Tower Escapism Documentation

Created by
Pumwaree Pipithsuksunt 6731338121
Thirada Thomnam 6732015621

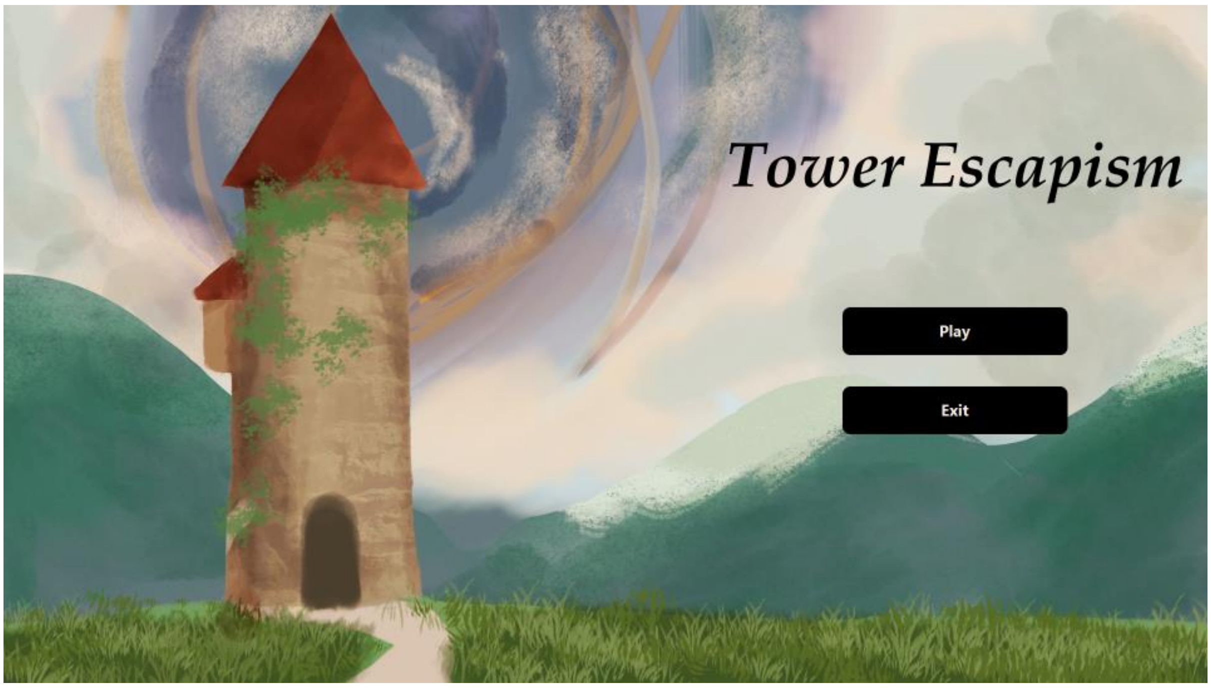
2110215 Programming Methodology
Semester 2 Year 2024
Chulalongkorn University

Introduction

This game is a puzzle-based adventure in which players must solve a series of interconnected challenges to complete the game.

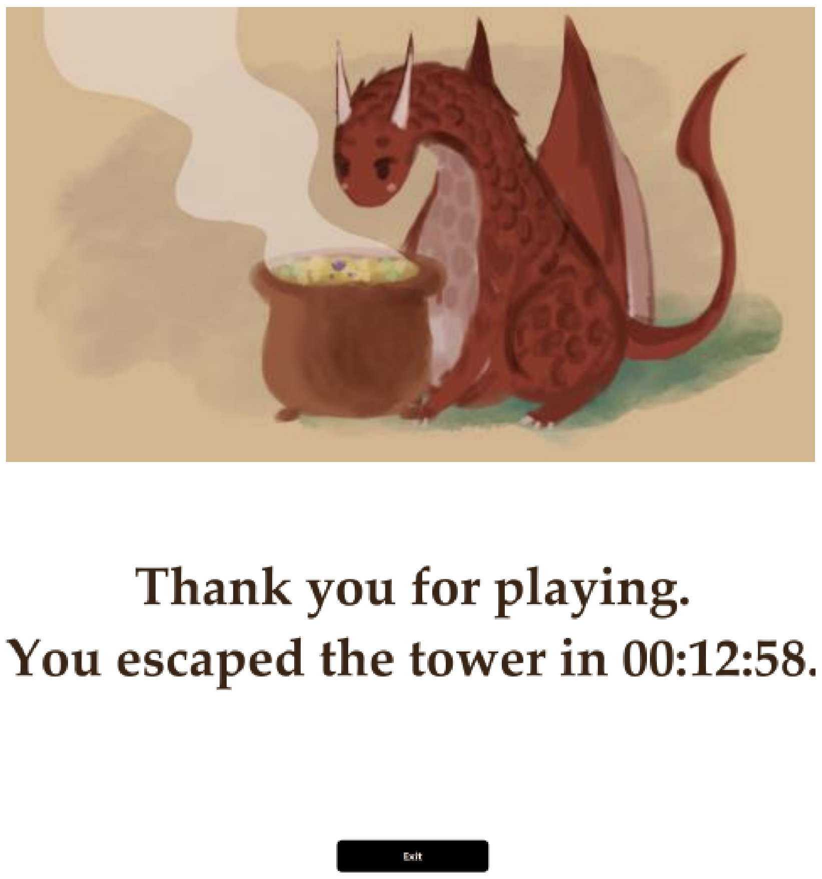
Players take on the role of an individual trapped inside a mysterious tower, seeking a way to escape. The only path to freedom lies in preparing a special dish capable of awakening a dragon. To do so, players must explore the tower and uncover hidden ingredients by solving various puzzles scattered throughout the environment.

Main Menu scene



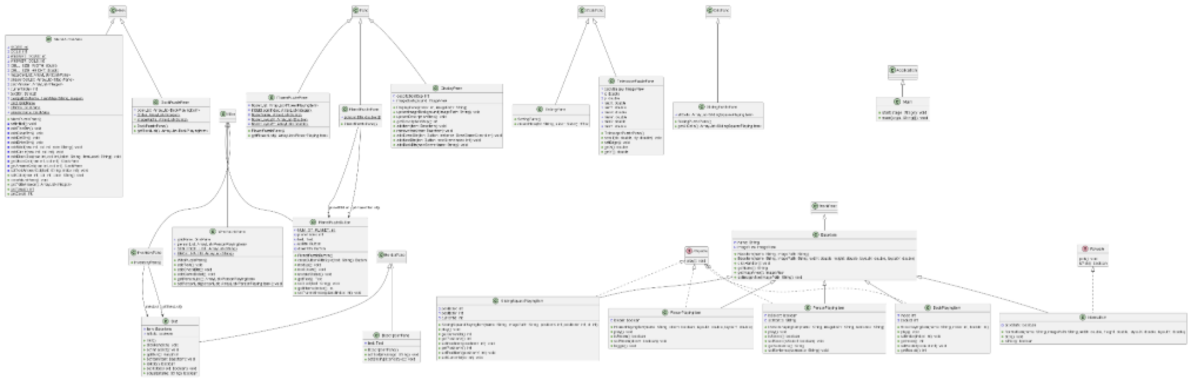
Click the play button to start the game and the exit button to exit the game.

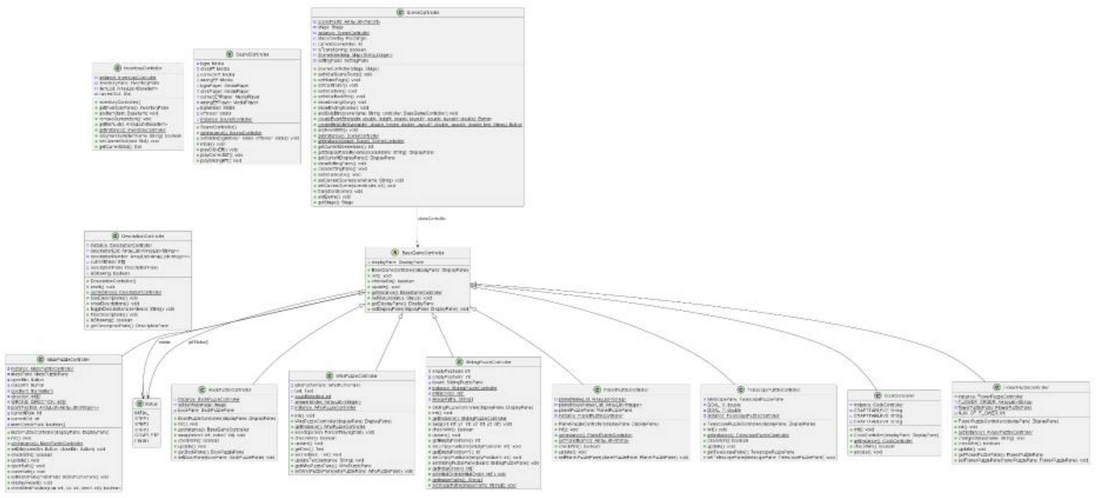
End of the game


Thank you for playing.
You escaped the tower in 00:12:58.

Collect all three essential ingredients—charred flowers, a feather, and a gold bar. Once gathered, fill the cauldron with water and add the ingredients in the correct sequence to prepare the mystical dish required to awaken the dragon.

Class Diagram





* Noted that Access Modifier Notations are listed below

+ (public)

# (protected)

- (private)

<u>Underline</u> (static)

Italic (abstract)

1. Package config

1.1 class Config

1.1.1 Fields

+ Date startTime	To display the time spent in the game by the player.
+ double screenWidth	The width of the screen
+ double screenHeight	The height of the screen
+ double imageBackgroundWidth	The width of the background
+ double imageBackgroundHeight	The height of the background
+ double gap	To ensure that the image is vertically centered on the screen.
+ double inventoryHeight	The height of the inventory’s bar
+ double inventoryWidth	The width of the inventory’s bar
+ final String MAIN_FONT	MAIN_FONT = "Palatino Linotype"
+ final String SUB_FONT	SUB_FONT = "DejaVu Serif"
+ final double HAEDING_SIZE	HAEDING_SIZE = 35
+ final double SUBHEADING_SIZE	SUBHEADING_SIZE = 24
+ final double BODY_SIZE	BODY_SIZE = 20
+ final double CAPTION_SIZE	CAPTION_SIZE = 16
+ final int NUM_OF_SCENE	NUM_OF_SCENE = 18

1.1.2 Methods

+ void setWindowSizeConfig(double width,double height)	Set the configuration values for the game's window and layout.

+ <u>void setStartTime()</u>	Capture the current time and save it to startTime using the system clock.

