RISE of MOODENG Documentation

Created by

Chayutphong Soisri	6630054621
Siwat Sonkadam	6632219221

2110215 Programming Methodology
Semester 1 Year 2024

Chulalongkorn University

RISE of MOODENG

Storyline

Introduction:

Welcome to ***Rise of a Moodeng***! Journey through the mystical lands of **Beast**, **Forest**, **Jungle**, and **Apocalypse**, where fruits fall from the sky. Your mission is simple: feed Moodeng as much fruit as possible. Help her enjoy the delicious bounty and see how far she can go!

Opening Scene:

Under a magical sky filled with falling fruits, Moodeng awakens at the edge of the Beast region. Beyond lie the enchanted Forest, the wild Jungle, and the mysterious Apocalypse. The only goal is to catch as many fruits as possible. Moodeng is hungry and ready—let the fruit feast begin!

Narrative Progression:

1. Setting the Foundation:

"Welcome to your new adventure, young Moodeng! A vast world awaits you, filled with lush forests, jungles, and mysterious lands. Your journey begins in the Forest, where you'll learn the basics of survival. Your goal is simple: gather fruits, grow stronger, and explore new lands. Each fruit you collect is one step closer to discovering the wonders of this world. You have [initial fruit amount] in your reserves, now let’s start this exciting journey!"

2. Building the Essentials:

"The forest is teeming with life, but it’s time to build up your strength. Collect more fruits to sustain your energy and unlock new areas. The more you eat, the more powerful you become! Remember, your progress is tracked as you explore new regions and collect fruits. Soon, you’ll be able to venture into the Jungle."

3. Unlocking New Opportunities:

"With every fruit you gather, your journey grows. New opportunities are opening up. You can now explore the **Jungle**, where even rarer fruits await! But be prepared, the deeper you go, the tougher the challenges. The more fruits you collect, the more powerful you become, and soon, the *Apocalypse* awaits—will you be ready?"

4. Challenges and Triumphs:
"The path ahead is filled with challenges. The Apocalypse is no easy journey. You'll need to manage your resources wisely, collect the right fruits, and push your limits to face the final challenge. But with every challenge comes the opportunity for greatness. Your connection with the land is growing stronger, and soon you'll discover the ultimate secret of the Moodeng world."

5. Celebrating Success:
"Congratulations, Moodeng! Your journey has come to an end, but the story of your growth and perseverance will echo for generations. You've overcome the challenges, collected every fruit, and transformed yourself into a true hero of the land."

Conclusion:
As the final sun sets over the **Apocalypse** map, a new horizon rises. Moodeng, once a curious traveler, has evolved into a mighty force of nature, having conquered every challenge that stood in the way. The fruits collected, the wisdom gained, and the lands explored are all part of a larger story—a testament to perseverance, growth, and the bond shared with the land.

1. Statement

Objective:

"Guide Moodeng through four diverse landscapes to collect fruits and unlock the secrets of the world. Overcome challenges and grow stronger with every fruit you gather."

Game Setup:

1. This is a single-player game.menu page.
2. The player must first enter their name before starting the game.
3. After entering their name, the player will be directed to the map selection page where they can choose which map they want to play (Beast, Forest, Jungle, Apocalypse).
4. The "How to Play" section will also be available on the map selection page to explain the game mechanics.

Scene Layout:

1. Main menu page
2. Intro to Game Page
3. Map Selector Page
4. Gameplay Scene
5. Game Exit Scene

Game Elements:

Player:

The player’s name will be saved and displayed throughout the game.The player must feed Moodeng different fruits which provide special abilities.

Fruit Abilities:

Each fruit has a unique effect on Moodeng (e.g., increase speed, heal, give temporary invincibility).The player can collect these fruits as they play and observe how Moodeng's abilities change based on the fruit consumed.

Maps

There are four maps: Beast, Forest, Jungle, and Apocalypse. Each map offers different challenges and fruits to collect.

Rule :

Your goal is to guide Moodeng through four different landscapes—Beast, Forest, Jungle, and Apocalypse. Along the way, you will feed him various fruits, each of which imparts special powers, such as increased speed, invincibility, healing, and more.

