Tha_akatsayan_suwannaphum Documentation

Created by

Jirapat Chaiya 6732007621

Pitiwat Kittiwimonchai 6732023621

2110215 Programming Methodology

Semester 2 Year 2024

Chulalongkorn University

LandWorm

Introduction

LandWorm was inspired by the game paper.io, which is a real-time territory control game. In LandWorm, players control a character that moves outside their area to draw a line and form a new boundary. When they manage to return to their base successfully, the enclosed area becomes theirs.

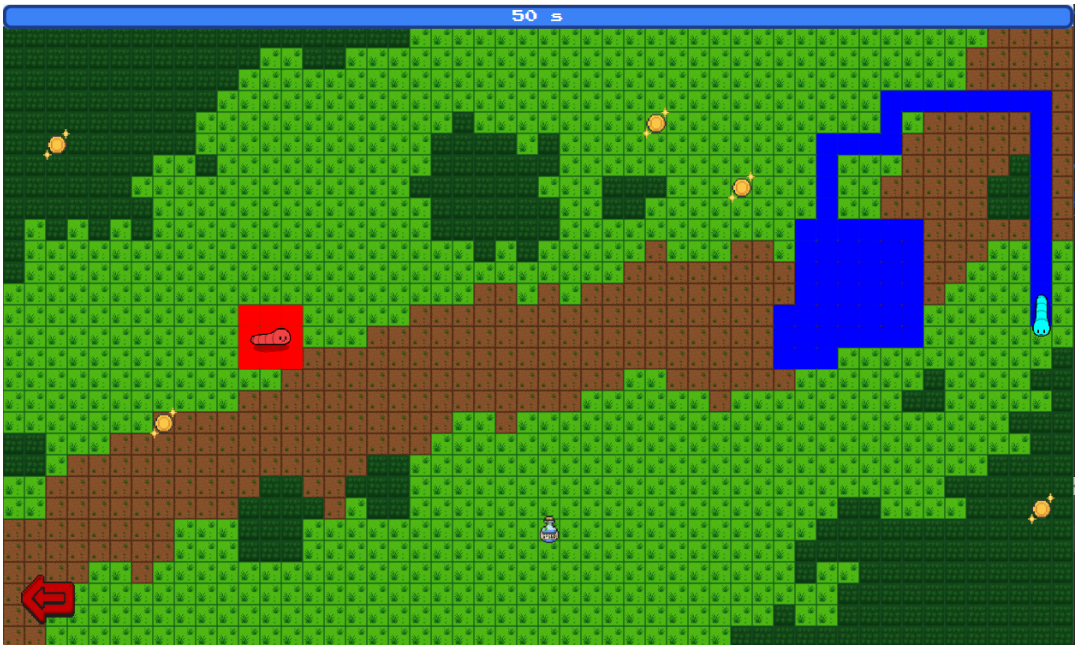
Rules

There are two players, each with a different color. Each player starts with a safe zone at their initial position. They can move outside their territory, leaving a trail in their color. If the other player steps on their trail—or if they accidentally step on it themselves—they die and respawn at their starting area. However, if they manage to return to their safe zone without being interrupted, the area enclosed by their trail becomes part of their territory. Territory can always be invaded and taken over. When time runs out, the winner is the player who controls the largest area.

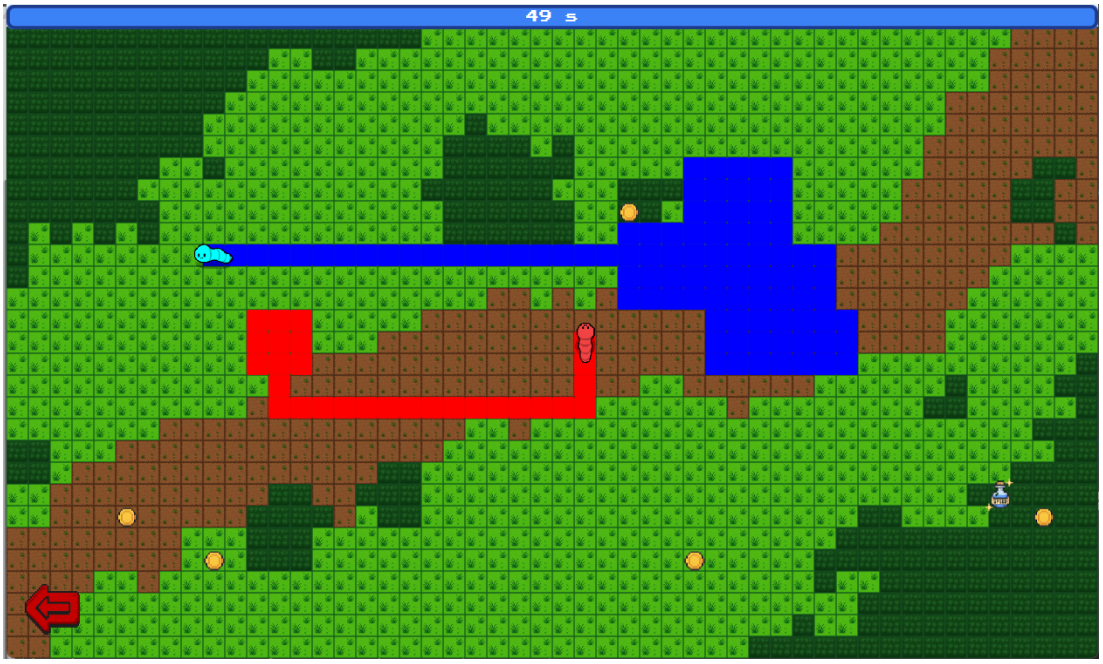
Controls

The player on the left uses the W, A, S, D keys, and the player on the right uses the arrow keys.

