Final Project Documentation

Created by
Veerapat Lumsiricharoenchoke
6531343021
Sukrit Chanachaimonkonkun
6531345321

2110215 Programming Methodology
Semester 2 Year 2022
Chulalongkorn University

Minigolf game

Introduction

Welcome to our exciting 2D minigolf game! Put your precision and strategy to the test as you aim to guide the golf ball into the hole within the specified par. With a variety of unique maps to choose from, each with its own obstacles that affect the ball's movement, you'll need to navigate the courses carefully. Keep an eye on your score, as failing to meet the par will lead to defeat. Get ready for a thrilling minigolf adventure where every stroke matters. Can you conquer the challenges and achieve victory on the greens?

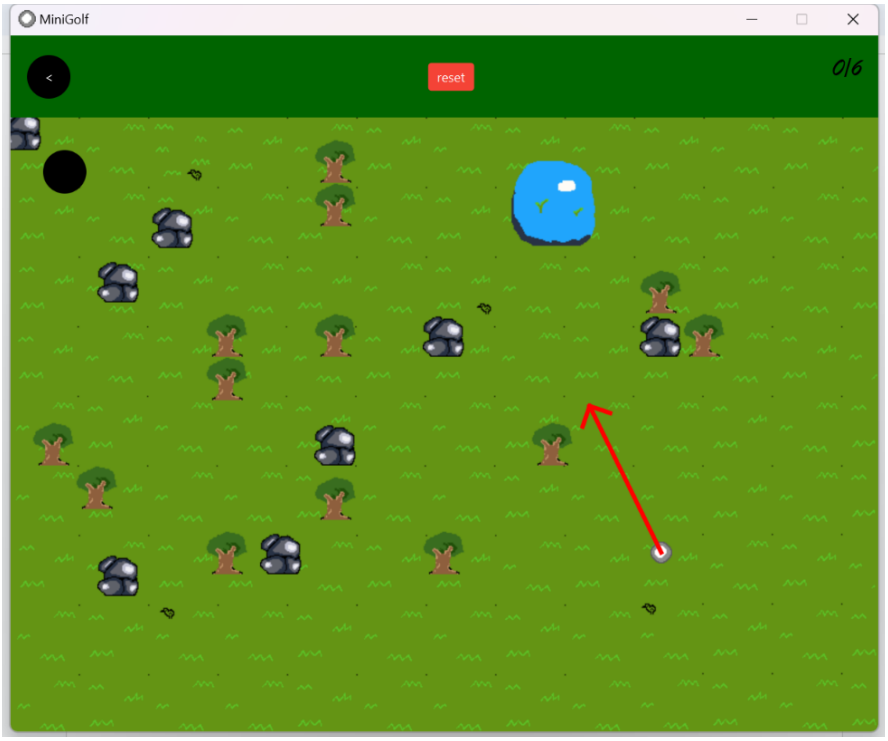
Rules

In this game, you will come across different obstacles that have special effects on the ball. Your goal is to get the ball into the hole within the specified par. If you don't reach the par, you will lose the game. Pay attention to the obstacles and use your skills to complete each level successfully.