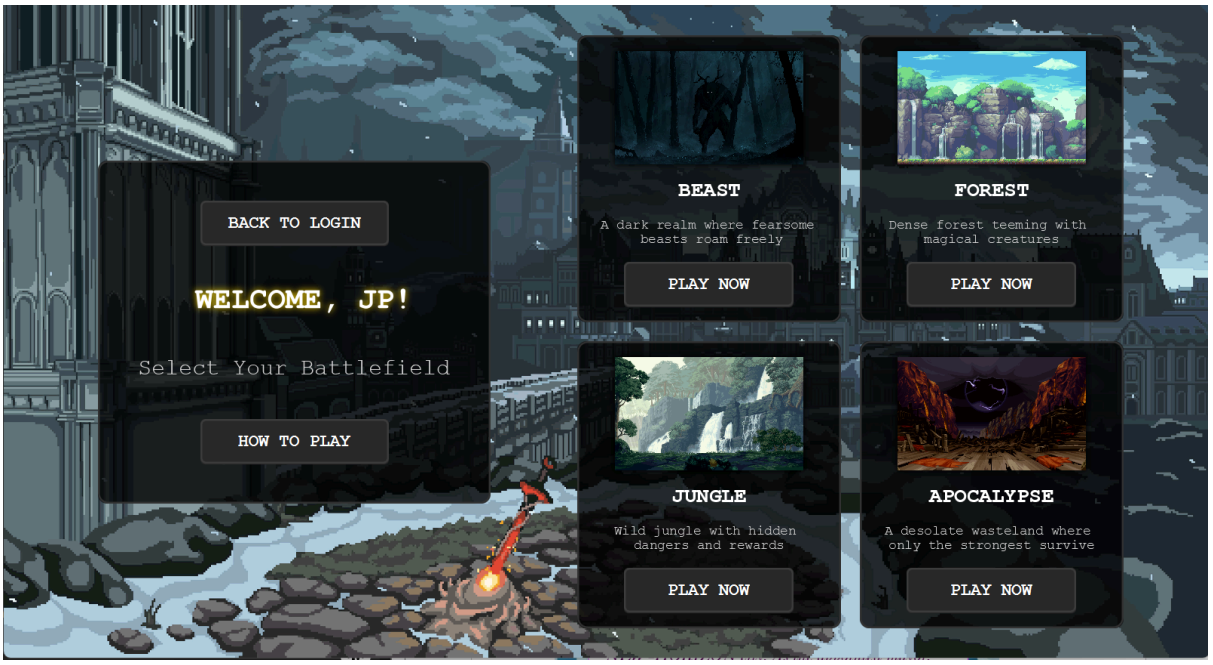
Fruits and Their Effects:

1. **Banana:** Boosts Moodeng’s speed temporarily by 1.5 times.
2. **Coconut:** do nothing.
3. **Watermelon:** Boosts Moodeng’s speed temporarily by 2 times.
4. **Pineapple:** do nothing.

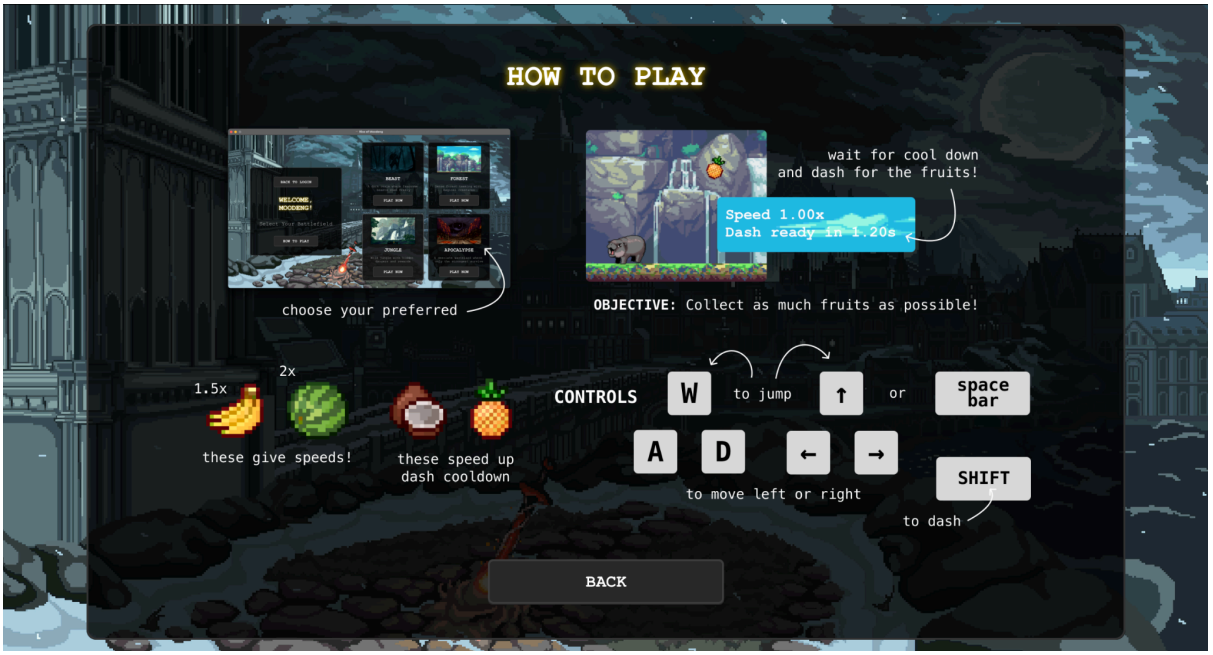
Main Menu Scene

Here is our LoginMenu Scene. When entering the game you will see this scene very first ! Moreover, you will experience mesmerizing game background sound as well.

After clicking **BEGIN ADVENTURE** Button, the page below will appear. You can click to choose the map and then click **PLAY NOW** Button to start the game.

