

# Space Scavenger Documentation

Created by

“DengNoey”

6733185821 Penpitcha Piyawaranont

6733255021 Sirikarn Fugsrimuang

6733293921 Aitsayaphan Limmuangnil

2110215 Programming Methodology

Semester 2 Year 2024

Chulalongkorn University

## Preface

This project report is an integral component of the game development endeavor entitled "Space Scavenger", a project undertaken as part of our autonomous drone development and simulation research. The objective of this project is to design and develop a space-themed survival and exploration game that integrates resource management and navigation mechanics. The project team comprises 6733185821 Penpitcha Piyawaranont, 6733255021 Sirikarn Fugsrimuang, and 6733293921 Aitsayaphan Limmuangnil. We extend our gratitude to Associate Professor Vishnu Kotrajaras, Associate Professor Peerapon Vateekul, and Associate Professor Chate Patanothai, whose expertise and guidance have been instrumental in shaping both the development of this document and the game itself. Additionally, we appreciate the support provided by our colleagues and peers, whose feedback has been invaluable throughout the development process.

Furthermore, we would like to acknowledge the unwavering encouragement and inspiration from our families, who have continuously fostered an environment conducive to learning and innovation.

In conclusion, we hope this document will serve as a valuable resource for students, researchers, and individuals interested in game development and simulation using modern programming methodologies. Should any errors or inconsistencies be found, we welcome your feedback and encourage you to contact the project team for necessary amendments. Your input is greatly appreciated.

The Game Developer Team of Space Scavenger

March 7th, 2025

## Table of contents

<b>Introduction</b>	<b>3</b>
<b>Game play</b>	<b>3</b>
<b>Game rules</b>	<b>3</b>
<b>Game components</b>	<b>4</b>
<b>User interface (UI)</b>	<b>11</b>
<b>Class Diagram</b>	<b>21</b>
<b>Implementation Details</b>	<b>22</b>

## Introduction

---

Space Scavenger is a space-themed adventure and survival game where players take on the role of an astronaut piloting a spacecraft on a crucial mission to travel from Earth to . However, with limited fuel that may not be sufficient for the entire journey, players must explore and collect space debris to convert it into energy fuel. Along the way, they will face numerous challenges and obstacles, making the journey more exciting and unpredictable. "Let's navigate the spaceship to Saturn for a grand mission!"

## Game play

---

The player has to control a spaceship, and your mission is to collect space debris to gain points and progress through different levels while avoiding dangerous obstacles. As you collect more debris, your score increases, and if you reach the required score, you will advance to the next level with increased difficulty. However, if you collide with a space tornado and aliens, the game will be over.

## Game rules





---


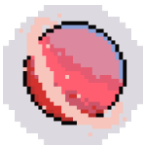

1. In *Space Scavenger*, the player controls a spaceship to collect space debris.
2. The player can obtain special items to increase speed and score.
3. When the player reaches the required score, they can collect the next piece of debris.
4. To progress to the next level, the player must first complete the previous one.

Presentation Video Link : <https://youtu.be/GN9HOUjmzHg?si=Q2ZpWO-i5O4ggLC9>



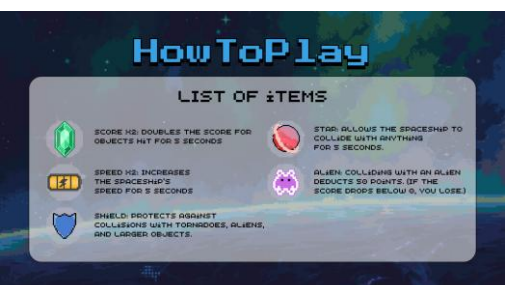
## Game components


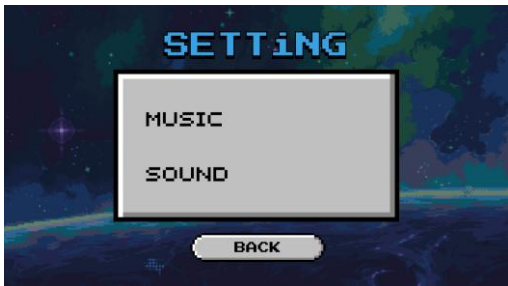



### object and items in game

image	name	description
	image_object_spacescraft	The player's controllable spaceship, used to navigate through space, collect debris, and avoid obstacles.
	image_object_collect_1	An object that the player must collect at the start of the game to gain points and progress.
	image_object_collect_2	An object that the player must collect at the start of the game to gain points and progress.
	image_object_collect_3	An object that the player must collect at the start of the game to gain points and progress.
	image_object_enemy	An object that causes the player to lose the game upon collision unless they have a shield for protection.
	Image_object_powerup_shield	Protects against collisions with tornadoes, aliens, and larger objects.
	Image_object_powerup_scorex2	Doubles the score for objects hit for 5 seconds

	Image_object_powerup_alien	Colliding with an alien deducts 50 points. (If the score drops below 0, you lose.)
	Image_object_powerup_star	Allows the spaceship to collide with anything for 5 seconds.
	Image_object_powerup_speedx2	Increases the spaceship's speedx2 for 5 seconds









## Backgrounds

image	name	description
	image_bg_home	The main menu background, displaying the game title <i>Space Scavenger</i> .
	Image_bg_home_howto play_page_1	The first tutorial page, explaining movement controls and basic gameplay mechanics.
	Image_bg_home_howto play_page_2	The second tutorial page, describing different in-game items and their effects.

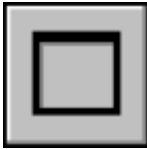
	<p>Image_bg_home_level_selection</p>	<p>The level selection screen, allowing the player to choose between different levels.</p>
	<p>Image_bg_home_setting</p>	<p>The settings menu where the player can adjust music and sound options.</p>
	<p>image_bg_level_1</p>	<p>The background for Level 1</p>
	<p>image_bg_level_2</p>	<p>The background for Level 2</p>
	<p>image_bg_level_3</p>	<p>The background for Level 3</p>

	Image_bg_home_credit	The background for the Credit screen, displaying the names of developers and contributors to the game.
---	----------------------	--




## Buttons



image	name	description
	image_button_playgame	Starts the game and takes the player to level selection.
	image_button_setting	Opens the settings menu to adjust sound and music.
	image_button_credit	Displays the game credits, showing the developers and contributors.
	image_button_exit	Exits the game and closes the application.
	image_button_back	Returns to the previous menu or screen.
	image_button_howtoplay	Opens the tutorial to explain the game mechanics.
	image_button_home	Returns to the main menu from any screen.
	image_button_checkbox_in	Toggles an option on (e.g., enabling/disabling music or sound).



	image_button_checkbox_out	Toggles an option off (e.g., enabling/disabling music or sound).
---	---------------------------	--

## Others

image	name	description
	image_button_level_check	Represents a completed or unlocked level that the player has successfully passed.
	image_button_level_locked	Represents a locked level that the player cannot access until they complete the previous one.
	image_button_level_locked	Another representation of a locked level, possibly indicating a different lock condition.

	image_other_youwin	The victory screen displayed when the player successfully completes the level. Options to Play Again or return to the Menu.
	image_other_gameover	The game over screen displayed when the player fails the level. Options to Play Again or return to the Menu.

