HERRAMIENTA LIBRE MULTIPLATAFORMA PARA FACILITAR EN LA ETAPA DE CODIFICACIÓN, LA CONSTRUCCIÓN DE ESCENARIOS PARA VIDEOJUEGOS 2D GENERANDO CÓDIGO PARA UN FRAMEWORK ESPECÍFICO

ASTRID VANESSA SERNA SÁNCHEZ
0956222
astridserna17@gmail.com

Facultad de Ingeniería
Escuela de Ingeniería de Sistemas y Computación
Programa Académico de Ingeniería de Sistemas
Tuluá-Valle
2014

HERRAMIENTA LIBRE MULTIPLATAFORMA PARA FACILITAR EN LA ETAPA DE CODIFICACIÓN, LA CONSTRUCCIÓN DE ESCENARIOS PARA VIDEOJUEGOS 2D GENERANDO CÓDIGO PARA UN FRAMEWORK ESPECÍFICO

ASTRID VANESSA SERNA SÁNCHEZ
0956222
Astridserna17@gmail.com

Trabajo de grado presentado como requisito para obtener el título de
Ingeniero de Sistemas

Director:
Pavel Franco Marín, Ing

Facultad de Ingeniería
Escuela de Ingeniería de Sistemas y Computación
Programa Académico de Ingeniería de Sistemas
Tuluá-Valle
2014

Nota de aceptación:

Firma del presidente del jurado

Firma del jurado

Firma del jurado

DEDICATORIA

Gracias a esas personas importantes en mi vida, que siempre estuvieron listas para brindarme toda su ayuda. Con todo mi cariño este trabajo de grado se las dedico a:

Mi novio Carlos

Papá Marco

Mamá Yolanda

Mis hermanas Tatiana y Maryi

Abuelita Elcy

AGRADECIMIENTOS

Agradezco a Dios, mis padres y hermanas por acompañarme en todo momento.

A mi novio Carlos por su gran ayuda, motivación y soportarme las rabietas y enojos durante todo el camino.

A mi amiga Jessica por la comprensión, apoyo y por todas las tardes de compañía.

Agradezco a mi Director por todos los consejos y guías que me permitieron avanzar y culminar este trabajo de grado.

ÍNDICE GENERAL

Pag.

ÍNDICE DE TABLAS

Pag.

ÍNDICE DE FIGURAS

Pag.

RESUMEN

El desarrollo de videojuegos es la actividad por la cual se diseña y crea un videojuego, desde el concepto inicial hasta el videojuego en su versión final. Ésta es una actividad multidisciplinaria, que involucra profesionales de la informática, el diseño, el sonido, la actuación, entre otros. El desarrollo de un videojuego generalmente sigue el proceso desde la concepción de la idea, diseño, planificación, pruebas, producción y mantenimiento.

Con base en lo planteado anteriormente, en este trabajo se trata el problema de la implementación o codificación de escenarios para videojuegos 2D que se hace lenta, ya que se atribuye este problema, en muchas ocasiones, a la falta de herramientas que aceleren los procesos de desarrollo, pues las que existen, limitan a los desarrolladores al uso de ciertos sistemas operativos o al pago de costosas licencias.

Así, con esta propuesta de trabajo de grado se busca describir los conceptos importantes para comprender la problemática, definir una solución a la problemática indicada, brindar la posibilidad de incrementar la productividad al momento de desarrollar proyectos de videojuegos y definir las actividades a realizar para culminar exitosamente el trabajo.

ABSTRACT

The videogame development is the activity for which it is designed and created a video game, from the initial concept to the videogame its final version. This is a multidisciplinary activity, involving computer professionals, design, sound, acting, among others. The development of a video game generally follows the process from idea conception, design, planning, testing, production and maintenance.

Based on the previously proposed, this paper the problem of implementation or coding scenarios for 2D videogames is slowed it, and attributed this problem, in many cases to the lack of tools to accelerate processes the development, for which limit developers to using certain operating systems or to pay expensive licenses.

So, with this grade work proposal seeks to describe the important concepts to understand the problem, identify a solution to the problem indicated, provide the ability to increase productivity when developing videogame projects and define the activities to be performed to complete work successfully.

2. PLANTEAMIENTO DEL PROBLEMA

2.1 DESCRIPCIÓN DE LA SITUACIÓN PROBLEMÁTICA

Actualmente las herramientas para el desarrollo de videojuegos 2D, presentan diversos inconvenientes que van desde lo costoso de su adquisición hasta estar limitados a funcionar en un único sistema operativo, dónde este último no da la posibilidad de incluir una plataforma de desarrollo opere para Android, IOS, Blackberry, Linux, Windows, Bada, entre otras.

Lo anterior, sumado a la complejidad de la utilización de las herramientas, ha generado que desarrolladores o participantes de un proyecto, desistan de utilizarlas por lo que aparecen problemas mayores relacionados con tener que codificar manualmente [1], y con ello, la dificultad de editar mediante texto algo que está destinado a ser visualizado, además de que las iteraciones de diseño y pruebas son más extensas generando "pérdida" de tiempo, creando así una barrera de comunicación entre los participantes del proyecto, especialmente con diseñadores no técnicos y los desarrolladores.

Entendiendo que el proceso de creación de escenarios consta de diferentes etapas, entre las cuales encontramos:

- Análisis. En esta etapa es necesario definir los aspectos fundamentales que conformarán el videojuego y se deciden qué elementos harán parte de la escena.

- Diseño. En esta fase se detallan todos los elementos que compondrán el juego, dando una idea clara a todos los miembros del grupo desarrollador acerca de cómo son. Aquí los artistas crean bosquejos donde se plasman las ideas propuestas, partiendo de conocimientos técnicos previos, los artistas digitales crean estos escenarios.

- Planificación. En esta fase se identifican las tareas necesarias para la ejecución del videojuego y se reparten entre los distintos componentes del equipo desarrollador y también se fijan plazos para la ejecución de dichas tareas

- Implementación. En esta fase se llevan a cabo todas las tareas especificadas en la fase de planificación, teniendo como guía fundamental el documento de diseño. Esto incluye entre otras cosas la codificación del programa, la creación de los elementos que componen el escenario, ubicación, tamaño, filtros, entre otros.

- Pruebas. En esta fase los videojuegos deben pasar por una etapa donde se corrigen los errores inherentes al proceso de programación y a diferencia de aquellos, los videojuegos requieren un refinamiento de su característica fundamental, la de producir diversión de manera interactiva (jugabilidad).

Si se realiza un adecuado proceso de ingeniería para las primeras etapas, se encuentra que la etapa de producción o codificación es considerada como la más crucial [2], aquí es donde se integran los trabajos de diferentes participantes como los artistas gráficos, sonido y los programadores. Dentro de las actividades que allí se encuentran, hay una que resalta porque de esta depende el resultado final y es la construcción de escenarios, específicamente para

videojuegos 2D, donde el trabajo de los demás grupos que participan se une y gran parte de la responsabilidad recae sobre los programadores.

Es necesario entonces, crear una solución que ayude a los desarrolladores a construir adecuada y correctamente escenarios para videojuegos 2D en el marco de un proyecto determinado, específicamente en la etapa de codificación, pues así, los participantes del proyecto aprenden a reconocer las ventajas de una aplicación útil y asequible.

2.2 FORMULACIÓN DEL PROBLEMA

¿Cómo brindar a los desarrolladores de escenarios de videojuegos 2D, la posibilidad de implementarlos en múltiples plataformas con un framework específico, de manera sencilla, sin que ello implique costos de licenciamiento?

3. JUSTIFICACIÓN

Actualmente los videojuegos se han convertido en un gran negocio. Se está hablando de una industria de miles de millones de dólares que ha superado muchas otras industrias en un período muy corto de tiempo [3]. Para la mayoría de las personas los videojuegos son un mercado aislado, la realidad es muy diferente gracias a los avances tecnológicos de hoy día, en especial con los Smartphone que son multipropósito, permite abrir más posibilidades para un potencial de usuarios amantes a los videojuegos y desarrolladores de éstos [4].

Lo anterior también deriva en que haya un mayor interés de negocio en la producción de videojuegos [5]. Por su creciente demanda, vale la pena pensar en crear herramientas que vayan a este ritmo. Partiendo de que consta de diversas etapas o fases, la creación de escenarios puede ser una de las actividades que más tiempo consume durante la etapa de codificación, por lo que la construcción de una herramienta para el diseño de escenarios 2D puede ser demasiado útil en la actualidad.

Dicho lo anterior, en esta propuesta de trabajo de grado se planea desarrollar una herramienta que se apoyará en un framework especializado en el desarrollo de videojuegos, que soporte las principales plataformas del mercado [6], que no genere restricciones por licenciamientos en código abierto y por último que permita agilizar el proceso de creación de escenarios durante el desarrollo del videojuego.

Finalmente el proyecto permitirá a los desarrolladores de videojuegos el diseño y finalización de la etapa de codificación ofreciendo diversas ventajas entre las cuales podemos resaltar:

- WYSIWYG “What You See Is What You Get”, es mucho más fácil diseñar cuando se puede ver lo que está editando.

- Iteración de diseño más rápido y eficiente, es crucial poder moverse fácilmente entre el diseño y las fases de prueba.

- Hace la fase de diseño menos técnica, facilitando la comunicación entre los artistas y los programadores eliminando posibles barreras de comunicación.

Por otra parte, teniendo en cuenta los editores que se nombrarán en el Marco de Antecedentes, las ventajas que ofrecería el editor propuesto son: a) estará orientado a ser multiplataforma, es decir funcionará sobre los principales sistemas operativos (Windows, Mac y Linux), b) será gratuito, lo que no generará restricciones por licenciamientos, y c) al no poseer un motor de juego propio, teniendo en cuenta que se desarrollará específicamente para Cocos2D-HTML5, quedará abierta la posibilidad de hacer desarrollos futuros e integrar diferentes motores y así el usuario se sentirá a gusto con las características que ofrece. A continuación se expone un resumen de las principales características de algunos editores existentes (ver Tabla 1).

Editor	Plataformas soportadas	Licencias	Soporte	Motor de Juego	Lenguaje de desarrollo
CocosBuilder	MacOSX	Gratis (OpenSource)	Foro.	Cocos2d Cocos2d-html5 Cocos2d-x	Javascript
Unity	Windows MacOSX	Gratuita limitada (free) Comercial (Pro - 1500 USD/persona/año)	Foro. Sitio dedicado a preguntas. Sitio dedicado a solicitud de características. Documentación. Wiki. Soporte directo	Propio	UnityScript
GameSalad	Windows MacOSX	Gratuita limitada Educacional limitada Comercial (Pro - 299 USD/persona/año)	Foro. Libros. Soporte directo.	Propio	Ninguno
RPGMaker	Windows	Prueba (30 días) Comercial (Pro - 69.99 USD	Foro. Documentación. Soporte directo.	Propio	RGSS
GameMaker	Windows	Gratuita limitada Educacional limitada Estandar limitada(49.99 USD) Profesional limitada(99.99 USD) Master (499.99 USD)	Foro. Documentación. Wiki. Soporte directo.	Propio	GML
LevelHelper	MacOSX	Prueba (7 días) Comercial (24.99 USD - Requiere software adicional)	Foro. Documentación. Soporte directo limitado.	Cocos2d-x Cocos2d Cocos2d-html5 Gideros-Mobile Corona-Labs Lanica	Ninguno

Tabla 1. Cuadro comparativo de las principales características de algunos editores existentes. Fuente: Creación del autor.

En consecuencia con lo anterior, el editor propuesto soportará las plataformas Windows, Mac y Linux, será gratuito, contará con toda la documentación técnica y funcional necesaria, quedará abierta la posibilidad de hacer desarrollos posteriores de modo que se pueda exportar el código a motores diferentes a Cocos2D-HTML5 que será para el motor para el se trabajará y por ende, se trabajará bajo los lenguajes C++ y JavaScript.

4. OBJETIVOS

4.1 OBJETIVO GENERAL

Desarrollar una herramienta libre multiplataforma, que facilite la codificación de escenarios para videojuegos 2D y permita generar código para un framework específico.

4.2 OBJETIVOS ESPECÍFICOS

- Proporcionar una interfaz gráfica de usuario usable para la edición de escenarios de videojuegos 2D.

Resultados obtenidos:

1. Los elementos de usabilidad, previa investigación y análisis, con los que el usuario podrá interactuar en el proceso de edición de escenarios.

2. Integración en la herramienta de los elementos de usabilidad con los que el usuario interactuará, permitiendo la edición de escenarios de forma rápida, sencilla y efectiva.

- Analizar sobre los frameworks existentes para determinar cuál será usado, buscando que ofrezca las mejores prestaciones para el desarrollo de videojuegos 2D.

Resultados obtenidos:

1. Cuadro comparativo de los principales frameworks identificando en ellos características, ventajas y desventajas.

2. Framework seleccionado identificando el porqué de dicha selección y profundizando en sus características.

- Realizar un mecanismo de generación de código compatible con el framework seleccionado, partiendo del diseño creado mediante la herramienta.

Resultados obtenidos:

1. Esquema que garantizará el almacenamiento de la información correspondiente al escenario creado por el usuario.

2. Esquema que permitirá, a partir de lo almacenado previamente, crear el código fuente para el framework seleccionado.

- Generar múltiples ejecutables de la herramienta que funcionen sobres los principales sistemas operativos.

Resultados obtenidos:

1. Listado de los principales sistemas operativos para los que se generará los ejecutables de la herramienta.

2. Mecanismo que permitirá generar diversos ejecutables para sistemas operativos específicos.

5. ALCANCE

Debido a la escasez de herramientas especializadas en la creación de escenarios para el desarrollo de videojuegos 2D, los personas enfrentadas a este tipo actividades que son repetitivas y complicadas sino poseen suficiente experiencia, lo que conlleva a pérdida de tiempo, falta de calidad, reproceso y/o el no cumplimiento de las fechas pactadas en el proyecto.

En consecuencia, se propone desarrollar una herramienta que haga más fácil, entendible y eficiente la actividad descrita anteriormente a través de la visualización de lo que se está haciendo, permitiendo agilizar el proceso de construcción de escenarios y pruebas unitarias.

Las características con las que contará la aplicación partiendo de lo expuesto anteriormente serán:

- Interfaz gráfica de usuario usable para crear escenarios para videojuegos 2D.
- Edición rápida de escenas mediante una interfaz WYSIWYG (what you see is what you get).
- Proveer un mecanismo para extender la funcionalidad de la herramienta.
- Exportar la escena en archivos XML brindando la posibilidad de usar un cargador personalizado para usar la escena en diferentes frameworks.
- Proveer un cargador de escena para un framework[1] específico.
- Edición de las principales características de los elementos comunes en una escena: Sprites, Layers, Labels.

En este sentido, en principio se definirán unos módulos para la realización de esta propuesta como se describen a continuación:

Módulo myscener o principal**:** Su función principal es la integración de los módulos que articularán la herramienta.

Módulo de GUI (Graphical User Interface): Incorporará los componentes visuales con los que el usuario va a interactuar, actúa como interfaz entre el usuario y la administración de escenarios.

Módulo de administración de escenario**:** Controlará la visualización de los elementos que componen una escena, permitiendo alterar ciertas características predeterminadas como se indica a continuación:

[1]. Este framework se seleccionará a partir de un análisis en el marco de este proyecto

- Nombre de la escena
- Nodo raíz
- Clase básica de la cual se hereda la escena

Además características propias de cada elemento:

- Nombre
- Clase base
- Nodo padre
- Posición
- Tamaño
- Rotación
- Escala de tamaño
- Posición en el eje Z
- Etiqueta
- Propiedades adicionales

Módulo de administración de plugins: Es posible extender la aplicación sin que esto implique reconstruirla totalmente, para lograr dicho fin se hace uso de plugins que extienden ciertas funcionalidades, en este caso, añadir soporte para importar y exportar en diferentes formatos entendibles por diferentes frameworks de videojuegos.

Módulo de persistencia: Permitirá guardar la escena en un formato estándar legible por la aplicación.

Módulo de exportación: Basado en el documento de texto generado durante el proceso de persistencia generará código fuente usable por el framework de desarrollo de videojuegos elegido.

Queda por fuera del alcance todo aquello que no esté plasmado en este apartado.

6. MARCO DE REFERENCIA

6.1 MARCO DE ANTECEDENTES

Muchos desarrolladores de videojuegos optan por crear sus propios editores de escenarios por cada videojuego en el que trabajan, esto con el fin de facilitar el desarrollo del videojuego, por otra parte implica invertir bastante tiempo en la creación de cada editor.

Dada esta situación, se han creado varias herramientas que buscan facilitar la creación de escenarios de videojuegos 2D, sin embargo, no todas ofrecen una solución genérica, obligando a los miembros del proyecto a usar no solo el Editor sino también el SDK, con lo cual nuevos problemas aparecen. Algunas herramientas por nombrar son:

Algunos nombres de editores de escenarios:

6.1.1 Cocosbuilder

Es una herramienta gratuita (publicado bajo la licencia MIT) para desarrollar rápidamente aplicaciones y juegos. CocosBuilder se construye para cocos2d enlazada con Javascript, lo que significa que el código, animaciones e interfaces se ejecutan sin modificar en el iPhone, Android y HTML 5. Si prefiere ir nativa hasta el final, hay lectores disponibles para cocos2d-iphone y cocos2d-x [7].

Permite probar un juego en un dispositivo móvil, de una manera más rápida o más fácil, sólo tiene que instalar CocosPlayer en su dispositivo (o en el simulador) y CocosBuilder impulsará su código a través de WiFi en tan sólo unos segundos. Pulse el botón de publicar y su juego se guardará al instante a una página web HTML5. CocosBuilder tiene un amplio conjunto de funciones, pretende ser un entorno de creación de estilo flash, con la línea de tiempo, los fotogramas clave, los recursos navegador, editor de código, herramienta de publicación de todo lo construido. Soportada por plataformas como Android y Windows.

Esta herramienta permite la colocación gráficamente de sprites, capas y escenas. Es ideal para la rápida y precisa la creación de menús y otras partes de la interfaz, sino que también se puede utilizar para crear niveles, sistemas de partículas o cualquier otro tipo de gráficos. Además se pueden incluir las escenas que construye en CocosBuilder en un proyecto, agregando los archivos CCBReader.

6.1.2 Unity

Unity es un ecosistema de desarrollo de juegos: un potente motor de renderizado totalmente integrado con juego completo de herramientas intuitivas y flujos de trabajo rápidos para crear contenido 3D interactivo; publicación multiplataforma fácil, miles de activos de calidad, listos para usar en la Tienda de Activos y una comunidad de intercambio de conocimientos [8] .

