Dog-VS-Monkey Documentation

Created by

Amphikapha Thathong 6633287021

Jirameth Wannasiwaporn 6633033021

Thipbadee Ngamsukkasemsri 6633109021

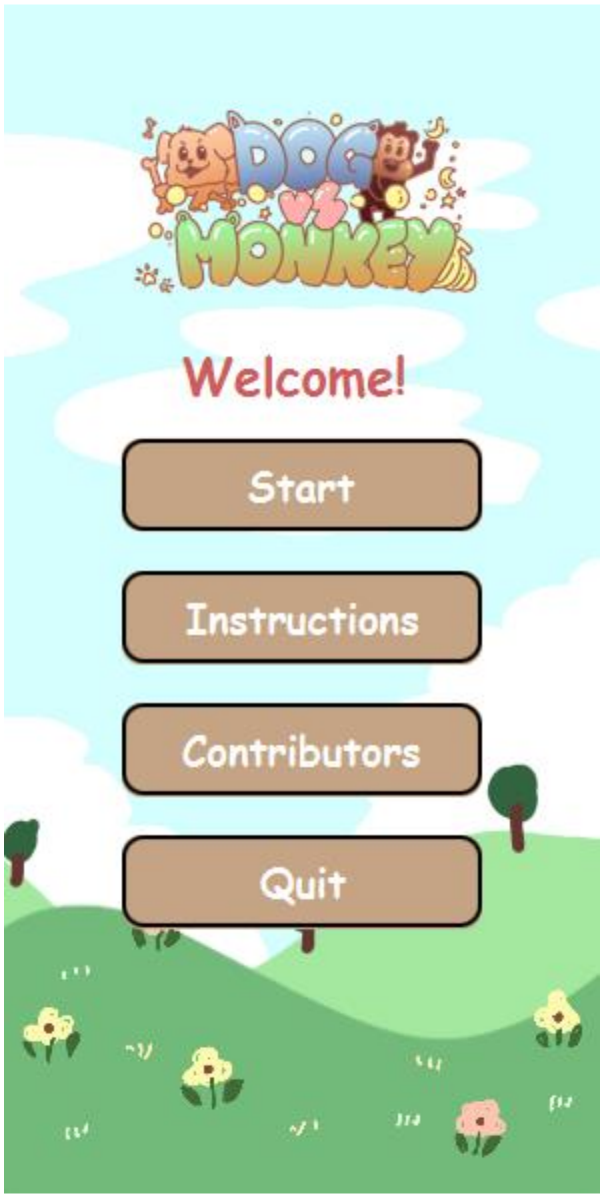
2110215 Programming Methodology

Semester 2 Year 2023

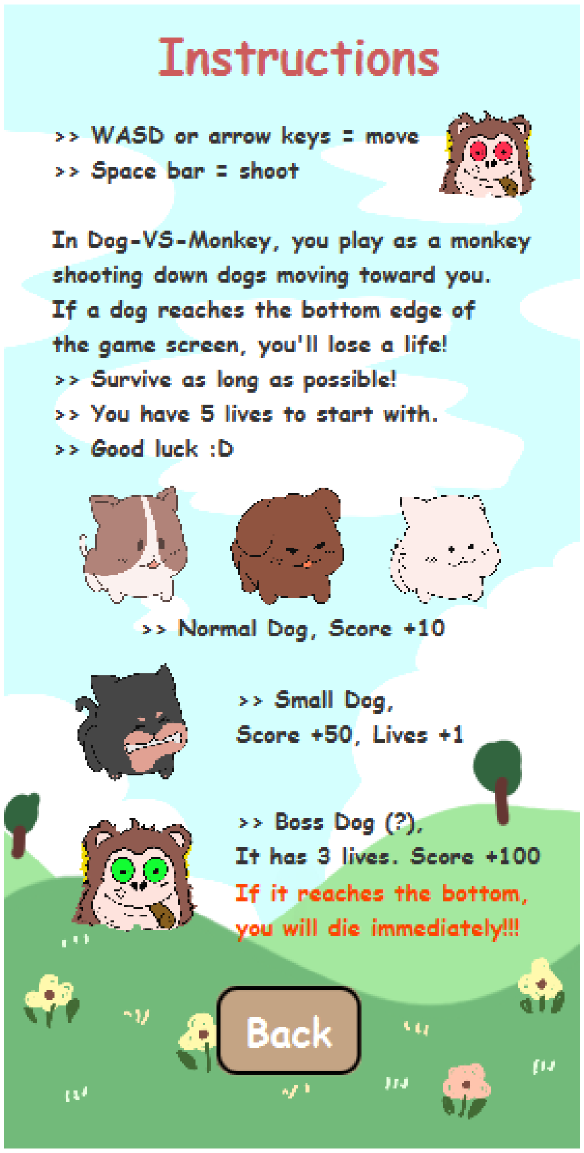
Chulalongkorn University

1.Introduction:

Dog-VS-Monkey is shooting game to overcome a dog that spawn and move down to monkey. You are a monkey in this game and you have to shoot every dogs that coming to you as soon as possible cause a dog are so cruel!!!! ≽^•ω•^≼