## Background Music and special effect

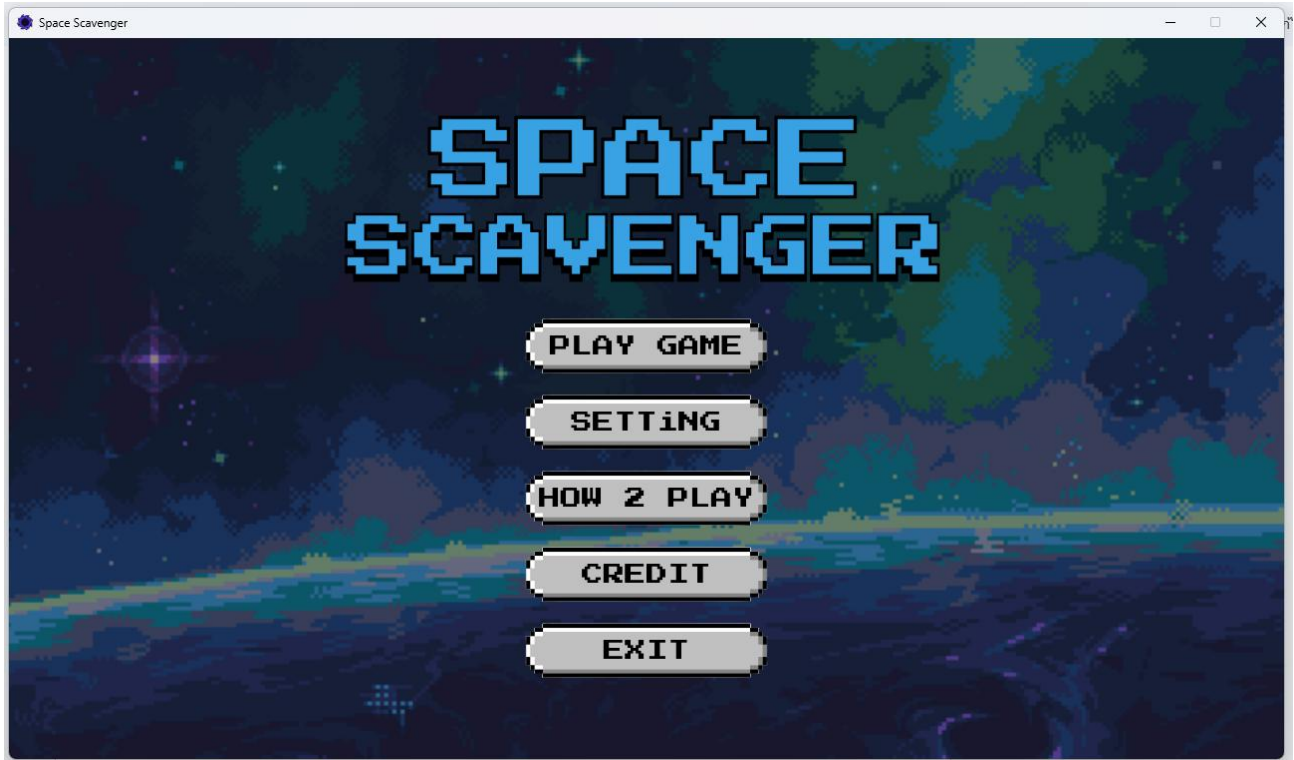
name	description
sound_bg_ending.mp4	Background music played at the end of the game, typically after a win or loss to provide a sense of closure.
sound_bg_level_1_2.mp4	Background music for Levels 1 and 2, setting the atmosphere for the early stages of the game.
sound_bg_level_3.mp4	Background music for Level 3, often more intense to match the increased difficulty of the final stage.
sound_bg_lost.mp4	A background track played when the player loses, emphasizing failure and encouraging retrying.
sound_bg_opening.mp4	Background music for the main menu or title screen, creating an immersive first impression for players.
sound_bg_win.mp4	A celebratory background track played

	when the player wins a level or the game, enhancing the sense of achievement.
sound_fx_alien.mp4	A sound effect triggered when the player collects the alien power-up, indicating a score reduction penalty.
sound_fx_eating_trash.mp4	A sound effect played when the player successfully collects space debris, reinforcing positive feedback.
sound_fx_meteor.mp4	A sound effect for meteor collisions, providing an impactful audio cue when an asteroid appears or hits something.
sound_fx_on_click.mp4	A standard click sound effect for UI buttons, improving user experience by providing audible feedback on interactions.
sound_fx_storm.mp4	A sound effect played when activating the shield power-up, signaling temporary protection from hazards.
sound_fx_x2.mp4	A sound effect for the score multiplier or speed boost power-up, emphasizing the activation of a special ability.

## User interface (UI)

---

Start Menu before entering the game



Click **play game** to select the levels page.

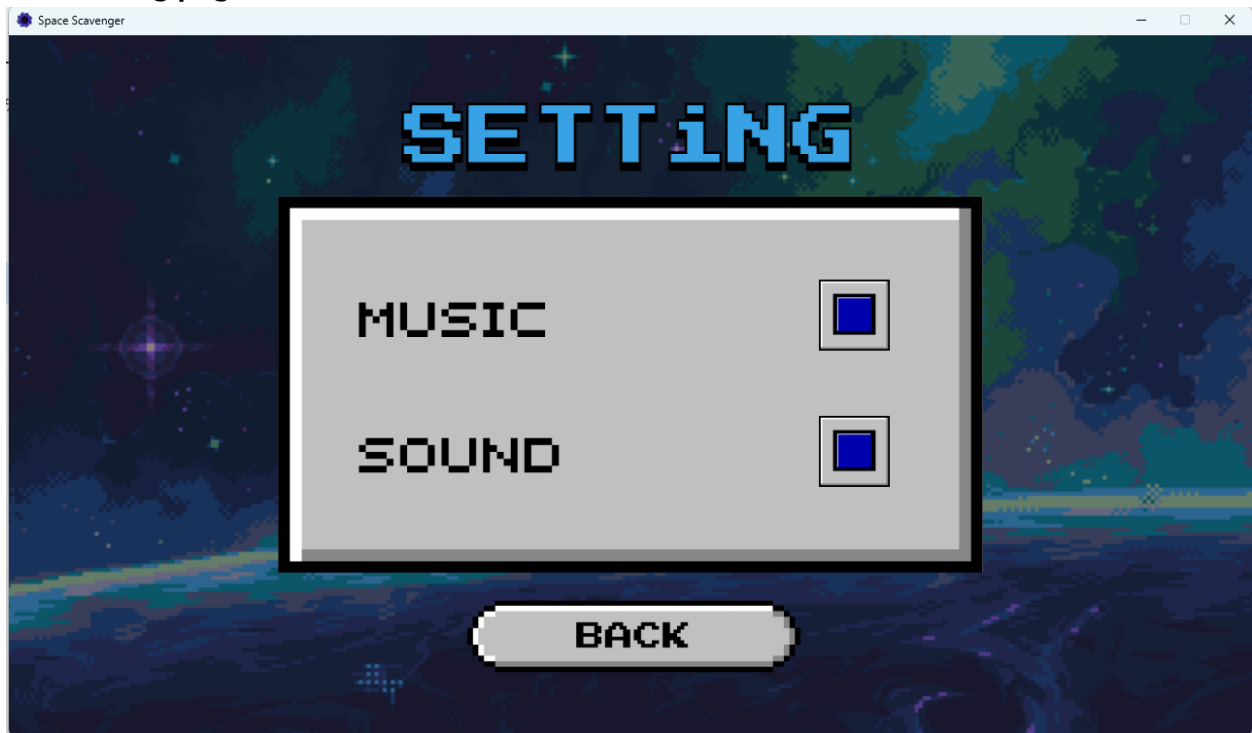
Click **setting** to Adjust game settings.

Click **how 2 play** to go to the tutorial page.

Click **credit** to go to the credit page.

Click **exit to** exit the game.

