Editor de escenas gráficas

Enoc Martinez Masip

Resumen: A lo largo de los últimos años el sector de las máquinas tragamonedas ha sufrido un gran cambio tecnológico, pasando de ser simples maquinas con microcontroladores y rodillos mecánicos, a maquinas complejas multipantalla que incorporan ordenadores (maquinas con microprocesador ARM CortexA, x86, x64) y deben reproducir pequeños juegos como si de una videoconsola se tratara. Estas nuevas características crean nuevos desafíos para el desarrollo de los juegos, siendo necesario el diseño de escenas visuales para estos. De esta necesidad nace este proyecto, con el objetivo de agilizar y simplificar el desarrollo de estas escenas en la medida de lo posible, mejorando de esta forma la productividad.

Tragamonedas, juego, editor, escenas, OpenGL, c++, Qt, gui, eventos, lua, LuaJIT, imgui, scripting, recreativa, videojuego, renderizado, grafos, secuencias graficas

Abstract: Over the last few years, the slot machine sector has undergone a great technological change, going from being simple machines with microcontrollers and mechanical reels, to complex multi-screen machines that incorporate computers (machines with microprocessor ARM Cortex A, x86, x64) and Must play small games as if it were a game console. These new features create new challenges for the development of the game machines, being necessary the design of visual scenes for these. From this need is born this project, with the aim of streamlining and simplifying the development of these scenes as much as possible, thus improving productivity.

Slot Machines, game, editor, scenes, opengl, c++, Qt, gui, events, lua, LuaJIT, imgui, scripting, game machine, videogame, render, graph, graphic sequences.

1 INTRODUCCIÓN

Este proyecto ha sido desarrollado como trabajo de fin de grado en colaboración con la empresa Interseven Gaming Team SL, cuyo negocio principal es el desarrollo de juegos para máquinas tragamonedas (Bar/Salón/Casino), se ha decidido proceder al desarrollo de un editor de escenas para así poder ser más productivos y obtener mejores resultados en el trabajo diario, además de resolver algunos de los problemas que se daban con las formas de desarrollo utilizadas hasta el momento.

A la hora de desarrollar videojuegos hay que tener en cuenta los diferentes aspectos en los que nos enfrentaremos. Uno de los que requiere más trabajo es el diseño/implementación de las "escenas" que se mostraran por pantalla, ya que aquí es donde los jugadores obtienen el feedback visual. Es el elemento en el que transcurre la partida. Este elemento es común para todos los juegos que utilicen gráficos por computador. Para llevar a cabo el diseño de las escenas es típico que las empresas desarrollen sus propios editores de escenas, o adquieran uno de los comerciales que hay en el mercado (Sección 3). Estas herramientas se utilizan para agilizar tareas como son la ubicación de los elementos visuales, animaciones, reproducción de sonidos, configuración de los eventos, previsualizar el estado actual del juego, ubicar los objetos dinámicos, todo esto sin necesidad de estar recompilando el juego. La idea detrás de un editor es que todos estos aspectos puedan ser desarrollados sin conocimientos de programación. Ya que gran parte del desarrollo para el que está orientada la herramienta son aspectos más de diseño y artísticos, y estos perfiles no tienen porque ser desarrolladores. Aunque simplifican las tareas de diseño, también suelen incorporar mecanismos para utilizar Scripts, estos scripts son desarrollados en algún lenguaje destinado para este fin Lua **[1]**, Python **[2]**, Squirrel **[3]**, Mono C# **[4]**... son algunos de lenguajes de scripting más usados en el mercado. Convirtiendo así estas herramientas en el núcleo desde el que se desarrollan todos los aspectos no artísticos de los videojuegos.

[En la **Figura 1** se muestra una escena de las que se persigue desarrollar con la herramienta.]

El siguiente documento está organizado en las siguientes secciones:

- Objetivos
- Estado del arte
- Metodología seguida durante el desarrollo.
- Resultados
- Conclusiones
- Líneas de trabajo futuro.
- Agradecimientos
- Bibliografía

Figura 1 Ejemplo de escena que se quiere poder crear.

2 OBJETIVOS

2.1 Objetivos generales

El objetivo del proyecto es el diseño y desarrollo de una herramienta para el diseño y animación de los aspectos interactivos en los juegos de las Slot machine. Esta herramienta tiene que ser lo más intuitiva posible y capaz de agilizar el proceso de desarrollo de los juegos, es fundamental poder visualizar el juego en tiempo real y poder pre visualizar distintas escenas. Las escenas que se desarrollarán con la herramienta, si bien tendrán cierta complejidad, en esta primera versión de la aplicación se ofrecerán mecanismos para realizar las acciones más típicas como son la captura de entrada/salida, gestión de eventos y una serie de transformaciones geométricas. Relegando a un lenguaje de Scripting la realización de las tareas con más detalle y sofisticación, que requieran de un ajuste más fino. La salida es un juego que se debe poder ejecutar sin la necesidad del editor.

2.2 Objetivos específicos

Los requisitos que debe cumplir el proyecto para considerarlo como satisfactorio son los siguientes.

- La aplicación debe ser multiplataforma (Linux, Windows, Mac)
  -Para dicha labor se ha decidido desarrollar la aplicación utilizando el framework Qt **[5]**. Este framework con licencia LGPL3 incluye una API GUI multiplataforma.

