Papito's Adventure

Versión del documento número 1

Escrito por:
Álvarez Sánchez Héctor Antonio
Cervantes Ramirez José Luis
Cruz Morales David Edwin
Delgado Anguiano Jorge Alfredo
Moran Vazquez Brandon Alexander
Piñón Canchola Salvador
Ponce Ricarte Isaac Eduardo
Emmanuel Camacho Moctezuma

1. Lista de características obtenidas del GDD

- Metas del juego:
    - Concepto del juego: Juego de plataformas 2D con combate, acertijos y exploración, donde el jugador controla a Papito, un joven caballero que busca demostrar su valentía en unas ruinas antiguas.
    - Mecánicas: Sistema de combate adaptativo, resolución dinámica de acertijos, y una exploración profunda con caminos ocultos.
- Historia:
    - Papito se adentra en unas ruinas que contienen un oscuro poder. La misión es descubrir los secretos y superar los desafíos dentro de las ruinas.
- Controles:
    - Teclado: Movimiento lateral, salto, ataque, interacción con objetos, y sprint desbloqueable.
- Jugador:
    - Papito: Un caballero novato que busca convertirse en un héroe, crece a medida que avanza en el juego enfrentando a monstruos y resolviendo acertijos.
- Progresión:
    - El juego está estructurado en niveles, cada uno con enemigos y desafíos específicos. Cada nivel culmina con una pelea de jefe y un avance en la historia.
- Mecanicas universales:
    - Salto de Precisión, Sistema de Vida, Puntos de Control, y Sistema de Inventario.
- Enemigos y jefes:
    - Cada nivel presenta enemigos únicos con patrones de ataque. Los jefes requieren estrategia y el uso de habilidades adquiridas.
- Música y SFX:
    - Banda sonora orquestal épica que refleja los distintos ambientes del juego, junto con efectos sonoros que acompañan los movimientos, ataques y exploración.

2. Elección de Game Engine

Motor seleccionado: Godot Engine

Justificación:

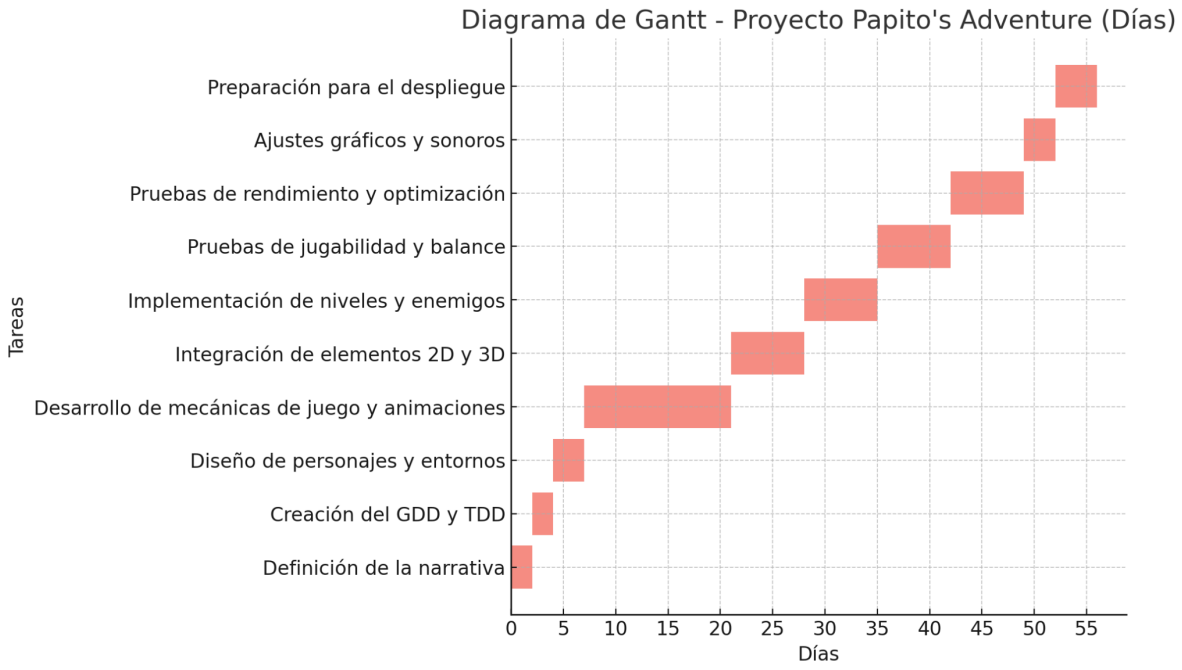
- **Facilidad de uso y escalabilidad:** Godot es un motor de código abierto conocido por su interfaz amigable y una curva de aprendizaje accesible. Esto

permite que el equipo de desarrollo se adapte rápidamente a sus herramientas, lo que es ideal para el desarrollo ágil de Papito's Adventure. Su flexibilidad permite escalar el proyecto sin grandes complicaciones a medida que el juego se expande.