Unity es un motor de desarrollo totalmente integrado que ofrece un sin número de

funcionalidades innovadoras para crear juegos y otros contenidos 3D interactivos. Utilice Unity para combinar su material gráfico y activos en distintas escenas y entornos; añadir físicas; jugar en modo de prueba y editar su juego de manera simultánea; y, cuando esté listo, publicar su juego en las plataformas de su elección, como computadoras de escritorio, la Web, iOS, Android, Wii, PS3 y Xbox 360.

Hay dos licencias principales para desarrolladores: Unity y Unity Pro,11 con la versión Pro que está disponible por un precio ya que la versión Pro no es gratis, a pesar de que originalmente costaba alrededor de 200 USD (ahora 1500 USD por persona y por año, aunque ofrece soporte para nuevas plataformas como iOS y Android). La versión Pro tiene características adicionales, tales como render a textura, determinación de cara oculta, iluminación global y efectos de post-procesamiento. La versión gratuita, por otro lado, muestra una pantalla de bienvenida (en juegos independientes) y una marca de agua (en los juegos web) que no se puede personalizar o desactivar.

6.1.3 GameSalad

Es una herramienta de edición desarrollada por GameSalad, Inc. (anteriormente Juegos Gendai) dirigido principalmente a los no programadores para la composición de los juegos en de la moda drag-and-drop, la utilización de editores visuales y una lógica basada en el comportamiento sistema. Es utilizado por los consumidores y los profesionales creativos, como diseñadores gráficos, animadores y desarrolladores de juegos para la rápida creación de prototipos, y la construcción de auto-publicación multiplataforma juegos y medios interactivos [9].

La aplicación se ejecuta en Mac OS X para la producción de juegos para iPhone, que también se ejecuta en de Windows que está optimizado para la producción de juegos en Android dispositivos aunque los juegos desarrollados en Windows tienen que ser convertidos a archivos Mac OS que se publicará en el App Store, el navegador web contenido basado en HTML 5, y las aplicaciones de Android.

6.1.4 RPGMaker

Es un programa que permite a los usuarios crear sus propios videojuegos de rol. La mayoría de las versiones incluyen un mosaico creado editor de mapas base, (juegos de fichas se llaman chipsets en versiones anteriores a XP), un sencillo lenguaje de scripting para los eventos de secuencias de comandos y un editor de batalla. Todas las versiones incluyen juegos de fichas iniciales prefabricados, caracteres, y eventos que se pueden utilizar en la creación de nuevos juegos. Una de las características de las versiones de PC de programas RPG Maker es que un usuario puede crear nuevos juegos de fichas y personajes, y añadir nuevos gráficos que él / ella quiere [10].

Desarrolladora por ASCII / Enterbrain / Agetec / CoGen Media Co., Ltd. Soporta plataformas como PC-8801, MSX 2, PC-9801, Super Nintendo, Microsoft Windows, Sega Saturn, PlayStation, Game Boy Color, PlayStation 2, Game Boy Advance, Nintendo DS.

6.1.5 GameMaker

Es una herramienta de desarrollo rápido de aplicaciones, basada en un lenguaje de programación interpretado y un kit de desarrollo de software (SDK) para desarrollar videojuegos, creado por el profesor Mark Overmars en el lenguaje de programación Delphi, y orientado a usuarios novatos o con pocas nociones de programación. El programa es gratuito, aunque existe una versión comercial ampliada con características adicionales. Actualmente se encuentra en su versión 8.1. Overmars liberó la primera versión pública el 15 de noviembre de 1999 [11].

6.1.6 LevelHelper

LevelHelper y SpriteHelper es un completo conjunto de herramientas para la creación de juegos 2D con cocos2d y otros motores. Con una interfaz intuitiva y amigable con el foco en la productividad, velocidad y facilidad de uso, LevelHelper y SpriteHelper le ofrece características avanzadas de desarrollo de juegos, tales como: Física avanzada[2], Animaciones avanzadas, Textura, Activos juego creando estructuras avanzadas con / sin juntas y se multiplican tantas veces como sea necesario con un simple arrastrar y soltar o por código, entre muchas más [12].

Es una aplicación de Mac OS creado por Vladu Bogdan, que ayuda a los desarrolladores a escribir los niveles para sus aplicaciones de iOS. Los niveles se escriben con el diseñador de la herramienta LevelHelper. LevelHelper utiliza imágenes regulares que se cargan en el diseñador utilizando la SpriteHelper herramienta. Cuando el usuario ha terminado de diseñar el nivel de él / ella puede exportar el nivel de código Objective C que se puede utilizar en una aplicación IOS.

[2] En este contexto, cuando se hace referencia a Física Avanzada, se habla por ejemplo de la emulación de leyes físicas tales como la gravedad, la velocidad, fuerza, caída libre, entre otros.

6.2 MARCO TEÓRICO

Muchas personas incluso desarrolladores de software no ven más allá de un videojuego como su resultado final, se desconoce que detrás de toda su producción existe un proceso de ingeniería similar al desarrollo de cualquier producto de software; el cual requiere llevar a cabo numerosas tareas, dentro de etapas como el análisis, diseño, codificación, pruebas, instalación y finalmente un mantenimiento.

Dentro de las tareas o actividades de la etapa de codificación, la construcción de escenarios, específicamente para videojuegos 2D, es una de las que depende en gran parte el resultado final de un videojuego. Durante esta actividad los participantes pueden optar por apoyarse en diferentes herramientas para su desarrollo, entre estas tenemos los editores de escenarios, que permiten reducir la complejidad de la implementación así como brindar un mayor dinamismo entre las etapas de diseño del escenario y pruebas.

A continuación se definirán los conceptos principales tratados hasta el momento.

6.2.1 Ingeniería de software

La ingeniería de software es un área de las ciencias de la computación que ofrece los métodos y herramientas para el desarrollo y mantenimiento de un software de calidad, o también es definido como el establecimiento y uso de principios de ingeniería robustos, orientados a obtener software económico que sea fiable y funcione de manera eficiente sobre máquinas reales.

Los videojuegos nacen desde la misma concepción que un software tradicional, emplean la misma base teórica en su desarrollo, sin embargo algunas particularidades lo diferencian del resto de software existente.

6.2.2 Metodología

Son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos software. Debido a que el desarrollo de software no es una tarea fácil, existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo.

6.2.3 Metodología Cascada

La metodología más común en el desarrollo de videojuegos es la de cascada, que consiste en una adaptación del método de proceso de desarrollo de software de gestión empresarial llamado también Cascada o el Ciclo de Vida Clásico o Tradicional de un software. En éste, cada una de las fases de desarrollo del software se suceden una tras otra así: Planeación, Análisis, Diseño, Programación, Pruebas, Producción y Mantenimiento.

Para el desarrollo de un videojuego las fases adaptables del método de cascada son:

Especificación del juego: Se realiza un documento que especifica el juego desde la perspectiva del usuario (jugador). Es el equivalente a la fase de Planeación del ciclo de vida de un software de gestión empresarial.

La Biblia del arte y la historia: Realizado por los productores y los directores artísticos donde especifican las herramientas y los conceptos a usar desde la parte artística. Se define también la historia, el guión y el diseño del juego desde distintos escenarios. Es el equivalente a la fase de Análisis del ciclo de vida de un software de gestión empresarial.

Especificaciones técnicas: Describe las herramientas ingenieriles como UML o diagramas de sistema, e interacción entre tareas y el código (esta es la única fase estrictamente ingenieril). Es el equivalente a la fase de Diseño del ciclo de vida de un software de gestión empresarial.

Construcción: Se inicia la programación del videojuego, la creación de modelos, texturas, niveles, arte conceptual y toda aquella tarea que dependa del diseño. Es el equivalente a la fase de Programación del ciclo de vida de un software de gestión empresarial.

Aseguramiento de calidad: El equipo de QA debe verificar que todo se contraste con lo planteado en los diseños originales de los documentos adecuadamente. Es el equivalente a la fase de Pruebas del ciclo de vida de un software de gestión empresarial.

Pruebas de juego: Sesiones con los directivos y productores donde se les muestran las características del juego y se hace realimentación que debería modificar el proceso para obtener un producto a total satisfacción.

Pruebas alfa: Una vez se determina que el juego está en una fase estable, se libera a un selecto grupo de evaluadores que determinan la calidad del juego, y plantean cambios para mejorar la experiencia del juego en general (se supone que no deben surgir cambios radicales). Pruebas beta: Se libera el juego a una gran audiencia y se determinan aspectos que gustan o disgustan del juego para el mercado en general.

Golden master: Momento en el que se libera el juego para el consumo masivo. Es el equivalente a la fase de Mantenimiento del ciclo de vida de un software de gestión empresarial.

6.2.4 Metodología RUP (Rational Unified Process)

El Proceso Unificado Racional, es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. El RUP no es un sistema con pasos firmemente establecidos, sino que trata de un conjunto de metodologías adaptables al contexto y necesidades de cada organización, donde el software es organizado como una colección de unidades atómicas llamados objetos, constituidos por datos y funciones, que interactúan entre sí.

Existe la metodología Game Unified Process adaptada de la versión ágil de RUP, para videojuegos. Básicamente esta metodología es una combinación de dos metodologías de desarrollo utilizadas comúnmente en el software tradicional. La primera es la utilización del RUP el cual plantea un diseño estricto y una documentación rigurosa de cada paso y funcionalidad a implementar. Y la metodología eXtreme Programming con algunas variaciones para que pueda ser aplicada por personas de otras disciplinas. En este proceso los ciclos cortos ayudaron a mantener la comunicación fluida entre equipos y el componente artístico no se restringe tanto, como cuando se utiliza el RUP, proporcionando así mayor capacidad comunicativa.

6.2.5 Metodología Scrum

Es una metodología ágil iterativa orientada a la comunicación y a los resultados, disciplina y una rápida producción de versiones visibles del juego, de manera que se puedan hacer las evaluaciones cualitativas que exigen una visión e interacción del producto. Para realizar un seguimiento adecuado al producto se plantean algunas pautas: verificación constante del producto, iteraciones de hasta 30 días, estimación de tiempo de ejecución de tareas y equipos capaces de auto-organización.

Scrum plantea que se debe desarrollar entre pocas personas, un equipo de desarrollo de 5 a 8 personas es ideal, y pueden inclusive subdividirse en la medida en que lo vean pertinente, manteniendo un "scrum de scrums". Los equipos pueden dividirse de acuerdo a los enfoques que cada equipo tenga en el videojuego. Para este enfoque, el cliente es el director creativo y diseñador líder del proyecto, puesto que son los encargados de conceptualizar lo que debe ser el videojuego. Además, la realimentación se debe realizar a medida que se vaya desarrollando el videojuego y no a posteriori que pueda no serle de utilidad al estudio. Deben utilizarse unidades de prueba para las distintas funcionalidades del videojuego.

6.2.6 Videojuego

Desde el comienzo se han entendido a los videojuegos como "un juego", pero un videojuego se entiende también como un medio de almacenamiento en el cual se graba, de manera digital y computarizada, un tipo de juego especial que se le ha bautizado con el mismo nombre que el medio en el que se guarda.

Un videojuego 2D es un juego que posee dos dimensiones, alto y ancho. Un concepto bastante sencillo de entender, pero que ha sido el pilar de los videojuegos desde sus orígenes .Básicamente, están constituidos por sprites, o imágenes animadas en frames, de los elementos del juego. En las dos dimensiones tienen cabida todo tipo de géneros, sin excepción alguna, ya que han pasado por las 2D géneros tan dispares como el RPG y la conducción, pasando por los shooter o las recordadas aventuras gráficas.

6.2.7 Escenario

Es una sección o parte de un juego. La mayoría de los juegos son tan grandes que se dividen en niveles, por lo que sólo una parte del juego tiene que cargar al mismo tiempo. Para completar un nivel de juego, un jugador por lo general tiene que cumplir metas específicas o realizar una tarea específica para avanzar al siguiente nivel. En los juegos de puzzle, los niveles pueden ser similares, pero más difícil a medida que progresas en el juego.

En los videojuegos, un nivel es un área concreta dentro del mundo virtual, donde los jugadores deben superar diferentes desafíos o cumplir con ciertos objetivos hasta lograr completarlo y pasar a la siguiente área [14].

6.2.8 Editor de escenario

Un editor de escenarios (también conocido como un editor de mapas, niveles y campañas) es una herramienta de software utilizada para diseñar niveles, mapas, campañas, etc. y los

mundos virtuales en un videojuego, cuyo objetivo es el proceso de creación de un entorno de juego en el que el jugador del juego interactúa con el universo de juego; el diseño de escenarios implica ahora no sólo la colocación de objetos y enemigos, es analizar el comportamiento en respuesta a los acontecimientos en el mundo o las acciones del usuario(s) en el juego. Se trata de analizar cómo los jugadores se comportan en un entorno ficticio y su manipulación con el fin de crear una experiencia. Un individuo involucrado en la creación de niveles de juego es un diseñador de niveles o mapper.

6.2.9 Framework

De acuerdo con "Design Patterns" [15] un framework es "un conjunto de clases cooperantes que componen un diseño reusable para un tipo específico de software". Pero según se presentará más adelante, un framework no es sólo clases, por lo cual se definirá framework de la siguiente forma:

Un framework es un conjunto de elementos cooperantes que componen un diseño reusable para un tipo específico de software.

6.3 MARCO CONCEPTUAL

Multiplataforma: es un atributo conferido a los programas informáticos o los métodos de cálculo y los conceptos que se ejecutan e interoperar en múltiples plataformas informáticas.

Herramienta: Subprograma o módulo encargado de funciones específicas y afines entre sí para realizar una tarea, que proveen un marco de trabajo amigable para el desarrollo de software. Una aplicación o programa puede contar con múltiples herramientas a su disposición.

2D: si tiene dos dimensiones, por ejemplo, ancho y largo, pero no profundidad. Los planos son bidimensionales, y sólo pueden contener cuerpo.

Exportar: transferir datos desde un programa hacia otro.

Implementación: se define como la es la realización de una aplicación, o la ejecución de un plan, idea, modelo científico, diseño, especificación, estándar, algoritmo o política.

App Store: es un servicio para el iPhone, el iPod Touch, el iPad, Mac OS X Snow Leopard y Mac OS X Lion, creado por Apple Inc , que permite a los usuarios buscar y descargar aplicaciones informáticas de iTunes Store o Mac App Store en el caso de Mac OSX, desarrolladas con el iPhone SDK y publicadas por Apple.

Android: es un sistema operativo basado en Linux, diseñado principalmente para móviles con pantalla táctil como teléfonos inteligentes o tabletas inicialmente desarrollados por Android, Inc., que Google respaldó económicamente y más tarde compró en 2005.

Windows: es el nombre de una familia de sistemas operativos desarrollados y vendidos por Microsoft. Microsoft introdujo un entorno operativo denominado Windows el 20 de noviembre de 1985 como un complemento para MS-DOS en respuesta al creciente interés en las interfaces gráficas de usuario (GUI).

Linux: es uno de los términos empleados para referirse a la combinación del núcleo o kernel libre similar a Unix con el sistema GNU. Su desarrollo es uno de los ejemplos más prominentes de software libre; todo su código fuente puede ser utilizado, modificado y redistribuido libremente por cualquiera bajo los términos de la GPL y otra serie de licencias libre.

Mac OS: es el nombre del sistema operativo creado por Apple para su línea de computadoras Macintosh. Es conocido por haber sido el primer sistema dirigido al público en contar con una interfaz gráfica compuesta por la interacción interfaces.

6.4 MARCO CONTEXTUAL

El desarrollo de esta herramienta representa beneficios al proceso de construcción de escenarios para videojuegos 2D específicamente en la etapa de codificación, algunos de los beneficios más notables son:

- Fácil edición de escenarios para videojuegos 2D. El diseño de escenarios es uno de los aspectos más importantes, influyentes y complejos de los videojuegos. Esto es completamente una nueva forma de diseño que está creciendo rápidamente. Debido a que las capacidades de hardware y la tecnología asociada al desarrollo de estos sigue aumentando en complejidad, de igual forma el diseño de escenarios será cada vez más complejo [16].

- Generación de código a partir de los escenarios creados, esto ahorra mucho tiempo para ser invertido en otras actividades.

- Disminución en los errores entre la Iteración de diseño debido a que es crucial poder moverse fácilmente entre el diseño y las fases de prueba.

- La fase de diseño se hace menos técnica, facilitando la comunicación entre los artistas y los programadores eliminando posibles barreras de comunicación.

Además, al tener la posibilidad de utilizar un framework para el desarrollo de videojuegos específicamente 2D, presentará ventajas como:

- Permite verificar la fidelidad[3] de los diferentes escenarios para videojuegos 2D generados mediante la herramienta, al ofrecernos una estructura y una serie de herramientas para la creación de los videojuegos.

- Ofrecen abstracciones de muy alto nivel para tareas comunes de programación.

- El objetivo es ayudar al programador facilitando el prototipado y acelerando los tiempos de desarrollo. Los frameworks son los paradigmas de la reutilización.

Implementación adecuada de las metodologías en la Ingeniería de software, mejorar la comunicación y aprovechar mejor el tiempo y los beneficios se ven proyectados en casi todos los campos, un proyecto que genere un producto de buena calidad implica una mejora en las ventas y/o en su reputación.

[3]. La "fidelidad" de un producto tiene a que ver con su calidad percibida y sus características propias.

7. METODOLOGÍA

Para el cumplimiento de la problemática planteada, los objetivos y el alcance de esta propuesta de trabajo de grado, se recopilará información la cual será extraída de libros, artículos, sitios web y demás elementos bibliográficos; también se contará con el apoyo del director de trabajo de grado, quien aportará información de vital importancia, lo anterior permitirá evaluar todo lo relacionado al tema de videojuegos en 2D y las herramientas utilizadas durante su desarrollo, específicamente el diseño de escenarios y así se pueda entender de manera detallada y hallar una solución adecuada de acuerdo a esta problemática.

Con base a lo descrito anteriormente la metodología propuesta obedece principalmente a 3 fases (Ver Figura 1), necesarias para el entendimiento e identificación de las actividades o elementos indispensables para la solución de la problemática.

Figura 1 Fases de la investigación. Fuente: Creación del autor

Teorización. Esta fase se encargará de estudiar, comprender y analizar los elementos de orden teórico que a criterio del autor y basado en su experiencia, se deberían tener en cuenta para el desarrollo de una herramienta de edición de escenarios para videojuegos 2D.

Adaptación. Esta fase busca ajustar o establecer en la teoría, cuáles son los elementos que están inmersos en el proceso de desarrollo de una herramienta para la edición de escenarios para videojuegos 2D y la forma en cómo se relacionan.
Construcción. Esta fase busca construir o desarrollar una herramienta para la edición de escenarios para videojuegos 2D tomando como material todos y cada uno de los elementos obtenidos en las fases anteriores.

