forma que el usuario distingue a la hora de ejecutar la aplicación, desde aquí se derivan los requerimientos planteados por el usuario para darle cumplimiento a la herramienta. Como se puede apreciar en la imagen la interfaz presenta diferentes ventanas y dentro de estos botones como Script, Clasificación, Por Nombre, Por Clasificación, Editar Script y Lista de Script; cada uno de ellos va a tener una funcionalidad de acuerdo a los requisitos funcionales que debe cumplir el sistema.


**Figura 16 Interfaz de la herramienta de gestión**

.


Autores: Fernando Javier Mesa Quiles
Roger Terrero Hernández

### 3.10.2 Interfaz para Buscar Script por Nombre.

Esta interfaz permite al diseñador ingresar el nombre del script que desea encontrar o simplemente una letra con la que comience un script determinado. Como se aprecia en la figura 15.

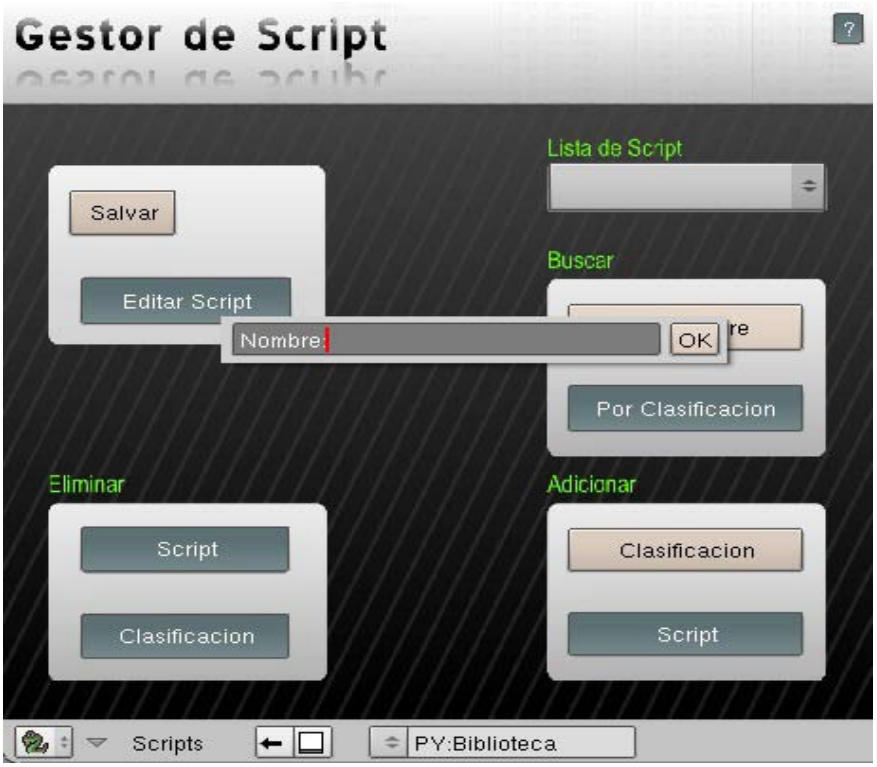



**Figura 17 Interfaz para buscar script**

.

El usuario al dar clic en el botón **Por Nombre** de la ventana **Buscar** se muestra una ventana que le permite teclear el nombre del script que desea buscar, si se encuentra en la biblioteca se muestra en el Editor de Texto de Blender, si no el sistema muestra el mensaje “El script no se encontró”. También existe la opción de teclear una letra, al presionar **ok** se listan los nombres de los scripts que comienzan con dicha letra, al dar clic sobre uno de ellos se muestra en el Editor de Texto de Blender.

El diseñador tiene la opción de buscar un script por su Clasificación como se muestra en la figura 16.


Autores: Fernando Javier Mesa Quiles
Roger Terrero Hernández


**Figura 18 Interfaz para Buscar Script por clasificación**

.

El diseñador al presionar el botón **Por Clasificación** de la ventana **Buscar** escoge la clasificación donde quiere buscar el script y se muestra un listado de los script de esa clasificación como se muestra en la figura 17.

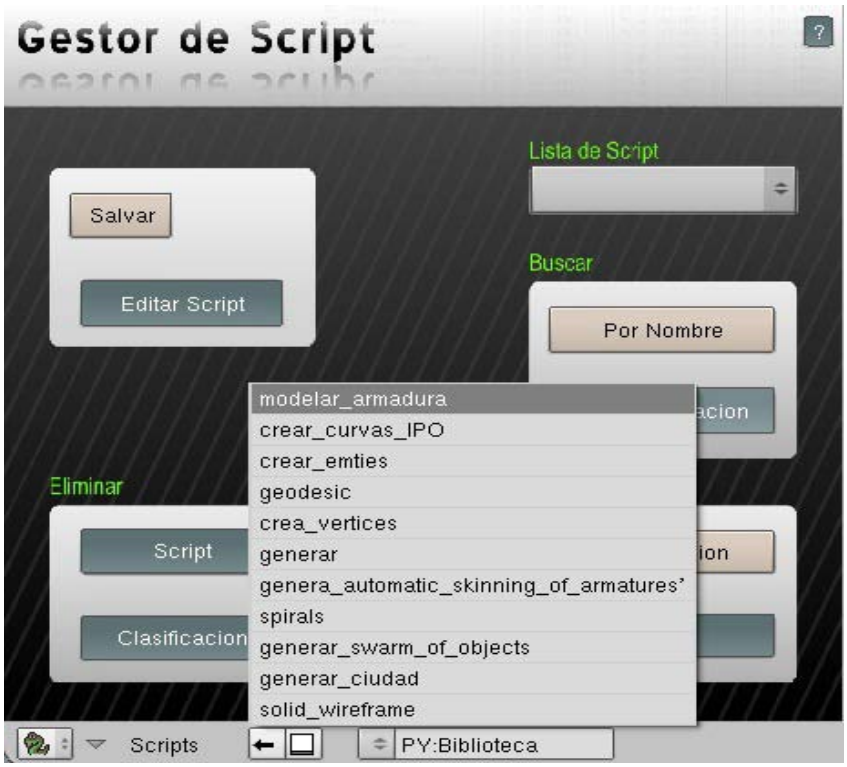



**Figura 19 Interfaz para Buscar Script por clasificación listado.**

Si selecciona un nombre el script se muestra en el Editor de Texto de Blender.

### 3.10.3 Interfaz para Adicionar Script.

Al ejecutar la aplicación el diseñador tiene la opción de Adicionar un script con el que desee trabajar o le sea importante para mantenerlo en la biblioteca. El diseñador al dar clic sobre el botón **Script** de la ventana **Adicionar** se le muestra una nueva ventana de búsqueda como se muestra en la figura 18.