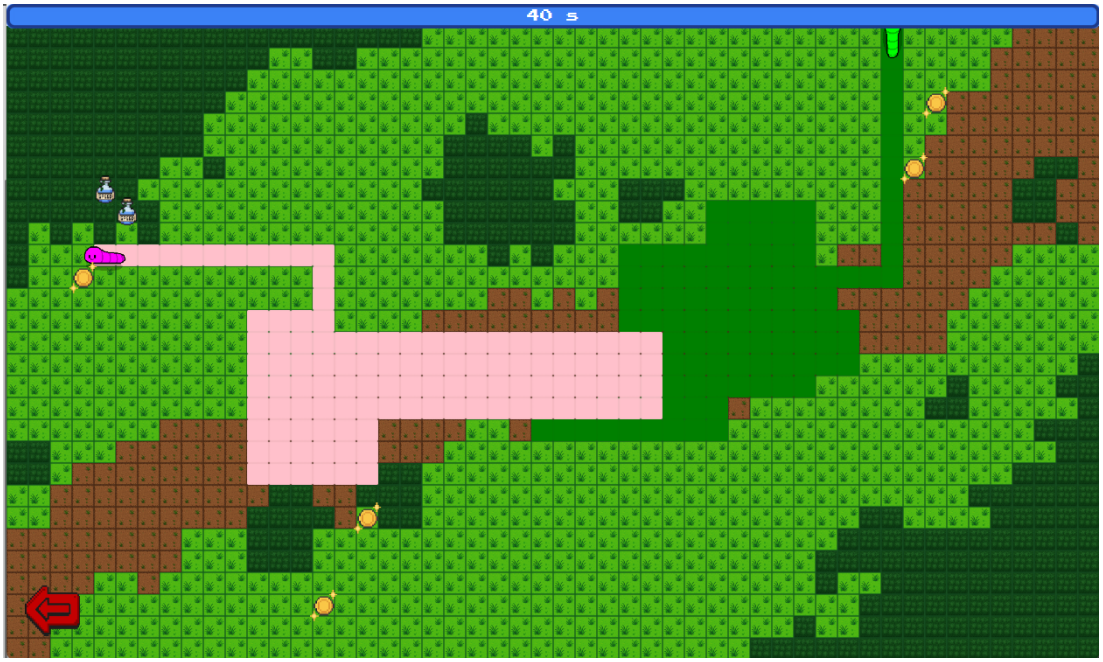
Example



You have your own starting area. You go out, leave a trail, and if you make it back, your territory grows.

