PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

CONSTRUCCIÓN DE UN EDITOR DE MAPAS PARA EL JUEGO "1814: LA REBELIÓN DEL CUZCO"

Tesis para optar el Título de Ingeniero Informático, que presenta el bachiller:

Luis Ignacio Landa Torrejón

ASESORA: Mag. Claudia Zapata Del Río

Lima, Mayo 2017

Dedicatoria

A mis padres, quienes desde el principio de la carrera me apoyaron tanto económica como emocionalmente. Ellos han sido mis principales aliados en este largo viaje que ahora termina.

A mis hermanos, que siempre traen distintas perspectivas a mi vida y me impulsan a seguir mejorando como persona y como profesional.

A mis amigos y profesores de la universidad, que hicieron que mi vida universitaria fuera más entretenida y productiva. Gracias a su ayuda he llegado tan lejos.

Agradecimientos

Un agradecimiento especial a mi grupo de trabajo en el curso de Lenguajes de Programación 2, pues del proyecto del curso nació la idea de esta tesis y gracias a ellos logramos hacer un buen trabajo y presentarlo al grupo Avatar.

También agradecer al grupo Avatar, que me permitió desarrollar este proyecto para su videojuego y me ayudó a gestionar los avances.

Además, no puede faltar un agradecimiento a mi asesora, Claudia Zapata. Sin su apoyo y orientación no habría logrado sustentar de forma tan sólida el proyecto de fin de carrera.

Resumen

En el 2014, el grupo de investigación AVATAR lanzó su juego de estrategia en tiempo real “1814: La rebelión del Cuzco”, y ahora desean expandir el contenido del juego con nuevos mapas y niveles. Sin embargo, crear contenido de este tipo requiere iniciar un ciclo de desarrollo extenso y costoso, el cual produce una cantidad limitada de mapas.

Frente a esta necesidad, se propone un editor de mapas para el juego. Una herramienta que permite crear una cantidad ilimitada de mapas y, a diferencia de un algoritmo de generación de niveles, aprovecha la creatividad de los jugadores para generar nuevo contenido e interés en el juego.

El editor se construyó siguiendo la estructura de un compilador que recibe como entrada un lenguaje visual. Se elaboró primero una interfaz gráfica donde se edita el mapa de forma intuitiva, el cual luego es procesado por el compilador y traducido a una estructura de datos interpretable por el motor de juego. Por último, se realizó una prueba con usuarios para validar la usabilidad del editor y el correcto funcionamiento de los mapas editados.

Al finalizar el proyecto de fin de carrera se logró construir satisfactoriamente un editor de mapas para “1814: La rebelión del Cuzco”, capaz de ser utilizado, tanto por jugadores expertos como casuales, para crear fácilmente nuevos mapas y luego integrarlos al juego.

FACULTAD DE
CIENCIAS E INGENIERÍA
ESPECIALIDAD DE
INGENIERÍA INFORMÁTICA

PONTIFICIA
UNIVERSIDAD CATÓLICA
DEL PERÚ

TEMA DE TESIS PARA OPTAR EL TÍTULO DE INGENIERO INFORMÁTICO

TÍTULO: **Construcción de un editor de mapas para el juego "1814: La rebelión del Cuzco"**

ÁREA: Ciencias de la computación
ASESOR: Mag. Claudia María del Pilar ZAPATA DEL RÍO
ALUMNO: Luis Ignacio LANDA TORREJON
CÓDIGO: 20100114
TEMA N°: #651
FECHA: San Miguel, 9 de 11 de 2016

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
Facultad de Ciencias e Ingeniería
2 8 NOV. 2016
MIGUEL MEJIA PUENTE
DECANO

DESCRIPCIÓN

En el 2014, el grupo AVATAR, perteneciente a la Pontificia Universidad Católica del Perú, lanzó su juego de estrategia en tiempo real "1814: La rebelión del Cuzco", en el cual se narra los eventos sucedidos durante el suceso histórico del mismo nombre. El grupo ahora desea agregar nuevos mapas al videojuego para incrementar su tiempo de jugabilidad y mantener el interés de los usuarios, sin embargo, el tiempo y los recursos para iniciar un nuevo ciclo de desarrollo sólo permitirían crear un conjunto de contenidos fijos para el juego.

Varias casas de desarrollo de videojuegos han enfrentado este problema desde distintas perspectivas, una de ellas busca aprovechar la creatividad de los jugadores interesados en el juego: desarrollar un editor de niveles y ponerlo a la disposición del público. No obstante, este enfoque usualmente requiere que el editor se desarrolle en conjunto con el juego para facilitar la interacción con su lógica interna, lo que también ocasiona que no exista una herramienta genérica con las características mencionadas.

Considerando este enfoque, el problema consiste en permitir la generación de nuevos niveles para el juego de "1814: La rebelión del Cuzco", de acuerdo a las preferencias y disposición de elementos seleccionada por los usuarios. Además, es menester que los usuarios puedan crear nuevos niveles sin la necesidad de poseer conocimientos técnicos acerca del juego o lenguajes de programación, pues está actividad busca impulsar el interés en el juego y permitir a los jugadores ejercitar su creatividad.

Ante la problemática planteada se propone el desarrollo de una herramienta informática que permita la creación de niveles para el juego descrito, basada en los principios y técnicas utilizados para el diseño de compiladores y empleando una interfaz gráfica en conjunto con un lenguaje visual para interactuar con el usuario.

OBJETIVO GENERAL

Implementar un editor de mapas para generar niveles del juego "1814: La rebelión del Cuzco", considerando el diseño planteado por un usuario. Este permitirá traducir la información visual a la estructura de datos que el motor del videojuego utiliza.

OBJETIVOS ESPECÍFICOS

Los objetivos específicos son:
OE1. Elaborar una interfaz gráfica que permita diseñar mapas sin la necesidad de conocimientos técnicos acerca del juego o programación.
OE2. Desarrollar un compilador que convierta el diseño creado en el editor a una estructura de datos con el formato definido en el juego "1814: La rebelión del Cuzco".
OE3. Validar el uso del editor de mapas para diseñar un mapa que se ejecute con el motor del videojuego "1814: La rebelión del Cuzco".

ALCANCE

La herramienta implementada en el proyecto permitirá la edición y generación de nuevos mapas para el juego de estrategia en tiempo real "1814: La rebelión del Cuzco" mediante el uso de un entorno visual; esta no será una herramienta genérica, compatible con todos los juegos del mismo género, debido a la diversidad de lógicas y estructuras de videojuegos presentes en la industria.

Además, para facilitar el diseño gráfico de los mapas, el entorno visual permitirá a los usuarios reutilizar recursos gráficos utilizados en el juego y colocarlos en el mapa de acuerdo a sus preferencias. Sin embargo, la herramienta no permitirá al usuario insertar nuevas imágenes en el editor o modificar imágenes existentes para mantener la compatibilidad con el motor del juego.

Las funcionalidades principales de la herramienta desarrollada incluirán: edición del terreno en el mapa, disposición de unidades y edificios en el mapa, modificación del

tamaño del mapa a editar y la compilación de todo el contenido editado en el entorno visual a una estructura de datos soportada por el motor del juego "1814". Funcionalidades adicionales incluidas en editores comerciales, como la inclusión de scripts para crear eventos personalizados o la modificación de unidades existentes, no serán consideradas dentro del alcance del proyecto pues son características propias a un motor de juego.

Máximo: 100 páginas

Contenido

Figuras

Tablas

CAPÍTULO 1: GENERALIDADES

A continuación se presenta una introducción al proyecto de fin de carrera, incluyendo la problemática a la que responde, una revisión del estado del arte y los objetivos que se busca alcanzar.

1.1 PROBLEMÁTICA

Desde los inicios de la industria de los videojuegos ha existido el desafío de mantener el interés de los jugadores a través de un período de tiempo largo. Un videojuego que logra afrontar este reto satisfactoriamente puede representar una mejor compra para los usuarios, al brindar más contenido o tiempo de diversión por el dinero invertido en él, característica importante para ganar una ventaja competitiva en el mercado (Tassi, 2015).

Para incrementar el tiempo del videojuego se han desarrollado distintos mecanismos y estrategias. Las más clásicas involucran desarrollar una cantidad de contenido (gráficos, código, sonidos e historia) proporcional al tiempo que se desea alcanzar (Snow, 2011). Generar más contenido implica mayores costos en la producción del videojuego y los riesgos en caso el producto no sea exitoso se incrementan, dificultando las inversiones en el mismo.

Otra forma de incrementar el tiempo de juego está asociado al concepto de *rejugabilidad* o "*replay value*": el potencial del videojuego para brindar entretenimiento después de su completitud (Wolf, 2012). Por ejemplo, el desbloqueo de una nueva modalidad de juego o dificultad después de completarlo o la inclusión de un modo multijugador, permitiendo competir o cooperar con otros jugadores. La ventaja de este concepto y sus posteriores implementaciones es que añaden tiempo de juego sin requerir una inversión tan grande en desarrollo de contenido, pues se reúsan recursos aplicados previamente.

Sin embargo, algunas casas de desarrollo de videojuegos adoptaron el concepto de *rejugabilidad* desde una perspectiva completamente distinta: brindar una herramienta integrada a su juego que permita a los jugadores crear nuevos niveles, reutilizando los componentes existentes, y compartirlos con otros usuarios. Este tipo de herramienta se conoce ahora como un editor de niveles o mapas, dependiendo del género de videojuego al que se aplica. Algunos ejemplos de este enfoque se evidencian en los juegos Starcraft ("Blizzard Entertainment," 2015) y Age of Empires 1 y 2 ("Age of Empires II HD," 2015), que incluyen dentro de su contenido un editor de mapas.

Los editores de niveles establecen un marco para la construcción y divulgación de contenido generado por usuarios o UCC (*user created content*), para el videojuego al que pertenecen (Graft, 2012). Esto permite al estudio de desarrollo expandir el contenido del juego, incrementar la rejugabilidad e impulsar el interés en el mismo, sin la necesidad de invertir en nuevos niveles o recursos, además de aprovechar la creatividad de sus jugadores (Graft, 2012).

Pese a sus ventajas, un editor de mapas no siempre es una estrategia viable para impulsar la popularidad y llegada de un videojuego, esto depende estrechamente de su género y el diseño preliminar de la herramienta (Reilly, 2012). No obstante, si sus creadores comprenden las características clave del videojuego frente a los jugadores e implementan el editor basado en este conocimiento, el incremento de popularidad e interés puede llegar a ser sustancial, tal como lo demuestra el éxito de Sony con su juego LittleBigPlanet, un "*platformer*" (juego que involucra mover un personaje por una serie de plataformas y obstáculos) que en el 2012 logró alcanzar 6.7 millones de niveles creados por su comunidad desde su lanzamiento en el 2008 (Dumitrescu, 2012).

En el 2014, el grupo AVATAR, perteneciente a la Pontificia Universidad Católica del Perú, lanzó su juego de estrategia en tiempo real (RTS, por sus siglas en inglés) "1814: La rebelión del Cuzco", en el cual se narran los eventos sucedidos durante el suceso histórico del mismo nombre, llevada a cabo por los hermanos Angulo y Mateo Pumacahua en 1814 (B. Guerra, 2014). Ahora se desea agregar nuevos mapas al juego, pero no se tienen los recursos y el personal para asignar esta tarea.

Teniendo en cuenta esta necesidad, se plantea la siguiente pregunta que direccionará el proyecto, ¿cómo se puede desarrollar una herramienta que permita desarrollar nuevos mapas, siguiendo las preferencias del usuario, para el juego de "1814: La rebelión del Cuzco"? Basado en lo explicado anteriormente, se podría sostener que esta pregunta ya tiene una respuesta: un editor de mapas adaptado para juegos de RTS, es más, ya existen juegos de este género en el mercado actual que poseen un editor de mapas integrado. No obstante, existen dos grandes problemas relacionados a estos editores, los cuales se exponen a continuación.

No se encontró en el mapeo sistemático de la literatura editores de mapas o herramientas para construir un nivel de un juego RTS genéricas. Es decir, cada editor de mapas que se encuentra disponible en el mercado fue desarrollado en conjunto con su respectivo videojuego y está diseñado para interactuar solo con este. Lo que implica que, si bien es verdad que existen herramientas para la

construcción de niveles o mapas, ninguna de ellas se puede integrar al juego de “1814: La rebelión del Cuzco”.

El segundo problema radica también en la idea de que un editor de mapas se desarrolla en conjunto con su videojuego respectivo. Fuera de las similitudes del género RTS, cada videojuego tiene su propia estructura lógica y método para almacenar las unidades dispuestas en el mapa. Considerando que “1814: La rebelión del Cuzco” ya cerró su ciclo de desarrollo y no se conoce su funcionamiento interno, ¿cómo puede la herramienta o software propuesto interactuar con la estructura del juego? El problema consistirá en traducir o compilar de un lenguaje que el usuario pueda entender a otro que el juego interprete correctamente.

Entonces, el traductor requerido para satisfacer la necesidad de crear nuevos mapas deberá establecer un lenguaje que pueda ser fácilmente interpretado y utilizado por los jugadores, validar los elementos y estructuras dispuestas en el mapa basado en las reglas del juego en tiempo real, y luego compilarlo en una estructura de datos capaz de ser descifrada por la lógica del juego.

En resumen, el problema principal consiste en permitir la generación de nuevos niveles para el juego de “1814: La rebelión del Cuzco”, de acuerdo a las preferencias y disposición de elementos seleccionada por los usuarios. Además, es menester que los usuarios puedan crear nuevos niveles sin la necesidad de poseer conocimientos técnicos acerca del juego o lenguajes de programación, pues esta actividad busca impulsar el interés en el juego y permitir a los jugadores ejercitar su creatividad.

Ante la problemática planteada se propone el desarrollo de una herramienta informática que permita la creación de niveles para el juego descrito, basada en los principios y técnicas utilizados para el diseño de compiladores y empleando una interfaz gráfica en conjunto con un lenguaje visual para interactuar con el usuario.

