DANGER WORLD

Created by

6633289321	Ameen	Chebueraheng
6633078221	Daniel	Samae
6633068021	Natkamon	Maleehuan

2110215 Programming Methodology

2nd Semester, Year 2023

Chulalongkorn University

Introduction

Main menu page

Credits page

Game over page

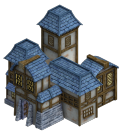
If you get hit 3 times by a ghost, you will be killed.

Win page

If you can kill Boss Ghost in Level 5 you will win!!.

How to play

 Player : You can press ⬆ , ⬇ , ▶ or ◀ to control player on the footpath and press spacebar to shoot the monsters.

 House : If you have a key you can stand at House and Press E button to go to the next level.

 Chest : You can buy a key in this Chest by pressing the E button. Which cost 50 x (current level +1) points.

 Normal Ghost : Have hp = 1, Speed = 1,If you can kill Tank Ghost you will get 10 points.

 Tank Ghost : Have hp = 2, Speed = 1, If you can kill Tank Ghost you will get 20 points.

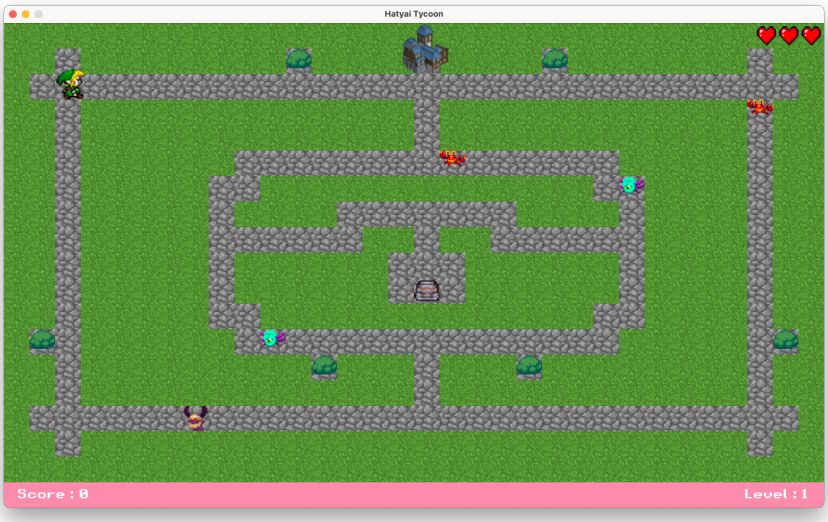
 Speed Ghost : Have hp = 1, Speed = 2, If you can kill Speed Ghost you will get 15 points.

 Boss Ghost : Have hp = 20, Speed = 1, If you can kill Boss Ghost you will win this game.

Your HP : Start game you will have 3 HP.

Bush : In the bushes, you can hide from ghosts, and they won't be able to attack you. Additionally, boss blades cannot damage you while you're hiding in the bushes.