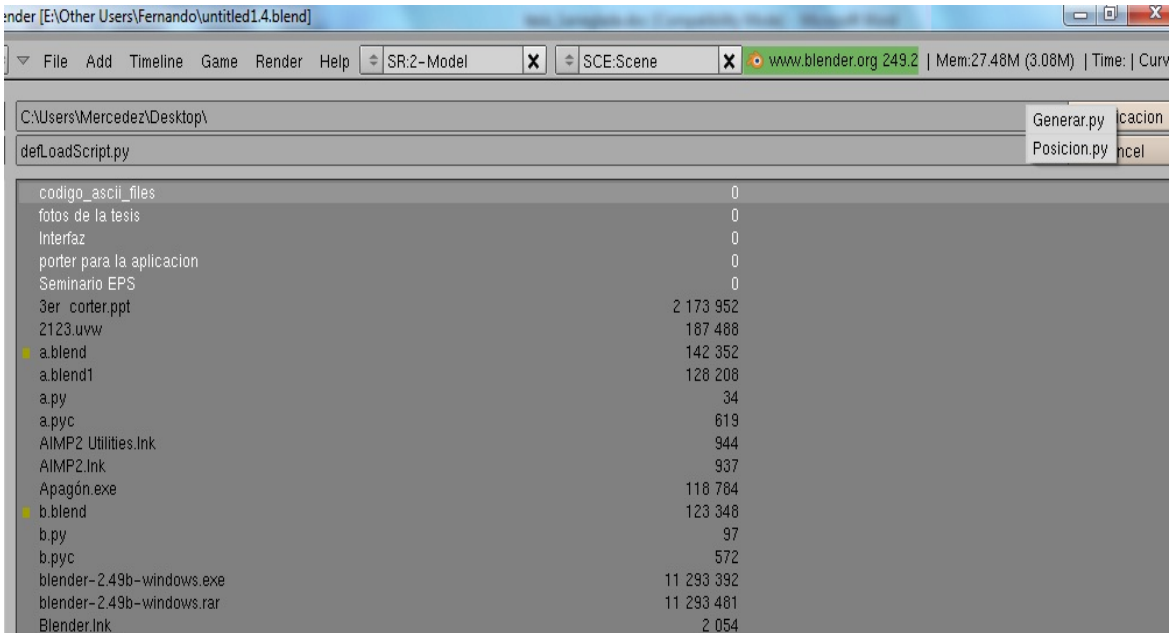



**Figura 20 Interfaz para cargar un script**

Desde esta ventana se puede navegar y seleccionar el script que desee adicionar, una vez seleccionado el script debe dar clic sobre el botón **Clasificación** que aparece en esa interfaz para seleccionar el grupo donde desea guardar el script.

### 3.10.4 Interfaz Listado de Script de la biblioteca.

Como se muestra en la figura 19, el diseñador tiene la opción en la interfaz de poder ver todos los script con que cuenta la biblioteca en ese momento.

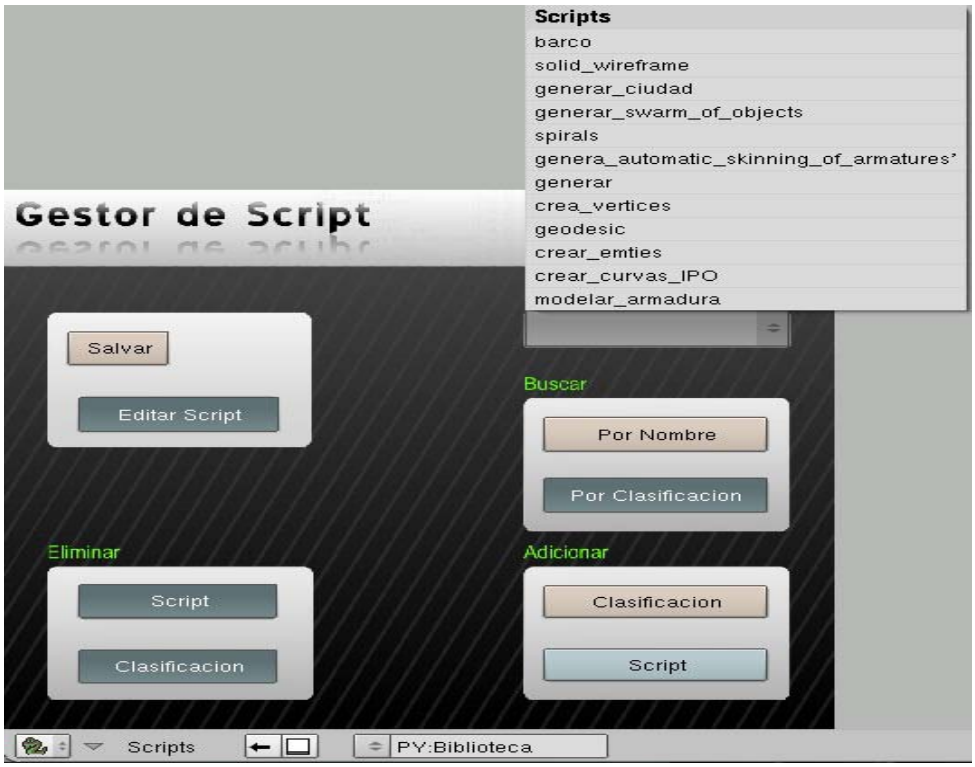



**Figura 21 Interfaz para mostrar los script de la biblioteca**

### 3.10.5 Interfaz para Adicionar Clasificación.

El diseñador tiene la opción en la interfaz de la herramienta de adicionar una clasificación que no esté en la biblioteca como se muestra en la figura 20.


**Figura 22 Interfaz para adicionar una nueva clasificación a la biblioteca.**

### 3.10.6 Interfaz para Eliminar un script.

La interfaz le da la opción al diseñador de eliminar un script de la biblioteca como se muestra en la figura 21.


Autores: Fernando Javier Mesa Quiles
Roger Terrero Hernández


**Figura 23 Interfaz para Eliminar un script de la biblioteca**

El diseñador al presionar el botón **Script** de la ventana **Eliminar** selecciona la clasificación en la cual se encuentra el script que desea eliminar como se muestra en la siguiente figura.


**Figura 24 Lista que se muestra para eliminar un script**

Si el script se encuentra en la lista debe dar clic sobre él para eliminarlo de la biblioteca.

### 3.10.7 Interfaz para Editar un Script.

La interfaz de la aplicación le da la opción al diseñador de poder editar un script determinado para que este sea más eficiente en su ejecución, para editar un script el diseñador presiona el botón **Editar Script** y se le muestran las diferentes clasificaciones con que cuenta la biblioteca como se muestra en la figura 23.


**Figura 25 Clasificaciones del botón Editar script**

El diseñador busca el script que desea editar en algunas de las clasificaciones como se muestra en la figura 24.

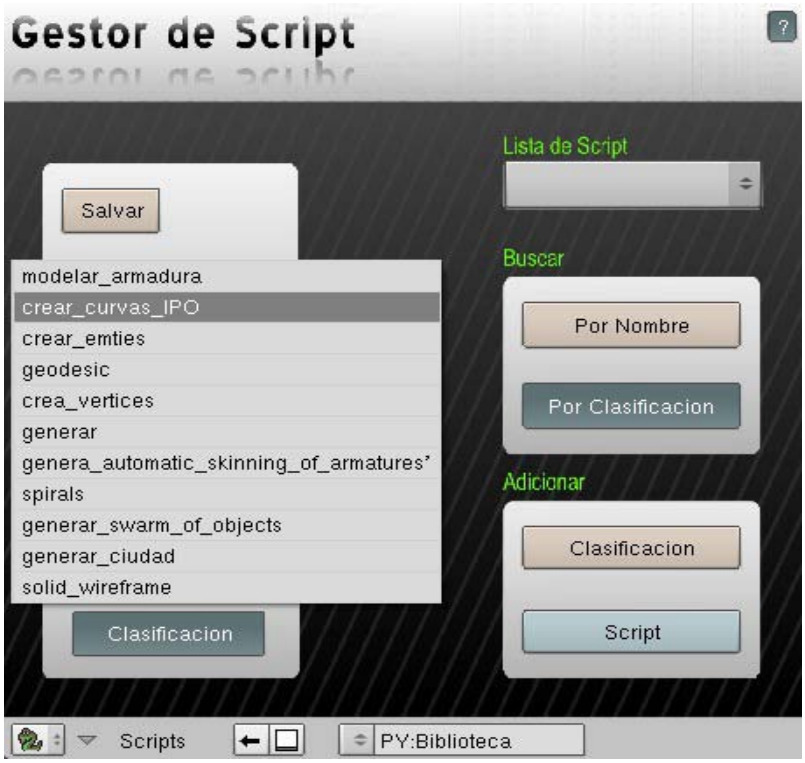



**Figura 26 Lista de script del botón Editar script**

El diseñador escoge un script de la lista y éste se muestra en el Editor de Texto de Blender como se muestra en la figura 25, para poder editarlo. Al terminar el diseñador tiene la opción de salvar los cambios presionando el botón **Salvar**.