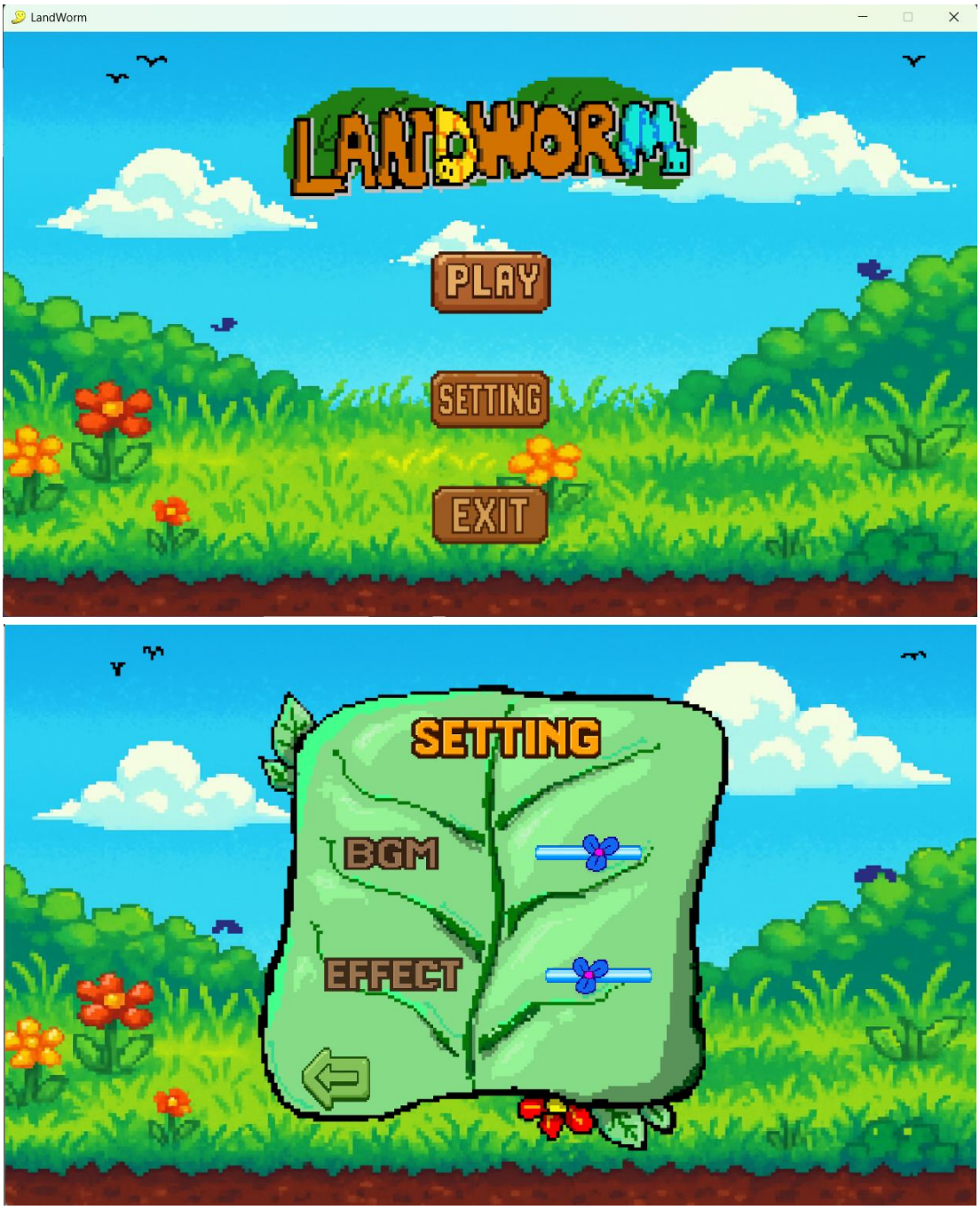


As shown in the image, the blue player will die and return to the point where they left their area. They won’t gain the blue trail as part of their territory.



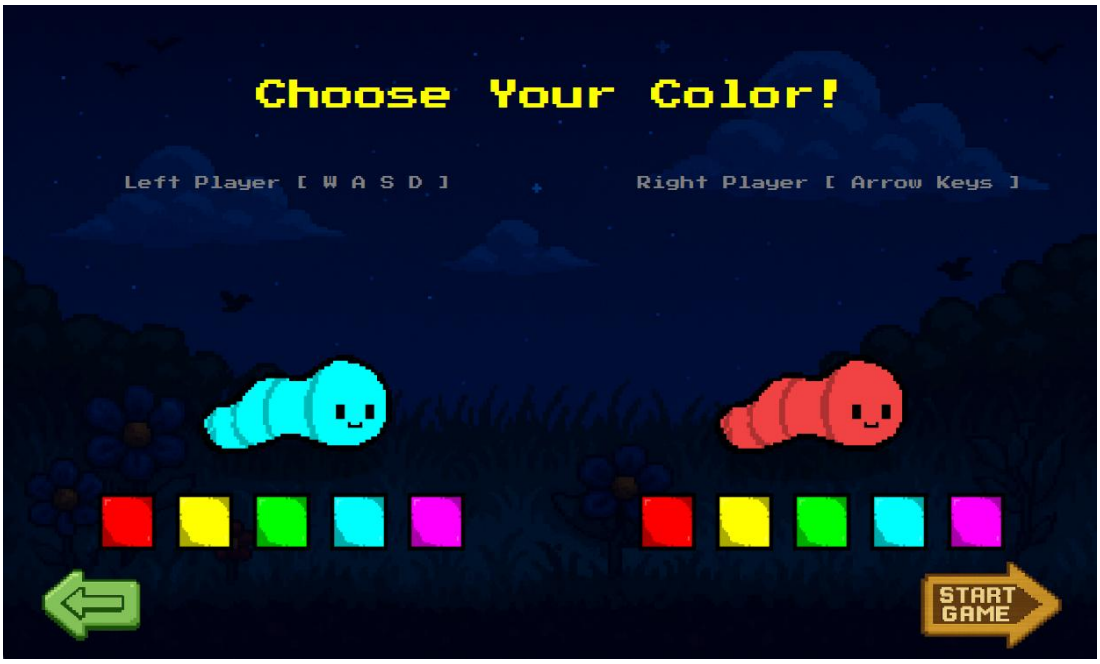
You can invade and take over the other player’s territory if you safely return to your own area. You can also collect coins, which act as a bonus, giving you the equivalent of 10 free tiles. This can help determine the winner at the end of the game. In addition, there are potions that temporarily increase the speed of the player who picks them up.

MainMenuScene



Press the PLAY button to go to the color selection screen (PlayerSettingScene).
Press the SETTING button to adjust the BGM and effect volume, as shown in the image below.
Press the EXIT button to quit the game.