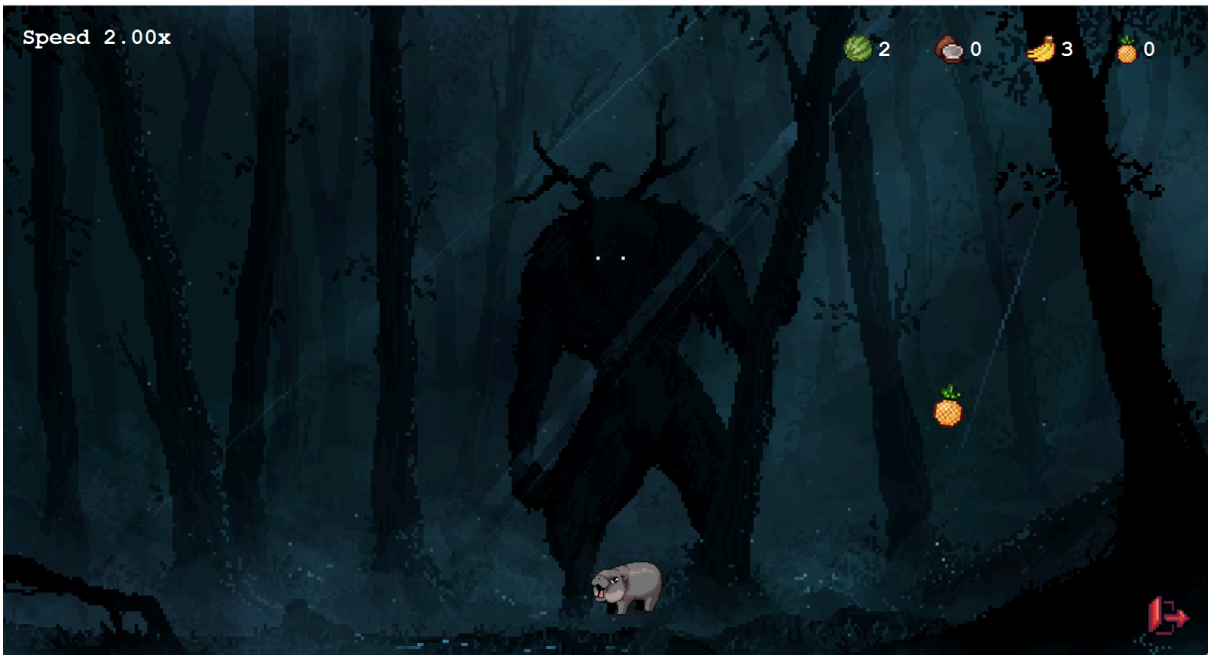


but if you click the HOW **TO PLAY** button. this page will pop up



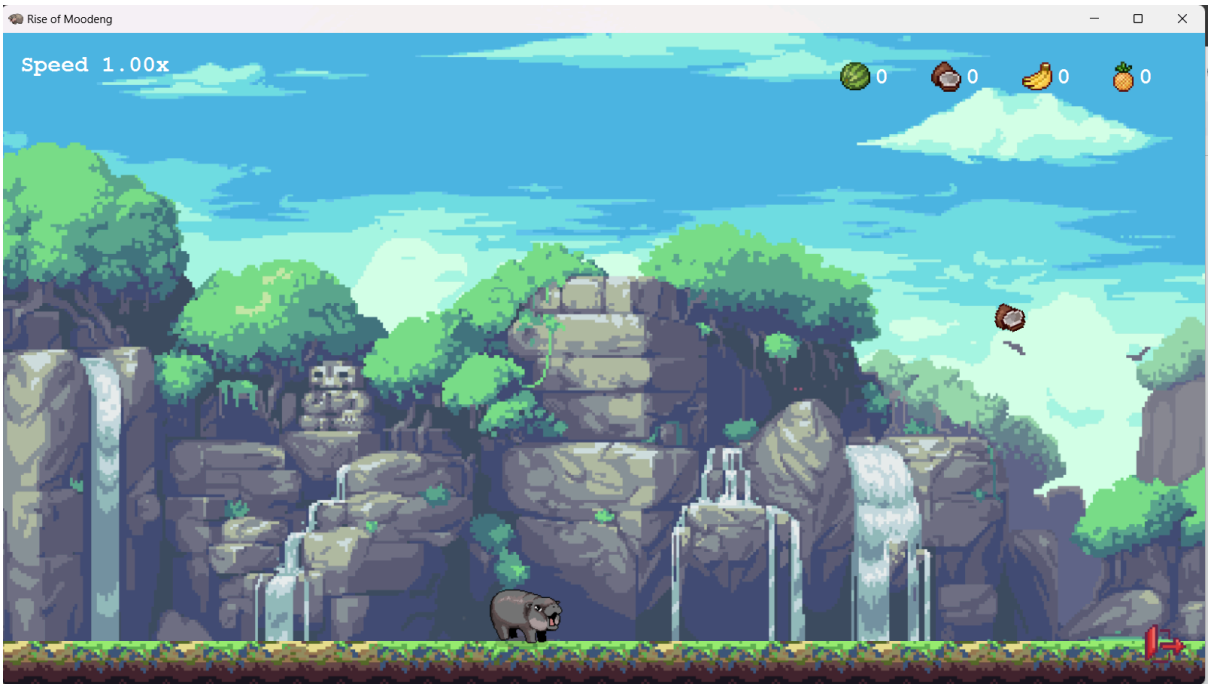
BEAST MAP Scene

Here is our BEAST map.



