|  |
| --- |
| Project Dissent    Project DISSENT |
|  |

**Introduction .......................................................................................................................................... 3**

Objective ........................................................................................................................................... 3

Gameplay Features .......................................................................................................................... 3

Visual and Audio Experience ............................................................................................................ 3

Title Screen ....................................................................................................................................... 4

Game Screen .................................................................................................................................... 4

Game over ........................................................................................................................................ 4

**Code Structure ..................................................................................................................................... 6**

Project Dissent

1. Player Character ........................................................................................................................... 6

Initialization and Appearance ...................................................................................................... 6

Movement and Control ................................................................................................................ 6

Actions ......................................................................................................................................... 7

1. Enemies ........................................................................................................................................ 8

Enemy Class ............................................................................................................................... 8

Spawning and Movement ............................................................................................................ 9

Visual Representation ................................................................................................................ 10

1. Bullets ......................................................................................................................................... 10

Initialization ................................................................................................................................ 10

Behavior ..................................................................................................................................... 11

1. Game Loop ................................................................................................................................. 12

Core Functionality ...................................................................................................................... 12

Game State Management ......................................................................................................... 12

1. Sound System in the Game ........................................................................................................ 17

Components of the Sound System ............................................................................................ 18

**DSA Used in Game ............................................................................................................................. 20**

1.Vector (Dynamic Array) ................................................................................................................ 20

2.List (Doubly Linked List) ............................................................................................................... 20

3.Vector2f (2D Vector) ..................................................................................................................... 22

**Flow Chart ........................................................................................................................................... 23**

**Conclusion .......................................................................................................................................... 24**

Project Dissent

**Introduction**

"DISSENT" is an engaging and fastpaced 2D survival shooter game where players confront hordes of

relentless zombies. With a captivating combination of action, strategy, and reflexes, players must navigate a

battlefield teeming with enemies, using skillful movement and precise aim to stay alive.

Objective

The primary goal is to eliminate as many zombies as possible while avoiding collisions and preventing zombies

from crossing the screen. The game ends if the player's health is depleted due to zombie attacks or missed

enemies. Each zombie kills rewards the player with points, offering a highscore challenge.

Gameplay Features

●

**Player Movement: The**

player can freely navigate the

battlefield using keyboard controls (W, A, S, D).

Players must avoid zombies and strategically position themselves for better shots.

●

**Bullet Mechanics: Armed**

with a rapidfiring gun, the

player can launch bullets toward approaching

zombies using the "F" key. The bullets are designed to travel horizontally and deactivate once they

leave the screen or hit an enemy.

●

**Enemies:**

Zombies spawn randomly from either side of

the screen and move toward the opposite side.

Colliding with a zombie or letting one escape costs the player health, adding an intense survival

element.

●

**Health and Lives:**

The player starts with four lives,

represented visually. Each zombie collision or

missed enemy decreases the player's health. The game ends when all lives are lost.

●

**Scoring System:**

Each zombie kill grants one point.

Players can compete with themselves or others to

achieve the highest score possible.

Visual and Audio Experience

"DISSENT" immerses players with dynamic visuals, engaging sound effects for gunfire and zombie hits, and a

thrilling background soundtrack to keep the adrenaline high. The vibrant graphics and smooth animations

enhance the overall experience.

This game is a test of reflexes, precision, and strategy, offering an exhilarating challenge to both casual players

and enthusiasts seeking to hone their survival skills. Are you ready to face the horde and emerge victorious in

"DISSENT"?

[***Game Play Video***](https://drive.google.com/file/d/1UcY6ACKJRks6IrLWGFDXtT8rfYk-Ixi_/view?usp=drive_link)

[***Source Code***](https://drive.google.com/file/d/1BV9hZTymFuNWf8l_qB2n040O21l0qvTn/view?usp=sharing)

[***Project File***](https://drive.google.com/drive/folders/1o-cNc-RzJXwgl5zRSEoQPexhU3ag7pEp?usp=sharing)

