Technical Design Document

Picky Rats

CITM 2024

CONTENIDO

1.VISIÓN GENERAL

1.1.DESCRIPCIÓN GENERAL DEL PROYECTO

El videojuego que finalmente se ha decidido crear es un tipo RPG ambientado en el Antiguo Egipto. Además queremos fomentar la exploración y priorizar el factor evolutivo del personaje con distintas mejoras. El tipo de combate es por turnos.

Básicamente, nuestro protagonista es un esclavo al cual le cae un escupitajo divino que le da poderes. Lo capta un grupo contrario al régimen tirano del esclavista que quiere entrar a la pirámide. En esta aventura en la pirámide, el protagonista encontrará desafíos y respuestas a sus preguntas.

1.2.OBJETIVO

El objetivo de nuestro juego es entretener a los jugadores y que sea un juego divertido. Creemos que un juego ambientado en Egipto se aleja drásticamente de lo que solemos ver en los distintos juegos RPG, que siempre están ambientados en la Edad Media.

2.REQUISITOS DE DESARROLLO

2.1.LISTA DE FEATURES

Sistema de combate

El sistema de combate elegido será por turnos, los jugadores tendrán la oportunidad de realizar dos acciones principales: moverse y atacar. Para llevar a cabo un ataque exitoso, el jugador deberá posicionarse dentro del rango adecuado para la habilidad que desee utilizar.

Modo exploración

La exploración será la actividad central del jugador a lo largo del juego, dado que se trata de un juego en 2D. La exploración se asemejará a un juego de plataformas, con diversos niveles que el jugador podrá recorrer. Una característica adicional será la inclusión de puertas perpendiculares al camino principal. Estas puertas conducirán a otras habitaciones, por lo tanto se requerirá que el jugador se desvíe del camino principal para acceder a ellas.

Puzzles

Durante la exploración, el jugador se enfrentará a diversos obstáculos en forma de puzzles. Estos puzzles variarán en su naturaleza y se resolverán mediante diferentes mecánicas de juego. Conforme el jugador avance, la complejidad de estos puzzles irá en aumento, ofreciendo un desafío cada vez mayor.

Inventario

El sistema de inventario proporcionará al jugador acceso a diversas opciones. Podrá verificar el equipamiento del personaje seleccionado, revisar sus habilidades y estadísticas, y explorar una sección donde se listarán los objetos clave obtenidos durante el transcurso del juego. Como un añadido adicional, habrá un espacio reservado en el inventario para que cada vez que el jugador seleccione un objeto clave, reciba información adicional sobre él.

Sistema de misiones

El sistema de misiones del juego constará de dos tipos principales: misiones principales y misiones secundarias, cada una con su propio propósito y contribución a la experiencia del jugador. Las principales te harán avanzar en la historia, mientras que las secundarias harán al jugador descubrir cosas inesperadas.

Selección de Dioses

Al comenzar el juego, los jugadores tendrán la oportunidad de elegir entre cuatro tipos de dioses, lo que determinará por completo su experiencia de juego. Cada dios ofrecerá un enfoque único en términos de jugabilidad y estadísticas.

Sistema de partículas

El sistema de partículas en el juego ofrece una experiencia visualmente impresionante, agregando efectos adicionales que sumergen aún más al jugador en el mundo del juego.

Gamepad

El sistema de gamepad será una funcionalidad adicional que permitirá al jugador disfrutar del juego utilizando cualquier tipo de mando, además del teclado.

3.PLATAFORMA OBJETIVO Y REQUISITOS GENERALES

El juego será programado en su mayor parte en C++ utilizando la librería SDL.

SDL es un conjunto de librerías que proporcionan unas funciones básicas para realizar operaciones de renderizado y dibujado 2D, gestionar efectos de sonido y cargar imágenes.

3.1.PLATAFORMAS JUGABLES

El juego estará disponible en ordenador, PC Windows, pudiendo jugar con mando de PlayStation o Xbox además de teclado y ratón.

3.2.REQUISITOS DE HARDWARE

Para poder jugar a nuestro juego será necesario disponer de un teclado y ratón o por el contrario, con un mando de PlayStation o Xbox. La resolución en la que vamos a desarrollar el juego es 720p así que mínimo el monitor debe soportar esta resolución.