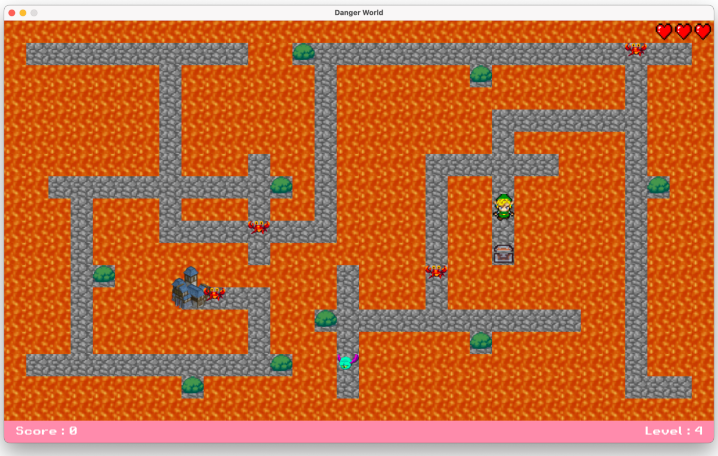
Map Level 1



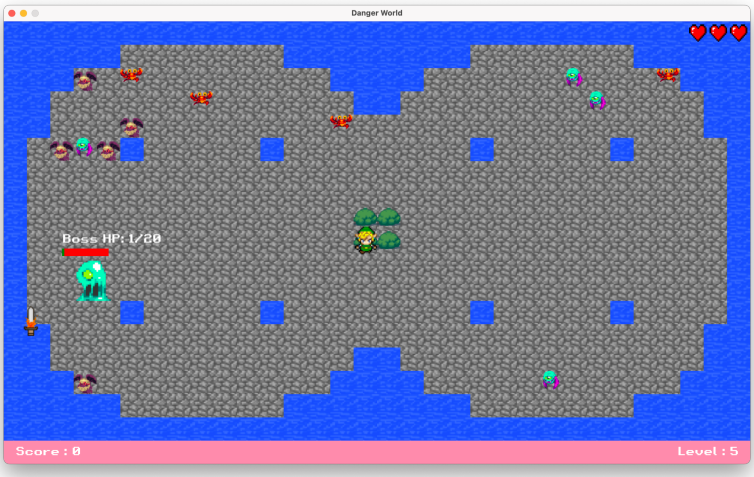
Map Level 2

Map Level 3

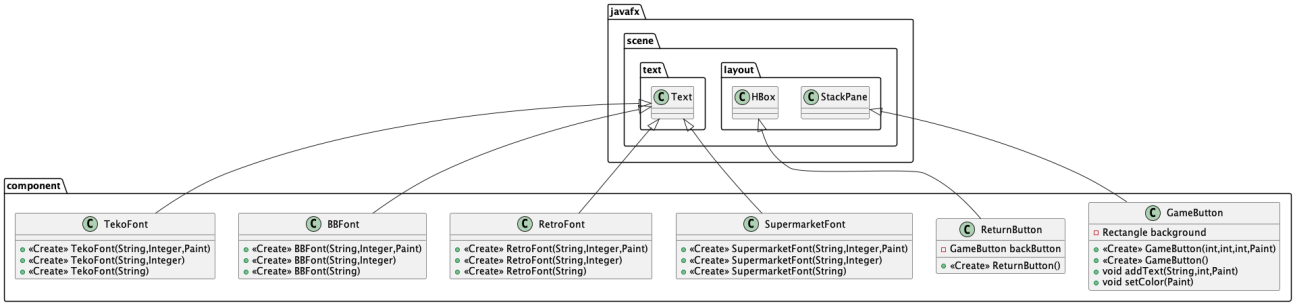
Map Level 4

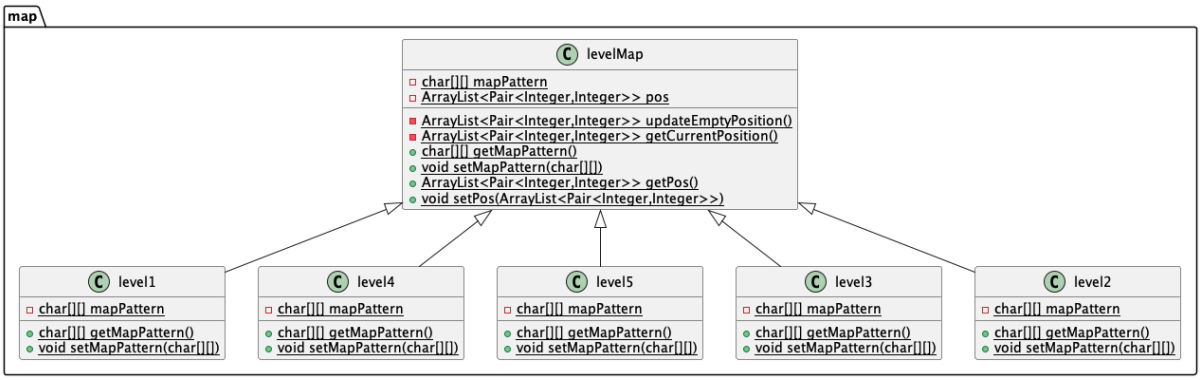


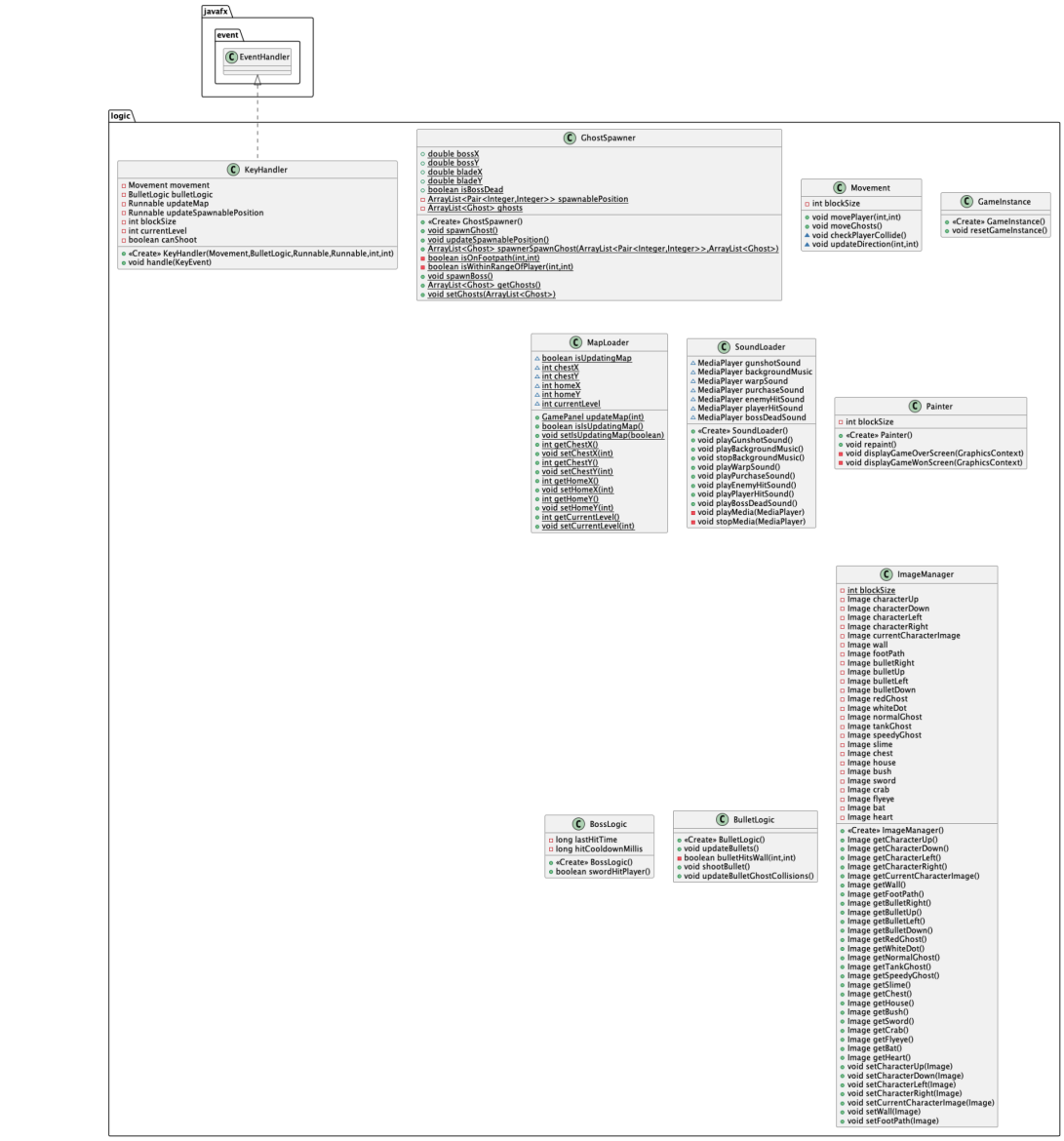
Map Level 5

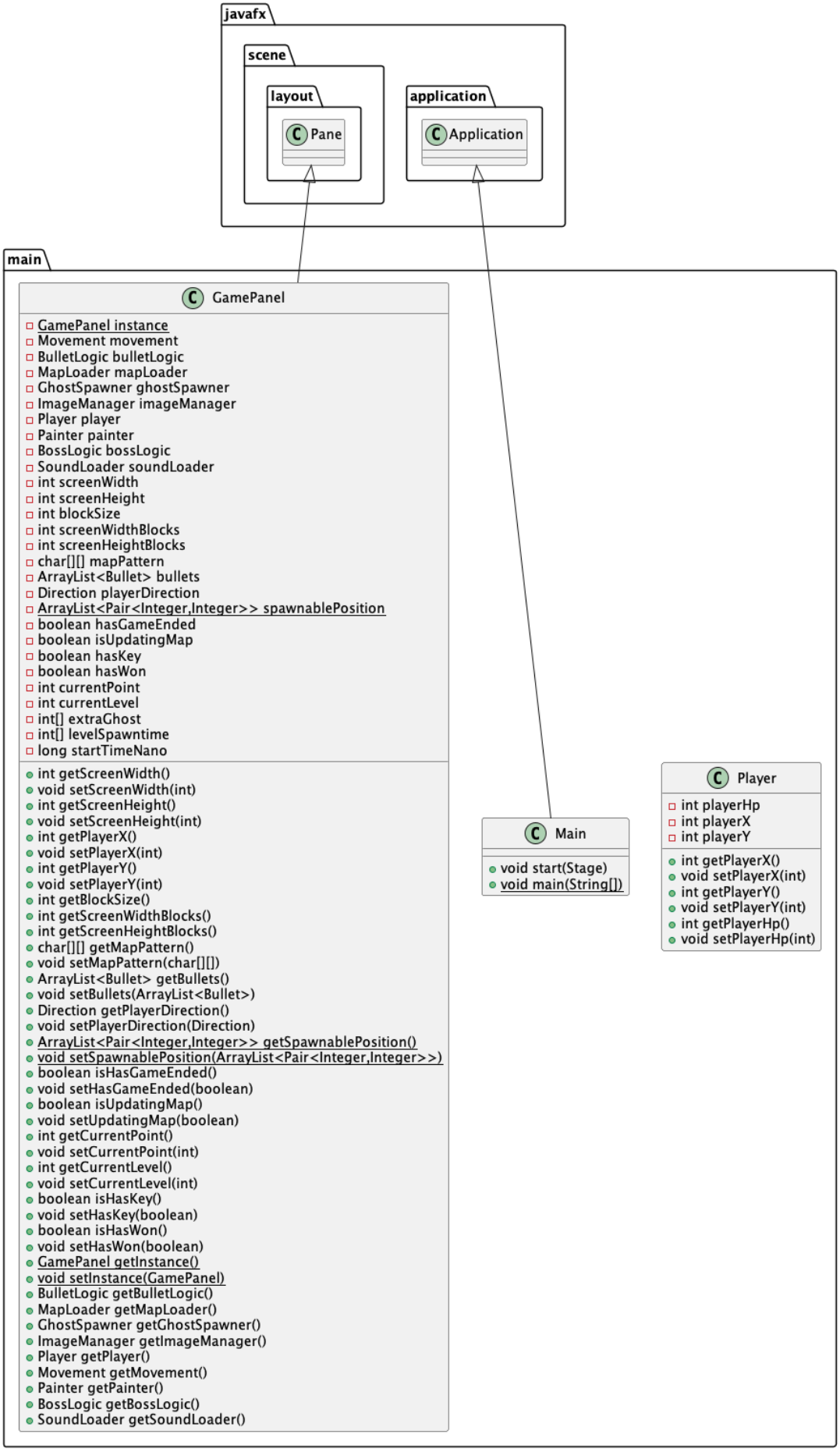


Class Diagram






javafx
event
EventHandler
logic
KeyHandler
- Movement movement
- BulletLogic bulletLogic
- Runnable updateMap
- Runnable updateSpawnablePosition
- int blockSize
- int currentLevel
- boolean canShoot
+ «Create» KeyHandler(Movement,BulletLogic,Runnable,Runnable,int,int)
+ void handle(KeyEvent)
GhostSpawner
+ double bossX
+ double bossY
+ double bladeX
+ double bladeY
+ boolean isBossDead
- ArrayList<Pair<Integer,Integer>> spawnablePosition
- ArrayList<Ghost> ghosts
+ «Create» GhostSpawner()
+ void spawnGhost()
+ void updateSpawnablePosition()
+ ArrayList<Ghost> spawnerSpawnGhost(ArrayList<Pair<Integer,Integer>>,ArrayList<Ghost>)
- boolean isOnFootpath(int,int)
- boolean isWithinRangeOfPlayer(int,int)
+ void spawnBoss()
+ ArrayList<Ghost> getGhosts()
+ void setGhosts(ArrayList<Ghost>)
Movement
- int blockSize
+ void movePlayer(int,int)
+ void moveGhosts()
+ void checkPlayerCollide()
+ void updateDirection(int,int)
GameInstance
+ «Create» GameInstance()
+ void resetGameInstance()
MapLoader
+ boolean isUpdatingMap
+ int chestX
+ int chestY
+ int homeX
+ int homeY
+ int currentLevel
+ GamePanel updateMap(int)
+ boolean isIsUpdatingMap()
+ void setIsUpdatingMap(boolean)
+ int getChestX()
+ void setChestX(int)
+ int getChestY()
+ void setChestY(int)
+ int getHomeX()
+ void setHomeX(int)
+ int getHomeY()
+ void setHomeY(int)
+ int getCurrentLevel()
+ void setCurrentLevel(int)
SoundLoader
- MediaPlayer gunshotSound
- MediaPlayer backgroundMusic
- MediaPlayer warpSound
- MediaPlayer purchaseSound
- MediaPlayer enemyHitSound
- MediaPlayer playerHitSound
- MediaPlayer bossDeadSound
+ «Create» SoundLoader()
+ void playGunshotSound()
+ void playBackgroundMusic()
+ void stopBackgroundMusic()
+ void playWarpSound()
+ void playPurchaseSound()
+ void playEnemyHitSound()
+ void playPlayerHitSound()
+ void playBossDeadSound()
+ void playMedia(MediaPlayer)
+ void stopMedia(MediaPlayer)
Painter
- int blockSize
+ «Create» Painter()
+ void repaint()
+ void displayGameOverScreen(GraphicsContext)
+ void displayGameWonScreen(GraphicsContext)
ImageManager
- int blockSize
- Image characterUp
- Image characterDown
- Image characterLeft