1.2 SOLUCIONES ALTERNATIVAS

El enfoque que utiliza un editor de niveles para generar nuevo contenido para un videojuego presenta muchos beneficios, como se mencionó previamente. Sin embargo, existen otras soluciones a esta problemática que, dada las circunstancias, podrían ser aún más ventajosas.

Una solución es iniciar un nuevo ciclo de desarrollo de contenido para el videojuego existente, dando como resultado una expansión del mismo. Esta opción implica un uso intensivo de recursos humanos para los recursos gráficos, sonoros y el desarrollo o modificación de funcionalidades del juego. La ventaja de esta solución es la capacidad de controlar la calidad del producto final, a diferencia de un editor de mapas donde cualquier usuario tiene completa libertad sobre el nivel que elabora.

Otra solución posible es implementar un algoritmo que, bajo ciertas reglas, genere automáticamente los mapas para el juego y las posiciones de las unidades. El principal desafío está en lograr que el algoritmo produzca mapas que tengan sentido en el contexto del juego y mantengan una progresión lógica en su dificultad, lo cual es aún más complicado para los juegos de estrategia en tiempo real. Sin embargo, pese a su alta complejidad, esta solución supone la posibilidad de generar mapas ilimitados para el videojuego, aumentando sustancialmente su rejugabilidad.

Tomando en cuenta estas soluciones, se eligió realizar un editor de mapas por el siguiente motivo: el grupo AVATAR desea incluir a los jugadores en el proceso creativo de diseñar nuevos mapas para el juego, permitiendo la generación de contenido sin limitaciones a lo largo del tiempo. Un nuevo ciclo de desarrollo para una expansión del juego y el algoritmo de generación automática de mapas no involucran a los jugadores en el proceso de elaboración de mapas y en el caso de la expansión, esta solo produce un contenido estático para el videojuego.

1.3 MARCO CONCEPTUAL

Como se mencionó en la sección anterior, el problema que busca solucionar el proyecto de fin de carrera consiste en crear nuevos mapas o niveles para el juego "1814: La rebelión del Cuzco" de acuerdo a configuraciones establecidas por un usuario.

Siguiendo está definición, es válido enfocar el problema como una traducción desde un lenguaje fuente, que permita al usuario diseñar un nivel, hacia una estructura de datos capaz de ser descifrada por el motor del juego. El enfoque descrito coincide con el concepto de un compilador, cuya estructura posee una serie de pasos que es importante conocer para comprender el proceso de traducción y cómo se adapta al caso.

Otro aspecto importante acerca del enfoque del problema como una traducción es el lenguaje fuente, la herramienta que el usuario utilizará para diseñar el nivel. Para esto es relevante entender el concepto del contenido generado por usuarios, el cual nos indica hacia qué público está dirigido el lenguaje y que características debe poseer para fomentar el uso del mismo. Asociado a esto se encuentran los lenguajes de programación visual, los cuales proveen una orientación para diseñar el lenguaje fuente.

Cabe resaltar que el proceso de traducción que implica la problemática se desenvuelve en un contexto específico: un videojuego de estrategia en tiempo real. Será entonces necesario aclarar este tipo de videojuego y comprender que elementos lo conforman para definir las configuraciones o preferencias disponibles para el usuario en el diseño del nivel.

A continuación, se detallan los conceptos mencionados previamente: el compilador y sus fases aplicables al problema, el contenido generado por usuarios, los lenguajes de programación visual y los juegos de estrategia en tiempo real.

1.3.1 Compilador

En su esencia, el problema del proyecto consiste en establecer un lenguaje que personas, sin conocimientos de programación o la estructura del juego, puedan usar para diseñar un mapa y luego traducirlo al lenguaje que utiliza el motor del juego.

Esto calza con la definición de un compilador: un programa que lee otro programa escrito en un lenguaje, llamado lenguaje fuente, y lo traduce a un programa

equivalente en otro lenguaje, denominado lenguaje objeto, notificando al usuario acerca de errores durante el proceso de traducción (Aho, Sethi, Ullman, Flores Suárez, & Botella i López, 1990).

La estructura de un compilador, además, se divide en dos etapas generales según Aho et al. (1990):

- **Análisis:** como su nombre lo sugiere, en esta etapa el programa analiza el código fuente desde tres aspectos: léxico (análisis lineal de caracteres, construcción de tokens/palabras), sintáctico (agrupar tokens/palabras en frases gramaticales con sentido) y semántico (verificación de tipos en las frases construidas). Luego, si no se encontró ningún error, construye en código intermedio que será utilizada en la siguiente etapa.

- **Síntesis:** recibe como entrada el lenguaje intermedio, lo optimiza de modo que sea más rápido de ejecutar y lo convierte al lenguaje objeto.

1.3.2 Contenido Generado por Usuarios (UCC)

El UCC (*user created content*) se define como cualquier contenido que puede ser publicado libremente en Internet, requiere de cierto nivel creativo y se desarrolla fuera de un ámbito o prácticas profesionales (Vickery, Wunsch-Vincent, & Organisation for Economic Co-operation and Development, 2007). Este se realiza por varios motivos, no necesariamente ligados a una ganancia monetaria: libre expresión y exploración creativa, conectar con otras personas a partir del contenido y alcanzar un cierto nivel de fama o reconocimiento a través del medio (Vickery et al., 2007).

La solución planteada para la problemática debe considerar las características del UCC, para el diseño de su interfaz o forma de interacción con los usuarios finales, pues se busca con ella facilitar la generación de este tipo de contenido, específicamente para el juego de “1814: La rebelión del Cuzco”.

1.3.3 Lenguajes de Programación Visual

Parte del desafío que presenta el problema de generar nuevos niveles para el juego de “1814” es permitir que usuarios fuera de un contexto profesional, bajo la definición del UCC, puedan diseñar fácilmente mapas. El enfoque planteado para entender este desafío gira en torno a los lenguajes de programación visual y como simplifican el proceso de diseñar un mapa.

Un lenguaje visual es una representación pictórica de conceptos y operaciones, el cual es usado para armar sentencias visuales y representar una secuencia de acciones o un concepto de mayor orden de complejidad (Chang, 1999). Entonces, la programación visual consiste en programar utilizando medios visuales, usualmente agrupando íconos que representan componentes básicos de código para generar un programa (Chang, 1999).

La ventaja de la programación visual es que resulta más fácil para los usuarios construir un programa por medio de una interfaz gráfica y bloques de código predefinidos, además de facilitar el análisis léxico y sintáctico al limitar las acciones permitidas por medio de la interfaz (Grigorenko, Saabas, & Tyugu, 2005).

1.3.4 Videojuego de estrategia en tiempo real (RTS)

Un juego de estrategia en tiempo real o RTS (*Real Time Strategy*) es una aplicación interactiva en la que los jugadores deben competir por recursos, expandir su economía para soportar una producción estable de unidades, dominar zonas estratégicas del mapa y eliminar a los demás jugadores (Ballinger & Louis, 2013). Es decir, consiste en gestionar distintas unidades y edificios de forma eficiente para vencer a la competencia, usualmente en un contexto militar, sin ningún tipo de sistema de turnos, visto en otro tipo de juegos como ajedrez o damas.

Bajo este contexto, un mapa se refiere a la configuración geográfica en la que se desarrolla el juego, este incluye varios elementos vitales para el inicio de la partida: disposición de elementos neutrales (montañas, vegetación y edificios neutrales) y disposición de recursos de distintos tipos, la ubicación inicial de cada jugador en el mapa. Además, en escenarios especiales algunos jugadores pueden tener unidades o edificios posicionados en el mapa desde el inicio.

En la Figura 1 se muestra un ejemplo de un juego RTS comercial, Starcraft 1 (“Blizzard Entertainment,” 2015). Se pueden observar los elementos descritos previamente de forma más clara: las unidades, que dependiendo del color pertenecen a un jugador o al otro, los edificios y la configuración geográfica, que en esta pantalla es plana pero tiene distintos tipos de terreno en ella: tierra de color azul, superficie rocosa de color gris y piso laminado de color verde azulado oscuro. Es importante mencionar el elemento ubicado en la esquina inferior izquierda, una versión pequeña del mapa real denominada “minimapa”, característica indispensable de este género de videojuegos.

Figura 1 Ejemplo de un juego de RTS. Imagen recuperada de Blizzard Entertainment, 2015.
Fuente: http://us.blizzard.com/en-us/games/sc/

1.4 ESTADO DEL ARTE

A continuación se detallará la metodología utilizada para realizar el mapeo sistemático del estado del arte para el proyecto de fin de carrera. Se empleó la herramienta StArt (State of the Art through Systematic Review) v2.3 para orientar el proceso.

Al final del mapeo se espera conocer los últimos avances y técnicas utilizadas para hacer compiladores para programas que interactúan con los usuarios mediante lenguajes visuales, sin importar el lenguaje objeto de cada estudio. Además, conocer y entender las últimas implementaciones de editores de mapas o niveles, en el contexto de videojuegos, en los años recientes.

1.4.1 Preguntas de Revisión

Pregunta principal: ¿Cuáles son los últimos avances en la construcción de compiladores para programas que utilizan lenguajes visuales para la interacción con el usuario?

Pregunta secundaria: ¿Cuáles son los últimos avances en editores de mapas o niveles para videojuegos, en un contexto académico?

1.4.2 Palabras Clave y Cadenas de Búsqueda

Las palabras para realizar la búsqueda en la base de datos seleccionada se derivaron de las preguntas de revisión establecidas anteriormente siguiendo la estructura de población, intervención, comparación y resultados propuesta por Kitchenham & Charters (2007):

- Población: siguiendo las preguntas, se tienen dos poblaciones distintas. Por un lado compiladores y por el otro editores de niveles.
- Intervención: para los compiladores, se buscan todos los artículos que utilicen metodologías basadas en lenguajes visuales.
- Comparación y resultados: en el mapeo sistemático no se buscó una comparación con una metodología o herramienta en específico. Tampoco es parte de las preguntas de investigación un resultado en especial, pues se busca un panorama más general del estado del arte.

Palabras Clave: "Game Engine" "Level Editor" "Visual Language" "Visual Programming Language" “Compiler” “Editor”

Cadena de búsqueda: (Compiler AND “Visual language”) OR ((“Game engine” AND “Editor”) OR “Level editor”) AND

Nivel de búsqueda: Título, resumen y palabras clave

Fuentes: Base de datos indexada Scopus

1.4.3 Criterios de Inclusión

Los criterios utilizados para determinar la inclusión de un artículo al mapeo:

- El artículo detalla la implementación o mejora de un compilador que recibe un lenguaje visual como fuente.
- El artículo explica el diseño y/o construcción de un editor de niveles o mapas, para cualquier género de videojuego.

1.4.4 Criterios de Exclusión

Por otro lado, se siguieron los siguientes criterios para excluir artículos:

- El artículo tiene una antigüedad mayor a los 10 años. Esto se debe al ritmo acelerado al que avanzan las tecnologías asociadas a videojuegos y su usabilidad. Revisar artículos más antiguos podría dar resultados obsoletos en el presente año.
- El artículo explica la aplicación de un motor de juego y no detalla un editor de niveles o mapas, incluyendo su diseño.
- El artículo menciona una herramienta o programa que utiliza programación con lenguajes visuales más no ahonda en el compilador asociado.

1.4.5 Extracción de Datos

Para extraer la información relevante al mapeo de los artículos seleccionados se utilizó el formulario mostrado en el Anexo A. Para los estudios acerca de compiladores se buscó conocer en que dominio se aplicó y que metodologías utilizó. Por otro lado, en el caso de artículos relacionados a editores para videojuegos se hizo énfasis en sus características principales o innovadoras.

1.4.6 Resultados de la Búsqueda

La ejecución de la búsqueda planificada en la base de datos Scopus dio como resultado 87 artículos, de los cuales 3 eran repetidos, 66 fueron rechazados y 18 fueron seleccionados en base a una revisión del resumen. Después de ejecutada una verificación de texto completo, 16 trabajos fueron aceptados y 2 excluidos por carecer de profundidad en los temas investigados.

En la Tabla 1 se muestra una tabla con el resumen de los artículos analizados en la revisión sistemática, separados por la pregunta a la que dan respuesta y categorizados según los temas predominantes en su contenido. Es importante aclarar que 2 investigaciones pertenecen a 2 categorías cada una, por lo que la suma en la columna final da como resultado 18 artículos, menos 2 repetidos.

Tabla 1 Resultados de la revisión sistemática. (Elaboración propia)

Tabla de Resultados			
Pregunta	Categoría	Subcategoría	Artículos
Avances recientes en compiladores visuales	Lenguajes visuales específicos de dominio	Librerías digitales	1
		Computación ubicua	1
		Motor de juego	1
		Reglas armónicas y musicales	1
		Análisis Semántico	1
	Meta programación	Compiladores de compiladores visuales	2
		Meta lenguajes visuales	3
	Diagrama de Flujo de Datos	Programación en paralelo	2
		Flujo y control de datos	2
Avances recientes en editores de mapas	Editores en 2D	Realidad Aumentada	1
	Editores en 3D	Genéricos	1
		No-genéricos	2

La revisión dio como resultado un mayor número de investigaciones relacionadas a compiladores y lenguajes visuales, sin embargo se pudieron hallar avances relevantes en el área de editores mapas o niveles. A continuación se describe brevemente las categorías mencionadas en la Tabla 1 y los avances relacionados a cada una.

Lenguajes visuales específicos de dominio

Incluye aquellos trabajos que definen un lenguaje visual para tratar un problema específico y luego desarrollan un compilador para traducirlo a un programa funcional (Pautasso & Alonso, 2005).

Dentro de los artículos más resaltantes se encontró una herramienta llamada VIVO, utilizada para diseñar y aprender progresiones armónicas musicales mediante un lenguaje visual basado en PWGL (lenguaje de programación visual multiplataforma) y un compilador para la verificación de las reglas de composición (Kuuskankare & Laurson, 2007).