- **Soporte 2D**: Dado que Papito's Adventure es un juego de plataformas en 2D, Godot ofrece un sólido conjunto de herramientas especializadas para juegos 2D, sin las limitaciones que algunos motores tridimensionales imponen al desarrollo en 2D. Su motor 2D nativo optimiza el rendimiento, lo que garantiza un desarrollo fluido y un rendimiento robusto.
- **GDScript**: Godot utiliza GDScript, un lenguaje de programación sencillo y específico del motor que se basa en Python, lo que facilita la escritura de scripts eficientes y la rápida iteración de código. También ofrece soporte para otros lenguajes como C# y VisualScript, lo que añade flexibilidad si se requieren características adicionales.

3. Planeación (Diagrama de Gantt)

Este diagrama de Gantt muestra la planificación detallada para el desarrollo del proyecto *Papito's Adventure*. El proyecto está dividido en cuatro fases principales: Diseño, Implementación, Pruebas y Ajustes, con tareas específicas distribuidas a lo largo de 8 semanas. Cada tarea tiene una duración específica y se visualiza de manera secuencial para garantizar que el proyecto avance de manera eficiente en el tiempo previsto.



Este cronograma asegura una distribución clara de las actividades a lo largo del proyecto. A través de una buena gestión del tiempo y el seguimiento de cada fase, se busca cumplir con los plazos establecidos para cada tarea, desde la definición de

la narrativa hasta la preparación para el despliegue del juego. Esta planificación permite identificar fácilmente las tareas críticas y garantiza que el equipo de desarrollo pueda ajustar sus esfuerzos conforme a las necesidades del proyecto.

4. Diagramas de alto nivel

5. Herramientas de arte

1. Herramientas para la creación de Sprites y Texturas 2D

Dado que **Papito's Adventure** es un juego en 2D con vista lateral, gran parte de los recursos visuales incluyen sprites, texturas y elementos gráficos 2D. A continuación, se describen las herramientas utilizadas:

- **Adobe Photoshop**:
  - Utilizado para la creación de texturas detalladas, fondos y personajes.
  - Funcionalidades clave: Herramientas de pintura digital, edición de píxeles y creación de texturas personalizadas.
  - Formato de salida: PNG con transparencia para sprites y fondos.
- **Aseprite**:
  - Herramienta especializada en la creación de **sprites animados**.
  - Funcionalidades clave: Animación cuadro por cuadro, manipulación de píxeles y exportación en secuencias de imágenes o GIFs.
  - Formato de salida: PNG para sprites individuales o GIF para secuencias animadas.
- **Affinity Designer**:
  - Utilizado para el diseño de interfaces de usuario (UI) y gráficos vectoriales.
  - Funcionalidades clave: Creación de iconos, HUD y menús escalables sin pérdida de calidad.
  - Formato de salida: SVG o PNG dependiendo del uso.

2. Herramientas de Animación 2D

Para las animaciones de personajes, enemigos y efectos visuales en **Papito's Adventure**, se utilizarán las siguientes herramientas:

- **Spine 2D**:
  - Herramienta para animación de esqueletos 2D, permitiendo mover partes del cuerpo de los personajes de manera fluida y eficiente.
  - Funcionalidades clave: Animaciones de esqueletos, rigging para personajes y exportación optimizada para motores de juego.

    - Formato de salida: JSON y PNG para su integración en el motor de juego.
- **DragonBones**:
    - Alternativa a Spine 2D, especialmente útil para exportar animaciones compatibles con motores de juego.
    - Funcionalidades clave: Animaciones esqueléticas, soporte para múltiples plataformas.
    - Formato de salida: JSON y PNG.

3. Herramientas para Modelado y Texturizado 3D

Aunque el enfoque del juego es 2D, algunos elementos en el fondo o efectos especiales pueden requerir modelado en 3D para dar una mayor sensación de profundidad.

- **Blender**:
    - Herramienta de modelado 3D gratuita y de código abierto utilizada para la creación de modelos simples o efectos visuales que pueden ser renderizados en 2D.
    - Funcionalidades clave: Modelado de objetos básicos, texturizado y animación.
    - Formato de salida: PNG para texturas renderizadas, o FBX/OBJ si se utiliza algún modelo 3D.