FOREST MAP Scene

Here is our FOREST map



JUNGLE MAP Scene

Here is our JUNGLE map

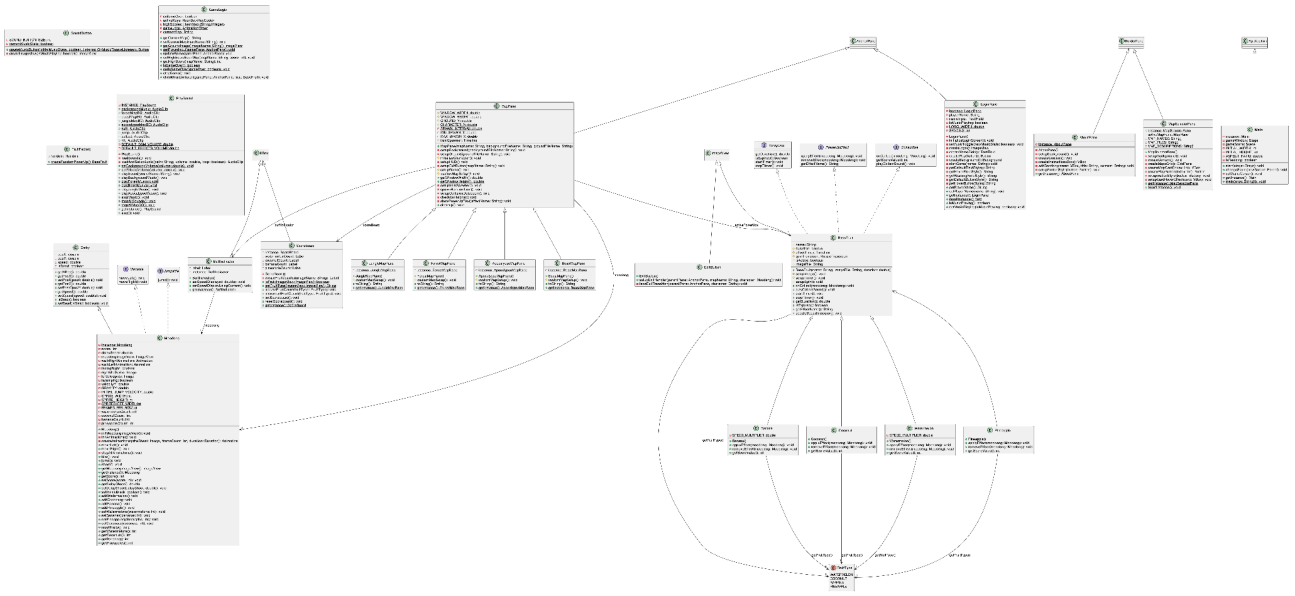
APOCALYPSE MAP Scene

Here is our APOCALYPSE map

GameEnd Scene

when you click on the red exit button on the bottom right of the page.you will go back to map selector page

Class Diagram



2. Implementation Details

To complete this project, we have this as Interfaces.

*** Note that Access Modifier Notations can be listed below**

\+ (public)

\# (protected)

\- (private)

underlined (static)

ALL_CAPS (final variable)

italic (abstract)

2.1 Package character

This package contains all types of character, including Entity class (abstract class) and interface for items.

2.1.1 abstract class Entity

This class is a generalized class of all entities in the game.

Fields

Name	**Description**
- double posX	The X position of the entity.
- double posY	The Y position of the entity.
- double speed	The speed of the entity.

Methods

Method	**Description**
+ double getXPos()	Returns the X position of the entity.
+ double getPosX()	Returns the X position of the entity (same as getXPos()).

+ void setPosX(double posX)	Sets the X position of the entity.
+ double getPosY()	Returns the Y position of the entity.
+ void setPosY(double posY)	Sets the Y position of the entity.
+ double getSpeed()	Returns the speed of the entity.
+ void setSpeed(double speed)	Sets the speed of the entity, ensuring it's not less than 0.

2.1.2 Class Moodeng extends Entity implements Movable, Jumpable

This class is a subclass of Entity. It can move and jump

Fields

Name	**Description**
- Moodeng instance	**The instance of Moodeng class**
- int score	The score of the Moodeng character.
- double delayShoot	The delay time between shooting actions.

- ImageView moodengImageView	The ImageView object used to display the Moodeng sprite.
- Animation walkRightAnimation	The animation for moving right.
- Animation walkLeftAnimation	The animation for moving left.
- boolean facingRight	Indicates if Moodeng is facing right.
- boolean isJumping	Whether Moodeng is currently jumping.
- boolean isDash	Whether Moodeng is currently dashing.
- boolean isDashOnCooldown	Indicates if dash is on cooldown.
- boolean isGravity	Enables gravity effect.
- Timeline cooldownTimeline	Timeline controlling dash cooldown display.
- long lastDashTime	Timestamp of the last dash.
- double velocityY	Vertical velocity for jumping.
- static final int DASH_COOLDOWN_MS	Cooldown duration in milliseconds.

- static final int DASH_DURATION	Duration of dash in milliseconds.
- static final int DASH_IGNORE_GRAVITY_DURATION	Gravity ignore duration after dashing.
- static final double GRAVITY	Gravity acceleration value.
- static final double INITIAL_JUMP_VELOCITY	Initial jump velocity when jumping starts.
- static final int SPRITE_WIDTH	Width of Moodeng's sprite.
- static final int SPRITE_HEIGHT	Height of Moodeng's sprite.
- static final int SPRITESHEET_WIDTH	Total width of sprite sheet.
- static final int FRAMES_PER_ROW	Number of frames in a sprite row.
- int watermelonCount	The number of watermelons collected by Moodeng.
- int coconutCount	The number of coconuts collected by Moodeng.
- int bananaCount	The number of bananas collected by Moodeng.