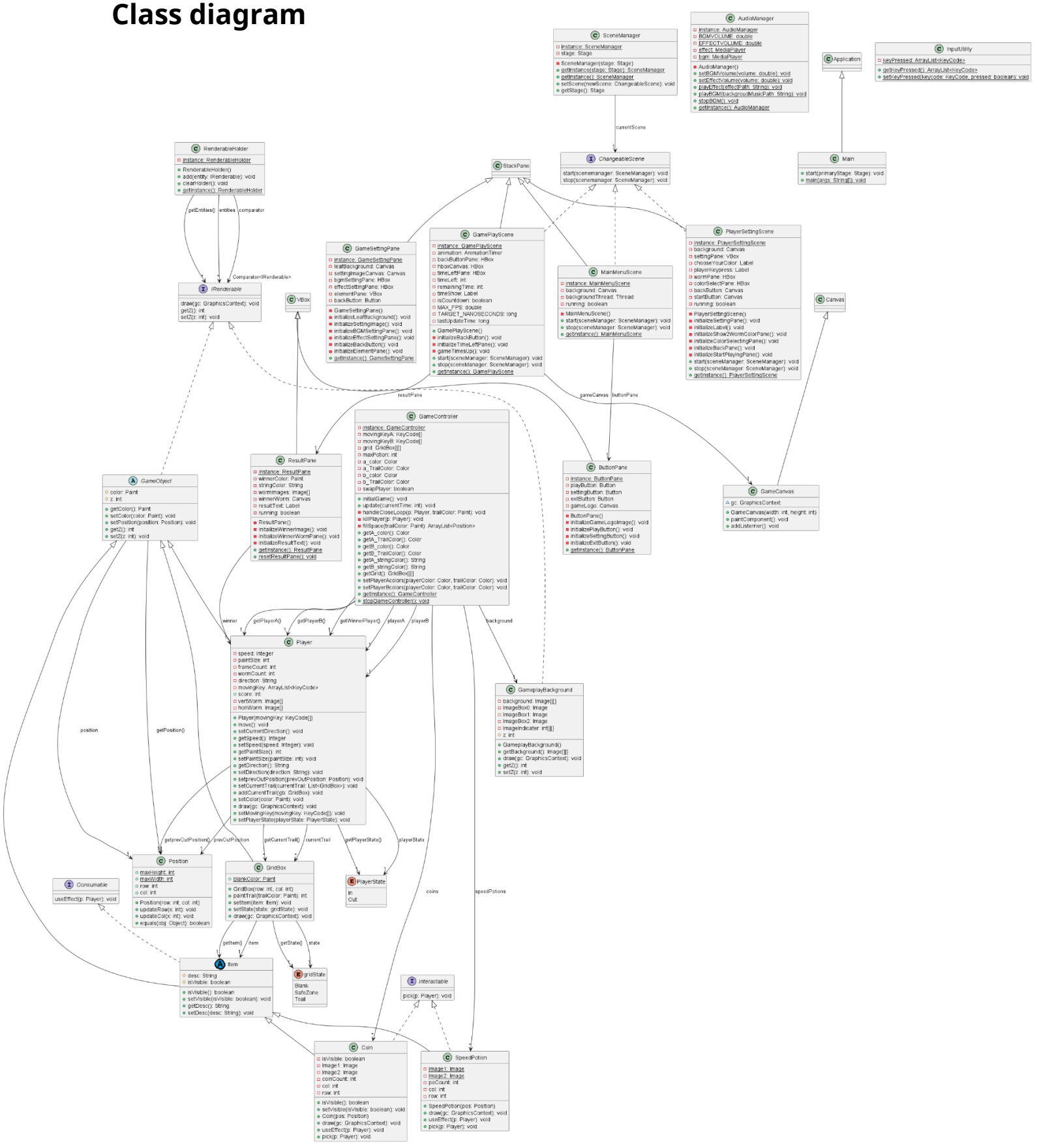
PlayerSettingScene



A short tutorial is provided to explain how to play. Each player chooses a color to play, and both players can not select the same color. After selecting, press the button to start the game.

When time runs out, the winner will be shown.

Class diagram



Project Structure

src/
├─── application/
├─── game/
│ ├─── controller/
│ ├─── object/
│ │ ├─── interfaces/
│ │ └─── Items/
├─── gui/
│ └─── scene/
├─── input/
├─── Manager/
├─── map/
└─── sharedObject/

Package game

1 Package controller

1.1 class GameCanvas extends Canvas

1.1.1 Fields

- GraphicsContext gc	- Graphics context for drawing on the canvas.

1.1.2 Constructor

+ GameCanvas(int width, int height)	- Creates a GameCanvas with the specified width and height. - Set the canvas to be visible.

1.1.3 Methods

+ paintComponent()	- Clears the canvas. - Iterates through a list of IRenderable entities and calls their draw method using the graphics context.
+ addListerner()	- Sets up key press and key release event listeners. - Updates an InputUtility class to track which keys are currently pressed.

1.2 class GameController

1.2.1 Fields

- static GameController instance	- Singleton instance of the GameController
- KeyCode[] movingKeyA	- Array of KeyCode for Player A's movement
- KeyCode[] movingKeyB	- Array of KeyCode for Player B's movement
- Player playerA	- Player A object.
- Player playerB	- Player B object.
- ArrayList<SpeedPotion> speedPotions	- List of SpeedPotion objects.
- GridBox[][] grid	- 2D array representing the game grid.
- int maxPotion	- Maximum number of potions.
- Color a_color:	- Color for Player A.
- Color a_TrailColor	- Trail color for Player A.
- Color b_color:	- Color for Player B.
- Color b_TrailColor	- Trail color for Player B.

- GameplayBackground background	- Background object for the game.