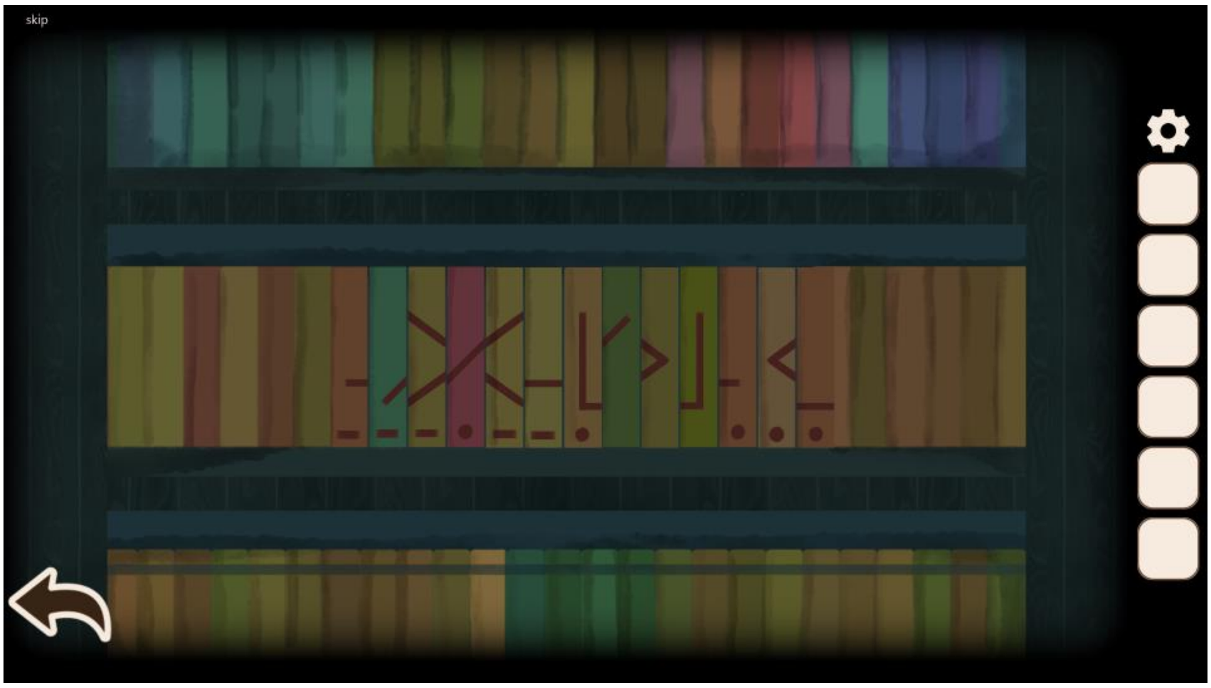
1.2 class MyColor

1.2.1 Fields

+ <u>final Color BISTRE</u>	BISTRE = Color.web("372414")
+ <u>final Color CREAM</u>	CREAM = Color.web("F7EBDF")
+ <u>final Color SAND</u>	SAND = Color.web("B7A087")
+ <u>final Color MOCHA</u>	MOCHA = Color.web("825A3C")
+ <u>final Color BROWN</u>	BROWN = Color.web("674831")

2. Package gui

2.1 class BookPuzzlePane extends HBox



2.1.1 Fields

- ArrayList<BookPlayingItem> bookList	Store books
- ArrayList<Integer> Order	Order of the books
+ ArrayList<String> imagePaths	Image path of each book

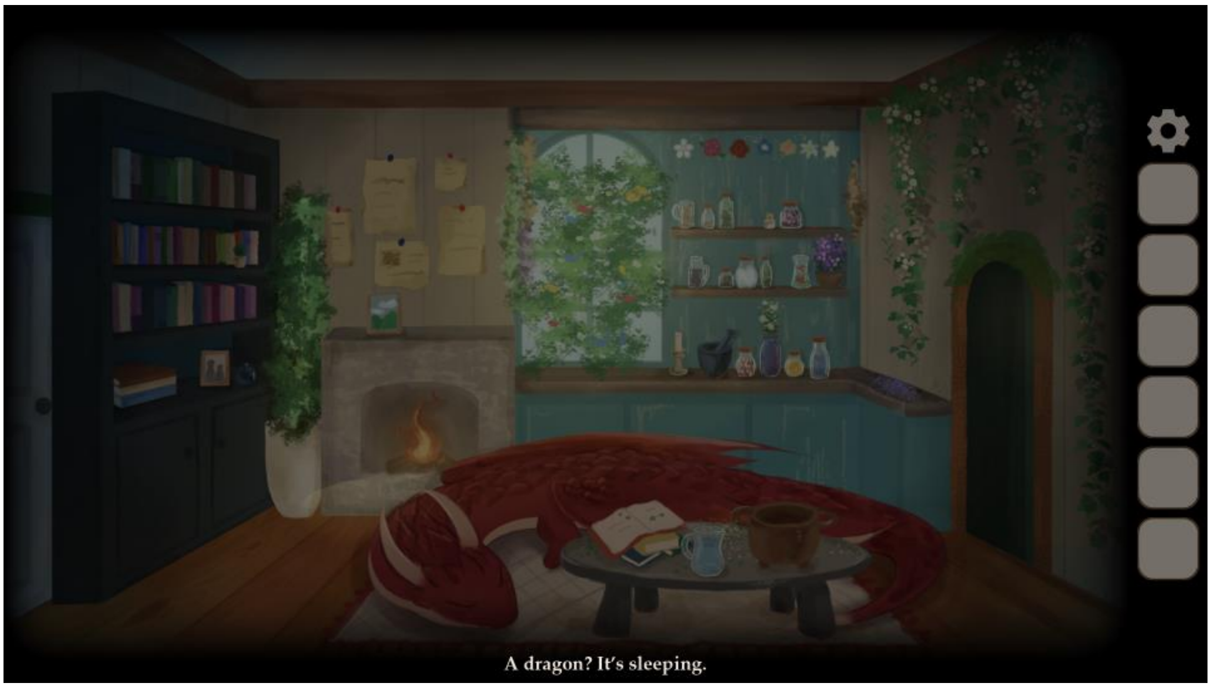
2.1.2 Constructor

+ BookPuzzlePane()	Initialize and display 13 book items in the puzzle layout at a specific screen position

2.1.3 Method

+ ArrayList<BookPlayingItem> getBookList()	Getter for bookList

2.2 class Description extends StackPane


A dragon? It's sleeping.

2.2.1 Field

- Text text	The text that is displayed as a description.

2.2.2 Constructor

+ Description()	- Set up a transparent black background to fit the screen's width and height. - Set up a stylized, centered text area at the bottom of the screen for displaying messages.

2.2.3 Method

+ void setText(String message)	Setter for text
+ void setEndingSceneStyle()	Set up a transparent background and fill text with MyColor.BISTRE.

2.3 class DisplayPane extends Pane

2.3.1 Fields

- int descriptionStep, index	
- ImageView imageBackground	Background image

2.3.2 Constructor

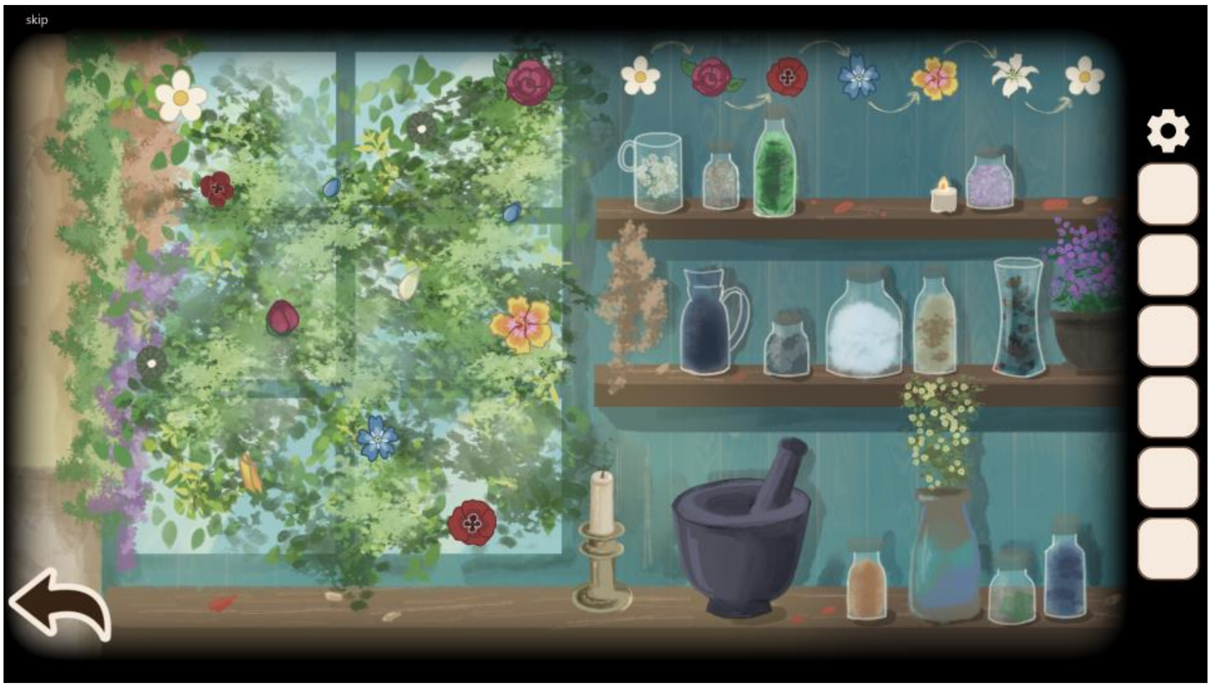
+ DisplayPane(int index, String imagePath)	- Set descriptionStep to 0. - Initialize a scene with a shadowed image background.

2.3.3 Methods

+ void updateImageBackground(String imagePath)	Change the background image during runtime.
+ void updateDescriptionStep()	Advance to the next step in a description sequence.
+ int getDescriptionStep()	Getter for descriptionStep
+ void addItem(BaseItem item)	Add a BaseItem to the pane's visual children.
+ void removeItem(BaseItem item)	Remove a previously added BaseItem from the pane.
+ void addEventBtn(Button btn, BaseGameController instance)	- Set a mouse click event that calls update() on the passed controller instance. - For WhoPuzzleController: ● Only show hand cursor if player has "candle" or

	"match" in inventory ● Otherwise show default cursor - For BookPuzzleController: ● Only show hand cursor if player has "whisk" in inventory ● Otherwise show default cursor - For all other controllers: ● Always show hand cursor on hover - Finally add the configured button to the current pane's children.
+ void addNextBtn(Button btn, int nextSceneIndex)	- Create a "Next" button that navigates to another scene when clicked. - Show hand cursor when mouse hovered; otherwise, show default cursor.
+ addBackBtn(int sceneIndex)	- Create a pre-styled "Back" button to return to a previous scene when clicked. - Show hand cursor when mouse hovered; otherwise, show default cursor.

2.4 class FlowerPuzzlePane extends Pane



2.4.1 Fields

- ArrayList<FlowerPlayingItem> flowerList	Store flowers
- ArrayList<Integer> initialBloomIndex	Initial index of bloomed flowers
- ArrayList<String> flowerName	Name of each flower
- ArrayList<Double> flowerLayoutX	Determine horizontal placement of each flower on the screen.
- ArrayList<Double> flowerLayoutY	Determine vertical placement of each flower on the screen.

2.4.2 Constructor

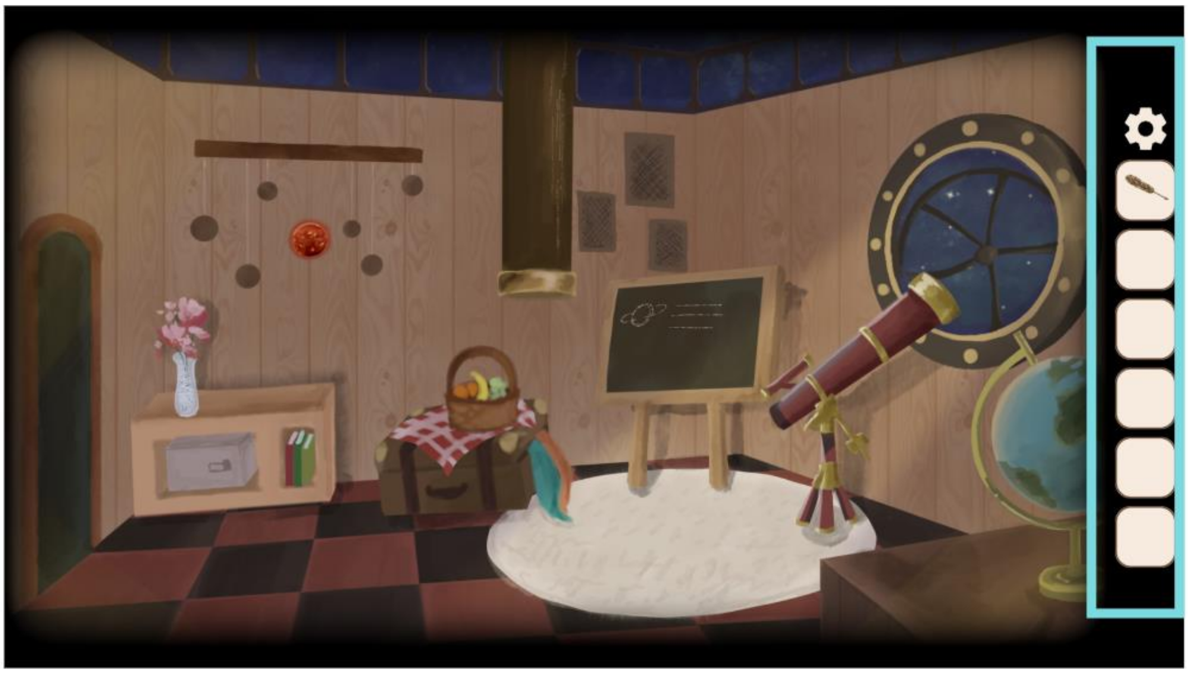
+ FlowerPuzzlePane()	Initialize flowerList. Loop 13 times to: ● Create a FlowerPlayingItem

	with its name and position. ● Check if its index is in initialBloomIndex, and if so, call setBloom(true) to show it bloomed. ● Add each flower to the pane which displays it. ● Also add each to flowerList.

