AE2DMS-CW-20215538

"Zixiang Hu" 20215538 scyzh6@nottingham.edu.cn

> word count: 500

Refactor:

- Replace swing with Javafx.
- Split clumsy class to achieve single responsibility. Split InteractableWorld.java to GameScene.java(model) and GameSceneController.java(controller). Where GameScene.java encapsulate all the sub-models exist in the game like hero, enemy. It also provides public methods like getGameScenePainter().paintComponent() for controller to invoke.
- Apply MVC pattern. Categorize the files into three main packages: model, controller and util. Put the view(fxml) part in resources folder. Split the game's view into three parts: menu, gameScene & gameOverScene, each part have its own FXML and controller.
- Extract multi-level abstraction. Extract abstract class like GameObject.java, WallObject.java, SpriteObject.java. Other concrete class would extends these abstract class, thus get rid of duplicate codes.
- Encapsulate fields. Encapsulate crucial fields of game object with getter and setter, to have a better protection and encapsulation.

Feature:

- Complicated setting page. Player could select theme, difficulty and volume. With different difficulty, the chance to shoot, speed and size of enemy would vary.
- Randomly dropped fruit with collect effects and sounds. When enemy killed, random kind of fruit drops. Collect fruit pop up hint tells the bonus it worth.
- Game scene equipped with level hint, score hint, time hint, boss remaining life hint and hero charging status hint.
- Multi-level with final level's boss. The boss can withstand more shoots, and shoots a different kind of bubble.
- Animation between the switch of scene. Click subpage in menu, the subpage would gently slide from the bottom to the top. Click back button in game scene, the background would blur with the confirm page pop up. All these are implemented by nested FXML with nested controller.
- No actual buttons/checkBox/choiceBox used, all simulated by imageView combined with animation and sounds.

Design pattern:

- Singleton pattern: GamePanel.java, the director of the game, would only be constructed once and its instance would be returned by getInstance() as a handle.
- FlyWeight pattern: When creating WallObjects, it gets wall's image from WallImageFactory. Factory would check if the required image already exist, and if exist, return that image. It guarantees the image

is created only once and shared by all wallObjects, which would reduce the amount of duplicate memory.
- Template pattern: Abstract class SpriteObject.java defines a template for all sprite object to extends. It implements method that is duplicate for all the sprite objects like turnAround(). For method that differs, like collideWithCeiling() the implementation detail is down to each subclass.
- Factory pattern: BossDropFruitFactory.java and EnemyDropFruitFactory.java are two fruit factories that extends FruitFactory.java. They create and return different kind of fruits according to requirements.
- Strategy pattern: Interface CollideStrategy.java is implemented by CollideWithWall, CollideWithCeiling & CollideWithFloor. Each WallObject contains a specific collide strategy, which decides how it collide with other object.
- Mediator Pattern: GameSceneController.java act as the mediator of GameScene.java and GameScene.fxml

Git work flow:

When an issue is raised, a new branch would be created to handle . When solved, branch would be merge back.

Commit message follows AngularJS Git Commit Message Conventions.

Test case:

com.ae2dms.util

Test ID	Purpose of test	Input	Expect Outcome	Pass/Fail
1.1.1 - GameRecorderTest	test if returns highest score	<"Zixiang Hu", 200, 106> <"Bryan", 360, 60><"Yu Heng", 300, 54>	360	Pass
1.1.2 - GameRecorderTest	test if reads records from file	<"Zixiang Hu", 200, 106> <"Bryan", 360, 60><"Yu Heng", 300, 54>	<"Bryan", 360, 60> <"Yu Heng", 300, 54> <"Zixiang Hu", 200, 106>	Pass
1.1.3 - GameRecorderTest	test if sort the records with score	<"Zixiang Hu", 200, 106> <"Bryan", 360, 60><"Yu Heng", 300, 54>	<"Bryan", 360, 60> <"Yu Heng", 300, 54> <"Zixiang Hu", 200, 106>	Pass
1.1.4 - GameRecorderTest	test if returns the name list	<"Zixiang Hu", 200, 106> <"Bryan", 360, 60><"Yu Heng", 300, 54>	<"Bryan", "Yu Heng", "Zixiang Hu">	Pass
1.1.5 - GameRecorderTest	test if it save records to file	<"Bub", 500, 50>	<"Bub", 500, 50>	Pass

