UNIVERSIDAD DE COLIMA

Facultad de Telemática

Ingeniería de Software

Desarrollo de videojuegos

Individual: Technical Design Document

7 ° I

Elabora:

Ponce Ricarte Isaac Eduardo

Catedrático:

Rodriguez Ortiz Miguel Angel

Colima, Colima; Martes 01 de Octubre de 2024

1. Lista de características obtenidas del GDD

- **Historia y Ambientación**: Juego de peleas 2D ambientado en un México precolonial, donde un guerrero maya enfrenta conquistadores españoles y seguidores del oscuro culto "El Milagro".
- **Personaje principal**: Guerrero maya que usa armas tradicionales y habilidades místicas otorgadas por dioses mayas.
- **Enemigos y Jefes**: Conquistadores y figuras como sacerdotes e inquisidores con elementos sobrenaturales. El jefe final es una versión oscura de Cristóbal Colón.
- **Estilo visual**: Arte pixelado con un tono sombrío y elementos sobrenaturales.
- **Mecánicas de juego**: Combates basados en combos y defensa, habilidades místicas recargables, y enfrentamientos con enemigos cada vez más poderosos.

2. Elección de Game Engine

- Me he decantado por **Godot Engine** debido a sus capacidades de desarrollo en 2D, su rendimiento en plataformas PC, y sus herramientas avanzadas para manejar gráficos pixel art y físicas simples. Además, su código abierto y comunidad activa lo hacen una opción adecuada para un juego de este tipo.

3. Planeación (Diagrama de Gantt)

Fecha	Semana	Fase 1: Preproducción	Fase 2: Desarrollo	Fase 3: Testing y Ajustes	Fase 4: Pulido y Lanzamiento
1/10 - 6/10	Semana 1	Redacción del GDD y TDD			
7/10 - 13/10	Semana 2	Definición de arte conceptual (asets)			
14/10 - 20/10	Semana 3	Prototipo en Godot	Mecánicas básicas de combate		
21/10 - 27/10	Semana 4		Diseño de personajes y enemigos		
28/10 - 3/11	Semana 5		Desarrollo de niveles y jefes		
4/11 - 10/11	Semana 6		Integración de audio	Pruebas de jugabilidad	
11/11-17/11	Semana 7			Corrección de bugs	
18/11-24/11	Semana 8			Ajustes de balance y dificultad	Mejora de comentarios del profesor
25/11 - 1/12	Semana 9				lanzamiento o finalización

4. Diagramas de alto nivel

- **Diagrama de Arquitectura del Sistema**: Se utilizará una arquitectura de juego en capas:
    - **Capa de Presentación**: Menús, HUD, y pantallas de selección.
    - **Capa de Lógica de Juego**: Mecánicas de combate, detección de colisiones, y lógica de IA.
    - **Capa de Datos**: Gestión de estados de los personajes, enemigos, y progresión.
- **Diagrama de Componentes**: Desglosa los sistemas de combate, control de jugadores, física y colisiones.

5. Herramientas de arte

- **Pixel Studio**: Para la creación de sprites y animaciones en pixel art, para diseñar los niveles y entornos en 2D.
- **Photoshop**: Para ediciones y ajustes finos de texturas.
- **kenney:** Para búsqueda de assets

6. Objetos 3D, Terreno y Escenas

- Aunque el juego es en 2D, algunas escenas pueden incluir elementos de fondo y perspectiva que se gestionarán con capas para crear profundidad en los escenarios.
- **Escenas**: Diseñadas como ambientes temáticos basados en México precolonial, como junglas y ruinas mayas.

7. Detección de colisiones, físicas e interacciones

Godot incluye un sistema robusto para la detección de colisiones, usando cuerpos físicos como *KinematicBody2D*, *RigidBody2D*, y *Area2D*. Cada uno tiene un propósito distinto:

- **KinematicBody2D:** Se utiliza para personajes controlados por el jugador, manejando su movimiento mediante la función `move_and_slide()` o `move_and_collide()`. Estas funciones permiten detectar colisiones y ajustar el movimiento del personaje.
- **RigidBody2D:** Simula cuerpos bajo las leyes físicas, permitiendo que la gravedad y las colisiones sean manejadas automáticamente.
- **Area2D:** Detecta cuerpos en su área de influencia, útil para crear triggers o áreas de efecto.

Técnicas utilizadas:

- **Colisión por bounding boxes:** Se usarán *CollisionShape2D* o *CollisionPolygon2D* para definir las formas de colisión de los objetos. Estas formas simplificadas permiten optimizar el rendimiento del juego al comparar formas geométricas en lugar de cada píxel del sprite.
- **Sistemas de máscaras y capas:** Godot permite asignar máscaras y capas a los objetos para determinar con qué otros pueden colisionar, facilitando el manejo de interacciones específicas entre ciertos tipos de objetos.

Simulación Física:

Godot tiene un motor de física incorporado que facilita la simulación realista de fuerzas, gravedad, y otros comportamientos físicos.

Librerías y técnicas utilizadas:

- **Physics2DServer:** El sistema de simulación física 2D de Godot. Controla la gravedad, la inercia, y las colisiones.
- **Body Mode:** Los objetos como *RigidBody2D* ofrecen varios modos: dinámico, estático, kinemático y modo personaje, cada uno con comportamientos específicos.
- **Fuerzas y velocidades:** Para simular colisiones y la interacción con el entorno, se aplicarán fuerzas, velocidades y aceleraciones, como en la función `apply_impulse()` o el ajuste de la propiedad `linear_velocity`.

Interacción entre objetos:

Para manejar las interacciones entre los objetos del juego, se utilizarán señales (signals) y scripts personalizados en GDScript.

Técnicas de interacción:

- **Señales (Signals):** Godot permite emitir señales cuando ocurren ciertos eventos (por ejemplo, cuando dos cuerpos colisionan), lo que facilita la conexión de eventos como el daño al personaje o la recolección de un ítem.
- **Métodos de colisión:** Los métodos como `_on_body_entered()` o `_on_area_entered()` detectan cuando un objeto entra en el área de otro, permitiendo responder de manera inmediata a la interacción.
- **Script personalizado para interacciones:** Cada objeto tendrá scripts específicos para manejar sus interacciones con otros objetos, usando funciones como `queue_free()` para eliminar objetos o `change_scene()` para transicionar entre diferentes niveles del juego.
-

8. Lógica de juego e inteligencia artificial

- **Lógica de juego**: Se basará en un sistema de control por estados (FSM), con estados para ataques, defensa, y combos.
- **IA**: Los enemigos utilizarán un comportamiento básico, con patrones de ataque predecibles que se ajustan según la dificultad.

9. Networking (si aplica)

- Por el momento mis intenciones no sobrepasan el multijugador local
-

10. Audio y efectos visuales

- **Sonido**: Se utilizarán efectos descargados y personalizados para los golpes, bloqueos y poderes especiales.
- **Música**: Se integrarán temas musicales acorde a la ambientación de cada zona y jefe.
- **Herramientas**: **Audacity** para la edición de audio y **FMOD** para la integración avanzada de sonido.

11. Plataforma y requerimientos de software

- **Plataforma**: PC
- **Requerimientos de hardware**:
    - CPU: Intel Core i5 o superior
    - RAM: 4 GB
    - GPU: Integrada, compatible con DirectX 10
    - Espacio en disco: 500 MB
- **Requerimientos de software**:
    - Sistema operativo: Windows 10 o superior, Linux
    - Motor de juego: Godot 4.x