2.4.3 Mehod

+ ArrayList<FlowerPlayingItem> getFlowerList()	Getter for flowerList

2.5 class InventoryPane extends VBox



2.5.1 Field

- ArrayList<Slot> slotsList	Slots

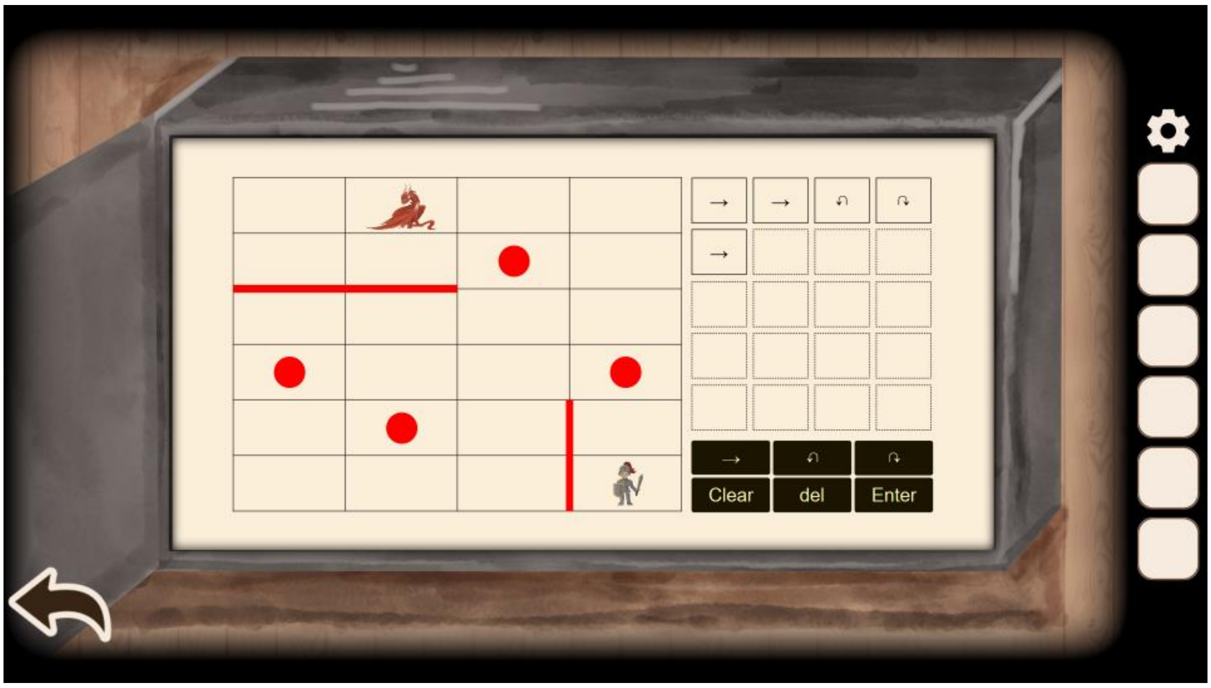
2.5.2 Constructor

+ InventoryPane()	- Initialize the setting button and show the settings pane when clicked, show the hand cursor when mouse hovered; otherwise, show the default cursor. - Initialize 6 slots to store items.

2.5.3 Method

+ ArrayList<Slot> getSlotsList()	Getter for slotsList

2.6 class MazePuzzlePane extends HBox



2.6.1 Fields

- final int ROWS	ROWS = 6
- final int COLS	COLS = 4
- final int ANSWER_ROWS	ANSWER_ROWS = 5
- final int ANSWER_COLS	AMSWER_COLS = 4
- final double CELL_SIZE_WIDTH	CELL SIZE WIDTH = Config.imageBackgroundWidth * 0.1
- final double CELL_SIZE_HEIGHT	CELL SIZE HEIGHT = Config.imageBackgroundHeight * 0.1
- ArrayList<StackPane> mazeCellList	mazeCellList = new ArrayList<StackPane>()
- ArrayList<StackPane> answerCellList	answerCellList = new ArrayList<StackPane>()
- ArrayList<Integer> pathAnswer	pathAnswer = new ArrayList<Integer>()
- int currentIndex	currentIndex = 0
- String[] textBtn	textBtn = { "→", "↶", "↷" }
- final HashMap<String, Integer> nextpathByName	nextpathByName = new HashMap<String, Integer>()
- GridPane grid	Grid of the game
- GridPane btnBox	Btn box of the game
- GridPane answerPane	Answer Pane of the game

2.6.2 Constructor

+ MazePuzzlePane()	- Initialize the direction buttons mapping (→, ↶, ↷) to values (0, 1, -1). - Set the pane's preferred size based on background image dimensions. - Call setInitial() to build and style the maze game.

2.6.3 Methods

- void setInitial()	- Set the background color and layout position of the pane. - Apply drop shadow and inner shadow effects. - Build the maze grid with StackPane cells and add them to mazeCellList. - Add predefined walls using addWall(). - Place red circles in specific cells using addCircle(). - Add two black boxes (start and end) using addBlackBox(). - Create an answer pane (8x8 grid) for user input and store cells in answerCellList.

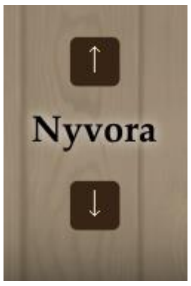
	- Add buttons for input (→, ↶, ↷), clear, delete, and enter. - Combine all UI parts in a vertical layout and add them to the main pane.
- void addTextBtn(GridPane btnBox)	- Iterate through a predefined array textBtn[] - Create one styled button for each text entry in the array - Use textBtn[i] as button label - When clicked, check if there's space for more answers (currentIndex < ANSWER COLS * ANSWER ROWS) - If space exists: ● Record the button's text in an answer cell via setTextAnswerCell() ● Add a corresponding path value from nextpathByName map to pathAnswer list ● Increment the currentIndex counter - Change to hand pointer on hover - Revert to default cursor when mouse exits - Add all buttons to a btnBox GridPane at position (i, 0) - Create a horizontal row of buttons
- void addClearBtn(GridPane	- Add a "Clear" button that resets

btnBox)	the answer grid and clears the pathAnswer when clicked. - Show the hand cursor when mouse hovered; otherwise, show the default cursor. - Add it to a btnBox GridPane at position (0, 1).
- void addDelBtn(GridPane btnBox)	- Add a "Del" button that removes the last entered direction from the answer grid and pathAnswer when clicked. - Show the hand cursor when mouse hovered; otherwise, show the default cursor. - Add it to a btnBox GridPane at position (1, 1)
- void addEnterBtn(GridPane btnBox)	- Add an "Enter" button that calls displayResult() from MazePuzzleController when clicked. - Show the hand cursor when mouse hovered; otherwise, show the default cursor. - Add it to a btnBox GridPane at position (2, 1)
- void addWall(int row, int col, String side)	Visually add a red wall to a specified side of a cell (top, right, bottom, or left).
- void addCircle(int row, int col)	Add a red circle shape to the center of a specified cell.
- void addBlackBox(int row, int col, String label, String itemLabel)	Add a clickable black box to a specific cell. If the current inventory item

	matches itemLabel: ● Replace the black box with a label. ● Make the background transparent. ● Update the puzzle's status using MazePuzzleController. - Show the hand cursor when the mouse hovered.
- StackPane getMazeCell(int row, int col)	Return the StackPane (cell) from the maze grid at a specific row and column.
- StackPane getAnswerCell(int row, int col)	Return the StackPane from the answer grid at a specific row and column.
- void setTextAnswerCell(String text, int index)	- Set the text of an answer cell at a linear index. - Change the cell border to solid if it has text, dashed if empty.
- void setColor(int row, int col, String color)	Change the background color of a maze cell: ● Green for success path ● Red for error ● Transparent for reset
+ void clearMazePane()	Clear all maze cells by setting their

	color to transparent.
+ ArrayList<Integer> getPathAnswer()	Getter for pathAnswer
+ <u>int getRows()</u>	Getter for ROWS
+ <u>int getCols()</u>	Getter for COLS

2.7 class PlanetPuzzleButton extends VBox



2.7.1 Fields

- <u>final int NUM_OF_PLANET</u>	NUM_OF_PLANET = 6
- int planetIndex	Index of each planet
- Text text	Text that will be displayed.
- Button upBtn, downBtn	Buttons for changing text of each planet

2.7.2 Constructor

+ PlanetPuzzleButton()	- Set the button layout size, alignment, and spacing. - Initialize the Text node to display the current planet name with styling.

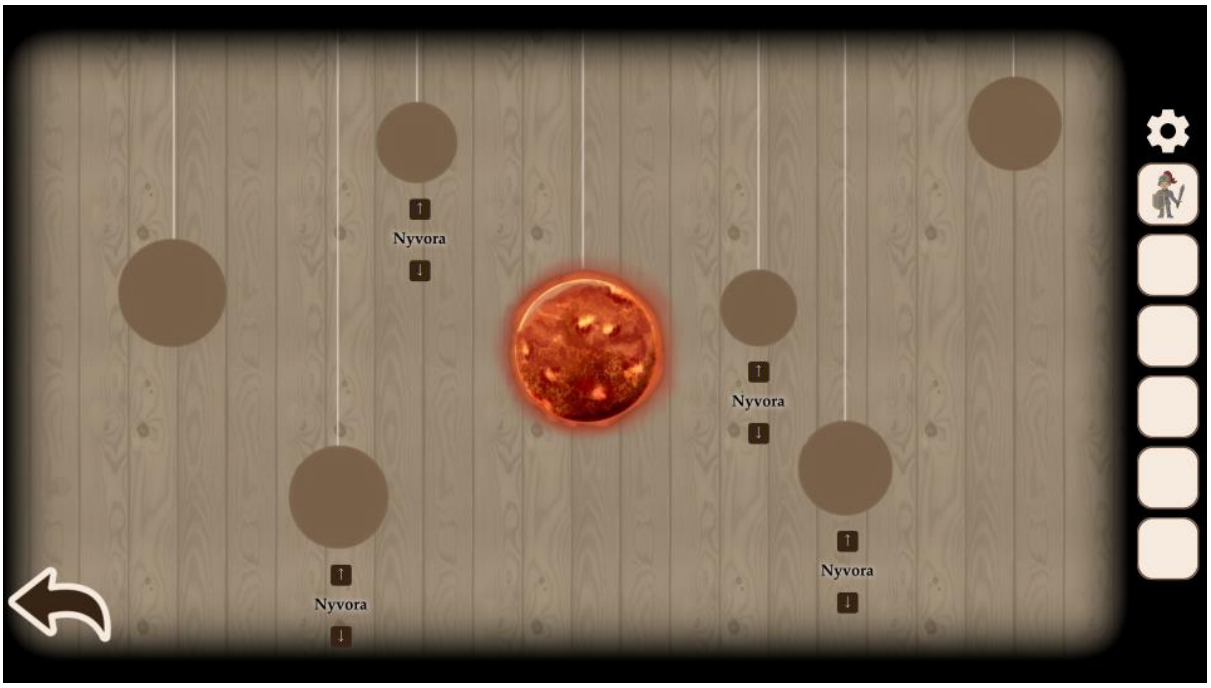
	- Initialize planetIndex to 0. - Create two styled buttons (↑ and ↓) using creatButtonWithStyle(). Assign event handlers to the buttons: ● ↑ button calls nextUp() to go to the next planet when clicked, show the hand cursor when mouse hovered; otherwise, show the default cursor. ● ↓ button calls nextDown() to go to the previous planet when clicked, show the hand cursor when mouse hovered; otherwise, show the default cursor. Add the buttons and text node to the layout VBox.