Igualmente, vamos a desarrollar un sistema de adaptación de pantalla fullscreen dependiendo de la resolución de cada uno, para que sea más cómodo jugar.

El juego funciona a 60 FPS con función activa de vSync. El vSync se puede activar y desactivar. De igual manera, con el vSync desactivado, el control de fotogramas queda inactivo.

Componentes	Requerimientos
Sistema operativo	Windows 7 o superior
Procesador(CPU)	Intel Core i3 / AMD Ryzen 3
Tarjeta de video (GPU)	AMD Radeon R7 200 Series o superior
Memoria RAM	100 MB
Almacenamiento	200 MB

4.ARQUITECTURA Y LENGUAJE DE PROGRAMACIÓN

4.1.ARQUITECTURA

En lo que a la planificación del código se refiere, vamos a usar distintas clases durante el desarrollo de nuestro juego para que funcione de la manera más eficiente posible. La clase principal será la del Player, donde se creará todos los controles del jugador que controlamos, además, se crearán las funciones con todo lo que permite hacer al jugador.

Por otro lado, se creará la clase padre de Enemigos, de la cual todos los distintos enemigos que nos encontremos, heredarán. Lo mismo sucede con la clase de NPC 's.

Por otro lado, se crearán las clases más técnicas como por ejemplo la clase de físicas, la del mapa, la de renderizado, la de audios, la de texturizado, la de la ventana, el pathfinding de los enemigos y las de las distintas entidades.

4.2.LENGUAJE DE PROGRAMACIÓN

Nuestro juego va a ser desarrollado en el lenguaje C++ y usando la librería de Simple DirectMedia (SDL). Esta librería es compatible con Windows, macOS, Linux, iOS, y Android.

SDL nos permite realizar operaciones de dibujo en dos dimensiones, gestión de efectos de sonido y música, además de carga y gestión de imágenes.

Existen distintas extensiones de SDL, las que vamos a utilizar son las siguientes:

SDL_IMAGE permite a los desarrolladores cargar imágenes en distintos formatos sin tener que escribir código específico para cada formato.

SDL_MIXER dota a SDL de funcionalidades que permiten manipular y reproducir sonidos y música. Hay opciones de controlador de volumen y velocidad de reproducción.

SDL_TTF permite a SDL cargar y mostrar texto utilizando fuentes TrueType de manera sencilla y eficiente, también renderiza diferentes estilos, tamaños y colores.

Otra biblioteca externa que vamos a usar es Box2D, esta, es una biblioteca que se encarga de las físicas de un videojuego en 2 dimensiones, permite simular la aplicación de fuerzas en distintos momentos de la partida.

Se implementa la librería de Optick, un profiler para C++ que nos permite hacer análisis de rendimiento y optimización del juego.

Disponemos de la librería FFMPEG para poder implementar las distintas cinemáticas que habrá en el juego.

La librería PUGIXML es una librería externa que permite analizar y manipular documentos XML.

5.HERRAMIENTAS NECESARIAS

5.1.TILED

Tilde es un programa de software de código abierto utilizado principalmente para crear mapas para videojuegos en 2D. Adicionalmente, es una herramienta profesional que permite a los diseñadores de juegos construir y editar entornos de juego mediante la disposición de tiles, que son pequeñas imágenes cuadradas que componen los elementos del mapa, como suelo, paredes, objetos y otros elementos decorativos. Para el desarrollo del videojuego se ha utilizado la versión 1.10.2 del programa en un formato basado en XML.

5.2.VISUAL STUDIO

Visual Studio es un entorno de desarrollo integrado. Su utilidad se enfoca en la programación, dedicado a la escritura, la depuración y la compilación en una variedad de lenguajes de programación. En concreto, estos lenguajes incluyen C#, C++, F#, Python y otros.
En cuanto al desarrollo del proyecto, se ha optado a usar lla versión 2022 Community Edition como entorno de desarrollo. con una integración de Git que facilita la colaboración y el control de versiones.