Autores: Fernando Javier Mesa Quiles
Roger Terrero Hernández

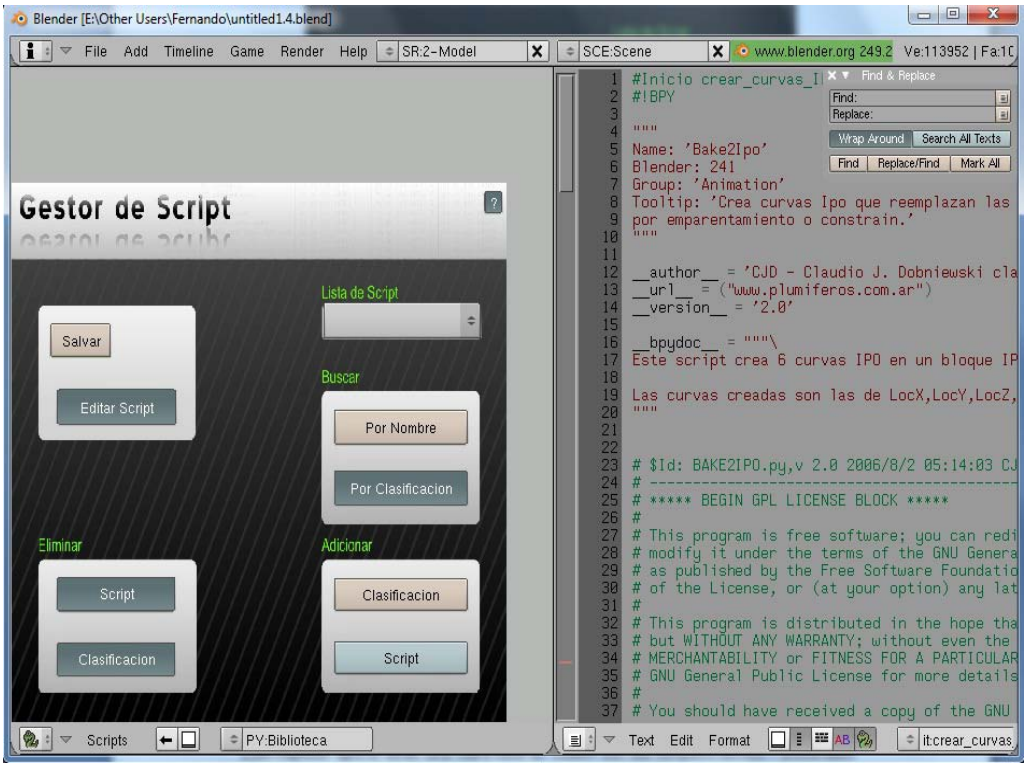



**Figura 27 Editar script**

### 3.10.8 Interfaz para Eliminar una Clasificación.

La herramienta presenta un botón que es para que el diseñador elimine una clasificación en caso de que esta no tenga ningún script o no haga falta en la biblioteca tal y como se muestra en la figura 26.

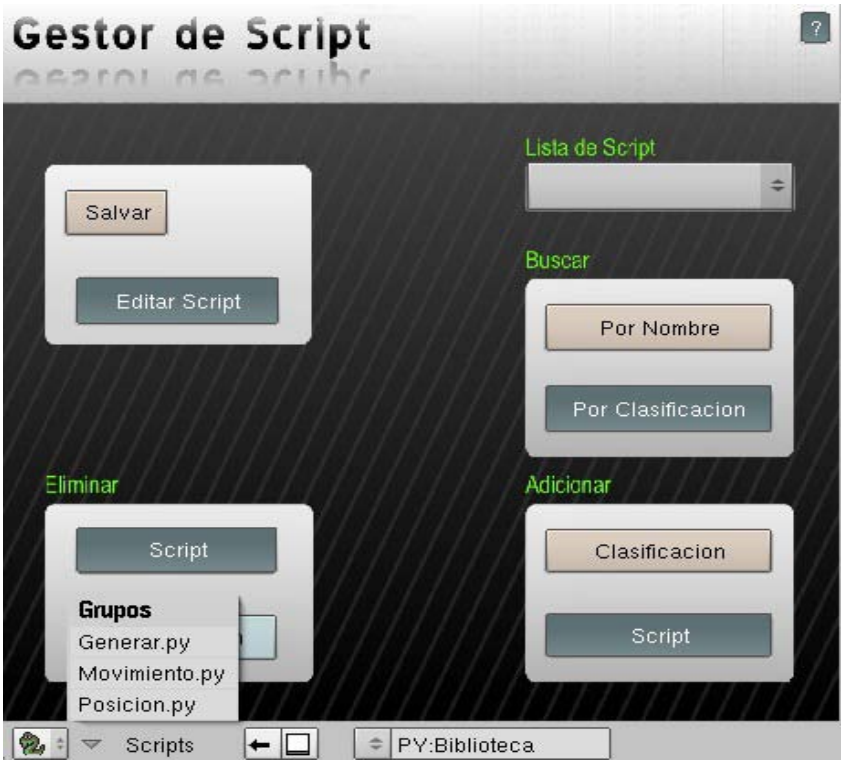



**Figura 28 Eliminar Clasificación**

El diseñador al presionar el botón **Clasificación** de la ventana **Eliminar** debe dar clic sobre la clasificación que desea eliminar.

### 3.10.9 Interfaz para mostrar la ayuda.

La herramienta presenta el botón **(?)**, al dar clic sobre él se muestra un listado con los botones de la herramienta como se muestra en la figura 27.