Por otro lado, se revisó un trabajo en el que se desarrolla un enfoque y herramienta para desarrollar aplicaciones ubicuas, programas que hacen uso de varios componentes y sensores externos, llamado Pantagruel (Drey & Consel, 2010). La investigación elabora un lenguaje visual para desarrollar este tipo de aplicaciones, sin embargo, previamente establece un meta lenguaje para definir los componentes, sensores y actores que serán usados por la herramienta (Drey & Consel, 2010).

Meta programación

En esta categoría se ha clasificado los artículos que no pretenden establecer un lenguaje de programación visual o un compilador para un dominio o problema específico, sino que buscan diseñar herramientas para su construcción. En otras palabras, un lenguaje visual para crear otro lenguaje: un meta lenguaje, o un compilador de compiladores visuales.

En el mapeo sistemático se encontró el proyecto COCOVILA, *compiler-compiler for visual languages,* una herramienta constituida de una meta lenguaje visual para el diseño de esquemas de lenguajes y un compilador asociado, basado en Java (Grigorenko et al., 2005). El objetivo de la herramienta es incrementar la rapidez y eficiencia en el desarrollo de lenguajes visuales para dominios específicos, y una de sus características más resaltantes es el editor de esquemas, el cual es completamente dirigido por sintaxis, es decir, la validez del esquema es forzada durante la edición del mismo (Grigorenko et al., 2005).

Otro avance importante es el desarrollo de un meta lenguaje para definir la estructura de un juego y sus niveles. En líneas generales, la investigación propone un meta lenguaje para establecer un tipo o género de juego, por ejemplo un RTS, en él se indican los componentes que utiliza el videojuego y las reglas que lo rigen. Luego, esta especificación de género de juego se compila y trae dos productos: un lenguaje de programación visual específico para el género de juego y un editor de niveles asociado al lenguaje (Maier & Volk, 2008).

Por último, uno de los estudios acerca del diseño de meta-lenguajes y lenguajes visuales establece una estructura para compilar la información representada en

ellos (Kastens & Schmidt, 2006). Este propone que la diferencia clave entre la programación tradicional y la visual consiste en el uso de elementos de lenguaje definidos (funciones y tipos definidos) para construir un programa, lo que además permite una revisión a tiempo real del ingreso de datos por el compilador (Kastens & Schmidt, 2006). Esto se indica en la rama derecha del diagrama representado en la Figura 2 con las flechas bidireccionales hacia el editor visual.

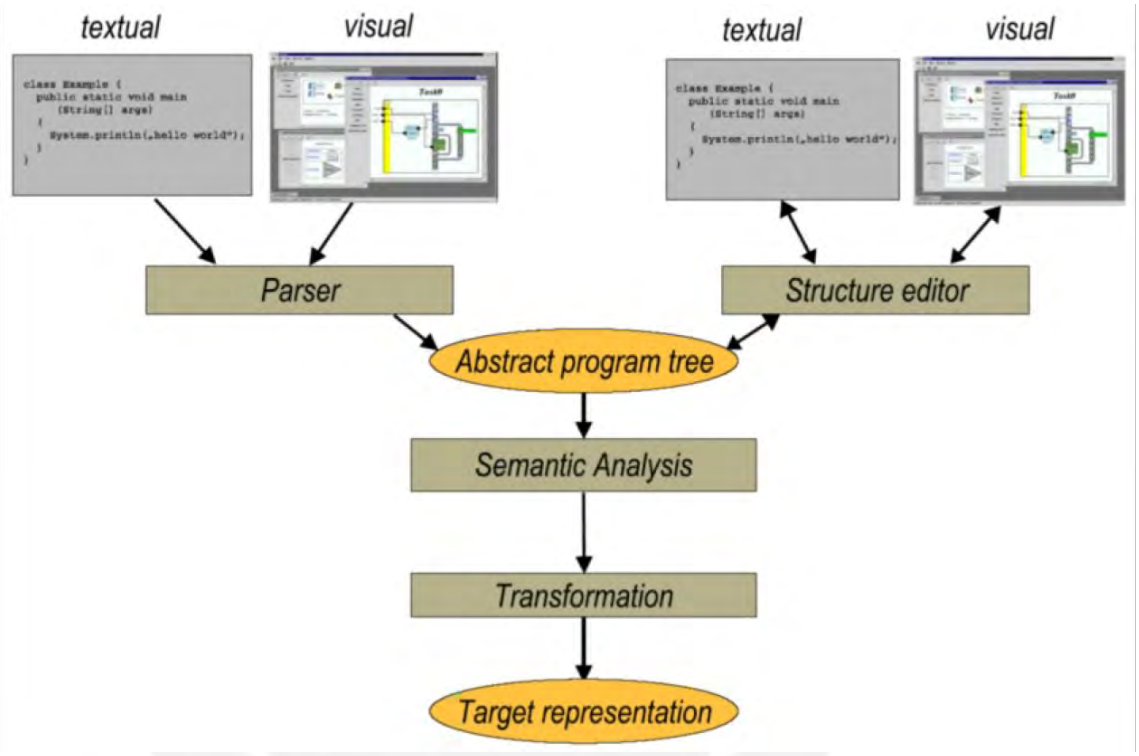

Figura 2 Procesando descripciones visuales o textuales. Imagen recuperada de Kastens & Schmidt (2006).

Diagrama de flujo de datos

Contiene a los trabajos que tienen como tema principal un tipo específico de lenguaje de programación visual: diagramas de flujo. Estos diagramas están compuestos de varias figuras, que representan bloques pequeños de código, relacionadas por flechas direccionadas, las cuales indican un flujo o secuencia de ejecución (Nummenmaa, Marttila-Kontio, & Nummenmaa, 2013).

En este ámbito se hallaron dos investigaciones relacionadas con el diseño y compilación de diagramas de flujo concurrentes, es decir, diagramas donde varios pasos o bloques de código se ejecutan en paralelo (Cox, Gauvin, & Rau-Chaplin, 2005). Uno de los artículos propone un modelo para convertir diagramas de flujo secuenciales en un lenguaje que permita programar actividades en paralelo,

mediante técnicas de sincronización y ejecución concurrente (Cox et al., 2005); mientras que el otro plantea un método para validar diagramas de flujo concurrentes, convirtiéndolos en un conjunto de procesos de estado finito (FSPs, por sus siglas en ingles), los cuales pueden ser analizados y validados por varias herramientas existentes (Nummenmaa et al., 2013).

Editores en 2D y 3D

Está categoría contiene a los artículos que detallan el diseño y desarrollo de un editor de mapas o niveles para cualquier género de videojuego. Se hace una distinción entre los editores en dos dimensiones y tres dimensiones, al tener requerimientos y tecnologías distintas.

El principal avance encontrado en este tema es la implementación de un editor de niveles que utiliza el concepto de realidad aumentada para diseñar niveles (Oswald, Tost, & Wettach, 2014). Esto básicamente implica que el usuario puede utilizar objetos en el mundo real para crear los niveles del juego. La herramienta se apoya en un proyector para renderizar el juego sobre una superficie plana como una pared y utiliza la tecnología del Microsoft Kinect para detectar los objetos reales usados como obstáculos o plataformas (Oswald et al., 2014).

Otros avances revisados incluyen dos editores de niveles que utilizan tecnologías más convencionales. El primero es un editor de niveles 3D que no está asociado a ningún motor de videojuegos. Esta herramienta se enfoca únicamente en la disposición de elementos en una pantalla y evaluar que cada objeto no se superpone sobre otro, luego el nivel se exporta a dos archivos de texto con formatos simples junto con los recursos gráficos usados (Cowan & Kapralos, 2011). El concepto del editor descrito da una primera aproximación a lo que sería un editor de mapas genérico, capaz de ser utilizado por cualquier motor gráfico en 3D.

En el segundo estudio, se detalla el desarrollo de un editor de niveles en 3D utilizando la tecnología OGRE, un motor de gráficos en 3D basado en el paradigma de programación orientada a objetos (Cai & Chen, 2010). La arquitectura del editor sigue una estructura similar al patrón modelo-vista-controlador empleado en desarrollo de software, donde el modelo es representado por dos grafos, uno para el terreno y otro para los objetos dispuestos en él, la vista es el motor de renderizado de los gráficos 3D más el UI y el controlador permite al usuario realizar cualquier acción sobre el editor (Cai & Chen, 2010).

1.4.7 Conclusiones

Queda claro después de hacer la revisión de los compiladores asociados a lenguajes de programación visual que estos responden a un diseño similar a los compiladores tradicionales, pero omitiendo algunos pasos dependiendo de su estructura. Un ejemplo de esto es el compilador de un entorno de scripting visual desarrollado con la librería OpenBlocks, el cual no necesita hacer un análisis léxico o sintáctico, debido a que cada bloque representa una palabra y el mismo editor controla que bloques se pueden conectar a otros, asegurando que una oración siempre sea sintácticamente correcta (Msiska & van Zijl, 2012).

Por otro lado, la existencia de 5 artículos, de los 16 revisados, relacionados con meta-lenguajes o meta-programación indica la existencia de un interés en la automatización e incremento de eficiencia en el diseño de lenguajes visuales de programación y sus respectivos compiladores. En especial, el descubrimiento de la aplicación del concepto de meta-lenguajes en los videojuegos en uno de los trabajos revisados (Maier & Volk, 2008) abre nuevas ramas de investigación y plantea la posibilidad de crear un lenguaje de programación visual para crear editores de niveles.

Los estudios encontrados acerca de editores de mapas o niveles, permiten concluir que no existe mucha documentación sobre esta área, a diferencia de temas como motores de física o inteligencia artificial para juegos. Sin embargo, a pesar de la reducida cantidad de investigaciones se hallaron dos ideas interesantes para explorar. La primera involucra el uso de las tecnologías relacionadas a la realidad aumentada para mejorar el proceso de creación de niveles, incrementando el aspecto intuitivo del editor y su usabilidad (Oswald et al., 2014); y la segunda rompe con el paradigma de los editores comerciales, los cuales se desarrollan para un motor de juego en particular, y propone un editor de mapas en 3D genérico (Cowan & Kapralos, 2011). Aunque este editor fue utilizado en el desarrollo de dos juegos, la investigación no muestra aplicaciones de la herramienta en la industria.

Por último, de los cuatro artículos revisados acerca de editores de niveles se pudieron encontrar similitudes en su interfaz gráfica y las opciones que ofrecen a sus usuarios. Todos utilizan sistemas parecidos al "*drag and drop*" para facilitar el diseño de los niveles y reutilizar componentes existentes. Además, los editores permiten cambiar las texturas del terreno o mapa mediante comandos parecidos a los de un editor de imágenes, incluyendo en algunos casos las opciones "deshacer" o "rehacer".

1.5 OBJETIVO GENERAL

Implementar un editor de mapas para generar niveles del juego “1814: La rebelión del Cuzco”, considerando el diseño planteado por un usuario y permitiendo traducir la información visual a la estructura de datos que el motor del videojuego utiliza.

1.6 OBJETIVOS ESPECÍFICOS

Los objetivos específicos del proyecto son:

O 1. Elaborar una interfaz gráfica que permita diseñar mapas sin la necesidad de conocimientos técnicos acerca del juego o programación.

O 2. Desarrollar un compilador que convierta el diseño creado en el editor a una estructura de datos con el formato definido en el juego “1814: La rebelión del Cuzco”.

O 3. Validar el uso del editor de mapas para diseñar un mapa que se ejecute con el motor del videojuego “1814: La rebelión del Cuzco”.

1.7 RESULTADOS ESPERADOS

Los resultados esperados asociados a los objetivos específicos son:

R 1. Interfaz gráfica que permita editar mapas del videojuego “1814: La rebelión del Cuzco”. (O1)

R 2. Resultados de la evaluación de usuario que permita validar que una persona sin conocimientos del juego o de programación pueda utilizar la interfaz. (O1)

R 3. Un analizador sintáctico en tiempo real que verifique la correctitud del diseño que el usuario está proponiendo y que genere el árbol sintáctico necesario para el compilador. (O2)

R 4. Un compilador que a partir del árbol sintáctico transforme éste en la estructura de datos soportada por el motor del juego. (O2)

R 5. Resultados de la evaluación de usuario que permita validar que una persona pueda, a través del uso del editor, generar un mapa que se ejecute con el motor del videojuego. (O3)

1.8 HERRAMIENTAS, MÉTODOS Y PROCEDIMIENTOS

En el desarrollo del proyecto planteado se utilizarán las siguientes herramientas y metodologías:

1.8.1 Diseño e implementación de lenguaje visual según Kastens y Schmidt

Para el diseño del editor de mapas propuesto en el proyecto se seguirá la metodología y estructura propuesta en el artículo "Visual patterns associated to abstract trees"(Kastens & Schmidt, 2006). Sin embargo, no se utilizará las herramientas sugeridas para la generación del lenguaje visual debido a diferencias importantes en el lenguaje objeto: en el artículo se genera un programa, mientras que en el editor se genera una estructura de datos.

La metodología planteada por el artículo incluye una serie de componentes o módulos cuya interacción se puede observar en la Figura 2, los cuales deben estar presentes para establecer un lenguaje visual y su compilador asociado (Kastens & Schmidt, 2006):

- **Editor de estructura:** es el componente con el que interactúa el usuario para editar la información del programa o en este caso, el mapa. Este editor debe ser utilizado mediante la disposición de patrones visuales y la conexión entre ellos, y puede mantener la correctitud sintáctica en tiempo real.
- **Árbol abstracto:** es la estructura central del lenguaje, representa la entrada del proceso de compilación posterior a la edición por el usuario. El árbol se modifica únicamente mediante el editor y los eventos manejados en el mismo.
- **Análisis semántico:** recibe como entrada el árbol abstracto del programa y determina la validez de los tipos, declaraciones y operadores contenidos en él. En el caso del mapa, este análisis se realizará en función a las reglas y la lógica del juego objeto.
- **Transformación:** recibe como entrada el árbol verificado por el análisis semántico y procede a transformarlo a un programa en el lenguaje objeto especificado. Para el proyecto, el lenguaje objeto será la estructura de datos soportada por el motor del juego.