- Image characterRight
- Image currentCharacterImage
- Image wall
- Image footPath
- Image bulletRight
- Image bulletUp
- Image bulletLeft
- Image bulletDown
- Image redGhost
- Image whiteDot
- Image normalGhost
- Image tankGhost
- Image speedyGhost
- Image slime
- Image chest
- Image house
- Image bush
- Image sword
- Image crab
- Image flyeye
- Image bat
- Image heart
+ «Create» ImageManager()
+ Image getCharacterUp()
+ Image getCharacterDown()
+ Image getCharacterLeft()
+ Image getCharacterRight()
+ Image getCurrentCharacterImage()
+ Image getWall()
+ Image getFootPath()
+ Image getBulletRight()
+ Image getBulletUp()
+ Image getBulletLeft()
+ Image getBulletDown()
+ Image getRedGhost()
+ Image getWhiteDot()
+ Image getNormalGhost()
+ Image getTankGhost()
+ Image getSpeedyGhost()
+ Image getSlime()
+ Image getChest()
+ Image getHouse()
+ Image getBush()
+ Image getSword()
+ Image getCrab()
+ Image getFlyeye()
+ Image getBat()
+ Image getHeart()
+ void setCharacterUp(Image)
+ void setCharacterDown(Image)
+ void setCharacterLeft(Image)
+ void setCharacterRight(Image)
+ void setCurrentCharacterImage(Image)
+ void setWall(Image)
+ void setFootPath(Image)
BossLogic
- long lastHitTime
- long hitCooldownMillis
+ «Create» BossLogic()
+ boolean swordHitPlayer()
BulletLogic
+ «Create» BulletLogic()
+ void updateBullets()
- boolean bulletHitsWall(int,int)
+ void shootBullet()
+ void updateBulletGhostCollisions()
javafx
fxml
Initializable
scene
HomeScreen
- Stage stage
- Scene scene
- StackPane root
- VBox buttonContainer
- SupermarketFont title
- BBFont title2
- GameButton playButton
- GameButton tutorialButton
- GameButton creditButton
- GameButton quitButton
- BBFont titleBorder
- BBFont title3
- boolean fullScreen
- Color buttonColor
+ «Create» HomeScreen(Stage)
- void componentSetup(GameInstance)
- void eventSetup(GameInstance)
- void stageSetup()
+ StackPane getRoot()
+ Scene getScene()
StackPane
LoadingScreen
- ImageView imageView
+ void initialize(URL,ResourceBundle)
CreditScene
- VBox container
- BBFont title
- BBFont title2
- VBox titleContainer
- StackPane topContainer
- VBox textContainer
+ «Create» CreditScene()
- void textContainerSetup()
object
Bullet
- int x
- int y
- Direction direction
+ «Create» Bullet(int,int,Direction)
+ void move(int)
+ int getX()
+ int getY()
+ Direction getDirection()
+ void setX(int)
+ void setY(int)
+ void setDirection(Direction)
Direction
UP
DOWN
LEFT
RIGHT


javafx
scene
layout
Pane