7.1 TIPO DE ESTUDIO

Para el correcto desarrollo de la presente propuesta de trabajo de grado se llevarán a cabo una serie de pasos que, en su correcto cumplimiento, permitirán terminar

satisfactoriamente cada fase del proyecto, por ello se hace necesario desarrollar un tipo de estudio descriptivo con el fin de lograr desarrollar una herramienta de software capaz de ayudar a los diseñadores y programadores, en general participantes en la construcción de escenarios para videojuegos 2D.

Con base en el conocimiento y en los hechos observados, se analizarán las diferentes herramientas para la construcción de escenarios para videojuegos 2D aplicados en la etapa de codificación y las nuevas tecnologías referente a los frameworks para evaluar las falencias y aspectos positivos que poseen y así determinar la mejor forma para la implementación de la herramienta que se plantea hacer en la presente propuesta de trabajo de grado.

8. FASE DE ANALISIS

8.1 ELEMENTOS O PRINCIPIOS DE USABILIDAD

La usabilidad es una característica que está relacionada con la medida de la calidad de los sistemas interactivos usados por usuarios específicos en un contexto de uso, para conseguir objetivos específicos con efectividad, eficiencia y satisfacción, en términos de utilidad, facilidad de uso, facilidad de aprendizaje y apreciación [17].

- Efectividad: la precisión y completitud con la que el usuario alcanza sus objetivos. Relacionados con este concepto está la facilidad de aprendizaje, la tasa de errores cometidos y la facilidad de recordar sus funcionalidades y procedimientos.

- Eficiencia: los recursos empleados relacionados con la precisión y completitud con que el usuario alcanza sus objetivos. Relacionados con este concepto está la facilidad de aprendizaje, la tasa de errores cometidos y la facilidad de la aplicación para ser recordada.

- Satisfacción: la ausencia de incomodidad y una actitud positiva hacia el uso de la aplicación. La aplicación debe ser agradable de usar de forma que el usuario esté satisfecho al utilizarla.

Teniendo en cuenta lo anterior, a continuación se definirá los términos de usabilidad:

- Utilidad: capacidad de la aplicación para ayudar en la realización de tareas. Facilidad de uso: está relacionada con la eficiencia o efectividad con que se realizan las tareas. En una aplicación fácil de usar se realizarán las tareas más rápidamente.

- Facilidad de aprendizaje: es una medida del tiempo requerido para usar la aplicación con cierta eficiencia y llegar a recordar los procedimientos después de no usar la aplicación durante un tiempo determinado.

- Apreciación: es una medida de la percepción, opinión, sentimiento y actitud generada en el usuario por el uso de la aplicación. Es una medida subjetiva pero muy importante.

La Organización Internacional para la Estandarización (ISO) propone dos definiciones del término usabilidad [18]:

> *"La usabilidad se refiere a la capacidad de un software de ser comprendido, aprendido, usado y de ser atractivo para el usuario, en condiciones específicas de uso"* (ISO/IEC 9126).

> *"Usabilidad: es la efectividad, eficiencia y satisfacción con la que un producto permite alcanzar objetivos específicos a usuarios específicos en un contexto de uso específico"* (ISO/IEC 9241-11).

Además de las anteriores interacciones, Donald Norman indica que el diseño de usabilidad de cualquier interfaz gráfica de usuario para los medios digitales necesita fundamentarse en cuatro elementos [19]:

"*1) facilitar la determinación de qué actos son posibles en cada momento (utilizar limitaciones). 2) Hacer que las cosas sean visibles, comprendiendo el modelo conceptual del sistema, los diversos actos posibles y los resultados de esos actos. 3) Hacer que resulte fácil evaluar el estado actual del sistema. 4) Seguir las topografías naturales entre las interacciones y los actos necesarios, entre los actos y el efecto consiguiente, y, entre la información que es visible y el estado del sistema*".

Desde esta perspectiva, centrada en el usuario, atendiendo a los factores que intervienen en la usabilidad y, teniendo en cuenta los elementos en los que ésta se fundamenta, es posible responder a muchos interrogantes y tener una buena interacción con el usuario.

Además de la gran importancia de la usabilidad:

"El gran avance en la tecnología de los ordenadores ha incrementado la potencia de éstos a la vez que ha realzado el ancho de banda de comunicación entre las personas y los ordenadores. Aún así, el impacto de la tecnología no es suficiente para realzar la usabilidad. Los principios para la interacción son independientes de la tecnología y dependen mucho más de un conocimiento más profundo de los elementos humanos de dicha interacción" [20].

8.1.1 Atributos de Usabilidad

La usabilidad es una cualidad demasiado abstracta como para ser medida directamente. Para poder estudiarla se descompone habitualmente en los siguientes cinco atributos básicos de Nielsen [21]:

Facilidad de aprendizaje: Cuán fácil es aprender la funcionalidad básica del sistema, como para ser capaz de realizar correctamente la tarea que desea realizar el usuario. Se

mide normalmente por el tiempo empleado con el sistema hasta ser capaz de realizar ciertas tareas en menos de un tiempo dado (el tiempo empleado habitualmente por los usuarios expertos). Este atributo es muy importante para usuarios noveles.

Eficiencia: El número de transacciones por unidad de tiempo que el usuario puede realizar usando el sistema. Lo que se busca es la máxima velocidad de realización de tareas del usuario. Cuanto mayor es la usabilidad de un sistema, más rápido es el usuario al utilizarlo, y el trabajo se realiza con mayor rapidez. Nótese que eficiencia del software en cuanto su velocidad de proceso no implica necesariamente eficiencia del usuario en el sentido en el que aquí se ha descrito.

Recuerdo en el tiempo: Para usuarios intermitentes (que no utilizan el sistema regularmente) es vital ser capaces de usar el sistema sin tener que aprender cómo funciona partiendo de cero cada vez. Este atributo refleja el recuerdo acerca de cómo funciona el sistema que mantiene el usuario, cuando vuelve a utilizarlo tras un periodo de no utilización.

Tasa de errores: Este atributo contribuye de forma negativa a la usabilidad de un sistema. Se refiere al número de errores cometidos por el usuario mientras realiza una determinada tarea. Un buen nivel de usabilidad implica una tasa de errores baja. Los errores reducen la eficiencia y satisfacción del usuario, y pueden verse como un fracaso en la transmisión al usuario del modo de hacer las cosas con el sistema.

Satisfacción: Éste es el atributo más subjetivo. Muestra la impresión subjetiva que el usuario obtiene del sistema.

Algunos de estos atributos no contribuyen a la usabilidad del sistema en la misma dirección, pudiendo ocurrir que el aumento de uno de ellos tenga como efecto la disminución de otro. Por ejemplo, esto puede ocurrir con la facilidad de aprendizaje y la eficiencia. Es preciso realizar el diseño del sistema cuidadosamente si se desea tanto una alta facilidad de aprendizaje como una alta eficiencia; siendo el uso de aceleradores (combinaciones de teclas que ejecutan operaciones de uso habitual) la solución más común para conjugar ambos atributos de usabilidad.

8.1.2 Heurísticas de Usabilidad

La Heurística método desarrollado por Molich y Nielsen, que consiste en analizar la conformidad de la interfaz con unos principios reconocidos de usabilidad (heurísticos) mediante la inspección de varios evaluadores expertos [22].

Visibilidad del estado del sistema. La aplicación debe mantener siempre informado al usuario del estado del sistema así como de los caminos que este pueda tomar con una

retroalimentación visual apropiada en un tiempo razonable. El sistema ofrecerá al usuario una respuesta que le indique lo que está sucediendo en cada una de las operaciones que realiza. Por ejemplo, en una aplicación que imprime diferentes páginas, mostraría al usuario la actividad que está realizando con la posibilidad de deshacer la operación. En la siguiente ventana, la aplicación informa al usuario del formateado de las páginas y del envío de estas a la impresora con movimiento de dos círculos [23].

Figura 2 Visibilidad del estado del sistema. Fuente: issi 2011

Control y libertad del usuario. La interfaz debe ser diseñada de tal manera que el control de la interacción con el sistema lo tenga el usuario de manera que interactúe directamente con los objetos de la pantalla. De esta forma, este, se sentirá más cómodo y no se sentirá un módulo más de la aplicación. Esto se consigue cuando el usuario manipula los objetos como si fueran objetos físicos. Se le proporcionarán salidas de emergencia sin tener que pasar por un montón de diálogos. Se deberá proporcionar también acciones de deshacer y rehacer sin perder el trabajo realizado hasta el momento.

Figura 3 Funciones deshacer y rehacer. Fuente: issi 2011

Se debe tener en cuenta que existen diferentes tipos de usuarios, con diferentes niveles de conocimientos informáticos y percepción de la aplicación. Los desarrolladores deben crear grupos de usuarios con características comunes y para cada uno de los grupos crear un nivel de uso de la aplicación. Por ejemplo se pueden tener niveles de uso para usuarios noveles, intermedios y avanzados. La presentación de la interfaz será diferente para cada grupo de usuarios.

Consistencia y estándares. Una buena interfaz contribuye al aumento de la productividad si es consistente en todos los diálogos que desarrolla, basándose en el conocimiento que el usuario ha adquirido con otras aplicaciones y en la aplicación propia. Se debe mantener la consistencia en todas las aplicaciones relacionadas. Deberán implementar las mismas reglas de diseño para mantener la consistencia en toda la interacción. El usuario debe ser capaz de saber en cada momento en qué contexto está trabajando, de donde viene y a donde va. Esto se puede realizar con indicadores gráficos como iconos o colores diferentes para cada situación.

Existen dos tipos de estándares: estándares de iure y estándares de facto. Los estándares de iure son aquellos generados por comités con status legal y avalados por gobiernos o instituciones. Requieren una elaboración y proceso complejo. En el mundo informático, entre los comités más importantes dedicados a la realización de normas están:

- ISO (International Organization for Standardization).
- ANSI (American National Standards Institute)
- IEEE (Institute of Electrical and Electronics Engineers)
- IEC (International Electrotechnical Commission)
- W3C (World Wide Web)
- CEN(European Committee for Standardization)

Los estándares de facto son aquellos que nacen de productos de la industria con mucho éxito en el mercado informático o a partir de trabajos de investigadores y que han tenido una gran difusión. La utilización de estándares en el diseño de interfaz gráfica de usuario produce una serie de beneficios, como pueden ser:

- Una terminología común: permitiendo a los diseñadores hablar de los mismos conceptos y poder realizar comparativas sobre ellos.

- Una identidad común: resultando sistemas fáciles de reconocer.

- Reducción de la formación**:** el conocimiento ‘se hereda’ de una sistema a otro con la consiguiente reducción de costes de formación y productividad.

- Mantenimiento y evolución: al tener estructuras y estilos comunes se reducen costes por este concepto.

- Salud y seguridad: los sistemas con controles de estandarización tienen pocos comportamientos no esperados.

En la organización nos debemos asegurar que los estándares son seguidos por los desarrolladores para mantener la consistencia.

Prevención de errores. El mejor tratamiento de los errores es prevenirlos con un buen diseño de los diálogos desde el primer momento en que ocurren, minimizando los riesgos de que puedan ocurrir. Se debe realizar un buen diseño de mensajes de error que den la posibilidad al usuario de retraerse antes de que se realice la acción y se comprometan los datos.

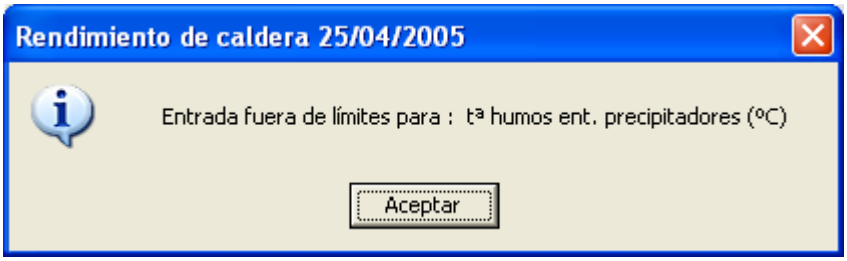

Figura 4 Información de error. Fuente: issi 2011

La ventana del ejemplo anterior está informando al usuario que está introduciendo un valor incorrecto en la celda "tª humos ent. Precipitadotes (ºC)" para que lo corrija. El sistema no realizará los cálculos hasta que los datos de entrada sean correctos.

Correspondencia entre el sistema y el mundo real. El sistema debe hablar el lenguaje de los usuarios, con palabras, frases y conceptos familiares para el usuario, siempre en el contexto de la aplicación. Se debe hacer que la información aparezca en un orden lógico y natural.

El usuario no tiene por qué conocer los términos técnicos utilizados en el mundo informático. La aplicación debe interactuar con el usuario de forma que este perciba las palabras y frases cotidianas de la metáfora de la aplicación. La aplicación debe ser lo más parecida posible al objeto del mundo real que representa.

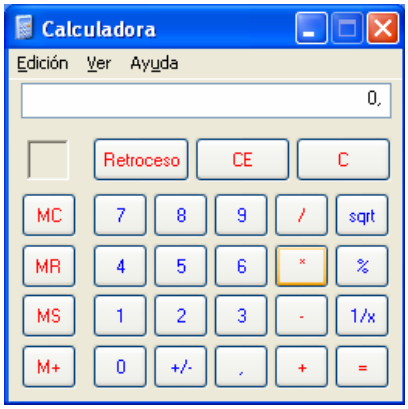

Figura 5 Correspondencia entre el sistema y el mundo real. Fuente: issi 2011

El programa de la calculadora del sistema operativo Windows es un buen ejemplo de lo expuesto anteriormente. Para el usuario, a la vista de la ventana anterior, es inmediata la realización de cálculos como lo haría con una calculadora real.

Reconocer antes que recordar. Reducir la carga de memoria del usuario para reducir la propensión a errores en su interacción con el sistema. El usuario no debería tener que recordar información desde una parte de un diálogo a otro.

Las instrucciones para usar el sistema deben ser visibles o fácilmente accesibles. Se deben establecer unos valores por defecto para la aplicación, con la posibilidad de que el

usuario pueda especificar sus preferencias. También se debe tener la opción de reinicializar los valores por defecto.

Otra manera de reducir la carga cognitiva del usuario es cuando se utiliza la mnemónica para la realización de algunas acciones, como por ejemplo en las aplicaciones de Microsoft siempre se guarda el trabajo con Ctrl+G y siempre se imprime con Ctrl+P. Otra forma de reducir la carga cognitiva del usuario es que la interfaz de usuario sea un metáfora del mundo real. Por ejemplo, una aplicación para control del volumen de sonido imitará los controles de una cadena de música:

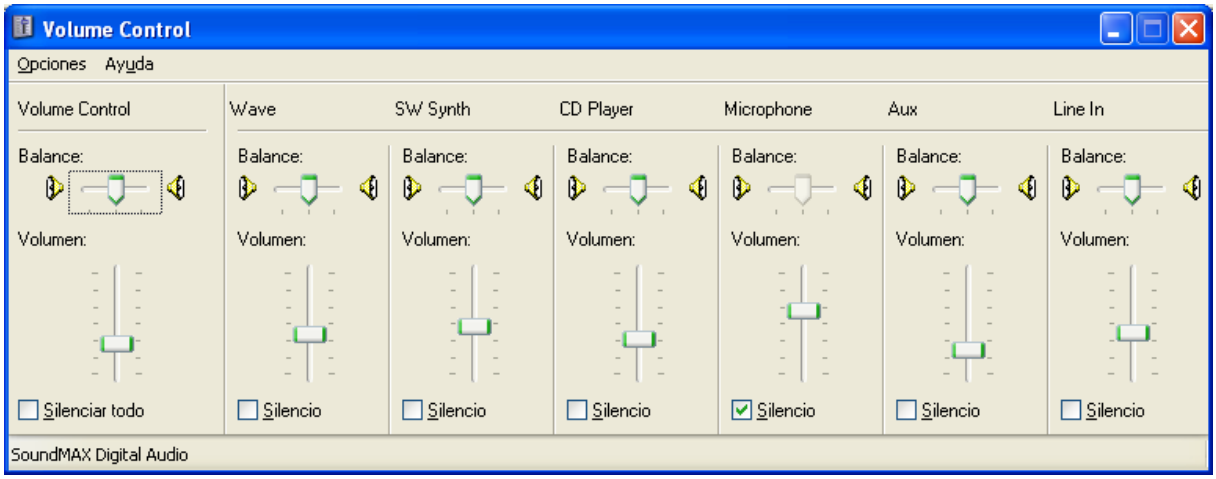

Figura 6 Reconocer antes que recordar. Fuente: issi 2011

Flexibilidad y eficiencia de uso. El sistema se debe diseñar para que lo puedan manejar diferentes tipos de usuarios, en función de su experiencia con la aplicación. De esta manera se aumentará la productividad del usuario y se ganará en usabilidad.

Una innovación muy importante en el desarrollo de interfaz de usuario es proporcionar al sistema técnicas de adaptabilidad. De esta forma la interfaz se adecua de forma automática a las características del usuario. Un sistema adaptativo puede estar basado en la experiencia del usuario con la aplicación o en la observación de las tareas que el usuario realiza repetidamente.

Establecer aceleradores para aumentar la velocidad de la interacción para los usuarios expertos. Esto se puede realizar con pulsaciones de teclas rápidas, iconos.

El usuario deberá poder adaptar la interfaz a su conveniencia, por ejemplo, tendrá la posibilidad de quitar o añadir iconos, menús o barras de iconos. El usuario deberá poder adaptar la interfaz a su conveniencia, por ejemplo, tendrá la posibilidad de quitar o añadir iconos, menús o barras de iconos.

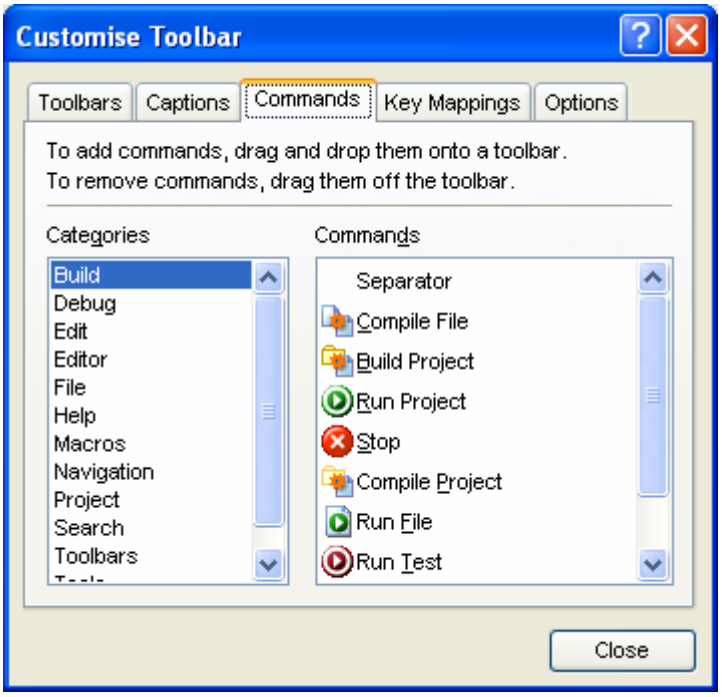

Figura 7 Flexibilidad de uso. Fuente: issi 2011

Estética y diseño minimalista. Los diálogos no deben contener información que sea irrelevante para la tarea que está realizando el usuario. Debe ser una interfaz simple, fácil de aprender y de usar y con fácil acceso a las funcionalidades que ofrece la aplicación. La información extra no necesaria disminuye la visibilidad al usuario causando errores en la interacción y distrayendo al usuario en la realización de la tarea.