Start screen



Instructions

>> WASD or arrow keys = move
>> Space bar = shoot

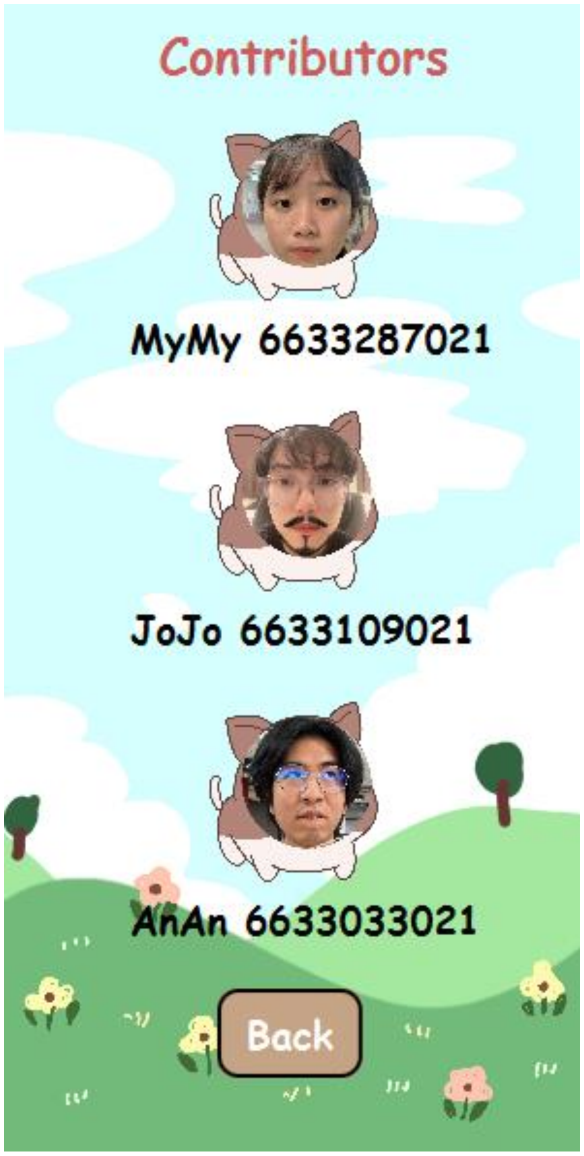
In Dog-VS-Monkey, you play as a monkey shooting down dogs moving toward you. If a dog reaches the bottom edge of the game screen, you'll lose a life!
>> Survive as long as possible!
>> You have 5 lives to start with.
>> Good luck :D

>> Normal Dog, Score +10

>> Small Dog,
Score +50, Lives +1

>> Boss Dog (?),
It has 3 lives. Score +100
If it reaches the bottom, you will die immediately!!!