application
Application
main
GamePanel
- GamePanel instance
- Movement movement
- BulletLogic bulletLogic
- MapLoader mapLoader
- GhostSpawner ghostSpawner
- ImageManager imageManager
- Player player
- Painter painter
- BossLogic bossLogic
- SoundLoader soundLoader
- int screenWidth
- int screenHeight
- int blockSize
- int screenWidthBlocks
- int screenHeightBlocks
- char[][] mapPattern
- ArrayList<Bullet> bullets
- Direction playerDirection
- ArrayList<Pair<Integer,Integer>> spawnablePosition
- boolean hasGameEnded
- boolean isUpdatingMap
- boolean hasKey
- boolean hasWon
- int currentPoint
- int currentLevel
- int[] extraGhost
- int[] levelSpawntime
- long startTimeNano
+ int getScreenWidth()
+ void setScreenWidth(int)
+ int getScreenHeight()
+ void setScreenHeight(int)
+ int getPlayerX()
+ void setPlayerX(int)
+ int getPlayerY()
+ void setPlayerY(int)
+ int getBlockSize()
+ int getScreenWidthBlocks()
+ int getScreenHeightBlocks()
+ char[][] getMapPattern()
+ void setMapPattern(char[][])
+ ArrayList<Bullet> getBullets()
+ void setBullets(ArrayList<Bullet>)
+ Direction getPlayerDirection()
+ void setPlayerDirection(Direction)
+ ArrayList<Pair<Integer,Integer>> getSpawnablePosition()
+ void setSpawnablePosition(ArrayList<Pair<Integer,Integer>>)
+ boolean isHasGameEnded()
+ void setHasGameEnded(boolean)
+ boolean isUpdatingMap()
+ void setUpdatingMap(boolean)
+ int getCurrentPoint()
+ void setCurrentPoint(int)
+ int getCurrentLevel()
+ void setCurrentLevel(int)
+ boolean isHasKey()
+ void setHasKey(boolean)
+ boolean isHasWon()
+ void setHasWon(boolean)
+ GamePanel getInstance()
+ void setInstance(GamePanel)
+ BulletLogic getBulletLogic()
+ MapLoader getMapLoader()
+ GhostSpawner getGhostSpawner()
+ ImageManager getImageManager()
+ Player getPlayer()
+ Movement getMovement()
+ Painter getPainter()
+ BossLogic getBossLogic()
+ SoundLoader getSoundLoader()
Main
+ void start(Stage)
+ void main(String[])
Player
- int playerHp
- int playerX
- int playerY
+ int getPlayerX()
+ void setPlayerX(int)
+ int getPlayerY()
+ void setPlayerY(int)
+ int getPlayerHp()
+ void setPlayerHp(int)

Implementation Detail

1. Package component

1.1 Class BBFont extends Text (javaFX)

Constructor

Name	Description
+ BBFont(String text, Integer fontSize, Paint fontColor)	Creates text with custom font size, color.
+ BBFont(String text, Integer fontSize)	Creates text with custom font size.
+ BBFont(String text)	Creates text with default font size and black color.