Ayudar a los usuarios a reconocer, diagnosticar y recuperación de errores. Los mensajes de error deben estar expresados en lenguaje que el usuario entienda y no con códigos de error, indicando el problema y sugiriendo la solución al problema que causa el error.

El siguiente ejemplo muestra un mensaje de error que no sigue la línea del párrafo anterior. Sólo dice que no es correcto, pero no dice el qué y cómo solucionarlo. Esto es un ejemplo de lo que no se debe hacer.

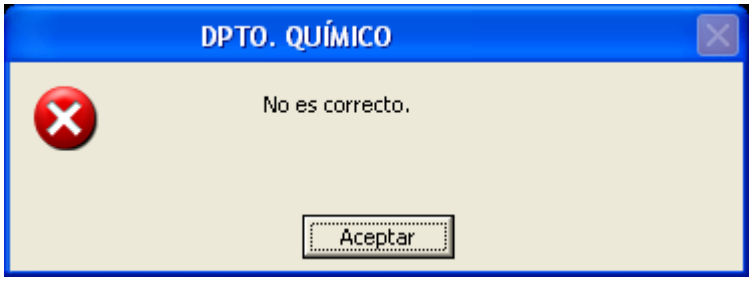

Figura 8 Mensaje de error mal formateado, sin información. Fuente: issi 2011

Ayuda y documentación. El mejor sistema es el que no necesita ningún tipo de documentación, pero de todas formas hay que proporcionar al usuario ayuda y

documentación. Esta debe ser fácil de encontrar y enfocada a la tarea que el usuario realiza. Se deben listar sólo los pasos necesarios para la realización de la tarea.

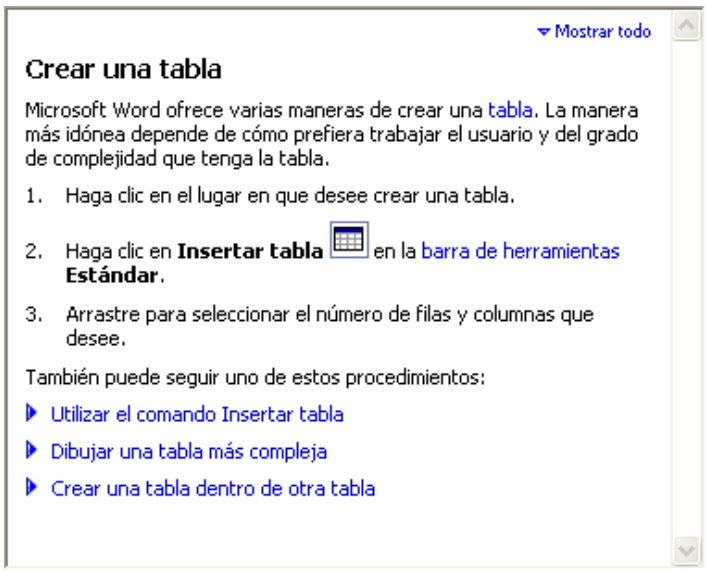

Figura 9 Ayuda con pasos concretos y claros. Fuente: issi 2011

El anterior ejemplo es la ayuda de Microsoft Word para crear una tabla. Como se ve en la imagen, la ayuda está estructurada en pasos muy concretos y claros.

8.2 PRINCIPALES FRAMEWORKS EXISTENTES PARA EL DESARROLLO DE VIDEOJUEGOS 2D

Los motores de juego (Game Engine) son herramientas para facilitar y acelerar la creación de videojuegos. Dichos motores típicamente proveen un framework para el desarrollo de juegos que incluye componentes que se encargan de resolver aspectos tales como rendering de gráficas, audio, animación, inteligencia artificial, física, localización, y habilitación de red, entre los más comunes.

Existe una gran variedad de motores para crear juegos, con distintas capacidades, enfoques y modelos de licenciamiento algunos por nombrar:

8.2.1 Unreal Engine / UDK

El Unreal Engine (actualmente en su 3ra generación) es un motor para juegos creado por Epic Games, y que ha sido utilizado para construir varios de los videojuegos de consola más populares en el mercado, tales como la serie Gears of War. Durante muchos años, el Unreal Engine solo estuvo disponible para estudios de videojuegos profesionales, sin embargo en noviembre del 2009 Epic liberó el Unreal Development Kit (UDK), que es una versión del Unreal Engine disponible al público en general con el que se puede crear

juegos para distintas plataformas incluyendo consolas, PC, Mac, Android, iOS y Playstation Vita [24].

Este motor destaca por sus capacidades avanzadas para el rendering de gráficas, incluyendo técnicas avanzadas como iluminación por pixel, sombras dinámicas y high dynamic range rendering (HDRR). También cuenta con su propio ambiente de edición (Unreal Editor). Los scripts se crean en un lenguaje de programación propietario (UnrealScript) y son conectados entre sí usando el editor visual Unreal Kismet.

Uno de los inconvenientes de UDK es que solo está disponible para Windows.
Para fines educativos o sin lucro, el uso del UDK es gratuito. En el caso de empresas que planean crear juegos comerciales con UDK, requieren firmar un contrato de licencia comercial en el que hacen un pago inicial de 99 dólares y se comprometen a que en caso de que obtengan ganancias superiores a los 50 mil dólares darán una comisión del 25% de sus ingresos a Epic.

8.2.2 Cocos 2D

Cocos2d es un Framework open source para crear juegos 2D. La versión original de cocos2d fue programada en Python, pero existen diversos para otros lenguajes y plataformas. El más conocido de estos es cocos2d-iphone, escrito en Objective-C para crear juegos para iOS y Mac OSX [25]. Entre las ventajas de cocos2d destacan las siguientes:

Fácil de usar. Su API es sencillo e incluye una gran variedad de ejemplos. Provee abstracciones de alto nivel para las tareas más comunes.

Rápido. Cocos2d utiliza las mejores prácticas de OpenGL ES y estructuras de datos optimizadas.

Es software libre. Cocos2d está bajo licencia MIT, una licencia muy flexible que permite utilizarlo tanto para hacer juegos de código abierto como cerrado. Además puedes extenderlo e integrarlo con bibliotecas de terceros.

Comunidad activa. La comunidad de cocos2d es grande y activa, en los foros típicamente puedes obtener respuestas rápidamente.

Cocos2d para iOS utiliza las herramientas y lenguajes de esta plataforma (Objective-C, XCode), lo cual puede ser una ventaja si ya estás familiarizado con ellos o una desventaja si no te son familiares. Otra desventaja de Cocos2d es que a diferencia de las herramientas comerciales como Unity o Unreal, no posee un editor gráfico para animaciones o escenas.

También existen Cocos2d para Android, HTML5 y XNA entre otros y los siguientes son algunos por nombrar:

8.2.3 Cocos 2D-X

Cocos2D-X es la extensión a otras plataformas del Framework cocos2D para iPhone. Mantiene la estructura del API original pero permite utilizar otros lenguajes de programación como C++, Lua, Javascript y C#. Además habilita el desarrollo en sistemas operativos: Windows, Mac y Linux, y amplía el soporte hacia plataformas móviles (iOS, Android, Windows Phone) e inmóviles (win32, Linux, Windows 8, Mac OS X).

8.2.4 Cocos2d-html5

Cocos2d-html5es la versión de JavaScript de Cocos2d-x, diseñada para los navegadores web y en base a la tecnología HTML5. Tizen proporciona mejor marco de aplicación web compatible con HTML5, por lo Cocos2d-html5 se puede hacer para trabajar en él con facilidad. Cocos2d-html5 utiliza objeto Canvas para representar los gráficos [26].

8.2.5 Corona

Es un framework multiplataforma (abarca iOS, Android, Kindle Fire y NOOK) y dispone de librerías para incluir OpenGL, OpenAL, Google Maps, Box2D, Facebook Connect o Game Center en tus desarrollos, gracias a lo cual puedes crear tanto juegos como apps de todo tipo. Está basado en LUA, un lenguaje de script muy ligero y rápido de aprender y sobre el que la mayoría de profesionales coinciden al decir que se tarda menos tiempo y se necesitan menos líneas de código al desarrollar que en Java, Objective-C o C++ (siempre que no sean cosas avanzadas como, por ejemplo, incluir modo multijugador o sincronizar datos “en la nube”).

Sin embargo se debe depender de ellos al compilar, ya que se realiza de manera online y se deben pagar las siguientes tarifas anuales si quieres publicar tu juego: 199$ en Android, 199$ en iOS ó 349$ en iOS, Android, Kindle Fire y NOOK [27].

8.2.6 Libgdx

Es un marco de desarrollo de juegos Java que proporciona una API unificada que funciona en todas las plataformas soportadas.

El marco proporciona un entorno para la creación rápida de prototipos y de iteraciones rápidas. En lugar de implementar para Android / iOS / Javascript después de cada cambio de código, puede ejecutar y depurar su juego en el escritorio, de forma nativa. Escritorio JVM ofrece como código hotswapping reducir sus tiempos de iteración considerablemente [28].

Libgdx trata de no ser el "fin de todo, ser todo" solución. No obliga a un diseño específico en usted. Escoger y elegir las características a continuación.

Una sola API para: Windows, Linux, Max OS X, Android (1.5), iOS (requiere una licencia Xamarin.iOS, evaluación gratuita de 30 días, $ 79 para estudiantes, $ 299 de "Indies"), Applet Java (JVM requiere para ser instalado), Javascript / WebGL (Chrome, Safari, Opera, Firefox, IE a través de Google Chrome Frame).
Algunas ventajas:

- Soporte 2D full (bajo y alto nivel).
- Mucha documentación, tutoriales, ejemplos de código.
- Releases en forma periódica.
- Se puede probar en desktop (antes de subir a mobile).
- Maneja Audio, input (usuario), física, matemática, archivos.
- Soporta 3D
- Posibilidad de tomar un juego hecho en java jar 2d o 3d y adaptarlo a libgdx para que funcione nativo en android,eso hicieron con el juego Droid Invaders 3d.

8.2.7 Marmalade Quick

Es una herramienta multiplataforma que permite a los desarrolladores crear sin compromiso. Poderoso SDK que permite a los desarrolladores llegar a más plataformas y más dispositivos con una sola base de código, ahorrando tiempo de desarrollo y esfuerzo y te deja libre para crear aplicaciones y juegos increíbles.

El SDK Marmalade ofrece el máximo rendimiento. Ya sea que elija para el código nativo (C++), un enfoque híbrido (HTML5 nativo), o si desea ponerse en marcha rápida con Lua (Marmalade Quick), con Marmalade, tendrá la opción de desplegar para telefonía móvil, tabletas y plataformas de escritorio, incluyendo iOS, Android, Windows Phone 8, BlackBerry 10, Windows y Mac, así como Smart TV seleccionado y las plataformas de streaming de televisión también [29].

Tanto si eres un desarrollador a tiempo parcial, una organización de desarrollo más grande o algo intermedio, Marmalade tiene una gran gama de opciones de licencia de valor disponibles con niveles de apoyo y plataforma de acceso diferentes. Echa un vistazo a ellos aquí y registrarse para una conexión 30 días de prueba.

8.2.8 Allegro

Es una biblioteca libre y de código abierto para la programación de videojuegos desarrollada en lenguaje C. Allegro es un acrónimo recursivo de «Allegro Low Level Game Routines» (rutinas de bajo nivel para videojuegos).

La biblioteca cuenta con funciones para gráficos, manipulación de imágenes, texto, sonidos, dispositivos de entrada (teclado, ratón y mandos de juego) y temporizadores, así como rutinas para aritmética de punto fijo y acceso al sistema de archivos. Hasta agosto de 2011, hay 2 versiones de Allegro que cuentan con soporte oficial por parte de los desarrolladores, la versión clásica (Allegro 4) y la nueva versión (Allegro 5). La versión más reciente de Allegro 4 incluye soporte para el manejo de archivos de datos y una

implementación por software de funciones para gráficos en 3D. La versión 5 de Allegro cuenta con una nueva API y cambia la implementación por software de las rutinas gráficas por una implementación basada en OpenGL o Direct3D [30].

Aunque Allegro ofrece una API en lenguaje C, actualmente existen envolventes y bibliotecas adicionales que permiten utilizarlo en otros lenguajes como Python, D, Lua y Pascal.

8.2.9 Construct Classic

Es un creador de juegos gratis para Windows, diseñado para juegos 2D. Es el precursor de la más moderna Construct 2. Construir Classic utiliza un sistema basado en eventos para definir el comportamiento del juego, de una manera visual, legible por humanos - no es necesario al programa o script nada. Es intuitivo para los principiantes, pero lo suficientemente potente como para usuarios avanzados para trabajar sin obstáculos [31].

Scirra ya no desarrolla Construct Classic, con el fin de centrarse en Construct 2. Construct Classic es, sin embargo, de código abierto, y nos hemos entregado durante el desarrollo de la comunidad. Varios voluntarios han estado haciendo cambios de mantenimiento. Scirra ahora actúa como una incubadora para el proyecto. Aún así, es libre para descargar la versión completa - sin pantallas de la queja, anuncios o funciones restringidas, sólo el programa completo de forma gratuita. Construct está licenciado bajo la GPL.

Diseñadores indie de juegos y los aficionados pueden utilizar Construct para entrar en el mundo de la creación del juego. Los artistas pueden producir juegos sin tener que utilizar ningún tipo de programación. Profesores y estudiantes pueden usar Construir para enseñar los principios de la lógica de una manera divertida. Los desarrolladores pueden utilizar para crear rápidamente maquetas y prototipos - o simplemente como una alternativa más rápida a la codificación.

Es de código abierto y desarrollado por los voluntarios que trabajan en su propio tiempo. Construct ejecuta en Windows XP, Vista y Windows 7. Se recomienda una tarjeta gráfica con Pixel Shader 2.0 y al menos 64 MB de memoria de vídeo.

8.2.10 Unity

Es un ambiente de desarrollo integrado (IDE) para la creación de juegos 3D, video y otros contenidos interactivos tales como visualizaciones arquitectónicas en tiempo real. Los juegos producidos por Unity se pueden ejecutar en gran variedad de consolas y sistemas operativos tales como Windows, Mac, Xbox 360, PlayStation 3, Wii, iPhone/iPad, Android, Chrome, Flash y próximamente Linux [32].

Consta de dos elementos principales: un editor para el desarrollo/diseño de contenidos y un motor de juego. Ambos están estrechamente integrados, lo cual permite que desde el

mismo editor se puedan realizar acciones que invocan al motor de juego. Un ejemplo de esto es que desde el mismo editor puedes visualizar en vivo tus creaciones. El editor de Unity también cuenta con un profiler que provee estadísticas sobre los distintos aspectos relacionados con la construcción del juego.

En general, podemos decir que Unity es un ambiente de desarrollo de juegos que destaca por soportar múltiples plataformas para ejecutar los juegos creados, así como por la facilidad de uso y productividad de su editor. Esto lo ha convertido en una herramienta muy popular en los últimos años.

Unity está disponible en distintas versiones y precios. Aunque existe una versión gratuita, esta no contiene toda la funcionalidad de la edición Pro, además de que los juegos publicados tienen un sello de agua o despliegan un splashscreen de Unity. Unity Pro no contiene estas limitaciones y su precio es de $1,500 USD. Sin embargo ahí no termina la cosa porque dependiendo de la plataforma para la que deseas desarrollar/desplegar puedes requerir complementos adicionales. Por ejemplo, para crear juegos para iPhone/iPad necesitas el complemento para iOS, que a su vez está disponible en dos ediciones: básica ($400 USD) y Pro ($1,500 USD). Entonces, una licencia de Unity Pro + el add-on de iOS Pro saldría en $3,000 USD.

8.2.11 Torque 2D

Además de ser un motor gráfico 3D, proporciona código para redes, scripting, su propio editor y creación de GUI. EL código fuente puede ser compilado para las plataformas Windows, Macintosh, Linux, Wii, Xbox 360, y iPhone. Es proporcionado con un kit para principiantes que contiene un tutorial sobre como hacer un FPS. Un kit de principiantes para desarrollo de un Videojuego de estrategia en tiempo real está disponible para ser comprado por separado. Estos paquetes para principiantes pueden ser modificados para adaptarse a las necesidades del desarrollador, o para que el desarrollador pueda empezar desde cero [33].

El motor admite la carga de Modelos 3D en el Formato de archivo. DTS y el formato de archivo DIF. Ofrece un motor de terrenos que crea automáticamente LODs de la tierra par que se renderice el menor número de polígonos necesarios en cada momento. El terreno se ilumina automáticamente y las texturas aplicadas al terreno pueden ser mezcladas a la perfección. El motor de renderizado del juego ofrece sombreado, niebla volumétrica y otros efectos como las etiquetas que permiten texturas que se proyectan en interiores en tiempo real (por ejemplo, un jugador en un juego de TGE puede disparar un arma que deja un agujero de bala en la pared. el agujero de bala sería una calcomanía). TGE soporta juegos en red a través de LAN e Internet, con una arquitectura cliente-servidor tradicional. Los objetos de servidor crean una copia fantasma en los clientes y se actualizan periódicamente por eventos.

8.2.12 GameMaker Studio

Es un clásico para el desarrollo en iOS, Android y Web. Una plataforma a la que es muy sencillo entrar y que dispone de las suficientes herramientas como para que los menos avanzados en el desarrollo se sientan cómodos y puedan adelantar sus prototipos [34].

Especialmente diseñado para crear juegos en dos dimensiones incorpora de serie la popular librería de físicas Box 2D y conexión directa con los principales servicios de monetización, anuncios y analíticas del mercado. Que esto no sólo va de hacer juegos chicos, que se trata de monetizarlos.

8.2.13 DX Studio

El sistema incluye editores de diseño 2D y 3D, y permite el control de JavaScript de escenas, objetos y medios de comunicación en tiempo real. Los documentos también se pueden controlar desde el exterior del reproductor mediante la interfaz de ActiveX / COM o un puerto TCP / IP. El motor detrás de DX Studio utiliza DirectX 9.0c para aprovechar al máximo la aceleración de hardware de gráficos 3D, e incluye soporte para el último pixel y efectos de sombreado de vértices que se encuentran en los más potentes tarjetas gráficas 3D [35].

