UNIVERSIDAD DE LAS AMÉRICAS

FICA

TRABAJO GRUPAL

PROGRESO 2

Proyecto Programación 2

Programación Orientada a Objetos

Estudiantes:

Alisson Armas, Samuel Cobo,Amelia Povea, Julio Mera, Jeremy Tomaselly.

Identificación del Problema

Problema:

Una empresa de software, está incursionando en el desarrollo de videojuegos, para esto solicita empezar con un juego RPG con estilo pixel art, que no sea multijugador.

Descripción del Problema:

El problema que se desea resolver es la creación de un juego top-down RPG estilo dungeon con temática de exploración, combate y progresión de personaje, este juego busca ofrecer una experiencia inmersiva y desafiante al jugador, donde cada mazmorra generada de manera única permita diferentes retos con enemigos, y recompensas.

Link Proyecto Solución Seleccionada (Avance Código):

https://drive.google.com/drive/folders/1Ote5TF74SiEf1U_m2G-CZTGDFhxCqMge?usp=sharing

Requerimientos funcionales y no funcionales

Tipo	ID	Requerimiento
Funcional	GeneraciónProceduralMazmorras	El sistema debe generar mazmorras de manera procedural para cada partida.
Funcional	GeneraciónAleatoriaEnemigos	El sistema debe generar enemigos de forma aleatoria dentro de las mazmorras.
Funcional	ExploraciónHabitacionesDinamicas	El jugador debe poder explorar habitaciones generadas dinámicamente.
Funcional	GeneraciónObjetosInteractivos	El sistema debe permitir la aparición aleatoria de objetos interactivos en las habitaciones.
Funcional	CombateConEnemigos	El jugador debe poder combatir enemigos utilizando diferentes armas y habilidades.
Funcional	GuardadoProgresoJugador	El sistema debe guardar el progreso del jugador en tiempo real o en puntos de control.
Funcional	CargaPartidasGuardadas	El sistema debe permitir cargar partidas guardadas.
Funcional	RetroalimentacionVisualSonora	El jugador debe recibir retroalimentación visual y sonora al interactuar con enemigos y objetos.
Funcional	DificultadProgresivaGeneracion	El sistema debe gestionar la generación de niveles con dificultad progresiva.
Funcional	ConfiguracionBasicaUsuario	El sistema debe permitir configuraciones básicas de usuario como sonido, controles y resolución.
No funcional	TiempoCargaMazmorra	El tiempo de carga de una nueva mazmorra no debe exceder los 3 segundos.
No funcional	RendimientoMinimoFPS	El juego debe mantener una tasa de actualización mínima de 30 FPS en configuraciones recomendadas.
No funcional	BuenasPracticasPOO	El sistema debe ser desarrollado utilizando buenas prácticas de programación orientada a objetos.
No funcional	OptimizacionConsumoRecursos	El juego debe estar optimizado para un consumo moderado de CPU y RAM en PC.
No funcional	ModularidadSistema	El sistema debe ser modular para facilitar la adición de nuevas mazmorras, enemigos y objetos.
No funcional	ActualizacionesSinPerdidaProgreso	El sistema debe estar preparado para actualizaciones sin afectar el progreso de los jugadores.
No funcional	CoherenciaEstiloPixelArt	El diseño del arte debe seguir un estilo coherente de pixel art retro de baja resolución.
No funcional	RegistroErroresGeneracion	El sistema debe permitir registrar errores de generación procedural para facilitar el debugging.

Variables Globales Generales

Variable	Tipo	Descripción
nivelActual	int	Número del nivel o piso en la mazmorra.
mazmorra	Mazmorra	Instancia actual de la mazmorra generada.
jugador	Jugador	Instancia del jugador actual.
enemigosTotales	int	Cantidad total de enemigos vivos.
juegoTerminado	bool	true si el juego ha terminado (ganaste o perdiste).
tiempoDeJuego	float	Tiempo total que lleva la partida.
semillaGeneracion	int	Semilla usada para la generación procedural.
objetosRecolectados	List<Objeto>	Lista global de objetos recogidos.
vidasRestantes	int	Vidas extra disponibles.
dificultad	string	Nivel de dificultad: Fácil, Normal, Difícil.
puntuacion	int	Puntos acumulados por el jugador.