Project Dissent

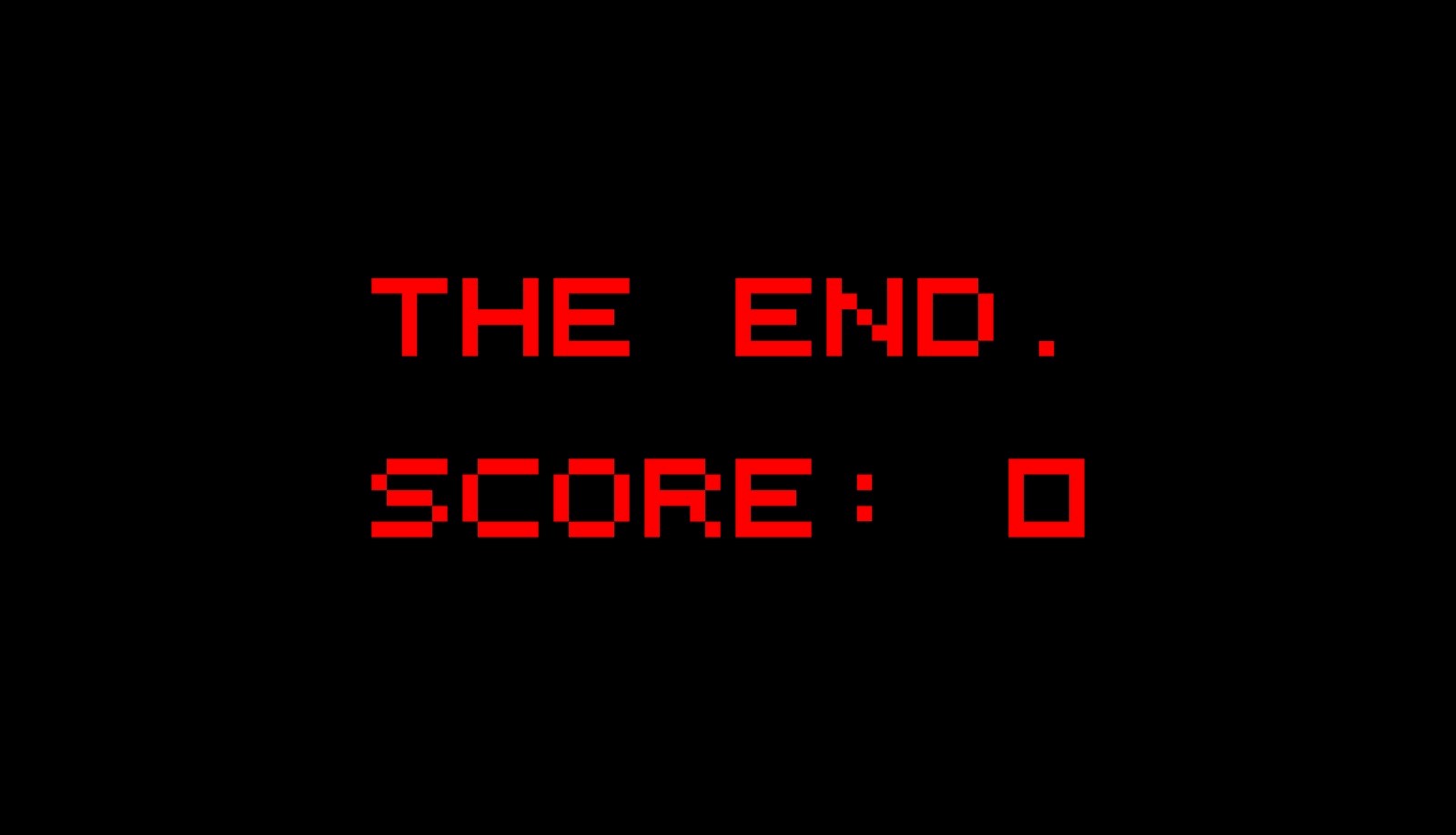
Title Screen

Game Screen

Game over



Project Dissent



Project Dissent

**Code Structure**

The game is divided into the following core components:

**1**

**.**

**Player Character**

**2**

**.**

**Enemies**

**3**

**.**

**Bullets**

**.**

**4**

**Game Loop**

**5**

**.**

**Collision Handling**

**6**

**.**

**Scoring and Lives System**

7

.

**Game Start and End States**

1

. Player Character

The player is the protagonist, controlled by the user, and performs various actions:

Initialization and Appearance

●

**Sprite and Texture:**

The player's visual representation

is managed using a sprite, with textures

(

textureRight

,

textureLeft

)

dynamically loaded to

depict movement direction.

●

**Starting Position:**

The player begins at

(375.0

f, 275.0f)

with scaling and texture dimensions

ensuring proper size and placement.

●

**Bounding Box Adjustment:**

Collision detection is optimized

by resizing the bounding box to align

closely with the visible sprite.

Movement and Control

**●**

**Keyboard Inputs:**

o

**W/A/S/D Keys:**

Allow movement in four directions (up,

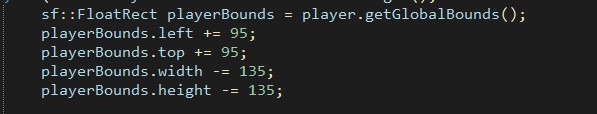
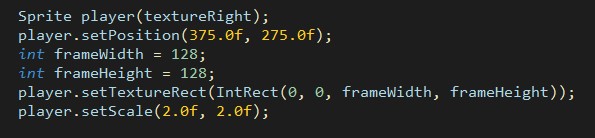
left, down, right).

o

**Boundary Handling:**

Prevents the player from moving

off-screen by clamping positions.



Project Dissent

●

**Smoothness:**

Movement speed is scaled by

deltaTime

for consistent gameplay across different frame

rates.

Actions

**●**

**Shooting Bullets:**

o

Triggered using the

**F key**

.

o

Bullets fire in the direction the player is facing.

o

A sound effect (

shotSound

)

enhances the shooting experience.