2.7.3 Methods

+ Button creatButtonWithStyle(String text)	Create and return a styled Button with Bistre background color, Cream text color, Rounded corners and Font based on Config.SUB_FONT and Config.CAPTION_SIZE

+ void nextUp()	- Increment planetIndex by 1 and wrap it using modulo to loop back to 0 after the last planet. - Update the display and check win status via updateStatus()
+ void nextDown()	- Decrement planetIndex by 1 and wrap it using modulo to stay within valid range. - Update the display and check win status via updateStatus()
+ void updateStatus()	Call checkWin() on PlanetPuzzleController: ● If the current state is a winning state, it sets the puzzle status to COMPLETE and updates the UI accordingly.
+ Text getText()	Getter for text
+ void setText(String text)	Setter for text
+ int getPlannetIndex()	Getter for planetIndex
+ void setPlannetIndex(int planetIndex)	- Setter for planetIndex - Update the Text node to reflect the planet name at the new index from PlanetPuzzleController.getPlanetName().

2.8 class PlanetPuzzlePane extends Pane



2.8.1 Fields

- <u>final double[][] positionBtn</u>	positionBtn = { { 0.225, 0.82 }, { 0.295, 0.28 }, { 0.596, 0.52 }, { 0.675, 0.77 } }
- ArrayList<PlanetPuzzleButton> planetBtnList	Store planet's button.

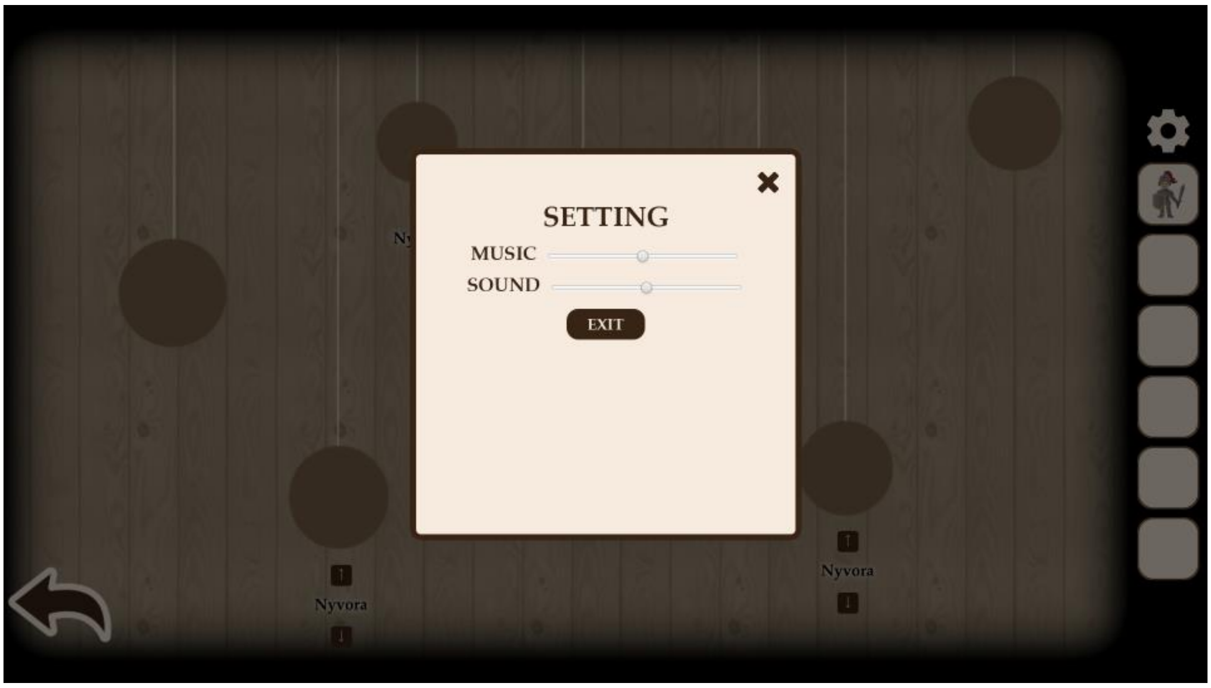
2.8.2 Constructor

+ PlanetPuzzlePane()	- Set the pane's preferred size - Initialize planetBtnList - Create 4 buttons, set each of them their positions on screen, add the buttons to planetBtnList and the layout

2.8.3 Method

+ ArrayList<PlanetPuzzleButton> getPlanetBtnList()	Getter for planetBtnList

2.9 class SettingPane extends StackPane



2.9.1 Constructor

+ SettingPane()	1. Set a semi-transparent black background to dim the main screen behind the settings pane. 2. Set the pane's dimensions to fill the entire screen. 3. Settings Box (VBox): - A centered box (VBox) with fixed width/height (500x500), cream background, and a brown border with rounded corners and padding.

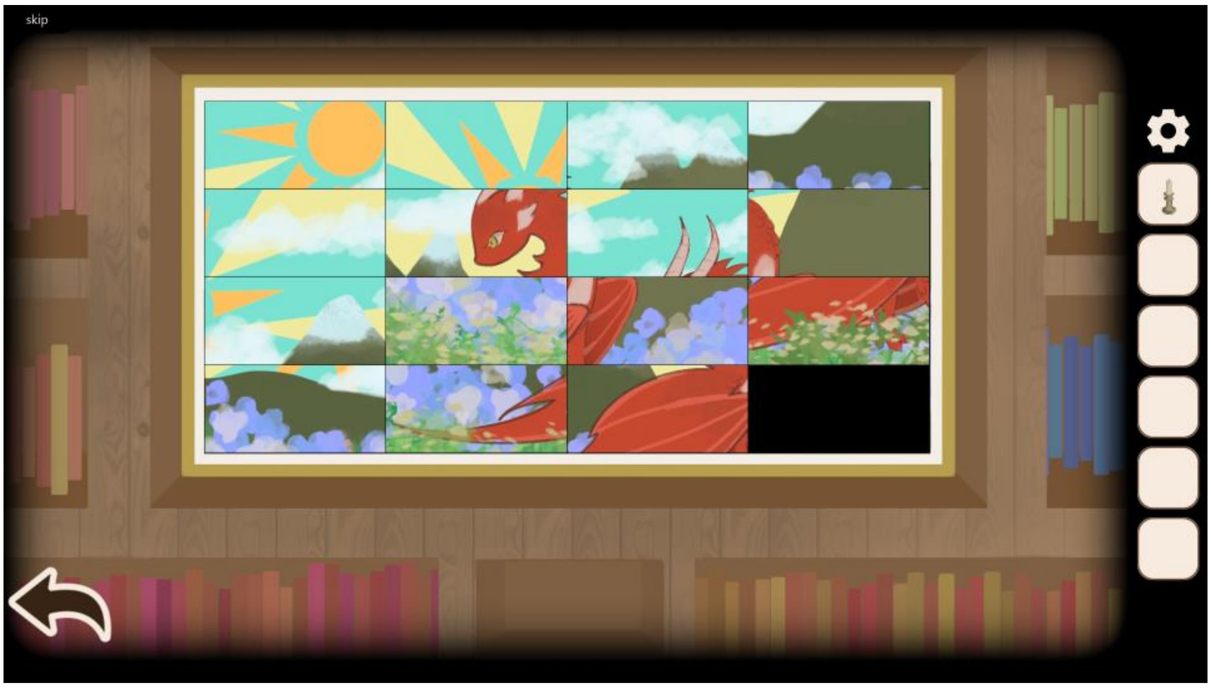
	- Alignment is set to TOP_CENTER. 4. Close Button: - An ImageView with a close icon image. - Clicking the image calls SceneController.getInstance().closeSettingPane() to close the settings. - Show the hand cursor when mouse hovered; otherwise, show the default cursor. 5. Header: - Display the title "SETTING" with bold font and custom color. 6. Sliders: - Create two Slider controls: one for background music (BGM), one for sound effects (EFF), both initially set to 0.5. 7. Exit Button: - A styled button labeled "EXIT". - Clicking it calls SceneController.getInstance().exitGame() to quit the game. - Show the hand cursor when mouse hovered; otherwise, show the default cursor. 8. Add Components: - Assemble all elements

	(borderPane, header, sliders, and exit button) into the settingBox. - Align settingBox in the center of the StackPane. 9. Sound Initialization: - Initialize the sound system and connect sliders to control volume via SoundController.

2.9.2 Method

+ HBox createHbox(String txt, Slider slider)	- Create a new HBox with 10-pixel spacing and center alignment. - Create a Text label using the provided txt string. - Set styling (font and color) for the label. - Set the slider's preferred width to 250 pixels. - Add the label and slider to the HBox. - Return the configured HBox to be added to the settingBox.

2.10 class SlidingPuzzlePane extends GridPane



2.10.1 Field

- ArrayList<SlidingSquarePlayingItem> allCells	allCells = new ArrayList<>()

2.10.2 Constructor

+ SlidingPuzzlePane()	- Set horizontal gap and vertical gap to 1 - Set the inset padding of 0 - Set layoutX to config.Config.imageBackgroundWidth*0.1785 - Set layoutY to config.Config.imageBackgroundHeight*0.152 - Set background to BLACK color, corner radii EMPTY, insets EMPTY - Set border to BLACK color, SOLID borderstrokestyle, corner radii EMPTY, insets width = 1. - Initialize

	SlidingSquarePlayingItem and add them to allCells and to this pane in the form of 4*4 grid square

2.10.3 Method

+ ArrayList<SlidingSquarePlayingItem> getAllCells()	Getter for allCells

2.11 class Slot extends Pane

2.11.1 Fields

- BaseItem item	Item in the slot
- boolean isHold	Check whether the slot is hold or not

