Module Code: CS2JA16

Assignment report Title: Java GUI final Version Report

Student Number (e.g. 25098635): 27016428

Date (when the work completed): 21/01/20

Actual hrs spent for the assignment: 36

Assignment evaluation (3 key points):

· Learned how to create collision effect

· Learned how to create buttons in Java

· Learned lots of different libraries

Abstract

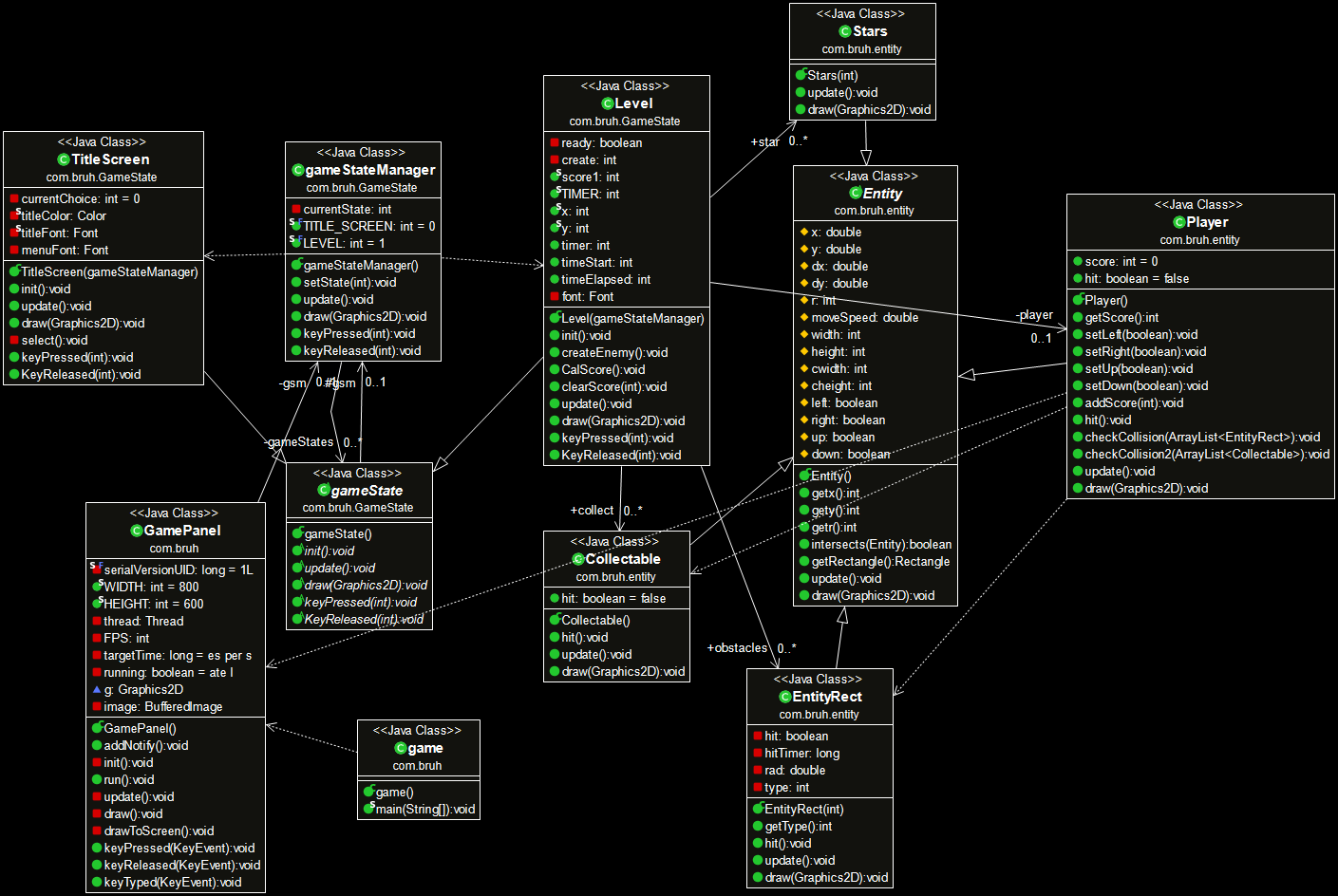
This Java project was to create a GUI which required us to expand upon the initial design ideas that we learned and implemented in the first part of the java module(this included the drone game but with the eclipse interface). The final GUI created is a game that has the user control a player using arrow keys and collecting the collectables while dodging the rectangular enemies that keep spawning in from top of the arena. The user receives points for both collecting and surviving for as long as they are able to. The game also has a background to it which is presented as white stars in a galaxy which makes the game look more appealing to the user.