- Su uso tiene que ser bastante ágil y simple (según el criterio de los usuarios potenciales de Interseven) para ello la interfaz tiene que permitir drag and drop de los diversos elementos, así como intentar que su uso sea lo más evidente posible.

- Posibilidad de secuenciar elementos en una escena mediante las siguientes acciones
  - Conexión de eventos
  - Transformaciones
  - Trayectorias
  - Reproducción de audio
  - Entrelazamiento escenas
  - Reproducción de videos

- Permitir crear componentes reusables.
  -Para ello se pueden crear Meta objetos con los propios componentes del editor o desarrollarlos directamente en Lua.

- El resultado se debe poder compilar como un juego standalone.
  -El editor es un software independiente, el juego se debe poder ejecutar sin la necesidad el editor. Esto obliga a ciertos diseños arquitecturales a la hora de desarrollar la aplicación.

- Facilidad para que los juegos desarrollados con la herramienta se puedan ejecutar en otras plataformas
  - El diseño del cliente standalone que ejecutara los juegos tiene que ser fácilmente portable a otras plataformas como pueden ser Web y móvil.

- Extensibilidad de la aplicación
  -Añadir nuevas características a la aplicación tiene que ser fácil.

- Soporte para scripting (Todo debe poder realizarse con scripting además de directamente con la herramienta).
  - El editor expondrá una API hacia el lenguaje LUA desde el cual se podrá extender cualquier detalle.

- Performance

- Los "juegos" desarrollados con la herramienta se ejecutarán en máquinas con unas prestaciones limitadas, es muy importante el correcto uso de los recursos de los que se dispone.

- Integrable con el resto de la plataforma de la empresa.
  -El proyecto ha sido desarrollado para la empresa Interseven Gaming Team SL. La cual tiene actualmente en el mercado español más de 50.000 máquinas de juego. La tecnología utilizada en el editor tiene que ser compatible con la ya existente en el mercado.

3 ESTADO DEL ARTE

Actualmente existen cantidad de herramientas para agilizar el desarrollo de videojuegos, quizás la más popular actualmente es **Unity** [6] Esta herramienta es una de las más completas que existen en el mercado, siendo además gratuita mientras no se superen unos ingresos de 100.000$ anuales con los productos desarrollados con la misma, ofrece un motor de avanzado renderizado 3D, motores físicos, 3 lenguajes distintos para scripting, simulación, previsualización en tiempo real…
Las razones principales por las que no es considerado como opción y se opta a desarrollar una herramienta propia son:
- Royalties cuando se utiliza en slot machine por máquina
- No es compatible con Linux (Esto añade un coste extra).
- Costes añadidos por trabajador.
- Curva de aprendizaje

En la **Figura 2** se puede ver una escena 3D desarrollada

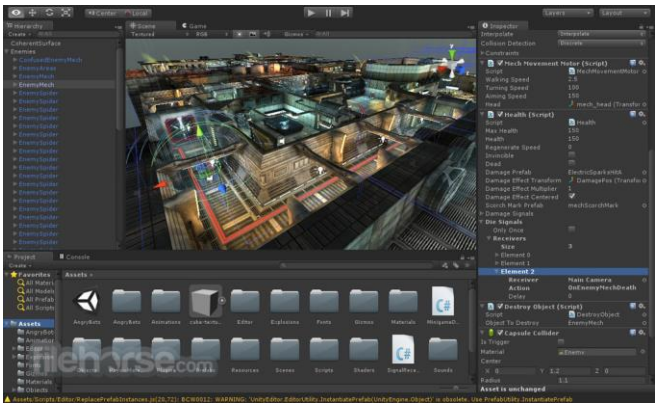

Figura 2 Escena desarrollada en unity

Otra herramienta interesa es el framework **Sony ATF** [7]: desarrollado por Sony (el cual es usado por varios estudios de desarrollo) este framework está pensado para la creación de herramientas como la que se persigue en este proyecto, tal y como se puede apreciar en la **Figura 3**

Está desarrollado en C# con licencia Apache, por lo que se puede utilizar libremente el código fuente. Incluye una base para un editor de escenas. En nuestro caso el uso de este framework ha sido descartado por algunas razones similares a las del anterior, el desarrollo está basado en Windows, mientras que la plataforma que utilizamos es Linux.

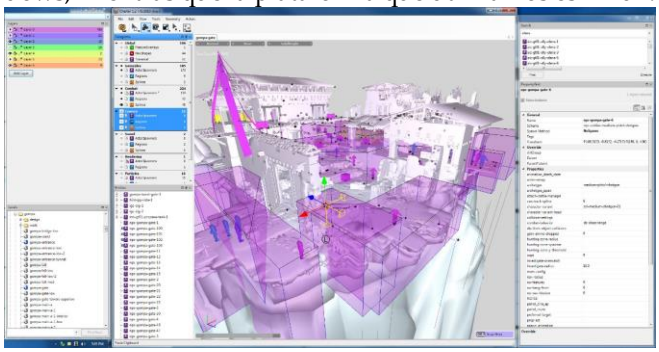

Figura 3 Charter, herramienta de edición basada en ATF de Naugthy Dog

4 METODOLOGÍA I DESARROLLO

4.1 Introducción

Para organizar el desarrollo del proyecto se han tomado algunas ideas organizativas de la metodología Scrum **[8],** como son el desarrollo iterativo basado en Sprints, las ventajas que nos ofrece un desarrollo basado en Sprints es que siempre tenemos una versión de la aplicación funcionando, a la que de forma iterativa le vamos añadiendo características. Estas características no pueden ser seleccionadas al azar ya que para implementar algunas, necesitaremos otras previas. La planificación completa de tareas se puede encontrar en el **Anexo A3**.