2.11.2 Constructor

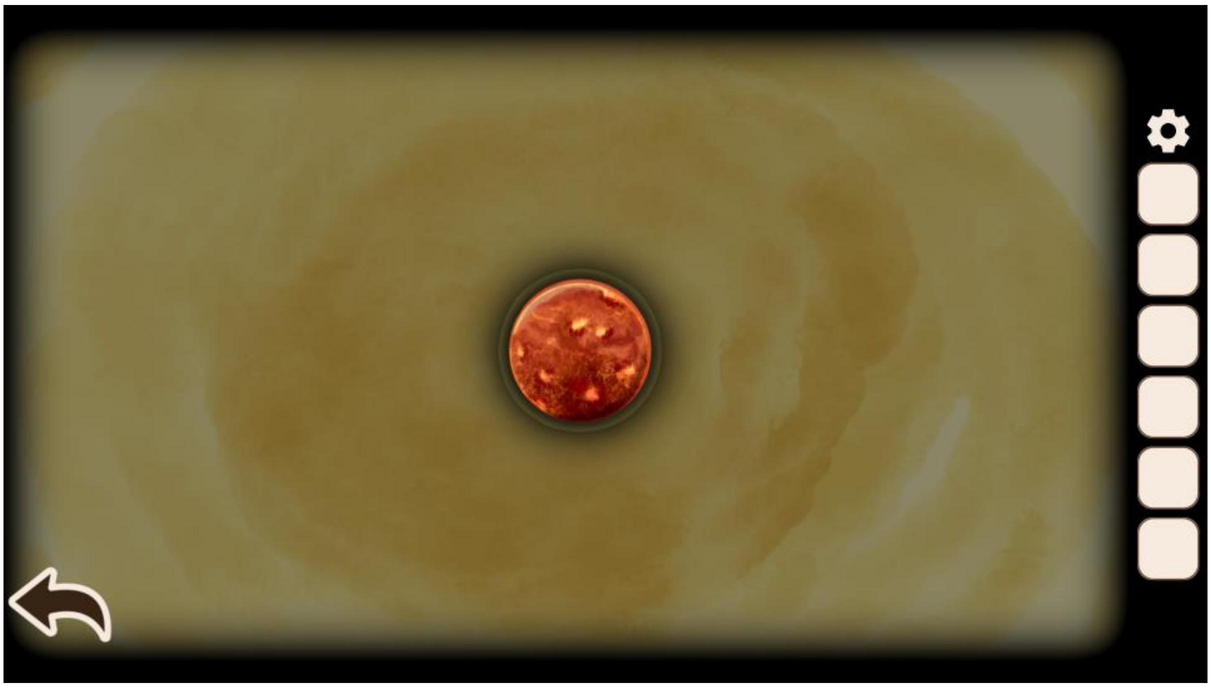
+ Slot()	-Set preferred width and height to 80 - Call setInitialSlot() to initialize slot - Set onMouseClicked to handle with clickHandler() method.

2.11.3 Methods

+ void clickHandler()	This method is the handler method. Does the following: - if item is null, do nothing - if the slot is held, set effect to null, set hold to false and deselect this slot by using InventoryController.getInstance().setCurrentSlot(null). - Otherwise: if InventoryController.getInstance().getCurrentSlot() is not null, set effect to null by using InventoryController.getInstance().getCurrentSlot().setEffect(null) and set hold to false by using InventoryController.getInstance().getCurrentSlot().setHold(false). Select this slot by using InventoryController.getInstance().setCurrentSlot(this), set effect to drop shadow and set hold as true.
+ void setInitialSlot()	- Set background to CREAM color, 20 corner radii, EMPTY insets. - Set border to SAND color, stroke style SOLID, corner radii 20, with 2 width. - Set hold to false. - Set effect to null. - Clear the pane if it is not empty.
+ BaseItem getItem()	Getter for item
+ void setItem(BaseItem item)	Setter for item and add its

	imageView to the layout.
+ boolean isHold()	Getter for isHold
+ void setHold(boolean isHold)	Setter for isHold
+ boolean equals(String name)	Check whether the provided name is equal to the item's name from this slot.

2.12 class TelescopePuzzlePane extends StackPane



2.12.1 Fields

- ImageView backGalaxy	Store the image of the backGalaxy.
- double x	x = 0
- double y	y = 0
- double lastX, lastY	Store latest positions of the backGalaxy.

- double maxX, maxY, minX, minY	Max and min value of x and y

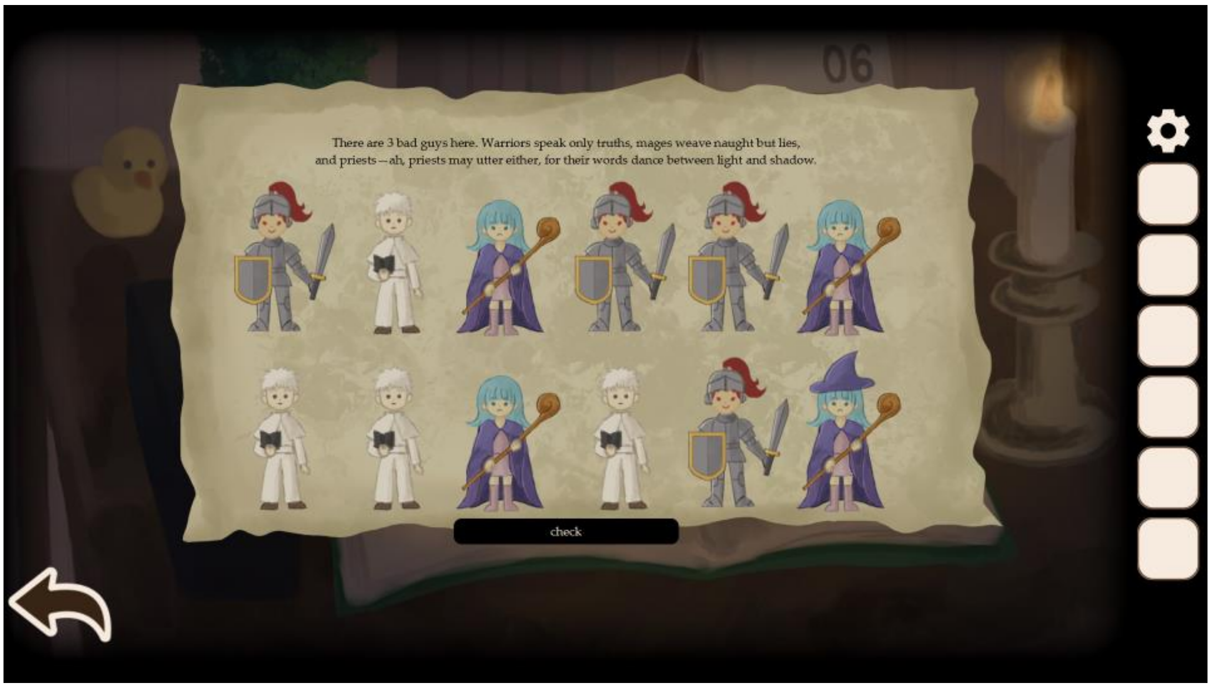
2.12.2 Constructor

+ TelescopePuzzlePane()	- Set preferred width and height using values from Config - Vertically centers the pane on the screen. - Load and set up the movingGalaxy.JPG image as the scrollable background and ensure it fits the defined width and maintains its aspect ratio. - Load the telescope.PNG image as the foreground overlay, positioned at the top-left and apply an InnerShadow to simulate a telescope view. - Add the galaxy background first, then the telescope image to ensure proper stacking. - Apply a rectangle clip so that only the defined area is visible (acts like a telescope viewport) - On **mouse press**, store the initial X and Y coordinates. - On **mouse drag**, call scroll(dx, dy) to move the galaxy image based on how far the mouse has moved, and update the last known position. - Show the hand cursor when mouse hovered; otherwise, show the default cursor. - Call setEdge() to calculate min/max bounds for galaxy movement.

2.12.3 Methods

+ void scroll(double dx, double dy)	- Update internal x and y coordinates using delta movement (dx, dy), but clamp them to stay within defined minX, maxX, minY, maxY bounds. - Move the backGalaxy image accordingly. - Check for a win condition using the TelescopePuzzleController. If the puzzle is solved: ● Set status to STEP1. ● Call update() to trigger whatever logic is needed post-win.
+ void setEdge()	Calculate the maximum and minimum limits for dragging the galaxy image.
+ double getX()	Getter for x
+ double getY()	Getter for y

2.13 class WhoPuzzlePane extends GridPane


There are 3 bad guys here. Warriors speak only truths, mages weave naught but lies,
and priests—ah, priests may utter either, for their words dance between light and shadow.

2.13.1 Fields

- GridPane gridPane	For placing characters
- ArrayList<PersonPlayingItem> personList	Store 12 character.
- final ArrayList<String> SENTENCE_LIST	Store each character’s sentence.
- final ArrayList<String> IMAGE_PATHS	Store images of characters.

2.13.2 Constructor

+ WhoPuzzlePane()	- Set fixed dimensions from configuration. - Create a non-repeating centered background image ("blankPaper.png").

	- Background scales to fill pane while preserving aspect ratio. - Align child elements to top-center. - Create a 1*1 GridPane (likely expands based on content). - Initialize storage for a person's items. - Create a 6*2 grid (6 columns, 2 rows) of characters. - Each PersonPlayingItem contains: ● Base name "person" ● Image path from IMAGE_PATHS ● Associated text from SENTENCE_LIST - Store all persons in personList for later access. - Add each to the grid at proper positions. - Center the character grid. - Add supporting UI elements via: ● addText(): Likely instructions or title ● addGameDetail(): Possibly scoring or clues ● addCheckBtn(): Submission/verification button

	- Add the grid to the main pane.

2.13.3 Methods

+ void addCheckBtn()	- Uses a pre-configured button style from SceneController - Standardizes appearance using game fonts and sizes - The button's text "check" suggests it verifies player answers - When clicked, trigger the puzzle controller's update logic. - Show the hand cursor when the mouse hovered; otherwise, show the default cursor.
+ void addGameDetail()	- Use a BorderPane for flexible text positioning. - Maintain consistent styling with game configuration. - Contain poetic, lore-friendly puzzle instructions. - Explain the core mechanics: ● 3 antagonists among the characters ● Warriors always tell the truth ● Mages always lie ● Priests may do either

+ ArrayList<PersonPlayingItem> getPersonList()	Getter for personList
+ void setPersonList(ArrayList<PersonPlayingItem> personList)	Setter for personList

3. Package items

3.1 interface Pickable

This interface defines methods for BaseItem that can be picked.

3.1.1 Methods

+ void pick()	This method will be called when the BaseItem is picked.
+ boolean isPick()	Self-explanatory

3.2 interface Playable

This interface defines methods for BaseItem that can be played.

3.2.1 Method

+ void play()	This method will be called when the BaseItem is played.

3.3 class BaseItem extends StackPane

3.3.1 Fields

- String name	BaseItem’s name
- ImageView imageView	BaseItem’s imageView

3.3.2 Constructors

+ BaseItem(String name, String imagePath)	- Initialize the BaseItem with the provided name and imagePath. - Add the ImageView to the pane. - Set onMouseClicked to handle with clickHandler() method. - Set up a listener for mouse movement events (setOnMouseMoved) on a JavaFX node (likely an ImageView or a container holding one). - Get the Image from an ImageView. - Use a PixelReader to inspect pixel data. - If the PixelReader is unavailable, it sets the cursor to the default arrow (Cursor.DEFAULT). - Calculate the actual pixel position in the image by scaling the mouse coordinates (e.getX(), e.getY()) based on the ratio of the image's dimensions to the ImageView's display bounds. - Check if the computed position is within the image boundaries. ● If the pixel at the scaled position has an opacity > 0.1, it changes the cursor to a hand (Cursor.HAND), indicating interactivity (e.g., clickable areas); otherwise, it

	reverts to the default cursor. - Ensure mouse events are triggered even over transparent regions of the node (otherwise, only opaque parts would trigger events).
+ BaseItem(String name, String imagePath,double width,double height,double layoutX,double layoutY)	- Initialize the BaseItem with the provided name and imagePath. - Set the ImageView’s width to fit config.Config.imageBackgroundWidth * width, set preserve ratio to true and set onMouseClicked to handle with clickHandler() method. - Add the ImageView to the pane. - Set this layoutX to config.Config.imageBackgroundWidth * layoutX. - Set this layoutY to config.Config.imageBackgroundHeight * layoutY - Set up a listener for mouse movement events (setOnMouseMoved) on a JavaFX node (likely an ImageView or a container holding one). - Get the Image from an ImageView. - Use a PixelReader to inspect pixel data. - If the PixelReader is unavailable, it sets the cursor to the default arrow (Cursor.DEFAULT).