Es importante resaltar que la metodología y los componentes descritos son una adaptación de las técnicas de diseño de compiladores propuestas por Aho y Ullman a un compilador que recibe como entrada un lenguaje visual, siendo la principal

diferencia la ejecución del análisis léxico y sintáctico en tiempo real mediante el editor de estructura (Aho et al., 1990). Debido a que la fase de análisis semántico depende únicamente del árbol sintáctico, esta y la posterior fase de transformación no presentan mayor adaptación en la metodología.

1.8.2 Evaluación de usuario según Rubin y Chisnell

La planificación, diseño y ejecución de las evaluaciones de usuario destinadas a validar el funcionamiento de la herramienta se realizará bajo la metodología detallada en el libro “Handbook of Usability Testing” (Rubin & Chisnell, 2008).

Los objetivos de las evaluaciones de usuario, en el contexto del proyecto, son validar la usabilidad de la interfaz gráfica de la herramienta y asegurar que la herramienta en sí permite a los usuarios crear y modificar mapas para el juego “1814: La rebelión del Cuzco”.

El tipo de evaluación que se empleó en el proyecto es el de validación, el cual se realiza en las últimas etapas del desarrollo del producto para verificar que este cumple con determinado nivel o *benchmark* de usabilidad. Por este motivo, las evaluaciones de este tipo tienden a realizarse sin casi alguna orientación del evaluador y recaban información mayoritariamente cuantitativa en base a métricas previamente identificadas. Para el caso del editor de mapas se utilizó uno de los componentes de calidad de Nielsen, explicado en la siguiente sección.

1.8.3 Componentes de calidad en usabilidad según Nielsen

Nilesen sostiene en uno de sus artículos que la usabilidad es un atributo de calidad definido en cinco componentes (Nielsen, 2012):

1. Aprendizaje: ¿qué tan fácil le resulta a los usuarios que utilizan por primera vez la interfaz realizar tareas básicas?
2. Eficiencia: después de aprender a usar la herramienta, ¿qué tan rápido pueden los usuarios realizar tareas?
3. Memorabilidad: después de volver a usar la herramienta pasado un tiempo, ¿qué tan rápido pueden los usuarios recordar su uso?
4. Errores: ¿cuántos errores cometen los usuarios? ¿Cuál es la gravedad de ellos? ¿Qué tan fácil es recuperarse de ellos?
5. Satisfacción: ¿qué tan cómodo se siente el usuario con la herramienta?

Dado que se busca que el editor de mapas sea utilizado por usuarios sin necesidad de conocimientos técnicos acerca del juego o programación, para el diseño de la interfaz gráfica y su posterior evaluación se consideraron dos componentes: la facilidad de aprendizaje y en menor medida la satisfacción, ya que esta depende de la apreciación de los jugadores no solo del editor, sino también del juego.

1.8.4 Lenguaje de programación Java

La principal herramienta para la construcción del editor de mapas propuesto en el proyecto es el lenguaje de programación Java en conjunto con el entorno de desarrollo Netbeans. Esta se eligió por dos razones:

- Portabilidad: gracias a la máquina virtual de Java los programas desarrollados bajo su lenguaje pueden ser ejecutados en cualquiera de los sistemas operativos más comerciales: Windows, iOS y Linux. Esto permite brindar mayor difusión tanto a la herramienta como al juego.
- Estandarización: el juego de "1814: La rebelión del Cuzco" fue desarrollado en este lenguaje y para disminuir los problemas que implicaría distribuirlo en conjunto con un editor de mapas implementado con otras herramientas se optó por seguir el mismo estándar.
- Conocimiento: en menor medida, se seleccionó el lenguaje Java por la experiencia previa obtenida en el uso del mismo, cuyas principales ventajas son la disminución del tiempo de implementación de la herramienta y la mejora en la precisión de los tiempos estimados en el cronograma.

1.9 ALCANCES Y LIMITACIONES

La herramienta implementada en el proyecto permitirá la edición y generación de nuevos mapas para el juego de estrategia en tiempo real "1814: La rebelión del Cuzco" mediante el uso de un entorno visual; esta no será una herramienta genérica, compatible con todos los juegos del mismo género, debido a la diversidad de lógicas y estructuras de videojuegos presentes en la industria.

Además, para facilitar el diseño gráfico de los mapas, el entorno visual permitirá a los usuarios reutilizar recursos gráficos utilizados en el juego y colocarlos en el mapa de acuerdo a sus preferencias. Sin embargo, la herramienta no permitirá al usuario insertar nuevas imágenes en el editor o modificar imágenes existentes para mantener la compatibilidad con el motor del juego.

Las funcionalidades principales de la herramienta desarrollada incluirán: edición del terreno en el mapa, disposición de unidades y edificios en el mapa, modificación del tamaño del mapa a editar y la compilación de todo el contenido editado en el entorno visual a una estructura de datos soportada por el motor del juego “1814”. Funcionalidades adicionales incluidas en editores comerciales, como la inclusión de scripts para crear eventos personalizados o la modificación de unidades existentes, no serán consideradas dentro del alcance del proyecto pues son características propias a un motor de juego.

Por otro lado, dado que para las evaluaciones de usuario se requiere de la participación de personas con perfiles determinados y posiblemente externas al proyecto, no se pueden realizar muchas sesiones con una gran cantidad de usuarios. Considerando esto, se ha limitado la cantidad de personas por evaluación, considerando que son dos: una para la interfaz gráfica y otra para la validación de todo el flujo de trabajo de la herramienta.

1.10 JUSTIFICACIÓN

La razón primordial para realizar el editor de mapas es la capacidad de generar nuevo contenido para el juego “1814: La rebelión del Cuzco” involucrando a los jugadores en el proceso creativo, sin incurrir en gastos de recursos y tiempo, permitiendo incrementar el tiempo de vida del videojuego y a su vez el interés de los jugadores. Por otro lado, el proyecto establece una forma ordenada y basada en metodologías de compiladores para hacer editores de mapas, la cual puede ser adaptada por cualquier casa de desarrollo de videojuegos que desee incluir una herramienta de este tipo en sus productos.

1.11 VIABILIDAD

La siguiente sección presenta las justificaciones de la viabilidad del proyecto desde tres dimensiones distintas: temporal, técnica y económica.

1.11.1 Temporal

El tiempo necesario para implementar la interfaz gráfica y el analizador sintáctico se ha reducido considerablemente gracias a que se ha avanzado previamente el editor con el grupo AVATAR. No obstante, igual se considera un esfuerzo para la adaptación de los componentes desarrollados a la nueva metodología.

Tarea	Días	Comienzo	Fin
Editor de Mapas 1814	**87**	**13/02/16**	**09/05/16**
Implementación de interfaz gráfica	**27**	**13/02/16**	**05/04/16**
Implementar GUI	7	13/02/16	19/02/16
Implementar renderizador isométrico	10	20/02/16	29/02/16
Pruebas de interfaz	4	01/03/16	04/03/16
Revisión post-prueba de interfaz	6	31/03/16	05/04/16
Implementación de compilador	**46**	**13/02/16**	**30/04/16**
Implementación de analizador sintáctico	**10**	**13/02/16**	**22/02/16**
Revisión y adaptación de validador sintáctico	7	13/02/16	19/02/16
Pruebas unitarias	3	20/02/16	22/02/16
Implementación de analizador semántico	**12**	**05/03/16**	**16/03/16**
Validación de edificios y sus respectivas puertas	6	05/03/16	10/03/16
Validación de héroes y límite de unidades	3	11/03/16	13/03/16
Pruebas unitarias	3	14/03/16	16/03/16
Implementación de proceso de transformación	**24**	**07/04/16**	**30/04/16**
Generación de estructura de datos	17	07/04/16	23/04/16
Integración con el juego	5	24/04/16	28/04/16
Pruebas unitarias	2	29/04/16	30/04/16
Desarrollo de evaluaciones de usuario	**23**	**17/03/16**	**09/05/16**
Diseño de las evaluaciones de usuario	6	17/03/16	22/03/16
Ejecución de evaluación de interfaz gráfica	8	23/03/16	30/03/16
Ejecución de evaluación de herramienta integrada	9	01/05/16	09/05/16

1.11.2 Técnica

Dentro de las herramientas, metodologías y conocimientos necesarios para la ejecución del proyecto no existe un elemento que represente una limitación a su viabilidad, esto se debe a que:

- Para el caso del lenguaje de programación Java, ya se posee experiencia previa en el desarrollo de aplicaciones de escritorio utilizando este lenguaje gracias a los cursos de la carrera, además de existir un amplio soporte y documentación.
- El diseño e implementación de compiladores visuales se ha investigado extensamente en la revisión del estado del arte y se conoce mejor las metodologías asociadas.
- Se posee un extenso manual para la planificación, diseño y ejecución de evaluaciones de usuario, el cual ya se ha revisado previamente para determinar su aplicabilidad al proyecto y disminuir el tiempo de aprendizaje de la herramienta.

1.11.3 Económica

Las herramientas propuestas para la ejecución del proyecto no suponen ningún costo a considerar gracias a que su uso es completamente gratuito, sin embargo, se deben considerar otro tipo de costos existentes de forma implícita en el desarrollo del mismo.

Por un lado, existe el costo asociado al uso de recursos humanos, el cual se traduce en las horas hombre necesarias para la ejecución del proyecto. Estimando 5 horas de trabajo en el proyecto por día, se calcula un costo total por RRHH de 435 horas hombre.

Un costo total de 435 horas hombre para el desarrollo de una herramienta de este tipo, incluyendo la ejecución de pruebas de usabilidad, resulta aceptable para el contexto peruano y no afecta la viabilidad del proyecto.

CAPÍTULO 2: INTERFAZ GRÁFICA

En el presente capítulo se explica en detalle la estructura y funcionamiento de la interfaz gráfica del editor. Esto incluye la disposición de los elementos gráficos, su interacción con el usuario y la impresión isométrica del mapa.

2.1 ARQUITECTURA DE LA INTERFAZ

Para facilitar la legibilidad, mantenimiento y depuración del código de la interfaz, esta se dividió en tres subcomponentes cuyas funciones se diferencian claramente y trabajan de forma independiente:

1. **Vista:** compromete todos los elementos gráficos, provistos por las librerías nativas de Java, que se muestran en la pantalla, esto incluye ventanas, controles, diálogos y el panel principal donde se edita el mapa.
2. **Renderizador:** es una librería que se encarga de dibujar el mapa, las unidades y edificios manteniendo una perspectiva isométrica. Además, se encarga de gestionar los recursos gráficos para disminuir el tiempo de dibujo y el uso de memoria.
3. **Controlador:** contiene una serie de funciones utilitarias que no comprometen al compilador. Esto incluye: la carga de recursos del editor, la creación de un mapa y su posterior almacenamiento.

Por otro lado, la arquitectura del compilador sigue la estructura recomendada en el artículo acerca de la implementación de lenguajes visuales (Kastens & Schmidt, 2006) e interactúa con la interfaz por medio de llamadas desde la vista.

En la Figura 3 se muestra en un diagrama las dependencias entre todos los elementos de la aplicación. Cabe mencionar que el modelo es utilizado tanto por la interfaz gráfica como el compilador, pues este define los objetos y estructuras básicas para representar un mapa, los cuales brindan una forma simple de transmitir información entre los componentes, evitando utilizar identificadores y búsquedas como se haría en un sistema cliente-servidor convencional.

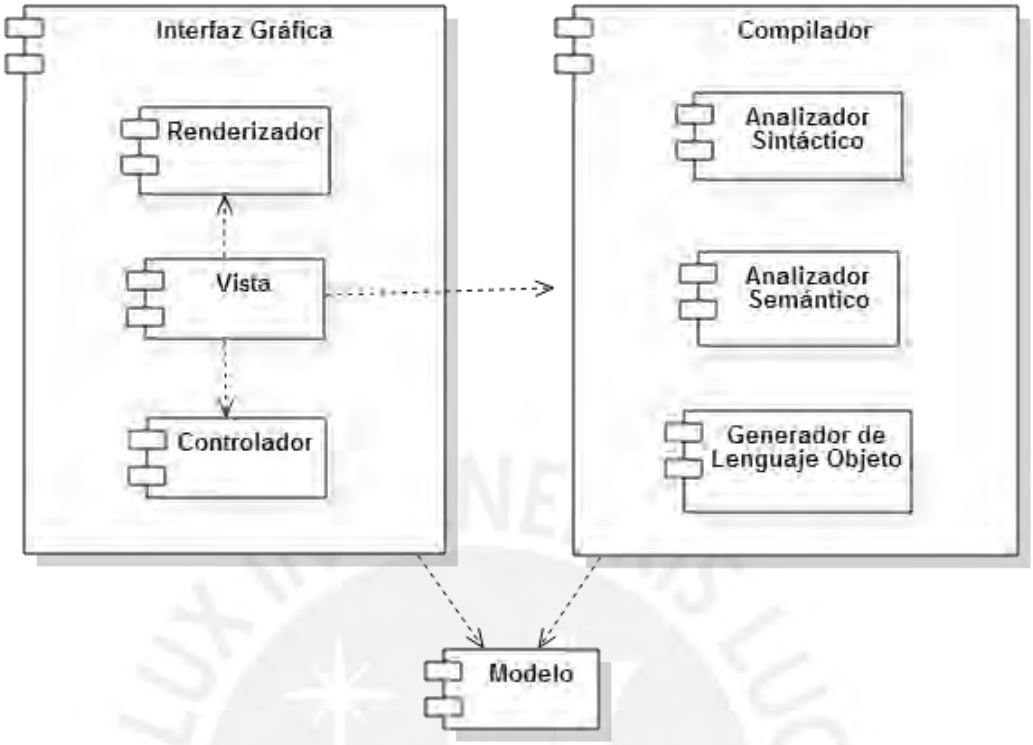

Figura 3 Diagrama de componentes del editor. (Elaboración propia)

2.2 PANTALLAS Y CONTROLES

En la Figura 4 se puede observar la ventana principal del editor, en la que se realizan todas las acciones de edición del mapa, a excepción de la creación de uno nuevo, lo cual se configura en una ventana adicional. Basándose en el editor comercial de Starcraft, un exitoso juego de RTS (Edge, 2008), la pantalla principal se compone de cuatro elementos:

1. **Barra de menú:** barra ubicada en el borde superior de la ventana que contiene opciones generales como crear, abrir, guardar o exportar un mapa, deshacer y rehacer. Estos comandos tienen sus respectivos atajos con el teclado para permitir mayor rapidez en el trabajo.
2. **Panel de opciones:** ubicado en la esquina inferior izquierda, contiene los controles necesarios para agregar objetos al mapa y modificar su terreno.
3. **Panel de trabajo:** ubicado en el centro, y el elemento de mayor tamaño. En él se muestra el mapa que está siendo editado.
4. **Minimapa:** una versión miniatura del mapa que está siendo mostrado en el panel de trabajo.