4. Integración en el Motor de Juego

El equipo de arte trabajará en estrecha colaboración con el equipo de programación para asegurar que todos los recursos visuales se integren correctamente en el motor de juego. El motor principal de **Papito's Adventure** será **Unity**, que es compatible con los siguientes formatos:

- **Texturas y Sprites**: PNG, JPG
- **Animaciones**: JSON y PNG para Spine/DragonBones
- **Modelos 3D** (si son utilizados): FBX, OBJ

5. Almacenamiento y Control de Versiones

Para asegurar la correcta organización y control de versiones de los recursos visuales, se utilizarán las siguientes herramientas:

- **Git + GitHub**: Para el control de versiones de los archivos gráficos y sprites, permitiendo colaborar de manera eficiente entre los miembros del equipo.
- **Google Drive/Dropbox**: Utilizado como respaldo de los archivos de trabajo y versiones finales de los recursos visuales.

6. Flujo de Trabajo y Pipeline de Producción

- Los artistas comenzarán por crear bocetos y prototipos de los personajes y entornos utilizando herramientas como Adobe Photoshop o Affinity Designer.
- Una vez aprobados, estos bocetos serán refinados en Aseprite (para sprites) o Blender (para texturizado 3D).
- Las animaciones se desarrollarán en Spine o DragonBones, y se exportarán en formatos compatibles con Unity.
- Todos los recursos serán integrados y probados en el motor de juego para asegurar que se mantengan dentro de los límites de rendimiento.

6. Objetos 3D, Terreno y Escenas

7. Detección de colisiones, físicas e interacciones

Detección de colisiones:

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

Section 8.1 Game Logic:

- Implementar la lógica del juego utilizando el motor de juego de Unity.
- Utilizar un sistema de máquinas de estado para controlar las diferentes fases del juego y el comportamiento de Papito.
- Implementar un sistema de eventos para manejar interacciones entre Papito, enemigos y elementos del entorno.
- Utilizar un sistema de colisiones para detectar interacciones entre Papito, plataformas, enemigos y objetos coleccionables.

- Implementar un sistema de guardado y carga para manejar los puntos de control y el progreso del jugador.

Section 8.2 Artificial Intelligence:

Enemigos menores (ratas gigantes, espíritus pequeños):
Utilizar un sistema de patrullaje simple:

- Los enemigos se moverán en un patrón predefinido hasta detectar a Papito.
- Si Papito entra en su rango de visión, lo perseguirán directamente.
- Si Papito sale del rango de visión, volverán a su patrón de patrullaje después de un breve período.

Enemigos medianos (gólems, criaturas de sombra):
Implementar un sistema de IA más avanzado:

- Utilizar comportamientos más complejos.
- Incluir estados como "patrullar", "perseguir", "atacar" y "huir".
- Implementar un sistema de percepción que les permita detectar a Papito por visión y sonido.

Jefes (criatura mágica, Guardián del Poder Oscuro):
Diseñar patrones de ataque específicos para cada fase del combate:

- Utilizar máquinas de estado para controlar las diferentes fases y ataques.
- Implementar un sistema de "cooldown" para los ataques especiales.
- Añadir comportamientos adaptativos que cambien según la salud del jefe o las acciones de Papito.

NPCs y elementos interactivos:
Implementar un sistema de diálogo simple para NPCs:

- Utilizar árboles de diálogo para manejar conversaciones con múltiples opciones.
- Crear un sistema de activación para trampas y mecanismos:
- Usar "triggers" para activar trampas cuando Papito entre en ciertas áreas.
- Implementar puzzles lógicos que requieran la interacción de Papito para ser resueltos.

Sistema de dificultad adaptativa:
Implementar un sistema que ajuste la dificultad del juego basado en el rendimiento del jugador:

- Ajustar la frecuencia de aparición de enemigos y su agresividad.
- Modificar la cantidad de recursos (pociones, monedas) disponibles en el nivel

9. Audio y efectos visuales

9.1 Efectos de audio

- Música de fondo
- Efectos de sonido para las acciones del jugador, enemigos y entorno

- Sistema de diálogos para interacciones con NPCs
- Música para batallas

9.2 Efectos visuales

- Efectos de transición de niveles
- Efectos de iluminación para el ambiente

10. Plataforma y requerimientos de software

10.1 Plataforma de Entrega

- PC (Windows) es la plataforma principal de lanzamiento.

10.2 Requerimientos de Hardware

- Procesador Intel i3 o equivalente
- 4GB de RAM
- GPU compatible con DirectX 11
- 4GB de espacio libre en disco

10.3 Requerimientos de Software

- Sistema Operativo: Windows 10
- DirectX: Versión 11 o superior