	- Calculate the actual pixel position in the image by scaling the mouse coordinates (e.getX(), e.getY()) based on the ratio of the image's dimensions to the ImageView's display bounds. - Check if the computed position is within the image boundaries. ● If the pixel at the scaled position has an opacity > 0.1, it changes the cursor to a hand (Cursor.HAND), indicating interactivity (e.g., clickable areas); otherwise, it reverts to the default cursor.

3.3.3 Methods

+ void clickHandler()	- If the BaseItem is pickable, call the pick() method. - Otherwise, if it is playable, call the play() method.
+ String getName()	Getter for name
+ ImageView getImageView()	Getter for imageView
+ void setImageView(String imagePath)	Setter for imageView

3.4 class BookPlayingItem extends BaseItem implements Playable

This class represents BookPlayingItem which can be played.

3.4.1 Fields

- int index	the current index of the book
- int bookId	the book’s id

3.4.2 Constructor

+ BookPlayingItem(String name, int index, int bookId)	- Initialize the BookPlayingItem with the provided name and the imageView as an empty string. - Set the imageView from the base BaseItem to have width fit toconfig.Config.imageBackground Width*0.034 and set its preserve ratio to true. - Set the index and the bookId with the provided index and bookId. - Set up a listener for mouse movement events (setOnMouseMoved) on a JavaFX node (likely an ImageView or a container holding one). - Get the Image from an ImageView. - Use a PixelReader to inspect pixel data. - If the PixelReader is unavailable, it sets the cursor to the default arrow (Cursor.DEFAULT). - Calculate the actual pixel position in the image by scaling the mouse coordinates (e.getX(), e.getY()) based on the ratio of the image's dimensions to the ImageView's

	display bounds. - Check if the computed position is within the image boundaries. ● If the pixel at the scaled position has an opacity > 0.1, it changes the cursor to a hand (Cursor.HAND), indicating interactivity (e.g., clickable areas); otherwise, it reverts to the default cursor.

3.4.3 Methods

+ void play()	Define the behavior when the BookPlayingItem is "played", as required by the Playable interface. - If the index is 12, it calls swap(index-1, index) on the BookPuzzleController instance. - Otherwise, it calls swap(index, index+1) on the same controller.
+ void setIndex(int index)	Setter for index
+ int getIndex()	Getter for index
+ void setBookId(int bookId)	Setter for bookId
+ int getBookId()	Getter for bookId

3.5 class FlowerPlayingItem extends BaseItem implements Playable

This class represents FlowerPlayingItem which can be played.

3.5.1 Field

- boolean bloom	the flower’s state

3.5.2 Constructor

+ FlowerPlayingItem(String name, boolean bloom, double layoutX, double layoutY)	- Initialize the FlowerPlayingItem with the provided name, layoutX, layoutY, and An image path that depends on whether the flower is bloomed or unbloomed. - Set up a listener for mouse movement events (setOnMouseMoved) on a JavaFX node (likely an ImageView or a container holding one). - Get the Image from an ImageView. - Use a PixelReader to inspect pixel data. - If the PixelReader is unavailable, it sets the cursor to the default arrow (Cursor.DEFAULT). - Calculate the actual pixel position in the image by scaling the mouse coordinates (e.getX(), e.getY()) based on the ratio of the image's dimensions to the ImageView's display bounds. - Check if the computed position is within the image boundaries. ● If the pixel at the scaled

	position has an opacity > 0.1, it changes the cursor to a hand (Cursor.HAND), indicating interactivity (e.g., clickable areas); otherwise, it reverts to the default cursor.

3.5.3 Methods

+ void play()	- Toggle the bloom state of the flower (bloomed ↔ unbloomed). - Notify the FlowerPuzzleController (via changeStatus) using the flower's name. Likely used to update game logic or track progress.
+ boolean isBloom()	Getter for bloom
+ boolean setBloom(boolean bloom)	Setter for bloom
+ void toggle()	- If it is bloomed, set it to unbloomed. - If it is unbloomed, set it to bloomed.

3.6 class NormalItem extends BaseItem implements Pickable

This class represents NormalItem which can be picked.

3.6.1 Field

- boolean pickState	The state of the NormalItem

3.6.2 Constructor

+ NormalItem(String name, String imagePath,double width,double height,double layoutX,double layoutY)	- Initialize the NormalItem with the provided name, imagePath, width, height, layoutX, layoutY. - Set the pickState to false. - Set up a listener for mouse movement events (setOnMouseMoved) on a JavaFX node (likely an ImageView or a container holding one). - Get the Image from an ImageView. - Use a PixelReader to inspect pixel data. - If the PixelReader is unavailable, it sets the cursor to the default arrow (Cursor.DEFAULT). - Calculate the actual pixel position in the image by scaling the mouse coordinates (e.getX(), e.getY()) based on the ratio of the image's dimensions to the ImageView's display bounds. - Check if the computed position is within the image boundaries. ● If the pixel at the scaled position has an opacity > 0.1, it changes the cursor to a hand (Cursor.HAND), indicating interactivity (e.g., clickable areas); otherwise, it reverts to the default cursor.

3.6.3 Methods

+ void pick()	If it is not picked, add it to the inventory using addItem(BaseItem) method from InventoryController and set the pickState to true.
+ boolean isPick()	Getter for pickState

3.7 class PersonPlayingItem extends BaseItem implements Playable

This class represents PersonPlayingItem which can be played.

3.7.1 Fields

- boolean isSelect	Check whether the person is selected or not.
- String sentence	The person's sentence

3.7.2 Constructor

+ PersonPlayingItem(String name, String imagePath, String sentence)	- Initialize the PersonPlayingItem with the provided name and imagePath. - Set the image width relative to the background width (10%) and preserve the aspect ratio. - Set selection state (isSelect) to false. - Set the sentence used for description or interaction.

	- Add two mouse event handlers: ● On Mouse Entered: Update some on-screen text with the sentence (via WhoPuzzleController). ● On Mouse Exited: Clear the on-screen text (reset it to an empty string).

3.7.3 Methods

+ void play()	Call the select method in WhoPuzzleController, passing in this item. This likely marks the item as selected within the puzzle logic.
+ boolean isSelect()	Getter for isSelect
+ void setSelect(boolean isSelect)	- If true, apply a white glowing DropShadow effect. - If false, set the effect to null. - Update the isSelect field.
+ String getSentence()	Getter for sentence
+ void setSentence(String sentence)	Setter for sentence

3.8 class class SlidingSquarePlayingItem extends BaseItem implements Playable

This class represents SlidingSquarePlayingItem which can be played.

3.8.1 Fields

- int positionX	Current x position of the grid
- int positionY	Current y position of the grid
- int currentId	The grid’s id

3.8.2 Constructor

+ SlidingSquarePlayingItem(String name, String imagePath, int positionX, int positionY, int id)	- Initialize the SlidingSquarePlayingItem with the provided name and an empty imagePath. - Set the square's X and Y grid positions using setters. - Set its ID (which also determines the image shown). - Configure the image view's width to 16% of the background width and maintain the aspect ratio. - Show the hand cursor when mouse hovered; otherwise, show the default cursor.

3.8.3 Methods

+ void play()	- Calculate the Manhattan distance between this square and the empty square's position. - If the distance is 1, call swap() in SlidingPuzzleController to move this square into the empty space.

+ int getCurrentId()	Getter for currentId
+ int getPositionX()	Getter for positionX
+ void setPositionX(int positionX)	Setter for positionX
+ int getPositionY()	Getter for positionY
+ void setPositionY(int positionY)	Setter for positionY
+ void setCurrentId(int id)	- Setter for currentId - If the ID is 15, it sets the image to an "empty square". Otherwise, it constructs an image path based on the row and column (derived from id) and loads the corresponding image.

4. Package logic

4.1 abstract class BaseGameController

4.1.1 Fields

- Status status	The BaseGameController's status
- DisplayPane displayPane	The BaseGameController's displayPane

4.1.2 Constructor

+ BaseGameController(DisplayPane displayPane)	- Initialize the BaseGameController with the provided displayPane. - Set the status to Status.**INITIAL**

4.1.3 Methods

+ void init()	A method used for knitting the games. Its behavior varies between different types of games.
+ boolean checkWin()	A method used for verifying if the game is won or not. Its behavior varies between different types of games.
+ void update()	A method used for updating UI. Its behavior varies between different types of games.
+ BaseGameController getInstance()	Always return null.
+ Status getStatus()	Getter for status
+ void setStatus(Status status)	Setter for status
+ DisplayPane getDisplayPane()	Getter for displayPane
+ void setDisplayPane(DisplayPane displayPane)	Setter for displayPane

4.2 class BookPuzzleController extends BaseGameController

4.2.1 Fields

- BookPuzzleController instance	A singleton instance of the controller.
- final Image spiderWebImage	spiderWebImage = new Image(ClassLoader.getSystemResource("images/bookPuzzle/spiderWeb.png").toString())
- BookPuzzlePane bookPane	The main UI component that contains the book puzzle elements.

4.2.2 Constructor

+ BookPuzzleController(DisplayPane displayPane)	- Initialize the BookPuzzlePane with the provided displayPane. - Set instance = this.

4.2.3 Methods

+ void Init()	- Create a BookPuzzlePane (the puzzle layout) and assign it to the instance. - Add a spider web background image to the display pane. - Add an event button to the display pane, linking it to this controller.
+ BookPuzzleController getInstance()	Provide access to the singleton instance of the controller.
+ void swap(int index1,int index2)	- Swap the bookId values between two BookPlayingItem instances. - If the puzzle is in a winning state, set the game status to COMPLETE and call update().
+ boolean checkWin()	- Iterate through the first 13 book items. - If any book's bookId does not match its index, return false.

	- Return true if all are correctly ordered
+ void update()	If BaseGameController's status is : INITIAL - Check if the current inventory item is `"whisk"`. - If it is, play sound, update status to STEP1, update the description text, replace the background with the puzzle, and remove the item. COMPLETE - Play sound, remove an old UI component from "room1", add a new event button that transitions to scene 2 when clicked, update the background image, and set the current scene to "room1". DEFAULT - Included to suppress warnings (no behavior defined yet).
+ BookPuzzlePane getBookPane()	Getter for bookPane
+ void setBookPane(BookPuzzlePane bookPane)	Setter for bookPane

4.3 class CookController extends BaseGameController

4.3.1 Fields

- <u>CookController instance</u>	A singleton instance of the controller.
- final String CRAFTTABLEV2	CRAFTTABLEV2 = "images/background/craftTableV2.jpg"
- final String CRAFTTABLEV3	CRAFTTABLEV3 = "images/background/craftTableV3.jpg"
- final String CRAFTTABLEV4	CRAFTTABLEV4 = "images/background/craftTableV4.jpg"

4.3.2 Constructor

+ CookController(DisplayPane displayPane)	- Initialize the CookController with the provided displayPane. - Set the instance to this.

4.3.3 Methods

+ void Init()	- Create a "water" button: ● When clicked, it triggers the update() method and removes the button from the display pane. ● Show the hand cursor when the mouse hovered. - Create a "pot" button:

	<ul><li>When clicked, it triggers update() *only if* the game status isn't INITIAL.</li><li>Show the hand cursor when the mouse hovered.</li></ul> - Add both buttons to the displayPane.
+ <u>CookController getInstance()</u>	Provide access to the singleton instance of CookController.
+ boolean checkWin()	Always return false
+ void update()	Check the current Status (from the superclass). Depending on the current Status, perform different updates: <ul><li>INITIAL: Plays a sound, advances to STEP1, update the background image to CRAFTTABLEV2.</li><li>STEP1: If the current inventory item is "flowerCharred", play sound, set status to STEP2, update background to CRAFTTABLEV3, and remove the item.</li><li>STEP2: If the item is "feather", play sound, set status to STEP3, update</li></ul>

	background to CRAFTTABLEV4, and remove the item. ● STEP3: If the item is "goldBar", play sound, set status to COMPLETE, change the scene to scene 16, and remove the item.