Instruction



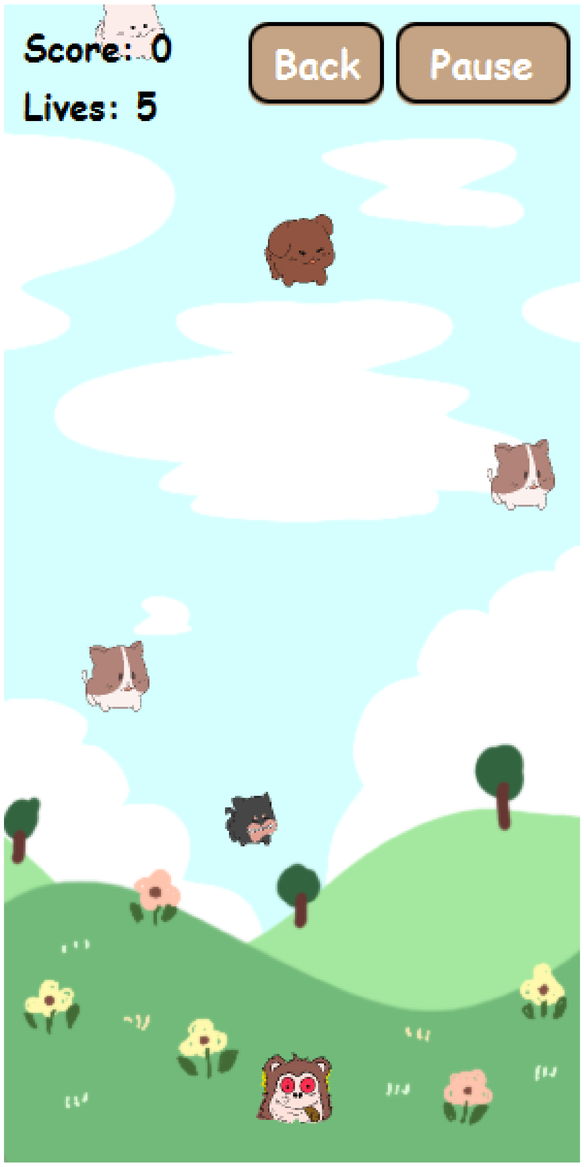
Contributors

MyMy 6633287021

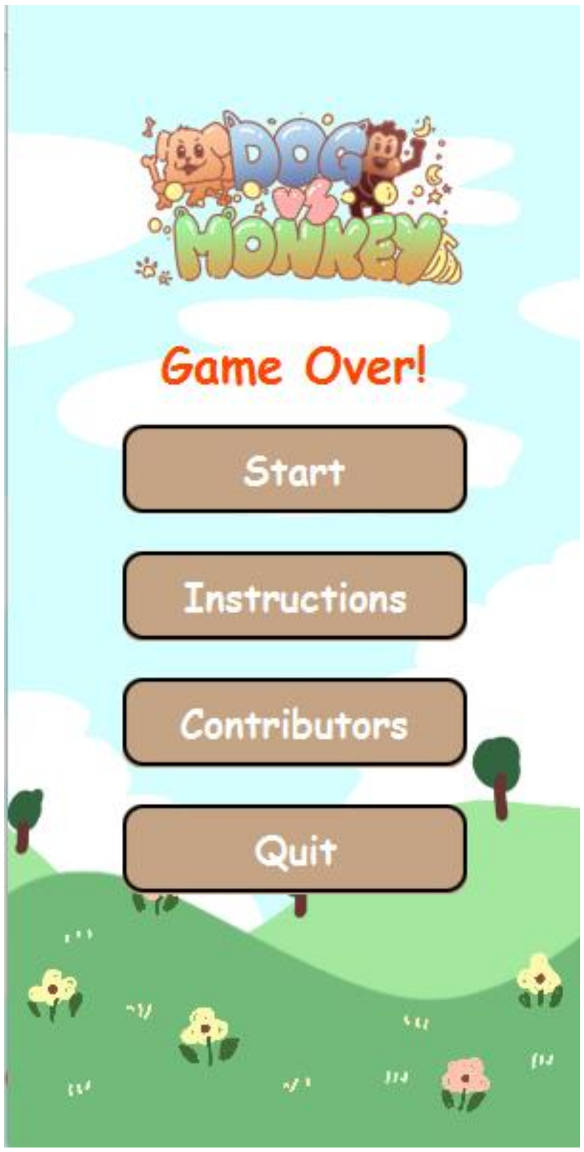
JoJo 6633109021

AnAn 6633033021

Contributors



In gameplay

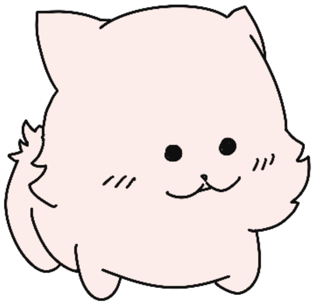


Losing screen

Characters

If a dog reaches bottom of the screen, you will lose a life, but if you shoot them, you will get the score.

1. **Normal Dog** = Score +10



2. **Small Dog** = Score +50, Lives +1

3. **Boss Dog** = Score +100, Watch out! If it reaches the bottom, you will die immediately

How to play

- Use the A, W, S, D keys or arrow keys to move the monkey.

- Press SPACE to shoot at dog.

- Avoid letting dogs reach the bottom of the screen.

- The game ends when all lives are lost and Boss Dog reaches the bottom of the screen.

- Use pause button at top-right corner to pause the game.

- Use back button to redirect to main menu.

Features

- Control a monkey using keyboard inputs (A, W, S, D, or arrow keys) to move and SPACE to shoot.

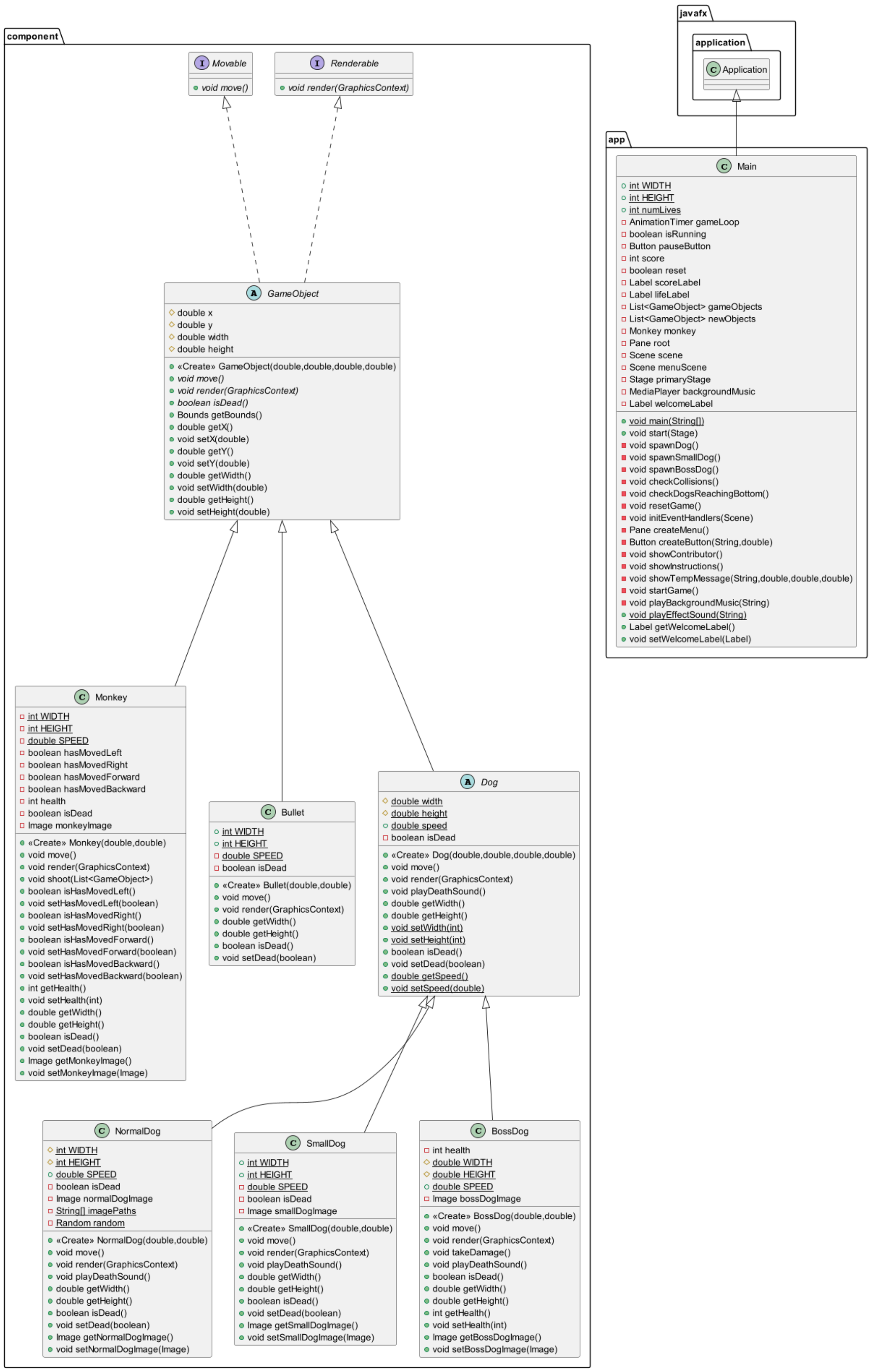
- Dogs and boss dogs spawn in different time, Normal Dog in every 1 second, Small Dog 10 second and Boss Dog when the score can mod 200.