## The setting page

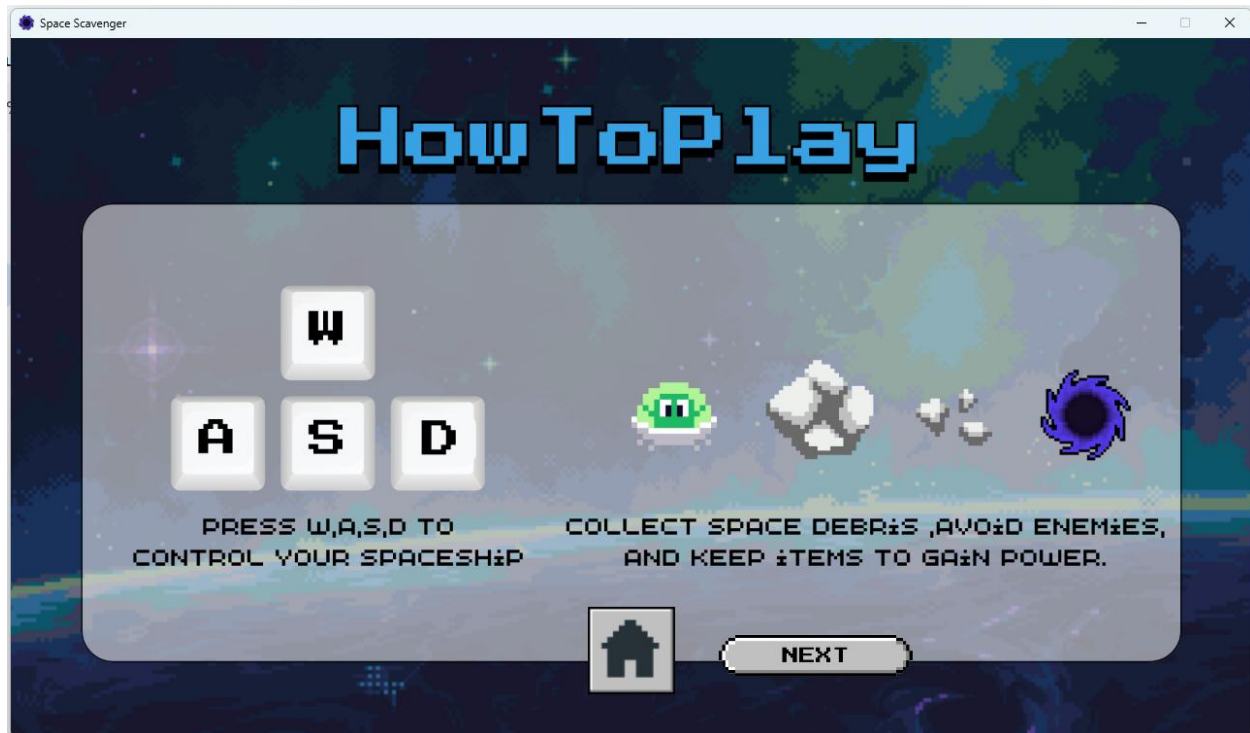


The setting page.

Click the **checkboxes** to toggle Music and Sound on or off.

Click **back** to return to the previous menu.

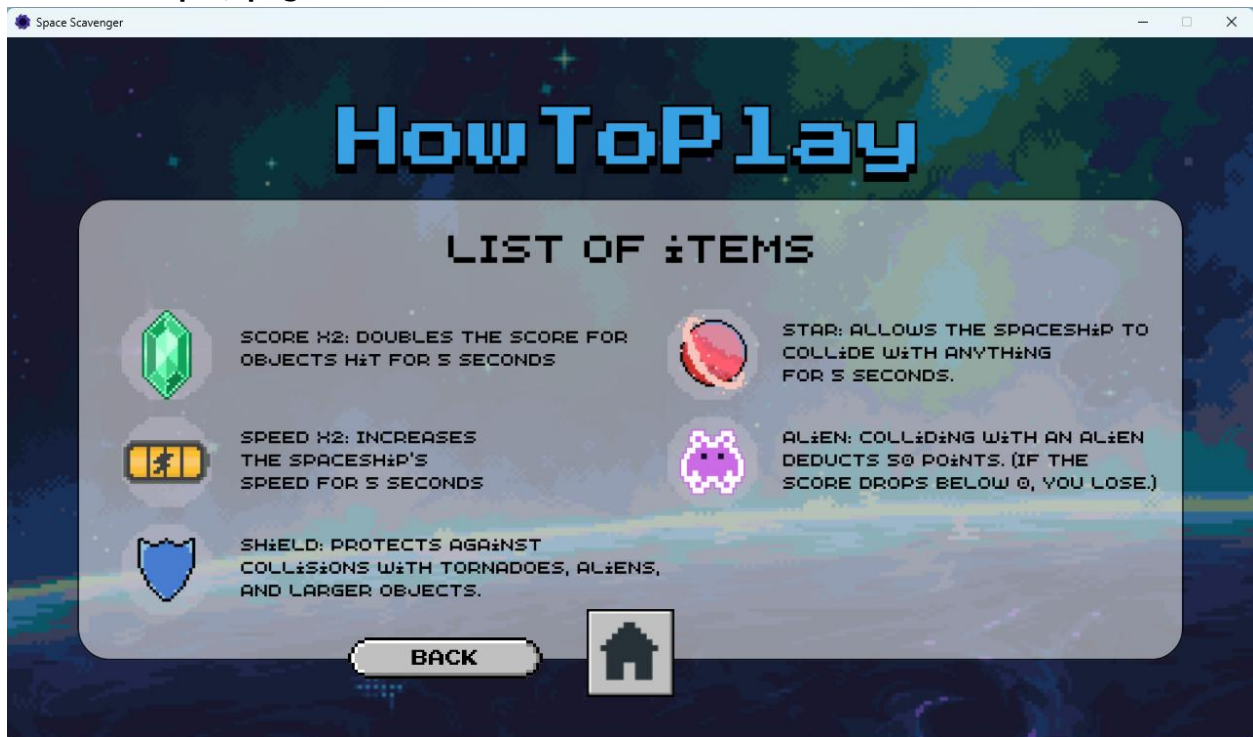
## The How to play page (1)



Click the **Home** button to return to the main menu.

Click **Next** to see more instructions.

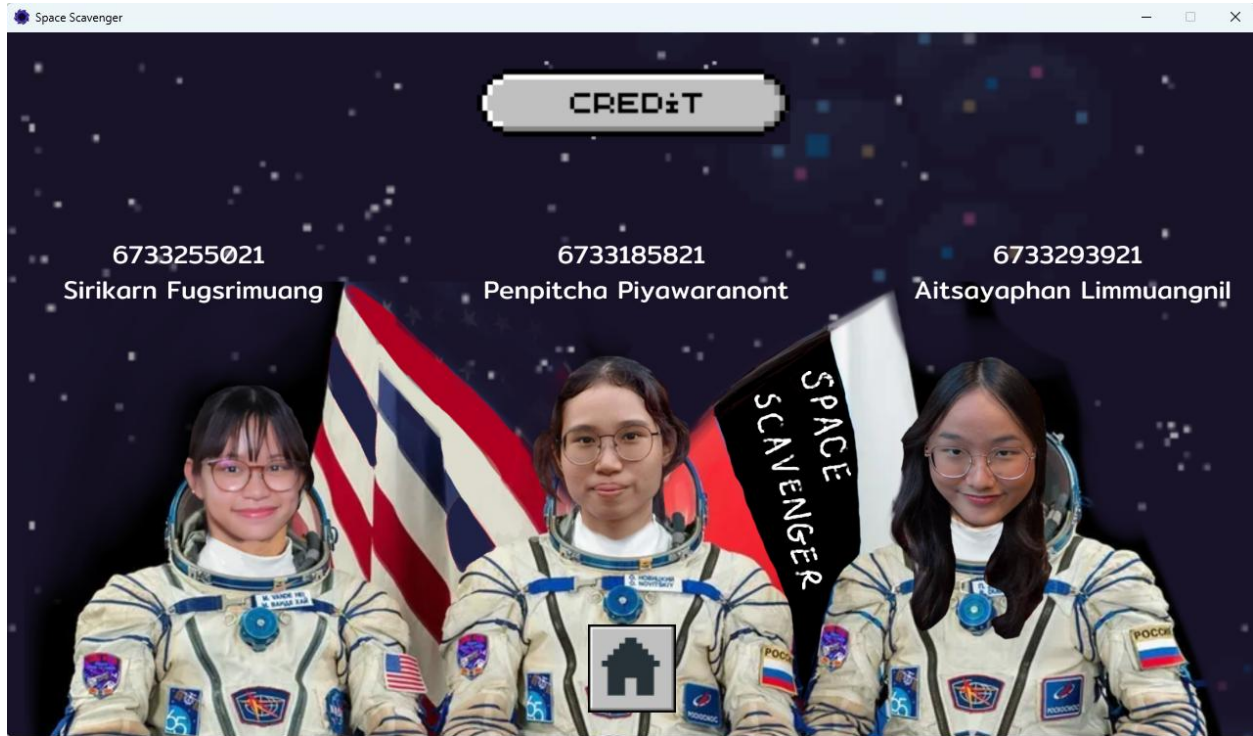
## The How to play page (2)



Click **back** to return to the previous page.