Restricciones Globales Comunes

Restricción	Explicación
Vida máxima limitada	Un personaje no puede tener más de vidaMaxima de vida.
Ataque y defensa mínimos y máximos	Valores de ataque y defensa dentro de un rango permitido.
Inventario limitado	Número de objetos en inventario limitado (ej. 20 máximo).
Sólo una arma equipada	Solo puede equiparse una arma al mismo tiempo.
Mana limitado (para magos)	No se puede lanzar hechizos si el mana es 0.
Una habitación a la vez	El jugador solo puede estar en una habitación activa.
Generación consistente	Todas las habitaciones deben ser accesibles.
No duplicar enemigos u objetos	No pueden coexistir múltiples enemigos/objetos en una posición.
Progresión de niveles	Debe completar condiciones antes de avanzar de nivel.
Condición de victoria y derrota clara	Terminar juego en vida=0 o al llegar al final.

Descripción 1ra alternativa: RPG Procedural y No Lineal

Un mundo infinito de aventuras que nunca se repite, este sistema genera mazmorras, enemigos y desafíos de forma aleatoria, ofreciendo al jugador una experiencia única en cada partida, con una arquitectura flexible y modular basada en programación orientada a objetos, permite expandir el juego fácilmente, aumentando la rejugabilidad y reduciendo los costos de expansión a futuro.

UML:

https://drive.google.com/file/d/1eGqLXRLLm2jywX_IuEdfvRBYLOZp2DTC/view?usp=sharing

Aspecto	Ventaja	Desventaja
Flexibilidad	Genera diferentes mazmorras y enemigos en cada partida, ofreciendo experiencias únicas.	Dificulta controlar una progresión narrativa precisa y coherente.
Rejugabilidad	Aumenta la duración del juego, motivando a los jugadores a repetir partidas.	Puede perderse sentido de avance o logro si las partidas parecen demasiado aleatorias.
Escalabilidad	Permite agregar nuevos elementos (enemigos, objetos, mazmorras) sin rehacer estructuras existentes.	A mayor cantidad de contenido, puede ser más difícil balancear correctamente la dificultad.
Uso de principios de POO	Alto aprovechamiento de herencia, composición, bajo acoplamiento y alta cohesión.	Mayor complejidad inicial en el diseño de clases y sus relaciones.
Costos a largo plazo	Menor necesidad de crear manualmente nuevos contenidos, reduciendo trabajo futuro.	Requiere buena planeación inicial para evitar retrabajos costosos si se estructura mal.
Innovación	Experiencias dinámicas y variadas, atractivas para jugadores modernos.	Puede ser menos atractivo para jugadores que prefieren una narrativa lineal tradicional.
Adaptabilidad	Permite crear variantes de juego solo ajustando parámetros como semilla o dificultad.	Exige una correcta validación de las configuraciones para evitar errores en la generación.
Automatización de contenido	Reducción significativa de trabajo manual para crear escenarios.	Requiere invertir más esfuerzo en pruebas automáticas y aseguramiento de calidad.
Menor redundancia	Se evita la duplicación de código creando clases genéricas y reutilizables.	Puede ser difícil de entender o modificar para nuevos programadores sin documentación adecuada.
Modularidad	Permite actualizar o expandir funcionalidades específicas sin afectar todo el sistema.	Una mala división de módulos puede hacer que el sistema sea difícil de mantener o escalar.
Complejidad inicial de programación	(No aplica)	Programar generación procedural desde cero es más complejo que programar estructuras fijas.
Pruebas extensivas	(No aplica)	Necesidad de muchas pruebas para asegurar que todas las mazmorras generadas sean jugables y divertidas.
Gestión de errores	(No aplica)	Hay que prever fallos en la generación, como habitaciones inaccesibles o errores de conexión.
Optimización necesaria	(No aplica)	Los algoritmos de generación procedural pueden impactar el rendimiento si no están bien optimizados.

Descripción 2da alternativa: RPG No Procedural y Lineal

Una historia cuidadosamente diseñada, donde cada nivel está construido de forma manual, este sistema ofrece una estructura fija y controlada, ideal para desarrollar narrativas lineales y progresiones claras, aunque sencillo de implementar, limita la variedad de partidas y dificulta la expansión del juego conforme evoluciona el proyecto.