- Score tracking and display as your score rises.

- Lives system where monkey loses a life if a dog reaches the bottom of the screen.

- Reset mechanism to start over once all lives are lost.

UML


component
Movable
void move()
Renderable
void render(GraphicsContext)
GameObject
double x
double y
double width
double height
«Create» GameObject(double,double,double,double)
void move()
void render(GraphicsContext)
boolean isDead()
Bounds getBounds()
double getX()
void setX(double)
double getY()
void setY(double)
double getWidth()
void setWidth(double)
double getHeight()
void setHeight(double)
Monkey
int WIDTH
int HEIGHT
double SPEED
boolean hasMovedLeft
boolean hasMovedRight
boolean hasMovedForward
boolean hasMovedBackward
int health
boolean isDead
Image monkeyImage
«Create» Monkey(double,double)
void move()
void render(GraphicsContext)
void shoot(List<GameObject>)
boolean isHasMovedLeft()
void setHasMovedLeft(boolean)
boolean isHasMovedRight()
void setHasMovedRight(boolean)
boolean isHasMovedForward()
void setHasMovedForward(boolean)
boolean isHasMovedBackward()
void setHasMovedBackward(boolean)
int getHealth()
void setHealth(int)
double getWidth()
double getHeight()
boolean isDead()
void setDead(boolean)
Image getMonkeyImage()
void setMonkeyImage(Image)
Bullet
int WIDTH
int HEIGHT
double SPEED
boolean isDead
«Create» Bullet(double,double)
void move()
void render(GraphicsContext)
double getWidth()
double getHeight()
boolean isDead()
void setDead(boolean)
Dog
double width
double height
double speed
boolean isDead
«Create» Dog(double,double,double,double)
void move()
void render(GraphicsContext)
void playDeathSound()
double getWidth()
double getHeight()
void setWidth(int)
void setHeight(int)
boolean isDead()
void setDead(boolean)
double getSpeed()
void setSpeed(double)
NormalDog
int WIDTH
int HEIGHT
double SPEED
boolean isDead
Image normalDogImage
String[] imagePaths
Random random
«Create» NormalDog(double,double)
void move()
void render(GraphicsContext)
void playDeathSound()
double getWidth()
double getHeight()
boolean isDead()
void setDead(boolean)
Image getNormalDogImage()
void setNormalDogImage(Image)
SmallDog
int WIDTH
int HEIGHT
double SPEED
boolean isDead
Image smallDogImage
«Create» SmallDog(double,double)
void move()
void render(GraphicsContext)
void playDeathSound()
double getWidth()
double getHeight()
boolean isDead()
void setDead(boolean)
Image getSmallDogImage()
void setSmallDogImage(Image)
BossDog
int health
double WIDTH
double HEIGHT
double SPEED
Image bossDogImage
«Create» BossDog(double,double)
void move()
void render(GraphicsContext)
void takeDamage()
void playDeathSound()
boolean isDead()
double getWidth()
double getHeight()
int getHealth()
void setHealth(int)
Image getBossDogImage()
void setBossDogImage(Image)
javafx
application
Application
app
Main
int WIDTH
int HEIGHT
int numLives
AnimationTimer gameLoop
boolean isRunning
Button pauseButton
int score
boolean reset
Label scoreLabel
Label lifeLabel
List<GameObject> gameObjects
List<GameObject> newObjects
Monkey monkey
Pane root
Scene scene
Scene menuScene
Stage primaryStage
MediaPlayer backgroundMusic
Label welcomeLabel
void main(String[])
void start(Stage)
void spawnDog()
void spawnSmallDog()
void spawnBossDog()
void checkCollisions()
void checkDogsReachingBottom()
void resetGame()
void initEventHandlers(Scene)
Pane createMenu()
Button createButton(String,double)
void showContributor()
void showInstructions()
void showTempMessage(String,double,double,double)
void startGame()
void playBackgroundMusic(String)
void playEffectSound(String)
Label getWelcomeLabel()
void setWelcomeLabel(Label)

2. Implementation Details:

* Note that Access Modifier Notations can be listed below

+ (public)

# (protected)

- (private)

underlined (static)

ALL_CAPS (final variable)

italic (abstract)

2.1. Package component

2.1.1. Interface Movable:

This interface represents objects that can be moved within the game.