El DX Studio 2D y editores 3D se pueden utilizar para construir capas y secuencias interactivas, que se combinan para producir un documento interactivo completo. Con los usuarios de tecnología ActiveX puede construir sus propias aplicaciones C + +, C # o VB.Net y soltar el jugador DX Studio en un componente. Un documento interactivo completo se puede compilar en un solo EXE redistribuible.

Los archivos producidos por DX Studio uso estándar XML para describir la totalidad de las escenas. Los archivos también contienen todos los recursos necesarios para mostrar al mundo 3D, comprimidos en el mismo archivo utilizando algoritmos compatibles ZIP estándar. Una opción de seguridad permite que estos datos sean encriptados si es necesario.

Construido en los efectos especiales incluyen erupciones llenas de lentes, ondulaciones del agua, sistemas de partículas, sombras en tiempo real, proyección de vídeo en 3D (en formato MPEG o AVI), sonido posicionado 3D y efectos de post-producción (como "sepia", "flor 'y' corona ').

Para los usuarios avanzados un plug-in SDK está disponible que, con un poco de DirectX / C + + o conocimiento HLSL, los usuarios pueden codificar sus efectos.

Framework	Características	Ventajas	Desventajas
Unreal Engine / UDK	Escrito en C++ y utiliza distintas librerías gráficas para abarcar una amplia gama de sistemas de entretenimiento, de su primer motor hay versiones para Windows bajo DirectX, GNU/Linux y Macintosh bajo OpenGL, PlayStation2 y Dreamcast. Los scripts se crean en un lenguaje de programación propietario (UnrealScript).	Ayuda al desarrollo completo de un videojuego, como las físicas, la música, los efectos de sonido, la comunicación en red o el diseño de niveles. Se puede crear juegos para distintas plataformas incluyendo consolas, PC, Mac, Android, iOS y Playstation Vita.	Solo está disponible para Windows. Para fines educativos o sin lucro, es gratuito. En el caso de empresas que planean crear juegos comerciales con UDK, un pago inicial de 99 dólares y en caso de que obtengan ganancias superiores a los 50 mil dólares dará una comisión del 25% de sus ingresos a Epic.
Cocos 2D-X	Mantiene la estructura del API original pero permite utilizar otros lenguajes de programación como C++, Lua, Javascript y C#. Motor de juegos 2D rápido para dispositivos móviles. Lenguaje escrito en C++.	Utiliza la API gráfica de OpenGL ES mejorando el desempeño de la aplicación en el dispositivo. Desarrollo multiplataforma (Windows, Mac y Linux). Todos los códigos y herramientas de código están bien documentados y siempre libres a cualquiera. Es Open Source y gratuito.	No está claro su futuro, está siendo desarrollado por un equipo oriental, no se sabe si en algún momento dejará de darle soporte, pero de momento mantiene un ritmo constante de actualizaciones.
Cocos 2D HTML 5	Fácil integración de físicas, potente sistema de acción de Sprite, sistema de partículas, sistema de sonido y música. Diseñado para navegadores web y tecnología HTML5. Permite embeberse dentro de un webkit y conseguir una versión nativa de la aplicación.	Desarrollo multiplataforma (Windows, Mac y Linux). Todos los códigos y herramientas de código están bien documentados y siempre libres a cualquiera. Es Open Source y gratuito.	No está claro su futuro, está siendo desarrollado por un equipo oriental, no se sabe si en algún momento dejará de darle soporte, pero de momento mantiene un ritmo constante de actualizaciones.
Corona	Dispone de librerías para incluir	Posee un motor de física muy	Se pueden presentar

	OpenGL, OpenAL, Google Maps, Box2D, Facebook Connect o Game Center en los desarrollos. Lenguaje escrito LUA.	avanzado. Desarrollo Multiplataforma, tanto para iOS (iPhone, iPad) como para Android. Integración automática con OpenGL-ES	inconvenientes al compilar, ya que se realiza de manera online y se debe pagar las siguientes tarifas anuales para publicar 199$ en Android, 199$ en iOS ó 349$ en iOS, Android, Kindle Fire y NOOK.
Libgdx	Soporte 2D full (bajo y alto nivel) Lenguaje escrito java. Maneja Audio, input (usuario), física, matemática, archivos.	Desarrollo multiplataforma (Windows Linux OS X iOS Android HTML5). Buena documentación, tutoriales, ejemplos de código.	El soporte de alto nivel en esta en construcción actualmente
Marmalade Quick	Se basa en componentes de código abierto lo mejor en su clase, como Cocos2d-x y Box2D. Lenguaje escrito LUA.	Libre para un juego rápido en 2D y desarrollo de aplicaciones, ahora con apoyo para Windows Phone 8 Puede dirigirse a iOS, Android, BlackBerry, y ahora Windows Phone 8.	La opción de descarga gratuita no permite tener todas las funcionalidades del Framework, para las versiones de pago hay pruebas que permiten evaluar durante de 30 días.
Allegro	Bibliotecas dirigidas principalmente a los videojuegos y programación multimedia. Lenguaje escrito C. Se encarga de tareas tales como la creación de ventanas, aceptar la entrada del usuario, la carga de datos, elaboración de imágenes, reproducir sonidos, etc.	Desarrollo multiplataforma (Windows Linux OS X DOS).	A pesar de las grandes funcionalidades, no es un motor de juego: la persona es libre de diseñar y estructurar el programa como desee. Sólo admite primitivas gráficas 2D de forma nativa, pero se puede utilizar junto a una API de 3D (por ejemplo, OpenGL, Direct3D, y bibliotecas de nivel superior).
Construct Classic	Creador de juegos HTML5 diseñada específicamente para los juegos 2D.	Varios tutoriales y buen soporte/foros. No requiere programar.	Solo esta disponible para la plataforma HTML5. Incluye versión gratuita, pero limitada y para uso no comercial.
Unity	Desarrollado por Unity Technologies que se está abriendo	Desarrollo multiplataforma (para consolas, iOS, Android, Linux,	La versión Free que es gratuita le faltan característica más

	pasó en la comunidad por su sencillez de uso. Lenguaje Scripts C# – JS	Window). Está diseñado para un uso muy amigable. Además la comunidad de desarrolladores es enorme y hay multitud de foros de ayuda, así como una extensa documentación.	avanzadas y la versión Pro que cuesta inicialmente 1,500 $.
Torque 2D	El código fuente puede ser compilado en plataformas de Windows, Macintosh, Linux, Wii, Xbox 360 y iPhone. Escrito en C ++.	Versión con licencia MIT ya está disponible en GitHub. Permite a los usuarios compartir fragmentos de código fácilmente mediante comportamientos y módulos. Desarrollo multiplataforma (Windows Linux OS X).	Incluye acceso al lenguaje de Scripting TorqueScript permitiendo a los desarrolladores escribir la misma lógica del juego pero solo para Windows y OS x y así si funcione en otras plataformas.
GameMaker Studio	Incluye su fácil entorno de desarrollo integrado (IDE) potente lenguaje de scripting y Box 2D Physics Está diseñado para permitir a sus usuarios desarrollar fácilmente videojuegos sin tener que aprender un complejo lenguaje de programación como C + + o Java.	Apertura a hacer todo tipo de juego, incluso online, o juegos estratégicos, todo esta en cuestión de la programación. Exportación multiplataforma.	Solo permite crear juegos casuales y sociales para, iOS, Android, PC y la Web (HTML5). El IDE sólo funciona en sistemas Microsoft Windows.
DX Studio	Desarrollo exhaustivo que permite crear gráficos 3D interactivos para juegos, simulaciones, salvapantallas y otras aplicaciones, ya sea en solitario o integrados en una web.	Buen soporte y documentación. Ofrece buenas características en elementos como sistema de terreno, luz y sombra, inclusión de física, efectos especiales, controles y módulos entre otros.	Solo funciona para el sistema operativo Windows.

Tabla 2 Comparación Frameworks o motores de juego existentes. Fuente: Creación del autor.

8.3 LISTADO DE LOS PRINCIPALES SISTEMAS OPERATIVOS

8.3.1 Microsoft Windows

Es el nombre de una familia de sistemas operativos desarrollados y vendidos por Microsoft. Microsoft introdujo un entorno operativo denominado Windows el 25 de noviembre de 1985 como un complemento para MS-DOS en respuesta al creciente interés en las interfaces gráficas de usuario (GUI).1 Microsoft Windows llegó a dominar el mercado mundial de computadoras personales, con más del 90% de la cuota de mercado, superando a Mac OS, que había sido introducido en 1984.

Las versiones más recientes de Windows son Windows 8 para equipos de escritorio, Windows Server 2012 para servidores y Windows Phone 8 para dispositivos móviles. La primera versión en español fue Windows 3.0.

Windows es un sistema operativo basado en ventanas. La primera versión se lanzó en 1990 y comenzó a utilizarse de forma generalizada gracias a su interfaz gráfica de usuario (GUI, Graphical User Interface). Hasta ese momento, el sistema operativo más extendido era MS-DOS (Microsoft Disk Operating System), y la interfaz consistía en una línea de comandos.

8.3.2 Linux

GNU/Linux es uno de los términos empleados para referirse a la combinación del núcleo o kernel libre similar a Unix denominado Linux con el sistema GNU. Su desarrollo es uno de los ejemplos más prominentes de software libre; todo su código fuente puede ser utilizado, modificado y redistribuido libremente por cualquiera bajo los términos de la GPL (Licencia Pública General de GNU, en inglés: General Public License) y otra serie de licencias libres.1

A pesar de que Linux es, en sentido estricto, el núcleo del sistema operativo,2 parte fundamental de la interacción entre el hardware y el usuario (o los programas de aplicación) se maneja usualmente con las herramientas del proyecto GNU y con entornos de escritorio basados en GNOME, que también forma parte del proyecto GNU aunque tuvo un origen independiente. Sin embargo, una parte significativa de la comunidad, así como muchos medios generales y especializados, prefieren utilizar el término Linux para referirse a la unión de ambos proyectos. Para más información consulte la sección "Denominación GNU/Linux" o el artículo "Controversia por la denominación GNU/Linux".

A las variantes de esta unión de programas y tecnologías, a las que se les adicionan diversos programas de aplicación de propósitos específicos o generales se las denomina distribuciones. Su objetivo consiste en ofrecer ediciones que cumplan con las necesidades de un determinado grupo de usuarios. Algunas de ellas son especialmente conocidas por su uso

en servidores y supercomputadoras.3 donde tiene la cuota más importante del mercado. Según un informe de IDC, GNU/Linux es utilizado por el 78% de los principales 500 servidores del mundo,4 otro informe le da una cuota de mercado de 89% en los 500 mayores supercomputadores.5 Con menor cuota de mercado el sistema GNU/Linux también es usado en el segmento de las computadoras de escritorio, portátiles, computadoras de bolsillo, teléfonos móviles, sistemas embebidos, videoconsolas y otros dispositivos.

8.3.3 Mac OS x

OS X antes llamado Mac OS X, es una serie de sistemas operativos basados en Unix, desarrollados, comercializados y vendidos por Apple Inc. que ha sido incluido en su gama de computadoras Macintosh desde el año de 2002.7 8 Es el sucesor del Mac OS 9 (la versión final del Mac OS Classic), el sistema operativo de Apple desde 1984.9 Está basado en BSD, y se construyó sobre las tecnologías desarrolladas en NeXT entre la segunda mitad de los 80's y finales de 1996, cuando Apple adquirió esta compañía.10 11 Desde la versión Mac OS X 10.5 Leopard para procesadores Intel, el sistema tiene la certificación UNIX 03.12

La primera versión del sistema fue Mac OS X Server 1.0 en 1999, y en cuanto al escritorio, fue Mac OS X v10.0 «Cheetah» (publicada el 24 de marzo de 2001).13 Para dispositivos móviles Apple produce una versión específica: el iOS para el iPhone, el iPod Touch,14 el iPad y el Apple TV, que usa una versión adaptada.15 Los nombres de las versiones de Mac OS X tienen nombre de grandes felinos, por ejemplo: Mac OS X v10.7 es denominado «Lion». En Mac OS X, la X denota el 10 en número romano y se constituye en parte prominente de la identidad de la marca.16

La variante para servidores, Mac OS X Server, es arquitectónicamente idéntica a su contraparte para escritorio, además de incluir herramientas para administrar grupos de trabajo y proveer acceso a los servicios de red. Estas herramientas incluyen un servidor de correo, un servidor Samba, un servidor LDAP y un servidor de dominio entre otros. Viene preinstalada en Apple Xserve, aunque puede ser utilizado en la gran mayoría de computadores actualmente distribuidos por el fabricante.

Nivel de participación en el mercado de los sistemas operativos mencionados anteriormente a la fecha junio de 2013[36]:

1. Windows: 91.51%
2. Mac: 7.20%
3. Linux: 1.28%

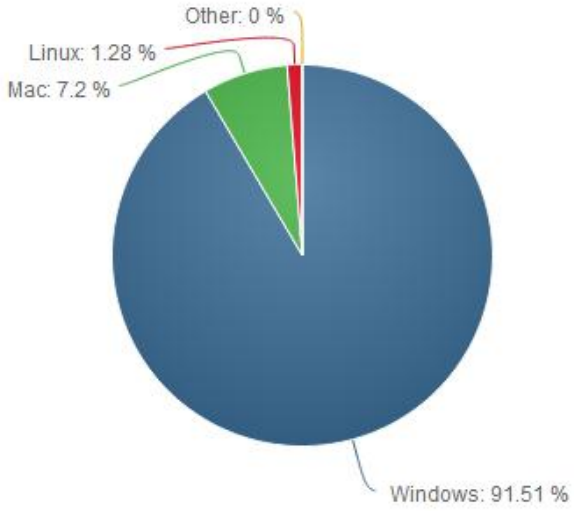

Figura 10 Sistemas Operativos según Arquitecturas .Fuente: Zenca.es 2013

Tras visualizar esta gráfica, queda muy claro que Windows sigue siendo el sistema por excelencia en el mercado de PC, ya que se sitúa con una cuota de mercado de mas del 91%, seguido del sistema Mac OS X de Apple, y por último están los sistemas Linux, ligeramente por en cima del 1%.

También el portal de Net Market Share nos han publicado los datos en referencia a las estadísticas de ventas en el periodo relativo a Agosto de 2012 a Junio de 2013, con información más detallada [37]

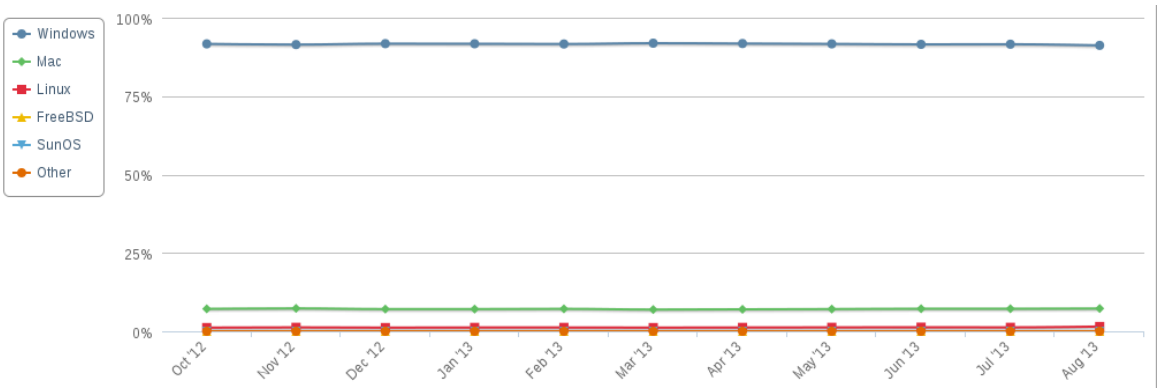

Figura 11 Sistemas operativos más utilizados del mercado Fuente: netmarketshare 2013

MONTH	WINDOWS	MAC	LINUX	FREEBSD	SUNOS	OTHER
October, 2012	91.67%	7.16%	1.17%	0.00%	0.00%	0.00%
November, 2012	91.45%	7.30%	1.25%	0.00%	0.00%	0.00%
December, 2012	91.74%	7.07%	1.19%	0.00%	0.00%	0.00%
January, 2013	91.71%	7.08%	1.21%	0.00%	0.00%	0.00%
February, 2013	91.62%	7.17%	1.21%	0.00%	0.00%	0.00%
March, 2013	91.89%	6.94%	1.17%	0.00%	0.00%	0.00%
April, 2013	91.78%	7.01%	1.21%	0.00%	0.00%	0.00%
May, 2013	91.67%	7.07%	1.26%	0.00%	0.00%	0.00%
June, 2013	91.51%	7.20%	1.28%	0.00%	0.00%	0.00%
July, 2013	91.56%	7.19%	1.25%	0.00%	0.00%	0.00%
August, 2013	91.19%	7.28%	1.52%	0.00%	0.00%	0.00%

Figura 12 Porcentaje de utilización de los sistemas operativos más utilizados del mercado en meses.
Fuente: netmarketshare 2013

9. FASE MODELADO

9.1 CRITERIO DE SELECCIÓN ELEMENTOS O PRINCIPIOS DE USABILIDAD

Las 10 reglas heurísticas de usabilidad se refieren a las técnicas basadas en la experiencia para la solución de problemas, el aprendizaje y el descubrimiento. Cuando la búsqueda exhaustiva es impracticable, los métodos heurísticos son utilizados para acelerar el proceso de búsqueda de una solución satisfactoria; atajos mentales para aliviar la carga cognitiva de tomar una decisión [38].

Además estas reglas nos permiten realizar inspecciones de usabilidad en un plazo de tiempo muy corto y proporcionan resultados inmediatos en cuanto a problemas de usabilidad. Cuando se realizan desde el principio del ciclo de vida del producto (por ejemplo, durante la especificación o el diseño) se pueden conseguir ahorros importantes, ya que es más fácil cambiar y adaptar la aplicación antes de que esté totalmente desarrollada.

Basado en el listado anterior de la heurísticas de usabilidad el criterio de selección para los elementos que tendrá el editor propuesto se esperaría de aquellos que permitan editar un escenario simple, consistente y eficiente, que no sea una herramienta por la cual deba pasar por un proceso de aprendizaje extenso, para cumplir con ello se seleccionaron los siguientes elementos:

- Visibilidad del estado del sistema

El sistema siempre debe mantener a los usuarios informados sobre lo que está pasando, a través de información adecuada en un plazo razonable.