Para obtener los resultados esperados se ha realizado previamente una extensa lectura e investigación de las técnicas empleadas en el desarrollo tanto de herramientas para desarrollar videojuegos como de los motores y sistemas implicados en la misma. Los que se han utilizado como referencia para este desarrollo son Game Engine Architecture **[9],** Game Coding Complete **[10]**, Game Design Patterns **[11]** así como diversos libros de referencia del lenguaje C++ en sus versiones 2011 y 2014 **[12]**. Parte de la bibliografía utilizada está incluida en las recomendaciones de lectura para ingenieros de software de Intel del año 2014, por la portabilidad a otros ámbitos de los conocimientos adquiridos. Para las tareas de integración el lenguaje de scripting y entender su funcionamiento han sido fundamentales los conocimientos adquiridos en la asignatura de Compiladores, así como para el desarrollo de los motores de renderizado los conocimientos adquiridos en Visualización Grafica Interactiva.

Los diferentes módulos desarrollados que dan vida al proyecto son los siguientes:

4.2 Previsualizador inicial

Una de las piezas fundamentales para que la herramienta sea útil, es la posibilidad de previsualizar el desarrollo sin necesidad de recompilar a cada cambio, para eso una de las pestañas es un visor donde se puede ver la "ejecución" de todas las secuencias introducidas.
En la **Figura 4** se puede apreciar en una escena siendo reproducida.

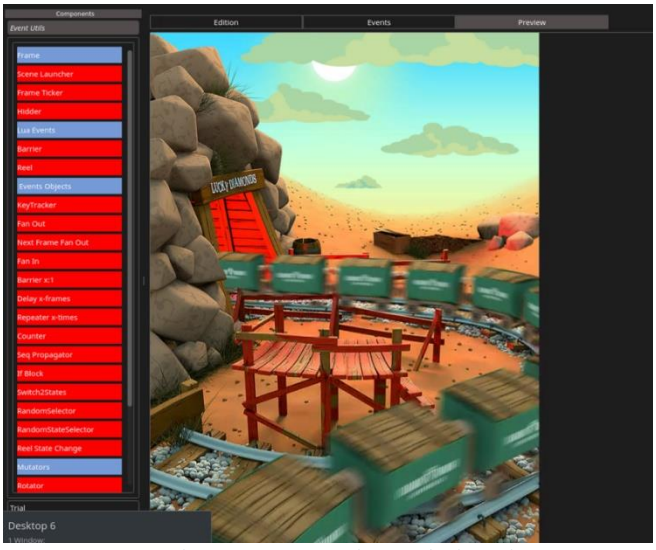

Figura 4 Escena de "La mina" creada con el editor de escenas.

Para poder previsualizar una escena las etapas que se realizan son las siguientes:
- Obtener todos los eventos para esa escena
- Separar los nodos que inician eventos en el mismo frame de los que se encolan para un frame posterior. **Figura 5**
- Construir todos los grafos de eventos
- Cada grafo e eventos debe ser acíclico de lo contrario esa escena acabara en un desbordamiento de stack. **(Sección 4.6)**
- Obtener todos los objetos de juego en esa escena
- Eliminar todas las conexiones de eventos previas.

- Conectar todos los eventos a sus correspondientes objetos
- Cargar los eventos conectados al punto de entrada y comenzar la ejecución.

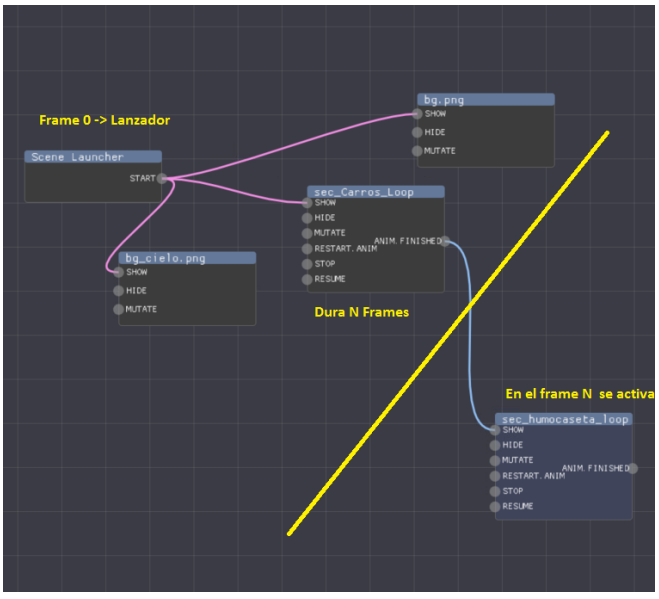

Figura 5 Eventos de la escena de la Figura 4, hay 2 puntos de origen uno desencadenado por el Scene Launcher en el Frame 0 y otro desencadenado al acabar la animación sec_carros_loop que será en su frame correspondiente.