5.3.GITHUB

Github es una plataforma de desarrollo colaborativo basada en la web, que utiliza el sistema de control de versiones Git. Es una de las herramientas más populares para el desarrollo de software colaborativo y código abierto. En cuanto al proyecto, se ha usado la versión 3.3.8 de la plataforma para el desarrollo.

5.4. ADOBE PHOTOSHOP

Adobe Photoshop es un software de edición de imágenes. Este es ampliamente utilizado por diseñadores gráficos, fotógrafos, artistas digitales y profesionales de medios visuales para editar y manipular imágenes. En el desarrollo del proyecto se ha enfocado su uso para la creación artística del juego, que incluye personajes, sprites, objetos, paisajes, y texturas. En cuanto a la versión utilizada, se ha optado por utilizar la 25.5.1.

5.5. AUDACITY

Audacity es un programa de software de código abierto diseñado para la grabación y edición de audio digital. Su uso se focaliza principalmente en la producción de audio y es una herramienta popular en el ámbito musical. Consecuentemente, se ha escogido usar la versión 3.4.2 para el desarrollo del proyecto, y se ha focalizado su uso para edición de sonidos del juego como efectos musicales y fx.

5.6.TOON BOOM HARMONY 22 PREMIUM

Toon Boom Harmony es un software utilizado en la industria de la animación 2D y 3D para la creación de películas, series de televisión, comerciales y otros proyectos animados.

Para el desarrollo del proyecto se ha utilizado la edición Toon Boom Harmony 22 PREMIUM, concretamente la versión 21.1 del programa. Adicionalmente, se ha enfocado su uso para la creación de animaciones de los personajes del juego.

5.7.ADOBE AFTER EFFECTS

Adobe After Effects es un software de composición y postproducción de vídeo. Es ampliamente utilizado en la industria del cine, la televisión y producción de medios digitales. Consecuentemente, para el desarrollo del proyecto se ha enfocado su uso para crear efectos visuales, animaciones, gráficos en movimiento y otros elementos visuales dinámicos. Adicionalmente, la versión utilizada es la 24.2

5.8.MEDIBANG PAINT

Medibang Paint es una aplicación de software de creación de arte digital enfocada a la ilustración digital, el pintado digital, el diseño y conceptualización de personajes, y la edición de imágenes.

Durante el desarrollo del proyecto se ha utilizado la versión 29.1 para crear la concepción y el diseño de los personajes.

5.9. CLIP STUDIO PAINT

Clip Studio Paint es un software de creación de arte digital popular entre los artistas digitales, ilustradores y animadores. En cuanto a su uso, se focaliza principalmente en ofrecer herramientas específicas para la creación de cómics y manga.

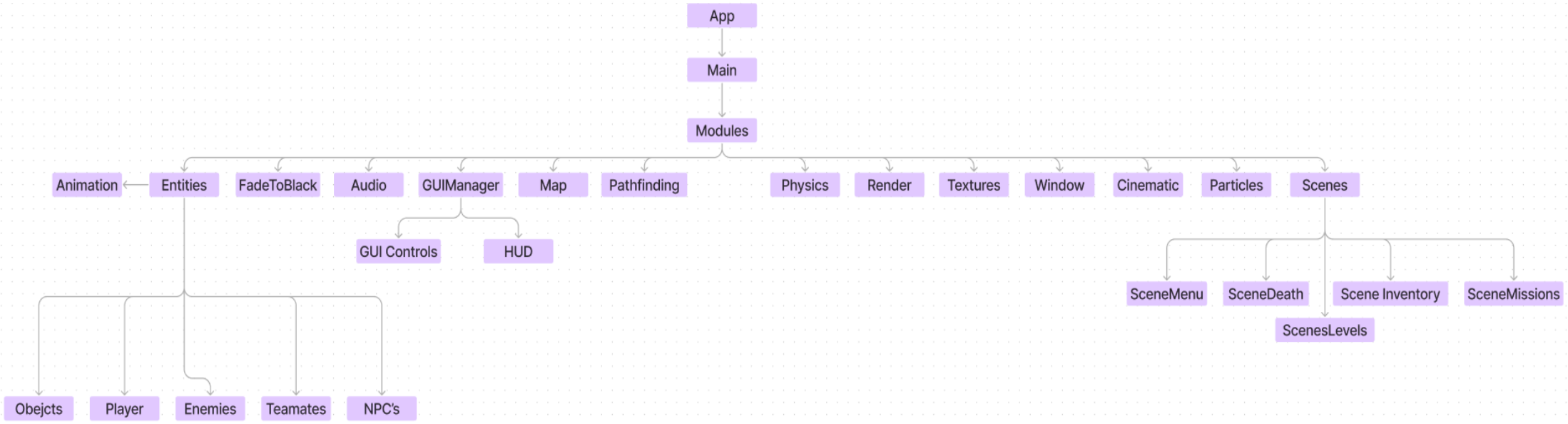
En cuanto al desarrollo del proyecto, la versión utilizada es la 2.3.4. Consecuentemente, se usará exclusivamente para la creación de concept y diseño de personajes.