Click the **Home button** to return to the main menu.

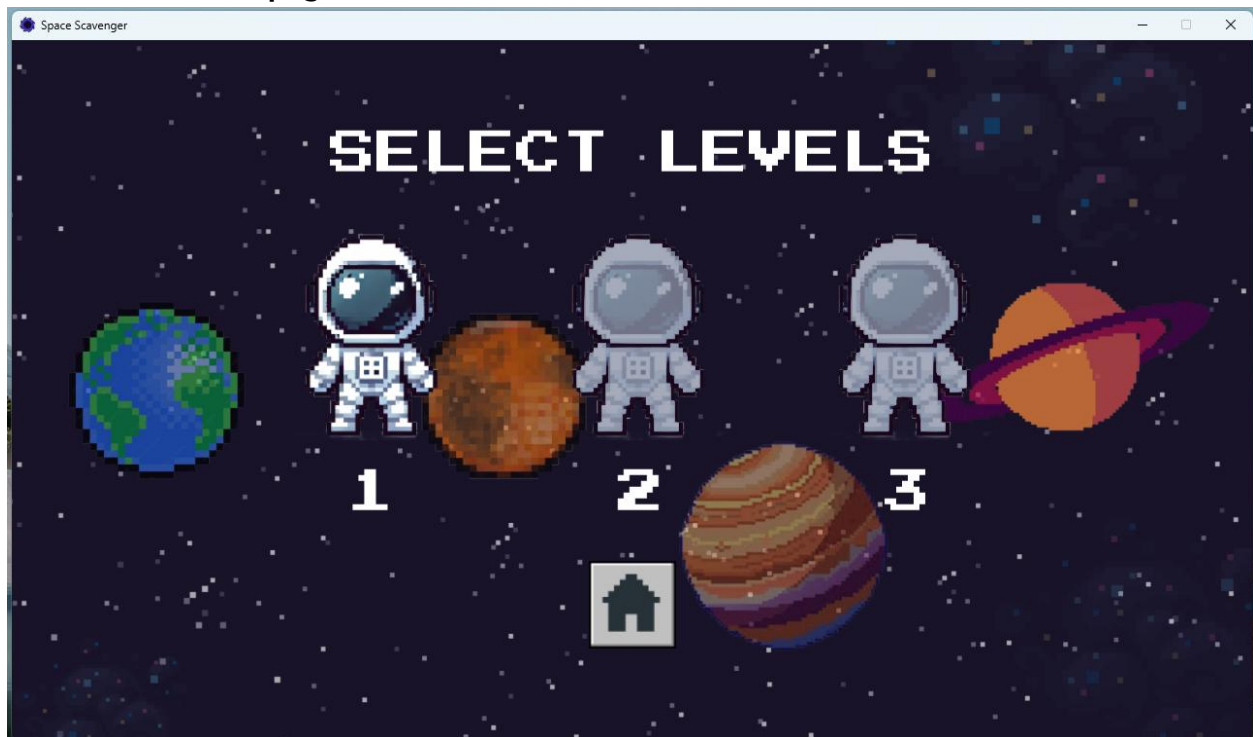
## The end credit page



Click the **Home button** to return to the main menu.



## The Select Levels page



Click on **astronauts 1** to play on level 1.

Click on **astronauts 2** to play on level 2.(*Complete Level 1 to unlock.*)

Click on **astronauts 3** to play on level 3.(*Complete Level 2 to unlock.*)

Click the **Home button** to return to the main menu.

## The Level 1 Gameplay Screen of Space Scavenger



## HUD (Heads-Up Display) Explanation in Space Scavenger



### 1. Fuel Level & Progress Bar:

- Displays the current fuel level.
- If the bar is full, the player wins the level.
- When full, the player can collide with objects without taking damage.

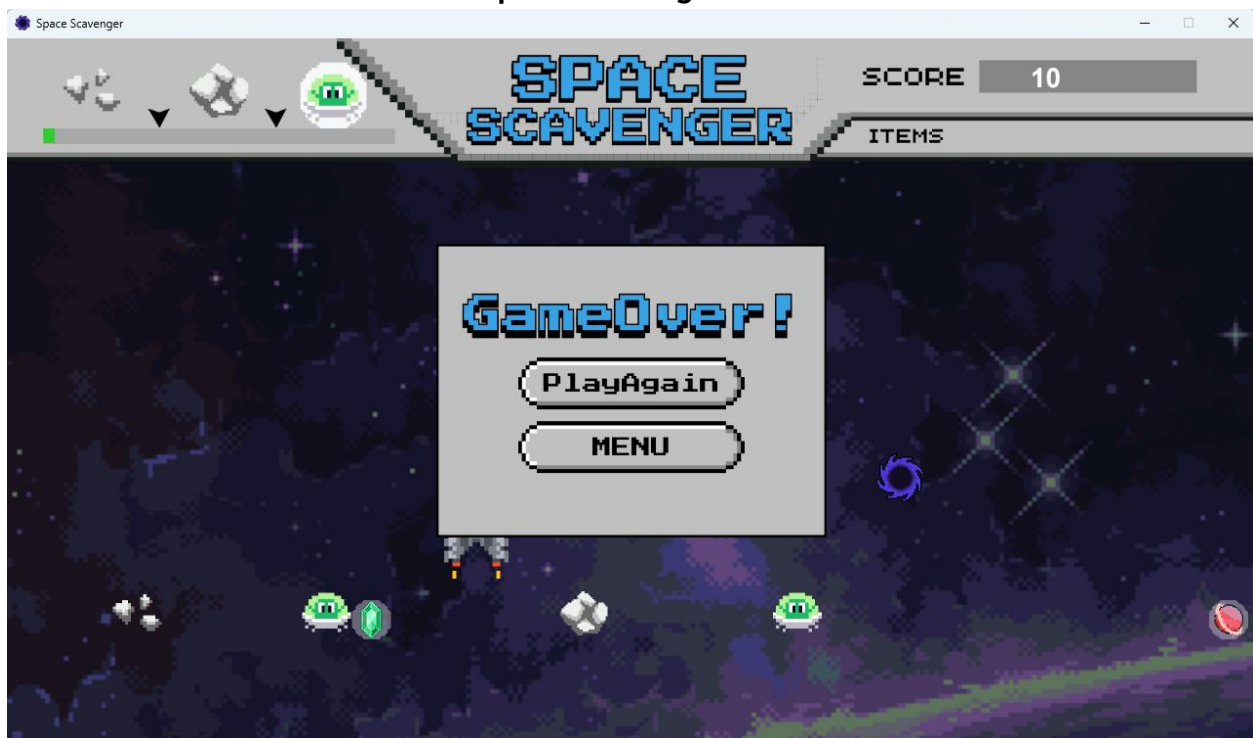
### 2. Score Indicator:

- Shows the player's current score.

### 3. Active Items Display:

- Displays the power-ups or items currently in use.

## The Level 1 Gameover Screen of Space Scavenger



Click **Play Again**: Restart the level from the beginning.