- int pineappleCount	The number of pineapples collected by Moodeng.

Methods

Method	**Description**
+ Moodeng()	Initializes Moodeng’s position, image, and animations.
- void initMoodengImageView()	Returns the X position of the entity (same as getXPos()).
- void initAnimations()	Loads and configures Moodeng's image.
- Animation createAnimation(Image image, int frames, Duration duration)	Creates sprite animation.
+ void moveLeft()	Moves Moodeng to the left and plays animation.
+ void moveRight()	Moves Moodeng right and plays animation.
- void stopAllAnimations()	Stops active animations and sets idle sprite.
+ void idle()	Sets Moodeng's state to idle.

+ void jump()	Makes Moodeng jump if allowed.
+ void dash()	Triggers dash action with cooldown.
+ boolean isDashAvailable()	Check if dash is available.
+ long getDashCooldownRemaining()	Returns remaining dash cooldown.
+ void shoot()	Placeholder for shooting action.

2.2 Package components

This package contains enumerators of the building.

2.2.1 Class ExitButton extends ImageView

class that extends ImageView, making it a specialized button for exiting the game.

Constructor	**Description**
+ ExitButton()	Initializes the exit button image, size, and position on the screen.

Method	Description
+ setupExitHandler(AnchorPane parentPane, String mapName, Moodeng character)	Sets up the exit button’s event handlers, including mouse click, release, enter, and exit events. Updates game logic, resets the scoreboard, and fades out the game screen.
- fadeExitTransition(AnchorPane parentPane, Entity character)	Creates a fade-out transition effect when the exit button is clicked. Stops the game, sets the game state to "Game Over," and switches back to the map selection screen.

2.2.2 Class BuffIndicator extends VBox

This class Manages the display of Moodeng's buff

Constructor	Description
+ BuffIndicator()	Initializes labels and sets up the layout.

Method	Description
+ void setSpeedDisplay(double d)	Updates the speed label with the given value.

+ void setCurrentSpeedDisplay()	Updates speed label using Moodeng's speed.
+ void setDashCooldownDisplay(Double cd)	Updates dash cooldown label with the given time.
+ void setCurrentDashCooldownDisplay()	Updates dash cooldown label based on Moodeng's cooldown status.

2.2.3 Class ScoreBoard extends HBox

This class Displays and updates the player’s collected fruits.

Fields	**Description**
- ScoreBoard instance	instance of the scoreboard.
- Label watermelonCount	Number of watermelons.
- Label coconutCount	Number of coconuts.
- Label bananaCount	Number of bananas.
- Label pineappleCount	Number of pineapples.

Constructor	**Description**
- ScoreBoard()	Initializes labels and layout.

Method	**Description**
- Label createFruitCounter(String imageName)	Creates a fruit label with an icon.
+ boolean isFruit(ImageView imageView)	Returns true if the image is a fruit.
+ String getFruitType(ImageView imageView)	Returns the fruit type by ID.
+ void updateFruitCount(FruitType fruitType)	Updates the count for the fruit type.
+ void incrementFruitCount(FruitType fruitType)	Increments the count for the fruit type.
+ void setScoreboard()	Sets the scoreboard from Moodeng's fruit counts.
+ void resetScoreboard()	Resets all fruit counts to zero.

+ ScoreBoard getInstance()	Returns the instance.

2.2.4 Class SoundButton

This class Handles the creation and functionality of a sound toggle button.

Field	**Description**
- final int SOUND_BUTTON_SIZE	Button icon size.
- boolean currentMusicState	Current state of the music.

Field	**Description**
+ Button createMusicButton(boolean initialMusicState, OnMusicToggleListener listener)	Creates a sound button with the initial state.
- ImageView createImageView(boolean isMusicPlaying)	Returns the corresponding sound icon.
+ interface OnMusicToggleListener	Called when the state changes.

2.3 Package gui

This package contains all interfaces for building.

2.3.1 class AboutPane extends BorderPane

This class is how to page.

Field	**Description**
- static AboutPane instance	instance of the AboutPane

Methods

Method	**Description**
- AboutPane()	Constructor for AboutPane. Initializes background and content.
- void setupBackground()	Sets the background of the screen. Attempts to load a GIF image; if it fails, applies a fallback gradient.
- void createContent()	Creates the content for the About screen: title, instructions, and a back button.

- VBox createInstructionsBox()	Creates and returns a VBox containing the instruction sections
- void setupButtonStyle(Button button)	Sets up the styling and hover effects for the back button.
+ static AboutPane getInstance()	Returns instance of AboutPane

2.3.2 Class ApocalypseMapPane extends MapPane

This class represents a specific map in the game: the "Apocalypse" map.

Field	**Description**
- static ApocalypseMapPane instance	instance of the AboutPane

Methods

Method	**Description**
+ ApocalypseMapPane()	Constructor for the ApocalypseMapPane. Sets up the map's background and plays the apocalypse music.