**●**

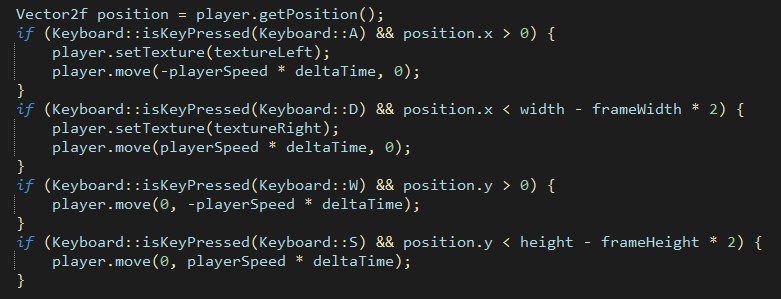
**Interactions with Enemies:**

o

Collision with an enemy reduces lives and removes the enemy.

o

Missed enemies also reduce lives proportionally.



Project Dissent

2

. Enemies

Enemies provide the challenge by appearing randomly and moving towards the bottom of the screen.

Enemy Class

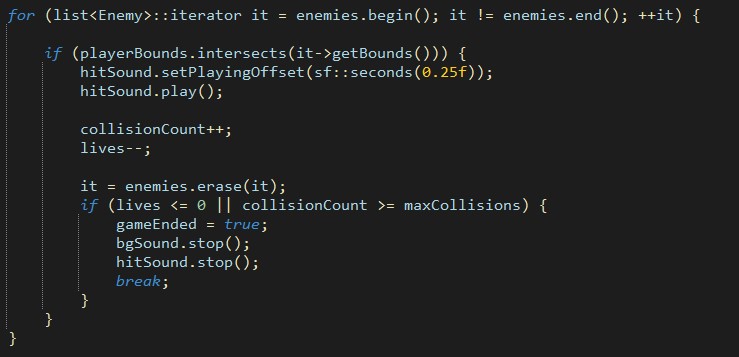
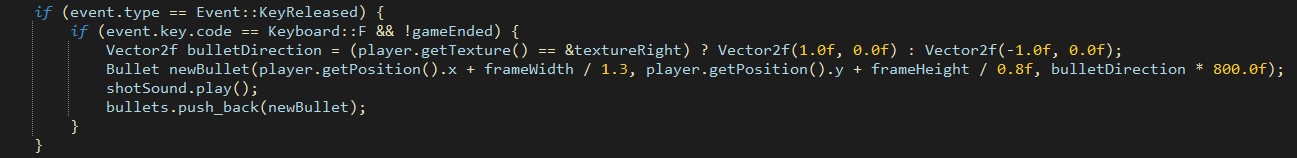
The

**Enemy**

class is responsible for representing enemy

objects in the game. Each enemy is instantiated with

unique attributes and behaviors, making the game more dynamic and challenging.



Project Dissent

Spawning and Movement

**●**

**Randomized Spawning:**

o

Enemies spawn at random horizontal positions at the top of the screen, ensuring

unpredictability.

o

The frequency of spawning is controlled by a timer mechanism.

**●**

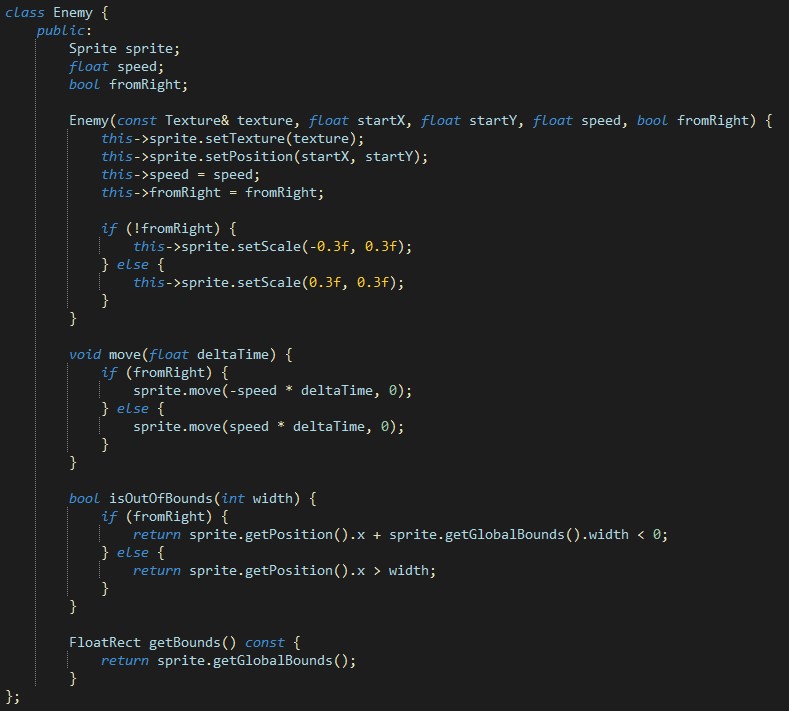
**Vertical Movement:**

o

Enemies descend at a consistent speed.

o

If an enemy moves off-screen, it is removed from the list and reduces the player’s health.



Project Dissent

Visual Representation

●

Each enemy is represented using a unique sprite with textures, adding variety.

3

. Bullets

Bullets serve as the player's primary means of interacting with and eliminating enemies.