1.2.2 Methods

+ void initialGame()	- Initializes the game state (players, grid, safe zones, background, speed potions).
+ void update(int currentTime)	- Updates the game state (player movement, trail painting, potion picking, and safe zone filling).
- void handleCloseLoop(Player p, Paint trailColor)	- Handles the logic when a player closes a loop, claiming territory.
- killPlayer(Player p)	- Handles the logic when a player is killed, resetting their position and clearing their trail.
- ArrayList<Position> fillSpace(Paint trailColor)	- Fills space enclosed by trail; returns list of positions.
+ Player getPlayerA()	- Returns Player A.
+ Player getPlayerB()	- Returns Player B.
+ Color getA_color()	- Returns Player A's color.
+ Color getA_TrailColor()	- Returns Player A's trail color.
+ Color getB_color()	- Returns Player B's color.
+ Color getB_TrailColor()	- Returns Player B's trail color.
+ String getA_stringColor()	- Returns Player A's color as a string.
+ String getB_stringColor()	- Returns Player B's color as a string.
+ GridBox[][] getGrid()	- Returns the game grid.
+ void setPlayerAcolors(Color playerColor, Color trailColor)	- Sets Player A's colors.

+ void setPlayerBcolors(Color playerColor, Color trailColor)	- Sets Player B's colors.
+ static GameController getInstance()	- Returns the singleton instance of GameController. Creates the instance if it doesn't exist.
+ Player getWinnerPlayer()	- Determines who is the winner and returns Player’s object.
+ static void stopGameController()	- Resets the instance to null.

2 Package object

(this package have class (.java file) and package inside)

2.1 Class Position

2.1.1 Fields

+ static int maxHeight	- Maximum row value (29).
+ static int maxWidth	- Maximum column value (49).
+ int row	- Row coordinate.
+ int col	- Column coordinate.

2.1.2 Constructor

+ Position(int row, int col)	- Creates a Position object with the given row and column.

2.1.3 Methods

+ void updateRow(int x)	- Updates the row coordinate, clamping it within the bounds.
+ void updateCol(int x)	- Updates the column coordinate, clamping it within the bounds.
+ boolean equals(Object obj)	- Checks if two Position objects are equal based on their row and column.

2.2 Enum PlayerState

2.2.1 Values

In	- Represents the player being in a safe zone.
Out	- Represents the player being outside a safe zone.

2.3 Class GridBox extends GameObject

2.3.1 Fields

- Item item	- Item contained in the grid box.
- gridState state	- Current state of the grid box (Blank, SafeZone, Trail).
+ static Color blankColor	- Default color for a blank grid box (BISQUE).

2.3.2 Constructor

+ GridBox(int row, int col)	- Creates a GridBox object with the given row and column, setting its color to blankColor.

2.3.3 Methods

+ int paintTrail(Paint trailColor)	- Paints the grid box with the given trail color, updating its state and returning a value indicating the result.
+ Item getItem()	- Returns the item contained in the grid box.
+ void setItem(Item item)	- Sets the item contained in the grid box.
+ gridState getState()	- Returns the current state of the grid box.
+ void setState(gridState state)	- Sets the current state of the grid

	box.
+ void draw(GraphicsContext gc)	- Draw the grid box on the canvas if its color is not BISQUE.

2.4 Abstract Class Item extends GameObject implements Consumable

2.4.1 Fields

- String desc	- Description of the item.
- boolean isVisible	- Visibility status of the item.

2.4.2 Methods

+ boolean isVisible()	- Returns the visibility status of the item.
+ void setVisible(boolean isVisible)	- Sets the visibility status of the item.
+ String getDesc()	- Returns the description of the item.
+ void setDesc(String desc)	- Sets the description of the item.

2.5 Abstract Class GameObject implements IRenderable

2.5.1 Fields

- Position position	- Position of the game object.
- Paint color	- Color of the game object.
- int z	- Z-order of the game object for rendering.

2.5.2 Methods

+ Paint getColor()	- Returns the color of the game object.
+ void setColor(Paint color)	- Sets the color of the game object.

+ Position getPosition()	- Returns the position of the game object.
+ void setPosition(Position position)	- Sets the position of the game object.
+ int getZ()	- Returns the Z-order of the game object.
+ void setZ(int z)	- Sets the Z-order of the game object.

2.6 Class Player extends GameObject

2.6.1 Fields

- Integer speed	- Speed of the player.
- int paintSize	- Size of the trail painted by the player.
- int frameCount	- Frame counter for movement.
- int wormCount	- Frame counter for animation.
- PlayerState playerState	- Current state of the player (In, Out).
- String direction	- Current direction of the player (N, UP, LEFT, DOWN, RIGHT).
- Position prevOutPosition	- Previous position of the player when outside a safe zone.
- List<GridBox> currentTrail	- List of grid boxes representing the player's current trail.
- ArrayList<KeyCode> movingKey	- List of KeyCode for player's movement
+ int score	- Player’s score

2.6.2 Constructor

+ Player(KeyCode[] movingKey)	- Creates a Player object,

	initializing its speed, paint size, direction, trail, and moving keys.