Figura 4 Pantalla principal del editor (Elaboración propia)

2.2.1 Panel de opciones

Como su nombre lo indica, contiene todas las opciones disponibles para el usuario, agrupadas en tres pestañas cómo se muestra en la Figura 5. A continuación se detallará cada una de ellas:

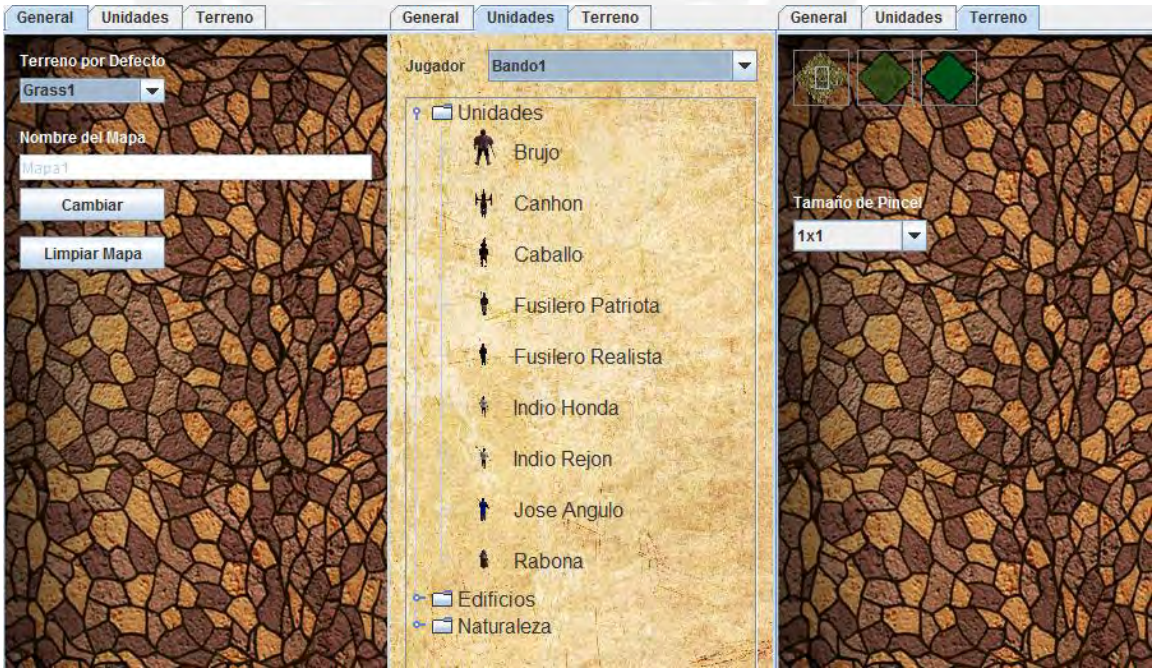

Figura 5 Panel de opciones (Elaboración propia)

Pestaña general

En está pestaña se agruparon los comandos u opciones que corresponden a características generales del mapa o cuyo uso es de poca frecuencia:

- Terreno por defecto: cambia todo el terreno del mapa al tipo especificado. No se incluyó en la pestaña de terreno por su frecuencia de uso baja y similitud con el comando de “Limpiar Mapa”.
- Limpiar mapa: elimina todos los objetos dispuestos en el mapa.
- Nombre del mapa: permite cambiar el nombre del mapa, este se usa luego para determinar el nombre de los archivos exportados.

Pestaña de unidades

Esta pestaña contiene un directorio o paleta con las unidades disponibles para colocar en el mapa, las cuales le pertenecerán al jugador que esté seleccionado en el cuadro combinado ubicado en la parte superior del componente.

Pestaña de terreno

Las opciones que permiten cambiar las texturas o apariencia del terreno fueron colocadas bajo esta pestaña. Para el editor se consideraron dos: un listado de botones con la textura a la que se desea cambiar el terreno y un cuadro combinado que cambia el tamaño del “pincel de terreno”. Esta forma de cambiar el terreno se diseñó para asemejar a editores de imágenes como Paint, permitiendo a los usuarios familiarizarse rápidamente con las opciones. No se agregó una opción de tamaño del mapa a esta pestaña ya que las dimensiones se definen previamente, al momento del crear el mapa.

2.2.2 Panel de trabajo

Es el componente encargado de realizar las dos funciones más importantes en el editor: mostrar el mapa que está siendo editado, reflejando la estructura lógica interna que luego será compilada, y manejar las acciones que el usuario desea realizar sobre el mapa.

Acciones del usuario

Para recibir cada acción del usuario en el panel se definieron los siguientes botones o teclas:

- Agregar un objeto: se realiza con el botón izquierdo del mouse, habiendo seleccionado previamente una unidad en la pestaña de unidades.

- Eliminar un objeto: se realiza con el botón de la rueda del mouse.
- Mover un objeto: se realiza manteniendo el botón derecho del mouse presionado sobre un objeto y desplazándolo hacia otra ubicación.
- Cambiar la orientación de un objeto: se realiza girando la rueda del mouse sobre un objeto dispuesto en el mapa.
- Pintar el terreno: se realiza presionando el botón izquierdo del mouse, habiendo seleccionado previamente un tipo de terreno en la pestaña de terreno.

El principio detrás de esta configuración es el de minimizar el número de botones y la distancia entre ellos, facilitando su memorización para el usuario. Así, se optó por usar todos los botones disponibles en un mouse convencional.

2.2.3 Minimapa

La mayoría de mapas en el juego "1814: La rebelión del Cuzco" son bastante amplios y su edición implica un desplazamiento constante entre cada zona que se desea modificar. Para facilitar este trabajo se incluyó el minimapa, el cual brinda dos ventajas:

- Desplazamiento instantáneo: haciendo click en un punto del minimapa la cámara en el panel de trabajo se centra inmediatamente en el punto del mapa real equivalente. Así, el usuario no tiene que utilizar las flechas direccionales para moverse entre zonas muy lejanas.
- Visibilidad del mapa: el minimapa permite ver el mapa en su totalidad y tener una idea general de cómo están dispuestos todos los elementos, algo que no se puede visualizar en el panel de trabajo.

2.3 RENDERIZADOR ISOMÉTRICO

Renderizar es un término que proviene de la palabra en inglés "*render*" cuyo significado, según la universidad de Oxford, es: procesar computacionalmente una imagen general utilizando color y sombreado para hacer que luzca sólida y tridimensional (Oxford Dictionaries, n.d.). En el caso del editor, este proceso se realiza para representar visualmente la estructura lógica del mapa en una perspectiva isométrica, la cual genera una sensación de profundidad la imagen.

Además, el renderizador cuenta con los métodos necesarios para gestionar de forma eficiente los recursos gráficos empleados para dibujar el mapa.

2.3.1 Gestión de recursos gráficos

Una de las limitaciones más grandes de la librería gráfica de Java es que los métodos para transformar imágenes, como encoger o agrandar, son sumamente caros en memoria y procesamiento.

Para disminuir el impacto de esta limitación en los métodos de dibujo del renderizador, este mantiene un *cache* con las imágenes que se podrían necesitar para dibujar el mapa o el minimapa. Así, si se requiere de una imagen más pequeña, está se encoge y luego se guarda, asegurando que no se tenga que realizar la transformación constantemente. Esta técnica mejoró considerablemente el tiempo de impresión del minimapa.

Además, antes de guardar cualquier imagen en *cache*, la librería se encarga de optimizarla a un formato de datos que sea más conveniente para el sistema operativo donde se ejecuta, disminuyendo el tiempo de dibujo del renderizador.

2.3.2 Impresión isométrica

Antes de entrar a los detalles de la impresión y la fórmula utilizada para ubicar cada componente del mapa, es necesario describir brevemente las propiedades geométricas de una vista en perspectiva isométrica.

Originalmente, esta perspectiva se ideó para simplificar la representación de objetos tridimensionales en dos dimensiones y su característica principal se puede observar en la Figura 6: una diferencia angular de 30° entre los ejes X y Y con la recta perpendicular al eje Z (Krikke, 2000). Sin embargo, la propiedad que mejor se aprovechó es la simetría en el rombo que representa la tapa del cubo (denominado mosaico), tal como se muestra en la Figura 6.

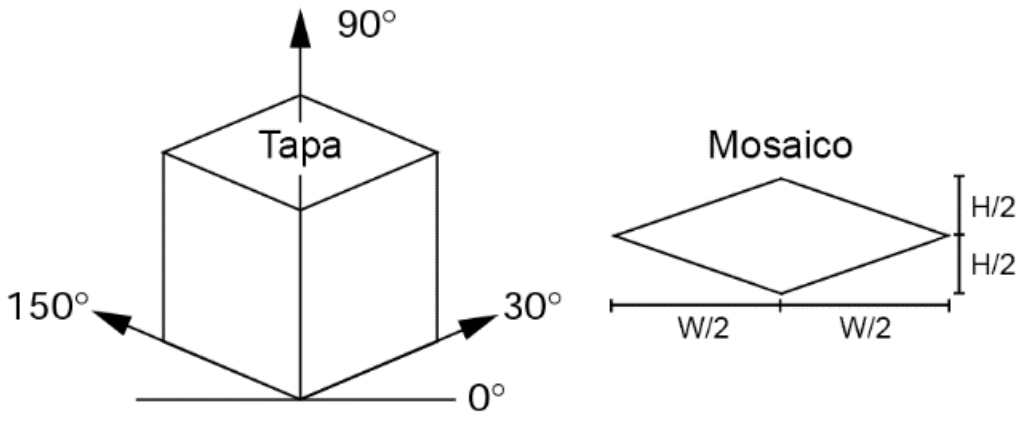

Figura 6 Propiedades de la perspectiva isométrica. Adaptado de Krikke, J. (2000).

A continuación se presenta las técnicas que se usó para dibujar primero el terreno del mapa, representado por un conjunto de mosaicos, y luego las unidades.

Terreno

Considerando las características presentadas previamente, se determinó las siguientes fórmulas para calcular la posición de impresión de cada elemento (esquina izquierda superior del mosaico), basado en su posición en la matriz especificada en la Figura 7, donde:

- J es la columna en la matriz
- I es la fila en la matriz
- W es la anchura del mosaico
- H es la altura del mosaico
- OffsetY es una constante calculada cada vez que se carga un mapa, la cual evita imprimirlo cortado a la mitad horizontalmente.

$$IsoX = (j + i) * \frac{W}{2}$$

$$IsoY = (j - i) * \frac{H}{2} - \frac{H}{2} + offsetY$$

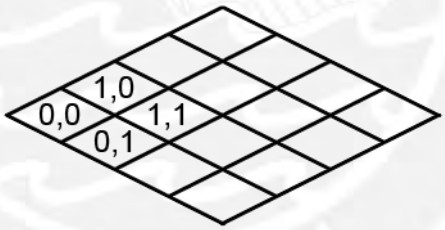

Figura 7 Sistema de coordenadas isométrica. (Elaboración propia)

No obstante, recorrer toda la matriz de mosaicos e imprimir cada uno utilizando estas fórmulas representa un gasto innecesario de recursos y tiempo, ya que el usuario no visualiza el mapa completo en un mismo instante, a menos que sea muy pequeño.

Por esta razón, se implementó un método para determinar qué mosaicos necesitan ser dibujados, basado en las dimensiones del panel de trabajo y el mapa, logrando que el tiempo de renderizado, previamente proporcional al tamaño de mapa, sea constante.

Unidades

La impresión de unidades también utiliza las fórmulas y técnicas explicadas en la sección anterior, no obstante, la forma en la que se recorre la matriz de mosaicos es distinta. Mientras que en el terreno se recorre la matriz fila por fila, el renderizado de las unidades debe recorrerla de forma transversal para asegurar que los objetos que están “atrás” en la perspectiva se dibujen primero y los que están más adelante encima de ellos. La diferencia en el procesamiento de la matriz se puede observar de forma gráfica en la Figura 8.

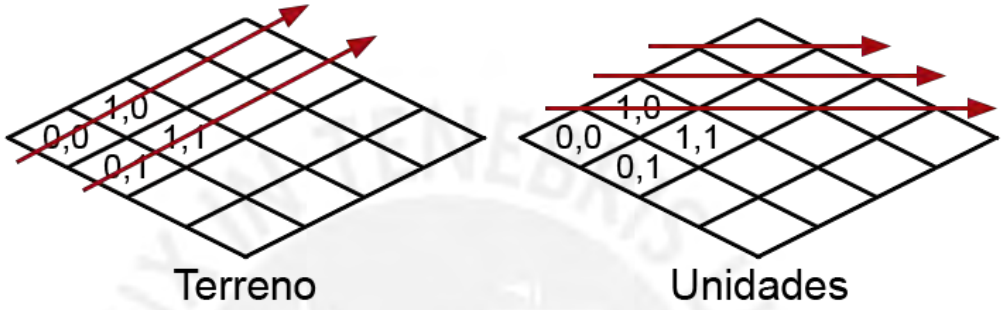

Figura 8 Recorrido de la matriz de mosaicos. (Elaboración propia)

Por otro lado, como parte del proceso de validación en tiempo real y retroalimentación visual para el usuario, se imprime en el mapa un objeto preliminar transparente cuando se está agregando o moviendo objetos. Este cambia de color cuando la posición sobre la que se encuentra es inválida, permitiendo al usuario entender porque no puede realizar determinada acción.