Esto podría ser uno de los más fáciles de poner en práctica en el diseño de un sistema, aunque es de los que más se pasa por alto. Sin embargo esto puede sonar simplista, esto puede ser la causa de una gran frustración para los usuarios. Los usuarios necesitan una cierta clase de señal visual para asegurarles que la aplicación está respondiendo a sus acciones. Esto incluye haciéndoles saber si han hecho algo bien o no, y también muestra una situación en relación con la acción que el sistema está tomando.

- Control y libertad del usuario

Los usuarios a menudo eligen funciones del sistema por error y necesitarán un marcado claramente como "salida de emergencia" para salir del estado no deseado sin tener que pasar por un diálogo extendido.

Proporcionar a los usuarios la libertad les hace saber que tienen el control de la aplicación y no al revés.

No todas las acciones en una tarea necesariamente se pueden revertir. Puede haber situaciones en aplicaciones en las que después de realizar una acción que el estado de la tarea ha cambiado y no sería posible restaurarlo a su estado anterior completo para solucionar esto se hace necesario pedir confirmación cada vez que puedas sin ser molesto.

Control del usuario y la libertad proporciona a los usuarios la oportunidad de centrarse más en la tarea a realizar y se preocupan menos acerca de cómo utilizar la aplicación con cuidado, sin causar ningún daño.

- Prevención de errores

No todos los errores son causados por los usuarios. Algunos errores son inevitables porque son causados por la conectividad y la dependencia de otros sistemas de red, entre otros. En la medida de lo posible tratar de mitigar las consecuencias de los errores mediante la comprobación. Es mejor prevenir que curar.

- Reconocer antes que recordar

Minimizar la carga de memoria del usuario mediante objetos de decisiones, acciones y opciones visibles. El usuario no debería tener que recordar información de una parte del diálogo a otra. Las instrucciones de uso del sistema deben ser visibles o fácilmente recuperables cuando sea apropiado.

Es más fácil para los usuarios cuando se muestran las opciones y dejar que ellos elijan, en lugar de verse obligados a remitir opciones. El cerebro humano está diseñado para reconocer las cosas rápidamente, pero un recuerdo es más lento. Esta fue la razón por la que se introdujeron interfaces gráficas de usuario (GUI) que tuvieron un gran paso hacia una mejor usabilidad.

Los usuarios reconocer imágenes muy rápidamente y también puede asociar una acción o información específica con la imagen. Esto hace las imágenes una gran manera de expresar la funcionalidad para los usuarios. Por ejemplo, cuando los usuarios ven un botón con el dibujo de una impresora se reconoce de inmediato lo que el botón haría. Sin embargo, las imágenes no siempre tienen que ser los objetos conocidos de la vida real - también podrían ser iconos y símbolos diseñados a medida. Los usuarios pueden aprender a iconos y símbolos asociados si están bien diseñados.

- Flexibilidad y eficiencia de uso

Los usuarios avanzados utilizan sistemas de forma diferente a los usuarios novatos. Para que los usuarios principiantes se familiaricen con el sistema y realicen ciertas tareas con frecuencia, el sistema debe ser capaz de dejar que esas tareas sean accesibles de manera más eficiente.

Los sistemas deben tener "aceleradores" que ayudan a los usuarios avanzados realizan las tareas más utilizadas eficientemente. Incluyen:

- Atajos para las tareas más frecuentes, abrir, guardar, cerrar, exportación, etc.
- Navegación mediante el teclado, navegar a través de un álbum de fotos en línea con las flechas del teclado.
- Migas, sino que permita que los usuarios sepan dónde se encuentran y dejan fácilmente navegar a una página anterior en lugar de tener que recordar de memoria donde estaban antes.
- Menú contextual; tareas de uso frecuente se puede colocar en el menú contextual (botón derecho del ratón en Windows, Ctrl-clic del ratón en Mac OS, pulsación larga en dispositivos móviles)

- Estética y diseño minimalista

Los diálogos no deben contener información que es irrelevante. Cada unidad adicional de información en un diálogo compite con las unidades relevantes de información y disminuye su visibilidad relativa.

Un diseño minimalista atrae a los usuarios a centrarse en el tema principal que nos ocupa sin distraerse con imágenes o textos irrelevantes. Mantenga la información que aparece en la simple aplicación. Incluir sólo la información pertinente, de lo contrario, el mensaje se pierde en el "ruido".

- Clasificar por categorías: Evitar visualizar información repetitiva. Para aumentar la legibilidad, clasificar información repetitiva en las secciones pertinentes. Los usuarios pueden acceder rápidamente a la sección que quieren, y tomar de inmediato la información que tienen que ver con la mayoría.

- Colores: son una gran manera para que enfoque la atención del usuario en un área en particular de la pantalla, por lo que la elección de la paleta de color y detalles son importantes. Comprender la ciencia del color y saber el significado de los colores y lo bien que interactúan entre sí es fundamental para un buen diseño minimalista

- Espacio en blanco: añade un respiro a la información que se muestra y hace que parezca menos "ocupado". Un buen uso de los espacios en blanco puede dar la información que se muestra un aspecto elegante y rico y puede mejorar la legibilidad.

9.1.1 Método evaluación de usabilidad

Recorrido cognitivo (cognitive walkthrough): Este método de inspección de la usabilidad se centra en evaluar la facilidad de aprendizaje de un prototipo de un sistema, básicamente por exploración, y esto esta motivado por la observación de que muchos usuarios prefieren aprender software a partir de explorar sus posibilidades (Wharton y otros, 1994) [39].

En el recorrido cognitivo, los revisores evalúan una propuesta de interfaz en el contexto de una o más de sus tareas.

9.2 CRITERIO DE SELECCIÓN PRINCIPALES FRAMEWORKS EXISTENTES PARA EL DESARROLLO DE VIDEOJUEGOS 2D

Tras considerar detenidamente el cuadro comparativo anterior y la investigación hecha sobre los diversos Frameworks para el desarrollo de juegos, teniendo como criterios de decisión para el desarrollo del editor propuesto se seleccionará aquel que sea libre (no genere restricciones por licenciamiento), que sea multiplataforma, es decir, que funcione sobre los principales sistemas operativos del mercado (Linux, Windows y Mac Os) y por último que sea fácilmente integrable a un Framework de GUI (Por ejemplo Qt creator y GTK).

A partir de lo anterior se decidió emplear Cocos2D-HTML 5 como solución ya que, gracias a C++, podemos exportar nuestros juegos a un porcentaje muy elevado de dispositivos. Además, nos atrae enormemente la comunidad de desarrolladores tan activa que hay detrás de Cocos2D-X y permite una flexibilidad en la integración.

Cocos2d-html5 es un motor de juegos 2D multiplataforma escrito en Javascript, sobre la base de Cocos2d-X y bajo licencia MIT. Incorpora la misma API de alto nivel como "motor de JS de Cocos2d" y compatible con Cocos2d-X. Actualmente soporta lienzo (Canvas).

El objetivo de cocos2d-html5 es utilizar el mismo conjunto de código para ejecutar en cocos2d-x y cocos2d-iphone escritas en javascript para navegadores compatibles con HTML5. Todas las API se derivan de cocos2d-x y cocos2d. Así que está familiarizado con cualquiera de los motores de juego cocos2d Y como cocos2d-html5 se basa en web, y puede escribir su juego en cualquier plataforma con un editor de texto simple. Y ya que está basado en la web, se puede ejecutar el juego en cualquier dispositivo que tenga acceso a un navegador compatible con HTML5. Algunas de las características más relevantes son:

- Control de flujo: Administrar el control de flujo entre diferentes escenas de una manera fácil.
- Sprites: sprites rápidos y fáciles.

- Acciones: Simplemente dile a los sprites lo que usted quiere que hagan. Acciones componibles como mover, rotar, escalar y mucho más.
- Efectos: Los efectos como olas, giro, lentes y mucho más.
- Mapas de baldosas: Soporte para mapas de baldosas rectangulares y hexagonales.
- Transiciones: para pasar de una escena a otra con estilo.
- Menús: Construido en clases para crear menús
- Texto Rendering: Etiqueta y HTML Label con el apoyo de acción
- Documentación: Guía de programación + Referencia API + Tutoriales en vídeo - Máximo en pruebas sencillas que muestran cómo usarlo-
- Construido en el interprete de Python: Para propósitos de depuración
- Licencia BSD
- soporte WebGL, con la última cocos2d-html5-v2.2.1, ahora soporta WebGL, que es un gran aumento de rendimiento en los navegadores modernos.
- ¿Qué pasa con el acelerómetro y otras características de hardware?
- OpenGL basada en: la aceleración de hardware

9.2.1 ¿Por qué HTML 5?

Cada desarrollador de juegos tiene su propia opinión sobre la perspectiva de juegos construidos con HTML5, pero independientemente del contexto teórico, una simple pero realista ventaja es que hay una gran cantidad de plataformas puede desplegar su juego y significa más usuarios, más exposición y más ganancias [40].

Además, HTML5 está lleno de magia:

Figura 13 Ejemplo HTML5.
Fuente:https://dl.dropboxusercontent.com/u/30871449/Website/Resource/2013/04/ClearFiveStages-Share/A%20trip%20from%20cocos2d-iphone%20to%20cocos2d-html5.pdf

9.2.2 ¿Javascript?

Si bien hay preocupación por la eficiencia del "lenguaje de secuencias de comandos", que es más lento que C++, pero las tecnologías como el motor Javascript V8 y el renderizado acelerado por hardware de Canvas hacen posible el desarrollo del juego. En este momento, el hardware que se encuentra en los teléfonos móviles les falta un poco de "empuje" para ejecutar javascript de manera eficiente, sin embargo, el equipo de Cocos2d-x está trabajando en algo llamado "Javascript binding for Cocos2d". Lo que esto significa es que su mismo código que se ejecuta en el motor Cocos2d-html5 puede trabajar sin problemas en Cocos2d-X y Cocos2d-iPhone sin o con muy pocas modificaciones. Y todo lo que se traduce en "rapidez casi nativa en los teléfonos móviles".

9.2.3 ¿Qué pasa con WebGL?

WebGL significa OpenGL para la web, lo que significa aceleración de hardware 3D. Cuando los teléfonos móviles adoptan WebGL, la norma de los juegos Cocos2d-html5 se desarrollará sin problemas y sólo limitado por la imaginación del diseñador del juego.

9.3 CRITERIO DE SELECCIÓN LISTADO DE LOS PRINCIPALES SISTEMAS OPERATIVOS

Para cumplir con el objetivo de que la herramienta propuesta sea multiplataforma y libre, a partir de la investigación anterior y de las estadísticas de los principales sistemas operativos utilizados en el mercado para aplicaciones de escritorio desde la fecha de agosto de 2012-2013, se seleccionarán los tres sistemas operativos descritos.

Entendiendo que principalmente el desarrollo de la mayoría de los videojuegos 2D se hace para el sistema operativo Windows y Mac como lo muestran las estadísticas, dejando a Linux muy por debajo, a continuación se mostrará un artículo donde Valve's SteamOS and Steam Machine muestran a Linux como el sistema operativo del futuro para los videojuegos.

Valve ha anunciado el próximo lanzamiento de SteamOS, un nuevo sistema operativo basado en Linux construido alrededor de Steam. Con el nuevo espíritu de código abierto de Steam, Valve también anunció un prototipo de hardware hoy en día, que será lanzado al público en 2014, y la compañía pidió ayuda a la comunidad. "Como siempre, creemos que la mejor manera de asegurar que los productos correctos están siendo hechas para que la gente los pruebe y luego hacer cambios sobre la marcha", dice el anuncio. La compañía planea lanzar 300 de las cajas de beta para los usuarios de Steam, de forma gratuita, para su análisis [41].

Valve ve a Linux como el futuro de los videojuegos. Hace varios años, Valve comenzó a preocuparse por la dirección de la plataforma PC o escritorio y su modelo propietario, de

código cerrado. Ahora se están dando cuenta de la disminución importante año tras año en unidades de PC vendidas. En el futuro, Newell dice que la democratización de los juegos puede desdibujar las líneas entre el creador de juegos y de consumo. Y aquí es donde SteamOS tiene el potencial de ser un elemento de cambio real, por así decirlo.

10. FASE CONSTRUCCIÓN

10.1 SELECCIÓN METODOLOGÍA DE DESARROLLO SOFTWARE

Una metodología es una colección de procedimientos, técnicas, herramientas y documentos auxiliares que ayudan a los desarrolladores de software en sus esfuerzos por implementar nuevos sistemas de información. Una metodología esta formada por fases, cada una de las cuales se puede dividir en sub-fases, que guiarán a los desarrolladores de sistemas a elegir las técnicas más apropiadas en cada momento del proyecto y también a planificarlo, gestionarlo, controlarlo y evaluarlo [42].

Para asegurar el éxito durante el desarrollo de software no es suficiente contar con notaciones de modelado y herramientas, hace falta un elemento importante: la metodología de desarrollo, la cual nos provee de una dirección a seguir para la correcta aplicación de los demás elementos.

Generalmente el proceso de desarrollo llevaba asociado un marcado énfasis en el control del proceso mediante una rigurosa definición de roles, actividades y artefactos, incluyendo modelado y documentación detallada. Este esquema "tradicional" para abordar el desarrollo de software ha demostrado ser efectivo y necesario en proyectos de gran tamaño (respecto a tiempo y recursos), donde por lo general se exige un alto grado de ceremonia en el proceso. Sin embargo, este enfoque no resulta ser el más adecuado para muchos de los proyectos actuales donde el entorno del sistema es muy cambiante, y en donde se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad [43].

10.1.1 Comparación Metodologías Ágiles y Tradicionales

A continuación se enumera las principales diferencias de una Metodología Ágil respecto de las Metodologías Tradicionales (llamadas peyorativamente “no ágiles” o “pesadas”). La imagen recoge estas diferencias que no se refieren sólo al proceso en sí, sino también al contexto de equipo y organización que es más favorable a cada uno de estas filosofías de procesos de desarrollo de software.

Metodologías ágiles	Metodologías tradicionales
Se basan en heurísticas provenientes de prácticas de producción de código	Se basan en normas provenientes de estándares seguidos por el entorno de desarrollo
Preparados para cambios durante el proyecto	Cierta resistencia a los cambios
Impuestas internamente por el equipo	Impuestas externamente
Proceso menos controlado, con pocos principios	Proceso muy controlado, numerosas normas
Contrato flexible e incluso inexistente	Contrato prefijado
El cliente es parte del desarrollo	Cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños (<10)	Grupos grandes
Pocos artefactos	Más artefactos
Menor énfasis en la arquitectura del software	La arquitectura del software es esencial

Tabla 3 Comparación metodologías tradicionales y ágiles. Fuente: Canós, J et al, 2005.

La primera decisión que se debe tomar a la hora de seleccionar una metodología de desarrollo de software es si usar una metodología ágil o una metodología tradicional, en este, punto se descartan las metodologías tradicionales, esta decisión se toma en primer lugar ya que el proyecto tiene una curva de aprendizaje muy alta, debido a esto no se debe gastar mucho tiempo en investigar y documentar, sino en ir aprendiendo a medida que se interactúa con la aplicación funcional, como segundo punto, debe ser lo más flexible posible ante cambios posteriores, por tal razón se decide elegir una metodología ágil porque promueven las entregas rápidas, no más de dos semanas entre una entrega y otra; y se recibe desde el principio un producto tangible y puede detectar errores cometidos en la fase de análisis.

A continuación se mencionaran las metodologías más comunes y se elige la que se adapte mejor a las condiciones del proyecto para ser usada

- Scrum
- XP (eXtreme Programming)
- Kanban
- Scrumban

La metodología XP da mucha importancia o está centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en 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. XP se define como especialmente adecuada para proyectos con requisitos imprecisos, por lo que para el proyecto ya es claro su objetivo, además de que promueve que todo el código sea escrito en parejas trabajando en el mismo ordenador, a partir de esto se ha decidido descartar aquella metodología [44].

En Scrum un proyecto se ejecuta en bloques temporales (iteraciones-sprints) de un mes natural (pueden ser de dos o tres semanas, si así se necesita). Cada iteración tiene que proporcionar un resultado completo, un incremento de producto que sea susceptible de ser entregado con el mínimo esfuerzo cuando el cliente lo solicite. Por lo anterior pueden surgir varios problemas por las reuniones interminables de planificación y estimaciones poco afortunadas. Al analizar la metodología se necesita de alguna que sea algo más flexible.

La metodología Kanban, tiene como objetivo gestionar de manera general como se van completando tareas, pero en los últimos años se ha utilizado en la gestión de proyectos de desarrollo software. Las principales reglas de Kanban son las siguientes:

- Visualizar el trabajo y las fases del ciclo de producción o flujo de trabajo.
- Determinar el límite del "trabajo en curso" (WIP - Work In Progress).
- Medir el tiempo en completar una tarea (Lead time).

En resumen aunque la metodología esté preparada para escalar sin problemas, los temas operativos a pie son un verdadero reto, y cada vez que una nueva versión salga será un problema considerable, primero por lo costoso del despliegue y segundo porque aparecerían bugs que complican todo más.

Por último la metodología Scrumban es derivada de los métodos de desarrollo Scrum y Kanban.

De Scrum
•Roles: Cliente, equipo (con los diferentes perfiles que se necesiten).
•Reuniones: reunión diaria.
•Herramientas: pizarra

De Kanban
•Flujo visual
•Hacer lo que sea necesario, cuando sea necesario y solo la cantidad necesaria.
•Limitar la cantidad de trabajo (WIP)
•Optimización del proceso.

Normas	Scrum	Scrumban
Pizarra / Herramientas	Pizarra Backlogs Gráfica burn-down	Pizarra
Reuniones	Reunión diaria Planificación Retrospectiva	Reunión diaria
Iteraciones	Sí, Sprints	No, flujo continuo
Estimaciones	Sí	No
Esquipo	Multidisciplinar	Puede ser especializado
Roles	Product Owner Scrum Master Equipo	Equipo + otros
WIP (Work In Progress)	Controlado por el contenido del Sprint	Controlado por el estado de la tarea.
Cambios	Se pasan al siguiente Sprint	Se añaden al tablero en la columna "TO DO".
Impedimentos	Solución inmediata	Se evitan.

Tabla 4 Comparación metodologías Scrum y Scrumban. Fuente: Uvadoc

En este punto del análisis se llega a la conclusión de que algunas metodologías como XP no concuerdan con lo planteado y se descartan, en otros casos aparecen algunas que nos proporcionan ventajas e inconvenientes caso tal de las metodologías Scrum y Kanban.