1.2 Class GameButton extends StackPane

Field

Name	Description
- final Rectangle background	Background of button.

Constructure

Name	Description
+ GameButton(int width, int height, int cornerRadius, Paint colorPaint)	Create buttons with custom size, rounded corners, color, and drop shadow.

+ GameButton()	Create a default button (60x60, rounded corners, white background, drop shadow).

Methods

+ void addText(String s, int size, Paint paint)	Add text to the button with specified size and colour.
+ void setColor(Paint paint)	Change button background colour after creation.

1.3 Class RetroFont extends Text

Constructure

Name	Description
+ RetroFont(String text, Integer fontSize, Paint fontColor)	Creates text with custom font size, color.
+ RetroFont(String text, Integer fontSize)	Creates text with custom font size.
+ RetroFont(String text)	Creates text with default font size and black color.

1.4 Class SupermarketFont extends Text

Constructure

Name	Description
+ SupermarketFont(String text, Integer fontSize, Paint fontColor)	Creates text with custom font size, colour.
+ SupermarketFont(String text,	Creates text with custom font size.

Integer fontSize)	
+ SupermarketFont(String text)	Creates text with default font size and black color.

1.5 Class TekoFont extends Text

Constructure

Name	Description
+ TekoFont(String text, Integer fontSize, Paint fontColor)	Creates text with custom font size, color.
+ TekoFont(String text, Integer fontSize)	Creates text with custom font size.
+ TekoFont(String text)	Creates text with default font size and black color.

1.6 ReturnButton extends HBox

Fields

Name	Description
- final GameButton backButton	Button for back to the main menu.

Constructure

Name	Description
+ ReturnButton()	Creates a reusable back button.

2. Package ghost

This package used to create different ghosts.

2.1 Class Ghost

This class is base class for each type of ghost

Field

Name	Description
- int hp	This is Ghost's hp. Hp cannot be less than 0.
- int x	Represents the horizontal position of the Ghost.
- int y	Represents the vertical position of the Ghost.
- int speed	This is Ghost's speed. Speed cannot be less than 1.
- Direction ghostDirection	Represents the direction in which the Ghost is currently facing.

Constructure

Name	Description
+ Ghost()	Initialize this ghost with following attribute set - hp to 1 - x to 1 - y to 1 - speed to 1 - ghostDirection to Direction.UP

+ Ghost(Direction dir)	Initialize this ghost with following attribute set - hp to 1 - x to 1 - y to 1 - speed to 1 - ghostDirection to dir
+ Ghost(int hp, int x, int y)	Initialize this ghost with following attribute set - hp to hp - x to x - y to y - speed to 1 - ghostDirection to Direction.UP
+ Ghost(int hp, int x, int y, int speed)	Initialize this ghost with following attribute set - hp to hp - x to x - y to y - speed to speed - ghostDirection to Direction.UP
+ Ghost(int hp, int x, int y, int speed, Direction dir)	Initialize this ghost with following attribute set - hp to hp - x to x - y to y - speed to speed - ghostDirection to dir

Method

Name	Description
+ void move(char[ ][ ] mapPattern)	This method is responsible for controlling the ghost's movement. If the next step path is incorrect Call the method changeDirection()
# boolean isValidMove(int newX, int newY, char[ ][ ] mapPattern)	To check that the next step is not a wall (Valid step).
# void changeDirection()	This method is used to find new directions for ghosts.
+ int getHp()	Return hp of Ghost.
+ void setHp(int hp)	Set hp of Ghost. If hp is less than 0 Set hp to 0.
+ int getX()	Return the horizontal position of the Ghost.
+ void setX(int x)	Set the horizontal position of the Ghost.
+ int getY()	Return the vertical position of the Ghost.
+ void setY(int y)	Set the vertical position of the Ghost.
+ int getSpeed()	Return Speed of Ghost.
+ void setSpeed(int speed)	Set speed of Ghost. If speed is less than 1 Set speed to 1.
+ Direction getGhostDirection()	Return direction of Ghost.
+ void setGhostDirection(Direction	Set direction of Ghost.

ghostDirection)	

2.2 Class NormalGhost extends Ghost

Constructure

Name	Description
+ NormalGhost()	Initialize this normal ghost with following attribute set - hp to 1 - x to 1 - y to 1 - speed to 1
+ NormalGhost(int x, int y)	Initialize this normal ghost with following attribute set - hp to 1 - x to x - y to y - speed to 1
+ NormalGhost(int x, int y, int speed)	Initialize this normal ghost with following attribute set - hp to 1 - x to x - y to y - speed to speed
+ NormalGhost(int hp, int x, int y, int speed)	Initialize this normal ghost with following attribute set - hp to hp - x to x - y to y

	- speed to speed

2.3 Class SpeedyGhost extends Ghost

Constructure

Name	Description
+ SpeedyGhost()	Initialize this speed ghost with following attribute set - hp to 1 - x to 1 - y to 1 - speed to 2
+ SpeedyGhost(int x, int y)	Initialize this speed ghost with following attribute set - hp to 1 - x to x - y to y - speed to 2
+ SpeedyGhost(int x, int y, int speed)	Initialize this speed ghost with following attribute set - hp to 1 - x to x - y to y - speed to speed
+ SpeedyGhost(int hp, int x, int y, int speed)	Initialize this speed ghost with following attribute set - hp to hp - x to x - y to y

	- speed to speed

2.4 Class TankGhost extends Ghost

Constructure

Name	Description
+ TankGhost()	Initialize this tank ghost with following attribute set - hp to 2 - x to 1 - y to 1 - speed to 1
+ TankGhost(int x, int y)	Initialize this tank ghost with following attribute set - hp to 2 - x to x - y to y - speed to 1
+ TankGhost(int x, int y, int speed)	Initialize this tank ghost with following attribute set - hp to 2 - x to x - y to y - speed to speed
+ TankGhost(int hp, int x, int y, int speed)	Initialize this tank ghost with following attribute set - hp to hp - x to x - y to y

	- speed to speed

2.5 Class BossGhost extends Ghost

Field

Name	Description
- final int radius;	Radius of circle to that being used for boss sword calculation.