- void customMapSetup()	Custom setup for the Apocalypse map. Currently empty, can be extended in the future.
+ String toString()	Returns "ApocalypseMap"
+ static ApocalypseMapPane getInstance()	Returns instance of ApocalypseMapPane

2.3.3 Class BeastMapPane extends MapPane

This class represents a specific map in the game: the "Beast" map.

Methods

Field	**Description**
- static BeastMapPane instance	Singleton instance of the BeastMapPane class. Ensures only one instance exists.

Method	**Description**
+ BeastMapPane()	Constructor for the BeastMapPane. Sets up the map's background and plays the beast music.

- void customMapSetup()	Custom setup for the Beast map. Currently empty, can be extended in the future.
+ String toString()	Returns "BeastMap"
+ static BeastMapPane getInstance()	Returns instance of BeastMapPane.

2.3.4 Class ForestMapPane extends MapPane

This class represents a specific map in the game: the "Forest" map.

Field	**Description**
- static ForestMapPane instance	instance of the ForestMapPane class.

Method	**Description**
+ ForestMapPane()	Constructor for the ForestMapPane. Sets up the map's background and plays the forest music.
- void customMapSetup()	Custom setup for the Forest map. Currently empty, can be extended in the future.

+ String toString()	Returns "ForestMap"
+ static ForestMapPane getInstance()	Returns the instance of ForestMapPane

2.3.5 Class JungleMapPane extends MapPane

This class represents a specific map in the game: the "Jungle" map.

Field	**Description**
- static JungleMapPane instance	instance of the BeastMapPane class.

Method	**Description**
+ JungleMapPane()	Constructor for the JungleMapPane. Sets up the map's background and plays the jungle music.
- void customMapSetup()	Custom setup for the Jungle map. Currently empty, can be extended in the future.
+ String toString()	Returns the string "JungleMap"
+ static JungleMapPane getInstance()	Returns the instance of JungleMapPane

2.3.6 Class LoginPane extends AnchorPane

The class represents the login screen of the game.

Field	**Description**
- static LoginPane instance	instance of the LoginPane class.
- static String playerName	Holds the name of the player entered on the login screen.
- TextField nameInput	A TextField where the player can enter their name.
- boolean isMusicPlaying	A flag indicating whether the background music is currently playing.
- static final double LOGO_WIDTH	instance of the BeastMapPane class.
- static final int SPACING	Vertical spacing between elements in the GridPane.

Method	**Description**
+ LoginPane()	Constructor for LoginPane. Initializes the login screen with background, name field, buttons, etc.

+ void initializeLoginScreen()	Initializes the visual components of the login screen.
+ void onMusicToggle(boolean)	Handles toggling the background music on or off when the music button is pressed.
+ ImageView createLogo()	Creates the game logo and returns an ImageView displaying it.
+ TextField createNameField()	Creates the text field for the user to enter their name.
+ Button createPlayButton()	Creates the "BEGIN ADVENTURE" button.
+ Background createBackground()	Sets up the background of the login screen with a GIF.
+ void startGame(String)	Starts the game by changing the scene to MapSelectorPane and passing the player name.
+ String getDefaultFieldStyle()	Returns the default style for the name input field.
+ String getHoverFieldStyle()	Returns the style for the name input field when hovered.

+ String getWarningFieldStyle()	Returns the style for the name input field when invalid input is detected.
+ String getDefaultButtonStyle()	Returns the default style for the "BEGIN ADVENTURE" button.
+ String getHoverButtonStyle()	Returns the style for the "BEGIN ADVENTURE" button when hovered.
+ static String getPlayerName()	Returns the current player name.
+ static void setPlayerName(String)	Sets the player name and updates the TextField on the login screen.
+ static LoginPane getInstance()	Returns an instance of LoginPane.
+ static void resetInstance()	Resets the LoginPane instance and clears the player name.
+ boolean isMusicPlaying()	Returns the current state of the background music .
+ void setMusicPlaying(boolean)	Sets whether the background music is playing or not.

2.3.7 abstract class MapPane extends AnchorPane

This class is an abstract base class for different map types in the game.

Field	**Description**
# static final double WINDOW_WIDTH	The width of the game window.
# static final double WINDOW_HEIGHT	The height of the game window.
# static final double GROUND_Y	The Y-coordinate of the ground position in the window.
# static final double CHARACTER_Y	The Y-coordinate of the character's position above the ground.
- static final double SPAWN_INTERVAL	The time interval (in seconds) for spawning items.
- static final double MIN_SPAWN_X	The minimum X-coordinate for spawning items.
- static final double MAX_SPAWN_X	The maximum X-coordinate for spawning items.
# Timeline itemSpawner	The Timeline object that controls the interval of item spawning.
#List<BaseFruit> activePowerUps	List of power-ups that are active and falling on the screen.

# ScoreBoard scoreBoard	List of power-ups that are active and falling on the screen.
# Moodeng moodeng	The player character
# BuffIndicator buffIndicator	Displays active buffs.

Method	**Description**
+ MapPane(String mapName, String backgroundFileName, String groundFileName)	Constructor for MapPane. Initializes the map, background, ground, character, HUD, and sets up item spawner and collision detection.
+ void setupBackground(String backgroundFileName)	Sets up the background image of the map based on the provided filename.
+ void setupGround(String groundFileName)	Sets up the ground image and places it at the bottom of the window.
+ void initializeCharacter()	Initializes the Moodeng character and places it at the designated X and Y position on the map.
+ void setupHUD()	Sets up the ScoreBoard and places it on the screen.