Scrumban, es un modelo de desarrollo especialmente adecuado para proyectos de mantenimiento o proyectos en los que las historias de usuarios (requisitos del software) varíen con frecuencia o en los cuales surjan errores de programación inesperados durante todo el ciclo de desarrollo del producto. Para estos casos, los sprints (periodos de duración constante en los cuales se lleva a cabo un trabajo en sí) de la metodología Scrum no son factibles, dado que los errores/impedimentos que surgirán a lo largo de las tareas son difíciles de determinar y por lo tanto, no es posible estimar el tiempo que conlleva cada historia. Por ello, resulta más beneficioso adoptar flujo de trabajo continuo propio del modelo Kanban [45].

En conclusión dada la naturaleza de Scrum al no ser una metodología como tal sino un Framework, permite la adaptación de las herramientas que entrega sin mayor inconveniente, de este modo el equipo no tendrá que adaptarse a la metodología sino al contrario.

El actual proyecto cuenta con ciertas características que hacen difícil la adopción de metodologías tradicionales e incluso de algunas ágiles, es por esto que se optó por usar una variante, scrumban; ésta metodología extrae lo mejor de scrum y de kanban, adaptándose fácilmente a equipos de un solo desarrollador el cual deberá desempeñar diferentes roles.

Los errores son más fáciles de adoptar y no se hace necesario esperar a que el ciclo de entrega finalice para planificar la solución de un error, en su lugar se opta por el camino de kanban

donde inmediatamente se pone en lista el error para así solucionarlo lo antes posible y con ello tener un flujo de trabajo y entregas continuas.

Ambas metodologías por lo tanto proponen un número reducido de reglas que conviene seguir si queremos sacar el máximo partido a la metodología, y aunque la mezcla no suela ser muy aconsejable sí que se pueden desarrollas prácticas combinadas. Esta combinación nos permite las siguientes ventajas:

- Scrum para las tareas previstas (Historias de Usuario en la terminología Scrum).
- Kanban para gestionar los imprevistos y errores.

Por lo tanto se trabaja sobre dos tableros independientes aunque relacionados, uno para cada metodología. En el tablero Scrum se desarrollan las tareas siguiendo la metodología estándar de Scrum (sprints, reuniones de planificación) y en otro se colocan las tareas no previstas aplicando la metodología Kanban (entran en el tablero a medida que van surgiendo y avanzan respetando el Work In Process -WIP- estipulado por estado).

Para que esto funcione correctamente tenemos que conocer la velocidad del equipo y tener en cuenta el porcentaje de dedicación que se reservará a imprevistos. Esto nos permitirá calcular correctamente las tareas que podrán entrar en el sprint. En el día a día el equipo desarrolla las tareas que corresponden al sprint en curso del tablero Scrum. Cuando se detecta un error o se genera un imprevisto se coloca en el tablero Kanban, de esta forma se da un mejor manejo y por lo tanto se decidió trabajar sobre la metodología Scrumban.

10.1 CICLO 1

10.1.1 Esquema implementación del editor de escenarios para videojuegos 2D

Figura 14 Esquema implementación del editor de escenarios para videojuegos 2D. Fuente: Creación del autor.

La figura anterior resume a grandes rasgos toda la investigación hecha en la fase de análisis y modelado, lo cual permite entender qué elementos harán parte del editor y a partir de éstos cumplir con el objetivo, facilitar la edición de escenarios.

10.1.2 Historias de usuario

Tabla 5 Historias de usuario.

ID: 1	**Área de dibujo**
Descripción Como usuario de la aplicación quiero un área de dibujo para poder editar una escena.	
Criterios de aceptación - El área de dibujo debe aparecer junto con el resto de los elementos de la interfaz. - El fondo del área de dibujo inicialmente será de color blanco. - El tamaño mínimo del área de dibujo será de 480 x 320 pixeles.	

ID: 2	**Crear escenario**
Descripción Como usuario de la aplicación quiero crear una escena para agregar elementos que componen un escenario de videojuegos.	
Criterios de aceptación - La acción para agregar la escena se ubicará en la barra de menú. - Se deberá proporcionar un tamaño para la escena. - La escena se agregará al área de dibujo. - Solo habrá una escena a la vez.	

ID: 3	**Guardar escenario**
Descripción Como usuario de la aplicación quiero guardar el escenario para continuar posteriormente con la edición del escenario.	
Criterios de aceptación - El escenario se guardará mediante la activación de un botón. - El usuario debe indicar el nombre con el que será guardado el escenario. - El escenario se guardará periódicamente en caso de cerrarse la aplicación inesperadamente.	

ID: 4	Guardar escenario ubicación
Descripción Como usuario de la aplicación quiero elegir la ubicación donde se guardará el escenario para continuar posteriormente con la edición del escenario.	
Criterios de aceptación - El escenario se guardará mediante la activación de un botón. - El usuario debe indicar el nombre con el que será guardado el escenario. - Después de guardar por primera vez el escenario, se guardará periódicamente en caso de cerrarse la aplicación inesperadamente.	

ID: 5	Crear escenario validar tamaño
Descripción Como usuario de la aplicación quiero crear una escena para agregar elementos que componen un escenario de videojuegos.	
Criterios de aceptación - El tamaño proporcionado no puede ser un valor negativo ni mayor a 2000 x 2000 pixeles.	

ID: 6	Recuperar escenario
Descripción Como usuario de la aplicación quiero recuperar una escena guardada anteriormente para continuar posteriormente con la edición del escenario.	
Criterios de aceptación - El escenario se recuperará mediante la activación de un botón. - El usuario debe seleccionar donde se encuentra el archivo.	

ID: 7	Validación Recuperar Escenario
Descripción Como usuario de la aplicación quiero recuperar una escena guardada anteriormente para continuar posteriormente con la edición del escenario.	
Criterios de aceptación - Se Filtrará el formato soportado de la escena o se mostrará un mensaje de alerta en caso que el archivo no cumpla con el formato soportado.	

ID: 8	Agregar elemento
Descripción Como usuario del sistema quiero agregar un elemento al área de dibujo para editar una escena.	
Criterios de aceptación - El elemento se puede agregar mediante la activación de un botón. - El elemento tendrá un nombre por defecto. - La posición del elemento al agregarse será el centro del área de dibujo. - Habrá un indicador de que elemento ha sido agregado. - Para el elemento Sprite se debe proporcionar una imagen correspondiente.	

Nota: Los elementos disponibles serán Sprite, Label, Layer, LayerColor y LayerGradient.

ID: 9	Selección elemento
Descripción Como usuario del sistema quiero seleccionar un elemento del área de dibujo para editar una escena.	
Criterios de aceptación - Habrá un indicador de que el elemento ha sido seleccionado. - Solo se puede seleccionar un elemento a la vez. - El indicador de selección será un recuadro azul transparente que cubrirá el tamaño del elemento.	

Nota: La selección será para los elementos Sprite, Label, Layer, LayerColor y LayerGradient.

ID: 10	Deseleccionar elemento
Descripción Como usuario del sistema quiero deseleccionar un elemento del área de dibujo para editar una escena.	
Criterios de aceptación - El elemento deberá estar seleccionado. - Solo se puede deseleccionar un elemento a la vez. - Habrá un indicador de que el elemento ha sido deseleccionado. - Se permitirá deseleccionar un elemento por el teclado.	

Nota: La deselección será para los elementos Sprite, Label, Layer, LayerColor y LayerGradient.

ID: 11	Eliminar elemento
Descripción Como usuario del sistema quiero eliminar el elemento del área de dibujo para editar una escena.	
Criterios de aceptación - El elemento deberá estar seleccionado. - Solo se puede eliminar un elemento a la vez. - El elemento se puede eliminar mediante la activación de un botón. - Habrá un indicador de que el elemento ha sido eliminado. - Se permitirá eliminar un elemento por el teclado. - El elemento se eliminará permanentemente.	

Nota: se pueden eliminar los elementos Sprite, Label, Layer, LayerColor y LayerGradient.

ID: 12	Modificar posición elemento
Descripción Como usuario del sistema quiero modificar la posición del elemento del área de dibujo para editar una escena.	
Criterios de aceptación - El elemento deberá estar seleccionado previamente. - La posición del elemento se puede modificar mediante la activación de un botón. - Habrá un indicador de que la posición del elemento ha sido modificada.	

Nota: la modificación será para los elementos Sprite, Label, Layer, LayerColor y LayerGradient, también se podrá modificar la posición para el eje Z de cada elemento.

ID: 13	Modificar nombre elemento
Descripción Como usuario del sistema quiero modificar el nombre del elemento del área de dibujo para editar una escena.	
Criterios de aceptación - El elemento deberá estar seleccionado previamente. - El nombre del elemento se puede modificar mediante la activación de un botón. - Habrá un indicador de que el nombre del elemento ha sido modificado.	

Nota: la modificación será para los elementos Sprite, Label, Layer, LayerColor y LayerGradient, de igual forma se modificará el nombre de la clase.

ID: 14	**Modificar etiqueta elemento**
Descripción Como usuario del sistema quiero modificar la etiqueta del elemento del área de dibujo para editar una escena.	
Criterios de aceptación - El elemento deberá estar seleccionado previamente. - La etiqueta del elemento se puede modificar mediante la activación de un botón. - Habrá un indicador de que se ha modificado la etiqueta del elemento.	

Nota: la modificación será para los elementos Sprite, Label, Layer, LayerColor y LayerGradient.

ID: 15	**Modificar rotación elemento**
Descripción Como usuario del sistema quiero modificar la rotación del elemento del área de dibujo para editar una escena.	
Criterios de aceptación - El elemento deberá estar seleccionado previamente. - La rotación del elemento se puede modificar mediante la activación de un botón. - Habrá un indicador de que la rotación del elemento ha sido modificada.	

Nota: la modificación será para los elementos Sprite, Label, Layer, LayerColor y LayerGradient.

ID: 16	**Modificar escala elemento**
Descripción Como usuario del sistema quiero modificar la escala elemento del área de dibujo para editar una escena.	
Criterios de aceptación - El elemento deberá estar seleccionado previamente. - La escala del elemento se puede modificar mediante la activación de un botón. - Habrá un indicador de que la escala del elemento ha sido modificada.	

Nota: la modificación será para los elementos Sprite, Label, Layer, LayerColor y LayerGradient.

ID: 17	Modificar propiedades según elemento
Descripción Como usuario del sistema quiero modificar la propiedad según el elemento del área de dibujo para editar una escena.	
Criterios de aceptación - El elemento deberá estar seleccionado previamente. - La propiedad del elemento se puede modificar mediante la activación de un botón. - Habrá un indicador de que la propiedad del elemento ha sido modificada.	

Nota:

- Para el elemento Sprite la propiedad será modificar la imagen.
- Para el elemento Label las propiedades a modificar serán: texto, color, opacidad, tamaño fuente, tipo de fuente y alineación.
- Para el elemento Layer, LayerColor y LayerGradient algunas de sus propiedades son: color, opacidad, dirección del vector y algunas de eventos.

ID: 18	Agregar nuevas funcionalidades
Descripción Como usuario de la aplicación quiero agregar nuevas funcionalidades para facilitar el proceso de edición de escenarios.	
Criterios de aceptación - Las funcionalidades estarán extendidas en plugins de Qt. - Informar que se ha agregado correctamente la funcionalidad (plugin). - Los plugins deben cumplir con cierto estándar entendible por la aplicación.	

ID: 19	Generación escenario Framework
Descripción Como usuario de la aplicación quiero que el escenario creado sea usable por el Framework Cocos2DX para continuar con el desarrollo del videojuego.	
Criterios de aceptación - Se informará o mostrará un mensaje de que el proceso de creado ha concluido. - Se genera código fuente en C++ - El escenario se puede crear para el Framework mediante la activación de un botón para exportar. - El usuario seleccionará la ubicación y nombre de los archivos.	

10.2 CICLO 2

10.2.1 Diagrama de componentes

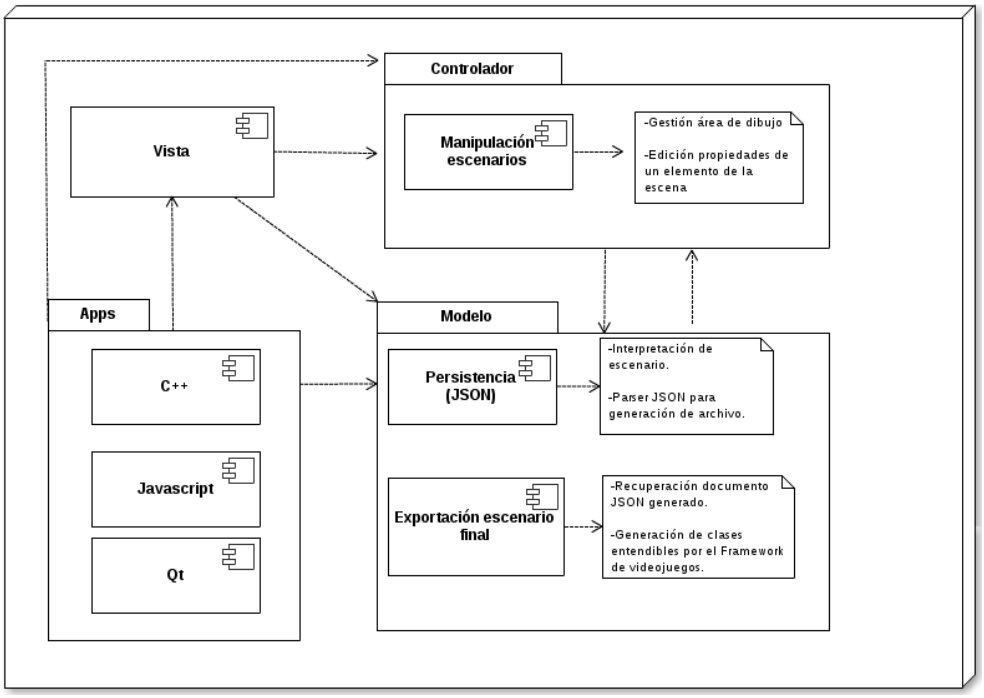

Figura 15 Diagrama de componentes. Fuente: creación del autor

La aplicación desarrollada, al ser de escritorio, tiene un diseño arquitectónico Modelo-Vista-Controlador, que separa los datos y la lógica de negocio de la aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Se puede apreciar que en el área del modelo existe una distribución de capas que permite comunicar la vista correspondiente a la interfaz de usuario escrita en el framework Qt creator junto con la vista escrita en Javascript.

10.2.2 Diagrama de clases

Al plantear el diseño de la aplicación para su correcta implementación, se decidió resumir el diagrama de clase, ocultando algunos métodos y atributos, para lograr una mejor compresión.

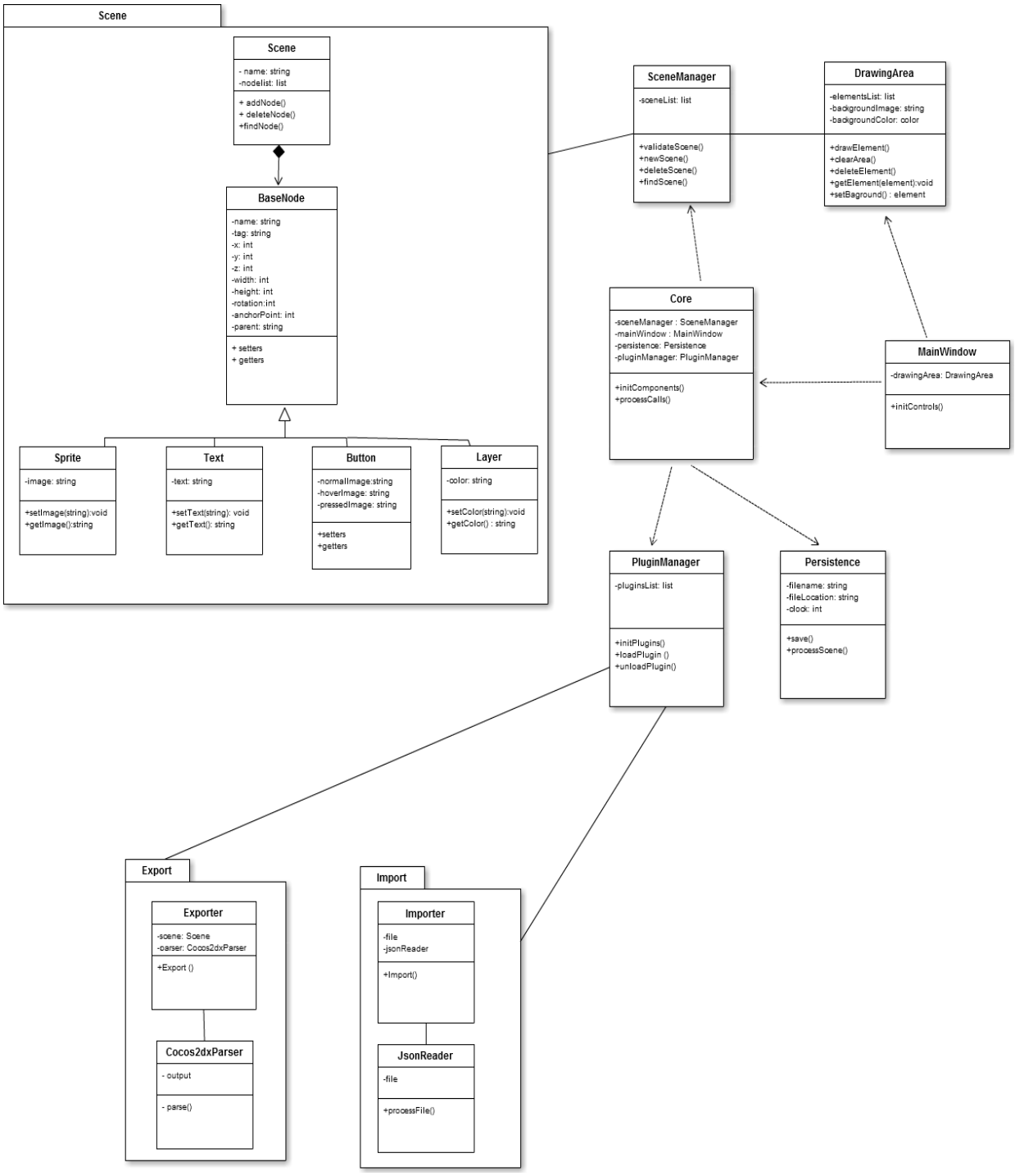

Figura 16 Diagrama de clases. Fuente: creación del autor

10.2.3 Configuración del ambiente de desarrollo

Dentro de los aspectos técnicos se optó por usar como sistema operativo para cumplir con los requisitos del presente proyecto Fedora GNU/Linux de 64 bits. Las tecnologías usadas fueron la versión LGPL del Framework Qt 5.1.1, Cocos2d-Html5 en su versión 2.2.