2.3.3 Impresión del minimapa

La dificultad más grande para lograr imprimir el minimapa es que, a diferencia del mapa en tamaño real, este debe visualizarse completamente en todo momento. Esto ocasionó inicialmente tiempos muy lentos de renderización para mapas de gran tamaño (100x100 o más mosaicos).

Para solucionar este problema se utilizó la combinación de dos estrategias: impresión única e impresión en dos etapas.

Impresión única

Resultó claro que recorrer a 30 imágenes por segundo un mapa amplio era demasiado ineficiente, aun usando mosaicos en *cache*. Por este motivo, se utilizó la siguiente estrategia: el terreno del mapa solo se imprime una vez y se guarda en memoria. Posteriormente, si el usuario decide modificar el terreno, esa misma modificación es replicada en la imagen guardada, pero a menor escala. De esta

manera, el minimapa se actualiza a tiempo real, pero no es necesario recorrer el mapa completo en cada renderización.

Impresión en dos etapas

Se estableció previamente en la gestión de recursos gráficos que las imágenes encogidas se mantienen en *cache* para evitar transformarlas constantemente. Sin embargo, esto trajo consigo un nuevo problema para la renderización del minimapa, asociado a otra propiedad de las librerías gráficas de Java: inexactitud decimal en la impresión.

Los métodos de dibujo en Java solo aceptan como parámetros de tamaño o posición números enteros y en la impresión del minimapa el tamaño de los mosaicos encogidos, para mapas de grandes dimensiones, resultó un número decimal como 1.5 o 0.85. La consecuencia de esta obligatoria aproximación fue la impresión del minimapa en dimensiones incorrectas, viéndose más pequeño que el componente visual que lo contiene.

Para solucionar esto, se decidió no encoger el mapa original al tamaño del minimapa directamente, sino en dos etapas. La estrategia se diseñó de la siguiente manera: primero se encoge el mapa real a una imagen de tamaño constante, el cual asegura que no se aproxime la dimensión de los mosaicos encogidos. Y luego, está imagen intermedia se transforma utilizando las librerías de Java al tamaño deseado.

La ventaja que supuso está estrategia es la capacidad de imprimir el minimapa correctamente minimizando el costo de la transformación de la imagen intermedia al tamaño deseado. Este método se combina con el anterior manteniendo la imagen intermedia como la imagen única, sobre la cual se replican las modificaciones.

2.4 PRUEBA DE USABILIDAD

Al terminar la construcción de la interfaz gráfica se realizó una prueba con la finalidad de verificar su usabilidad en la edición del mapa, siguiendo la metodología de Rubin y Chisnell. Para la prueba se siguió la definición de usabilidad indicada en la norma ISO 9241-210: medida en la que un producto se puede usar por determinados usuarios para conseguir objetivos específicos con efectividad, eficiencia y satisfacción en un contexto de uso especificado (ISO 9241, 2010). A continuación se presenta el diseño de la evaluación y los resultados obtenidos.

2.4.1 Objetivos

Los objetivos de la prueba de usabilidad se definieron como:

- Evaluar la eficiencia de la interfaz del editor para distintos tipos de usuarios.
- Identificar los principales problemas en el uso del editor, tanto en la disposición de controles como las teclas/botones para realizar acciones.
- Determinar si el nivel de retroalimentación visual en el mapa editado es el apropiado.

La utilización del editor se encuentra representada por un conjunto de acciones principales disponibles para el usuario: agregar objetos al mapa, moverlos, eliminarlos, rotarlos, cambiar el tipo de terreno y desplazar la cámara. Además de otras acciones secundarias como guardar o abrir un mapa existente.

2.4.2 Preguntas de investigación

Las siguientes preguntas se plantearon en base a los objetivos y las funcionalidades existentes en el editor:

- ¿Cuán fácil es para los usuarios crear o abrir un nuevo mapa?
- ¿Cuán fácil es para los usuarios guardar el mapa que se encuentran editando?
- ¿Cuál es la apreciación de los usuarios acerca de la disposición de los botones u otros controles?
- ¿Qué problemas experimenta el usuario para agregar, mover, rotar o eliminar un objeto del mapa?
- ¿Es cómodo para el usuario el botón asignado para agregar, mover, rotar y eliminar un objeto?
- ¿Cuán fácil es para los usuarios desplazar la cámara en el mapa? ¿Utilizan el minimapa?
- ¿Qué tan fácil le resulta al usuario editar el terreno? ¿Queda claro el concepto del tamaño de pincel?

2.4.3 Características de los participantes

Para la prueba de usabilidad de la interfaz no se definió una clasificación de usuarios, pero se establecieron ciertos aspectos que debían cumplir, considerando que una persona que utiliza el editor debió haber jugado previamente el juego:

- Adolescentes y jóvenes adultos, en el rango de 15 – 25 años.
- Experiencia con el juego “1814: La rebelión del Cuzco”

Debido a que la prueba de la interfaz gráfica es de menor escala y es reforzada por la prueba de validación final, esta se realizó con cuatro usuarios.

2.4.4 Descripción del método

Las pruebas se realizaron de forma individual, principalmente por la disponibilidad de tiempo de los usuarios, utilizando una laptop de gama media con el editor y una grabadora de pantalla instalados. La ejecución de la prueba se grabó para poder revisar comportamientos en el uso del editor que pudieron pasar desapercibidos.

Cada prueba constó de tres fases:

- Introducción: se explica los objetivos de la prueba y algunos controles básicos para el uso del editor al participante. Luego, si está de acuerdo, firma el consentimiento informado que se muestra en el Anexo B.
- Ejecución: se realiza la lista de tareas elaborada para la prueba. Mientras el participante avanza se anotan sus comentarios y si realiza con éxito las actividades.
- Post-prueba: se realiza un cuestionario después de acabar con las tareas para recoger datos cualitativos y opiniones finales acerca del editor.

2.4.5 Resultados de la evaluación

Los resultados de la ejecución de la prueba y el cuestionario post-prueba se pueden observar en el Anexo C. A continuación, se contestan las preguntas de investigación en base a la información recolectada:

- Crear un nuevo mapa resultó fácil para todos los usuarios. Algunos tardaron un momento en identificar el campo “Personalizado”, pero lo descubrieron intuitivamente. La funcionalidad de abrir un mapa fue bastante directa y simple.
- Guardar el mapa fue bastante fácil para los usuarios. Sin embargo, la forma previa de guardar el mapa automáticamente en un archivo con su nombre demostró no ser muy usable y se cambió a un selector de archivos.
- En general, la prueba confirmó que la disposición de elementos resulta usable para los participantes, permitiéndoles acostumbrarse rápidamente a la configuración del editor.

- Se presentaron dos problemas con respecto a las acciones ejecutadas por el usuario: no queda claro el significado del área gris en el mapa (área no editable) y los usuarios estaban acostumbrados a seleccionar un objeto antes de modificarlo.
- Los botones del mouse configurados para realizar las acciones resultaron ser bastante cómodos para los usuarios. Unos 10 minutos después de interactuar con el editor ya manejaban con facilidad todos los comandos.
- El desplazamiento utilizando el minimapa fue una de las funcionalidades más utilizadas por los usuarios y demostró ser muy intuitiva. No obstante, las teclas direccionales no se usaron mucho, posiblemente por mover despacio la cámara.
- Otro aspecto que resultó ser intuitivo y fácilmente utilizable es el editor de terreno. Los participantes rápidamente identificaron que funcionaba como un pincel, y podían mantener presionado el mouse para pintar sobre el mapa. Un usuario recomendó mostrar el tamaño del pincel en los mosaicos, lo cual fue implementado al finalizar las pruebas.

2.4.6 Conclusiones

Las pruebas realizadas dieron en su mayoría resultados positivos, indicando que la interfaz gráfica desarrollada es usable y fácil de aprender, dadas unas breves instrucciones iniciales. Sin embargo, se detectaron defectos leves en ciertos aspectos de la interfaz, como por ejemplo: la rotación de las unidades sin una interacción previa o la forma automática de guardar el mapa. Las observaciones que se compartían entre la mayoría de los participantes fueron trabajadas de distintas maneras, pero aquellas que representaban preferencias específicas y no afectaban tanto la usabilidad del editor fueron omitidas.

Estas mejoras a la interfaz fueron luego reevaluadas mediante la prueba de validación final, que incluye los criterios utilizados en esta prueba, más algunos aspectos adicionales.

CAPÍTULO 3: COMPILADOR VISUAL

En este capítulo se detalla el proceso de generación del árbol sintáctico y su posterior compilación a la estructura de datos que soporta el motor del videojuego "1814: La rebelión del Cuzco".

3.1 ANALIZADOR SINTÁCTICO

La función del analizador sintáctico, como se mencionó en el capítulo de marco conceptual, consiste en validar la sintaxis del código elaborado por el usuario o, en el caso del editor, el mapa.

Esta validez se determinó en base a las reglas del videojuego, ya que el mapa editado debe ser exportado e interpretado por su motor. Intentar cargar un mapa que no tiene coherencia con la lógica de juego puede traer como consecuencia comportamientos o errores inesperados.

A continuación se explica la estructura del árbol sintáctico y cómo se mantiene su validez en tiempo real.

3.1.1 El árbol sintáctico

Es la estructura de datos que el editor permite modificar mediante la interfaz gráfica. En el caso del editor, se diseñó la clase "Mapa", la cual representa todos los elementos de un mapa del videojuego y a su vez, cumple el rol de árbol sintáctico. La Figura 9 presenta un esquema simplificado del mapa:

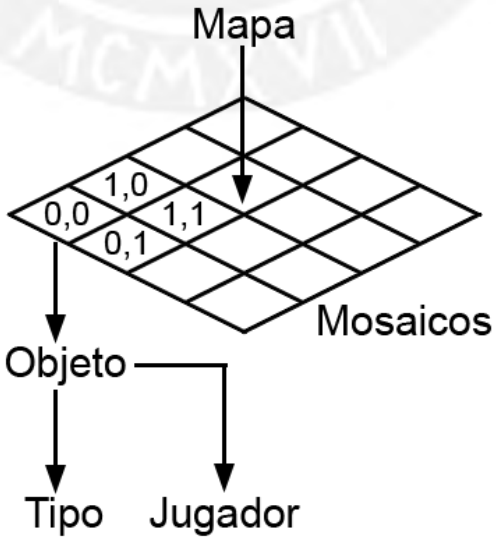

Figura 9 Esquema del árbol sintáctico. (Elaboración propia)

Un mapa está compuesto por la matriz de mosaicos que representa su terreno, cada mosaico puede tener un objeto posicionado en él y este objeto posee un clase base, o tipo, y pertenece a un jugador, que puede ser el usuario, computador o neutral. Se utilizó está estructura porque organiza de forma natural los elementos perteneciente al mapa y permite dibujarlos en un solo recorrido de la matriz de mosaicos.

Adicionalmente, fue requerido mantener en paralelo una colección con todos los objetos colocados en el mapa por dos motivos: facilitar la impresión de las unidades en el minimapa, pues la matriz solo se recorre una vez, y permitir la serialización del mapa a un archivo XML, lo cual se explica en la sección 3.1.3.

3.1.2 Validación de acciones

Las reglas del videojuego que se consideró en el diseño del analizador sintáctico fueron las siguientes:

1. Un objeto no puede ocupar un mosaico que se encuentra ocupado por otro.
2. Un objeto no puede posicionarse fuera de los límites lógicos del mapa.
3. Dos héroes con un mismo nombre no deben existir en el mismo mapa. Los héroes son unidades con propiedades especiales.

No se incluyó reglas asociadas al tipo de terreno (por ejemplo, un soldado no se puede poner en el agua) debido a que “1814: La rebelión del Cuzco” no las soporta. Siguiendo los requerimientos de estas reglas se diseñó las acciones que puede realizar el usuario, las cuales se pueden observar en la Tabla 2.

Tabla 2 Acciones de edición del mapa (Elaboración propia)

Acción	Validación	Actualización del árbol
Colocar objeto	• Todos los mosaicos que ocupa el objeto deben estar desocupados. • La posición indicada debe estar dentro de los límites del mapa. • En caso sea un héroe, no debe existir otro con el mismo nombre. • El número de unidades existente debe ser menor a 35.	• Se agrega el objeto al árbol y se ocupan los mosaicos correspondientes.
Eliminar objeto	• Existe un objeto en la posición indicada.	• Se elimina el objeto del árbol y se desocupan los mosaicos correspondientes

Mover objeto	• Existe un objeto en la posición indicada. • Todos los mosaicos que ocupa el objeto en la nueva posición deben estar desocupados. • La posición indicada debe estar dentro de los límites del mapa.	• Se elimina el objeto del árbol y se desocupan los mosaicos correspondientes. • Se agrega el mismo objeto al árbol y se ocupan los mosaicos correspondientes.
Rotar objeto	• Existe un objeto en la posición indicada. • Los mosaicos ocupados producto de la rotación deben estar desocupados.	• Se elimina el objeto del árbol y se desocupan los mosaicos correspondientes. • Se agrega el mismo objeto con rotación alterada al árbol y se ocupan los mosaicos correspondientes.

Se decidió aprovechar las funcionalidades básicas de la acción de agregar y eliminar para construir las acciones de mover y rotar. Así, en vez de crear una nueva funcionalidad “mover”, simplemente se elimina el objeto y se vuelve a agregar en la nueva posición, reutilizando hasta las validaciones.

Por otro lado, la funcionalidad de editar el tipo de terreno también es una acción permitida al usuario, sin embargo, esta no se incluyó en la tabla debido a que no tiene mayor consecuencia en la integridad del árbol sintáctico.

3.1.3 Guardado y cargado del mapa

Considerando que un usuario no necesariamente termina de editar un mapa en una sola sesión se implementó la funcionalidad de guardar y cargar, la cual serializa el mapa en un archivo con formato XML.

Para lograr esto, se utilizó JAXB (*Java Architecture for XML Binding*) y su implementación por defecto en las librerías de Java 1.6, gracias a que simplifica la conversión entre los datos XML y las clases. Esta herramienta se escogió también por tener una estructura compatible con el árbol sintáctico, ya que JAXB requiere que exista una estructura jerárquica entre las clases y un nodo raíz que contiene el resto de objetos.

