Informe Momento III Proyecto Final

Daniel Antonio Londoño Godoy

Facultad De Ingeniería, Universidad De Antioquia

Informatica II: 2598521

7 de Julio de 2025

Introducción

Este proyecto consiste en el desarrollo de un videojuego 2D usando C++ y Qt, donde el jugador controla a Goku en una aventura dividida en dos niveles: uno de plataformas con recolección de esferas del dragón y otro de combate contra un enemigo. El juego fue diseñado aplicando conceptos de programación orientada a objetos, física en videojuegos, gestión de recursos gráficos y sonoros, así como estructuras de datos de la STL para optimizar el comportamiento del juego en tiempo real.

Estructuras De Datos STL Utilizadas

list Para Proyectiles (*Obstaculo*)

Se utiliza una lista enlazada para almacenar los proyectiles activos en el juego. Esto se debe a que los proyectiles se agregan y eliminan de forma constante durante el juego y no se requiere iterar sobre ellos frecuentemente. Esto facilita una gestión segura y dinámica del conjunto de proyectiles lanzados por los enemigos.

vector Para Enemigos (*Enemigo*)

El almacenamiento de enemigos se realiza mediante un vector dinámico porque se requiere acceso rápido por índice y la cantidad total de enemigos es pequeña, por lo que no cambiará constantemente

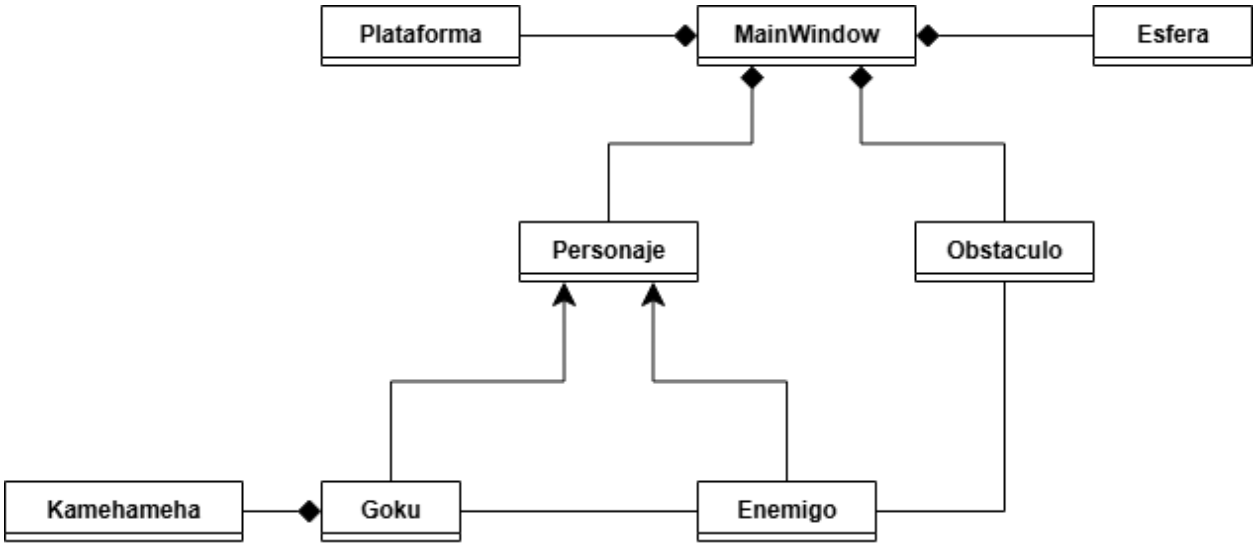
array Para Plataformas (*Plataforma*) Y Esferas (*Esfera*)

Se usa un arreglo para estos elementos debido a que la cantidad de elementos es constante y conocida de antemano distribuyendose así: 8 plataformas, 7 esferas del dragón.

Herencia Y Relaciones Entre Clases

Goku y Enemigo heredan de la clase base Personaje, que encapsula atributos y funciones comunes como la posición, vida, movimiento, etc.

Goku contiene un objeto Kamehameha, el cual gestiona el disparo y animación del ataque especial. La composición se favorece en este caso para mantener modularidad y encapsular la lógica del ataque dentro de su propia clase.



Nota. UML simplificado para mostrar las relaciones entre clases, ver Anexo 1 para su versión completa.

Arquitectura General

La clase MainWindow no solo representa la interfaz gráfica principal, sino que también actúa como controlador de niveles, conteniendo y coordinando a todas las demás clases (*Plataforma, Esfera, Personaje, Obstáculo*). Esta clase

- Controla el flujo del juego, como la transición entre niveles, la música de fondo, el temporizador de escena y la actualización gráfica.
- Se encarga de instanciar objetos del juego, agregarlos a la escena (QGraphicsScene) y eliminar correctamente los objetos al cambiar de nivel o finalizar el juego.

  Este enfoque mantiene el control centralizado y facilita el manejo del estado global del juego.

Física Utilizada En El Juego

El juego implementa distintos principios de física clásica para representar de manera más realista y dinámica el comportamiento de los personajes y objetos. A continuación, se describen las físicas aplicadas:

Movimiento Parabólico

El personaje principal, Goku, puede saltar mediante un sistema que simula movimiento parabólico. Se controla la velocidad vertical en cada cuadro, que va disminuyendo progresivamente para simular el efecto de la gravedad. La colisión con plataformas detiene la caída, mientras que la falta de soporte provoca una caída natural.

Además, algunos obstáculos usan este movimiento con velocidad inicial, gravedad y ángulo inicial aleatorios.

Movimiento Rectilíneo Uniformemente Acelerado (MRUA)

Algunos obstáculos se desplazan con MRUA. Se aplica una aceleración o velocidad constante en el eje X o Y, lo que genera trayectorias rectas con velocidad creciente lo cual permite simular ataques directos o lanzamientos con trayectoria realista, dependiendo del nivel.

Movimiento Oscilatorio (Esferas Del Dragón)

Las esferas del dragón presentan un movimiento oscilatorio vertical, con el fin de destacarlas visualmente dentro del entorno del juego. Este efecto se implementa mediante una pequeña variación periódica en su posición Y, simulando un movimiento de vaivén o flotación. Este comportamiento no responde a una física estricta como un péndulo o un resorte, pero imita una oscilación armónica simple para dar sensación de vida y movimiento.

Conclusiones

El proyecto demuestra el uso adecuado de las estructuras de datos estándar de C++ según el comportamiento esperado de los elementos del juego. Se aplicó una arquitectura basada en objetos con herencia y composición, además de una clase principal que coordina la lógica y la presentación del juego.Ademas, el uso de *std::list, std::vector* y *std::array* fue clave para garantizar eficiencia, seguridad y claridad en el manejo de datos, según el patrón de uso de cada tipo de entidad.

Anexo 1 (Diagrama De Clases Extendido)