# **DOCUMENTACIÓN DEL PROYECTO**

## **1. Portada**

* **Autores:** Andrés Eduardo Santos Ramos y Cesar Luis Alvarez Sanchez
* **Curso:** Informatica II
* **Semestre:** 2025-1

## **2. Introducción**

Este proyecto es un videojuego en 2D desarrollado con C++ y Qt, inspirado en el episodio 15 de Dragon Ball, donde Goku y Krilin entrenan con el Maestro Roshi. El objetivo del juego es aplicar conocimientos de programación orientada a objetos, física básica, manejo de memoria dinámica y diseño de interfaces gráficas para crear una experiencia interactiva dividida en tres niveles distintos con mecánicas y retos únicos.

## **3. Requisitos**

### **3.1. Requisitos funcionales**

* El jugador puede seleccionar entre tres niveles con retos diferentes.
* En el **Nivel 1**, debe recolectar 5 tarros de leche mientras esquiva obstáculos.
* En el **Nivel 2**, debe mantener su oxígeno mientras nada, esquiva tiburones y encuentra una roca especial.
* En el **Nivel 3**, debe esquivar y reflejar proyectiles lanzados por el Maestro Roshi hasta vencerlo.
* Cada nivel tiene su propia lógica de victoria y derrota.
* El jugador puede pausar, reiniciar el nivel o regresar al menú principal en cualquier momento.

### **3.2. Requisitos no funcionales**

* El juego corre a 60 FPS (aproximadamente, con QTimer a 30 ms).
* Resolución de pantalla: 1200x600 px.
* Interfaz gráfica implementada con Qt (QGraphicsView/QGraphicsScene).
* Se usan buenas prácticas de POO y manejo adecuado de recursos dinámicos.

## **4. Diseño del Sistema**

### **4.1. Clases principales**

* Nivel: Clase que representa cada nivel del juego. Contiene la lógica de actualización, generación de enemigos/ítems y verificación de victoria/derrota.
* Jugador: Clase que representa al jugador, incluyendo sus vidas, puntaje, movimiento (parabólico, acuático o esquiva), animaciones y contraataque.
* Obstaculo: Clase para objetos peligrosos que el jugador debe evitar (piedras, troncos, abejas).
* Item: Clase para objetos recolectables (leche, oxígeno, roca final).
* Enemigo: Representa enemigos como tiburones o el Maestro Roshi, con animaciones y lógica de ataque.
* Proyectil: Proyectiles disparados por el jugador o por enemigos.
* PausaPanel: Interfaz de pausa con opciones de reinicio y salida.

### **4.2. Niveles del juego**

* **Nivel 1 – Carrera con leche:** Movimiento lateral automático, salto parabólico, obstáculos y recolección.
* **Nivel 2 – Lago:** Movimiento libre en el agua, consumo de oxígeno, enemigos que se desplazan, y búsqueda de una roca especial.
* **Nivel 3 – Roshi:** Esquivar disparos, reflejarlos durante contraataque, y vencer al enemigo usando física de proyectiles.

## **5. Funcionalidades clave**

* Sistema de **vida** y **oxígeno** representado con barras gráficas dinámicas.
* Sistema de **puntaje** acumulativo y contador de ítems recolectados.
* Animación de personajes con sprites personalizados.
* **Contraataque** en el Nivel 3 que refleja proyectiles enemigos.
* **Reinicio y pausa** funcional con manejo robusto de recursos (delete, clear, etc.).
* **Regreso al menú** tras perder o ganar un nivel.

## **6. Tecnologías utilizadas**

* **Lenguaje:** C++
* **Framework:** Qt 6.x (QGraphicsScene, QTimer, QMediaPlayer, etc.)
* **IDE:** Qt Creator
* **Otros recursos:** Krita o Photoshop para edición de sprites, QPixmap para gráficos.

## **7. Gestión de memoria y estructuras dinámicas**

* Todos los objetos dinámicos (Jugador, Enemigo, Proyectil, Item, etc.) se crean con new y se eliminan en el destructor de Nivel.
* Uso de QVector y QList para manejar colecciones de objetos (proyectiles, obstaculos, enemigos, etc.).
* Verificación constante del estado de punteros antes de eliminar o acceder a ellos.
* Nivel::limpiarTodo() y el destructor garantizan que no haya memory leaks.

## **8. Problemas encontrados y soluciones**

* **Problema:** Crash al salir o reiniciar un nivel por punteros colgantes.  
   **Solución:** Mejor manejo de eliminación y reinicialización de objetos. Uso seguro de nullptr y control de flags como nivelFinalizado.
* **Problema:** El Nivel 2 no regresaba correctamente al menú.  
   **Solución:** Se verificó que emit regresarAlMenu() solo se ejecutara si nivelFinalizado == false, y se ajustaron condiciones en mostrarResultado().
* **Problema:** Al reiniciar el Nivel 1 y 3, el juego se quedaba congelado.  
   **Solución:** Se identificó que el temporizador no se reiniciaba correctamente. Se ajustó reiniciarNivel() para garantizar que los temporizadores se reactiven tras reiniciar.

## **9. Conclusión**

Este proyecto permitió aplicar conceptos avanzados de programación orientada a objetos, estructuras dinámicas y física básica en el desarrollo de videojuegos. Se logró un juego funcional, visualmente atractivo y con mecánicas diferentes para cada nivel. La correcta gestión de recursos, animaciones y eventos en tiempo real fue clave. Como mejora futura, se podrían añadir más niveles, personalización de personajes, un sistema de guardado o dificultad ajustable.