Term Project Report – 2D Platformer Game using C++ and Raylib

Project Overview

Our project is a 2D game developed in C++ using the Raylib library. The player must navigate from point A to point B while avoiding obstacles, jumping between platforms, and reacting to environmental hazards. While we began with a broader vision involving tilemaps and layered environments, we adapted our implementation to suit project constraints and technical challenges.

Character and Movement

We used a sprite sheet from itch.io to represent the main character. Initially, we simulated the player as a simple Rectangle shape to prototype movement, gravity, and collision. Later, we replaced this with a 2D sprite texture and scaled it to match the rectangle used for collision logic.

Player Movement Features:

- Forward and backward movement (arrow keys)
- Jumping with a `velocity.y` vector affected by gravity
- Gravity ensures the player falls if not on a platform
- We tried to draw the sprite sheet using DrawTexture but it was too small
- DrawTexturePro()` was used to draw a scaled portion of the sprite sheet

To keep the game frame-rate independent, movement and gravity calculations are multiplied by GetFrameTime(), ensuring smooth behavior across devices.

Platforms and Collision

Instead of relying on an external tilemap system, we defined walkable platforms using a `std::vector<Rectangle>`. These represent ground or platforms at various heights.

Coordinates were obtained using Tiled, an open-source map editor, the same map editor we used to design the background image.
Platforms were drawn using Raylib's `DrawRectangleRec()` function.
Collision detection was implemented using `CheckCollisionRecs()` to detect whether the player is standing on a platform or not.

This approach allowed us to simulate realistic gravity and jumping mechanics without requiring a full tilemap engine.

Tilemap Design Attempt

Originally, we planned to use Tiled in combination with the TMX format and a C library such as `libtmx` to handle layered map data. We discovered that:

- `.tmx` files are XML-based and require external parsers
- Raylib does not natively support TMX tilemaps
- Integrating an XML or CSV parser and drawing layers manually proved time-consuming
- Making the player interact with TMX-based collision layers was non-trivial

Due to the complexity and time constraints, we switched to using a full background image as the environment and overlaid rectangles to define walkable areas.

Obstacle System

We created a moving obstacle and implemented a basic projectile system:

A `Projectile` structure encapsulates bullet position, motion, and collision detection with the player.
Projectiles spawn at intervals and move vertically toward the player.
Collision with a projectile triggers game over.

This added a challenge to the gameplay loop and tested the player's timing and movement.

Sound System

We implemented both short sound effects and background music using Raylib :

Short sounds like "game over" and "win" use `Sound` and are triggered on events
Background music uses `Music` (a streaming type), which requires calling `UpdateMusicStream()` every frame
We load and unload all sounds at the start and end of the program using helper functions

This taught us the difference between buffered audio and streamed audio.

Challenges & Adaptations
We had difficulty integrating `.tmx` tilemaps so we switched to image background with manually defined rectangles
Sprite sheet showed all frames at once so we used `src` rectangle to crop one frame only
Sprite scale was too large so we used `DrawTexturePro()` and adjusted player rectangle size

Audio playback had inconsistencies so we ensured sounds were triggered only once and music was updated every frame

Conclusion

This project taught us key C++ and game development concepts, including:

- Game loop structure
- Sprite rendering and scaling
- Basic physics and collision detection
- Using `GetFrameTime()` for consistent gameplay
- Audio integration and debugging
- Trade-offs between ideal features and practical implementation