Initialization

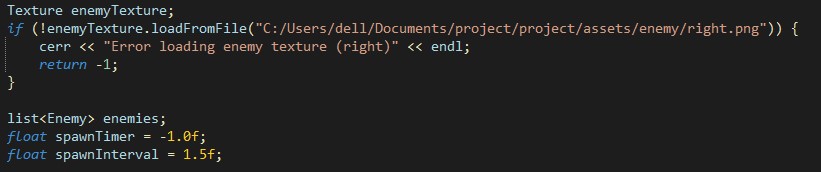
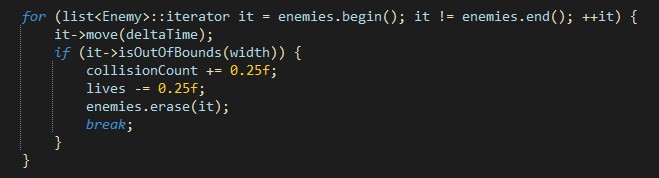
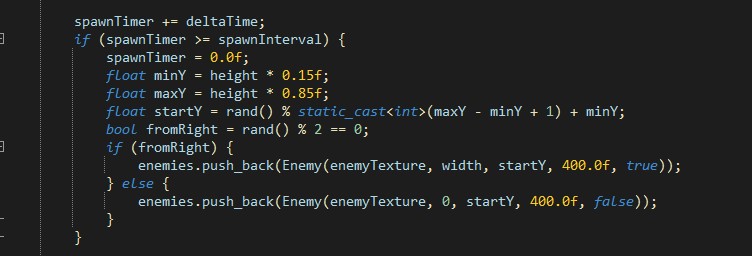
●

A

**Bullet**

class is used to manage each bullet’s position,

direction, and speed.



Project Dissent

Behavior

**●**

**Direction and Speed:**

o

Bullets travel in the direction the player was facing at the time of firing, at a fixed speed (

800.0

f

units per second).

**●**

**Lifespan:**

o

Bullets are removed from the game when they exit the screen or collide with an enemy.



Project Dissent

4

. Game Loop

The game loop is the central mechanism controlling the flow of the game.

Core Functionality

**●**

**Event Handling:**

o

Processes keyboard inputs for movement, shooting, and exiting the game.

**●**

**Time Management:**

o

Uses

deltaTime

to ensure frame rate-independent updates

for smooth gameplay.

**●**

**Updates:**

o

Updates the positions of the player, enemies, and bullets.

o

Handles collisions and interactions between game objects.

**●**

**Rendering:**

o

Clears the window and draws all active game objects, including the background, player,

enemies, bullets, and UI elements (e.g., lives and score).

