Desarrollar la arquitectura de software de acuerdo con el patrón de diseño seleccionado

José Ignacio Botero Osorio
Instructor

Lyam Acosta Forero
Aprendiz

SENA – Servicio Nacional de Aprendizaje
Tecnología en análisis y Desarrollo de Software
3118526
Bogotá D.C.

Contenido

Desarrollar la arquitectura de software de acuerdo con el patrón de diseño seleccionado

La arquitectura de Software para nuestro videojuego debe basarse en patrones de diseño, principios de mantenibilidad, vistas de componentes y despliegue con herramientas adecuadas para garantizar escalabilidad y calidad, desarrollaremos una arquitectura completa.

Incorporación de patrones de diseños y buenas practicas

En nuestro proyecto tendremos:

1. **Patrón MVC (Model-View-Control):** Separación clara entre la lógica de datos, la presentación y el control del flujo del juego.
2. **Patrón Observer:** Notificación automática de cambios de estado entre los componentes del juego.
3. **Patrón Factory:** Creación centralizada de objetos del juego (Enemigos, Power-Ups, Efectos)
4. **Patron State:** Gestión de estado del juego (Menú, jugando, pausa y Game Over)

En nuestro software las buenas prácticas implementadas serán:

1. **Código Limpio:** Nombres descriptivos, funciones pequeñas y responsabilidad única.
2. **Testing Automático:** Pruebas unitarias de integración para cada componente.
3. **Documentación:** Comentarios claros y documentación técnica actualizada.
4. **Versionado:** Control de versiones con Git y flujos de trabajo definido.

Vista de componentes del Software

Los componentes del software de nuestro juego son:

1. **Capa de Presentación**
   - Game UI: Interfaz de usuario principal.
   - Hud Manager: Gestión de elementos HUD.
   - MenuStyle: Sistema de menú y navegaciones.
2. **Capa de lógica del juego**
   - GameManager: Control principal de juego.
   - PlayerController: Contorl de Personaje Principal.
   - EnemyAI: Inteligencia Artifical de los Enemigos.
   - LevelManager: Gestion de Niveles y progresión.
3. **Capa de Renderizado**
   - GraphicEngine: Motor de Renderizado 2D.
   - AnimationSystem: Sistema de Animaciones.
   - ParticleSystem: Efectos de Partículas
4. **Capa de Datos**
   - SaveSystem: Sistema de Guardado.
   - DataManager: Gestion de datos del Juego.
   - ConfigManager: Configuraciones y ajustes.

Vista del despliegue del Software

La vista de despliegue de nuestro juego es:

1. **Cliente (Dispositivo de Usuarios):**
2. **Servidor (Si más adelante decidimos crear multijugador):** Game server
3. **Plataforma de Distribución:** Steam, Epic Games Store, Itch.io, Nintendo Switch, PlayStation 5, Xbox.

Herramientas para optimizar los procesos

En nuestro videojuego tendremos las siguientes herramientas para optimizar los procesos:

1. Herramientas de Desarrollo tales como Unity/Unreal Engine, Visual Code Studio como editor con extensiones para desarrollo y Blender para Modelado 3D y animación.
2. Las herramientas de análisis que usaremos es Unity Profiler para el análisis en tiempo real, también Memory Profile para un análisis de uso de memoria y por último Frame Debugger para análisis Frame por Frame.
3. Para las herramientas de control de versiones usaremos Git para control de versiones distribuidas, GitHub para colaboración CI/CD y Git LFS para manejo de archivos grandes.
4. Las herramientas de Testing serán Unity Test Runner para el Testing automático integrado, también el PlayModeTests para pruebas en modo de juego y por último el Device Testing para pruebas en múltiples dispositivos.