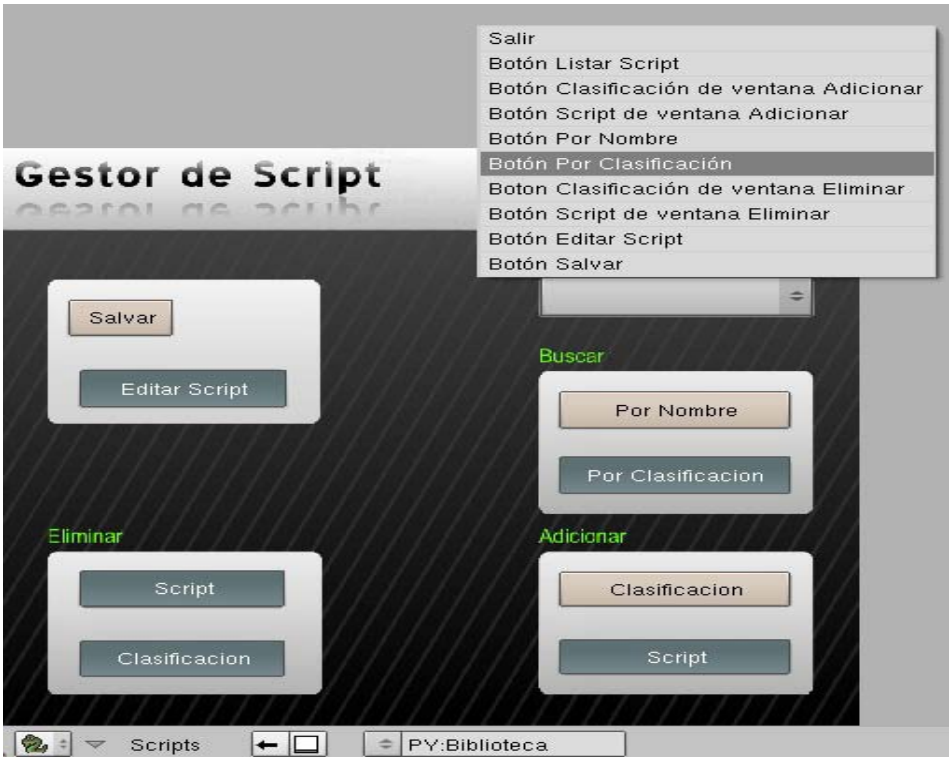



**Figura 29 Interfaz del listado de botones de ayuda**

El diseñador selecciona un botón de los listados y obtiene la explicación de su funcionalidad como se muestra en la siguiente figura.

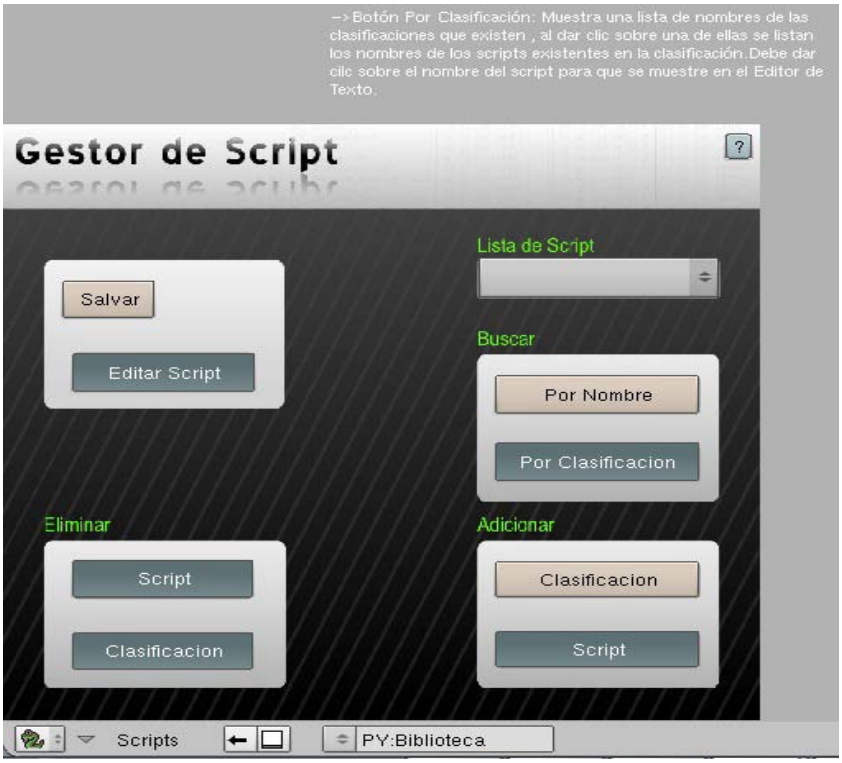



**Figura 30 Interfaz donde se muestra la ayuda**

### 3.11 Consideraciones del Capítulo

En este capítulo se definen los Requerimientos Funcionales y los Requerimientos No Funcionales que forman parte del sistema. A través de los diagramas de Casos de Usos se obtuvo una comprensión de cuáles son las principales funcionalidades que tendrá la aplicación. Luego de haber realizado el modelo conceptual y la descripción de la arquitectura del sistema, se obtiene una visión generalizada de cómo funcionará la aplicación una vez terminada, esto permitirá además que el usuario logre una mejor comprensión y obtenga una claridad de cada una de las acciones que podrá ejecutar haciendo uso de la aplicación, siguiendo los diagramas de secuencia construidos para cada una de las soluciones que realizará el usuario.

Al final de este capítulo se presentaron cada uno de los prototipos de interfaces de usuario que reflejan las diferentes acciones posibles a realizar. Cada una de estas imágenes presentadas le brinda una clara visión al usuario de cómo interactuar con la aplicación, lo cual resulta beneficioso para aquellos usuarios que no tienen mucha experiencia con estos tipos de herramientas.

# Capítulo 4:

# Implementación y Prueba

En este capítulo se presentará la implementación y las pruebas del sistema. Se presentarán las formas de implementación que se realizan para lograr el correcto funcionamiento del sistema. Se verán los distintos estándares de codificación utilizados en la implementación de las clases del diseño, que brindan una mejor comprensión de la estructura diseñada para la definición de los métodos y atributos de las clases. Por último se mostrarán las tablas del diseño de pruebas efectuados en la herramienta para su total funcionamiento.

## 4.1 Implementación del sistema

En la fase de Construcción se encuentra el Flujo de Trabajo de Implementación, el cual comienza con el resultado del diseño y se implementa el sistema en términos de componentes. El flujo de implementación está fuertemente determinado por el lenguaje de programación utilizado. En él se presentan los diagramas de despliegue y componentes que conforman lo que se conoce como modelo de implementación.

Como parte de este epígrafe del trabajo, se presentarán según lo descrito anteriormente, los estándares de codificación utilizados, el diagrama de despliegue y los diagramas de componentes.

## 4.2 Estándares de implementación

Seguidamente se describirán los estándares de codificación propuestos para el desarrollo de esta herramienta. El código de implementación contenido en la herramienta, está compuesto por los estándares de la biblioteca de Python y el estándar utilizado en la herramienta Blender. La otra parte constituye los estándares creados para cumplir con el objetivo básico del sistema. El conocimiento de los estándares de codificación reducirá perceptiblemente el riesgo que los desarrolladores cometan errores; durante el desarrollo los estándares de codificación ayudan a los ingenieros a producir un código de alta calidad y aumentan considerablemente la capacidad de mantenimiento y reutilización del producto final a largo plazo.

### 4.2.1 Métodos.

- def EliminarScript(self): método de los requerimientos de la herramienta que permite eliminar un script de la lista de script.
- def PorClasificacion(self): método de los requerimientos de la herramienta que permite buscar el script por su clasificación que el diseñador desee dentro de la biblioteca.
- def Buscar(self): método que permite al diseñador buscar un script por nombre o por una letra.
- def Men(self): método que le permite al diseñador buscar un script determinado para poder editarlo.
- def SalvarScript(self): método que le permite al diseñado salvar el script editado.
- def EliminarClasificacion(self): método que le permite al diseñador eliminar una clasificación dentro de la biblioteca.
- def ListarPorClasificacion (tipo): método de los requerimientos de la herramienta que permite al desarrollador listar los diferentes script por su clasificación.
- def AdicionClasificacion(self): método de los requerimientos que le permita al diseñador adicionar una nueva clasificación en caso de esta no existir en la biblioteca.
- def Adicionar(self,f): método de los requerimientos que le permite al diseñador adicionar un script a la biblioteca.
- def ListaScript(self): método que le permite al diseñado ver los script que contiene la biblioteca.
- def button_event (evt): global SHOW_HELP, Adicionar, Eliminar, Editar, Adicionar_Clasificacion, movimiento: Este método es el que permite interactuar con los botones de la herramienta (eventos de la herramienta)
- def gui (): este es el método para pintar la interfaz de la herramienta y todos los botones con que cuenta la misma.