10.3 CICLO 3

En este ciclo se trabajó en todo lo relacionado con la codificación de la solución propuesta. En principio se desarrolló la estructura básica del sistema que abarca las historias de usuario desde el id 1 al 11. Posteriormente se estructuró la comunicación entre C++ y JavaScript. Finalmente, se trabajó en toda la implementación incial del editor propuesto. A continuación se indican algunos de los aspectos más relevantes del proceso descrito:

- Se hizo uso de las tecnologías de Qt para contener documentos con soporte de Javascript.

- La comunicación desde la interfaz gráfica de usuario hacía el canvas, se hace mediante la inyección de código Javascript.

- La comunicación por parte del canvas y la interfaz gráfica de usuario se hace mediante la inyección de un objeto Qt el cual emite eventos.

10.4 CICLO 4

En este ciclo se trabajó en la implementación de las funcionalidades requeridas para el editor, donde se abordaron las historias de usuario desde el id 11 hasta la 18. Los aspectos más importantes para culminar con el editor fueron:

10.4.1 TreeView

Desarrollo de un componente que facilite la visualización de los elementos añadidos al área de dibujo por parte del usuario. Se encuentra basado en un modelo de árbol para mostrar la relación jerárquica entre los objetos.

La arquitectura Modelo / vista de Qt proporciona una forma estándar de vistas para manipular información en un origen de datos, utilizando un modelo abstracto de los datos para simplificar y estandarizar la forma en que se accede. Los modelos simples representan datos como una tabla de elementos, y permiten vistas para acceder a estos datos a través de

un índice basado en el sistema. De manera más general, los modelos pueden ser utilizados para representar los datos en la forma de una estructura de árbol que permite a cada elemento para que actúe como un padre a una tabla de elementos secundarios.

- Estructura simple del modelo en árbol

Los datos se almacenan internamente en el modelo usando objetos Treeitem que están unidos entre sí en una estructura de árbol, basados en punteros. Generalmente, cada TreeItem tiene un elemento principal, y puede tener un número de elementos secundarios. Sin embargo, el elemento raíz en la estructura de árbol no tiene elemento principal y nunca se hace referencia fuera del modelo.

Cada TreeItem contiene información acerca de su lugar en la estructura de árbol; puede devolver su elemento primario y su número de fila. Tener esta información disponible hace que la implementación del modelo más sencillo.

Dado que cada elemento en una vista de árbol por lo general contiene varias columnas de datos (en este caso el nombre del elemento y su tipo), es natural de almacenar esta información en cada artículo. Para simplificar, vamos a utilizar una lista de QVariant objetos para almacenar los datos de cada columna en el artículo.

10.4.2 Plugin

Desarrollo de rutinas que faciliten la extensión de funcionalidades de la aplicación a la hora de procesar un escenario (plugin que permite exportar).

No sólo en sí Qt, sino también la aplicación Qt se puede ampliar a través de plugins. Esto requiere que la aplicación que detecta y cargar plugins usando QPluginLoader. En ese contexto, los plugins pueden proporcionar funcionalidad arbitraria y no están limitados a los conductores de bases de datos, formatos de imagen, codecs de texto, los estilos y los otros tipos de plugins que extienden la funcionalidad de Qt.

- Cómo hacer una aplicación extensible mediante plugins implica los siguientes pasos:

    - Definir un conjunto de interfaces (clases con sólo las funciones virtuales puras) utilizados para referirse a los plugins.

    - Utilice la macro Q_DECLARE_INTERFACE () para decirle al sistema meta-objeto de Qt acerca de la interfaz.

    - Utilice QPluginLoader en la aplicación para cargar los plugins.

- Utilice qobject_cast () para comprobar si un plugin implementa una interfaz determinada.

- Escribir un plugin comprende los pasos siguientes:

  - Declare una clase de plugin que hereda de QObject y desde las interfaces que el plugin quiere ofrecer.

  - Utilice la macro Q_INTERFACES () para decirle el sistema meta-objeto de Qt sobre las interfaces.

  - Exportar el plugin con el macro Q_EXPORT_PLUGIN2().

  - Construir el plugin con un archivo .pro adecuado.

10.5 CICLO 5

10.5.1 Generación de los ejecutables

Después de terminada la implementación del editor, se generaron los diversos ejecutables de la aplicación para los sistemas operativos seleccionados.

Sistema operativo	Requerimientos
Microsoft Windows (32 y 64 bits)	Tener instalada la librería Visual c++.
Linux (32 y 64 bits)	Se requiere tener instalado el Framework Qt creator.
Mac	

Tabla 6 Listado sistemas operativos de los ejecutables del editor. Fuente: creación del autor.

10.5.1 Pruebas elementos de usabilidad

El diseño de interfaces define un grupo de principios a seguir para hacerla lo más usable posible. En el desarrollo de la aplicación se tuvieron en cuenta los principios de heurística que definen buenas prácticas para que haya una mejor interacción entre el usuario y la aplicación.

Las pruebas que se llevaron a cabo, se hicieron sobre una versión estable de la aplicación y en un entorno real, donde se enfocaron los esfuerzos en probar los elementos de heurística anteriormente seleccionados.

Prueba	Resultado
Visibilidad del estado del sistema	La aplicación le proporciona al usuario mensajes informativos sobres las actividades más importantes realizadas entre ellas se encuentra, agregar cualquier tipo de elemento al área de dibujo, guardar una escena, cargar una escena y exportar la escena.
Control y libertad del usuario	La aplicación le permite al usuario sentirse muy cómodo con la ventana principal, puesto que pueden acomodar los paneles de propiedades y nodos de la manera que crean más conveniente.
Prevención de errores	Para prevenir errores provocados cuando se guarda un escenario se hace uso del filtrado del formato .asep, de igual forma a la hora de cargarlo. En caso de que la aplicación se cierre inesperadamente, después de guardar una escena esta seguirá guardándose cada 5 minutos.
Reconocer antes que recordar	Se minimiza la carga de memoria del usuario al proporcionarle iconos o imágenes sobre los elementos que se pueden agregar en el editor, aunque también se brinda la opción de poder seleccionarlos del menú. El panel de propiedades está diseñado para ocultar información que no pertenezca a un elemento seleccionado
Flexibilidad y eficiencia de uso	La forma en que está diseñada la ventana principal de la aplicación permite acceder a las tareas importantes que es cumplir con el objetivo de editar un escenario, permitiéndole a un usuario experto o novato poder interactuar con ella. La aplicación proporciona navegación por teclado como el poder eliminar un elemento y deseleccionarlo. La aplicación tener atajos hacía los paneles, pues en caso de que el usuario los cierre se puedan agregar desde el menú View.

Estética y diseño minimalista	La aplicación nos muestra la información considera más importante, como los mensajes que explican algunas características que ha simple vista no se saben que pueden ser, como por ejemplo el seleccionar un elemento como Custom Class o asignar un Tag. Disminuye la visibilidad relativa al dejar que los elementos seleccionados al área de dibujo solo aparezca el nombre en diálogos cuando se selecciona el icono, permitiendo que el usuario no se distraiga. Se clasifica el panel de propiedades dependiendo del elemento seleccionado con ello no aparece información repetida y los usuarios pueden acceder de inmediato a lo que quieren, de igual forma cuando se listan los elementos que han sido agregados al área de dibujo. Se decidió aprovechar la tecnología provista por Qt para adaptarse al entorno donde se esta ejecutando, y así obtener una apariencia uniforme de acuerdo a la configuración que el usuario haya hecho en el sistema en general.

Tabla 7 Pruebas de usabilidad. Fuente: Creación del autor

11. DISCUSIÓN DE RESULTADOS

Este trabajo tuvo como propósito investigar y desarrollar una herramienta libre multiplataforma, que facilitará la codificación de escenarios para videojuegos 2D y permitiera generar el código necesario para el Framework Cocos2dx. Además, se identificaron factores asociados para hacer una interfaz gráfica de usuario usable y sobre qué sistemas operativos son más utilizados en el mercado actualmente. En este sentido, a continuación se discuten los principales hallazgos en el proceso de desarrollo de esta herramienta.

Antes de iniciar con la implementación del editor, se llevó a cabo una serie de investigaciones que tomaron un tiempo considerable que hacen parte de las fases de análisis y modelado, propuesta dentro de la metodología de investigación, y que son de suma importancia debido a que en primer lugar se estudió sobre los elementos de usabilidad que tienen las interfaces gráficas de usuario con el fin facilitarle al usuario el manejo de la aplicación. Como segundo punto se compararon los Framework's existentes en el mercado y que están orientados a facilitar el desarrollo de un videojuego 2D, identificando así sus características, ventajas y desventajas. Por último, se listaron los principales sistemas operativos para decidir sobre cuales se generarían los ejecutables de la herramienta.

Como se puede observar a lo largo del documento en la fase de modelado, se justificó la selección de cada uno de los puntos descritos anteriormente considerando los elementos de usabilidad, definidos por los principios de heurística, permitiendo la edición rápida de un escenario. Para el Framework seleccionado, Cocos2dx, se tomó base el que ofreciera mejores prestaciones y de igual forma para los sistemas operativos, basados en estadísticas de los más utilizados en el mercado.

Durante el desarrollo de la herramienta, para garantizar el almacenamiento del escenario creado por el usuario, se utilizó JavaScript para la creación de la estructura de un JSon, haciendo uso de las prestaciones ofrecidas por el Framework Cocos2dHTML5 sobre el cual trabaja la aplicación en el área de dibujo, por lo que gracias a ello, puede ser llamado por el núcleo de la aplicación que permite la comunicación con C++ a través de Qt Creator.

Por otro lado, gracias a la estructura que se dio al producto, es posible extender funcionalidades (Plugins) en caso de que un usuario se sienta más a gusto con otro Framework de desarrollo, dejando abierta la posibilidad de soportar muchos más. Para hacer esto posible, se hizo uso de la tecnología ofrecida por Qt y se probaron los archivos generados en un proyecto de ejemplo ofrecido por el Framework sin tener que hacer muchas modificaciones para incluirse.

Por último, a la hora de generar los diversos ejecutables de la aplicación para los sistemas operativos seleccionados, se tomó un tiempo importante en los procesos de instalación y pruebas, especialmente para los sistemas operativos Mac y Linux, donde se presentaron inconvenientes con las librerías necesarias para su ejecución, por lo que se optó, como recomendación al usuario final, tener instalado Qt creator, para Microsoft Windows lo único necesario era agregar todas las librerías en una correspondiente carpeta que se entregaría al usuario.

12. CONCLUSIONES

Las herramientas disponibles en la actualidad para la edición de escenarios para videojuegos 2D, son muy diversas y variadas en funcionalidades pero no están pensadas para diferentes tipos de usuarios interesados, debido a que carecen de características como ser multiplataforma y libres.

Un editor de escenarios acelera el proceso de desarrollo de video juegos al igual que de aplicaciones interactivas en general, disminuyendo los requerimientos de conocimiento y habilidades técnicas por parte de los diseñadores, lo cual permite dividir los equipos de desarrollo y optimizar el proceso.

Una adecuada interfaz de usuario es muy importante en herramientas para el desarrollo de aplicaciones en general, y través de los principios de heurística, el usuario interactúa con dichas herramientas con menos dificultad.

Los Frameworks especializados en el desarrollo de videojuegos 2D, especialmente los que tienen licencias libres, brindan muchos beneficios al usuario para una correcta terminación y permitirle abarcar nuevos problemas.

La biblioteca multiplataforma Qt usada para el desarrollo de la herramienta, permitió acelerar el proceso de diseño de la interfaz gráfica de usuario basada en Widgets Qt con el editor integrado, Qt Designer y unas de las mayores ventajas es desplegar la aplicación a múltiples sistemas operativos con una herramienta común para desarrollo y depurado.

13. BIBLIOGRAFÍA

[1] Gamedev tut+, "Make Your Life Easier: Build a Level Editor," Abril 2013. [En línea]. Disponible: http://gamedev.tutsplus.com/articles/workflow/make-your-life-easier-build-a-level-editor/

[2] KXPC, "How to: the Video Game Development Process", Mayo 2013, [En línea]. Disponible: http://www.kxpc.com/technology/how-to-the-video-game-development-process.html

[3] Record, "Los Videojuegos, la nueva mina de oro", Mayo 2013. [En línea]. Disponible: http://www.record.com.mx/circo/2011-11-27/los-videojuegos-la-nueva-mina-de-oro

[4] Enter.co, "La nueva división de la industria de videojuegos", Mayo 2013. [En línea]. Disponible: http://www.enter.co/otros/la-nueva-division-de-la-industria-de-videojuegos/

[5] Soy Periodista, "Crecimiento de la Industria de Videojuegos en Colombia," Mayo 2013. [En línea]. Disponible: http://www.soyperiodista.com/tecnologia/nota-17255-crecimiento-de-la-industria-de-videojuegos-colombi

[6] Marmalade, "Bringing Innovation to Multiplatform Development and Deployment", Mayo 2013. [En línea]. Disponible: http://www.madewithmarmalade.com/jp/node/50477

[7] CocosBuilder, Mayo 2013. [En línea]. Disponible: http://cocosbuilder.com/

[8] Unity3D, Mayo 2013. [En línea]. Disponible: http://unity3d.com/

[9] GameSalad, Mayo 2013. [En línea]. Disponible: http://gamesalad.com/

[10] RPG Maker, Mayo 2013. [En línea]. Disponible: http://www.rpgmakerweb.com/

[11] YOYOGAMES, Mayo 2013. [En línea]. Disponible:
http://www.yoyogames.com/gamemaker/studio

[12] Game Dev Helper, Mayo 2013. [En línea]. Disponible:
http://www.gamedevhelper.com/levelhelper/

[13] Definición.DE, Mayo 2013. [En línea]. Disponible:
http://definicion.de/niveles/

[14]Gamma,E.Helm,R.Johnson,R.Vlissides,J.DesignPatterns:Elements of Reusable Object-Oriented Software. Addison-Wesley Professional Computing Series.

[15] Hourences, Mayo 2013. [En línea]. Disponible:
http://www.hourences.com/the-hows-and-whys-of-level-design-about/

[16] ASOCIACIÓN DESARROLLADORES WEB DE ESPAÑA, Mayo 2013.[En línea]. Disponible:
http://www.adwe.es/general/conferencias-adwe-frameworks-para-el-desarrollo-de-juegos-para-dispositivo-moviles

[17] Mario Lorenzo Alcala and Jesus Maria Minguet Melia, "Medida de la usabilidad en aplicaciones de escritorio un método práctico", Curso: 2006-07,
p 17.

[18] Usability Net, International standards for HCI and usability. Septiembre 2013. [En línea]. Disponible:
http://www.usabilitynet.org/tools/r_international.htm

[19] Pilar Hernández, "La interfaz del objeto de aprendizaje",Diseño de una metodología para la construcción de objetos de aprendizaje desde una perspectiva constructivista.

[20] A Dix; J. Finlay; G. Abowd; R. Beale (1993). Human-computer Interaction, Londres: Prentice Hall. P 118.

[21] J. Nielsen. Usability Engineering. AP Professional, 1993.

[22] Toni Granollers i Saltiveri; Jesús Lorés Vidal. "Diseño de sistemas interactivos centrados en el usuario", 2005 Editorial UOC, p 177.

[23] Mario Lorenzo Alcala and Jesus Maria Minguet Melia, "Medida de la usabilidad en aplicaciones de escritorio un método práctico", Curso: 2006-07,
pp. 32-40

[24] Unreal Engine, "Features" Septiembre 2013. [En línea]. Disponible:
http://www.unrealengine.com/en/features/

[25] Cocos2DX, Septiembre 2013. [En línea]. Disponible:
http://www.cocos2d-x.org/features

[26] Cocos2DX, Septiembre 2013. [En línea]. Disponible:
http://www.cocos2d-x.org/wiki/Getting_Started_with_Cocos2d-html5

[27] CoronaLabs, Septiembre 2013. [En línea]. Disponible:
http://www.coronalabs.com/products/corona-sdk/

[28] LibGDX, Septiembre 2013. [En línea]. Disponible:
www.libgdx.badlogicgames.com/features.html

[29] Marmalade, Septiembre 2013. [En línea]. Disponible:
https://www.madewithmarmalade.com/sdk

[30] Allegro, Septiembre 2013. [En línea]. Disponible:
http://alleg.sourceforge.net/readme.html

[31] Scirra, Septiembre 2013. [En línea]. Disponible:
http://www.scirra.com/construct-classic

[32] Unity, Septiembre 2013. [En línea]. Disponible:
http://spanish.unity3d.com/unity/

[33] GarageGames, Septiembre 2013. [En línea]. Disponible:
https://www.garagegames.com/products/torque-2d

[34] YoyoGames, Septiembre 2013. [En línea]. Disponible:
http://www.yoyogames.com/studio

[35] DX studio, Septiembre 2013. [En línea]. Disponible:
http://www.dxstudio.com/index.aspx

[36] Zenca.es, Ranking Sistemas Operativos en el Mundo 2013. Septiembre 2013. [En línea]. Disponible:http://zenka.es/mercado-sistemas-operativos/

[37] netmarketshare, Desktop Top Operating System Share Trend. Septiembre 2013. [En línea]. Disponible: http://www.netmarketshare.com/

[38] qlip, 10 Usability Heuristics. Septiembre 2013. [En línea]. Disponible: http://qlip.in/article/10-usability-heuristics

[39] Toni Granollers, i Saltiveri; Jesús Lorés Vidal. "Diseño de sistemas interactivos centrados en el usuario", 2005 Editorial UOC, pp. 181-187.

[40] Cocos 2Dx,"A trip from cocos2D-iphone to cocos2D-html5", Septiembre 2013. [En línea]. Disponible: https://dl.dropboxusercontent.com/u/30871449/Website/Resource/2013/04/ClearFiveStages-Share/A%20trip%20from%20cocos2d-iphone%20to%20cocos2d-html5.pdf

[41] Linux, Valve's SteamOS and Steam Machine News Backs Linux as the Future of Gaming. Septiembre 2013. [En línea]. Disponible: https://www.linux.com/news/software/applications/740038-valves-steamos-and-steam-machine-news-backs-linux-as-the-future-of-gaming

[42] Redalyc.org, Criterios de selección de metodologías de desarrollo de software. Octubre 2013. [En línea]. Disponible: http://www.redalyc.org/articulo.oa?id=81619984009

[43] SECC, Metodologías ágiles. Octubre 2013. [En línea]. Disponible: www.seccperu.org/files/Metodologias Agiles.pdf

[44] Uvadoc, Guía Comparativa de Metodologías Ágiles. Octubre 2013. [En línea]. Disponible: http://uvadoc.uva.es/bitstream/10324/1495/1/TFG-B.117.pdf

[45] Frogtek, Evolucionando hacia Scrumban. Octubre 2013. [En línea]. Disponible: http://developing.frogtek.org/2010/12/22/evolucionando-hacia-scrumban/