2.6.3 Methods

+ void move()	- Moves the player based on its current direction.
+ void setCurrentDirection()	- Sets the player's current direction based on input.
+ Integer getSpeed()	- Returns the speed of the player.
+ void setSpeed(Integer speed)	- Sets the speed of the player.
+ int getPaintSize()	- Returns the paint size of the player.
+ void setPaintSize(int paintSize)	- Sets the paint size of the player.
+ String getDirection()	- Returns the direction of the player.
+ void setDirection(String direction)	- Sets the direction of the player.
+ Position getprevOutPosition()	- Returns the previous out position of the player.
+ void setprevOutPosition(Position prevOutPosition)	- Sets the previous out position of the player.
+ List<GridBox> getCurrentTrail()	- Returns the current trail of the player.
+ void setCurrentTrail(List<GridBox> currentTrail)	- Sets the current trail of the player.
+ void addCurrentTrail(GridBox gb)	- Adds a grid box to the player's current trail if it's not already in the trail and is not a safe zone.
+ void setColor(Paint color)	- set the color of player, initialize the player's worm Image array base on color

+ void draw(GraphicsContext gc)	- Draws the player on the canvas.
+ void setMovingKey(KeyCode[] movingKey)	- Sets the moving keys for the player
+ PlayerState getPlayerState()	- Gets the player state of the player
+ void setPlayerState(PlayerState playerState)	- Sets the player state of the player

2.7 Enum gridState

2.7.1 Values

Blank	- Represents an empty grid box.
SafeZone	- Represents a safe zone grid box.
Trail	- Represents a grid box with a trail.

3 Package interfaces

3.1 Interface Consumable

3.1.1 Methods

+ void useEffect(Player p)	- Defines the effect of consuming the object on a player.

3.2 Interface Interactable

3.2.1 Methods

+ void pick(Player p)	- Defines the action of picking up the object, affecting the player.

4 Package Items

4.1 Class SpeedPotion extends Item implements Interactable

4.1.1 Fields

- Image Image1	- An Image object representing one appearance of the speed potion (loaded from "Image/potion_1.png").
- Image Image2	- An Image object representing another appearance of the speed potion (loaded from "Image/potion_2.png").
- int poCount	- An integer used to animate the potion's appearance.
- int col	- The column position of the potion.
- int row	- The row position of the potion.

4.1.2 Constructor

+ SpeedPotion(Position pos)	- Constructor that sets the potion's position.

4.1.3 Methods

+ void draw(GraphicsContext gc)	- Draws the speed potion on the given GraphicsContext, animating its appearance using Image1 and Image2.
+ void useEffect(Player p)	- Applies the speed potion effect to the player, increasing their speed temporarily.
+ void pick(Player p)	- Handles the player picking up the speed potion, applying its effect and making it invisible.

4.2 Class Coin extends Item implements Interactable

4.2.1 Fields

- boolean isVisible	- A boolean indicating whether the coin is visible.

- Image Image1	- An Image object representing one appearance of the speed potion (loaded from "Image/coin1.png").
- Image Image2	- An Image object representing one appearance of the speed potion (loaded from "Image/coin2.png").
- int coinCount	- An integer used to animate the coin's appearance.
- int col	- The column position of the potion.
- int row	- The row position of the potion.

4.2.2 Constructor

+ Coin(Position pos)	- Constructor that sets the coin's position.

4.2.3 Methods

+ boolean isVisible()	- Returns whether the coin is visible.
+ void setVisible(boolean isVisible)	- Sets whether the coin is visible.
+ void draw(GraphicsContext gc)	- Draws the coin on the given GraphicsContext (currently empty).
+ void useEffect(Player p)	- Applies the coin effect to the player, increasing their score.
+ void pick(Player p)	- Handles the player picking up the coin, applying its effect and making it invisible.

5 Package gui

5.1 Class ResultPane extends VBox

5.1.1 Fields

- static ResultPane instance	- Singleton instance of the ResultPane class.
- Color winnerColor	- Color of the winning player
- String stringColor	- String representation of the winning player's color.
- Image[] wormImages	- Array of images representing the winning worm's animation frames.
- Canvas winnerWorm	- Canvas for displaying the winning worm animation.
- Label resultText	- Label displaying the result text (e.g., "The Winner is Red Player!!").
- boolean running	- Flag to control the worm animation thread.

5.1.2 Methods

+ static ResultPane getInstance()	- Returns the singleton instance of the ResultPane.
- ResultPane()	- Constructor for the ResultPane class. Initializes the UI elements.
- void initailizeWinnerImage()	- Loads the appropriate worm images based on the winner's color.

- void initializeWinnerWormPane()	- Creates and starts the animation thread for the winning worm.
- void initializeResultText()	- Creates the result text label.
+ void resetResultPane()	- Set instance to null.

5.2 Class GameSettingPane extends StackPane

5.2.1 Field

- static GameSettingPane instance	- Singleton instance of the GameSettingPane class.
- Canvas leafBackground	- Canvas for displaying the background image.
- Canvas settingImageCanvas	- Canvas for displaying the "Settings" title image.
- HBox bgmSettingPane	- HBox containing the BGM volume control elements (label and slider).
- HBox effectSettingPane	- HBox containing the sound effects volume control elements (label and slider).
- VBox elementPane	- VBox for organizing the setting elements vertically.
- Button backButton	- Button to return to the main menu.

5.2.2 Constructor

+ GameSettingPane()	- Constructor for the GameSettingPane class. Initializes the UI elements.

5.2.3 Methods

+ static GameSettingPane getInstance()	- Returns the singleton instance of the GameSettingPane.