Methods:

+ *void move()*	Defines a method for moving the object. Implementation provided by subclasses.

2.1.2. Interface Renderable:

This interface represents objects that can be rendered on a graphics context.

Methods:

+ *void render(GraphicsContext gc)*	Defines a abstract method for rendering the object on the provided graphics context. Implementation provided by subclasses.

2.1.3. Abstract Class GameObject implement Movable, Renderable:

This abstract class represents a generic game object in a game.

Fields:

# double x	Represents the x-coordinate of the game object.
# double y	Represents the y-coordinate of the game object.
# double width	Represents the width of the game object.
# double height	Represents the height of the game object.

Constructor:

+ GameObject(double x, double y, double width, double height)	Initializes a game object with the given position (x, y), width, and height.

Abstract Methods:

+ *void move()*	Abstract method for moving the game object. Implementation provided by subclasses.
+ *void render(GraphicsContext gc)*	Abstract method for rendering the game object. Implementation provided by subclasses.
+ *boolean isDead()*	Abstract method indicating whether the game object is dead. Implementation provided by subclasses.

Methods:

+ Bounds getBounds()	new Rectangle(x - getWidth() / 2, y - getHeight() / 2, getWidth(), getHeight()): This line creates a new Rectangle object.

	The x and y coordinates are adjusted by subtracting half of the object's width and height respectively. This is done to ensure that the rectangle is centered on the object's position (x, y). The width and height of the rectangle are set to the object's width and height obtained using getWidth() and getHeight() methods. Getting Bounds: .getBoundsInLocal(): This method retrieves the bounding box of the created Rectangle object. The bounding box represents the rectangular area that encloses the entire shape. Returning Bounds: The Bounds object representing the bounding box of the game object is returned by the method.
+ double getX()	Returns the x-coordinate of the game object.
+ void setX(double x)	Sets the x-coordinate of the game object.
+ double getY()	Returns the y-coordinate of the game object.
+ void setY(double y)	Sets the y-coordinate of the game object.
+ double getWidth()	Returns the width of the game object.
+ void setWidth(double width)	Sets the width of the game object.
+ double getHeight()	Returns the height of the game object.
+ void setHeight(double height)	Sets the height of the game object.

2.1.4. Abstract Class Dog extends GameObject:

This class represents a dog entity in a game. It is an abstract class.

Fields:

# int width	Represents the width of the dog.
# int height	Represents the height of the dog.
+ double speed	Represents the speed at which the dog moves. (value = 2)
- boolean isDead	Indicates whether the dog is dead. (value = false)

Constructor:

+ Dog(double x, double y, int width, int height)	Initializes a dog object with the given position (x, y), width, and height.

Methods:

+ *void move()*	Abstract method for moving the dog. Implementation provided by subclasses.
+ *void render(GraphicsContext gc)*	Abstract method for rendering the dog. Implementation provided by subclasses.
+ double getWidth()	Returns the width of the dog.
+ double getHeight()	Returns the height of the dog.
+ void setWidth(int width)	Sets the width of the dog.
+ void setHeight(int height)	Sets the height of the dog.
+ boolean isDead()	Returns whether the dog is dead.
+ void setDead(boolean dead)	Sets the status of the dog (dead or alive).
+ double getSpeed()	Returns the speed of the dog.
+ void setSpeed(double speed)	Sets the speed of the dog.

2.1.5. Class NormalDog extends Dog:

This class represents a normal dog entity in the game.

Fields:

# int WIDTH	Constant representing the width of the normal dog (Value = 40).
# int HEIGHT	Constant representing the height of the normal dog (Value = 40).
+ double SPEED	Constant representing the speed at which the normal dog moves (Value = 1).
- boolean isDead	Indicates whether the normal dog is dead (Value = false).
- Image normalDogImage	Represents the image of the normal dog.
- String[] imagePaths	Array of paths to different images of the normal dog. (Value = {"/pic/normalDog01.png", "/pic/normalDog02.png", "/pic/normalDog03.png"};
- Random random	Random object for selecting a random image path.

Constructor:

+ NormalDog(double x, double y)	this constructor initializes a NormalDog object with the given position(x, y) and selects a random image path from the imagePaths array to set as the image of the NormalDog object. The random image path ensures variety in the appearance of normal dogs in the game.

Methods:

+ void move()	Moves the normal dog downwards based on its speed.
+ void render(GraphicsContext gc)	this method draws the image of the NormalDog object onto the canvas at the specified coordinates (x - WIDTH / 2, y - HEIGHT / 2) with the specified dimensions (WIDTH and HEIGHT). This

	ensures that the dog is rendered centered at its position on the screen.
+ void playDeathSound()	Plays a sound effect representing the death of the NormalDog. The sound effect is retrieved from the file path "res/sound/effect/normalDog.wav". This method utilizes a class method playEffectSound() from the Main class (or associated class) to play the sound effect.
+ double getWidth()	Returns the width of the normal dog.
+ double getHeight()	Returns the height of the normal dog.
+ boolean isDead()	Returns whether the normal dog is dead.
+ void setDead(boolean dead)	-Sets the status of the NormalDog (dead or alive) based on the boolean parameter dead. -If dead is true, it sets the isDead flag to true, indicating that the NormalDog is dead. -If dead is false, it sets the isDead flag to false, indicating that the NormalDog is alive.
+ Image getNormalDogImage()	-Returns the image of the NormalDog. -This method retrieves and returns the normalDogImage, which represents the image of the NormalDog object. -The image is typically used for rendering the NormalDog on the game screen.
+ void setNormalDogImage(Image normalDogImage)	-Sets the image of the NormalDog object to the specified normalDogImage. -This method allows updating the image of the NormalDog during runtime, providing flexibility in visuals.