UML:

https://drive.google.com/file/d/1rG4bDu2Oj25BqHrrbcK69W1YoboVpYQx/view?usp=sharing

Aspecto	Ventaja	Desventaja
Facilidad de implementación inicial	Programación más sencilla: no requiere generación procedural ni algoritmos aleatorios.	El sistema es rígido y difícil de ampliar en el futuro.
Control total de la narrativa	Permite diseñar la historia de forma lineal y precisa, controlando cada evento.	Limita la libertad del jugador y puede hacer que el juego sea predecible.
Estructura clara	Cada mazmorra y habitación tiene una clase específica, facilitando la organización inicial.	Duplica esfuerzo: cada nueva mazmorra necesita una nueva clase o modificación de clases existentes.
Rendimiento	Menor carga para CPU y RAM, ya que no genera contenido dinámicamente.	No se aprovechan oportunidades para optimizar el contenido repetitivo o modularizar procesos.
Uso de principios de POO	Usa herencia básica para personajes y objetos, aplicando encapsulamiento en algunas entidades.	Bajo aprovechamiento de la modularidad: se repite código similar entre clases de habitaciones/mazmorras.
Costos a corto plazo	Más barato en tiempo y recursos durante las primeras etapas del desarrollo.	Costoso a largo plazo: el crecimiento del juego implica crear y mantener muchas clases nuevas.
Facilidad de pruebas	Las pruebas unitarias son más simples ya que cada clase es específica y conocida.	Necesidad de testear de forma manual cada mazmorra por separado, aumentando tiempos de prueba.
Adaptabilidad ante cambios pequeños	Cambios pequeños en una mazmorra específica son fáciles de hacer localmente en su clase correspondiente.	Cambios globales (ej. cambiar mecánicas o añadir nuevas dinámicas) requieren modificar muchas clases.
Requerimientos técnicos bajos	No requiere conocimientos avanzados de algoritmos de generación ni estructuras dinámicas.	El proyecto no incentiva el crecimiento técnico del equipo en habilidades de programación avanzada.
Control de dificultad	La dificultad puede ser cuidadosamente diseñada en cada nivel al ser manual.	No hay adaptación dinámica a la habilidad del jugador, lo cual puede afectar negativamente la experiencia de usuario.

Justificado de porque elegimos el sistema 1

Después de analizar las características técnicas de ambos sistemas propuestos para el desarrollo del RPG en pixel art, se decidió seleccionar el Sistema 1: RPG Procedural y No Lineal como la solución más adecuada, esta elección se basa en principios sólidos de Programación Orientada a Objetos (POO) y en las necesidades específicas del proyecto, el Sistema 1 ofrece una estructura escalable, ya que la generación procedural permite agregar nuevos enemigos, objetos y habitaciones sin reestructurar el código base, facilitando la expansión del juego, en contraste, el Sistema 2 incrementaría la rigidez del sistema y los costos de mantenimiento al requerir nuevas clases para cada mazmorra.

Desde la perspectiva de la programación orientada a objetos, el Sistema 1 aprovecha mejor los principios fundamentales al utilizar herencia, composición, bajo acoplamiento y alta cohesión, asegurando un proyecto robusto, modular y preparado para futuras actualizaciones, además, su generación procedural incrementa significativamente la rejugabilidad, ya que cada partida ofrece una experiencia diferente, lo que resulta esencial para construir una base sólida de usuarios sin necesidad de crear manualmente grandes cantidades de contenido, aunque el Sistema 1 implica mayor esfuerzo inicial en programación y pruebas, este esfuerzo se compensa a largo plazo mediante la reducción de costos de expansión y mantenimiento.

Su carácter dinámico y no lineal también se alinea con las tendencias actuales del mercado, que valora experiencias únicas y adaptativas. Si bien existen riesgos asociados a su complejidad, estos son asumibles mediante una adecuada planificación, un diseño modular y un sistema de pruebas riguroso, por todas estas razones, se concluye que el Sistema 1: RPG Procedural y No Lineal representa la opción más sólida, sostenible y estratégica para iniciar el proyecto de desarrollo de videojuegos, combinando robustez técnica, valor para el usuario y viabilidad a largo plazo.