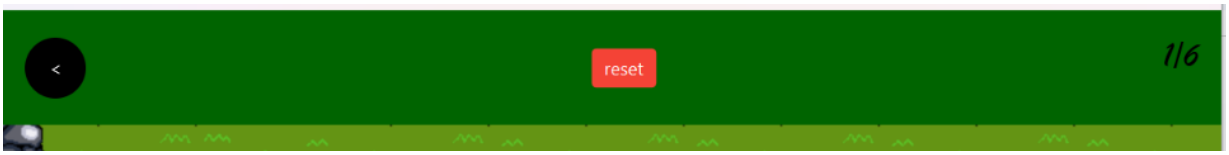
Gameplay

- game screen

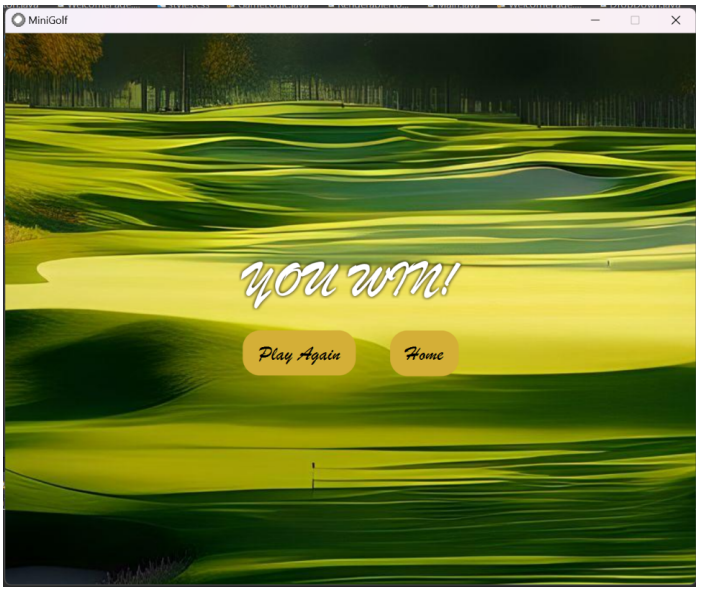
- drag the ball to the opposite direction you want the ball to go, the velocity of the ball depends on how far you drag it.

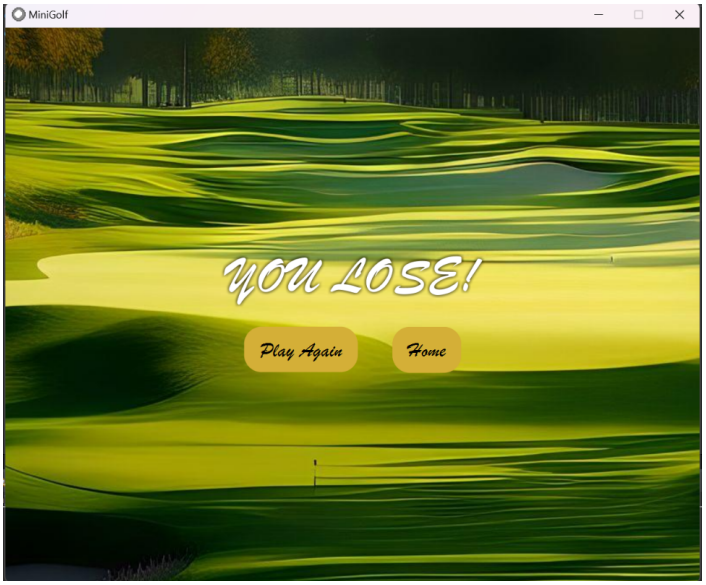


- the top of the game screen contains the back button (go back to welcome page), the reset button (reset progress of the map), shot count/ par