- def paragraph(color,text,width,size,staX,staY,spacing,square): método que es utilizado para elaborar la ayuda de la herramienta y para poder escribir textos en la interfaz.

### 4.2.2 Variables.

- Eliminar = 3: Este es un ejemplo de la declaración de una variable global, es decir una variable que va servir en cualquier lugar del código.

- Por ejemplo: if (evt==Eliminar): EliScript().EliminarScript() llama a la variable global que se declaró anteriormente y ejecuta el evento (en tiempo real es cuando el diseñador da clic sobre el botón Eliminar y ejecuta su funcionalidad).

- lista= []: declarada una lista vacía para ser utilizada en un método determinado, es también conocida como una estructura de datos que viene dentro de la librería estándar de Python.

- menu = '': de esta manera se puede declarar una variable de cadena (string) vacía.

### 4.2.3 Funciones y bibliotecas.

- open: función de Python que permite abrir ficheros.

- f=open ('. /Scripts/'+tipo, 'r'): esta línea de código abre un fichero en la dirección que se le pasa por parámetro.

- lines = f.readlines (): permite almacenar en un arreglo las líneas de un fichero.

- readlines: función de Python que permite leer las líneas de un fichero.

- c=Draw.PupStrInput ("Nombre:","",40): permite entrar una cadena en una ventana de texto.

- Draw.PupMenu: permite desplegar un menú en la interfaz de la herramienta al presionar un botón determinado.

- Draw: este módulo permite acceso a una interfaz de ventanas en Blender. Dentro de sus funcionalidades se incluyen varios tipos de botones: Create, PushButton,PupMenu, PupTreeMenu, PupIntImput,PupFloatImput, PupStrImput, PupBlock, Menu, Toggle,

ColorPicker, normal, number, string, text, label, image, etc. También incluye teclas del teclado y del ratón los valores de código en su diccionario.

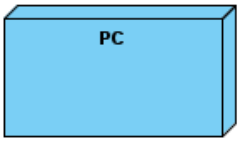
- os: librería de Python.
- listdir: función de la librería de **os**.
- os.listdir ('. /Scripts'): esta línea de código permite listar los nombres de todos los archivos que existen en la dirección pasada por parámetro.
- f.close (): línea de código que permite cerrar un fichero.
- Text.Load (f): línea de código que permite cargar un fichero de texto pasado por parámetro.
- write: función que permite escribir en un fichero.
- f.write("ejemplo"): Escribe al final del fichero f "ejemplo".
- __getslice__ (2,4): es una función que toma un rango de una lista según los valores que se pasan por parámetros.
- len (): determina la longitud de una lista:
- for j in lines: de esta manera es como se pueden recorrer las listas.
- txt =Text.New(d): esta línea permite crear un texto nuevo en el Editor de Texto de Blender.
- lista.append (x): esta línea permite adicionar un elemento a una lista.

### 4.2.4 Código para importar las bibliotecas y módulos que se van a utilizar en la herramienta.

- import Blender: para poder comenzar a programar en la herramienta Blender.
- from Blender import *: código para importar todos los módulos de Blender, de esta manera se puede utilizar diferentes estándares de implementación de la herramienta Blender.

- import os*: código para importar la librería **os** de Python con que se va a trabajar en la aplicación.
- import string: código para importar la librería **string** de Python.
- from defEditar import Edit: importa el módulo de la clase que implementa al método Editar Script.
- from defAdClasif import AdClasificacion: importa el módulo de la clase que implementa al método Adicionar Clasificación.
- from defEliClasif import EliClasificacion: importa el módulo de la clase que implementa al método Eliminar Clasificación.
- from defEliScript import EliScript: importa el módulo de la clase que implementa al método Eliminar Script
- from defSaveScript import SalvScript: importa el módulo de la clase que implementa al método Salvar Script.
- from defListarScript import Lista: importa el módulo de la clase que implementa al método Listar Script.
- from defBuscaScript import BuscarScript: importa el módulo de la clase que implementa el método Buscar Script.
- from defAdScript import AdicionarScript: importa el módulo de la clase que implementa el método Adicionar Script.
- from defBuscaPorClasificacion import BuscarSriptClasificacion: importa el módulo de la clase que implementa el método Buscar Script por Clasificación.

### 4.3 Diagrama de despliegue





Autores: Fernando Javier Mesa Quiles
Roger Terrero Hernández

**Figura 31 Diagrama de despliegue**

El sistema solo se desplegará en una sola computadora (PC), debido a que no dispone de funciones de red para realizar las diferentes operaciones que posee.

### 4.4 Diagrama de Componentes

Los diagramas componentes son utilizados para estructurar el modelo de implementación en términos de subsistemas de implementación y mostrar las relaciones entre los elementos de implementación. Se usan además, para mostrar las dependencias de compilación de los ficheros de código fuente, relaciones de derivaciones entre ficheros de código fuente y ficheros que son resultados de la compilación. Dependencias entre elementos de implementación y los correspondientes elementos de diseño que son implementados.

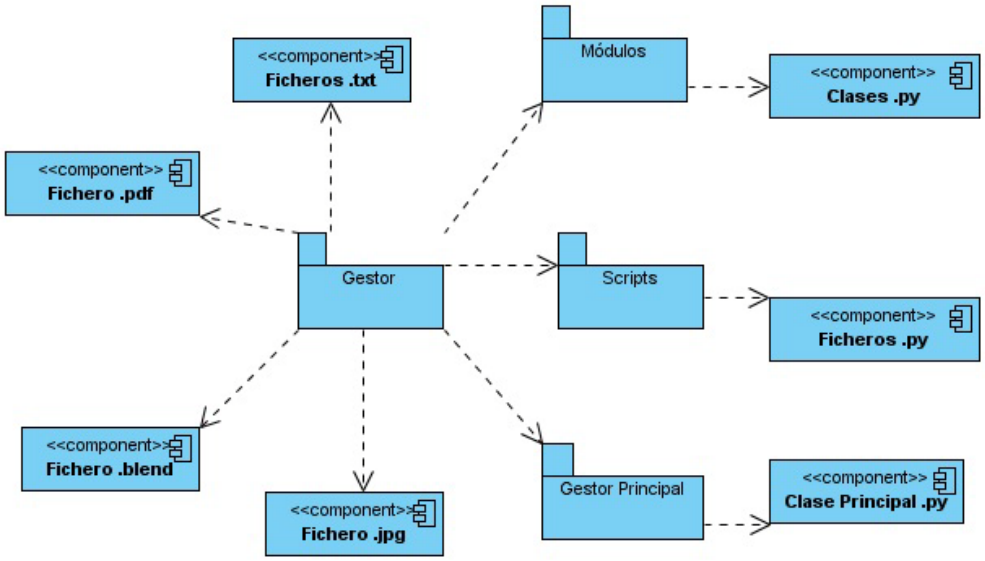



**Figura 32 Diagrama de Componentes**

Este diagrama viene representado por la carpeta Gestor que contiene varias carpetas donde se encuentran los ficheros del código fuente que se utilizan en la aplicación. Todas las carpetas se generan al ejecutar el instalador de la aplicación, cada una de estas carpetas presentan ficheros con la extensión .py pero con funcionalidades distintas. La carpeta Módulos presenta los ficheros que conforman las clases de la aplicación, la carpeta Gestor Principal presenta un

fichero que es la clase principal y la carpeta Scripts presenta los ficheros que conforman la biblioteca de la herramienta.