- void initializeLeafBackground()	- Loads and draws the background image on the canvas.
- void initializeSettingImage()	- Loads and draws the "Settings" title image on the canvas.
- void initializeBGMSettingPane()	- Creates the BGM volume control elements.
- void initializeEffectSettingPane()	- Creates the sound effects volume control elements.
- void initializeBackButton()	- Creates and styles the back button.
- void initializeElementPane()	- Arranges the setting elements in a vertical layout.

5.3 Class ButtonPane extends VBox

5.3.1 Fields

- static ButtonPane instance	- Singleton instance of the ButtonPane class.
- Button playButton	- Button to navigate to the player setting scene.
- Button settingButton	- Button to navigate to the game setting pane.
- Button exitButton	- Button to exit the application.
- Canvas gameLogo	- Canvas for displaying the game logo image.

5.3.2 Constructor

+ ButtonPane()	- Constructor for the ButtonPane class. Initializes the UI elements.

5.3.3 Methods

+ static ButtonPane getInstance()	- Returns the singleton instance of

	the ButtonPane.
- void initializeGameLogoImage()	- Loads and draws the game logo image on the canvas.
- void initializePlayButton()	- Creates and styles the play button.
- void initializeSettingButton()	- Creates and styles the setting button.
- void initializeExitButton()	- Creates and styles the exit button.

6 Package scene

6.1 Interface ChangeableScene

6.1.1 Methods

+ void start(SceneManager scenemanager)	- Method to start the scene, taking a SceneManager as an argument.
+ void stop(SceneManager scenemanager)	- Method to stop the scene, taking a SceneManager as an argument.

6.2 Class GamePlayScene extends StackPane implements ChangeableScene

6.2.1 Fields

- static GamePlayScene instance	- Singleton instance of the GamePlayScene class.
- AnimationTimer animation	- The animation timer for the gameplay loop.
- HBox backButtonPane	- HBox containing the back button.
- GameCanvas gameCanvas	- Custom canvas for rendering the

	game.
- HBox hboxCanvas	- HBox to hold and pad the game canvas.
- HBox timeLeftPane	- HBox to display the time left.
- int timeLeft	- Initial time left for the game (in seconds).
- int remainingTime	- The remaining time left in the game.
- Label timeShow	- Label to display the remaining time.
- ResultPane resultPane	- Instance of the ResultPane to display the game result.
- boolean isCountdown	- Flag to check if the countdown sound is playing.
- long MAX_FPS	- Limit for the game, set to 120 to control the update rate.
- TARGET_NANOSECONDS	- Target time interval in nanoseconds for each frame
-lastUpdateTime	- Stores the timestamp of the last update in nanoseconds

6.2.2 Constructor

+ GamePlayScene()	- Constructor for the GamePlayScene class. Initializes the UI elements and starts the game loop.

6.2.3 Methods

+ static GamePlayScene getInstance()	- Returns the singleton instance of the GamePlayScene.
+ void start(SceneManager sceneManager)	- Starts the scene, sets up the scene in the stage, and starts the BGM.
+ void stop(SceneManager sceneManager)	- Stops the scene, animation, BGM, clears resources and resets the instance.
- void initializeBackButton()	- Initializes the back button.
- void initializeTimeLeftPane()	- Initializes the time left display.
- void gameTimesUp()	- Method to handle the game over event.

6.3 Class MainMenuScene extends StackPane implements ChangeableScene

6.3.1 Field

- static MainMenuScene instance	- Singleton instance of the MainMenuScene class.
- ButtonPane buttonPane	- Instance of the ButtonPane to display the main menu buttons.
- Canvas background	- Canvas for displaying the animated background.
- Thread backgroundThread	- Thread for animating the background.
- boolean running	- Flag to control the background animation thread.

6.3.2 Constructor

+ MainMenuScene()	- Constructor for the MainMenuScene class. Initializes

	the UI elements and starts the background animation.

6.3.3 Methods

+ static MainMenuScene getInstance()	- Returns the singleton instance of the MainMenuScene.
+ void start(SceneManager sceneManager)	- Starts the scene, sets up the scene in the stage, and starts the BGM.
+ void stop(SceneManager sceneManager)	- Stops the scene, animation, BGM, and resets the instance.
+ void start(SceneManager sceneManager)	- Start the scene.
+ void stop(SceneManager sceneManager)	- Stop the scene.

6.4 Class PlayerSettingScene extends StackPane implements ChangeableScene

6.4.1 Field

- static PlayerSettingScene instance	- Singleton instance of the PlayerSettingScene class.
- Canvas background	- Canvas for displaying the background image.
- VBox settingPane	- VBox containing all the setting UI elements.
- Label chooseYourColor	- Label displaying "Choose Your Color!".
- Label playerKeypress	- Label displaying player keybindings.
- HBox wormPane	- HBox to display the worm color previews.

- HBox colorSelectPane	- HBox containing the color selection buttons.
- Canvas backButton	- Button to return to the main menu.
- Canvas startButton	- Button to start the game.
- boolean running	- Flag to control the worm animation thread.

6.4.2 Constructor

+ PlayerSettingScene()	- Constructor for the PlayerSettingScene class. Initializes the UI elements.