4.4 class DescriptionController

4.4.1 Fields

- <u>DescriptionController instance</u>	A singleton instance of the controller.
- ArrayList<ArrayList<String>> descriptionList	Store this game's description.
- ArrayList<ArrayList<Integer>> descriptionNumber	Store the numbers of description.
- Integer[] currentIndex	currentIndex = new int[Config.NUM_OF_SCENE]
- DescriptionPane descriptionPane	To be displayed
- boolean isShowing	isShowing = false

4.4.2 Constructor

+ DescriptionController()	- Initialize descriptionPane. - Initialize descriptionList (stores lines of text). - Initialize descriptionNumber

	(stores indices where steps end). - Call loadDescriptions() to populate those lists from a file.

4.4.3 Methods

+ <u>DescriptionController getInstance()</u>	- Return the existing instance of DescriptionController. - If instance is null, it creates a new one.
+ void reset()	currentIndex = new int[Config.NUM_OF_SCENE]
+ void loadDescriptions()	- Use getResourceAsStream to read /data/descriptions.txt from the classpath. - Throw IllegalStateException if the file is missing. - Read File Line-by-Line ● Wrap the InputStream in a Scanner for efficient line parsing. ● Process the file in pairs of lines: ● Line 1 (Numbers): Split by commas (",") → Convert to Integer → Store in descriptionNumber. ● Line 2 (Text): Split by hashes ("#") → Store

	substrings in descriptionList. - Catch and print exceptions (e.g., file corruption, parsing errors) but don't rethrow them (could hide failures).
+ void showDescriptions()	- Fetch the current scene and step. - Only show text if the current index hasn't reached its target (currentIndex < endIndex). Pull the next text from descriptionList (preloaded strings) and increment the index. - When finished (currentIndex == endIndex), it automatically hides the description pane.
+ void toggleDescriptions(String sentence)	If the description is hidden: ● It adds descriptionPane to the current display pane (accessed via SceneController). ● It sets the text of descriptionPane to the provided sentence. ● It updates isShowing to true. If the description is already showing:

	• It calls hideDescriptions() to hide the description pane.
+ void hideDescriptions()	If it’s currently being shown: • Clear the text in descriptionPane. • Remove descriptionPane from the current display pane. • Set isShowing to false.
+ boolean isShowing()	Getter for isShowing
+ DescriptionPane getDescriptionPane()	Getter for descriptionPane

4.5 class FlowerPuzzleController extends BaseGameController
4.5.1 Fields

- FlowerPuzzleController instance	A singleton instance of the controller.
- final ArrayList<String> FLOWER_ORDER	FLOWER ORDER = new ArrayList<String>(Arrays.*asList*("Daisy", "Rose", "Poppy", "ForgetMeNot", "Hibiscus", "Lily"))
- FlowerPuzzlePane flowerPuzzlePane	To be displayed
- final int NUM_OF_FLOWER	NUM_OF_FLOWER = 13

4.5.2 Constructor

+ FlowerPuzzleController(DisplayPane displayPane)	- Initialize the FlowerPuzzlePane with the provided displayPane. - Set the instance to this.

4.5.3 Methods

+ void init()	- Create flowerPane - Set the instance's FlowerPuzzlePane using the created flowerPane. - Also add the flowerPane to the displayPane.
+ <u>FlowerPuzzleController getInstance()</u>	Provide access to the singleton instance of FlowerPuzzleController.
+ void changeStatus(String name)	- Calculate the next flower's name in a circular list (FLOWER ORDER) based on the current name. ● If name is the last in the list, it wraps around to the first. - Iterate over the flower list in flowerPuzzlePane. ● Find the flower with the matching nextOrder name and Toggle its state If the puzzle is solved:

	• Set the object's status to COMPLETE. • Calls update().
+ boolean checkWin()	- Iterate over the flower list in flowerPuzzlePane. - If any flower does not bloom, return false. - Return true if all bloom.
+ void update()	Check if the puzzle status is COMPLETE. If so: • Play a "correct" sound effect. • Remove the flowerPuzzlePane from the display. • Add a new item (NormalItem) to the display pane, representing a flower image. • Set the puzzle status to FINISH.
+ FlowerPuzzlePane getFlowerPuzzlePane()	Getter for flowerPuzzlePane
+ void setFlowerPuzzlePane(FlowerPuzzlePane flowerPuzzlePane)	Setter for flowerPuzzlePane

4.6 class InventoryController

4.6.1 Fields

- <u>InventoryController instance</u>	A singleton instance of the controller.
- InventoryPane inventoryPane	To be displayed
- ArrayList<BaseItem> itemList	Store items.
- Slot currentSlot	The selected slot

4.6.2 Constructor

+ InventoryController()	Initialize itemList and inventoryPane

4.6.3 Methods

+ void addItem(BaseItem item)	- Add the BaseItem to this itemList. - Assign the item to a corresponding visual slot in the inventory pane
+ void removeCurrentItem()	- Assign an item to the currentSlot's item. - Set the current slot to null. - Remove the item from the itemList - Iterate through the first 6 slots: ● Clear each slot to ensure it's empty. ● Reassign items from the updated inventory list to slots in order.

+ InventoryController getInstance()	- Return the existing instance of InventoryController. - If instance is null, it creates a new one.
+ InventoryPane getInventoryPane()	Getter for inventoryPane
+ ArrayList<BaseItem> getItemList()	Getter for itemList
+ boolean isCurrentItem(String itemName)	Check whether the item's name in the current slot is the same as itemName.
+ void setCurrentSlot(Slot slot)	Setter for the currentSlot
+ Slot getCurrentSlot()	Getter for the currentSlot

4.7 class MazePuzzleController extends BaseGameController
4.7.1 Fields

- <u>MazePuzzleController instance</u>	A singleton instance of the controller.
- MazePuzzlePane mazePane	To be displayed.
- Button openBtn, closeBtn	For open and close the safe.
- NormalItem goldItem	To be displayed when cleared.
- <u>int[][] direction</u>	direction = new int[][] { { -1, 0 }, { 0, -1 }, { 1, 0 }, { 0, 1 } }
- final int[][] WRONG_DIRECTION	WRONG_DIRECTION = new int[][] {{5,3,1}, {4,3,1}, {2,0,0}, {2,1,0}, {5,2,3}, {4,2,3}, {1,0,2}, {1,1,2}}
- ArrayList<ArrayList<Integer>>	*pointPosition* = new

pointPosition	ArrayList<ArrayList<Integer>>(Arrays.*asList*(new ArrayList<Integer>(Arrays.*asList*(1, 2)), new ArrayList<Integer>(Arrays.*asList*(3, 0)), new ArrayList<Integer>(Arrays.*asList*(3, 3)), new ArrayList<Integer>(Arrays.*asList*(4, 1))))
- int currentRow, currentCol	currentRow = 5, currentCol = 3
- boolean[] memCheckPoint	memCheckPoint = new boolean[] { false, false, false, false }

4.7.2 Constructor

+ MazePuzzleController(DisplayPane displayPane)	- Initialize the MazePuzzleController with the provided displayPane - Determine goldItem as a created NormalItem with the name "goldBar", the imagePath "images/item/goldbar.png", 0.35 width, 0 height, 0.35 layoutX, 0.4 layoutY.

4.7.3 Methods

+ void init()	- Set the instance's MazePane using the created MazePuzzlePane. - Create a UI button with specific layout parameters and passed to

	the controller’s setBtn method,
+ <u>MazePuzzleController getInstance()</u>	Provide access to the singleton instance of MazePuzzleController.
+ void setBtn(Button openBtn, Button closeBtn)	- Initialize the closeBtn with the provided closeBtn. - When the closeBtn is clicked, it calls the closeSafe() method and when the mouse entered, set the cursor to a hand. - Initialize the openBtn with the provided openBtn. - When the openBtn is clicked, it calls the openSafe() method and when the mouse entered, set the cursor to a hand. - Add the openBtn to the displayPane
+ Boolean checkWin()	- Verify if the player is at the target maze cell at row 0 and column 1. If not, the player hasn't reached the goal — return false. - Iterate over an array memCheckPoint (likely boolean flags indicating if certain checkpoints were visited). If any of the 4 checkpoints weren’t triggered (false), return false.
+ void update()	If the BaseGameController’s status is COMPLETE, call the playCorrectEff() method, update the background image, remove the mazePane and add the goldItem.
+ void openSafe()	Add the close button to the display pane and remove the open button.

	If the puzzle is complete: ● Set the background to a final image (safeV4.jpg). ● Add the gold item to the display (if not already picked). If the puzzle is not complete: ● Add the maze puzzle UI (mazePane) to the display. ● Update the background to an intermediate image (safeV2.jpg).
+ void closeSafe()	- If the puzzle is complete and the gold item hasn't been picked, it removes the gold item. - If incomplete, it removes the maze puzzle UI (mazePane). - Restore the open button, remove the close button, and set the background to safeV1.jpg.
+ void setMazePane(MazePuzzlePane mazePane)	Setter for mazePane
+ void displayResult()	- Reset the starting position (currentRow = 6, currentCol = 3)

	and checkpoint flags (memCheckPoint). - Start a new thread to visually walk through the path given by mazePane.getPathAnswer(): ● 0 indicates moving forward. ● Non-zero values change the current direction. - If movement is valid: ● Update the maze with green (correct) path color. ● Mark checkpoints as visited if the position matches. - If invalid: ● Colors current cell red and stops. - If all checkpoints are visited and the final position is correct: ● Set status to COMPLETE and trigger the update() method on the JavaFX thread.

	- Otherwise, clear the maze after a delay.
+ boolean checkNextPosition(int row, int col, int direct)	- Ensure the next cell is within maze boundaries (6 rows × 4 columns). - Prevent specific invalid moves based on hardcoded maze walls or obstacles (e.g., certain cells blocking movement in specific directions).

4.8 class PlanetPuzzleController extends BaseGameController
4.8.1 Fields

- ArrayList<String> planetNameList	planetNameList = new ArrayList<String>(Arrays.asList("Nyvora", "Thalorix", "Zerenthia", "Elystra", "Morvex", "Kaelion"))
- ArrayList<Integer> planetIndexAnswerList	planetIndexAnswerList = new ArrayList<Integer>(Arrays.asList(1,3,5,2))
- PlanetPuzzlePane planetPuzzlePane	To be displayed.
- PlanetPuzzleController instance	A singleton instance of the controller.

4.8.2 Constructor

+ PlanetPuzzleController(DisplayPan	- Initialize the PlanetPuzzleController with the

e displayPane)	provided displayPane. - Set the instance to this.

4.8.3 Methods

+ void init()	- Create planetPane. - Set the instance's PlanetPuzzlePane using the created planetPane. - Add the planetPane to the displayPane.
+ PlanetPuzzleController getInstance()	Provide access to the singleton instance of PlanetPuzzleController.
+ ArrayList<String> getPlanetName()	Getter for the planetNameList
+ boolean checkWin()	- If the planetIndex on the button in planetButtonList does not match with the planetIndexAnswerList, return false. - If all is match, return true.
+ void update()	If the BaseGameController's status is COMPLETE, call the playCorrectEff() method,remove the planetPuzzlePane, update the background image, add the mazePiece1 item and set the game's status to FINISH.
+ void setPlanetPuzzlePane(PlanetPuzzlePane planetPuzzlePane)	Setter for planetPuzzlePane

4.9 class SceneController

4.9.1 Fields

- ArrayList<Parent> sceneRoots	
- final Stage stage	
- SceneController instance	A singleton instance of the controller.
- Rectangle blackOverlay	blackOverlay = new Rectangle()
- int currentSceneIndex	currentSceneIndex = 0
- boolean isTransitioning	isTransitioning = false
- Map<String, Integer> SceneIndexMap	SceneIndexMap = new HashMap<String,Integer>()
- SettingPane settingPane	settingPane = new SettingPane()
- ArrayList<BaseGameController> classController	classController = new ArrayList<BaseGameController>()

4.9.2 Constructor

+ SceneController(Stage stage)	- Initialize the scene controller with a Stage. - Set up scene roots from data. - Instantiate and add puzzle controllers to classController. - Call setController(), setHomePage(), setInitialItem(), setInitialBackBtn(). - Add skip buttons for specific puzzles.