2.1.6. Class SmallDog extends Dog:

This class represents a small dog object in the game.

Fields:

+ int WIDTH	Width of the small dog (Value = 30).
+ int HEIGHT	Height of the small dog (Value = 30).
- double SPEED	Speed of the small dog (Value = 1).
- boolean isDead	Flag to indicate if the small dog is dead (Value = false).
- Image smallDogImage	Image of the small dog.

Constructor:

+ SmallDog(double x, double y)	-This calls the constructor of the superclass (presumably a class named GameObject) with the provided x and y coordinates, as well as the width and height specified by the WIDTH and HEIGHT constants. This initializes the position and size of the SmallDog object. -set the image of the SmallDog object. It takes an Image object as its parameter. In this case, it creates a new Image object using the file path "/pic/smallDog.png". The getClass().getResource(...) method is used to load the image file from the application's resources.

Methods:

+ void move()	Moves the small dog vertically downwards.
+ void render(GraphicsContext gc)	draw the image of the SmallDog onto the canvas at the specified coordinates (x - WIDTH / 2, y - HEIGHT / 2) with the specified dimensions (WIDTH and HEIGHT).

+ void playDeathSound()	-triggers the playing of a sound effect representing the death of the SmallDog. -"res/sound/effect/smallDog.wav": Specifies the file path of the sound effect to be played.
+ double getWidth()	Returns the width of the small dog.
+ double getHeight()	Returns the height of the small dog.
+ boolean isDead()	Returns true if the small dog is dead, false otherwise.
+ void setDead(boolean dead)	Sets the status of the small dog's life.
+ Image getSmallDogImage()	Returns the image of the small dog.
+ void setSmallDogImage(Image smallDogImage)	Sets the image of the small dog.

2.1.7. Class BossDog extends Dog:

This class represents a boss dog entity in a game.

Fields:

- int health	Represents the health points of the boss dog.
# double WIDTH	Constant representing the width of the boss dog. (value = 50)
# double HEIGHT	Constant representing the height of the boss dog. (value = 50)
+ double SPEED	Constant representing the speed at which the boss dog moves. (value = 0.5)
- Image bossDogImage	Represents the image of the boss dog.

Constructor:

+ BossDog(double x, double y)	Initializes a boss dog object with the given position (x, y), width, and height. Sets the initial health to 5 and loads the boss dog image from a specified path.

Methods:

+ void move()	Moves the boss dog downwards at a constant speed.
+ void render(GraphicsContext gc)	Variable Initialization: ratio calculates how much the boss dog's health affects its size. It's a number between 0 and 1, where 1 means full health and 0 means no health. Calculating Current Width and Height: currentWidth and currentHeight determine the size of the boss dog's image based on its health. They're scaled versions of the original width and height (WIDTH and HEIGHT), adjusted according to the boss dog's health ratio. Rendering the Image: The boss dog's image is drawn on the screen using the drawImage method. It uses the scaled width and height (currentWidth and currentHeight) to adjust the size of the image. The image is centered at the boss dog's position (x, y), considering its size.
+ void takeDamage()	Reduces the health of the boss dog by 1 when it takes damage. Sets the boss dog as dead if its health reaches 0 or below.
+ void playDeathSound()	Calling Main.playEffectSound("res/sound/effect/bossDog.wav"): This line calls a method named playEffectSound from the Main class (or an associated class) and passes it the file path "res/sound/effect/bossDog.wav".

	The method likely plays the sound effect stored in the specified file path. File Path "res/sound/effect/bossDog.wav": This is the location of the sound effect file (bossDog.wav) within the project's directory structure. It's assumed that this file contains the sound effect for the boss dog's death.
+ boolean isDead()	Checks if the boss dog is dead (health is 0 or below).
+ double getWidth()	Returns the width of the boss dog.
+ double getHeight()	Returns the height of the boss dog.
+ int getHealth()	Returns the current health of the boss dog.
+ void setHealth(int health)	Sets the health of the boss dog.
+ Image getBossDogImage()	Returns the image of the boss dog.
+ void setBossDogImage(Image bossDogImage)	Sets the image of the boss dog.

2.1.8. Class Monkey extends GameObject:

This class represents a monkey entity in a game.

Fields:

<u>- int WIDTH</u>	Constant representing the width of the monkey (Value = 40).
<u>- int HEIGHT</u>	Constant representing the height of the monkey (Value = 40).
<u>- double SPEED</u>	Constant representing the speed at which the monkey moves (Value = 3.5).
- boolean hasMovedLeft	Indicates whether the monkey has moved left.

- boolean hasMovedRight	Indicates whether the monkey has moved right.
- boolean hasMovedForward	Indicates whether the monkey has moved forward.
- boolean hasMovedBackward	Indicates whether the monkey has moved backward.
- int health	Represents the health of the monkey (Value = 20).
- boolean isDead	Indicates whether the monkey is dead (Value = false).
- Image monkeyImage	Represents the image of the monkey.