### 4.5 Pasos para la instalación de la herramienta.

En la siguiente figura el usuario escoge el lenguaje con el cual desea realizar la instalación de la herramienta.


**Figura 33 Escoger idioma**

En la siguiente figura el usuario entra al asistente de instalación.

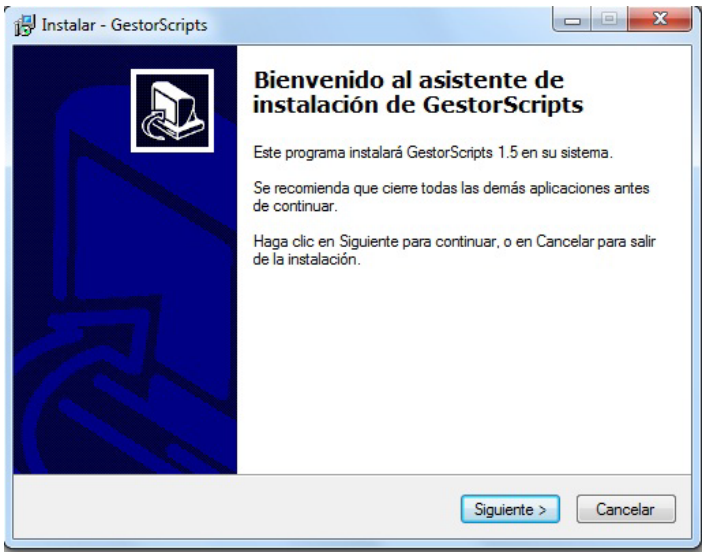



**Figura 34 Inicio de la instalación**

A continuación se muestra la licencia de la aplicación la cual es libre.


**Figura 35 Licencia de la herramienta**

En la figura se muestra la información que debe seguir el usuario después de instalar la herramienta, la cual es esencial para el funcionamiento de la misma.

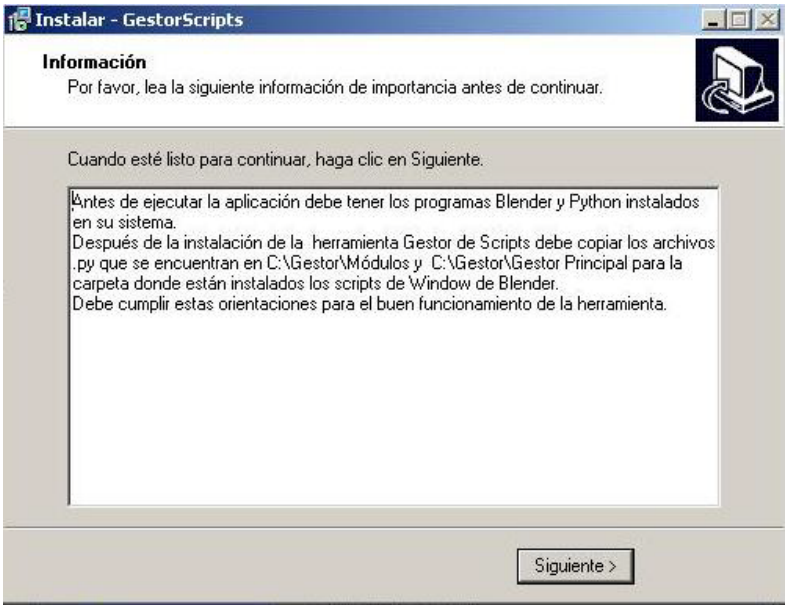



**Figura 36 Pasos para el funcionamiento de la aplicación**

En esta figura se muestra el asistente para terminar la instalación.


**Figura 37 Fin de la instalación**

### 4.6 Descripción de los Casos de Usos de Pruebas

En este epígrafe mostraremos los casos de pruebas utilizados para detectar el mal funcionamiento de la herramienta Gestor de Script para el desarrollo de Paseos Virtuales sobre Blender. El tipo de prueba utilizado es de caja negra porque este tipo de pruebas se centran en lo que se espera de un módulo. Estas pruebas se apoyan en las especificaciones de los requisitos funcionales del sistema, es decir, estas pruebas son estudiadas desde el punto de vista de las entrada que recibe el sistema y las salidas o respuestas que produce, sin tener en cuenta su funcionamiento interno, en otras palabras de las pruebas de caja negra nos interesa su forma de interactuar con el medio que le rodea.

A continuación especificamos cada uno de los casos de pruebas:

#### 4.6.1 Caso de prueba Adicionar Script.

| | |
|---|---|
| **Caso de uso** | Adicionar Script |
| **Caso de prueba** | El script no es esta con la extensión .py |
| **Entrada** | Se selecciona el script. |
| **Resultado** | Se muestra un mensaje al usuario anunciando que el script no esta en formato más indicado. |

| | |
|---|---|
| **Condiciones** | El script tiene una extensión. py |

**Tabla 12 Caso de Prueba para el Caso de Uso Adicionar script**

### 4.6.2 Caso de prueba Adicionar Clasificación.

| | |
|---|---|
| **Caso de uso** | Adicionar Clasificación. |
| **Caso de prueba** | Ya existe una clasificación. |
| **Entrada** | Se adiciona una clasificación. |
| **Resultado** | Se muestra un mensaje al usuario anunciando que la clasificación ya existe. |
| **Condiciones** | La clasificación no debe existir. |

**Tabla 13 Caso de Prueba para el Caso de Uso Adicionar Clasificación**

### 4.6.3 Caso de Prueba Eliminar script.

| | |
|---|---|
| **Caso de uso** | Eliminar script |
| **Caso de prueba** | El script no se encuentra en la biblioteca de script. |
| **Entrada** | Se listan los script por clasificación. |
| **Resultado** | No se puede eliminar porque no se encuentra en la biblioteca de script. |
| **Condiciones** | El script tiene que estar en la biblioteca de script. |

**Tabla 14 Caso de Prueba para el Caso de Uso Eliminar Script**

### 4.6.4 Caso de prueba Eliminar Clasificación.

| | |
|---|---|
| Caso de uso | Eliminar Clasificación. |
| **Caso de prueba** | No existen clasificaciones en la biblioteca. |
| **Entrada** | Se listan las clasificaciones. |
| **Resultado** | No se puede eliminar porque no se encuentran clasificaciones. |
| **Condiciones** | La biblioteca debe tener clasificaciones. |

**Tabla 15 Caso de Prueba para el Caso de Uso Eliminar Clasificación**

### 4.6.5 Caso de Prueba Buscar Script.

| | |
|---|---|
| **Caso de uso** | Buscar script |
| **Caso de prueba** | El script no se encuentra en la biblioteca. |
| **Entrada** | Se pasa el nombre del script que se desea buscar. |
| **Resultado** | Se muestra un mensaje al diseñador que el script no se encuentra en la biblioteca. |
| **Condiciones** | El script tiene que estar en la biblioteca. |

**Tabla 16 Caso de Prueba para el Caso de Uso Buscar Script**

### 4.6.6 Caso de Prueba Listar Script de la Biblioteca.

| | |
|---|---|
| **Caso de uso** | Listar script. |
| **Caso de prueba** | No existe listado de script en la biblioteca. |
| **Entrada** | Se listan los script. |
| **Resultado** | No se encontró listado. |
| **Condiciones** | Deben existir scripts en la biblioteca. |

**Tabla 17 Caso de Prueba para el Caso de Uso Listar Script**

## 4.7 Aspectos significativos

Para la prueba de la herramienta en el grupo de desarrollo de Paseos Virtuales se escogió una muestra de 3 diseñadores los cuales tuvieron la tarea de buscar 6 script en el gestor de bibliotecas de script y buscar los mismos en el grupo de script desorganizados con que cuenta el grupo de desarrollo.