Constructure

Name	Description
+ BossGhost()	Initialize this Boss ghost with following attribute set - hp to 50 - x to 1 - y to 1 - speed to 1
+ BossGhost(int hp, int x, int y, int speed)	Initialize this Boss ghost with following attribute set - hp to hp - x to x - y to y - speed to speed

Method

Name	Description
+ void move(char[ ][ ] mapPattern)	This method is responsible for controlling the ghost's movement. If the next step path is incorrect Call the method changeDirection()
+ void spinBlade()	Create a new thread to calculate boss spinning sword coordinate. Which has a delay of 0.005 seconds between each calculation. (Each 1-round rotation took 1.8 seconds) In summary, the player must kill the boss within 180 seconds (100 blade turns) or the player will lose.

3. Package logic

3.1 Class GameInstance

Constructure

Name	Description
+ GameInstance()	A constructor.

Methods

Name	Description
+ void resetGameInstance()	Reset all data to the state it was in when the game started.

	Here are things that getting resetted - Player position to (80,80) - Ghosts to new ArrayList - Bullets to new ArrayList - Boolean hasGameEnded to false - CurrentPoint to 0 - CurrentLevel to 0 - PlayerHp to 3 - Load map level 1 - PlayerDirection to Direction.RIGHT - Set current CharacterImage to CharacterRight(

3.2 Class Movement

Fields

Name	Description
+ final int blockSize	An integer represent blockSize.

Methods

Name	Description
+ void movePlayer(int dx, int dy)	This method handles the movement of the player character, calculates the new coordinates of the player, and updates if the player is moving to a valid position.
+ void moveGhosts()	This method is responsible for moving all the ghosts in the game.
- checkPlayerCollide()	This method checks for collisions between the player and any ghosts. If

	the player collides with any ghost that is not a BossGhost, the player will lose 1 HP, and the ghost will be removed from the map.
- updateDirection(int dx, int dy)	This method updates the player's facing direction based on the recent movement.

3.3 Class ImageManager

Fields

Name	Description
- final int blockSize	Size of blocks in the game.
- Image characterUp	Character up
- Image characterDown	Character down
- Image characterLeft	Character left
- Image characterRight	Character right
- Image currentCharacterImage	Current character image
- Image wall	Wall
- Image footPath	Foot path

Constructure

Name	Description
+ ImageManager()	Initialize all Images.

3.4 Class BossLogic

Field

Name	Description
- long lastHitTime	Using a long integer to calculate the last hit time prevents users from shooting bullets simultaneously.
- final long hitCoolDownMillis	Initialize a delay of 1500 milliseconds to prevent shooting conditions.

Constructor

Name	Description
+ BossLogic	Empty constructor

Methods

Name	Description
+ boolean swordHitPlayer()	This method checks if the Boss's spinning sword collides with the player. Upon collision, the player loses 1 HP and gains a 'god effect' that prevents the player from being hit by the sword again within a 1.5-second span.

3.5 Class SoundLoader

Field

Name	Description
- Media Player gunshotSound	Gunshot sound

- Media Player backgroundMusic	Background music
- Media Player warpSound	Warp sound
- Media Player purchaseSound	Purchase sound
- Media Player enemyHitSound	Enemy hit sound
- Media Player playerHitSound	Player hit sound
- Media Player bossDeadSound	Boss dead sound

Constructor

Name	Description
+ SoundLoader	Initialize all sounds.

3.6 Class BulletLogic

Constructor

Name	Description
+ BulletLogic()	Empty constructor

Methods

Name	Description
+ void updateBullets()	This method updates the bullet's position by moving the bullet toward the current direction by 1 block. If the bullet hits a wall, the bullet is removed.

	The approach involves iterating through an ArrayList of bullets
- boolean bulletHitsWall(int x, int y)	This method checks if a bullet hits a wall by examining the map pattern at the bullet's current X and Y positions.
+ void shootBullet()	The function shoots a bullet from the player's current position, If the player is not facing toward a wall. Then adds the new bullet to the list of bullets and plays a gunshotSound effect.
+ void updateBulletGhostCollisions()	This function updates bullets upon collision with a ghost by iterating through an ArrayList. If a bullet hits a ghost, the ghost's HP is reduced by 1. If the ghost's HP reaches 0, it is removed from the ArrayList.

3.7 Class GhostSpawner

Constructor

Name	Description
+ double bossX	Current boss’s X position
+ double bossY	Current boss’s Y position
+ double bladeX	Current blade’s X position
+ double bladeY	Current blade’s Y position

- ArrayList<Pair<Integer, Integer>> spawnablePosition	All possible locations where ghosts can spawn, excluding walls, chests, homes, or bushes
- ArrayList<Pair<Integer, Integer>> ghosts	Current list of living ghosts.

Constructor

Name	Description
+ GhostSpawner()	Initialize spawnablePosition and ghosts.