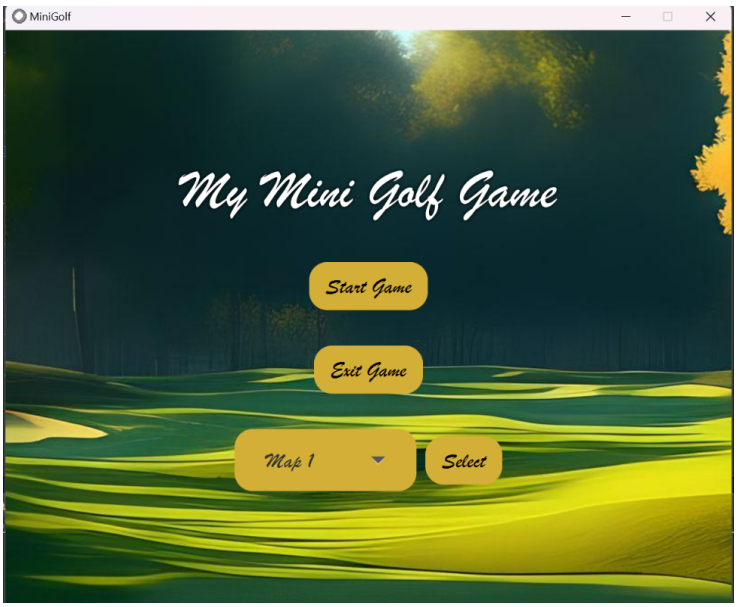


- game end scene

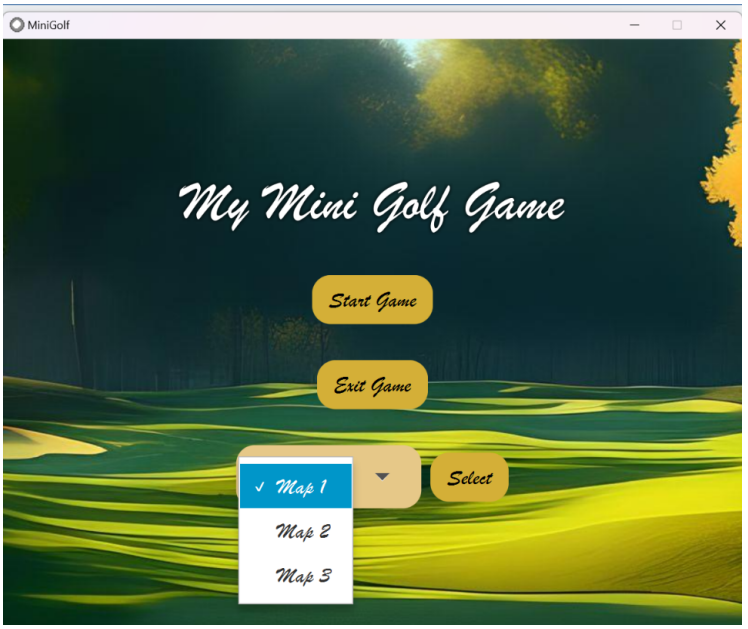




Main Menu (Welcome page)



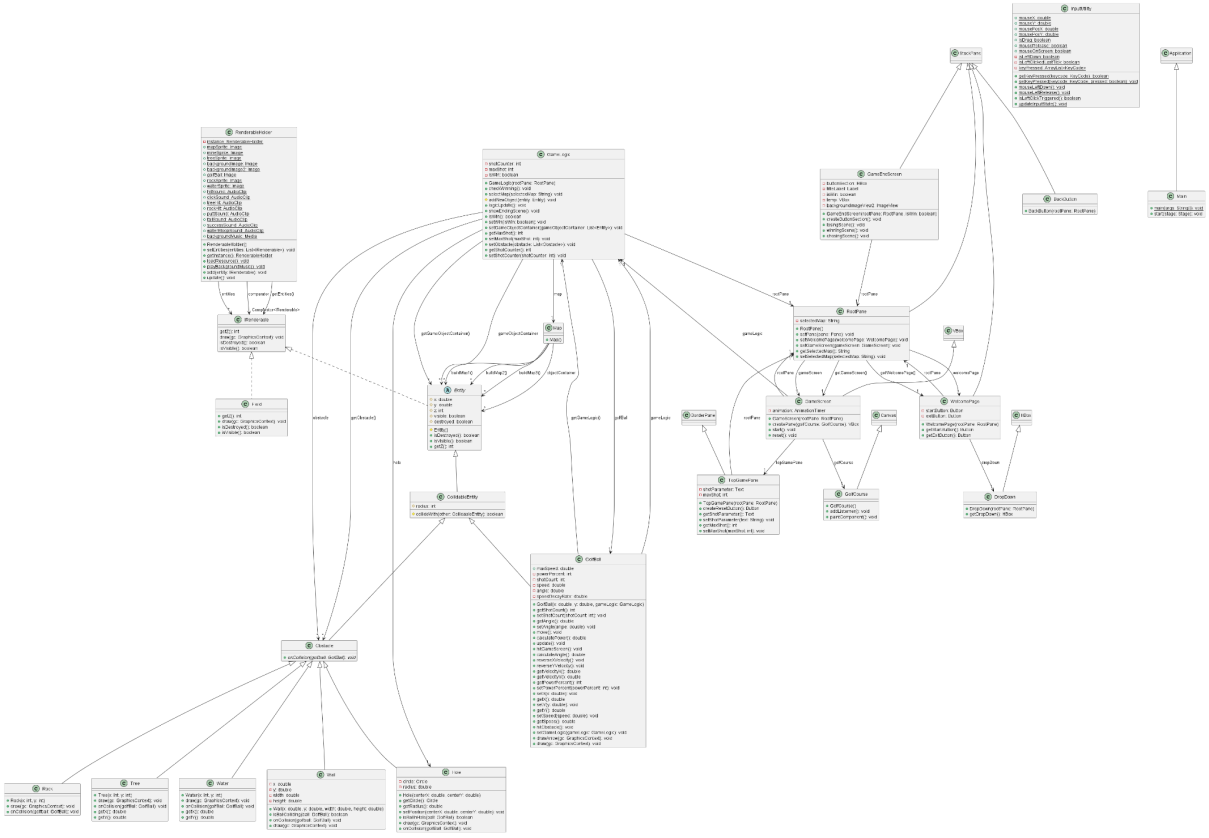
You can choose from 3 different maps and click the select button to select the map and then click start game to start.