	- Add specific event buttons for in-game interactions.

4.9.3 Methods

+ void setInitialSceneRoots()	- Load scene configuration from sceneRootsData.txt. - For each scene, create a DisplayPane and add it to sceneRoots. - Also read and configure the buttons that transition to other scenes
+ void setHomePage()	Set up the initial home screen with a background, game title, “Play” and “Exit” buttons.
+ void setController()	Call init() on each controller in classController to initialize them.
+ void setInitialItem()	Load item data from itemsData.txt and place items in their respective scenes.
+ void setInitialBackBtn()	Add back buttons to scenes 4 through 15. The button behavior depends on the index range: ● Scenes 4–8: back to scene 1. ● Scenes 9–11: back to scene 2. ● Scenes 12–15: back to scene

	3.
+ void showEndingStory()	- Set the current scene to endingStory. - Display the ending background and wait for a click to proceed to the final ending.
+ void showEndingScene()	- Show a congratulatory screen with time spent escaping. - Add an exit button.
+ void addSkipBtn(String sceneName, BaseGameController controller)	Add a “skip” button to a scene that forces the puzzle to complete when clicked.
+ <u>Button createEventBtn(double width,double height,double layoutY,double layoutX)</u>	- Create a transparent button used for triggering in-game events.
+ <u>Button createBtnWithStyle(double width, double height, double layoutY, double layoutX, String text)</u>	Create a stylized button with hover effects and custom font/colors.
+ void addEventBtn()	- Add interactive buttons in specific scenes (fireplace, doll, dragon). - Trigger logic depending on selected inventory items (e.g., flower or knife).
+ <u>SceneController getInstance()</u>	A singleton instance of the controller.
+ <u>SceneController getInstance(Stage stage)</u>	- Return the existing instance of SceneController.

	- If instance is null, it creates a new one.
+ int getCurrentSceneIndex()	Getter for currentSceneIndex
+ DisplayPane getDisplayPaneByName(String sceneName)	Return the DisplayPane associated with a scene name.
+ DisplayPane getCurrentDisplayPane()	Return the DisplayPane of the current scene.
+ void showSettingPane()	Show the settings pane overlay on top of the current scene.
+ void closeSettingPane()	Hide the settings pane overlay on top of the current scene.
+ void setInitialScene()	Set the first scene (home page) on the stage.
+ void setCurrentScene(String sceneName)	Change the current scene using its name.
+ void setCurrentScene(int sceneIndex)	Handle the visual transition between scenes with a fade effect.
+ void transitionScene()	- Implement the fade-to-black and fade-back-in transition effect. - Move InventoryPane and update the scene root visually.
+ void exitGame()	Display a confirmation dialog to the user. If confirmed, exit the application.
+ Stage getStage()	Return the primary Stage of the application.

4.10 class SlidingPuzzleController extends BaseGameController

4.10.1 Fields

- int emptyPositionX	The current x position of the empty grid.
- int emptyPositionY	The current y position of the empty grid.
- SlidingPuzzlePane board	To be displayed.
- SlidingPuzzleController instance	A singleton instance of the controller.
- int[] initialOrder	initialOrder = {4,0,3,9,1,6,2,7,5,12,11,14,8,13,10,15}
- String[] imagePaths	Store initial images' path.

4.10.2 Constructor

+ SlidingPuzzleController(DisplayPane displayPane)	- Initialize the SlidingPuzzleController with the provided displayPane. - Set the instance to this.

4.10.3 Methods

+ void init()	Set the instance's SlidingPuzzlePane using the created SlidingPuzzlePane.
+ SlidingPuzzleController getInstance()	A singleton instance of the controller.
+ void swap(int x1,int y1,int x2,int y2)	- It exchanges the currentId of two cells, effectively swapping their

	images. - Update the puzzle's internal empty tile position to the first tile’s coordinates (x1, y1). - If the puzzle is now solved (checkWin() returns true), it updates the status to COMPLETE and calls update().
+ boolean checkWin()	- Iterate through all 16 cells in the puzzle. - Return false if any cell’s currentId does not match its index. - Return true if all tiles are in the correct order.
+ void update()	If the BaseItemController’s status is COMPLETE: ● Play the correct sound effect. ● Remove the puzzle board from the display. ● Add a new item (match) to the display pane as a reward for completing the puzzle. ● Set the game’s status to FINISH.
+ int getEmptyPositionX()	Getter for the emptyPositionX
+ void setEmptyPositionX(int emptyPositionX)	Setter for the emptyPositionX

+ int getEmptyPositionY()	Getter for the emptyPositionY
+ void setEmptyPositionY(int emptyPositionY)	Setter for the emptyPositionY
+ void setSlidingPuzzlePane(SlidingPuzzlePane board)	- Store the board reference. - Add the SlidingPuzzlePane to the associated DisplayPane.
+ int[] getInitialOrder()	Getter for the initialOrder
+ void setInitialOrder(int[] initialOrder)	Setter for the initialOrdrt
+ String[] getImagePaths()	Getter for the imagePaths
+ void setImagePaths(String[] imagePaths)	Setter for the imagePaths

4.11 class SoundController

4.11.1 Fields

- Media bgm	bgm = new Media(ClassLoader.*getSystemResource*("sounds/backgroundMusic.mp3").toExternalForm())bgm = new Media(ClassLoader.*getSystemResource*("sounds/backgroundMusic.mp3").toExternalForm())
- Media clickEff	clickEff = new Media(ClassLoader.*getSystemResource*("sounds/clickEffect.mp3").toExternalForm())
- Media correctEff	correctEff = new Media(ClassLoader.*getSystemResource*("sounds/correctEffect.mp3").t

	oExternalForm())
- Media wrongEff	wrongEff = new Media(ClassLoader.*getSystemResource*("sounds/wrongEffect.mp3").toExternalForm())
- MediaPlayer bgmPlayer, clickPlayer, correctEffPlayer, wrongEffPlayer	
- Slider bgmSlider, effSlider	
- <u>SoundController instance</u>	A singleton instance of the controller.

4.11.2 Controller

+ SoundController()	- Set up `bgmPlayer` to loop continuously. - Initialize individual `MediaPlayer` instances for each sound effect.

4.11.3 Methods

+ <u>SoundController getInstance()</u>	- If instance is null, create a new SoundController. - Return the singleton instance.
+ void setSlider(Slider bgmSlider,Slider effSlider)	- Bind bgmPlayer volume to bgmSlider. - Bind all effect players (clickPlayer, correctEffPlayer, wrongEffPlayer) to effSlider.

+ void initial()	Start playing the background music when the game initializes.
+ void playClickEff()	Stop any current playback before playing again, ensuring the sound plays even on rapid repeated clicks.
+ void playCorrectEff()	Play the "correct" action sound effect.
+ void playWrongEff()	Play the "wrong" action sound effect.

4.12 enum Status

4.12.1 enum

INITIAL, STEP1, STEP2, STEP3, COMPLETE, FINISH

4.13 class TelescopePuzzleController extends BaseGameController

4.13.1 Fields

- TelescopePuzzlePane telescopePane	To be displayed.
- <u>final double goalX</u>	The goal value of x position.
- <u>final double goalY</u>	The goal value of y position.
- TelescopePuzzleController instance	A singleton instance of the controller.

4.13.2 Constructor

+ TelescopePuzzleController(DisplayPane displayPane)	- Initialize the TelescopePuzzleController with the provided displayPane - Set the instance to this.

4.13.3 Methods

+ void init()	- Create TelescopePuzzlePane. - Set the instance's TelescopePuzzlePane using the created TelescopePuzzlePane. - Also add the TelescopePuzzlePane to the displayPane.
+ <u>TelescopePuzzleController getInstance()</u>	Provide access to the singleton instance of TelescopePuzzleController.
+ boolean checkWin()	Return true if the telescope pane's X and Y positions are within 20 pixels of goalX and goalY.
+ void update()	If the current status is STEP1, it: ● Play a "correct" sound. ● Add a new item (mazePiece2) to the scene named "room3". ● Switch the current scene to scene 3. ● Update the controller status to COMPLETE.
+ TelescopePuzzlePane getTelescopePane()	Getter for the telescopePane
+ void setTelescopePane(TelescopePuzzlePane telescopePane)	Setter for the telescopePane

4.14 class WhoPuzzleController extends BaseGameController

4.14.1 Fields

- WhoPuzzlePane whoPuzzlePane	To be displayed.
- Text text	The characters' sentences that will be displayed.
- int countSelected	countSelected = 0
- ArrayList<Integer> answersIndex	answersIndex = new ArrayList<Integer>(Arrays.asList(2, 9, 11))
- WhoPuzzleController instance	A singleton instance of the controller.

4.14.2 Constructor

+ WhoPuzzleController(DisplayPane displayPane)	- Initialize the WhoPuzzleController with the provided displayPane. - Set the instance to this.

4.14.3 Methods

+ void init()	- Assign a new WhoPuzzlePane to the instance. - Add an event button to the display pane to start the puzzle.
+ WhoPuzzleController getInstance()	Provide access to the singleton instance of WhoPuzzleController.
+ void select(PersonPlayingItem person)	- Limit selections to a maximum of 3.

	- Toggle the selection state of a PersonPlayingItem and update countSelected. - Display a message if the selection limit is exceeded.
+ boolean checkWin()	- Iterate over a list of correct answer indices stored in answersIndex. - For each index: ● It checks if the corresponding person in the puzzle (getWhoPuzzlePane().getPersonList().get(index)) is selected. ● If any correct person is not selected, it: ○ Call updateText("Wrong answer") to provide feedback. ○ Return false — indicating the win condition is not met. - If all correct persons are selected, it returns true.
+ void update()	If the BaseGameController’s status is:

	INITIAL: ● If "candle" is the current inventory item, play the correct sound effect, update the status to STEP1, change the background image to show the candle added and remove the candle from inventory. STEP1: ● If "match" is used, play the correct sound effect, update the status to STEP2, change the background image to a lit room, remove two children from the display pane, add the puzzle pane (whoPuzzlePane) to the scene, add a back button to return to "room2", update "room2" background to a new version, remove the match from inventory, update and show the step description. STEP2: ● If the player has completed the puzzle (checkWin() returns true), play a correct sound effect, change status to COMPLETE, remove the puzzle pane, show a background image with a

	knife, add a new clickable knife button that changes the cursor on hover and link the button to the puzzle controller for interaction. COMPLETE: ● Update the status to FINISH. ● Change the background to show the knife has been taken. ● Remove the last added child from the display pane. ● Create and add a knife item to the player's inventory. ● Remove the knife from the "room2" scene.
+ Text getText()	Getter for the text
+ void setText(Text text)	Setter for the text
+ void updateText(String sentence)	Set the new message on the text.
+ WhoPuzzlePane getWhoPuzzlePane()	Getter for the whoPuzzlePane
+ void setWhoPuzzlePane(WhoPuzzlePane whoPuzzlePane)	Setter for the whoPuzzlePane

5. Package main

5.1 class Main extends Application

5.1.1 Methods

+ void start(Stage stage)	JavaFX Application start method override
+ void main(String[] args)	Launch application