5.10. PAINTTOOL SAI

Paint tool SAI o SAI es un software de dibujo y pintura digital que se usa popularmente para ilustrar y diseñar personajes. Dispone de herramientas de pintura, capas y modos de fusión o herramientas de selección y transformación.

Consecuentemente, para el desarrollo del proyecto se ha utilizado la versión 1.1.0 y se ha enfocado su uso para crear personajes.

6.DIAGRAMA DE ARQUITECTURA DE NIVEL ALTO



7.DESAFÍOS TÉCNICOS Y RIESGOS

Los desafíos técnicos a los que nos podemos enfrentar son los siguientes:

Balance de Gameplay

Uno de los problemas que nos podríamos encontrar es el no saber encontrar un balance en el gameplay por la limitación a nivel temporal, puesto que el tipo de juego que se va a desarrollar suele beneficiarse de tener una gran cantidad de armas, enemigos, y escenarios.

Progresión de dificultad

Otro problema con el que nos podríamos encontrar es buscar el punto correcto entre una progresión de dificultad acertada, que desafíe al jugador pero no lo frustre hasta el punto de que no lo quiera jugar. Para ello, se debe crear habilidades, mecánicas y mapas correctamente balanceados de manera progresiva. Teniendo en cuenta la falta de tiempo, encontrar ese punto medio podría ser un desafío considerable.

Implementación artística

Teniendo en cuenta las limitaciones técnicas que supone el motor que estamos utilizando, la implementación de todas las creaciones artísticas puede suponer un problema a nivel de cinemáticas o efectos de profundidad.

Problemas de pathfinding

Con la gran cantidad de enemigos que se podrían implementar para aumentar la jugabilidad, se debería trabajar en profundidad el pathfinding.

Implementación de librerías externas

Para crear una experiencia de juego completa, se intentarán incluir la mayor cantidad de librerías externas. Sin embargo, se tendrán que implementar de la mejor manera posible para evitar que fallen.

Organización de archivos

Además, teniendo en cuenta la gran cantidad del equipo, se tendrá que llevar una buena organización de los documentos y elementos del juego, para que sean fáciles de encontrar, archivar, y evitar malentendidos por duplicación de archivos.

Memory Leaks

Finalmente, considerando el tamaño del juego, habrá que evitar problemas de Memory Leaks.

8.MECÁNICAS DEL JUEGO

Las mecánicas principales que vamos a implementar en nuestro juego son las de moverse, subir y bajar escaleras.

En lo que al movimiento del jugador principal se refiere, para ir hacia la derecha o la izquierda, se sumará o restará un valor determinado a la velocidad en el eje x al personaje principal, lo que hará que se mueva hacia un lado u otro.

Para subir o bajar escaleras, se aplicará una velocidad constante en el eje Y cuando entre en contacto con una colisión de tipo escalera, hasta que alcance una que no sea de este tipo.
En lo que respecta a la interacción de objetos, esto se implementará de manera que cuando se entre en contacto con la respectiva colisión del objeto, aparecerá un texto que nos dirá que se puede interactuar con el objeto para guardar en el inventario, equipar o para usarlo.

También se permitirá al usuario abrir el inventario, pulsando una tecla o botón asignada, se abrirá un menú de inventario en el que administrar los objetos que se poseen.