Aunque la forma en la que se construyen las escenas es menos propensa a errores que la utilizada actualmente (Escribir en C todo, sin ningún mecanismo de sincronismo establecido) hay que conocer el momento en el que se ejecuta cada evento de los que se pueden desencadenar para evitar posibles errores mientras no se conoce al detalle cómo funciona la aplicación se hace la construcción de un grafo acíclico para intentar detectar todas las situaciones de desbordamiento posibles y detectarlas en tiempo de "construcción".
Para más información acerca de los grafos acíclicos se puede consultar el enlace en **[13].**

4.3 Transformaciones

Las trasformaciones se realizan mediante la manipulación de las propiedades correspondientes a su posición, ángulo. **Ver Figura 6.**

Propiedad	Valor
name	nubes_02.png
x	384
y	863
angle	0
channel	1
width	368
height	196
scale	1

Figura 6 Propiedades afectadas por transformaciones

Si en edición cambiamos alguna de estas propiedades podemos ver cómo se realiza la transformación correspondiente, aunque su potencial llega cuando son usadas para efectos.

En la **Figura 7** se puede ver el resultado de cambiar propiedades de posición a una imagen.

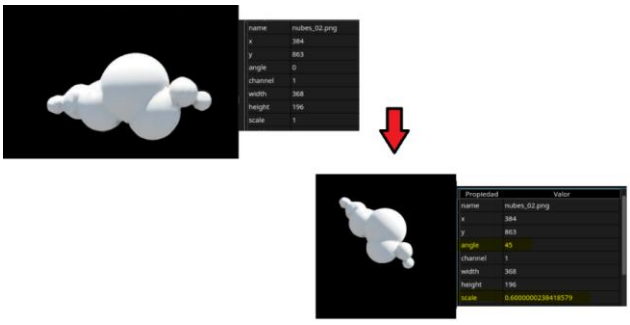

Figura 7 Imagen con ángulo y escala modificados

Al ser un mundo en 2 dimensiones, las transformaciones que se realizan son todas afines y utilizan transformaciones homogéneas para su cálculo. Se puede obtener más información de las transformadas homogéneas en **[14]**.

4.4 Animaciones con transformaciones.

Para realizar animaciones con las transformaciones, se ha incorporado la posibilidad de variar las propiedades descritas anteriormente en (4.3. Transformaciones).
Para hacer esto se introducen los interpoladores, los cuales se encargan de variar un valor de una propiedad a otro en un intervalo de tiempo. Es posible definir mediante el panel de Eventos objetos de tipo interpolador. Las propiedades de cada interpolador son las siguientes.

Rotación	Escalado	Traslación
Frames		
Angulo	EscalaX	Pos.X
	EscalaY	Pos.Y

Mientras que los eventos que desencadenan o las señales que pueden recibir son comunes para todos los interpoladores. En la **Figura 8** se aprecia el aspecto del objeto interpolador en el panel de Eventos.

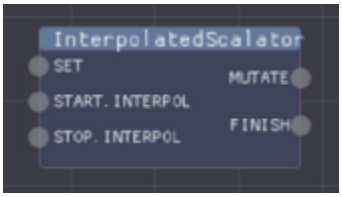

Figura 8 Caja de objeto interpolador

Para que se realice la interpolación es necesario conectar la salida "Mutate" del interpolador a las entradas "Mutate" de objetos interpolables, que son todos aquellos que dispongan de una entrada "Mutate" La conexión resultante será de *color verde*. En el caso de realizar una conexión errónea aparecerá en ***color rojo***.

En la **Figura 9** se muestra un ejemplo de uso de un interpolador rotator. Así como un uso incorrecto.

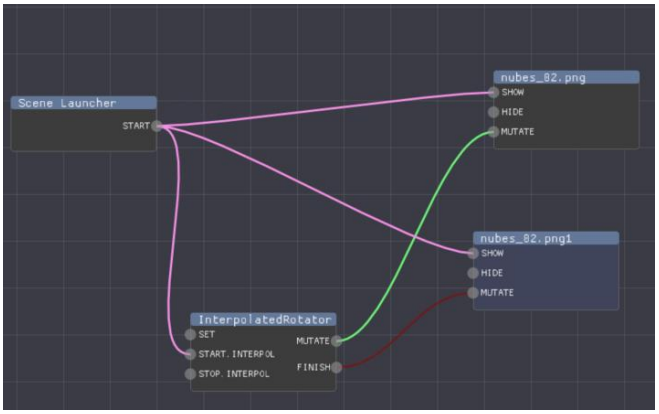

Figura 9 Interpolador Rotación

A una misma entrada "Mutate" se le pueden conectar todos los Interpoladores que queramos. Además, el interpolador al finalizar lanzara el evento Finish. Con lo cual se pueden conectar varios interpoladores en serie creando efectos más sofisticados.

En la **Figura 10** se pueden ver varios interpoladores afectando a un mismo componente gráfico. Los interpoladores marcados con 1 son los que se ejecutaran simultáneamente, mientras que el marcado con 2 inicia su proceso al finalizar el interpolador al que está conectado. Además, se puede identificar según el color de la conexión el tipo que es.