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

+ (public)
# (protected)
- (private)
Underline (static)

Class Diagram



1. Package component

1.1. abstract class **Entity** extends **IRenderable**

1.1.1. Fields

# double x	the x coordinate of the entity
# double y	the y coordinate of the entity
# int z	the ordering of what to draw first
# boolean visible	represents visibility
# boolean destroyed	indicates whether the entity has been destroyed or not.

1.1.2. Constructor

# Entity()	sets visible to true sets destroyed to false

1.1.3. Methods

+ boolean isDestroyed()	return destroyed
+ boolean isVisible()	return visible
+ int getZ()	return z

1.2. class **Field** implements **IRenderable**

1.2.1. Methods

+ int getZ()	return -9999
+ void draw(GraphicsContext gc)	draw the field using mapSprite as background
+ boolean isDestroyed()	return false
+ boolean isVisible()	return true

1.3. class **GolfBall** extends **CollidableEntity**

1.3.1. Fields

- GameLogic gameLogic	main logic of the game
+ *final double maxSpeed*	max speed of the ball = 10
- int shotCount	player’s shot count
- double speed	speed of the ball
- double angle	angle of the ball in radian

- final double speedDecayRate	speed decay rate of the ball

1.3.2. Constructor

+ GolfBall(double x, double y, GameLogic gameLogic)	- initialize fields - set powerPercent, speed, shotCount to 0 - set radius to 10

1.3.3. Methods

+ int getShotCount()	return shotCount
+ void setShotCount(int shotCount)	set shotCount
+ double getAngle()	return angle
+ void setAngle(double angle)	set angle
+ void move()	move the ball - increase x by cos(angle)*speed - decrease y by sin(angle)*speed
+ double CalculatePower()	calculate power of the ball from mouse drag input
+ void Update()	update the speed, angle, shotCount of the ball and move the ball
+ void hitGameScreen()	reverse the velocity of the ball when the ball hit the edge of game screen by calling

	hitObstacle()
+ double calculateAngle()	calculate angle of the ball
+ void reverseXVelocity()	reverse the velocity in x axis
+ void reverseYVelocity()	reverse the velocity in y axis
+ double getVelocityX()	return the velocity in x axis
+ double getVelocityY()	return the velocity in y axis
+ void drawArrow(GraphicsContext gc)	draw the arrow in front of the ball representing the power of the ball when the ball is dragged
+ void draw(GraphicsContext)	draw the ball and when the ball is still and is dragged, call drawArrow(gc)
getters/setters of fields	

1.4. class **Hole** extends **Obstacle**

1.4.1. Fields

- Circle circle	circle representing the hole
- double radius	radius = 20

1.4.2. Constructor

+ Hole(double centerX, double centerY)	- initialize the circle with black color - z = -200

1.4.3. Methods

+ void setPosition(double centerX, double centerY)	set the position of the hole
+ boolean isBallInHole(GolfBall ball)	return true if ball is in the hole
+ void draw(GraphicsContext gc)	draw the hole
+ void onCollision(GolfBall golfball)	set speed of the ball to 30% of the original speed
getters/setters of fields	

1.5. class **Map**

1.5.1. Fields

- List<Entity> objectContainer	represents the objects and obstacles in the map

1.5.2. Constructor

+ Map()	initialize objectContainer

1.5.3. Methods

+ ArrayList<Entity> buildMap1()	initialize objects in map1 and add them to objectContainer then return objectContainer
+ ArrayList<Entity> buildMap2()	initialize objects in map2 and add them to objectContainer then return objectContainer
+ ArrayList<Entity> buildMap3()	initialize objects in map3 and add them to objectContainer then return objectContainer

1.6. abstract class **Obstacle** extends **CollidableEntity**

1.6.1. Methods

+ abstract void onCollision(GolfBall golfBall)	method to handle ball collision

1.7. class **Rock** extends **Obstacle**

1.7.1. Constructor

+ Rock(int x, int y)	set x, y set z = -100 set radius = 20

1.7.2. Methods

+ void draw(GraphicsContext gc)	draw the rock
+ void onCollision(GolfBall golfBall)	call hitObstacle() and play sound

1.8. class **Tree** extends **Obstacle**

1.8.1. Constructor

+ Tree(int x, int y)	set x, y set z = -100 set radius = 20

1.8.2. Methods

+ void draw(GraphicsContext gc)	draw the tree
+ void onCollision(GolfBall golfBall)	- call hitObstacle() and play sound - set destroyed to true
getters/setters	

1.9. class **Wall** extends **Obstacle**

1.9.1. Fields

- double x	the x coordinate of the wall

- double y	the y coordinate of the wall
- double width	width of the wall
- double height	height of the wall