Click **Menu**: Return to the main menu.

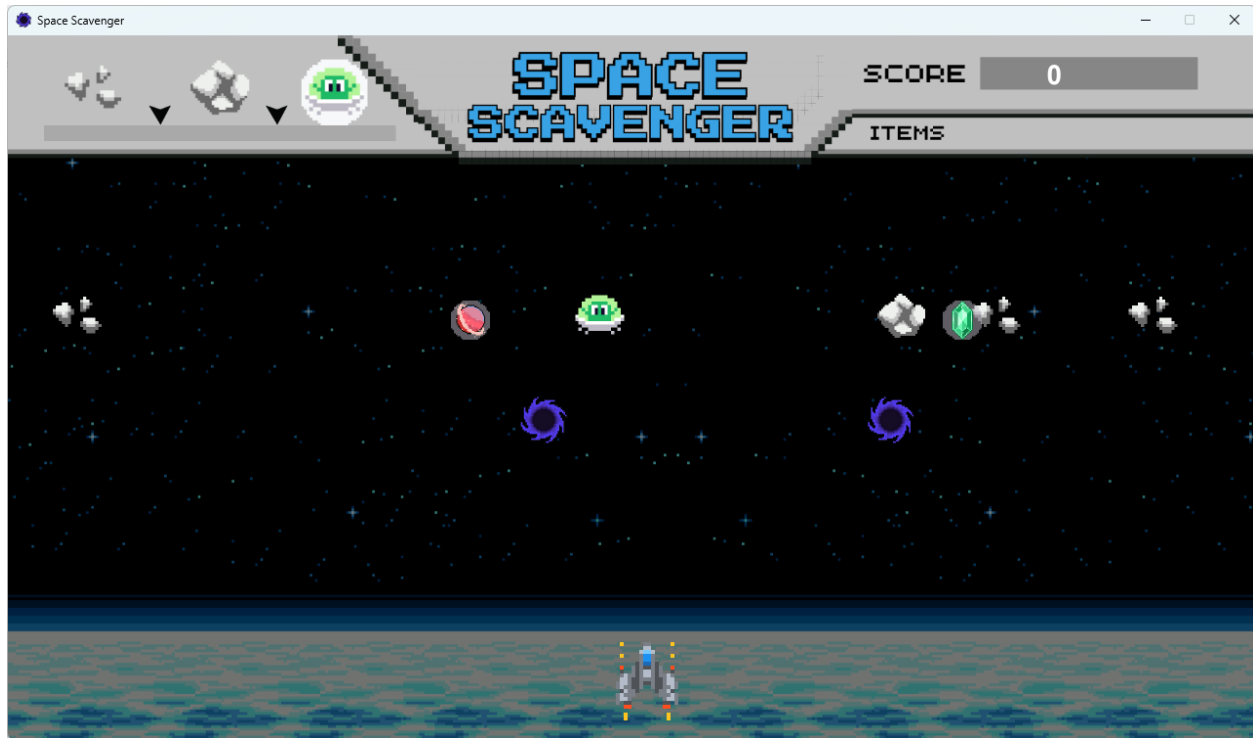
## The Level 2 Gameplay Screen of Space Scavenger



## The Level 2 Gameover Screen of Space Scavenger



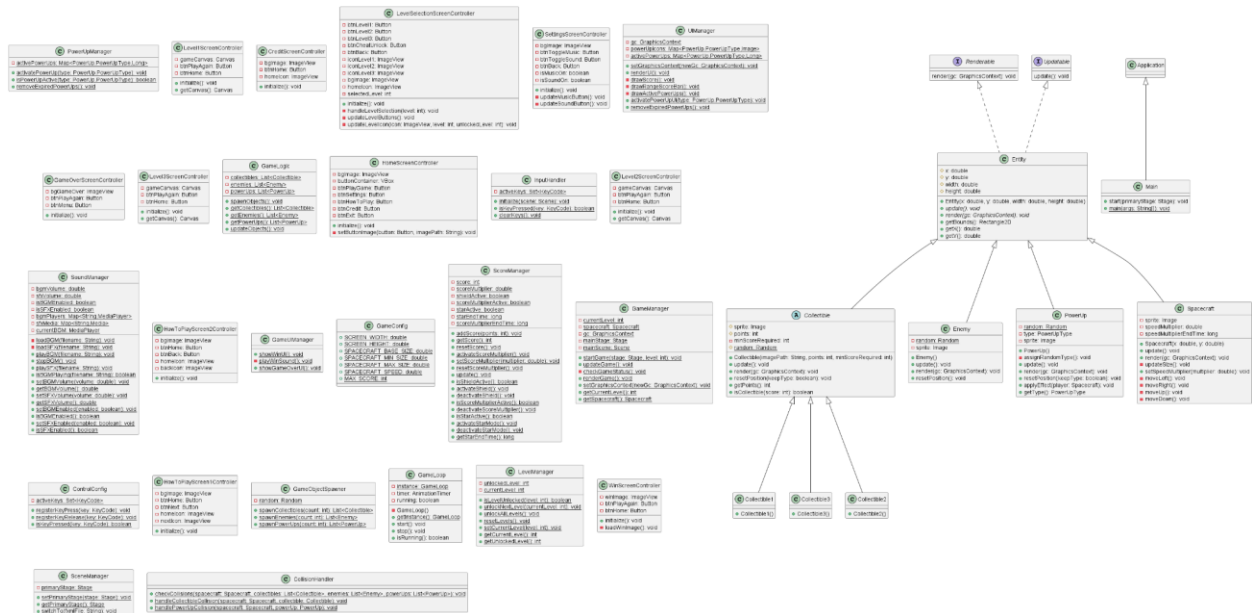
## The Level 3 Gameplay Screen of Space Scavenger



## The Level 3 Gameover Screen of Space Scavenger



The picture of class diagram is shown as follow



## Implementation Details

---

The following detail is the detailed description for implementation of the game.

Noted that the access modifier notations can be listed below.

- + public
- # protected
- - private
- underlined static
- ALLCAPS final variable
- *italic* abstract

### 1. Package main

#### 1.1 Class GameConfig

This class stores global configuration values for the game. These constants define screen dimensions, spacecraft properties, and game mechanics such as speed and scoring limits.

##### *Fields*

Name	Description
+ <u>double</u> SCREEN_WIDTH	The width of the game screen
+ <u>double</u> SCREEN_HEIGHT	The height of the game screen
+ <u>double</u> SPACECRAFT_BASE_SIZE	The default size of the player's spacecraft
+ <u>double</u> SPACECRAFT_MIN_SIZE	The minimum size the spacecraft can shrink to
+ <u>double</u> SPACECRAFT_MAX_SIZE	The maximum size the spacecraft can grow to
+ <u>double</u> SPACECRAFT_SPEED	The movement speed of the spacecraft
+ <u>int</u> MAX_SCORE	The maximum achievable score before winning the game

## 1.2 Class GameLoop

This class manages the core update cycle of the game using an AnimationTimer. It follows the singleton pattern to ensure only one instance of the game loop exists at any given time.

### Fields

Name	Description
- <u>GameLoop</u> instance	Singleton instance of the game loop
- AnimationTimer timer	The main loop for updating the game state and rendering
- boolean running	Indicates whether the game loop is active

### Methods

Name	Description
+ <u>GameLoop</u> getInstance()	Returns the singleton instance of GameLoop
+ void start()	Starts the game loop if not already running
+ void stop()	Stops the game loop
+ boolean isRunning()	Checks if the game loop is currently running

## 1.3 Class Main extends Application

This class serves as the entry point for the game, initializing the JavaFX application and setting up the primary stage.



### Methods

Name	Description
+ void start(Stage primaryStage)	Initializes and sets up the game window, loads the home screen, and displays the stage
+ <u>static void main(String[] args)</u>	Launches the JavaFX application

## 2. Package input

### 2.1 Class ControlConfig

This class handles player key input tracking by maintaining a set of currently pressed keys.

