Project Modules and Tools Overview

1. SFML (Simple and Fast Multimedia Library)
- Modules Used:
  - SFML/Graphics.hpp: For rendering sprites, shapes, and text.
  - SFML/Audio.hpp: For playing background music and sound effects.
  - SFML/System.hpp: For timing and system utilities.
- Purpose:
  SFML is the main multimedia library used for all graphics, audio, and window management in this project.

2. Game Modules
- Platform, Coin, and Obstacle Structures:
  Custom structs (Platform, Coin, Obstacle) represent game objects, each holding a sprite and relevant properties (like position, type, or collected status).
- Game State Management:
  An enum class GameState is used to manage the current state of the game (MENU, PLAYING, PAUSED, GAME_OVER, HIGH_SCORE).

3. Randomization(***changed from permanent repeating positions to random***)
- Tools Used:
  - <random>: For generating random positions for platforms and other game elements, ensuring varied gameplay.

4. Audio Tools
- Background Music:
  Two sf::Music objects (bgm1, bgm2) are used for layered background music, with logic to start/stop them based on game state.
- Sound Effects:
  sf::Sound and sf::SoundBuffer are used for coin collection, obstacle collision, and game over sounds.

5. User Interface
- Text and Buttons:
  sf::Text and sf::RectangleShape are used for displaying UI elements like the game title, buttons (Start, Resume, Restart, Main Menu, High Score, Back), and score/coin counters.
- Font:
  A custom font (arial.ttf) is loaded for all text rendering.

6. File I/O
- High Score Storage:
  <fstream> is used to read and write high scores to a text file (assets/highscores.txt), ensuring scores persist between sessions.

7. Game Logic
- Animation:
  Sprite sheet animation is handled by updating texture rectangles based on a timer.
- Physics:
  Gravity and jumping are implemented using velocity and position updates.
- Collision Detection:
  Bounding box intersection checks are used for platform landing, coin collection, and obstacle collision.

8. Build Tools
- Makefile/Command Line:
  The project is built using clang++ with appropriate SFML include and library paths, and the output is placed in a bin directory.

Summary:
This project uses SFML for all multimedia needs, C++ STL for data structures and randomization, and standard file I/O for persistent high scores. The code is modular, with clear separation between game objects, UI, audio, and game logic, making it easy to extend and maintain.

Text