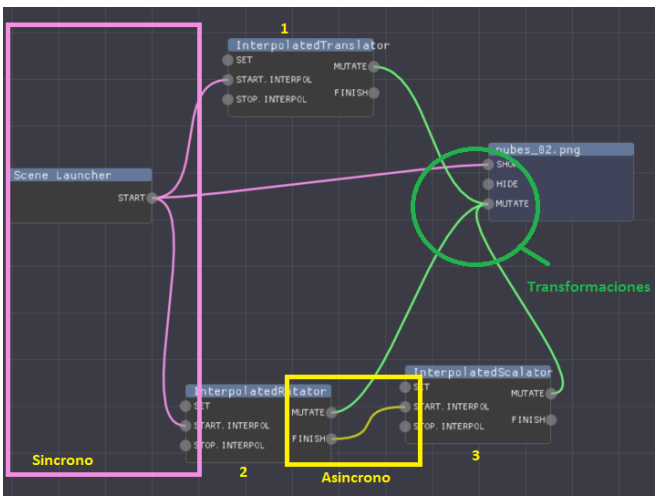

Figura 10 Escena con más interpoladores

4.5 Sistema de componentes

La idea detrás del sistema de componentes es que utilizando lo mostrado anteriormente (junto a más componentes) puedas construir comportamientos complejos y estos sean guardados como componentes.

Actualmente hay 2 componentes creados así, uno es el visor y otro el rodillo.

El componente visor se basa en un rectángulo donde aparece texto, el texto que aparece no forma parte del juego, sino que puede estar grabado en dispositivos físicos y simplemente es transmitido para que el juego lo capture.

Para homogenizar el uso de estos visores con el resto de elementos del juego. En la práctica los campos para fuente y texto aparecen como propiedades adicionales a las comunes en cualquier objeto gráfico. Como se ha mostrado en la Figura 6.

4.6 Grafo de eventos

La secuenciación de la escena se realiza mediante la conexión de elementos en un grafo. Aparte de los objetos "Gráficos" existen un conjunto de elementos añadidos para hacer comportamientos más elaborados, estos se pueden ver en la **Anexo A2**. Esta parte es extensible, por lo tanto, en el futuro habrá más componentes aun para automatizar más tareas siempre en función a las necesidades que vayan surgiendo con el uso de la herramienta.
El secuenciamiento además cuenta con 2 formas de ejecutarse *(síncrono/asíncrono*). Los eventos *síncronos* son aquellos en los que el evento es tratado en ese mismo frame. Mientras que un evento *asíncrono* será tratado en frame posteriores, pudiendo incluso a no ejecutarse nunca si la escena no llega a ese momento. En el grafo de eventos se puede ver que los eventos síncronos y los asíncronos difieren en el color **(Figura 5)**. A consecuencia de esto, encadenar una cantidad de eventos "síncronos" demasiado grande (o infinita) pueda causar un desbordamiento, ya que sería equivalente a una cadena de llamadas recursivas **Figura 11.**

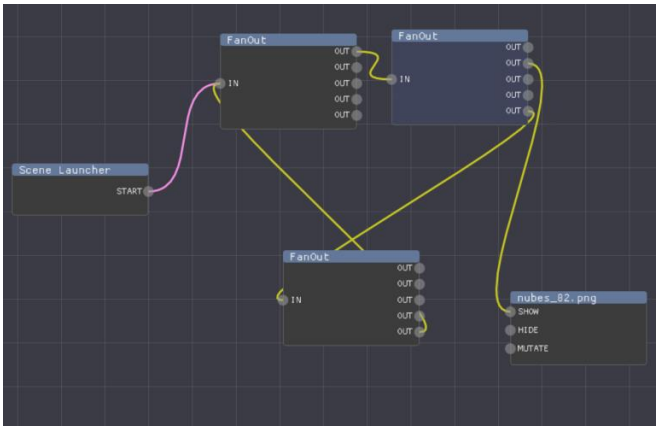

Figura 11 Cadena de llamadas síncronas (Produce StackOverflow)

4.7. Grafo de escena

Para la propagación de movimientos se utiliza un grafo de escena, esto consiste en una jerarquía de elementos, los cuales se rotan de forma relativa entre ellos. Por ejemplo: E*n el sistema solar la luna gira sobre la tierra, mientras que la tierra gira alrededor del sol, propagándose sus rotaciones.*

5 Resultados

5.1 Renderizado de imágenes

El renderizado de imágenes se realiza utilizando OpenGL 4.5 (Core Profile). Esta forma de trabajo es la conocida en el mundo de los gráficos por computador como "OpenGL Moderno". Para poder desarrollar los sistemas de renderizado es fundamental adquirir el conocimiento de asignaturas como *"Visualización Grafica Interactiva"* el cual está enfocado hacia OpenGL 1.1 (Legacy Mode), sin esos fundamentos es considerablemente difícil poder trabajar con el pipeline programable (4.5).

Las ventajas que ofrece trabajar así, es que la cantidad de trabajo grafico que puede realizar la GPU se ve multiplicada en ordenes de magnitud muy altas. Ya que el mayor cuello de botella es prácticamente la transferencia de las ordenes de dibujo, y con el pipeline moderno se pueden transmitir varias órdenes de golpe en los denominados "Rendering Batchs"

5.2 Sistema de eventos

El sistema de eventos es posiblemente la parte más importante de la herramienta, ya que permite gestionar comportamientos sofisticados de una manera muy ordenada. Es una forma de programar de manera visual y a muy alto nivel, el código generado al utilizar el sistema de eventos esta optimizado para funcionar demandando una cantidad de recursos muy baja. Esto es así porque parte del código utilizado en el editor también se incluye en la maquina **(Ver Sección 6 Arquitectura**).