Game State Management

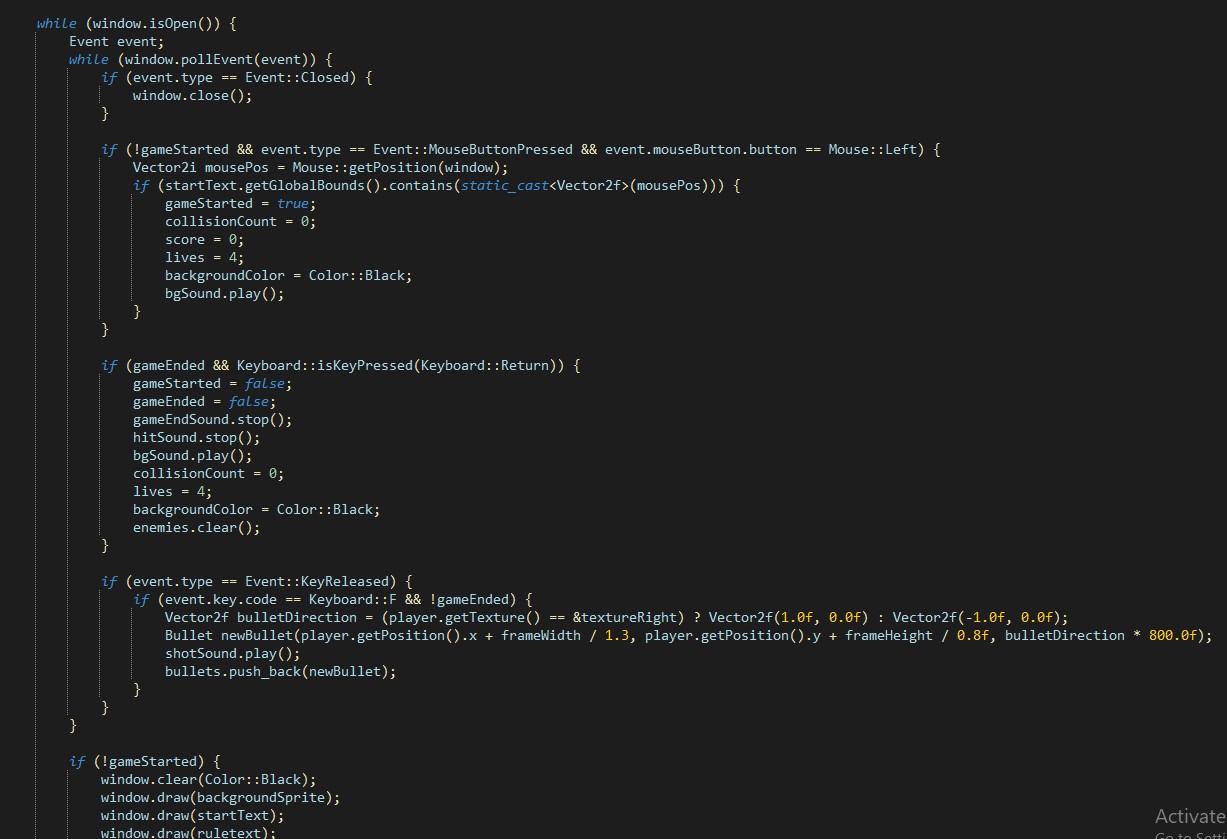
●

Checks for game-ending conditions, such as the player running out of lives or exceeding the maximum

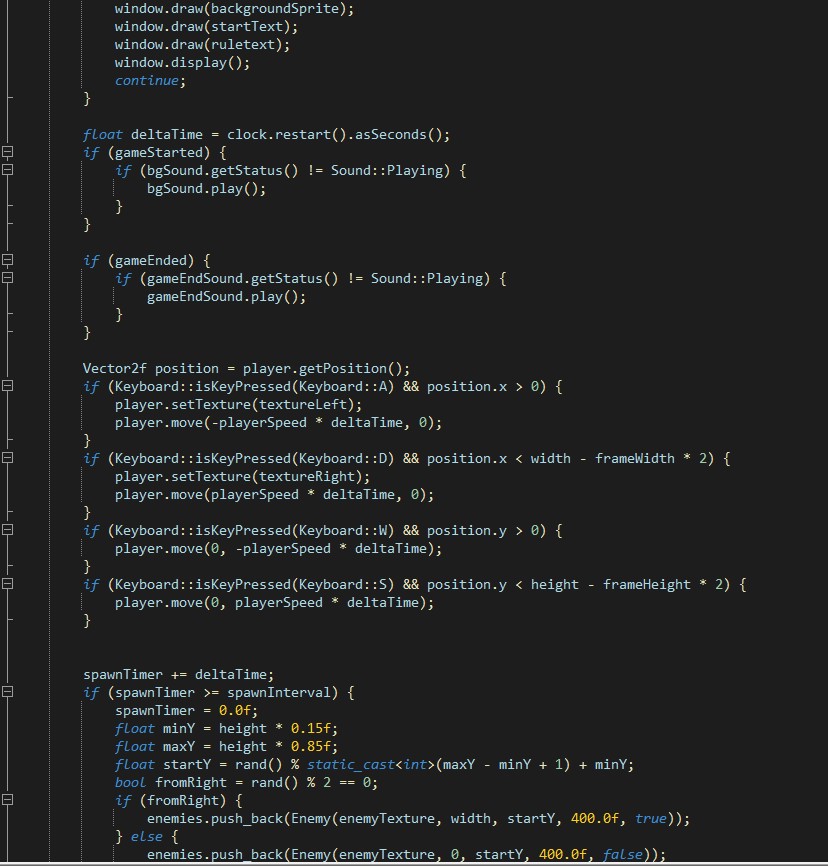
collision count.