1.9.2. Constructor

+ Wall(double x, double y, double width, double height)	set fields

1.9.3. Methods

+ boolean isBallColliding(GolfBall ball)	checks if the ball is colliding with the wall
+ void onCollision(GolfBall golfBall)	handle ball collision by reverse the velocity of the axis that the ball hit the wall
+ void draw(GraphicsContext gc)	draw the wall

1.10. class **Water** extends **Obstacle**

1.10.1. Constructor

+ Water(int x, int y)	- set x, y - set z = -100 - set radius = 40

1.10.2. Methods

+ void draw(GraphicsContext gc)	draw the water
+ void onCollision(GolfBall golfBall)	- the player loses when the ball touch the water - play sound
getters/setters	

2. Package input

2.1. class **InputUtility**

2.1.1. Fields

+ double mouseX	MousePosition at X (anytime)
+ double mouseY	MousePosition at Y (anytime)
+ double mousePosX	MousePosition at X (when clicked)
+ double mousePosY	MousePosition at Y (when clicked)
+ boolean isDrag	Check whether the users are dragging the mouse
+ boolean mouseRelease	Check if the mouse is released
+ mouseOnScreen	Check if the mouse is on screen
- boolean isLeftDown	Left-Click
- boolean	Check Whether the last frame

isLeftClickedLastTick	was clicked
- ArrayList<KeyCode> keyPressed	Store the input Key

2.1.2. Methods

+ boolean getKeyPressed(KeyCode keyCode)	Get the keys that are pressed
+ void setKeyPressed(KeyCode keyCode, boolean pressed)	Add the pressed key to list and print it in terminal frame by frame
+ void mouseLeftDown()	- set ifLeftDown to true - set isLeftClickedLastTick to true
+ void mouseLeftRelease()	set isLeftDown to false
+ boolean isLeftClickTriggered()	- return isLeftClickedLastTick
+ void updateInputState()	set isLeftClickedLastTick to false

3. Package logic

3.1. abstract class **CollidableEntity** extends **Entity**

3.1.1. Fields

# int radius	radius of the entity

3.1.2. Methods

# boolean collideWith(CollidableEntity other)	check if this entity collides with “other”

3.2. class GameLogic

3.2.1. Fields

- List<Entity> gameObjectContainer	list of entities in the map
- List<Obstacle> obstacle	list of obstacles in the map
- int shotCounter	shot count
- int maxShot	par
- boolean isWin	win state
- GolfBall golfBall	golf ball
- Hole hole	hole
- RootPane rootPane	the root pane
- final Map map	the map

3.2.2. Constructor

+ GameLogic(RootPane rootPane)	- set rootPane - initialize new Field - set isWin to true - shotCounter = 0 - add field to

	RenderableHolder - initialize gameObjectContainer and obstacle - call selectMap() to initialize map to the map selected

3.2.3. Methods

+ void checkWinning()	set isWin to false when shotCount exceeds maxShot
+ void selectMap(String selectedMap)	initializes objects and entities in the map selected
# void addNewObject(Entity entity)	add entity to gameObjectContainer and RenderableHolder instance and if the entity is an obstacle, add it to obstacle
+ void logicUpdate()	update the state of the game and check if the game has ended and if it has, show end screen
+ void showEndingScene()	shows ending scene
getters/setters	

4. Package main

4.1. class **Main** extends **Application**

4.1.1. Methods

+ void main(String[] args)	main application
+ void start(Stage stage)	initialize fields, show stage and play music

5. Package pane

5.1. class **BackButton** extends **StackPane**

5.1.1. Constructor

+ BackButton(RootPane rootPane)	set scene to WelcomePage scene when clicked

5.2. class **DropDown** extends **HBox**

5.2.1. Constructor

+ DropDown(RootPane rootPane)	- initialize a choiceBox with map1, map2, map3 for player to choose from - add the select button to select the map

5.2.2. Methods

+ HBox getDropDown()	return the DropDown

5.3. class GameEndScreen extends StackPane

5.3.1. Fields

- RootPane rootPane	the root pane
- HBox buttonSection	the “play again” and “home” button
- Label titleLabel	the text “YOU WIN” or “YOU LOSE”
- boolean isWin	win state
- VBox temp	titleLabel and buttonSection
- ImageView backgroundImageView2	background image

