INFORME DEL PROYECTO – VIDEOJUEGO “REVOLUCIÓN INDUSTRIAL”

Córdoba D. Jesus A.
jesus.cordobad@udea.edu.co

Páez M. Diego A.
diego.paez1@udea.edu.co

Introducción

Este proyecto consiste en el desarrollo de un videojuego educativo ambientado en la Revolución Industrial. La idea surgió porque queríamos una forma más dinámica y entretenida de explicar los cambios tecnológicos de esa época, especialmente para estudiantes que no están acostumbrados a aprender historia de manera tradicional. El público objetivo son jóvenes y adultos que quieran entender mejor cómo funcionaban las fábricas, los trenes y las construcciones metálicas del siglo XVIII y XIX, pero de una forma interactiva.

El juego está dividido en tres niveles, cada uno inspirado en un sector clave de la Revolución Industrial: la fábrica textil, la red ferroviaria y la recolección de oro. Cada nivel tiene mecánicas diferentes y usa principios físicos reales para que la experiencia sea más inmersiva.

Planteamiento del problema

La enseñanza de la historia suele ser muy teórica y poco práctica. Muchos estudiantes no logran imaginar cómo funcionaban las máquinas o los procesos de la época. El problema que buscamos resolver es precisamente esa falta de conexión entre la teoría y la experiencia.

Nuestro videojuego intenta cubrir esa necesidad: mostrar de forma visual y jugable cómo operaban las máquinas, cómo se movían los trenes y cómo se construían estructuras metálicas. De esta manera, el aprendizaje se vuelve más intuitivo y atractivo.

Definición general y objetivos

Objetivo general

Desarrollar un videojuego educativo que permita comprender, mediante interacción directa, algunos de los avances tecnológicos más importantes de la Revolución Industrial.

Objetivos específicos

- Representar tres escenarios históricos con mecánicas distintas.
- Integrar principios físicos reales en cada nivel.
- Implementar agentes autónomos que apoyen o desafíen al jugador.
- Lograr una experiencia fluida, clara y visualmente coherente.
- Mantener un nivel de dificultad progresivo.

Alcance funcional

El juego incluye:

- Tres niveles completos.
- Movimiento del jugador y animaciones.
- Colisiones, físicas básicas y físicas avanzadas según el nivel.
- Interfaz gráfica con HUD, menús y pantalla de Game Over.
- Enemigos y NPCs con comportamientos autónomos.

- Sistema de puntaje y condiciones de derrota.

Especificación de requerimientos

Requerimientos funcionales

- El jugador debe poder moverse, saltar, interactuar y recoger objetos.
- Cada nivel debe tener un objetivo claro (producir, transportar, recoger).
- Debe existir un HUD que muestre información importante.
- Los agentes autónomos deben reaccionar a lo que ocurre en el nivel.
- El juego debe detectar colisiones y aplicar físicas.

Requerimientos no funcionales

- El juego debe correr a 60 FPS en equipos estándar.
- La interfaz debe ser clara y fácil de entender.
- El código debe estar organizado por módulos y subdirectorios.
- El sistema debe ser estable y no cerrarse inesperadamente.

Restricciones

- Uso obligatorio de C + + y Qt.
- Tiempo limitado para el desarrollo.
- Recursos gráficos creados por nosotros o libres de uso.

Dependencias

- Qt para ventanas, gráficos y eventos.
- QMediaPlayer para música.
- QPainter para dibujar sprites.

Usuarios destinatarios

Personas interesadas en historia o videojuegos educativos.

Metodología y planificación

Usamos una metodología ágil muy básica: dividir el trabajo en tareas pequeñas y avanzar por semanas. Cada semana revisamos qué funcionaba, qué fallaba y qué había que corregir.

Fases del desarrollo

- Diseño inicial y documentación del Momento 1.
- Documentación del momento 2.
- Implementación del menú principal.
- Implementación del Nivel 1.
- Implementación del Nivel 2.
- Implementación del Nivel 3.
- Integración de menús, HUD y agentes.
- Pruebas y correcciones.