Constructor:

+ Monkey(double x, double y)	- This is a constructor method for the Monkey class, which takes x and y coordinates as parameters. - calls the constructor of the superclass, passing the x and y coordinates along with the WIDTH and HEIGHT parameters. This initializes the position and size of the monkey object. -sets the image of the monkey object. It loads an image file named "monkey_head_red.png" from the "/pic" directory relative to the class's location. getClass().getResource(...) obtains the URL of the image file as a resource. .toExternalForm() converts the URL to a string. new Image(...) creates an Image object using the URL obtained. setMonkeyImage(...) sets the loaded image as the image of the monkey object.

Methods:

+ void move()	-Checks if the monkey should move left (hasMovedLeft is true) and if moving left won't take it beyond the left boundary of the screen. setX(getX() - SPEED): If the conditions are met, it updates the monkey's x-coordinate (getX() gets the current x-coordinate) by subtracting the speed (SPEED). This moves the monkey towards the left side of the screen. -Checks if the monkey should move right (hasMovedRight is true) and if moving right won't take it beyond the right boundary of the screen (Main.WIDTH represents the width of the game window). setX(getX() + SPEED): If the conditions are met, it updates the monkey's x-coordinate by adding the speed. This moves the monkey towards the right side of the screen. -Similar to left and right movement, these conditions check whether the monkey should move forward or backward (hasMovedForward or hasMovedBackward is true), and if moving in that direction won't take it beyond the top or bottom boundary of the screen. setY(getY() - SPEED) or setY(getY() + SPEED): Updates the monkey's y-coordinate accordingly, either by subtracting or adding the speed. This moves the monkey either towards the top or bottom of the screen.
+ void render(GraphicsContext gc)	this method draws the image of the monkey object onto the canvas at the specified coordinates (x - WIDTH / 2, y -

	HEIGHT / 2) with the specified dimensions (WIDTH and HEIGHT). This ensures that the monkey is rendered centered at its position on the screen.
+ void shoot(List<GameObject> newObjects)	-creates a new Bullet object at the position of the monkey object. The x-coordinate (getX()) is set to the monkey's current x-coordinate. The y-coordinate is calculated as getY() - getHeight() / 2 - Bullet.HEIGHT. getY() retrieves the monkey's current y-coordinate. getHeight() / 2 adjusts the position vertically to ensure that the bullet originates from above the center of the monkey. Bullet.HEIGHT ensures that the bullet is positioned above the monkey's head. -adds the newly created bullet object to the list of new game objects (newObjects). This list is likely used by the game engine to manage new objects that need to be rendered and updated. - Main.playEffectSound("res/sound/effect/shooting_sound1.wav"); This line plays a sound effect representing the shooting action. It uses the playEffectSound() method from the Main class (or associated class) to play the sound effect stored in the specified file path ("res/sound/effect/shooting_sound1.wav").
+ boolean isHasMovedLeft()	Indicates if the monkey has moved left.

+ void setHasMovedLeft(Boolean hasMovedLeft)	Sets the flag indicating if the monkey has moved left.
+ boolean isHasMovedRight()	Indicates if the monkey has moved right.
+ void setHasMovedRight(boolean hasMovedRight)	Sets the flag indicating if the monkey has moved right.
+ boolean isHasMovedForward()	Indicates if the monkey has moved forward.
+ void setHasMovedForward(boolean hasMovedForward)	Sets the flag indicating if the monkey has moved forward.
+ boolean isHasMovedBackward()	Indicates if the monkey has moved backward.
+ void setHasMovedBackward(boolean hasMovedBackward)	Sets the flag indicating if the monkey has moved backward.
+ int getHealth()	Returns the health of the monkey.
+ void setHealth(int health)	Sets the health of the monkey.
+ double getWidth()	Returns the width of the monkey.
+ double getHeight()	Returns the height of the monkey.
+ boolean isDead()	Checks if the monkey is dead.
+ void setDead(boolean dead)	Sets the status of the monkey (dead or alive).
+ Image getMonkeyImage()	Returns the image of the monkey.
+ void setMonkeyImage(Image monkeyImage)	Sets the image of the monkey.

2.1.9. Class Bullet extends GameObject:

This class represents a bullet entity in a game.

Fields:

+ int WIDTH	Constant representing the width of the bullet. (value = 4)
+ int HEIGHT	Constant representing the height of the bullet. (value = 20)
- static double SPEED	Constant representing the speed at which the bullet moves. (value = 7)
- boolean isDead	Indicates whether the bullet is dead. (value = false)

Constructor:

+ Bullet(double x, double y)	Initializes a bullet object with the given position (x, y), width, and height

Methods:

+ void move()	Moves the bullet upwards at a constant speed.
+ void render(GraphicsContext gc)	Renders the bullet by drawing a yellow rectangle on the GraphicsContext.
+ double getWidth()	Returns the width of the bullet.
+ double getHeight()	Returns the height of the bullet.
+ boolean isDead()	Returns whether the bullet is dead.
+ void setDead(boolean dead)	Sets the status of the bullet (dead or alive)

2.2. Package app

2.2.1. Class Main extends Application:

This class serves as the main entry point for the game.