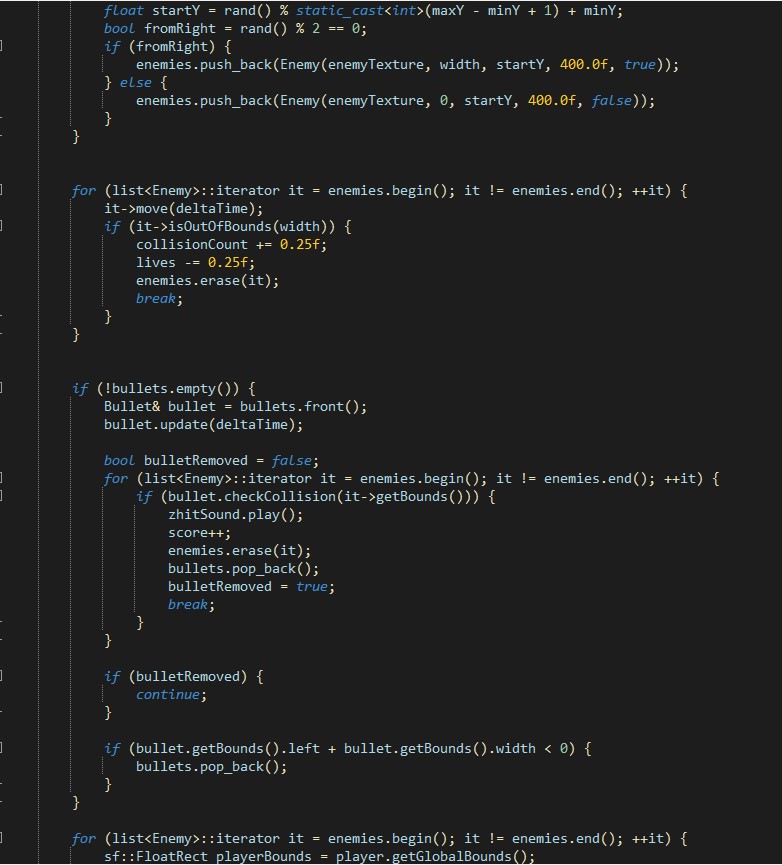
Project Dissent



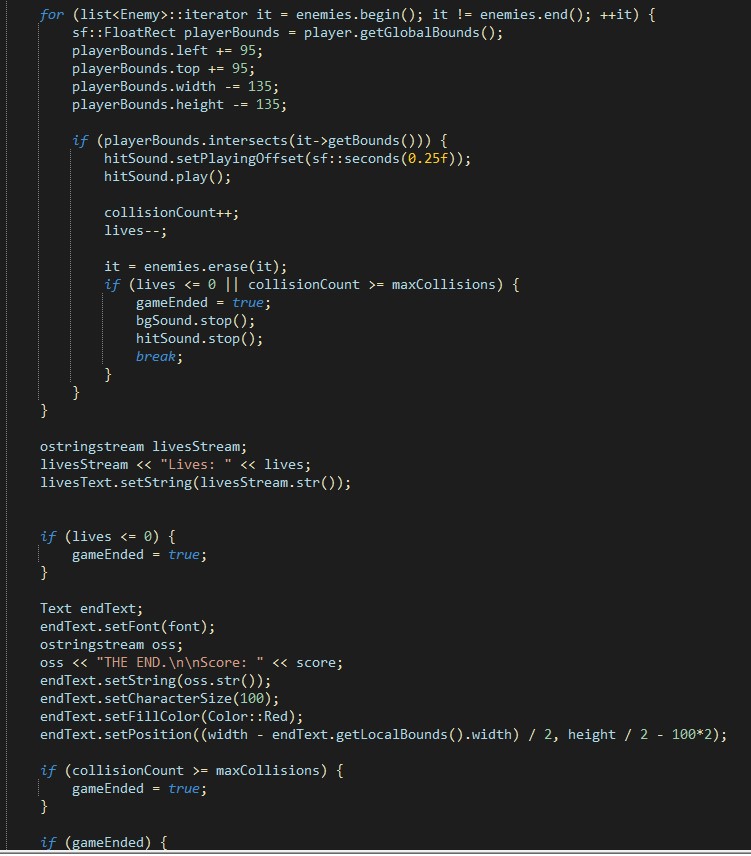
Project Dissent



Project Dissent



Project Dissent



Project Dissent

5

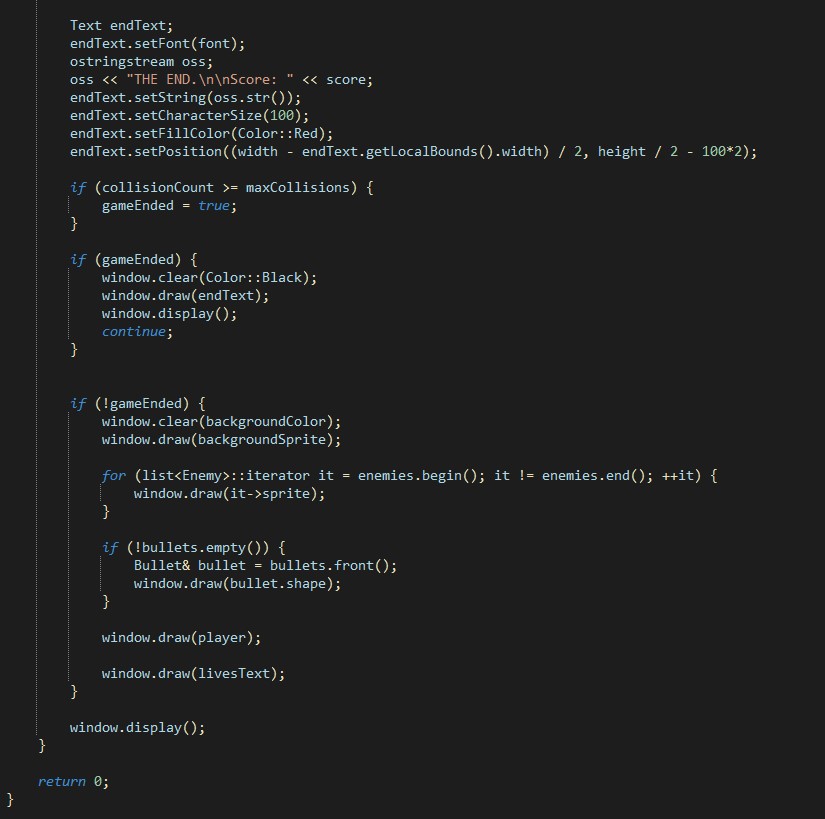
. Sound System in the Game

The sound system is an integral part of the game, enhancing the overall gameplay experience by

providing audio feedback and atmosphere. It covers multiple aspects, such as sound effects for

interactions and background music for immersion. Below is a detailed breakdown of how the sound

system is structured and implemented:



Project Dissent

Components of the Sound System

1

.

**Sound Effects (**

**sf::Sound**

**)**

a.

Short, event-triggered audio clips that provide immediate feedback to player actions.

b.