### Fields

Name	Description
- <u>Set&lt;KeyCode&gt; activeKeys</u>	Stores currently pressed keys

### Methods

Name	Description
+ <u>void registerKeyPress(KeyCode key)</u>	Adds a key to the active keys set when pressed
+ <u>void registerKeyRelease(KeyCode key)</u>	Removes a key from the active keys set when released
+ <u>boolean isKeyPressed(KeyCode key)</u>	Checks if a key is currently pressed

### 2.2 Class InputHandler

This class manages keyboard input events by storing currently pressed keys and handling key press/release events.

### Fields

Name	Description
- <u>Set&lt;KeyCode&gt; activeKeys</u>	Stores currently pressed keys

### Methods

Name	Description
+ <u>void initialize(Scene scene)</u>	Sets up key event listeners for the given scene
+ <u>boolean isKeyPressed(KeyCode key)</u>	Checks if a key is currently pressed
+ <u>void clearKeys()</u>	Clears all active keys and logs the action for debugging

## 3. Package entity

### 3.1 Abstract Class Collectible extends Entity

This class represents a general collectible object that the player can collect to gain points.

### Fields

Name	Description
# Image sprite	Image representing the collectible
# int points	Points awarded when collected
- int minScoreRequired	Minimum score required to collect the item
# <u>Random random</u>	Random instance for generating positions

### Methods

Name	Description
+ Collectible(String imagePath, int points, int minScoreRequired)	Constructor initializes collectible properties
+ void update()	Moves the collectible downward and resets position when off-screen
+ void render(GraphicsContext gc)	Draws the collectible on the canvas
+ void resetPosition(boolean keepType)	Resets the collectible position when needed
+ int getPoints()	Returns the points of the collectible
+ boolean isCollectible(int score)	Checks if the player has the required score to collect the item

### 3.2 Class Collectible1 extends Collectible

This class is a specific type of collectible that the player can collect in the game.

### Methods

Name	Description
+ Collectible1()	Constructor initializes the collectible with a specific image, point value, and minimum score requirement

### 3.3 Class Collectible2 extends Collectible

This class is a specific type of collectible that the player can collect in the game.

#### Methods

Name	Description
+ Collectible2()	Constructor initializes the collectible with a specific image, point value, and minimum score requirement

### 3.4 Class Collectible3 extends Collectible

This class is a specific type of collectible that the player can collect in the game.

#### Methods

Name	Description
+ Collectible3()	Constructor initializes the collectible with a specific image, point value, and minimum score requirement

### 3.5 Class Enemy extends Entity

This class represents an obstacle (Tornado) in the game that players must avoid.

#### Fields

Name	Description
- <u>Random random</u>	Used to generate random positions
- Image sprite	Holds the image representation of the enemy

#### Methods

Name	Description
+ Enemy()	Constructor initializes the enemy with a random position and an image
+ void update()	Moves the enemy downward and resets its position if it moves off-screen

+ void render(GraphicsContext gc)	Draws the enemy on the screen
+ void resetPosition()	Resets the enemy's position to the top of the screen

### 3.6 Abstract Class Entity

This class serves as the base class for all game objects, defining common properties such as position, size, and rendering.

#### Fields

Name	Description
# double x, y	Represents the position of the entity
# double width, height	Represents the size of the entity

#### Methods

Name	Description
+ Entity(double x, double y, double width, double height)	Constructor initializes position and size
+ abstract void update()	Abstract method for updating the entity's behavior
+ abstract void render(GraphicsContext gc)	Abstract method for rendering the entity
+ Rectangle2D getBounds()	Returns a bounding box for collision detection
+ double getX()	Returns the x position.
+ double getY()	Returns the y position.

### 3.7 Class PowerUp extends Entity

This class represents special collectible items that provide various effects when collected.

### Fields

Name	Description
- PowerUpType type	Stores the type of the power-up
- Image sprite	The image representing the power-up
- <u>Random random</u>	Used for generating random types and positions
+ enum PowerUpType	Enum SPEED_X2, SCORE_X2, SHIELD, ALIEN, STAR

### Methods

Name	Description
+ PowerUp()	Constructor that initializes the power-up with a random type and position
- void assignRandomType()	Assigns a random type and image to the power-up
+ void update()	Moves the power-up downwards and resets if it exits the screen
+ void render(GraphicsContext gc)	Draws the power-up on the game screen.
+ void resetPosition(boolean keepType)	Resets the power-up's position, optionally keeping its type
+ void applyEffect(Spacecraft player)	Applies the power-up's effect to the player
+ PowerUpType getType()	Returns the type of the power-up

### 3.8 Interface Renderable

This interface ensures that objects implementing it can be drawn on the game screen.

### Methods

Name	Description
void render(GraphicsContext gc)	Defines a method to render the object on a GraphicsContext

## 3.9 Class Spacecraft extends Entity

This class represents the player-controlled spaceship.

### Fields

Name	Description
- Image sprite	The image representing the spacecraft
- double speedMultiplier	A speed multiplier for power-ups
- long speedMultiplierEndTime	The time when the speed multiplier effect expires

### Methods

Name	Description
+ void update()	Updates movement and size based on player input and power-up effects
+ void render(GraphicsContext gc)	Draws the spacecraft on the screen
+ void updateSize()	Adjusts the spacecraft's size based on the player's score
+ void setSpeedMultiplier(double multiplier)	Sets the speed boost effect
- void moveLeft(), moveRight(), moveUp(), moveDown()	Moves the spacecraft in different directions

## 3.10 Interface Updatable

This interface defines objects that require updates every frame. This interface is implemented by all game objects that need to move or change over time.

### Methods

Name	Description
void update()	Updates the object's state.

## 4. Package gamelogic

### 4.1 Class CollisionHandler

This class is responsible for detecting and resolving collisions in the game. This class ensures that game objects interact properly when they collide.

### Methods

Name	Description
+ <u>checkCollisions(Spacecraft, List&lt;Collectible&gt;, List&lt;Enemy&gt;, List&lt;PowerUp&gt;)</u>	Checks if the spacecraft collides with collectibles, enemies, or power-ups
+ <u>handleCollectibleCollision(Spacecraft, Collectible)</u>	Determines if a collectible is within the valid score range before collecting
+ <u>handlePowerUpCollision(Spacecraft, PowerUp)</u>	Activates the effects of power-ups when collected

### 4.2 Class GameLogic

This class manages the game's core mechanics by handling object spawning and updating, ensuring that objects are created and updated properly within the game loop.

### Fields

Name	Description
- <u>List&lt;Collectible&gt; collectibles</u>	Stores all collectible objects
- <u>List&lt;Enemy&gt; enemies</u>	Stores all enemy objects
- <u>List&lt;PowerUp&gt; powerUps</u>	Stores all power-up objects



### Methods

Name	Description
+ <u>void spawnObjects()</u>	Calls GameObjectSpawner to generate new collectibles, enemies, and power-ups
+ <u>List getCollectibles(), getEnemies(), getPowerUps()</u>	Provide access to game objects
+ <u>void updateObjects()</u>	Updates the state of all objects

### 4.3 Class GameManager

This class controls the game's main flow, handling game initialization, updates, and rendering, ensuring the game state is managed properly throughout the game lifecycle.

### Fields

Name	Description
- <u>int currentLevel</u>	Tracks the current level number
- <u>Spacecraft spacecraft</u>	The player's spacecraft
- <u>GraphicsContext gc</u>	Graphics context for rendering
- <u>Stage mainStage</u>	The main game window
- <u>Scene mainScene</u>	The active game scene

### Methods

Name	Description
+ <u>void startGame(Stage stage, int level)</u>	Initializes and starts a new game
+ <u>void updateGame()</u>	Updates the game state each frame
+ <u>void checkGameStatus()</u>	Checks if the player has won or lost
+ <u>void renderGame()</u>	Renders game objects and UI
+ <u>void setGraphicsContext(GraphicsContext</u>	Sets the graphics context

<u>newGc()</u>	
+ <u>int getCurrentLevel()</u>	Returns the current level
+ <u>Spacecraft getSpacecraft()</u>	Returns the player's spacecraft

#### 4.4 Class GameObjectSpawner

This class is responsible for generating different game objects dynamically, ensuring help in dynamically generating game objects at the start of the game.