6.4.3 Methods

+ static PlayerSettingScene getInstance()	- Returns the singleton instance of the PlayerSettingScene.
+ void start(SceneManager sceneManager)	- Starts the scene, sets up the scene in the stage, and starts the BGM.
+ void stop(SceneManager sceneManager)	- Stops the scene, animation, BGM, and resets the instance.
- void initializeSettingPane()	- Initializes the main setting pane.
- void initializeLabel()	- Initializes the labels.
- void initializeShow2WormColorPane()	- Initializes the worm color preview pane.
- void initializeColorSelectingPane()	- Initializes the color selection buttons.
- void initializeBackPane()	- Initializes the back button.
- void initializeStartPlayingPane()	- Initializes the start button.

+ void start(SceneManager sceneManager)	- Starts the scene.
+ void stop(SceneManager sceneManager)	- Stop the scene.

7 Package input

7.1 Class InputUtility

7.1.1 Fields

- static ArrayList<KeyCode> keyPressed	- A static ArrayList to store the KeyCode of keys that are currently pressed.

7.1.2 Method

+ static ArrayList<KeyCode> getKeyPressed()	- Returns the keyPressed ArrayList.
+ static void setKeyPressed(KeyCode keycode, boolean pressed)	- Updates the keyPressed list based on whether a key is pressed or released. If pressed is true and the key is not already in the list, it's added. If pressed is false, the key is removed from the list.

8 Package Manager

8.1 Class AudioManager

8.1.1 Fields

- static AudioManager instance	- A static instance of the AudioManager class, used for the Singleton pattern.
- static double BGMVOLUME	- A static double representing the volume level for background music (default: 0.2).

- static double EFFECTVOLUME	- A static double representing the volume level for sound effects (default: 2).
- static MediaPlayer effect	- A static MediaPlayer for playing sound effects.
- static MediaPlayer bgm	- A static MediaPlayer for playing background music.

8.1.2 Methods

+ static void setBGMVolume(double volume)	- Sets the volume for the background music.
+ static void setEffectVolume(double volume)	- Sets the volume for sound effects.
+ static void playEffect(String effectPath)	- Plays a sound effect from the specified path.
+ static void playBGM(String backgroudMusicPath)	- Plays background music from the specified path, stopping any currently playing BGM.
+ static void stopBGM()	- Stops the currently playing background music.
+ static AudioManager getInstance()	- Returns the instance of the AudioManager class (Singleton pattern).

8.2 Class AudioManager

8.2.1 Fields

- static SceneManager instance	- A static instance of the SceneManager class, used for the Singleton pattern.
- Stage stage	- The JavaFX Stage managed by the SceneManager.
- ChangeableScene currentScene	- The currently displayed scene,

	which must implement the ChangeableScene interface.

8.2.2 Methods

- SceneManager(Stage stage)	- Private constructor to prevent direct instantiation without a Stage.
+ static SceneManager getInstance(Stage stage)	- Returns the instance of the SceneManager class, initializing it with a Stage if it doesn't exist (Singleton pattern).
+ static SceneManager getInstance()	- Returns the instance of the SceneManager class, throwing an exception if it hasn't been initialized with a Stage.
+ void setScene(ChangeableScene newScene)	- Sets a new scene, stopping the current scene and starting the new scene.
+ Stage getStage()	- Returns the JavaFX Stage.

9 Package map

9.1 Class GameplayBackground

9.1.1 Fields

- Image[][] background	- A 2D array of Image objects representing the background tiles.
- Image ImageBox0	- An Image object representing a specific background tile (loaded from "Image/box0.png").
- Image ImageBox1	- An Image object representing a specific background tile (loaded from "Image/box1.JPG").

- Image ImageBox2	- An Image object representing a specific background tile (loaded from "Image/box2.JPG").
- int[][] ImageIndicater	- A 2D array of integers used to determine which image to use for each background tile.
- int z	- An integer representing the Z-order of the background.

9.1.2 Constructor

+ GameplayBackground()	- Constructor that initializes the background array based on the ImageIndicater array.

9.1.3 Methods

+ Image[][] getBackground()	- Returns the background array.
+ void draw(GraphicsContext gc)	- Draws the background tiles on the given GraphicsContext.
+ int getZ()	- Returns the Z-order of the background.
+ void setZ(int z)	- Sets the Z-order of the background.

10 Package sharedObject

10.1 Class RenderableHolder

10.1.1 Fields

- static RenderableHolder instance	- A static instance of the RenderableHolder class, used for the Singleton pattern.
- List<IRenderable> entities	- A list of IRenderable objects to be rendered.

- Comparator<IRenderable> comparator	- A comparator used to sort the entities based on their Z-order.

10.1.2 Constructor

+ RenderableHolder()	- Constructor that initializes the entities list and the comparator.

10.1.3 Methods

+ void add(IRenderable entity)	- Adds an IRenderable object to the entities list and sorts the list based on the Z-order.
+ void clearHolder()	- Clears the entities list.
+ List<IRenderable> getEntities()	- Returns the entities list.
+ static RenderableHolder getInstance()	- Returns the instance of the RenderableHolder class (Singleton pattern).

10.2 Interface IRenderable

10.2.1 Methods

+ void draw(GraphicsContext gc)	- Defines the method for drawing the object on the given GraphicsContext.
+ int getZ()	- Defines the method for getting the Z-order of the object.
+ void setZ(int z)	- Defines the method for setting the Z-order of the object.