Examples include:

i.

Bullet firing.

ii.

Enemy destruction.

iii.

Collision between the player and an enemy.

iv.

Game-over alert or life loss.

2

.

**Background Music (**

**sf::Music**

**)**

a.

A looping track that plays throughout the game to set the mood.

b.

Typically calmer or action-themed music, depending on the game's pacing.

3

.

**Audio Assets (**

**sf::SoundBuffer**

**)**

a.

Stores raw audio data for sound effects.

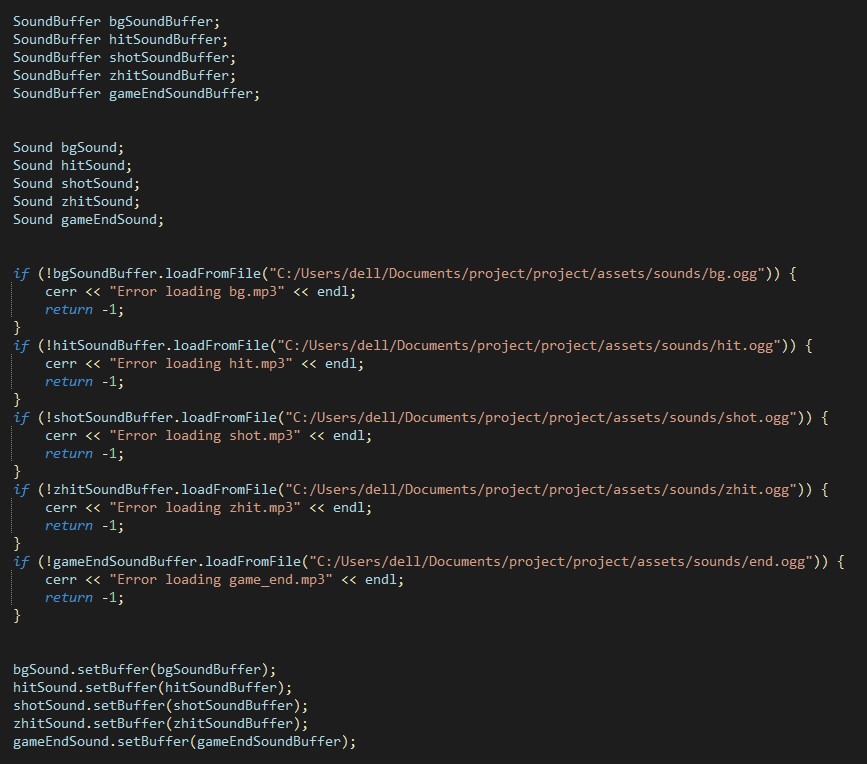
b.

Loaded into

sf::Sound

objects to play specific effects.

Project Dissent



Project Dissent

**DSA Used in Game**

1.

**Vector**

The

**std::vector**

is used to store bullets dynamically.

Each bullet is represented as an instance of the

Bullet class.

**Code Snippet:**

vector

Bullet> bullets;

<

*// Adding a new bullet when the player fires*

if

event.key.code == sf::Keyboard::F && !gameEnded)

(

{

sf::Vector2f bulletDirection = (player.getTexture() == &textureRight)

? sf::Vector2f(

1.0

f

,

f

0.0

)

:

(

sf::Vector2f

f

-1.0

,

0.0

f

)

;

Bullet newBullet(player.getPosition().x + frameWidth / 1.3,

player.getPosition().y + frameHeight / 0.8f,

bulletDirection \* 800.0f);

shotSound.play();

bullets.push\_back(newBullet);

*// Adding to the*

*vector*

}

*// Updating bullets*

for

(

size\_t

i =

0

; i < bullets.size(); ++i) {

bullets[i].update(deltaTime);

*// Removing bullets that move off-screen*

if

(

bullets[i].getBounds().left <

0

|| bullets[i].getBounds().left

>

width)

{

bullets.erase(bullets.begin() + i);

--

i;

*// Adjust the index after removal*

}

}

Why

**std::vector**

?

●

**Dynamic Sizing**

:

Bullets are created dynamically when

the player fires and removed when

off-screen or after hitting enemies.

