Engineering 1
Group Assessment 2
Implementation Document

Cohort 2 Team 17
Gnocchi Games

Implementation Explanation

GameState

Participates towards UR_SAVE.

A new class has been added, named GameState. Its purpose is to record all the variables pertaining to the current state of the game, in a serializable way. It achieves this by utilizing an internal, private class: SerializableGameObject. This class is used to have a serializable version of a game entity (Boat, Obstacle, Powerup).

GameState is created by passing in all the lists of entities to be saved, along with different variables pertaining to the state of the competition. The lists are then looped over, creating a SerializableGameObject for each in-game entity.

Finally, GameState has methods to get the lists of game entities that were saved within it. It does so by looping over its stored SerializableGameObjects, and recreating the corresponding entity with the stored characteristics.

As a conclusion, GameState is used to create save points, and restore from them. To save, a GameState instance is serialized to a string and written to disk; to restore, a serialized GameState instance is read from disk and subsequently deserialized. These two processes happen in the SceneMainGame saveGame() and restoreGame() methods.

Load Screen

Participates towards UR_SAVE

A new class SceneLoadScreen has been created. It allows to load, and delete save states (saved using GameState class). It has 4 flags stored as fields, which are used in the Screen's update logic to know if the game should switch screens, reinitialize the screen, load a save state or delete the selected state. It uses a SelectBox, provided by libGDX, to select between save states, stored in an Preferences object (used to save things to disk in libGDX).

Save Screen

Participates towards UR_SAVE

A new class SceneLoadScreen has been created. It allows one to save the current state of the game, or go back to the current game (at the same state). Similarly to SceneLoadScreen, it uses flags to know if it should save the game using the selected save name, or go back to the game screen. It uses a TextArea, provided by libGDX, to input the save name, to be stored in a Preferences object (same reason as above).

Difficulty Level

Participates towards UR_DIFFICULTY_LEVEL

A new class was added called SceneDifficulty inheriting from the Scene class . This gives a screen that gives the user three buttons to pick their difficulty level, which is stored in the variable diffLevel satisfying requirement UR_DIFFICUILTY_LEVEL. Function setDiff sets changes the players boats stats according to its parameter diffLevel satisfying requirement FR_PLAYER_DIFFICULTY. This function setPlayerSpec in SceneMainGame was changed to setPlayerStats and it now calls both setSpec and setDiff. This is then called in PixelBoat.

Power-Ups

Participates towards UR_POWER_UP

The PowerUp class implements the architecture of the game, inheriting from MoveableObject and works like the Obstacle class in that it constructs an object with a texture and bounds and has a hasCollided method to be called when hit. There are 5 power-ups inheriting from the PowerUp class; Speed, Stamina, Health, Drag and Rotation each with their own class that apply a texture and collision bounds to check if they are collided with. These power-ups are added to the scene in the BoatRace class and collisions between the player and an individual powerup is detected in the Boat class where their effects are applied.

The PowerUp class and its children were all designed to satisfy the requirements in particular UR_POWER_UP which states “During a game users must be able to pick up power-up packs that will increase certain characteristics.” This code fulfills the requirements fully.

AwardsScreen and GameOverScreen

Participates towards UR_AWARDS

Two new classes, SceneAwardScreen and SceneGameOver both allow the player to restart the game.

A new method in PlayerBoat called hasDied() returns true when the player's durability stat reaches a zero value. Then SceneGameOver is called.

In SceneResultsScreen's Update() function a new process now decides whether the player came first place and calls SceneGameOver or SceneAwardScreen as needed.

In SceneMainGame's Update() a new process disqualifies the player by calling SceneGameOver if they do not place in the top 3 in the 2nd and 3rd leg's combined times.

Trial Leg

Participates towards UR_TRIAL_LEG

Function getBestTime was changed to not include the first time in the list legTimes fulfilling the requirement UR_TRIAL leg.

Unimplemented features:

UR_AWARDS - The player does not get a gold, silver or bronze medal for finishing in 1st, 2nd or 3rd respectively.

UR_BOAT_SPEC - All boat types have the same stats.

Both of these were of low priority to implement hence why we didn't have time to implement as it took longer than expected to implement the save and load features.

Source code and jar file can be found through this link:
https://github.com/Frinksy/PixelBoat/releases/tag/v1.2