Sin embargo, se encontró un problema con la serialización de objetos con un ancho y/o alto mayor a un mosaico: al deserializarlos, por cada mosaico que ocupaban se creaba un nuevo objeto, en vez de todos tener un puntero al mismo. Por este motivo se tuvo que mantener una colección con todos los objetos en el mapa, la cual se carga primero y luego los mosaicos apuntan a algún elemento en ella.

3.2 ANALIZADOR SEMÁNTICO

En la fase de análisis semántico se busca verificar que todos los elementos dispuestos en el árbol sintáctico cumplen con ciertas reglas asociadas a sus tipos o en este contexto, a la lógica del juego. En el caso de “1814: La rebelión del Cuzco”, fue suficiente validar que no se posicionara más de un objeto en un mismo mosaico o fuera de los límites del mapa para asegurar su correcto funcionamiento.

Sin embargo, después de realizar las pruebas de integración se detectó una verificación adicional que pertenece a esta fase y se realiza únicamente antes de exportar el mapa: la cantidad de héroes. Para que el juego pueda cargar el mapa correctamente, es necesario que tanto el usuario como la máquina tengan como mínimo un héroe.

3.3 GENERADOR DE CÓDIGO OBJETO

Después de la edición del mapa y su validación, es necesario convertirlo a la estructura de datos soportada por el videojuego. Esta es la función del generador de código objeto y a continuación se presenta su funcionamiento y los archivos de salida esperados.

“1814: La rebelión del Cuzco” requiere de cuatro tipos de archivos para poder interpretar un mapa correctamente: colisiones, unidades (uno por cada bando), entradas/puertas y la imagen del mapa en formato PNG.

Archivo de unidades

Este archivo de texto en formato XML contiene la información básica para cargar las unidades en el juego: posición, nombre de la unidad, orientación, bando al que pertenece y otros atributos constantes propios del juego. En el Anexo F se puede observar cada propiedad exportada.

El método usado para generar el archivo de unidades es bastante directo: se recorre la colección que contiene todos los objetos del mapa y para cada unidad se leen sus atributos y se genera el objeto XML correspondiente.

Archivo de colisiones

Posiblemente el archivo más ilegible y complejo de los tres archivos de texto. Representa qué mosaicos pueden ser transitados, están fuera de los límites del mapa o están detrás de un edificio mediante una secuencia de caracteres organizados en una disposición que simula la matriz del mapa en perspectiva ortogonal. Una muestra de este archivo se incluye en el Anexo G.

Archivo de entradas/puertas

Contiene en un formato de texto simple todos los edificios que tienen puertas o entradas activas. Estas entradas ya se encuentran configuradas por defecto en el editor, en base al recurso gráfico correspondiente. El formato en detalle junto con un ejemplo del archivo se muestra en el Anexo H.

Al igual que el archivo de unidades, generar este archivo es bastante directo gracias a la estructuración previa del mapa. Basta con recorrer todos los edificios posicionados en él, analizar sus objetos "Entrada" activos y determinar la orientación de la entrada (suroeste, sureste, noroeste, noreste) para poder generar todos los registros del archivo.

Imagen del mapa

El último archivo requerido por el motor de juego para interpretar el mapa es su imagen en formato .PNG, sin incluir las unidades, ya que la imagen representa el "fondo" del nivel, y las unidades se dibujan encima de él.

La generación de la imagen reutiliza los métodos de renderización mencionados en la sección 2.3.2, no obstante, se tuvieron que tomar consideraciones adicionales acerca del segmento de mapa que se debía extraer. Esto fue necesario a causa de la forma rectangular de los niveles de "1814: La rebelión del Cuzco", la cual ocasiona la situación descrita en la Figura 10: el mapa naturalmente es un rombo aplanado, y la imagen exportada debe ser el rectángulo contenido.

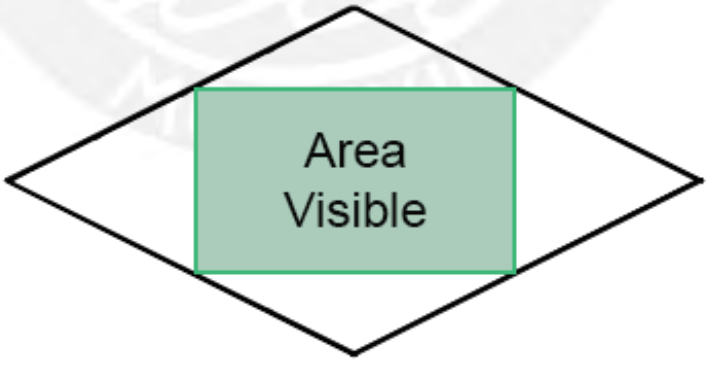

Figura 10 Área del mapa a ser exportada. (Elaboración propia)

Para solucionar este problema se modificó la forma de especificar las dimensiones al crear un nuevo mapa. Previamente se indicaba el ancho y alto, en mosaicos, del rombo; después del cambio el usuario ingresa el ancho y alto, en pixeles, del rectángulo visible y el editor calcula el rombo que lo contiene. De esta forma, al exportar el mapa, el área visible puede ser calculada con matemática básica.

CAPÍTULO 4: VALIDACIÓN

Después de la construcción del editor de mapas fue necesario hacer una prueba con usuarios que permita determinar que, efectivamente, la herramienta puede ser utilizada para crear un nuevo nivel, y este puede ser jugado. Esta prueba siguió la misma metodología utilizada en la evaluación de la interfaz, planteada por Rubin y Chisnell (2008). A continuación se presenta el diseño y resultados de la prueba.

4.1 OBJETIVO

La prueba de validación final del editor de mapas presenta dos objetivos principales:

1. Reforzar la prueba de usabilidad realizada previamente para la interfaz gráfica.
2. Validar que el editor puede ser utilizado para crear un nuevo mapa, y este puede ser jugado en "1814: La rebelión del Cuzco".

4.2 PREGUNTAS DE INVESTIGACIÓN

Debido a que está prueba busca reforzar la evaluación de usabilidad hecha previamente con la interfaz, se reutilizó las preguntas planteadas en la sección 2.4.2, agregando las siguientes preguntas:

- ¿Qué problemas de usabilidad se presentan al exportar un mapa?
- ¿Es posible probar en el juego un mapa diseñado en el editor? ¿El mapa mantiene la lógica/sentido experimentado previamente en el juego?

4.3 CARACTERÍSTICAS DE LOS PARTICIPANTES

A diferencia de la prueba de usabilidad con la interfaz, en la que se estableció como restricción la experiencia previa con el juego, en esta solo se requiere que los participantes tengan entre 16 – 25 años. No obstante, los participantes fueron clasificados en dos perfiles que reflejan su experiencia en juegos de estrategia en tiempo real o similares, permitiendo validar la herramienta con un público más diverso. Los perfiles se definieron de la siguiente forma:

- Casuales: personas que han dedicado máximo 1 hora diaria a juegos de estrategia o similares, siendo su principal motivación pasar el tiempo o relajarse.
- Expertos: personas que han dedicado más de 1 hora diaria a juegos de estrategia o similares, participando de forma competitiva en ellos.

La distribución de los participantes y sus características se presenta en la Tabla 3, la cual no fue realizada en la prueba de la interfaz debido a la poca cantidad de participantes requeridos.

Tabla 3 Características de los participantes. (Elaboración propia)

Características	Número
Tipo de Participante	
Piloto	1
Prueba	8
Respaldo	2
Total	**8**
Tipo de jugador	
Casual	4
Experto	4
Total	**8**
Edad	
16-25	8
Total	**8**

4.4 DESCRIPCIÓN DEL MÉTODO

Las pruebas se realizaron de forma individual, de la misma forma que en la prueba de usabilidad de la interfaz. Además de tener instalado el editor y una grabadora de pantalla, la computadora portátil utilizada contó con el videojuego "1814: La rebelión del Cuzco".

La estructura de la prueba de validación consta de tres fases, previa aceptación del participante y firma del consentimiento informado, especificado en el Anexo B:

1. Pre-prueba: el participante resuelve un breve cuestionario diseñado para determinar su perfil como jugador.
2. Ejecución de tareas: el participante primero juega un nivel del juego y luego procede a realizar una lista de tareas con el editor. Finalmente, carga un mapa de diseño propio en el juego.
3. Post-prueba: el participante contesta un cuestionario después de acabar con las tareas, cuyo objetivo es recolectar información cualitativa y complementar los datos obtenidos previamente.

Se consideró importante permitir a los participantes experimentar el juego antes de utilizar el editor y probar su mapa. Esto asegura que los usuarios tengan un contexto previo y cuenten con una forma de comparar el mapa editado y su jugabilidad.

4.5 RESULTADOS DE LA EVALUACIÓN

Los resultados de los cuestionarios y la ejecución de las evaluaciones se pueden observar en el Anexo E. A continuación se presenta un resumen con los principales hallazgos que responden a las preguntas de investigación.

- Resultó claro analizando la Tabla 4 y Figura 11 que los comandos del editor para mover la cámara sobre el mapa no eran suficientes para permitir su fácil edición. Varios participantes comentaron otros mecanismos utilizados tanto en juegos como en programas de edición gráfica, de los cuales se implementó el movimiento arrastrando el mouse por la pantalla, al ser el más compatible con la configuración del editor.

- La forma de mover los objetos en el mapa recibió opiniones muy distintas entre los participantes. Algunos consideraban mejor utilizar el click izquierdo o hacer una selección previa del objeto, mientras que otros estaban de acuerdo con la configuración. Pese a esto, después de poco tiempo la acción les resultaba natural.

- Otro defecto de usabilidad no detectado en la evaluación de la interfaz fue la falta de algún indicador visual que diferenciara las unidades de cada bando. En editores comerciales este indicador forma parte de la imagen misma de la unidad. Sin embargo, los recursos gráficos del juego no poseen estos indicadores, por lo que se optó por dibujar círculos con el color del bando debajo de cada unidad.

- La funcionalidad de exportar no presentó ningún problema de usabilidad, principalmente porque realiza el proceso de integración al juego de forma automática y el usuario solo debe presionar un botón.

- Todos los mapas pudieron ser cargados en el juego. No obstante, hubieron problemas con la cantidad de unidades que podían actuar al mismo tiempo. En el caso de dos mapas se tuvo que reducir esta cantidad para probar la jugabilidad de forma completa.

- La mayoría de usuarios lograron completar las actividades sin ser ayudados y opinaron que fue sencillo utilizar el editor, como se muestra en la Figura 12, empero la apreciación acerca de la comodidad de los controles fue más segmentada. Esto se puede observar en la Figura 11, donde se clasifican las respuestas del cuestionario post-prueba por tipo de usuario.

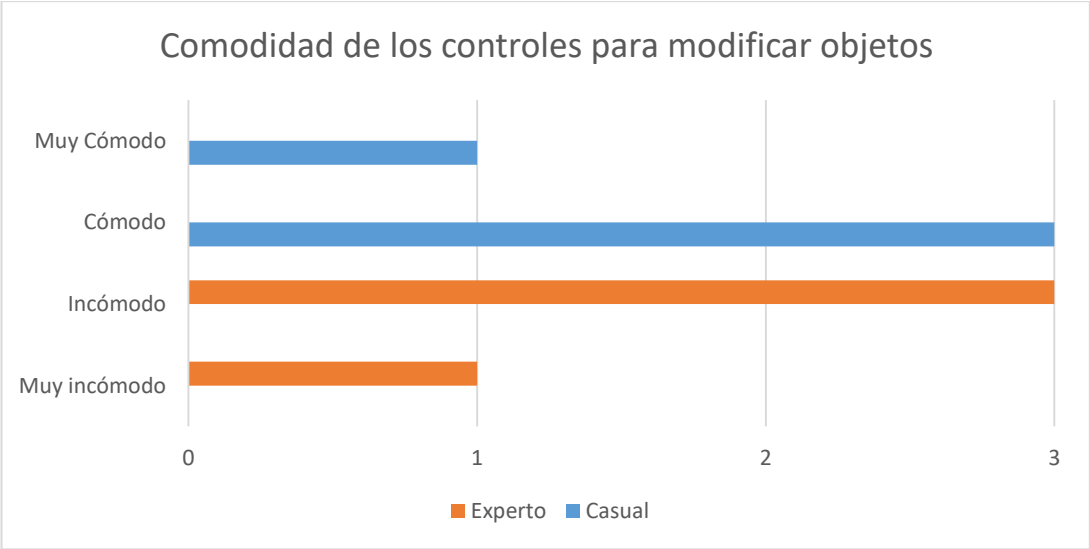

Figura 11 Comodidad de los controles para modificar objetos, según tipo de usuario. (Elaboración propia)

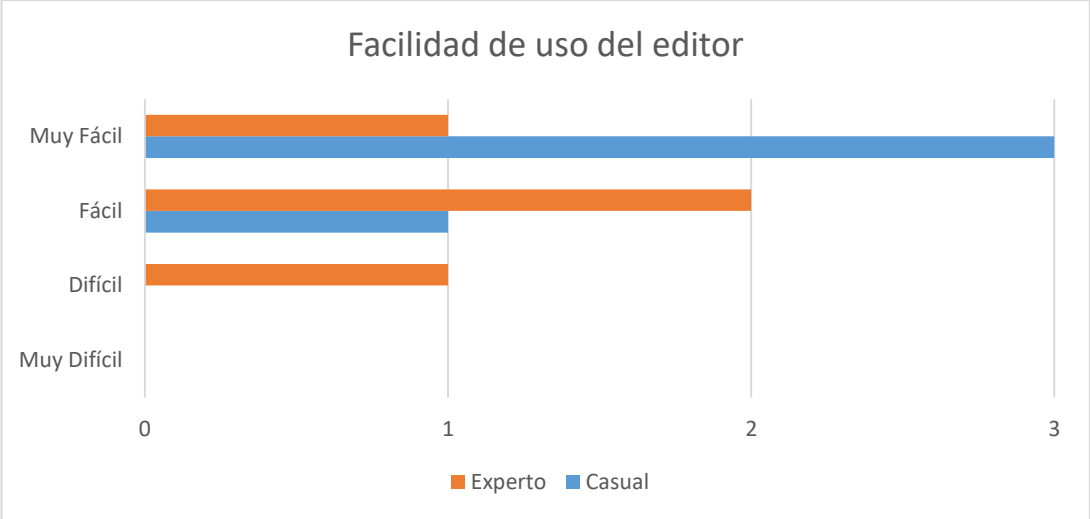

Figura 12 Facilidad de uso del editor, según tipo de usuario. (Elaboración propia)

- Al contrastar los resultados obtenidos en la Figura 11 con un resumen de los aspectos del editor que los participantes consideraron incómodos, expuesto en la Tabla 4, se observó lo siguiente: los usuarios casuales y expertos detectaron características o problemas muy similares, pero su opinión acerca de la comodidad fue distinta. Esto posiblemente indica una diferencia en el nivel de exigencia o importancia brindada a estos aspectos entre ambos perfiles.