En el menú de inventario, el movimiento será mediante las flechas del teclado o los joysticks, buscando favorecer la velocidad y eficiencia en este menú. Esto mediante código se crearán los distintos tipos de botón y sus estados. Cada botón tendrá asignado un id para identificar correctamente dónde está seleccionando el jugador.
Los huecos de inventario se crearán a través de una matriz. De esta forma, los objetos tendrán posiciones en el eje X e Y.

En el apartado de interacción con NPC 's se creará de igual manera que el objeto, cuando se pulse el botón de interactuar, empezará una serie de diálogos.

En lo que combate se refiere, será un combate por turnos. Esto permite a los jugadores tener un factor de estrategia en la partida. Durante el combate podrá moverse y atacar. Para esta sección, se creará un puntero que se moverá con las teclas WASD o con las flechas del mando. Dicho puntero tendrá un pathfinding que detecte las tiles desde el jugador hacia el puntero. Este valor se imprimirá por pantalla para ver los puntos que se gastan si se decide mover el jugador.

A la hora de atacar, se escoge una habilidad con un botón asignado, que una vez se use, depende de su status, tendrá tiempo de reutilización. Cada ataque tendrá su coste determinado.

9.NIVELES

A la hora de hacer los distintos niveles se crearán mediante el programa de Tiled. En dicho programa, será necesario disponer de un tileset para poder pintar el mapa a gusto del artista.
En este programa es muy importante utilizar las distintas capas para los fondos, plataformas, objetos, pathfinding y colisiones. De esta manera, posteriormente será más sencillo cargarlo por código.

Una vez hecho y exportado el mapa a gusto del encargado, se obtiene un archivo xml que será de donde se lea la información a cargar desde código.

Por código se obtiene información del mapa para cargarlo correctamente. Se conocen las medidas de altura y anchura del mapa, información de pathfinding, colisiones y dibujo de cada tile.

Finalmente se carga el mapa entero mediante distintos bucles y procesos para cada capa y apartado del mapa.

10. FÍSICAS

El sistema de físicas que encontramos en nuestro juego se proporciona gracias a la librería externa de Box2D, esto permitirá dotar de más realismo a nuestro juego.

Con dicha librería, podremos aplicar gravedad al escenario, haciéndolo de manera más realista posible, de igual manera, se aplicarán distintos tipos de fuerza durante el juego, como por ejemplo en los ataques, que se podría llegar a aplicar una especie de retroceso a la hora de recibir daño.

10.1. COLISIONES

En lo que a colisiones se refiere, habrá definidas en el módulo de físicas de nuestro juego distintos tipos de colisiones dependiendo del elemento al que pertenezcan. A priori podríamos tener las colisiones del jugador, aliados, enemigos y sus respectivos ataques, los items, las paredes…

Una vez definidos los tipos de colisiones, se crea un método que se activa cuando dos tipos de colisiones se chocan, de esta manera, se puede iniciar una función en caso de que dos objetos colisionan entre sí.

10.2.MOVIMIENTO

El movimiento en nuestro juego será en horizontal y en vertical, es decir, el personaje se puede mover de izquierda a derecha y también le es permitido saltar, subir escaleras y bajarlas, por lo que en el salto se le deberá aplicar una fuerza para que la simulación sea correcta.

11.IA

La inteligencia artificial (IA) implementada en nuestro juego será evidente durante los enfrentamientos. Utilizaremos una IA de combate basada en reglas, la cual se activará cuando sea el turno del enemigo. Durante estos momentos, la IA seguirá un conjunto de reglas predefinidas para tomar decisiones.

En primer lugar, evaluará si se encuentra dentro del rango de ataque del jugador. En caso afirmativo, verificará si dispone de los recursos necesarios, como estamina o maná, para ejecutar un ataque.

Si no se encuentra dentro del alcance apropiado, la IA realizará los movimientos necesarios para posicionarse estratégicamente y, posteriormente, evaluará nuevamente la posibilidad de ejecutar un ataque.

Según el tipo de enemigo puede evaluar cuál es su objetivo ya sea por proximidad, vida restante u otros elementos destacados.