Introduction

The final GUI is a simulation or a game where the user can play as a player which is controllable so that they can move around in the arena. The player score points by collecting and surviving in the arena. This report documents the design of the GUI along with a testing section to ensure that the final GUI worked the way it was meant to. Moreover, a user manual that will show how the user can start and exit the game. The Report will include a conclusion and how I reflected off from this assignment and the areas that might need improvement.

**Design of the Game**

UML:



Classes in order they were created

1.**GamePanel**

This is basically the canvas which will be used to draw all the images and animations that are needed for the GUI. This class implements the runnable interface in order to start and create a thread where thread will allow the user to process all UI events. Moreover, it includes the width and height of the canvas which is 800 x 600 (width and height). The class includes a constructor which makes use of super() and other parameters for keyboard inputs.

2.**gameState (Abstract class)**

This is an abstract class which was created to save time from redoing most things. It acts as a template for other classes. It includes five abstract methods that other classes can inherit (classes that inherit these include gameStateManager, Level and TitleScreen), for instance, it includes abstract methods called keyPressed and keyReleased, the TitleScreen class inherits these methods for keyboards inputs to select “start” or “exit” buttons for the game.

3.**gameStateManager**

This class is used for managing the state of the game in real time. It contains all the methods that will be used for the Level class. These includes keeping the game updated in its current state, getting the current state of the animation and the key events that are pressed and released for the player.

4.**TitleScreen**

This class inherits the abstract methods(which are public, so these methods can be accessed by any other class if required) from the abstract class gameState in order to make the title screen of the game. It uses the draw method with the graphics as its parameter to layout the font and color of the strings and to create start and exit buttons to either enter the game or leave. These buttons on the other hand make use of key events on the keyboard to choose and select the buttons.

5.**Level**

This class implements the inherited abstract methods from gameState which include the initiate, update, draw and key events. These methods were automatically loaded in since these are predefined methods. This class is basically the arena where other classes were imported such as the Player, EntityRect, Stars and Collectable are imported so that they can be drawn and displayed in the GUI.

* The Player class has the key events in the keyPressed method to move around in the arena.
* The EntityRect is the enemy class which are displayed as rectangles that the player needs to dodge. If there is a collision between them, the game stops and will be taken back to the title screen to start over.
* The Collectable class provides the player with points everytime it collects it. When the player collides with the collectable, the collectable gets removed.
* The Stars class acts as a background for the GUI which is displayed as white stars visible from a distance in the galaxy.

It also includes a timer to show how long the game has been running for. The class also includes a function that calculates the score by adding points received for surviving which is time and the collectable score which give you 5 points, if the player hits an enemy, the score will be set back to 0 since the player dies. The position of the player in the arena is also displayed in the arena

6.**Entity**

This is another abstract class which was created for the Player, EntityRect, Collectable and Stars. The variables are protected in this class so that it can only be accessed by classes that are within the package or by other subclasses such as Level. Therefore, this class includes protected variables such as the x, y coordinates for the player, dx and dy which are the distances in x and y directions, cwidth and cheight which are the collision width and height for the objects(Player, Collectable and EntityRect) and moveSpeed to adjust the movement speed of the objects.

7.**EntityRect**

This is the enemy class which extends from Entity which has he attributes and methods to create rectangular enemies that will descend from random positions in the arena and would be removed once they reached the bottom. Moreover, the rectangles were also given the width and height for the hitboxes. The width and height of the rectangles were 40 by 10 with a movement speed of 4.