La simplicidad con la que se puede usar el sistema de eventos es posiblemente uno de los puntos fuertes de la aplicación.

5.3 Integración con Lua

Esta característica es otra de las que hacen la herramienta realmente útil. El uso de un lenguaje de programación de scripting nos permite poder cambiar comportamientos complejos sin tener que recompilar. El api expuesto actualmente permite interactuar con todos los eventos directamente.
Para controlar grandes flujos de comportamiento desde LUA se ha introducido el concepto de "Process", un process es un bloque de código que respeta una interfaz particular **[Ver Figura 12]**

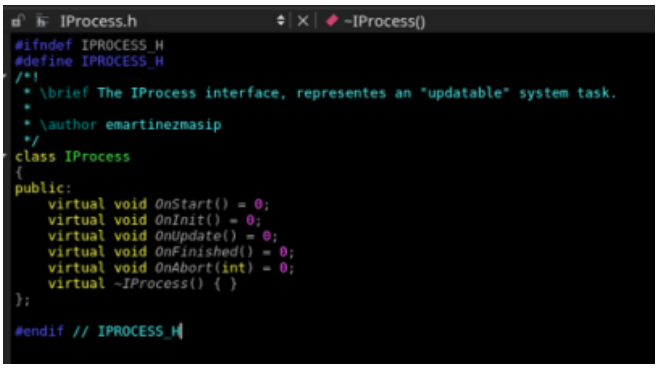

Figura 12 Interfaz IProcess

Esto permite añadir varios "Process" escritos en LUA que gobiernen el juego paralelamente a los eventos. La existencia de un lenguaje desde el que se pueda realizar TODO, elimina todas las restricciones de trabajar con la herramienta ya que, si algo no existe, simplemente, lo programas tú.
El intérprete de Lua utilizado **[15]** al ser un compilador Just in time reduce mucho el overhead de tener un lenguaje de Scripting. En caso de embedder algún otro lenguaje solo haría falta cargar el código del lenguaje en cuestión respetando la interfaz IProcess y todo continuaría funcionando correctamente.

5.4 Escena resultante

Si bien aún faltan algunas piezas para poder realizar cualquier cosa fácilmente. Ya se pueden realizar juegos "casi" completos totalmente funcionales desde la misma. En el siguiente enlace -> https://www.youtube.com/watch?v=oLJCWv-1hmA se puede ver un video de la creación de una secuencia muy simple utilizando la herramienta. En él **Anexo A1** se pueden ver enlaces a diferentes escenas creadas con la herramienta.

6 Arquitectura

Para desarrollar la aplicación se ha optado por una arquitectura basada en capas, donde quede clara una división entre 2 API's distintas. Por un lado, tenemos la API para ejecutar un juego, la cual contiene todo lo necesario para poder ejecutar los juegos de forma standalone mientras que por el otro lado encontramos una API complementaria destinada únicamente a las tareas de edición Figura 13. Encima de estas capas se ha exportado 2 Interfaces de Scripting totalmente aisladas, para poder extender por un lado los juegos y por otro el editor con la utilización del lenguaje LUA, para evitar acoplamiento entre estas, la única vía posible de comunicación entre estas 2 interfaces es mediante la capa más baja, que se encarga de cargar los distintos servicios.

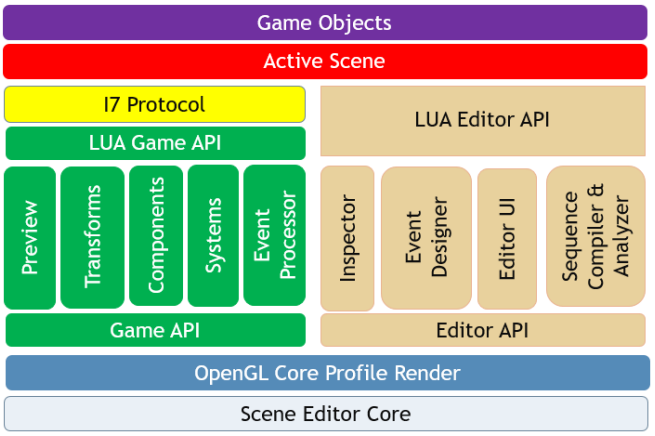

Figure 13 Arquitectura

Gracias al desacoplamiento, el cliente de juegos consiste en ejecutar la misma aplicación desactivando todos los módulos del lado "API de edición".

7 Conclusiones

Es un proyecto considerablemente complejo y ambicioso, las mejoras que ofrece dentro de la Empresa en la cual se usara son considerables, ya que el proceso de desarrollar la secuenciación de una escena es algo que en la forma de trabajar actual se necesitaban semanas además de ser muy propensa a errores. Mientras que con la herramienta ese tiempo queda reducido dramáticamente a "días" con muchísimos menos errores ya que su diseñado basado en eventos y componentes permite que el código sea reutilizable, lo cual hace que una vez testeado al ser idéntico siempre no hay que preocuparse de los errores, pues el procesamiento de la secuencia lo realiza el propio sistema. Durante el desarrollo del proyecto he aprendido además muchas cosas en bastante profundidad (OpenGL Moderno, GLSL, LUA, …) además de haber mejorado considerablemente en el manejo de otras tecnológicas que ya conocía previamente (C++, Qt, …).