Se les pidió a los diseñadores Yoan González González, Carlos Ernesto Díaz Torres y Yunior Frometa Carbonell que buscaran los siguientes script que están en el Gestor de bibliotecas de script:

**Generar ciudad**

Yoan (demoró 10 segundos), Carlos (demoró 13 segundos), Yunior (demoró 11 segundos)

**Carro todo terreno**

Yoan (demoró 11 segundos), Carlos (demoró 15 segundos), Yunior (demoró 13 segundos)

**Barco**

Yoan (demoró 9 segundos), Carlos (demoró 10 segundos), Yunior (demoró 12 segundos)

**Edificio**

Yoan (demoró 11 segundos), Carlos (demoró 9 segundos), Yunior (demoró 11 segundos)

El primer diseñador buscó los 4 scripts anteriores en un tiempo de 41 segundos, el segundo un tiempo de 47 segundos al igual que el último.

Se les pidió a los diseñadores que buscaran los mismos script en las carpetas donde se encuentran dispersos y sin clasificar arrojando cada diseñador en encontrar los 4 scripts los siguientes resultados:

Yoan-demoró 5 min.

Carlos-demoró 7 min.

Yunior-demoró 6min y 30 segundos.

A continuación se muestra una figura de tiempo en minutos que revela el promedio de tiempo en que demoran los diseñadores en buscar los scripts mencionados en ambos casos. Utilizando el Gestor de Script se obtiene un promedio total de tiempo de 45 segundos y sin utilizar la herramienta de 6 min y 30 segundos.

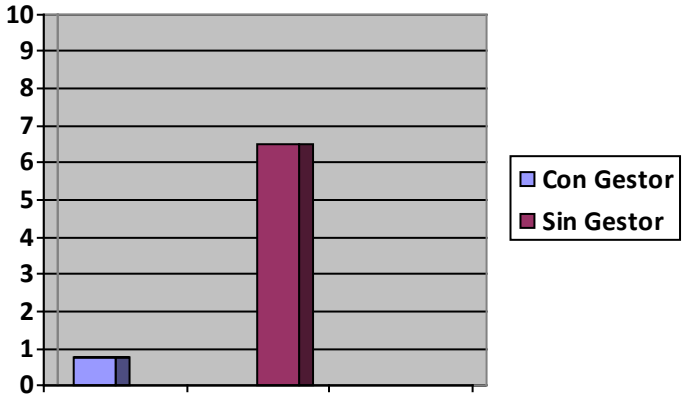



**Figura 38 Gráfica comparativa**

Como se aprecia en la gráfica anterior el gestor de script minimiza considerablemente el tiempo de búsqueda de scripts para la muestra seleccionada ahorrándose un tiempo de 5 min y 45 segundos.

### 4.8 Consideraciones del capítulo.

En este capítulo se presentaron los estándares de implementación de la herramienta para que el usuario alcance una visión más fácil y comprensible del código utilizado para el desarrollo de la aplicación. Se presentó también el diagrama de despliegue de la herramienta. También se presentaron diferentes tablas para describir las pruebas que se le hicieron a la herramienta, estas pruebas se denominan casos de pruebas. Por último se da una comparación entre la realización de un proyecto antes de existir la herramienta y después que la herramienta es terminada.

# Conclusiones

El grupo de desarrollo de Paseos Virtuales de la facultad 5 trabaja en diferentes aplicaciones que en su mayoría necesitan scripts que ya se encuentran implementados, pero debido al problema de desorganización existente en el grupo para el trabajo con estos se decidió almacenarlos según su funcionalidad y clasificarlos dentro de una biblioteca para poder gestionarlos, por lo que se decidió trazar como objetivo principal de este trabajo la realización de una investigación relacionada con el tema y la construcción de un prototipo de aplicación capaz de resolver los problemas existentes en el grupo de trabajo.

Una vez culminada la investigación y el desarrollo del sistema que se propuso, se llegó a la conclusión de que el objetivo fundamental de éste se logró. El *Gestor de Script* implementado permite disminuir considerablemente el tiempo de gestión de los scripts en Blender existentes dentro del grupo de desarrollo y se logra un menor tiempo en las tareas de diseño. Además se clasificaron los scripts con que cuenta el Grupo de Desarrollo de Paseos Virtuales.

# Recomendaciones

- Generalizar el uso de la herramienta para el desarrollo de nuevos productos sobre Blender que requieran del uso de scripts.
- Continuar el enriquecimiento de la biblioteca de script.


Autores: Fernando Javier Mesa Quiles
Roger Terrero Hernández

# Bibliografía

**Fuente electrónica.**


Sitio oficial de Blender 2010.Consultado [14-1-2010] Disponible en:
http://wiki.blender.org/index.php/Doc:ES/Manual/Extensions/Python

Linux-magazine 2008. Consultado [14-1-2010] Disponible en:
http://www.linux-magazine.es/issue/35/050-054_PythonLM35.crop.pdf

Aprenda a Pensar Como un Programador de Python 2002. Consultado [14-1-2010] Disponible en:
http://manuales.gfc.edu.co/python/thinkCSpy.es.pdf

Ibiblio. 2000. Consultado [14-1-2010] Disponible en
http://www.ibiblio.org/pub/Linux/docs/LuCaS/Tutoriales/Python/Tutorial-Python/tut.html

gfx 2008. Consultado [14-1-2010] Disponible en:
http://gfx.programasfull.com/blender-3d-software-libre-diseno-animacion-3d.html

KALAN 2008. Consultado [14-1-2010] Disponible en.
http://kalans.wordpress.com/category/blender/game-engine-blender-blender/

Nexo-Tech 2005. Consultado [4-2-2010] Disponible en:
http://www.nexo-tech.com/srv_realidad.php?section=1&menu=2&submenu=1

Bibliotecas 2010. Consultado [4-2-2010] Disponible en:
http://www.bibliotecas.us/publicas/escolares/biblioteca-informatica/

Curso C++ 2010. Consultado [4-2-2010] Disponible en:

http://www.zator.com/Cpp/E1_4_4b2.htm

Sitio oficial de Python 2009. Consultado [4-2-2010] Disponible en:
http://www.python.org/
Sitio oficial de Blender 2008. Consultado [4-2-2010] Disponible en:
http://www.blender.org/

Sitio oficial de Blender 2009. Consultado [21-2-2010] Disponible en
http://www.blender.org/documentation/249PythonDoc/API_intro-module.html

Sitio oficial de Blender 2009. Consultado [21-2-2010] Disponible en
http://www.blender.org/documentation/249PythonDoc/Draw-module.html#PushButton

taringa 2007. Consultado [21-2-2010] Disponible en
http://www.taringa.net/posts/info/936091/Tutorial:-Como-compilar-blender-con-MSVC.html

emagister 2009. Consultado [21-2-2010] Disponible en
http://grupos.emagister.com/documento/comparativa_3d_max_con_blender/1921-221704

casidiablo 2008. Consultado [21-2-2010] Disponible en
http://casidiablo.net/bibliotecas-compartidas-de-c-en-linux/

skrdz.wordpress 2009. Consultado [21-2-2010] Disponible en
http://skrdz.wordpress.com/2009/09/09/creando-una-libreria-de-clases-dll/