Diseño y arquitectura del sistema

El sistema está organizado por carpetas:

- core/ → Controlador principal del juego.
- levels/ → Cada nivel con su lógica independiente.
- entities/ → Jugador, vigas, monedas, trenes, etc.
- agents/ → NPCs inteligentes (capataz,, ladrón).
- ui/ → HUD, menús, pantallas.
- utils/ → Recursos, físicas, colisiones.

Estructura general

- GameController crea y cambia los niveles.
- Nivel es una clase base que comparten los tres niveles.
- Cada nivel implementa su propio actualizar(), dibujar() y reglas.
- El jugador y los agentes se actualizan cada frame.

- ResourceManager carga todos los sprites.
- FisicaMotor contiene las ecuaciones usadas en cada nivel.
- Dependencias externas
- QtCore
- QtWidgets
- QtMultimedia

Desarrollo e implementación

Durante el desarrollo tomamos varias decisiones importantes:

Separar cada nivel en su propio módulo para evitar mezclas de lógica. Crear agentes autónomos que reaccionan al entorno, lo cual hace el juego más dinámico. Usar físicas reales (MRUA, MUA caída libre, fricción) para que el juego sea educativo. Optimizar colisiones para que el jugador no atraviese vigas ni objetos. Mejorar animaciones para que el movimiento se vea natural.

En el Nivel 3 hicimos cambios importantes respecto al Momento 1, cómo ajustar una nueva metodología.

Procedimientos de prueba

Probamos el juego manualmente en cada nivel:

Verificamos colisiones del jugador con vigas, monedas y enemigos. Probamos que los agentes reaccionaran correctamente. Revisamos que el HUD mostrara la información correcta. Medimos el rendimiento para evitar caídas de FPS. Ajustamos posiciones, velocidades y físicas hasta que se sintieran naturales. También hicimos pruebas de estrés generando muchas monedas o moviendo el jugador rápidamente para detectar errores.

Guía de instalación y uso

Requisitos

- Qt 6 instalado.
- Compilador MinGW o MSVC.

Instalación

- Clonar el repositorio.
- Abrir el proyecto en Qt Creator.
- Compilar en modo Release.
- Ejecutar el archivo generado.

Uso

- Navegar por el menú con el mouse.
- Seguir las instrucciones de cada nivel.
- Completar los objetivos antes de que se acabe el tiempo o las vidas.

Resultados y discusión

El videojuego funciona correctamente y cumple con los objetivos planteados. Cada nivel tiene su propia identidad y mecánicas, y los agentes autónomos aportan variedad y desafío.

Limitaciones

- Algunos sprites podrían ser más detallados.
- Las físicas podrían ser aún más precisas.
- El sistema de sonido es básico. Casi nulo.

Dificultades

- Ajustar colisiones en el Nivel 3 fue lo más complejo.
- Integrar animaciones sin romper la lógica del juego.
- Mantener el rendimiento estable.