Methods

Name	Description
+ void spawnGhost()	This function verifies whether it is currently updating the map. If not, it proceeds to call the 'updateSpawnablePosition()' function to refresh and initiate ghost spawning.
+ void updateSpawnablePosition()	This function updates all possible locations where ghosts can spawn, limited to footpaths, and subsequently updates the 'spawnablePosition' ArrayList.
+ ArrayList<Ghost> spawnerSpawn(ArrayList<Pair<Integer,Integer>> spawnLocation, ArrayList<Ghost> currentGhosts)	This function takes in the spawnablePosition ArrayList and the current list of ghosts. It then iterates through the spawnable positions, filtering out those that are not on

	footpaths. After that, it randomly selects a footpath position and generates a random number to determine the type of ghost to spawn (NormalGhost, SpeedyGhost, or TankGhost). Finally, it creates and adds the selected ghost to the list of current ghosts and returns the updated list.
- boolean isOnFootpath(int x, int y)	This function returns true if the given coordinates (x, y) on the mapPattern represent a footpath, indicated by the character 'O'. Otherwise, it returns false.
- boolean isWithinRangeOfPlayer(int x, int y)	This function returns true if the given coordinates (x, y) on the mapPattern are within a range of 2 blocks from the current player position.
+ void spawnBoss()	This function add a ghost which is BossGhost with 20 HP, at the middle of screen with 1 speed
+ ArrayList<Ghost> getGhosts()	This function return current ghost list.
+ void setGhosts(ArrayList<Ghost> newGhosts)	Setter function for ghosts.

3.8 Class KeyHandler

Constructure

Name	Description
- Movement movement	Movement instance.

- BulletLogic bulletLogic	BulletLogic instance.
- Runnable updateMap	Runnable updateMap function.
- Runnable updateSpawnablePosition	Runnable updateSpawnablePosition function.
- int blockSize	Integer blockSize.
- int currentLevel	Integer currentLevel.
- boolean canShoot	Boolean to check can player shoot a bullet.

Methods

Name	Description
+KeyHandler(Movement movement, BulletLogic bulletLogic, Runnable updateMap, Runnable updateSpawnablePosition, int blockSize, int currentLevel) {	The constructor receives a Movement object, a BulletLogic object, two Runnable objects (updateMap and updateSpawnablePosition), an integer representing blockSize, and an integer representing currentLevel. These parameters are then assigned to their corresponding private variables.

Methods

Name	Description
+ void handle(KeyEvent event)	Handles keyboard events triggered by the player. It responds to specific key presses as follows:

	- When the player presses the arrow keys or 'W', 'A', 'S', 'D' keys, it moves the player character accordingly.
	- If the player presses the 'SPACE' key, it shoots a bullet if the shooting condition is met. It then enforces a cooldown period to prevent rapid firing.
	- Pressing the 'P' key restarts the game instance.
	- The 'E' key triggers actions such as buying keys, Go to next level, based on the player's position:
	  ● If the player is on a chest, it deducts points and grants a key if the player has enough points, and plays a purchase sound.
	  ● If the player is on a home and has a key, it resets the map and points, and plays a warp sound.
	- Pressing the 'O' key restarts the game instance when the game has ended.

3.9 Class MapLoader

Constructor

Name	Description

+ boolean isUpdatingMap	A boolean indicating whether a map update is currently in progress.
+ int chestX	An integer represent current map chest X coordinates.
+ int chestY	An integer represent current map chest Y coordinates.
+ int homeX	An integer represent current map home X coordinates.
+ int homeY	An integer represent current map home Y coordinates.
+ int currentLevel	An integer represent current level.

Methods

Name	Description
+ GamePanel updateMap(int level)	This method updates the game map based on the specified level. It sets the map pattern, updates image resources, and random positions the player character randomly within footpath areas. After updating, it increments the current level and marks the map update boolean as completed.
+ getter, setter of all fields.	getter, setter of all fields.

3.10 Class Painter

Fields

Name	Description
- int blockSize	An integer represent block size.

Constructor

Name	Description
+ Painter()	Empty constructor.

Methods

Name	Description
+ void repaint()	This methodhandles rendering the game environment on the canvas. It clears the canvas, fills it with a background color, and iterates through the map pattern, drawing walls, footpaths, chests, and bushes accordingly. Draw a ghost based on their type, Display a health bar for boss ghosts. Bullets are rendered according to their direction. Display player score and level, along with the player's character and remaining health hearts.

	Additionally, if the current level is the final level, it renders a spinning blade for the boss and handles game end screens for both “You Won” and “Game Over” scenarios.
- void displayGameOverScreen(Graphics Context gc)	This method displays a "GAME OVER" message centered on the screen and prompts the player to try again by pressing the 'O' key.
- void displayGameWonScreen(Graphics Context gc)	This method displays a "You Won" image at the center of the screen to indicate the player's victory.

4. Package main

4.1 Class GamePanel extends Pane

Fields

Name	Description
- GamePanel instance	Current game instance.
- Movement movement	Movement instance.
- BulletLogic bulletLogic	BulletLogic instance.
- MapLoader mapLoader	MapLoader instance.
- GhostSpawner ghostSpawner	GhostSpawner instance.
- ImageManager imageManager	ImageManager instance.

- Player player	Player instance.
- Painter painter	Painter instance.
- BossLogic bossLogic	BossLogic instance.
- SoundLoader soundLoader	SoundLoader instance.
- int screenWidth	An integer represent screenWidth.
- int screenHeight	An integer represent screenHeight.
- final int blockSize	An integer represent blockSize.
- final int screenWidthBlocks	An integer represent screenWidthBlock.
- final int screenHeightBlocks	An integer represent screenHeightBlock.
- char[][] mapPattern	An array of char represent current map pattern.
- ArrayList<Bullet> bullets	An ArrayList of bullet represent current bullets.
- Direction playerDirection	An Direction represent currency player direction.
- ArrayList<Pair<Integer,Integer>> spawnablePosition	Represents all possible positions where entities can spawn.
- boolean hasGameEnded	Indicates whether the game has ended.

- boolean isUpdatingMap	Indicates whether the map is currently being updated.
- boolean hasKey	Indicates whether the player has obtained a key.
- boolean hasWon	Indicates whether the player has won the game.
- int currentPoint	Represents the current score or points of the player.
- int currentLevel	Represents the current level of the game.
- final int[] extraGhost	An array of integers representing extra ghost data.
- final int[] levelSpawntime	An array of integers representing spawn time data for each level.
- long startTimeNano	Represents the start time of the game in nanoseconds.