Programar es fácil 2009. Consultado [21-2-2010] Disponible en
http://profeblog.es/blog/alfredo/2009/01/22/creacion-de-librerias-estaticas-en-c/

users.servicios.retecal 2007. Consultado [21-2-2010] Disponible en
http://users.servicios.retecal.es/tjavier/python/Un_poco_de_Python-2.html

Ubuntu 2009. Consultado [21-2-2010] Disponible en
http://www.ubuntu-es.org/?q=node/93791

rableb 2009. Consultado [21-2-2010] Disponible en
http://radleb.net/blog/2009/08/26/crear-un-robot-en-python-para-google-wave-i/

pelogo 2009. Consultado [21-2-2010] Disponible en
http://www.pelogo.org/2009/07/04/python-y-c-en-android/

Documentos de Python 2009. Consultado [21-2-2010] Disponibles en
http://docs.python.org.ar/tutorial/stdlib.html

elornitorrincoenmascarado 2009. Consultado [21-2-2010] Disponible en
http://www.elornitorrincoenmascarado.com/2009/03/historia-de-python.html

juanjoconti 2009. Consultado [21-2-2010] Disponible en
http://www.juanjoconti.com.ar/2009/03/18/la-historia-de-python-el-principio-del-diseno-y-desarrollo-del-lenguaje/

swaroopch 2004. Consultado [21-2-2010] Disponible en
http://www.swaroopch.com/notes/Python_sp-ar:Prefacio

docs.kde 2006. Consultado [5-3-2010] Disponible en
http://docs.kde.org/development/es/extragear-multimedia/amarok/script_manager.html

abdulet 2010. Consultado [5-3-2010] Disponible en
http://abdulet.net/?p=418

chrome. 2010. Consultado [5-3-2010] Disponible en
http://www.chrome-es.com/blank-canvas-script-handler-gestor-de-scripts/

ima.udg 2010. Consultado Consultado [5-3-2010] Disponible en
http://ima.udg.edu/~amendez/TIC2001/docs/EnginesUJI.pdf

Sitio oficial de Ogre 2008. Consultado [5-3-2010] Disponible en

www.ogre.org

Sitio oficial de panda3d 2008. Consultado [5-3-2010] Disponible en www.panda3d.org

**Tesis:**

Minardo Gollún González López, Kirenia Rojas Escobar. 2007 *MODIFICADOR DE ANIMACIONES DE HUESOS*. Consultado [5-3-2010]

Antonio Guerrero Cabrera, Joel Hernández Mendoza. 2008 *Herramienta para la conversión de formatos de escenas 3D*. Consultado [5-3-2010]

Reynaldo Heredia Rodríguez, Emilio Salcerio Paz. 2009 *Aplicación Web para el control del Proceso de Tesis de Grado en la Facultad 8*. Consultado [5-3-2010]

Zusel Días Pérez, Adony González Cárdenas 2009 Sistema Gestor de Trabajos de Diplomas. Consultado [10-5-2010]

**Libros:**

Guido van Rossum 1999 *Guía de aprendizaje de Python* Consultado [5-3-2010]

Raúl González Duque 2008 *Python para todos* Consultado [5-3-2010]

.

# Referencias Bibliográficas

Michael Louka, Noviembre 1998, An Introduction to Virtual Reality. [Fecha de consulta: 25 noviembre 2009]. Disponible en: http://www.ia.hiof.no/~michaell/home/vr/vrhiof98/index.html

Sitio oficial Blender.2008. *Blender*. [Fecha de consulta: 8-12-2009] Disponible en: http://www.blender.org

Motores de Juego, UEM (Universidad Europea de Madrid), 2007 *Motor de Juego*. [Fecha de consulta: 3-12-2009] Disponible en: http://www.esi.uem.es/jccortizo/motores.html/

Sitio oficial Python, 2008. *Python*. [Fecha de consulta: 18-1-2010] Disponible en: http://www.python.org/

Desarrolloweb 2008. *Scripts*. [Fecha de consulta: 25-1-2010] Disponible en: http://www.desarrolloweb.com/scripts/

neoteo. 2007 *Blender* [fecha de consulta: 3-12-2009] Disponible en http://www.neoteo.com/blender-la-alternativa-gratuita-al-diseno-3d.neo

# Glosario de términos

**Realidad virtual:** Te permite explorar un mundo generado por computadoras a través de tu presencia en él.

**Modelo matemático**: Describe un "espacio tridimensional".

**Simulación:** Es la experimentación con un modelo de una hipótesis o un conjunto de hipótesis de trabajo.

**Ficheros:** Los ficheros informáticos facilitan una manera de organizar los recursos usados para almacenar permanentemente datos en un sistema informático.

**Renderizado:** Es un término usado para referirse al proceso de generar una imagen desde un modelo. Este término técnico es utilizado por los animadores o productores audiovisuales y en programas de diseño en 3D.

**Texturizar:** Da a una capa la apariencia de tener una textura de otra capa. Por ejemplo, puede conseguirse que una imagen de un árbol tenga textura de ladrillos, y controlarse la profundidad de dicha textura, así como la fuente de luz que posee. En calidad óptima, la capa de la textura se coloca y se escala con una precisión a nivel de los subpixeles.

**Multiplataforma:** Es un término usado para referirse a los programas, sistemas operativos, lenguajes de programación, u otra clase de software, que puedan funcionar en diversas plataformas. Por ejemplo, una aplicación multiplataforma podría ejecutarse en Windows en un procesador x86, en GNU/Linux en un procesador x86, y en Mac OS X en uno x86.

**Motor de 3D:** Es un software de sistema diseñado para la creación y desarrollo de videojuegos.

**Multitexturas:** Es el uso de más de una textura a la vez.

**Plugins:** Es una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy específica. Esta aplicación adicional es ejecutada por la aplicación principal e interactúan por medio de la API (interfaz de programación de aplicaciones).

**Ejecutable:** Es un archivo binario cuyo contenido se interpreta por el ordenador como un programa. Generalmente, contiene instrucciones en código máquina de un procesador en

concreto, pero también puede contener bytecode que requiera un intérprete para ejecutarlo. Además suele contener llamadas a funciones específicas de un sistema operativo (llamadas al sistema).

**Python:** Lenguaje de programación interpretado, es decir está diseñado para ser ejecutado por medio de un intérprete, en contraste con los lenguajes compilados.

**Blender:** Es un programa informático multiplataforma, dedicado especialmente al modelado, animación y creación de gráficos tridimensionales.

**Requerimientos:** En la ingeniería de sistemas, un requerimiento es una necesidad documentada sobre el contenido, forma o funcionalidad de un producto o servicio.

**Aplicación:** Es un tipo de programa informático diseñado como herramienta para permitir a un usuario realizar uno o diversos tipos de trabajo. Esto lo diferencia principalmente de otros tipos de programas como los sistemas operativos (que hacen funcionar al ordenador), las utilidades (que realizan tareas de mantenimiento o de uso general), y los lenguajes de programación (con el cual se crean los programas informáticos).

**Plataformas:** Es precisamente el principio, en el cual se constituye un hardware, sobre el cual un software puede ejecutarse/desarrollarse.

**Interfaz:** Es el medio con que el usuario puede comunicarse con una máquina, un equipo o una computadora, y comprende todos los puntos de contacto entre el usuario y el equipo, normalmente suelen ser fáciles de entender y fáciles de accionar.

**Multiparadigma**: soporta varios paradigmas o estilos de programación.

**Paradigmas de Programación:** representa un enfoque particular o filosofía para la construcción del software.