5.3.2. Constructor

+ GameEndScreen(RootPane rootPane, boolean isWin)	- initialize fields and style

5.3.3. Methods

+ void createButtonSection()	- initialize the ButtonSection HBox - initialize “play again” button and “home” button and add it to the HBox
+ void losingScene()	set titleLabel and add all components
+ void winningScene()	set titleLabel and add all components

+ void chosingScene()	decides which scene to use

5.4. class **GameScreen** extends **VBox**

5.4.1. Fields

- GameLogic gameLogic	the main game logic
- GolfCourse golfCourse	GolfCourse
- TopGamePane topGamePane	TopGamePane
- RootPane rootPane	the root pane
- AnimationTimer animation	animationTimer

5.4.2. Constructor

+ GameScreen(RootPane rootPane)	initialize fields and style

5.4.3. Methods

+ VBox createPane(GolfCourse golfCourse)	add TopGamePane and GolfCourse to a VBox and return it
+ void start()	start the AnimationTimer and set rootPane to GameScreen
+ void reset()	Stop the AnimationTimer and reinitialize all field of

	GameScreen

5.5. class **GolfCourse** extends **Canvas**

5.5.1. Constructor

+ GolfCourse()	initialize fields and add listener

5.5.2. Methods

+ void addListener()	Listener of the canvas
+ void paintComponent()	draw entities in the golf course

5.6. class **RootPane** extends **StackPane**

5.6.1. Fields

- WelcomePage welcomePage	welcome page
- GameScreen gameScreen	game screen
- String selectedMap	the name of the map that player selected

5.6.2. Constructor

+ RootPane()	initialize welcomePage and gameScreen and show welcomePage

5.6.3. Methods

+ void setPane(Pane pane)	clear the stage and show the pane
getters/setters	

5.7. class **TopGamePane** extends **BorderPane**

5.7.1. Fields

- Text shotParameter	a shotCount/maxShot text ex. 2/6
- RootPane rootPane	the root pane
- int maxShot	par

5.7.2. Constructor

+ TopGamePane(RootPane rootPane)	- initialize fields - initialize back button and reset button and shot parameter and add them to the pane

5.7.3. Methods

+ Button createResetButton()	create a button that reset progress of the map
+ Text setShotParameter(String text)	return the shotParameter in the correct format
getters/setters	

5.8. class **WelcomePage** extends **StackPane**

5.8.1. Fields

- Button startButton	a button to start the game
- Button exitButton	a button to exit the game
- DropDown dropDown	map selector ChoiceBox and the select button
- RootPane rootPane	the root pane

5.8.2. Constructor

+ WelcomePage(RootPane rootPane)	- Initialize fields and add background image and style

5.8.3. Methods

getters/setters	

6. Package sharedObject

6.1. interface **IRenderable**

6.1.1. Methods

+ int getZ()	get the draw ordering of each object
+ void draw(GraphicsContext gc)	draw the renderable entity
+ boolean isDestroyed	return destroyed
+ boolean isVisible()	return visibility

6.2. class **RenderableHolder**

6.2.1. Fields

- List<IRenderable> entities	List of all entities in game
- Comparator<IRenderable> comparator	Use to compare among the IRendeable to tell which one is to draw first
+ Image mapSprite	Image of map's background
+ Image treeSprite	Image of tree
+ Image rockSprite	Image of rock
+ Image waterSprite	Image of water
+ Image backgroundImage	Background image used at the welcomeScreen
+ Image backgroundImage2	Background image used at the GameEndScreen

+ Image golfBall	Image of golfball
+ Audio hitSound	Golf's hit sound
+ Audio clickSound	Click sound
+ Audio treeHit	Tree when collide sound
+ Audio rockHit	Rock when collide sound
+ Audio puttSound	Golf get putt in the hole sound
+ Audio failSound	Failing Sound
+ Audio successSound	Sucess Sound
+ Audio waterBloopSound	Water Bloop sound
+ int gameWidth	gameWidth = 800 px
+ int gameHeight	gameHeight = 640 px
+ int topGameHeight	TopGamePane's height = 75 px

+ int golfCourseHeight	golfCourseHeight = 575 px
+ Media backgroundMusic	Media of background Music

6.2.2. Constructor

+ RenderableHolder()	initialize array list entities and comparator

6.2.3. Methods

+ RenderableHolder	Instance of RenderableHolder

getInstance()	to make sure that there is only one copy
+ void loadResource()	Load all resource at the beginning of the game
+ void playBackgroundMusic()	plays background music
+ void add(IRenderable entity)	Add objects to the List entities
+ void update()	Remove destroyed objects
getters/setters	Getters/Setters