Fields:

<u>+ int WIDTH</u>	Declares a constant integer variable named WIDTH with a value of 300, representing the width of the game screen.
<u>+ int HEIGHT</u>	Declares a constant integer variable named HEIGHT with a value of 600, representing the height of the game screen.
<u>+ int numLives</u>	Declares an integer variable named numLives with an initial value of 20, representing the number of lives the player has.
- AnimationTimer gameLoop;	Declares a private AnimationTimer variable named gameLoop which is used for the game loop.
- boolean isRunning	Declares a private boolean variable named isRunning with an initial value of false, indicating whether the game is running.
- Button pauseButton;	Declares a private Button variable named pauseButton which is used for pausing the game.
- int score	Declares a private integer variable named score with an initial value of 0, representing the player's score.
- boolean reset	Declares a private boolean variable named reset with an initial value of false, indicating whether the game is reset.
- Label scoreLabel	Declares a private final Label variable named scoreLabel and initializes it with a Label object displaying the score. (value = new Label("Score: " + score);)
- Label lifeLabel	Declares a private final Label variable named lifeLabel and initializes it with a Label object displaying the number of lives. (value = new Label("Lives: " + numLives);)

- final List<GameObject> gameObjects	Declares a private final List variable named gameObjects which stores game objects as GameObject instances. It is initialized as an empty ArrayList. (value = new ArrayList<>();)
- final List<GameObject> newObjects	Declares a private final List variable named newObjects which stores newly created game objects as GameObject instances. It is initialized as an empty ArrayList. (value = new ArrayList<>();)
- Monkey monkey	Declares a private Monkey variable named monkey and initializes it with a new Monkey object created at a position relative to the game screen's dimensions. (value = new Monkey(WIDTH / 2, HEIGHT - 40);)
- Pane root	Declares a private Pane variable named root and initializes it with a new Pane object, serving as the root container for the game's graphical elements. (value = new Pane();)
- Scene scene	Declares a private Scene variable named scene and initializes it with a new Scene object, setting its root node to root and specifying its dimensions and background color. (value = new Scene(root, WIDTH, HEIGHT, Color.BLACK);)
- Scene menuScene;	Declares a private Scene variable named menuScene which represents the menu scene of the game.
- Stage primaryStage;	Declares a private Stage variable named primaryStage which represents the primary stage of the JavaFX application.
- MediaPlayer backgroundMusic;	Declares a private MediaPlayer variable named backgroundMusic which is used for playing background music in the game.
- Label welcomeLabel;	Declares a private Label variable named welcomeLabel which is used to display a welcome message in the game.

Methods:

+ <u>void main(String[] args)</u>	Main method to launch the JavaFX application.
+ void start(Stage primaryStage)	This method is the entry point for JavaFX applications. It is called when the application is launched, and it initializes the primary stage (main window) and sets up the game scene.
- void spawnDog()	This method is responsible for spawning a new dog enemy in the game. It randomly generates a position for the dog to appear on the screen and checks if a boss dog should be spawned instead.
- void spawnSmallDog()	This method spawns a new small dog enemy in the game. Similar to spawnDog(), it generates a random position for the small dog to appear.
- void spawnBossDog()	This method spawns a new boss dog enemy in the game. It ensures that only one boss dog exists at a time and plays a warning message when the boss dog is spawned.
- void checkCollisions()	This method checks for collisions between bullets fired by the player and the enemy dogs. It detects when a bullet hits a dog and handles the appropriate actions (e.g., scoring points).
- void checkDogsReachingBottom()	This method checks if any enemy dogs have reached the bottom edge of the game screen. If a dog reaches the bottom, it decrements the player's lives and handles game over conditions.
- void resetGame()	This method resets the game state when the player loses, clearing all game objects, resetting the score and lives, and transitioning back to the main menu.
- void initEventHandlers(Scene scene)	This method initializes event handlers for keyboard input in the game scene. It listens for key presses and releases to

	control the movement of the player character (monkey) and shooting.
- Pane createMenu()	This method creates the main menu of the game, including buttons for starting the game, viewing instructions, seeing contributors, and quitting the game.
- Button createButton(String text, double y)	This method creates a customized button for the game's user interface. It accepts two parameters: text, which is the text to be displayed on the button, and y, which specifies the vertical position of the button.
- void showContributor()	This method displays a pane with information about the contributors who worked on the game, including their names and images.
- void showInstructions()	This method displays a pane with instructions on how to play the game, including movement controls, scoring, and the types of enemy dogs encountered in the game.
- void showTempMessage(String message)	This method temporarily displays a message on the screen, typically used for brief notifications or alerts during gameplay.
- void startGame()	This method starts the game by transitioning from the main menu to the game scene, initializing necessary components, and starting the game loop.
- void playBackgroundMusic(String musicFile)	This method plays background music during the game, taking the file path of the music file as input and looping the music indefinitely.
+ void playEffectSound(String soundFile)	This method plays a sound effect during the game, such as when a dog is killed or when a boss dog appears.

+ Label getWelcomeLabel()	This method is a getter for the welcomeLabel instance variable. It returns the welcomeLabel object, allowing access to its properties and methods.
+ void setWelcomeLabel(Label welcomeLabel)	This method is a setter for the welcomeLabel instance variable. It sets the value of welcomeLabel to the provided Label object, allowing for modification.

-- END --

Git repository

https://github.com/2110215-ProgMeth/project-cedt-2023-2-dog-monkey

Youtube link

https://youtu.be/cFkica9elyI?si=v5jLOaClsH5W04gT