##### Fields

Name	Description
- <u>Random random</u>	A Random object used for generating random types of collectibles and power-ups.

##### Methods

Name	Description
+ <u>List&lt;Collectible&gt; spawnCollectibles(int count)</u>	Spawns a specified number of collectibles with random types
+ <u>List&lt;Enemy&gt; spawnEnemies(int count)</u>	Spawns a specified number of enemies
+ <u>List&lt;PowerUp&gt; spawnPowerUps(int count)</u>	Spawns a specified number of power-ups with random types

#### 4.5 Class GameUIManager

This class manages the game-over and win screens, ensuring proper UI transitions and sound effects when the player wins or loses.

##### Methods

Name	Description
+ <u>void showWinUI()</u>	Displays the win screen, unlocks the next level, stops background music, and plays

	the win sound effect
+ <u>void playWinSound()</u>	Play the sound effect when the player wins.
+ <u>void showGameOverUI()</u>	Displays the game-over screen, stops background music, and plays the game-over sound effect.

#### 4.6 Class LevelManager

This class handles level progression, unlocking, and selection, ensuring that players can only access levels they have unlocked and provides cheat functionality to unlock all levels.

##### Fields

Name	Description
- <u>int unlockedLevel</u>	Stores the highest unlocked level
- <u>int currentLevel</u>	Stores the level currently selected by the player

##### Methods

Name	Description
+ <u>boolean isLevelUnlocked(int level)</u>	Checks if the given level is unlocked
+ <u>void unlockNextLevel(int currentLevel)</u>	Unlocks the next level if possible
+ <u>void unlockAllLevels()</u>	Unlocks all levels (used in cheat mode)
+ <u>void resetLevels()</u>	Resets unlocked levels back to level 1
+ <u>void setCurrentLevel(int level)</u>	Sets the current level
+ <u>int getCurrentLevel()</u>	Retrieves the currently selected level
+ <u>int getUnlockedLevel()</u>	Retrieves the highest unlocked level

#### 4.7 Class PowerUpManager

This class tracks and manages power-ups that have been collected by the player, ensuring that power-ups are properly handled and expire after a set duration.

##### Fields

Name	Description
+ <u>Map&lt;PowerUp.PowerUpType, Long&gt; activePowerUps</u>	A map that stores active power-ups and their expiration times. Methods

##### Methods

Name	Description
+ <u>void activatePowerUp(PowerUp.PowerUpType type)</u>	Activates a power-up and sets its duration
+ <u>boolean isPowerUpActive(PowerUp.PowerUpType type)</u>	Checks if a specific power-up is still active
+ <u>void removeExpiredPowerUps()</u>	Removes power-ups that have expired

#### 4.8 Class ScoreManager

This class is responsible for managing the player's score, power-up effects related to score calculation, and checking win conditions, ensuring that the scoring system and special power-up effects related to score calculation function properly.

##### Fields

Name	Description
- <u>int score</u>	Stores the player's current score
- <u>double scoreMultiplier</u>	Stores the score multiplier value
- <u>boolean shieldActive</u>	Indicates if the shield is currently active

- <u>boolean scoreMultiplierActive</u>	Indicates if the score multiplier is active
- <u>boolean starActive</u>	Indicates if the star mode is active
- <u>long starEndTime</u>	Stores the expiration time for star mode
- <u>long scoreMultiplierEndTime</u>	Stores the expiration time for the score multiplier

### Methods

Name	Description
+ <u>void addScore(int points)</u>	Adds points to the player's score and checks for a win condition
+ <u>int getScore()</u>	Returns the current score
+ <u>void resetScore()</u>	Resets the score to zero
+ <u>void activateScoreMultiplier()</u>	Activates the score multiplier for a short duration
+ <u>void setScoreMultiplier(double multiplier)</u>	Sets a custom multiplier for score calculation
+ <u>void resetScoreMultiplier()</u>	Resets the multiplier back to default
+ <u>void update()</u>	Checks and deactivates expired multipliers and star mode
+ <u>boolean isShieldActive()</u>	Returns whether the shield is active
+ <u>void activateShield()</u>	Activates the shield
+ <u>void deactivateShield()</u>	Deactivates the shield
+ <u>boolean isScoreMultiplierActive()</u>	Returns whether the score multiplier is active
+ <u>void deactivateScoreMultiplier()</u>	Deactivates the score multiplier
+ <u>boolean isStarActive()</u>	Returns whether the star mode is active.

+ <u>void activateStarMode()</u>	Activates the star mode for a limited duration
+ <u>void deactivateStarMode()</u>	Deactivates the star mode
+ <u>long getStarEndTime()</u>	Returns the expiration time for the star mode

#### 4.9 Class SoundManager

This class is responsible for handling background music (BGM) and sound effects (SFX) in the game, ensuring that all audio-related operations in the game are handled efficiently.

##### Fields

Name	Description
- <u>double bgmVolume</u>	Stores the background music volume
- <u>double sfxVolume</u>	Stores the sound effect volume
- <u>boolean isBGMEEnabled</u>	Determines if background music is enabled
- <u>boolean isSFXEnabled</u>	Determines if sound effects are enabled
- <u>Map&lt;String, MediaPlayer&gt; bgmPlayers</u>	Stores MediaPlayer instances for different background music tracks
- <u>Map&lt;String, Media&gt; sfxMedia</u>	Stores Media objects for sound effects
- <u>MediaPlayer currentBGM</u>	Stores the currently playing background music

##### Methods

Name	Description
+ <u>void loadBGM(String filename)</u>	Loads background music into memory
+ <u>void loadSFX(String filename)</u>	Loads sound effects into memory
+ <u>void playBGM(String filename)</u>	Plays the specified background music

+ <u>void stopBGM()</u>	Stops the currently playing background music
+ <u>void playSFX(String filename)</u>	Plays the specified sound effect
+ <u>boolean isBGMPlaying(String filename)</u>	Checks if a specific BGM is currently playing
+ <u>void setBGMVolume(double volume)</u>	Sets the background music volume
+ <u>double getBGMVolume()</u>	Gets the current background music volume
+ <u>void setSFXVolume(double volume)</u>	Sets the sound effects volume
+ <u>double getSFXVolume()</u>	Gets the current sound effects volume
+ <u>void setBGMEabled(boolean enabled)</u>	Enables or disables background music
+ <u>boolean isBGMEabled()</u>	Returns whether background music is enabled
+ <u>void setSFXEnabled(boolean enabled)</u>	Enables or disables sound effects
+ <u>boolean isSFXEnabled()</u>	Returns whether sound effects are enabled

#### 4.10 Class UIManager

This class is responsible for rendering UI elements such as the player's score, power-ups, and the range score bar, ensuring the user interface is updated dynamically based on the game's progress.