También se implementará un sistema de IA para ayudar al jugador en el modo de posicionamiento, ya que se le mostrará mediante pathfinding que rutas puede seguir y el estado de sus puntos de combate, es decir, podrá saber en todo momento cual es su estado final antes de realizar el movimiento.

12.DETALLES Y ESPECIFICACIONES DE AUDIO

Para integrar el audio y los efectos de sonido en nuestro juego desarrollado en C++ con SDL, utilizaremos las capacidades de SDL_mixer. Primero, configuraremos SDL_mixer en nuestro proyecto, asegurándonos de incluir las bibliotecas necesarias y configurar correctamente el sistema de audio.

Luego, procederemos a cargar los archivos de audio en formatos compatibles con SDL_mixer, como WAV, OGG o MP3, utilizando funciones proporcionadas por la biblioteca. La reproducción de estos archivos se activará en respuesta a eventos específicos del juego, como acciones del jugador o eventos de juego predeterminados. Además, SDL_mixer nos permite controlar diversos aspectos del audio, como el volumen, efectos de sonido y posicionamiento espacial en tiempo real, lo que nos permitirá crear una experiencia auditiva inmersiva y dinámica para el jugador.

13.ORGANIZACIÓN DEL CÓDIGO Y REQUISITOS

Para la correcta organización del proyecto, se pondrá todo en distintas carpetas para tener todos los documentos bien distribuidos y estructurados, al hacerlo de esta forma, los distintos tipos son más sencillos de encontrar.

Las carpetas dentro del Visual Studio están divididas según el contenido del archivo:

- **Modules** → Todos los archivos de módulos
    - **Scenes** → Escenas del juego
    - **UI** → Modulos de UI
- **Entities** → Todas las entidades como el player o los enemigos.
- **Tools y Útiles** → Herramientas adicionales como el timer o archivos como listas.

Las carpetas en la solución del proyecto están divididas de la siguiente manera:

- **Game**
    - Source → Ficheros de código
        - External → Librerías externas
- **Output**
    - Assets
        - Audio → Música y FX que encontramos en el juego.
        - Fonts → Toda la tipografía usada en menús y diálogos.
        - Maps → Mapas que se deben cargar realizados mediante Tiled.
        - Textures → Sprites de personajes y objetos.

Tener una buena organización del proyecto ayuda a tener un mejor control y distribución de los diferentes archivos.

14.SOURCE CONTROL POLICY

Git es un sistema de control de versiones distribuido que registra los cambios realizados en un archivo a lo largo del tiempo. Esto te permite recuperar versiones específicas más adelante. Permite trabajar en ramas separadas y fusionar los cambios a la rama principal mediante pull request.

Tendremos una rama principal (main) y otra de development a partir de la que crearemos ramas separadas para que cada programador pueda trabajar en sus objetivos, cuando se deba fusionar a la rama principal el lead de programación supervisará para que no ocurran problemas y así tener controlados los cambios que suceden a la rama principal.

Esta herramienta será muy útil para tener un buen flujo de trabajo, poder hacer un trabajo cooperativo y también para poder ver los cambios que ocurren en el código.

15.CODING GUIDELINES

Para las directrices de codificación de este proyecto seguiremos el c++ style guide CppCoreGuidelines

16.QUALITY PLAN

Para que todo pueda salir adelante, vamos a organizarnos de manera que cada coder va a tener unas tareas específicas asignadas, cuando éste complete la tarea pasará a estar en estado de revisión, por lo que el lead va a revisar que esté todo correcto (todo lo que debe estar implementado, que funcione bien el merge, que no haya errores…) y así poder hacer releases con las nuevas implementaciones.

De esta manera se puede tener un control del código y nos aseguramos que todo funcione correctamente.

Bug report:

Se asignará un rol a varias personas para hacer playtesting a menudo para comprobar la correcta implementación de las nuevas funcionalidades del juego y así poder detectar si hay algún error.

Se utilizará un documento para hacer un registro de los bugs que se detecten en el juego siguiendo una plantilla:

- Bug: Nombre del bug
- Descripción: Breve descripción sobre el error que se ha encontrado.
- Situación: Como se ha encontrado el bug y que debemos hacer para replicarlo.

- Solución: Pasos a seguir para resolver el error.