Motores Del Vapor

Resumen Ejecutivo

El proyecto Motores Del Vapor es un prototipo de juego de carreras 2D de desplazamiento lateral (side-scrolling racer) desarrollado en C++ utilizando el framework Qt 6.9.2 y el módulo QGraphicsScene. El proyecto se enfoca en simular una carrera con vehículos de temática steampunk que utilizan distintas mecánicas físicas a lo largo de tres niveles progresivos. El estado actual del proyecto es funcional Alpha, con la implementación completa de la física específica por nivel, la lógica de colisiones y la Inteligencia Artificial (IA) básica de los competidores.

Introducción al Proyecto

Título, Género y Estética

Título del Juego: Motores Del Vapor.

Género: Carreras/Plataformas 2D (Side-scrolling Racer).

Estética: El proyecto se adhiere firmemente a la estética Steampunk e industrial. Esto se observa en los diseños de los vehículos (con calderas y tubos de vapor) y en los fondos de los niveles, que evocan ciudades contaminadas por la industria (Nivel 1), entornos nocturnos urbanos (Nivel 2), y complejos de maquinaria gigante (Nivel 3).

Herramientas y Tecnologías

Motor Gráfico/Framework: Qt Framework (versión 6.9.2).

Lenguaje de Programación: C++ (Estándar C++17).

Componentes Clave de Qt: QGraphicsScene y QGraphicsItem para el renderizado y manejo de objetos móviles. El game loop es controlado por un QTimer con un intervalo de 16 ms (aproximadamente 60 FPS).

Arquitectura y Modelado de Clases

El diseño orientado a objetos utiliza las siguientes clases clave:

Clase	Descripción	Variables Clave
ObjetoMovil	Clase base para todos los elementos con posición y velocidad en la escena.	x_pos, y_pos, velocidad_x, velocidad_y.
Vehículo	Hereda de ObjetoMovil. Implementa la lógica de control, la física por nivel y el dibujado con sprites.	masa, fuerzaMotor, friccion, gravedad.
Obstáculo	Hereda de ObjetoMovil. Maneja la lógica de colisiones y el movimiento oscilatorio.	centroY, amplitud, frecuencia.
MainWindow	Clase principal que gestiona la escena (QGraphicsScene), el game loop (QTimer), la creación de niveles y la lógica de colisiones.	nivelActual, jugador, bot1, bot2, obstaculosNivel2.

Implementación Técnica y Física (Clase Vehiculo)

La complejidad del juego se gestiona mediante la implementación de dos modelos de física distintos en la clase Vehiculo, que se aplican según el nivel actual:

Física Nivel 1 (Movimiento Lineal Acelerado)

La función actualizarFisicaNivel1() simula el movimiento sin gravedad (solo horizontal) y utiliza una fórmula de fuerza neta:

Fuerza Neta: $F_{neta} = F_{motor} - F_{rozamiento}$

Fuerza de Rozamiento: $F_{rozamiento} = velocidad_x + friccion$

Cálculo de Aceleración: $Aceleracion = F_{neta}/Masa$

Este modelo es apropiado para un terreno plano, donde el movimiento se basa puramente en la inercia y la fricción.

Física Nivel 2 y 3 (Movimiento Parabólico/Con Gravedad)

La función actualizarFisicaNivel2() introduce la física de plataforma (salto y caída libre).

Gravedad: La velocidad vertical (velocidad_y) se incrementa constantemente por la gravedad (9.8).

Colisión con el suelo: Cuando la posición y_pos alcanza o excede la altura del suelo (400 p.), la velocidad vertical se anula y el vehículo ya no está "en el aire" (enElAire = false).

Salto: Al detectar la tecla ESPACIO (Qt: Key_Space), se aplica un impulso vertical negativo (-60) si el vehículo está en el suelo.

Colisiones, Niveles y Lógica del Juego

Aspecto	Nivel	Lógica de Interacción
Colisión de Penalización	Nivel 1	Colisión con muro_rojo. Si la velocidad es alta (>20.0), se reduce al 25% (velX * 0.25), forzando al jugador a gestionar la velocidad al pasar por el obstáculo.
Colisión de Reinicio	Nivel 2 y 3	Colisión con cualquier obstáculo (incluidos los pinchos y los pistones). Reinicia la posición a las coordenadas de inicio (50, 400), resultando en una pérdida de tiempo considerable.
Obstáculos Dinámicos	Nivel 3	Los obstáculos aplican un Movimiento Armónico Simple (MAS) vertical, determinado por las propiedades amplitud y frecuencia.
Lógica de Ganar/Perder	Todos	Se requieren 5 vueltas para que un vehículo gane. Al cruzar la meta, la posición se reinicia y se aplica una penalización de velocidad del 25% (velX * 0.75).

Conclusiones y Próximos Pasos

El proyecto Motores Del Vapor ha logrado implementar un sistema de juego de carreras 2D con una diferenciación clara de mecánicas físicas y obstáculos a lo largo de tres niveles, demostrando un manejo adecuado de la arquitectura de clases (ObjetoMovil, Vehiculo, Obstaculo) y la integración del framework Qt.

Próximos Pasos Prioritarios:

Despliegue de Interfaz (HUD): Integrar la imagen del velocímetro y otros elementos gráficos (como un contador de vueltas) para mostrar información al usuario.

Refinamiento de la IA: Introducir lógica que permita a los bots anticipar el salto en los Niveles 2 y 3, en lugar de depender únicamente de la probabilidad aleatoria.

Gestión de Memoria: Asegurar la correcta liberación de la memoria para los objetos dinámicos creados en MainWindow (e.g., escena, jugador, bots, muro_rojo y los obstáculos en obstaculosNivel2) en el destructor.