8 Líneas futuras

El proyecto en sí tiene muchas líneas futuras que se pueden desarrollar, sin contar los puntos que queda por desarrollar algunas posibles extensiones son:

- Simplificar el diseño de "animaciones" complejas a un simple DSL.
- Simular el juego a toda velocidad (para detectar errores).
- Generar un Previsualizador para más plataformas (Android, iOS, navegador web…)
- Extender las opciones del sistema de eventos para poder ser usado como un lenguaje de programación más completo
- Edición de eventos en caliente (sin tener que reconstruir).

- Y un largo etc.… de posibles mejoras.

9 Agradecimientos

En primer lugar, agradecer a Ramon Baldrich Caselles por tutorizar este proyecto, sin sus consejos y recomendaciones este documento no existiría como tal.

También agradecer al equipo de Interseven Gaming Team SL por facilitar la posibilidad de dedicar el tiempo necesario para el desarrollo de este proyecto.

Por último, agradecer tanto a los compañeros que me han acompañado durante estos años que finalizan con este trabajo, tanto dentro como fuera de la universidad.

Bibliografía

[1] Lua (2017). The Lua Programming Language [online] Available at: https://www.lua.org/ [Accessed 27 Jun.2017]

[2] Python (2017). Python.org [online] Available at: https://www.python.org/ [Accessed 27 Jun.2017]

[3] Squirrel (2017). Squirrel - the programming language [online] Available at: http://www.squirrel-lang.org/ [Accessed 27 Jun.2017]

[4] Mono (2017). Embedding mono [online] Available at: http://www.mono-project.com/docs/advanced/embedding/ [Accessed 27 Jun.2017]

[5] Qt (2017). Qt Documentation [online] Available at: http://doc.qt.io/ [Accessed 27 Jun.2017]

[6] Unity. (2017). Unity - Game Engine. [online] Available at: https://unity3d.com/es/ [Accessed 27 Jun.2017]

[7] Sony ATF. (2017). Sony ATF [online] Available at: https://github.com/SonyWWS/ATF/wiki [Accessed 27 Jun.2017]

[8] Scrum (2017). Scrum [online] Available at: https://proyectosagiles.org/que-es-scrum/ [Accessed 27 Jun.2017]

[9] Gregory, J. (2009). Game engine architecture. CRC Press

[10] McShaffry, M. (2014). Game coding complete. Nelson Education.

[11] Nystrom, R. (2014). Game programming patterns. Genever Benning.Free online at: http://gameprogrammingpatterns.com/contents.html

[12] Meyers, S. (2014). Effective modern C++: 42 specific ways to improve your use of C++ 11 and C++ 14. " O'Reilly Media, Inc."

[13] Grafos acíclicos dirigidos (2017). Grafos acíclicos dirigidos o DAG [online] Available at: https://es.wikipedia.org/wiki/Grafo_ac%C3%ADclico_dirigido [Accessed 27 Jun.2017]

[14] Transformation Matrix (2017). Transformation Matrix [online] Available at: https://en.wikipedia.org/wiki/Transformation_matrix [Accessed 27 Jun.2017]

[15] LuaJIT (2017). LuaJIT [online] Available at: http://luajit.org/ [Accessed 27 Jun.2017]

- *E-mail de contacto: Enoc.Martinez@e-campus.uab.cat*
- *Mención realizada: Computación*
- *Trabajo tutorizado per: Ramon Baldrich Caselles (Computación)*
- *Curso 2016/17*

ANEXO

A1. VIDEOS

- **Escena "**Lamina**", proceso de** desarrollo, versión del editor mayo 2017: https://youtu.be/oLJCWv-1hmA
- **Escena "Lamina", avanzada, versión** del editor junio 2017: https://youtu.be/pJup40r1iQ4
- **Escena "**Rio grande**", muestra** del resultado, versión editor junio 2017: https://youtu.be/Oi_85DDl9sc
- **Ejemplo componente "Rodillos",** versión editor junio 2017: https://youtu.be/sz19EO0LT1c
- Escena RioGrande presentación (énfasis en activación de eventos) versión editor junio 2017: https://youtu.be/KYOv54bYVcc
- Escena con rodillos presentación (énfasis en el rebote, y el control desde lua) versión editor junio 2017: https://youtu.be/HvV3ASs2XoQ
- Escena con objetos dinámicos presentación: https://youtu.be/xXVdWm7ZLC4