+ void setupExitButton(String mapName)	Sets up an exit button and its handler to close the game or go back to the map selection screen.
+ void setupGame()	making the pane focusable and updating game logic.
+ void setupItemSpawner()	Sets up the Timeline that spawns items at intervals defined by SPAWN_INTERVAL.
+ void spawnRandomItem()	Spawns a random power-up item at a random position on the screen and animates it falling down.
+ void setupCollisionDetection()	Sets up a collision detection mechanism that checks for interactions between the character and active power-ups.
+ void checkCollisions()	Checks for collisions between Moodeng and the falling items, collecting power-ups and updating the score accordingly.
+ void showPowerUpText(String effectName)	Displays a temporary visual effect when a power-up is collected, including text and animations (fade and move).
+ void cleanup()	Stops the item spawning timeline and clears active power-ups from the screen when cleaning up the game state.

+ double getWindowWidth()	Returns the width of the game window.
+ double getWindowHeight()	Returns the height of the game window.
# abstract void customMapSetup()	set up the custom map-specific configurations.

2.3.8 MapSelectorPane extends BorderPane

This class for map selector page.

Field	**Description**
+ MapSelectorPane instance	The singleton instance of MapSelectorPane.
+ MapPane activeMapPane	The active map pane currently displayed.
- String[] MAP_NAMES	Array of map names for selection.
- String[] MAP_FILES	Array of map image filenames.
- String[] MAP_DESCRIPTIONS	Array of descriptions for each map.

Method	**Description**

+ MapSelectorPane()	Constructor to initialize the map selector and set up the background and content.
+ void setupBackground()	Sets up the background image for the map selector pane.
+ void createContent()	Creates the content layout for the map selector including buttons and map cards.
+ GridPane createMapsGrid()	Creates a grid to display the map cards.
+ VBox createMapCard(int index)	Creates a card for each map with an image, title, description, and a play button.
+ Button createPlayButton(int index)	Creates a play button for each map that switches to the selected map.
+ void setupButtonStyle(Button button)	Sets up the style for the buttons used in the map selector.
+ void setupCardHoverEffect(VBox card)	Adds hover effects to map cards for visual feedback.
+ MapSelectorPane getInstance()	Returns the instance of the MapSelectorPane.
+ void resetInstance()	Resets the instance of MapSelectorPane.

2.4 Package interface

This package contains types of interface.

2.4.1 Interface Collectible

This interface is for items that can be collected.

Methods

Method	**Description**
void onCollect(Moodeng moodeng)	Handles the logic when a collectible item is collected by Moodeng.
int getScoreValue();	Returns the score value associated with the collectible.
void playCollectSound();	Plays the sound effect when the collectible is picked up.

2.4.2 Interface Jumpable

This interface is for entities that can be jumped.

Methods

Method	**Description**

void jump()	Makes the entity perform a jump action.

2.4.3 Interface Movable

This interface is for entities that can be moved.

Methods

Method	**Description**
void moveLeft();	Makes the entity perform a move to the left action.
void moveRight();	Makes the entity perform a move to the left action.

2.4.4 Interface PowerUpEffect

This interface is for applying and removing a power-up effect on a Moodeng character

Methods

Method	**Description**
void applyEffect(Moodeng moodeng)	Applies the power-up effect to the specified Moodeng instance.

void removeEffect(Moodeng moodeng)	Removes the power-up effect from the specified Moodeng instance.
String getEffectName()	Returns the name of the effect.

2.4.5 Interface Temporal

This interface is for applying and removing a power-up effect on a Moodeng character

Methods

Method	**Description**
double getDuration()	Returns the duration of the temporal effect (in seconds or desired time unit).
boolean isExpired()	Checks if the temporal effect has expired.
void startTimer()	Start the timer for the temporal effect.
void stopTimer()	Stop the timer for the temporal effect.

2.5 Package logic

This package contains the controller of the game.

2.5.1 Class GameLogic

This class is a generalized logic in the game.

Fields

Name	**Description**
- static HashSet<KeyCode> activeKeys	Tracks currently pressed keys.
- static HashMap<String, Integer> highScores	Stores high scores for each map.
- static AnimationTimer gameLoop	Main game loop for updating game state.
- static String currentMap	Name of the current map.

Methods

Method	**Description**
+ String getCurrentMap()	Returns the name of the current map.
+ void setCurrentMap(String mapName)	Sets the name of the current map.

+ ImageView getGroundImage(String imageName)	Returns an ImageView for the background image of a given ground.
+ void getPlayerInput(AnchorPane gamePane)	Sets up key press and release events for player movement.
+ void updateGame(AnchorPane gamePane)	Starts and runs the main game loop, handling player movement and collisions.
+ void setHighScoreEachMap(String mapName, int score)	Sets the high score for a particular map.
+ int getHighScore(String mapName)	Returns the high score for a particular map.
+ void stopGame()	Stop the game loop if it is running.
+ void checkFruitCollision(AnchorPane gamePane, BaseFruit fruit)	Checks and handles fruit collision with the player.

2.6 Package main

This package contains class Main.

2.6.1 Class Main extends Application

This is the main class of the game.