Tabla 4 Aspectos que dificultaron el uso del editor, por usuario. (Elaboración propia)

N° Usuario	Perfil	Aspectos
1	Casual	• Tener que eliminar los objetos uno por uno • El movimiento de la cámara • No se puede distinguir el bando de las unidades
3	Casual	• Poco común el uso del scroll para girar los objetos • Distribución de las unidades y edificios en un árbol
6	Casual	• El movimiento de la cámara, sólo mediante el panel izquierdo
8	Casual	• No se puede distinguir el bando de las unidades • Usar más teclas para los comandos
2	Experto	• No poder mover el mapa con el scroll del mouse • No se puede distinguir el bando de las unidades
4	Experto	• Mover la cámara • El control para mover las unidades
5	Experto	• El movimiento de la cámara • Los botones en general • El uso del mouse en general
7	Experto	• El movimiento de la cámara • No se puede distinguir el bando de las unidades.

4.6 CONCLUSIONES

A partir de los resultados presentados en la sección anterior se concluyó lo siguiente:

- El editor puede ser utilizado fácilmente para crear un nuevo mapa y cargarlo al juego “1814: La rebelión del Cuzco”. Sin embargo, existen todavía detalles en ciertos controles o características que hacen la herramienta menos cómoda o atractiva para los usuarios, en especial los jugadores expertos.

- Los principales problemas de usabilidad en el editor fueron el movimiento de la cámara y la falta de indicadores visuales para los bandos de las unidades. Ambos problemas fueron corregidos junto con otros detalles como una mejor señalización sobre los elementos seleccionados.

- La diferencia en la opinión sobre la comodidad de los controles entre los jugadores casuales y los expertos fue casi contraria, y esto puede responder a varios factores. Uno de ellos es la costumbre generada en los usuarios expertos, después de haber jugado varios juegos de estrategia, sobre los controles que consideran más apropiados. Las modificaciones posteriores que se hagan al editor deben responder a estas costumbres, pero sin alienar al público casual.
- La mayoría de las observaciones encontradas en la prueba de la interfaz no se vieron repetidas en está validación. No obstante, algunos aspectos que fueron considerados menores previamente demostraron ser importantes en esta validación. Por otro lado, se descubrieron problemas de usabilidad que no habían sido identificados en la prueba de la interfaz gráfica.

CAPÍTULO 5: CONCLUSIONES Y TRABAJOS FUTUROS

En el presente capítulo se presentan las conclusiones y trabajos futuros asociados al proceso de desarrollo y evaluación del editor de mapas.

5.1 CONCLUSIONES

Las conclusiones a las que se llegaron, luego de finalizar el proyecto de fin de carrera son las siguientes:

- La interfaz gráfica elaborada para el editor, aunque no perfecta, demostró tener un grado de usabilidad aceptable gracias a la simplicidad de sus comandos y la disposición de los controles, similar a la de otros editores comerciales. Realizar una pequeña prueba con la interfaz antes de terminar el desarrollo del editor y luego una validación final resultó ser provechoso, al permitir detectar deficiencias en la usabilidad y solucionar aquellas que no impactaban directamente en el alcance.
- Para el desarrollo del renderizador isométrico las librerías de dibujo propias de Java fueron suficientes, gracias a la poca cantidad de elementos en movimiento que se manejan en el editor. Sin embargo, para un editor más avanzado que requiera de acciones como un "*zoom*" o visualización de animaciones se recomendaría trabajar con librerías especializadas en renderización de juegos, como OpenGL.
- Para el diseño de un compilador que utiliza un lenguaje visual es importante mantener cierta flexibilidad entre las fases definidas inicialmente. En el caso del editor, se tuvo que incluir algunas funciones del analizador semántico en el sintáctico. Esto se debió a la necesidad de proveer la mayor cantidad de retroalimentación en tiempo real para el usuario, en vez de esperar hasta la acción de exportar para realizar las validaciones. Al final, estas consideraciones dependerán del tipo de lenguaje que se esté empleando o, en el caso de editores de niveles, la complejidad de la lógica de juego.
- La aplicación de técnicas para el diseño de compiladores y lenguajes visuales resultó ser útil para la construcción del editor de mapas, permitiendo organizar su estructura y separar componentes por distintas funciones. No obstante, se considera que este enfoque es válido para un editor desarrollado por separado del juego asociado. En caso se construyan ambos en conjunto, la estructura debe adaptarse, principalmente obviando la fase de generación de código objeto.

- El objetivo general del proyecto se alcanzó de forma satisfactoria a través del cumplimiento de los objetivos específicos y sus resultados esperados: una interfaz gráfica usable y probada, un compilador que mantiene coherencia con la lógica del juego y una prueba de validación completa con usuarios.

5.2 TRABAJOS FUTUROS

En el proyecto de fin de carrera se logró construir exitosamente un editor de mapas para el videojuego “1814: La rebelión del Cuzco”. Sin embargo, debido a dificultades de tiempo y alcance no se implementaron ciertos módulos que posteriormente se podrían incluir:

- Editor de subniveles o interiores de edificios: extender la funcionalidad del editor para poder modificar los interiores de cada edificio, los cuales actualmente tienen un diseño de interior por defecto.
- Módulo de unidades: incluir un pequeño módulo que permita editar los atributos de las unidades y héroes. Esto permitiría también crear nuevas unidades e importar recursos gráficos.

Por otro lado, siguiendo las tendencias reflejadas en el mapeo sistemático, sería interesante permitir al editor exportar el mapa en un formato o estructura más genérica o especificada por el usuario con un metalenguaje. Esto lograría que la herramienta puede ser utilizada en el ciclo de desarrollo de futuros videojuegos o en conjunto con otros motores. Un ejemplo de este tipo de editor, pero con formatos específicos, es la herramienta Tiled (Lindeijer, n.d.), la cual permite editar y exportar mapas ortogonales, isométricos y hexagonales a motores comerciales como Unity y Cocos2D.

BIBLIOGRAFÍA

Age of Empires II HD. (2015). Retrieved October 1, 2015, from http://www.ageofempires.com/news/games/aoeii/

Aho, A. V., Sethi, R., Ullman, J. D., Flores Suárez, P., & Botella i López, P. (1990). *Compiladores: principios, técnicas y herramientas*. Wilmington, Delaware: Addison-Wesley Iberoamericana.

Ballinger, C., & Louis, S. (2013). Comparing heuristic search methods for finding effective real-time strategy game plans. In *2013 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)* (pp. 16–22). http://doi.org/10.1109/CISDA.2013.6595422

Blizzard Entertainment: StarCraft. (2015). Retrieved September 30, 2015, from http://us.blizzard.com/en-us/games/sc/

Cai, L., & Chen, Z. (2010). Design and Implementation of OGRE-Based Game Scene Editor Software (pp. 1–4). IEEE. http://doi.org/10.1109/CISE.2010.5676829

Chang, S.-K. (1999). Visual Languages. In *Wiley encyclopedia of electrical and electronics engineering* (Vol. 23, pp. 265–275). New York: John Wiley.

Cowan, B., & Kapralos, B. (2011). A simplified level editor (pp. 52–54). IEEE. http://doi.org/10.1109/IGIC.2011.6115130

Cox, P., Gauvin, S., & Rau-Chaplin, A. (2005). Adding parallelism to visual data flow programs (pp. 135–144). Presented at the Proceedings SoftVis '05 - ACM Symposium on Software Visualization. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-28044448058&partnerID=40&md5=ed268d635357bcdc3b2c6d4de548421d

Drey, Z., & Consel, C. (2010). A visual, open-ended approach to prototyping ubiquitous computing applications (pp. 817–819). IEEE. http://doi.org/10.1109/PERCOMW.2010.5470549

Dumitrescu, A. (2012). Sharing Led to Little Big Planet Success, 6.7 Million Levels Created. Retrieved September 14, 2015, from

http://news.softpedia.com/news/Sharing-Led-to-Little-Big-Planet-Success-6-7-Million-Levels-Created-278113.shtml

Edge. (2008, April 1). Retrospective: Ten Years of Starcraft. Retrieved March 27, 2016, from http://web.archive.org/web/20120522171031if_/http://www.computerandvideogames.com/export/edge/epp.php

Graft, K. (2012, October 16). User-generated content: When game players become developers. Retrieved October 1, 2015, from http://www.gamasutra.com/view/news/179493/Usergenerated_content_When_game_players_become_developers.php

Grigorenko, P., Saabas, A., & Tyugu, E. (2005). COCOVILA – Compiler-Compiler for Visual Languages. *Electronic Notes in Theoretical Computer Science*, *141*(4), 137–142. http://doi.org/10.1016/j.entcs.2005.05.009

Guerra, B. (2014). 1814: La Rebelión del Cusco (actualizado con instalador) :: Grupo Avatar PUCP. Retrieved from http://avatar.inf.pucp.edu.pe/cusco-1814/

Guerra, E., Lara, J. D., & Malizia, A. (2007). Model driven development of digital libraries-Validation, analysis and code generation (Vol. WIA, pp. 35–42). Presented at the Webist 2007 - 3rd International Conference on Web Information Systems and Technologies, Proceedings. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-67649982129&partnerID=40&md5=703bc34dd68ae11b7db7c36910288253

ISO 9241. (2010). ISO 9241-210:2010 - Ergonomics of human-system interaction -- Part 210: Human-centred design for interactive systems. Retrieved from http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=52075

Kastens, U., & Schmidt, C. (2006). Visual Patterns Associated to Abstract Trees. *Electronic Notes in Theoretical Computer Science*, *148*(1), 5–18. http://doi.org/10.1016/j.entcs.2005.12.010

Krikke, J. (2000). Axonometry: a matter of perspective. *IEEE Computer Graphics and Applications*, *20*(4), 7–11. http://doi.org/10.1109/38.851742

Kuuskankare, M., & Laurson, M. (2007). VIVO - Visualizing harmonic progressions and voice-leading in PWGL (pp. 289–290). Presented at the Proceedings of the 8th International Conference on Music Information Retrieval, ISMIR 2007. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-84873579210&partnerID=40&md5=02874aed12520ca9d69295a8a9771bcd

Lindeijer, T. (n.d.). Tiled Map Editor. Retrieved May 20, 2016, from http://www.mapeditor.org/

Maier, S., & Volk, D. (2008). Facilitating Language-oriented Game Development by the Help of Language Workbenches. In *Proceedings of the 2008 Conference on Future Play: Research, Play, Share* (pp. 224–227). New York, NY, USA: ACM. http://doi.org/10.1145/1496984.1497029

Msiska, M. F., & van Zijl, L. (2012). From visual scripting to Lua (p. 94). ACM Press. http://doi.org/10.1145/2389836.2389848

Nielsen, J. (2012, January 4). Usability 101: Introduction to Usability. Retrieved November 6, 2012, from http://www.nngroup.com/articles/usability-101-introduction-to-usability/

Nummenmaa, J., Marttila-Kontio, M., & Nummenmaa, T. (2013). Checking visual data flow programs with finite process models (pp. 245–258). Presented at the 13th Symposium on Programming Languages and Software Tools, SPLST 2013 - Proceedings.

Oswald, P., Tost, J., & Wettach, R. (2014). The real augmented reality: real-time game editor in a spatial augmented environment. In *Proceedings of the 11th Conference on Advances in Computer Entertainment Technology* (p. 32). ACM.

Oxford Dictionaries. (n.d.). “render”. Oxford Dictionaries. Retrieved from http://www.oxforddictionaries.com/definition/english/render

Pautasso, C., & Alonso, G. (2005). The JOpera visual composition language. *Journal of Visual Languages and Computing*, *16*(1-2 SPEC. ISS.), 119–152. http://doi.org/10.1016/j.jvlc.2004.08.004

Reilly, B. L. (2012). Why Games That Make YOU a Developer Are a Crucial Part of the Future. Retrieved September 14, 2015, from

http://www.ign.com/articles/2012/08/27/why-games-that-make-you-a-developer-are-a-crucial-part-of-the-future

Rubin, J., & Chisnell, D. (2008). *Handbook of usability testing: how to plan, design, and conduct effective tests* (2nd ed). Indianapolis, IN: Wiley Pub.

Snow, B. (2011, August 17). Why most people don't finish video games. Retrieved October 1, 2015, from http://www.cnn.com/2011/TECH/gaming.gadgets/08/17/finishing.videogames.snow/index.html

Tassi, P. (2015). "The Order: 1886" Developer Makes A Flawed Argument About Game Length. Retrieved September 21, 2015, from http://www.forbes.com/sites/insertcoin/2015/02/16/the-order-1886-developer-makes-a-flawed-argument-about-game-length/

Vickery, G., Wunsch-Vincent, S., & Organisation for Economic Co-operation and Development (Eds.). (2007). *Participative Web and user-created content: Web 2.0, wikis and social networking*. Paris: Organisation for Economic Co-operation and Development.

Wolf, M. J. P. (Ed.). (2012). *Encyclopedia of video games: the culture, technology, and art of gaming*. Santa Barbara, Calif: Greenwood.