Maze Generation and Enemy Movement

Contributors:
1. Tushar Verma:
   - Implemented the SearchAndDestroy maze generation algorithm
   - set up the development environment
   - coordinated between parts;

2. Samyag Kothari:
   - Implemented music system and audio integration
   - Integrated maze generation algorithm with game logic
   - Coordinated maze generation with other game components

3. Abhishek Sonparote:
   - Developed user interface elements
   - Implemented player collision detection with walls
   - Created enemy movement and behavior system

4. Mrigank Sharma:
   - Added background image implementation
   - Managed game state transitions
   - Implemented smooth gameplay mechanics

1. Maze Generation:
   - **Algorithm**: SearchAndDestroy
   - **Techniques Used**:
     - Randomized direction shuffling to ensure unpredictable maze paths.
     - Oscillation between two states (search & destroy) to carve out paths in the maze.
     - Walls are removed between cells to create a continuous path.
   - **Functions**:
     - `initialize()`: Initializes the maze grid with all walls.
     - `isvalid(int x, int y)`: Checks if a cell is valid for carving paths.
     - `randomdirection(int arr[], int size)`: Shuffles directions for randomness.
     - `SearchAndDestroy(int startX, int startY)`: Implements the maze generation algorithm.
     - `generateMaze()`: Orchestrates the maze generation process.

2. Enemy Movement:
   - **Techniques Used**:
     - The enemy moves towards the player using vector calculations.
     - Movement speed is adjusted based on difficulty levels (easy, medium, hard).
     - Collision checks ensure the enemy does not move through walls.
   - **Functions**:
     - `MoveEnemy()`: Moves the enemy towards the player while avoiding walls.
     - `MoveEnemy1()`: Simplified enemy movement for medium and hard difficulty levels.

3. Player Movement:
   - **Techniques Used**:
     - The player moves based on keyboard input.
     - Collision checks ensure the player does not move through walls.
   - **Functions**:
     - `CanMove(float newX, float newY)`: Checks if the player can move to a new position.
     - `MovePlayer()`: Updates the player's position based on input.