Constructure

Name	Description
+ GamePanel()	This constructor initializes the GamePanel instance, configuring its screen width, screen height, style, and key event handling. It sets up the initial map pattern, spawns ghosts, and starts background music. The player's starting position is defined. Key

	and ghost handlers are assigned to manage user input and ghost behavior. Then run a continuous loop, controlled by AnimationTimer, updates bullet movements, collision logic, and graphic rendering. It also regulates ghost spawning and movement.

Methods

Name	Description
+ getter , setter of all fields	getter , setter of all fields.

4.2 Class Main extends Application

Methods

Name	Description
+ void start(Stage primaryStage)	To start the game.
+ void main(String[ ] args)	This method is the entry point for a JavaFX application

4.3 Class Player

Fields

Name	Description
- int playerHp	HP of player.
- int playerX	Player X position.

- int playerY	Player Y position.

Constructure

Name	Description
+ Player()	Set player HP to 3 Set X position to 2*40 Set Y position to 2*40.

Methods

Name	Description
+ int getPlayerX()	Return player X position.
+ void setPlayerX(int playerX)	Set player X position.
+ int getPlayerY()	Return player Y position.
+ void setPlayerY(int playerY)	Set player Y position.
+ int getPlayerHp()	Return HP of player.
+ void setPlayerHp(int playerHp)	Set HP of the player.

5. Package map

This package contains the pattern for generate map of each level.

5.1 Class levelMap

Fields

Name	Description
- char[ ][ ] mapPattern	Pattern of map

- ArrayList<Pair<Integer, Integer>> pos	Position.

Methods

Name	Description
- ArrayList<Pair<Integer, Integer>> updateEmptyPosition()	To update empty Position.
- ArrayList<Pair<Integer, Integer>> getCurrentPosition()	Return Current Position.
+ char[ ][ ] getMapPattern()	Return pattern of map
+ void setMapPattern(char[][] mapPattern)	Set pattern of map
+ ArrayList<Pair<Integer, Integer>> getPos()	Return Position.
+ void setPos(ArrayList<Pair<Integer, Integer>> pos)	Set position.

5.2 Class level1 extends levelMap

Fields

Name	Description
- char[ ][ ] mapPattern	The pattern of Map.

Methods

+ char[ ][ ] getMapPattern()	Return Map pattern.
+ void setMapPattern(char[][] mapPattern)	Set Map pattern.

5.3 Class level2 extends levelMap

Fields

Name	Description
- char[ ][ ] mapPattern	The pattern of Map.

Methods

+ char[ ][ ] getMapPattern()	Return Map pattern.
+ void setMapPattern(char[][] mapPattern)	Set Map pattern.

5.4 Class level3 extends levelMap

Fields

Name	Description
- char[ ][ ] mapPattern	The pattern of Map.

Methods

+ char[ ][ ] getMapPattern()	Return Map pattern.
+ void setMapPattern(char[][] mapPattern)	Set Map pattern.

5.5 Class level4 extends levelMap

Fields

Name	Description
- char[ ][ ] mapPattern	The pattern of Map.

Methods

+ char[ ][ ] getMapPattern()	Return Map pattern.
+ void setMapPattern(char[][] mapPattern)	Set Map pattern.

5.6 Class level5 extends levelMap

Fields

Name	Description
- char[ ][ ] mapPattern	The pattern of Map.

Methods

+ char[ ][ ] getMapPattern()	Return Map pattern.
+ void setMapPattern(char[][] mapPattern)	Set Map pattern.

6. Package object

6.1 Class Bullet

Fields

Name	Description
- int x	Position of bullet in X position.
- int y	Position of bullet in Y position.
- Direction direction	Direction of bullet

Constructure

Name	Description
+ Bullet(int x, int y, Direction direction)	Set x to x Set y to y Set direction to direction.

Methods

Name	Description
+ void move(int blockSize)	
+ int getX()	Return X position of bullet.
+ int setX(int x)	Set X position of bullet.
+ int getY()	Return Y position of bullet.
+ int setY(int y)	Set Y position of bullet.
+ Direction getDirection()	Return bullet direction.

+ setDirection(Direction direction)	Set bullet direction.

6.2 Enum Direction

UP,DOWN,LEFT,RIGHT

7. Package scene

7.1 class CreditScene extends StackPane

Field

Name	Description
- final VBox container	Container
- final BBFont title	Title
- final BBFont title2	Title2
- final VBox titleContainer	Title container
- final StackPane topContainer	Top container
- final VBox textContainer	Text container

Methods

Name	Description
+ CreditScene()	Set credit scene.
- void textContainerSetup()	Set text container.

7.2 class HomeScreen

Fields

Name	Description
- final Stage stage	Stage
- Scene scene	Scene
- StackPane root	Root pane
- VBox buttonContainer	Button container
- SupermarketFont title	Title
- BBFont title2	Title 2
- GameButton playButton	Play button
- GameButton tutorialButton	Tutorial button
- GameButton creditButton	Credit button
- GameButton quitButton	Quit button
- BBFont titleBorder	Title border
- BBFont title3	Title 3
- final boolean fullScreen	Set full screen by false.
- final Color buttonColor	Button color Color(82, 210, 145).

Methods

Name	Description
+ HomeScreen(Stage stage)	Set home screen.
- private void componentSetup(GameInstance gameInstance)	Set play button. Set tutorial button. Set credit button. Set quit button.
- void eventSetup(GameInstance gameInstance)	Set mouse clicked play button. Set mouse clicked tutorial button. Set mouse clicked credit button. Set mouse clicked quit button.
- void stageSetup()	Set up home screen. Set PrefSize 1280*760.
+ StackPane getRoot()	Return root.
+ Scene getScene()	Return scene.

7.3 class LoadingScreen implements Initializable

Fields

Name	Description
- ImageView	Image view

Methods

Name	Description

+ void initialize(URL location, ResourceBundle resources)	Set image loading_screen.jpg Set fit height 760. Set fit width 1280.