Proyecto Final Videojuego C++

Jhorys Osnaider Goez Renteria
1001034056

Karol Geraldine Cardona Gil
1000762951

UNIVERSIDAD DE ANTIOQUIA
Augusto Salazar
30 de Noviembre de 2025

Informe del Proyecto: Carrera Espacial - Rumbo a la Luna

1. Introducción

Este proyecto desarrolla un videojuego educativo basado en la Carrera Espacial (1957–1969), empleando C++ y el framework Qt. El objetivo principal es simular desafíos físicos y de control experimentados por ingenieros espaciales en tres misiones históricas: Sputnik 1957, Vostok 1961 y Apolo 11. El juego incorpora un agente autónomo, HAL-69, que analiza el estado del vuelo y entrega recomendaciones adaptativas al jugador.

2. Planteamiento del problema

El aprendizaje de conceptos físicos como gravedad, resistencia atmosférica, empuje y masa puede resultar abstracto para estudiantes principiantes. Este proyecto busca resolver esa dificultad mediante un videojuego interactivo que permite visualizar, manipular y experimentar con dichas variables, integrando además elementos históricos que contextualizan la simulación.

3. Definición general y objetivos

Objetivo general:

- Desarrollar un simulador interactivo que permita al jugador comprender principios básicos de dinámica de vuelo espacial reproduciendo misiones históricas.

Objetivos específicos:

- Implementar modelos físicos simplificados para tres entornos distintos: atmósfera terrestre baja (Sputnik), vuelo orbital tripulado (Vostok) y descenso lunar (Apolo 11).
- Integrar un agente autónomo (HAL-69) capaz de analizar telemetría y advertir riesgos.
- Crear una interfaz visual que muestre altura, velocidad, empuje, combustible y estado de misión.
- Diseñar niveles progresivos con retos diferenciados.

4. Especificación de requerimientos

Requerimientos funcionales:

- RF1: El jugador debe poder seleccionar un nivel histórico.
- RF2: El sistema debe simular física basada en empuje, gravedad y resistencia del aire.
- RF3: El agente HAL-69 debe emitir advertencias basadas en velocidad, combustible y fallos acumulados.
- RF4: La UI debe presentar telemetría actualizada en tiempo real.
- RF5: El sistema debe detectar victoria o derrota según condiciones de cada nivel.

Requerimientos no funcionales:

- RNF1: La simulación debe ejecutarse a intervalos regulares (100 ms).
- RNF2: El diseño debe ser modular y orientado a objetos.
- RNF3: La interfaz debe ser intuitiva y permitir control fino del empuje y variables iniciales.

5. Metodología y planificación

Se adoptó una metodología incremental y orientada a objetos:

1. Diseño del modelo físico (Cohete).
2. Implementación de niveles históricos (Nivel1_Sputnik, Nivel2_Vostok, Nivel3_Apolo11).
3. Integración del motor de juego (Juego.cpp).
4. Creación del asistente autónomo HAL-69.
5. Desarrollo de la interfaz gráfica Qt (MainWindow + VisualizacionWidget).
6. Pruebas individuales por nivel y ajustes del sistema físico.

6. Diseño y arquitectura del sistema

El sistema sigue una arquitectura orientada a clases:

- **Cohete:** Gestiona velocidad, altura, masa, empuje, combustible y estado estructural.
- **Nivel (abstracta) y subclases:**
  - Nivel1_Sputnik: ascenso atmosférico con riesgo de quemadura.
  - Nivel2_Vostok: vuelo tripulado con bandas de velocidad seguras.
  - Nivel3_Apolo11: descenso lunar con velocidad terminal crítica.
- **HAL-69:** Analiza telemetría y adapta advertencias según fallos acumulados.

- **Juego:** Coordina la simulación, evolución temporal, física y estados de victoria/derrota.
- **MainWindow y VisualizacionWidget:** Representan la UI y los gráficos animados del cohete, atmósfera, Tierra y Luna.

Las interacciones entre clases aseguran independencia modular: cada nivel define su física, Cohete encapsula su estado, y HAL-69 interpreta información sin modificar la simulación.

7. Desarrollo e implementación

Se implementaron ecuaciones físicas simplificadas:

- Empuje: $a = F/m$ variando según combustible restante.
- Gravedad: $g(h) = g_0 \left(\frac{R}{R+h}\right)^2$ en niveles terrestres.
- Arrastre atmosférico: proporcional a v^2 y dependiente de la altura.
- Consumo de combustible: según empuje y nivel.

HAL-69 analiza velocidad, combustible y nivel, ajustando umbrales después de fallos (aprendizaje mínimo). La interfaz Qt procesa eventos, controla la animación y muestra telemetría con colores según riesgo.

8. Procedimientos de prueba

Se realizaron pruebas para:

- Verificar ascenso progresivo en Nivel 1 sin quemaduras.
- Validar bandas seguras de velocidad en Nivel 2 y g-fuerzas máximas.
- Comprobar aterrizaje suave en Nivel 3 con $\|v\| \leq 5$ m/s.
- Evaluar la consistencia del consumo de combustible.
- Probar advertencias de HAL-69 bajo diversas condiciones anómalas.

9. Guía de instalación y uso

1. Compilar el proyecto en Qt Creator (versión 6.x recomendada).
2. Ejecutar la aplicación.
3. Seleccionar un nivel.
4. Ajustar empuje y, para el Nivel 1, velocidad inicial.
5. Iniciar simulación y controlar el cohete.
6. Observar telemetría, advertencias y resultados de misión.

10. Resultados y discusión

El simulador reproduce con fidelidad simplificada los retos de cada etapa histórica: quemaduras atmosféricas, control orbital tripulado y descenso lunar. HAL-69 ofrece un acompañamiento útil, con aprendizaje básico. La arquitectura modular facilita ampliaciones, como agregar nuevos niveles o mejorar la física.

11. Conclusiones

El proyecto logra integrar conceptos de programación orientada a objetos, física computacional, diseño de interfaces y elementos históricos, produciendo una herramienta educativa interactiva.

12. Referencias

- Documentación oficial de Qt 6
- Textos de dinámica clásica (Newton–Euler)