##### Fields

Name	Description
- <u>GraphicsContext gc</u>	Graphics context for drawing UI elements
- <u>Map&lt;PowerUp.PowerUpType, Image&gt; powerUpIcons</u>	Stores icons for different power-ups
- <u>Map&lt;PowerUp.PowerUpType, Long&gt; activePowerUps</u>	Tracks active power-ups and their expiration times

##### Methods

Name	Description
+ <u>void</u> <u>setGraphicsContext(GraphicsContext</u> <u>newGc)</u>	Sets the graphics context for UI rendering
+ <u>void</u> <u>renderUI()</u>	Calls methods to draw UI elements
+ <u>void</u> <u>drawScore()</u>	Displays the current score on screen
+ <u>void</u> <u>drawRangeScoreBar()</u>	Draws the score progress bar
+ <u>void</u> <u>drawActivePowerUps()</u>	Displays active power-ups with their remaining time
+ <u>void</u> <u>activatePowerUpUI(PowerUp.PowerUpType</u> <u>type)</u>	Adds a power-up to the UI and removes it after expiration
+ <u>void</u> <u>removeExpiredPowerUps()</u>	Clears expired power-ups from the UI

## 5. Package gui

### 5.1 Class CreditScreenController

The CreditScreenController class manages the credit screen, displaying background images and handling user navigation. This class ensures that users can easily return to the main menu from the credit screen.

#### Fields

Name	Description
- ImageView bgImage	The background image for the credit screen
- Button btnHome	A button that allows the user to return to the home screen
- ImageView homeIcon	An image for the home button

#### Methods



Name	Description
+ void initialize()	Loads images for the background and home button, and sets the action for the home button

## 5.2 Class GameOverScreenController

The GameOverScreenController class manages the game over screen, allowing the player to either restart the level or return to the home screen. This class ensures a smooth transition when the player loses the game, giving them options to restart or go back to the home screen.

### Fields

Name	Description
- ImageView bgGameOver	The background image for the game over screen
- Button btnPlayAgain	A button that restarts the current level
- Button btnMenu	A button that returns the player to the main menu

### Methods

Name	Description
+ void initialize()	Loads the game over background image and sets actions for the Play Again and Menu buttons

## 5.3 Class HomeScreenController

The HomeScreenController class manages the home screen UI, allowing the player to navigate between different game menus. This class ensures that the player can access different sections of the game from the home screen.

### Fields

Name	Description
- ImageView bgImage	The background image for the home screen
- VBox buttonContainer	A container for all the menu buttons
- Button btnPlayGame	A button that starts the level selection
- Button btnSettings	A button that opens the settings menu
- Button btnHowToPlay	A button that displays gameplay instructions
- Button btnCredit	A button that shows the game credits
- Button btnExit	A button that exits the game

#### *Methods*

Name	Description
+ void initialize()	Loads the background, sets button images, and assigns actions to buttons
- void setButtonImage(Button button, String imagePath)	Assigns an image to a button

### **5.4 Class HowToPlayScreen1Controller**

The HowToPlayScreen1Controller class manages the first How to Play screen, where players can learn about the game. This class ensures that players can navigate between tutorial pages and return to the home screen.

#### *Fields*

Name	Description
- ImageView bgImage	The background image for the How to Play screen

- Button btnHome	A button to go back to the home screen
- Button btnNext	A button to navigate to the next How to Play page
- ImageView homelcon	An image icon for the home button
- ImageView nextIcon	An image icon for the back button

#### Methods

Name	Description
+ void initialize()	Loads images and assigns actions to the buttons

### 5.5 Class HowToPlayScreen2Controller

The HowToPlayScreen2Controller class manages the second How to Play screen, where players can continue learning about the game. This class allows players to navigate between tutorial pages and return to the home screen.

#### Fields

Name	Description
- ImageView bgImage	The background image for the How to Play screen
- Button btnHome	A button to return to the home screen
- Button btnBack	A button to go back to the first How to Play page
- ImageView homelcon	An image icon for the home button
- ImageView backIcon	An image icon for the back button

#### Methods

Name	Description
------	-------------

+ void initialize()	Loads images and assigns actions to the buttons
---------------------	---

### 5.6 Class Level1ScreenController

The Level1ScreenController class manages the UI for the Level 1 game screen. This class is responsible for handling UI interactions and ensuring that Level 1 runs correctly.

#### Fields

Name	Description
- Canvas gameCanvas	The canvas where the game is drawn
- Button btnPlayAgain	A button to restart the current level
- Button btnHome	A button to return to the home screen

#### Methods

Name	Description
+ void initialize()	Loads the graphics context and assigns button actions
+ Canvas getCanvas()	Returns the game canvas

### 5.7 Class Level2ScreenController

The Level2ScreenController class manages the UI for the Level 2 game screen. This class is responsible for handling UI interactions and ensuring that Level 2 runs correctly.

#### Fields

Name	Description
- Canvas gameCanvas	The canvas where the game is drawn
- Button btnPlayAgain	A button to restart the current level
- Button btnHome	A button to return to the home screen

### Methods

Name	Description
+ void initialize()	Loads the graphics context and assigns button actions
+ Canvas getCanvas()	Returns the game canvas

## 5.8 Class Level3ScreenController

The Level3ScreenController class manages the UI for the Level 3 game screen. This class is responsible for handling UI interactions and ensuring that Level 3 runs correctly.

### Fields

Name	Description
- Canvas gameCanvas	The canvas where the game is drawn
- Button btnPlayAgain	A button to restart the current level
- Button btnHome	A button to return to the home screen

### Methods

Name	Description
+ void initialize()	Loads the graphics context and assigns button actions
+ Canvas getCanvas()	Returns the game canvas

## 5.9 Class LevelSelectionScreenController

The LevelSelectionScreenController class manages the UI for the level selection screen. This class ensures that players can navigate the level selection screen and start a game only if the level is unlocked.

### Fields

Name	Description
------	-------------

- Button btnLevel1, btnLevel2, btnLevel3	Buttons to select different levels
- Button btnCheatUnlock	A button to unlock all levels for testing
- Button btnBack	A button to return to the home screen
- ImageView iconLevel1, iconLevel2, iconLevel3	Icons indicating the level's lock status
- ImageView bgImage	Background image of the level selection screen
- ImageView homeIcon	Home button icon
- int selectedLevel	Stores the currently selected level

### Methods

Name	Description
+ void initialize()	Loads images, sets up button actions, and updates level buttons
- void handleLevelSelection(int level)	Handles level selection logic and starts the game if a selected level is clicked again
- void updateLevelButtons()	Updates the level buttons based on the unlocked levels
- void updateLevelIcon(ImageView icon, int level, int unlockedLevel)	Updates each level's icon based on its unlock status

## 5.10 Class SceneManager

The SceneManager class handles scene transitions in the game. This class ensures smooth scene transitions and automatically assigns click sounds to buttons.

### Fields

Name	Description
- <u>Stage primaryStage</u>	The primary application window where scenes are displayed

### Methods

Name	Description
+ <u>void setPrimaryStage(Stage stage)</u>	Sets the primary stage for scene switching
+ <u>Stage getPrimaryStage()</u>	Returns the current primary stage

## 5.11 Class SettingScreenController

The SettingScreenController class manages the game's settings screen. This class allows players to toggle background music and sound effects while ensuring the UI updates accordingly.

### Fields

Name	Description
- ImageView bgImage	Displays the background image of the settings screen
- Button btnToggleMusic	Button to enable or disable background music
- Button btnToggleSound	Button to enable or disable sound effects
- Button btnBack	Button to return to the home screen
- boolean isMusicOn	Stores the background music state (on/off)
- boolean isSoundOn	Stores the sound effects state (on/off)

### Methods

Name	Description
+ void initialize()	Loads the background image and sets up button actions
- void updateMusicButton()	Updates the UI of the music toggle button
- void updateSoundButton()	Updates the UI of the sound effects toggle button

### 5.12 Class WinScreenController

The WinScreenController class handles the win screen that appears when the player successfully completes a level. This class provides an interactive win screen, allowing players to restart their level or return to the main menu.

### Fields

Name	Description
- ImageView winImage	Displays the "YOU WIN!!" image
- Button btnPlayAgain	Button that allows the player to restart the same level
- Button btnHome	Button that takes the player back to the home screen

### Methods

Name	Description
+ void initialize()	Loads the win image and sets up button actions
- void loadWinImage()	Loads the "YOU WIN!!" image and handles any errors if the file is missing