A2. TABLA COMPONENTES

Objeto	Comportamiento
Scene Launcher	Se ejecuta al iniciar una escena (Activador inicial)
Frame Ticker	Cada frame activa todos sus elementos.
Hidder	Los objetos que se conectan a este componente son deshabilitados de forma síncrona.
KeyTracker	Lanza un evento por cada tecla registrada en el componente, permite conectar teclas a lanzadores.
FanOut	Cuando recibe la señal de activación, activa todas sus salidas de forma síncrona.
NextFrameFanOut	Cuando recibe la señal de activación, encola todas las salidas para ser ejecutadas en el siguiente frame.
Fan In	Cualquier señal de activación, dispara la salida.
Barrier	Espera a recibir todas las señales de activación, antes de activar la salida (Asíncrono).
Delayer	Una vez activado espera N frames (o segundos) antes de activar la salida (Asíncrono)
Repeater	Repite la tarea una vez por cada vez que se active. Este objeto es síncrono pudiendo funcionar como un bucle “for”.
Counter	Cada vez que se activa se incrementa su valor. Tiene 2 conectores especiales que al activarse llevan el contador hasta el estado 0, ejecutando la salida N veces de forma síncrona o asíncrona según el conector.
SeqPropagator	Cuando se activa ejecuta todas sus salidas de forma en el orden en el que se encuentran de forma síncrona.
StateSwitch	Permite definir estados y activarlos, activar un estado determinado activa todas sus salidas (síncrono)
RandomSelector	Al activarse se activa una salida aleatoria (síncrono)
ReelStateHandler	Según la entrada que se active se cambia el estado del rodillo (síncrono)
Rotator	Cada vez que se activa se rotan los objetos conectados (síncrono).
Translator	Cada vez que se activa se mueven los objetos conectados (síncrono).
Scalator	Cada vez que se activa se escalan los objetos conectados (síncrono).
InterpolatorRotator	Cuando se activa se van rotando los elementos durante N frames de forma asíncrona. Al finalizar activa los otros interpoladores conectados de forma asíncrona.
InterpolatorTranslator	Cuando se activa se van desplazando los elementos durante N frames de forma asíncrona. Al finalizar activan los otros interpoladores conectados de forma asíncrona
InterpolatorScalator	Cuando se activa se van escalando los elementos durante N frames de forma asíncrona. Al finalizar activan los otros interpoladores conectados de forma asíncrona
En expansión, a medida que se use mas la herramienta.	

A3. PLANIFICACIÓN

ID	Tarea	Descripción	Objetivo	Duración	Estado
1	Preparación	Definir los objetivos	Tener una planificación precisa de lo que se pretende realizar	03/02 a 20/02	
		Establecer tecnologías a utilizar			
		Formación en las tecnologías necesarias			
		Realizar la planificación			
2	Aplicación Base	Diseñar GUI inicial	Tener una aplicación esqueleto sobre la que se construirá, la aplicación esqueleto debe contener contextos OpenGL y objetos visuales para representar los datos.	20/02 a 03/03	
		Diseñar sistema de Objetos			
		Implementar sistema de renderizado inicial			
		Implementar sistema de objetos inicial			
		Implementar sistema de propiedades inicial			
3	Texturas y animación	Diseñar sistema de animación inicial	Debe ser capaz de ejecutar las animaciones de Sprites como si fueran un "Video", debe ser compatible con texturas simples y atlas de texturas.	03/03 a 17/03	
		Implementar carga de texturas			
		Implementar Previsualizador de texturas y animaciones			
		Implementar sistema de animaciones inicial			
		Verificar correcto funcionamiento			
4	Eventos	Diseñar sistema de eventos	Se tienen que poder conectar unos Objetos a Otros para que reaccionen entre ellos de forma simple, acciones como al finalizar una animación espera una tecla y arranca otra animación, al pulsar una tecla para las animaciones, etc...	17/03 a 05/04	
		Implementar sistema de eventos			
		Extender sistema de objetos para soportar eventos			
		Extender GUI con el diseñador de eventos			
		Verificar correcto funcionamiento de los eventos			
5	Previsualizador inicial	Diseño del Previsualizador de escenas inicial	Poderse ver una previsualización de la escena que se ha secuenciado por el momento y poder interactuar/hacer reaccionar los eventos	05/04 a 21/04	
		Implementación del Previsualizador			
		Verificar correcto funcionamiento			
6	Transformaciones	Diseñar sistema de transformaciones	Se deben poder aplicar transformaciones geométricas (Escalados/Rotaciones/Traslaciones) de una forma simple, además en caso de que algo este rotado tiene que ser	21/04 a 28/04	
		Implementar el sistema de transformaciones			
		Implementar "hints" visuales para envolventes y rotaciones.			
7	Animaciones con transforma	Extender animaciones con transformaciones	Las animaciones ahora incluirán transformaciones poder realizar desplazamientos por los puntos de las curvas o hacer otros efectos visuales (zooms, loopings,...)	28/04 a 5/05	
		Añadir soporte para curvas Spline			
		Añadir soporte para curvas Bezier			
8	Sistema de Componentes	Diseñar el sistema de componentes	Deben poder existir una serie de componentes externos (definidos mediante JSON o similar) que se puedan cargar en el juego y añadir a la escena, estos componentes serán en si "escenas guardadas"	5/05 a 19/05	
		Implementar el sistema de componentes			
		Crear componente Rodillo			
		Crear componente Visor (Visualiza texto)			
		Verificar el correcto funcionamiento de los componentes			
		Extender GUI para soportar componentes			
9	Construcción de la escena	Diseñar el sistema de compilación de escenas	Generar los archivos necesarios para poder cargar el visualizador externo y ejecutar la escena.	19/05 a 02/06	
		Externalizar el visualizador.			
		Verificar que la escena funciona			
10	Testeo aplicación completa	Testear la aplicación completa	Asegurarse de que todo ha salido bien de forma mas exhaustiva	02/06 a 11/06	
11	Ajuste GUI Detalle	Reajustes visuales al aspecto de la aplicación.	Tener una interfaz ágil y cómoda de utilizar.	02/06 a 26/06	
12	Documentación final	Redacción informe final	Entregables, Documentación.	02/06 a 27/06	
		Preparación presentación			
13	Defensa	Defensa del TFG		30-jun	