Fields

Name	**Description**
- Main instance	instance of the Main class
- Stage gameWindow	The main stage of the game.
- Scene gameScene	The current scene
- int INITIAL_WIDTH	Initial width of the game to 1280
- int INITIAL_HEIGHT	Initial height of the game to 720
- double ASPECT_RATIO	Initial Aspect ratio to (16:9)
- isResizing	check resizing.

Methods

Method	**Description**
+ void start(Stage stage)	Start the application Initializes gameWindow = stage set new image to “deng.png” Initializes the gameScene with LoginPane.getInstance() and width and height to INITIAL_WIDTH and INITIAL_HEIGHT set game window min width to 1280 set game window min height to 720 set scene for game window with gameScene set title for game window to "Rise of Moodeng" app icon of game window to new image set resizable of game window to true open the stage
+ void changeScene(Parent newScreen)	set the root of the current scene to a new screen
- void setGameCursor()	set a custom cursor image to `cursor.png`
+ static Main getInstance()	Returns instance of the Main class

+ static void main(String[] args)	Launch application

2.7 Package objects

This package contains fruit that moodeng can collect .

2.7.1 class Banana extends BaseFruit

This class contains banana fruit

Field

Method	**Description**
- static final double SPEED_MULTIPLIER	The speed multiplier applied when the banana effect is active.
- double originalSpeed	The player's original speed before the banana effect is applied.

Methods

Method	**Description**
+ Banana()	Constructor for the Banana class, initializes with the name "Swift Sprint" and an image "banana.png".
+ void applyEffect(Moodeng moodeng)	Applies the banana effect by increasing the player's speed.

+ void removeEffect(Moodeng moodeng)	Removes the banana effect by resetting the player's speed.
+ int getScoreValue()	Returns the score value for collecting the banana (75 points).
+ FruitType getFruitType()	Returns the type of the fruit, which is BANANA.

2.7.2 abstract class BaseFruit extends ImageView implements Collectible, Temporal, PowerUpEffect

This abstract class represents a collectible fruit that grants a power-up or effect to the player.

Field

Method	**Description**
# name	The name of the fruit.
# duration	The duration for which the effect of the fruit will last.
# effectTimer	The timer is used to track the fruit's effect duration.
# spinTransition	The rotation animation for the fruit (used for visual spinning effect).

# isActive	A boolean indicating whether the fruit's effect is currently active.
# imageFile	The file name of the image used for the fruit.

Methods

Method	**Description**
+ BaseFruit(String name, String imageFile, double duration)	Constructor for the Banana class, initializes with the name "Swift Sprint" and an image "banana.png".
# void setupImage()	Applies the banana effect by increasing the player's speed.
# void setupTimer()	Removes the banana effect by resetting the player's speed.
# void setupSpin()	Returns the score value for collecting the banana (75 points).
+ void onCollect(Moodeng moodeng)	Returns the type of the fruit, which is BANANA.
+ void playCollectSound()	sound when the fruit is collected.

+ void startTimer()	Start the timer for the effect.
+ void stopTimer()	Stops the timer and deactivates the effect.
+ double getDuration()	Returns the duration of the effect.
+ boolean isExpired()	Returns whether the fruit's effect has expired (timer is done).
+ String getEffectName()	Returns the name of the fruit effect.
+ abstract FruitType getFruitType()	specify the fruit type.
+ void updateSpeedIndicator()	Updates the speed indicator based on the effect (used in subclasses).

2.7.3 Class Coconut extends BaseFruit

This class contains coconut fruit

Methods

Method	**Description**
+ Coconut()	Constructor that sets the name, image, and effect duration for the coconut fruit.

+ void applyEffect(Moodeng moodeng)	Removes the invincibility effect from the Moodeng.
+ void removeEffect(Moodeng moodeng)	Removes the invincibility effect from the Moodeng.
+ int getScoreValue()	Returns the score value for collecting the coconut (100 points).
+ FruitType getFruitType()	Returns the FruitType “COCONUT”

2.7.4 class Pineapple extends BaseFruit

This class contains Pineapple fruit

Methods

Method	**Description**
+ Pineapple()	Constructor that sets the name, image, and effect duration for the pineapple fruit.
+ void applyEffect(Moodeng moodeng)	In this case, the pineapple does not apply any effect to Moodeng.
+ void removeEffect(Moodeng moodeng)	No effect to remove, so this method does nothing.

+ int getScoreValue()	Returns the score value for collecting the pineapple (150 points).
+ FruitType getFruitType()	Returns FruitType “PINEAPPLE”

2.7.5 class Watermelon extends BaseFruit

This class contains watermelon fruit

Methods

Method	**Description**
+ Watermelon()	Constructor that sets the name, image, and effect duration for the watermelon fruit.
+ void applyEffect(Moodeng moodeng)	Doubles the speed of Moodeng for a certain duration. Updates the speed indicator.
+ void removeEffect(Moodeng moodeng)	Restores Moodeng's original speed, removing the effect. Updates the speed indicator.
+ int getScoreValue()	Returns the score value for collecting the watermelon (50 points).
+ FruitType getFruitType()	Returns FruitType “WATERMELON”

2.8 Package types

This package contains the type of the fruit.

2.8.1 enum FruitType

This enum is the type of the fruit. There are WATERMELON, COCONUT, BANANA and PINEAPPLE.