Test ID	Purpose of test	Input	Expect Outcome	Pass/Fail
1.1.6 - GameRecorderTest	test if it returns score list	<"Zixiang Hu", 200, 106> <"Bryan", 360, 60> <"Yu Heng", 300, 54> <"Bub", 500, 50>	<500, 360, 300, 200>	Pass
1.1.7 - GameRecorderTest	test if it returns time consumed	<"Zixiang Hu", 200, 106> <"Bryan", 360, 60> <"Yu Heng", 300, 54> <"Bub", 500, 50>	<50, 60, 54, 106>	Pass
1.1.8 - GameRecorderTest	test if returns the number of records	<"Zixiang Hu", 200, 106> <"Bryan", 360, 60> <"Yu Heng", 300, 54>	3	Pass
1.2.1 - GameTimerTest	test if parse time to correct format	200, 0, 600	"03:20", "00:00", "10:00"	Pass
1.3.1 - MapReaderTest	test if correctly read map	Map one	the arraylist of objects read from map is not empty	Pass
1.3.2 - MapReaderTest	test if correctly read map	Map two	the arraylist of objects read from map is not empty	Pass
1.3.3 - MapReaderTest	test if correctly read map	Map three	the arraylist of objects read from map is not empty	Pass

com.ae2dms.controller

Test ID	Purpose of test	Input	Expect Outcome	Pass/Fail
2.1.1 - GameSceneControllerTest	test if the level hint, charge status, current score correctly displayed	null	element correctly displayed	Pass
2.2.1 - MenuControllerTest	test if buttons inside information page could be clicked and display animation	null	element correctly displayed	Pass
2.2.1 - MenuControllerTest	test if buttons inside setting page could be clicked and correspondly modify the model	null	element correctly displayed	Pass

Test ID	Purpose of test	Input	Expect Outcome	Pass/Fail
2.2.1 - MenuControllerTest	test if could click into highscore page and back from that page	null	element correctly displayed	Pass

com.ae2dms.GamePanelTest

Test ID	Purpose of test	Input	Expect Outcome	Pass/Fail
3.1 - GamePanelTest	test if loadHelper loads fxml and set the fxml as scene's root	null	the scene is not null	Pass
3.2 - GamePanelTest	test if switch scene to menu	null	switch scene to menu	Pass
3.3 - GamePanelTest	test if switch scene to gameScene	null	switch scene to gameScene	Pass
3.4 - GamePanelTest	test if switch scene to gameOverScene	null	switch scene to gameOverScene	Pass
3.5 - GamePanelTest	test if switch scene to highScoreScene	null	switch scene to highScoreScene	Pass
3.6 - GamePanelTest	test if the bonus is incremented	0	50	Pass

com.ae2dms.model.gameObject.sprite

Test ID	Purpose of test	Input	Expect Outcome	Pass/Fail
4.1.1-BossTest	test if boss get attacked when collide with projectile	null	damage to boss increase 1	Pass
4.1.2-BossTest	when boss die, test if switch game status to win and if drops fruit	null	game status switch to win and drop the specified fruit	Pass
4.1.3-BossTest	test if this method correctly return whether boss is bubbled	null	return true when boss is bubbled	Pass
4.1.4-BossTest	test if boss shoots projectile	null	the array of boss projectile is not null	Pass
4.2.1 - BossDropFruitFactoryTest	test if return the specified fruit	null	return the specified type of fruit	Pass
4.3.1 - EnemyDropFruitFactoryTest	test if return the specified fruit	null	return the specified type of fruit	Pass

Test ID	Purpose of test	Input	Expect Outcome	Pass/Fail
4.4.1 - BossProjectileTest	test if hero dies when boss's projectile collide with hero	null	hero die and game lose	Pass
4.5.1 - CollectEffectTest	test if collect effect's inner time counter decrease with time pass by	null	with each call of update() counter minus 1	Pass
4.5.2 - CollectEffectTest	test if collect effect's constructor returns collect effects	null	the returned collect effect is not null	Pass