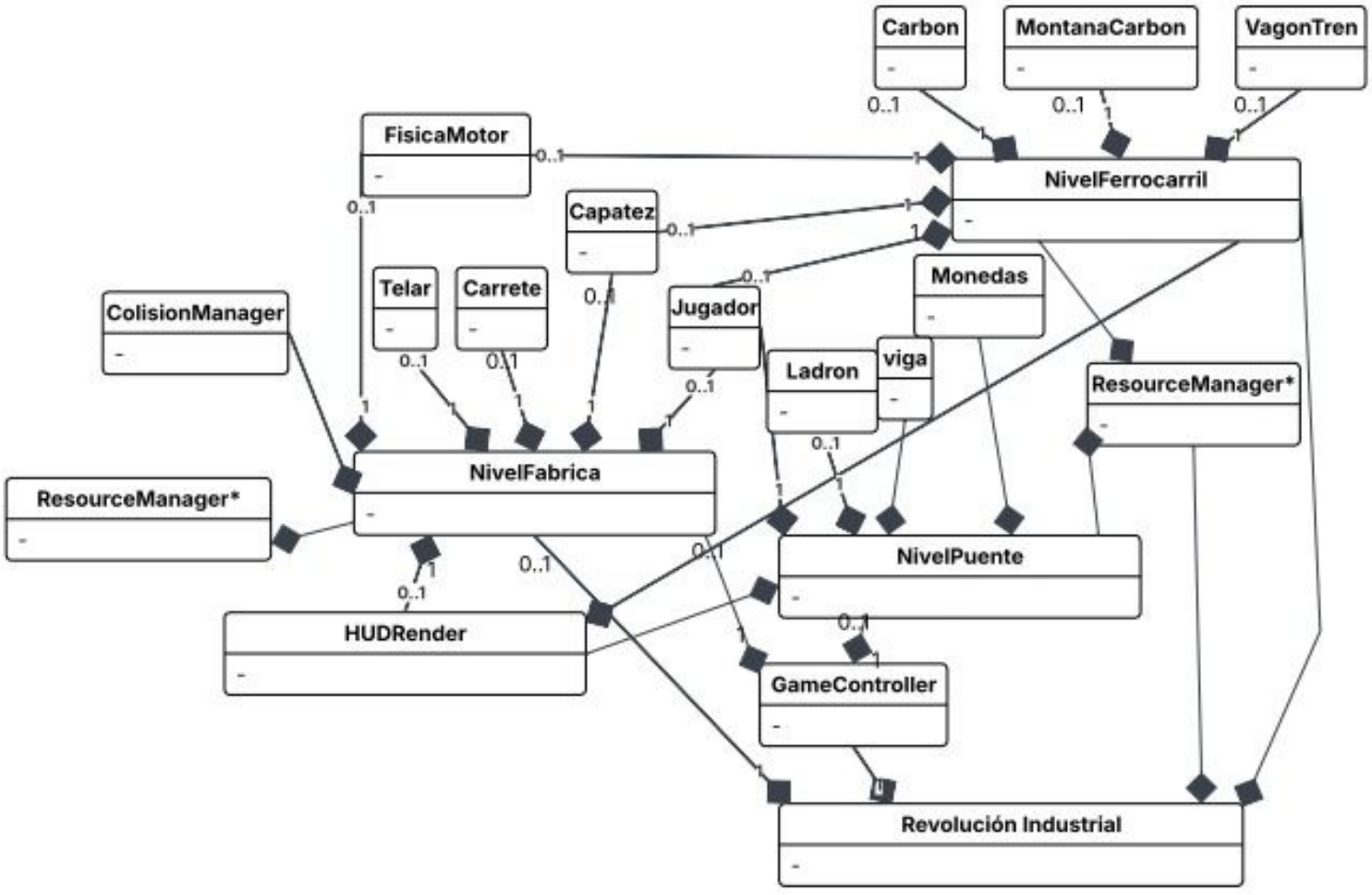
Mejoras futuras

- Añadir más niveles.
- Mejorar la IA de los agentes.
- Agregar más efectos visuales.

Conclusiones

Este proyecto nos permitió entender mejor cómo se estructura un videojuego real: desde la organización del código hasta la interacción entre clases, físicas, animaciones y agentes. También aprendimos a trabajar modularmente y a resolver problemas de forma iterativa.

Además, logramos crear una herramienta educativa que combina historia, física y programación, lo cual demuestra que el aprendizaje puede ser más entretenido cuando se mezcla con videojuegos.


Carbon
-
MontanaCarbon
-
VagonTren
-
FisicaMotor
-
NivelFerrocarril
-
Capatez
-
Telar
-
Carrete
-
Jugador
-
Monedas
-
ColisionManager
-
Ladron
-
viga
-
ResourceManager*
-
NivelFabrica
-
ResourceManager*
-
NivelPuente
-
HUDRender
-
GameController
-
Revolución Industrial
-
0..1
1