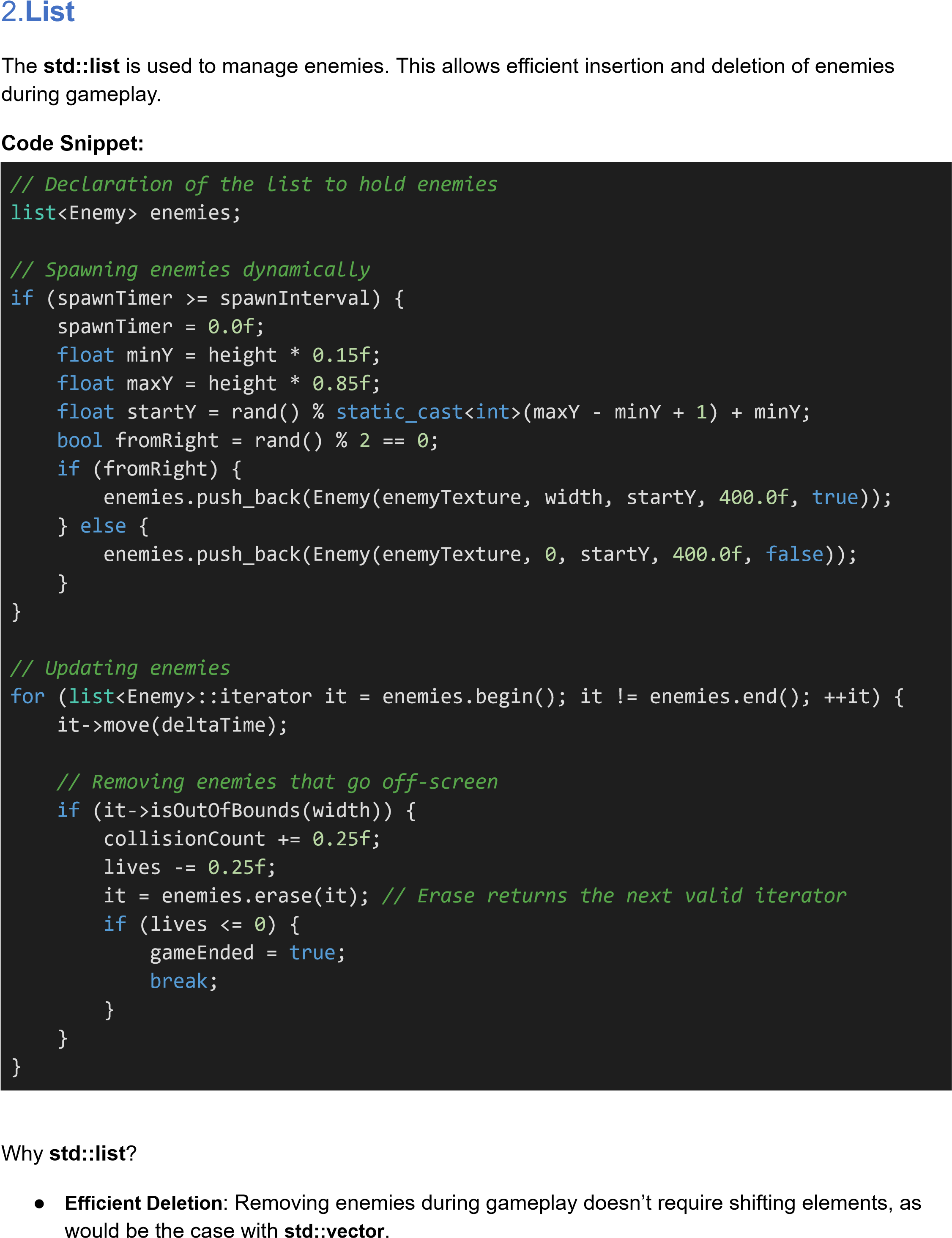
●

**Efficient Access**

Sequential traversal and updates are quick due to contiguous memory

:

storage.



Project Dissent

●

**Iterator Safety**

:

The

**std::list**

iterators remain valid after insertion or deletion, simplifying

collision handling and boundary checks.

3.

**Vector2f**

Used to manage positions and velocities of bullets, enemies, and the player.

**Code Snippet:**

Vector2f velocity(

f

800.0

,

0.0

f

)

;

*// Horizontal movement*

*for bullets*

shape.move(velocity \* deltaTime);

*// Updating bullet*

*position*

*// For enemy movement*

Vector2f position = sprite.getPosition();

sprite.move(speed \* deltaTime,

0

;

)

*// Move based on*

*speed and direction*

Why

**sf::Vector2f**

?

●

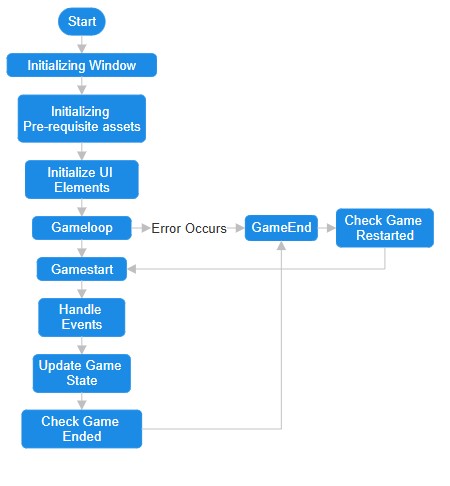
Simplifies coordinate calculations for movement.

●

Overloaded operators make vector arithmetic intuitive.

Project Dissent

Flow Chart



Project Dissent

Conclusion

In conclusion,

*DISSENT*

is an engaging and well-crafted

game that demonstrates the seamless

integration of programming concepts, game mechanics, and efficient data structures. The player,

enemy, and bullet interactions are managed dynamically, showcasing the practical use of data

structures such as queues for orderly bullet handling and lists for flexible enemy management. These

implementations enhance performance, scalability, and responsiveness, creating a smooth gaming

experience. By combining creative design, real-time interaction, and technical precision, the game

highlights the significance of computational thinking in building immersive and interactive applications,

serving as a strong foundation for future advancements in game development.