8.**Player**

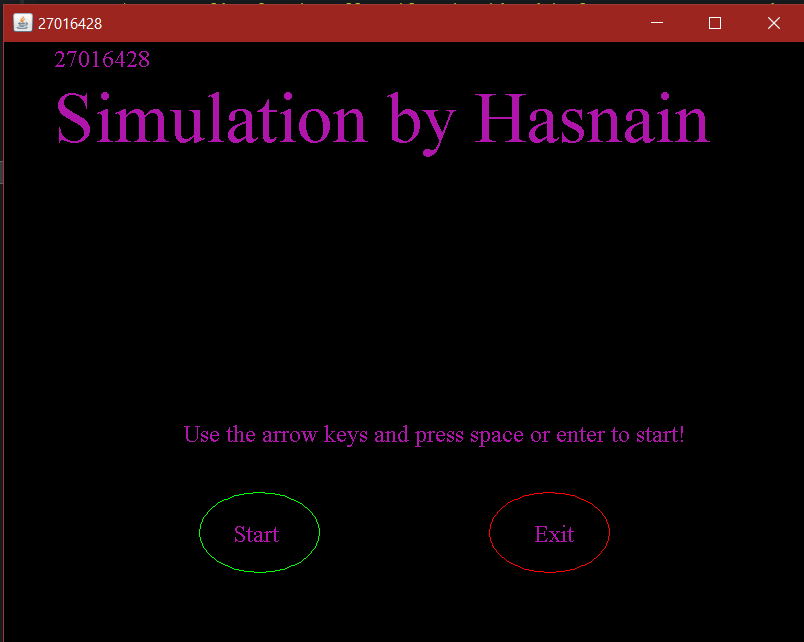
This is the player class which extends from Entity as well. It includes methods which have a parameter to move the player around using simple arrow key movements on the keyboard. Moreover, it has two check collision methods with parameters which consists of 2 for loops, one for checking the collision with the enemy and the other to check the collision with the collectables. The distance x and y were given speeds according so that the player can move in both the x axis and y axis of the arena. Furthermore, If statements were used as well to make sure that the player does not go out of the arena.

9.**Collectable**

This class inherits from Entity abstract class. Same methods applied as the EntityRect except that the collectables are oval rather than a rectangle with different color and movement speed.

10.**Stars**

The stars inherits from the Entity abstract class which uses most of the variables and methods but the difference is that it does not interact with any of the other objects in the arena. This object was created to act as a background in the arena.

User manual

When the user initially runs the GUI, the first thing they will see is the title screen with a start and exit button to either start or exit the game. The user what keys to use to enter the game as shown in the screenshot.

Upon entering the game, the user will require to dodge the obstacles and earn points by collecting the green collectables in the arena. Each collectable is worth 5 points. The user will also receive points for surviving as long as they can in the game. There is also a timer that shows for how long the user has survived in the arena.

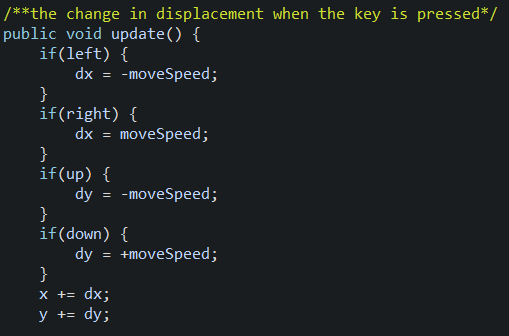
If the user gets hit by the enemy, the user will be taken back to the title screen to start over or just end the game.  


Tests performed

|  |  |  |  |
| --- | --- | --- | --- |
| **Tests** | **Expected Output** | **Actual Output** | **Solution** |
| 1.Player movement left | The player moves left when the left arrow key is pressed. | The play does move left when the key is pressed. | N/A |
| 2.Player movement right | The player moves right when the right arrow key is pressed. | The player does move right when the key is pressed. | N/A |
| 3.Player movement up | The player moves upwards when the arrow key pointing upwards is pressed. | The player does not move up when arrow key pointing upwards is pressed, instead it moved left. | A solution to this is to add a dy distance with movement speed set to minus so that the player moves up in the arena. |
| 4.Player movement down | The player moves downward when the arrow key pointing upwards in pressed. | The player does not move down when down arrow key is pressed and moves right instead. | A solution to this is to add a y axis with movement speed set to positive so that the player moves down in the arena. |
| 5.Title Screen with Start and Exit buttons | Select Start and Exit buttons using left and right arrow keys as movement keys and space or enter to select the option. | The arrow keys function properly along with the enter/space key functioning properly | N/A |
| 6.Player interacts with the collectable | When the player collects the collectable, the collectable is removed and 5 points are added to the score. | The collectable is removed and 5 points are added. | N/A |
| 7.Player interacts with Enemy which is a rectangle | The player dies and takes the user back to the title screen. | The player dies and is taken back to the title screen but the GUI crashes. | N/A |
| 8.Background Stars like in a galaxy. | White coloured stars are presented as a background and does not interact with enemy, player and collectable | The stars act as a background and does not interact with any of the objects. | N/A |
| 9.Collision hitbox check for the objects | Hitbox alignments are correct for the objects so that when the player comes into contact with the collectable and the enemy, the collision is detected at that instant and not from a distance. | The hitboxes were not aligned properly for the rectangular enemies and the player. This caused the collision to be detected from a distance rather than the two objects coming contact. | The solution to this was to subtract the x and y with the cwidth and cheight and then multiply cwidth and cheight so that the entire rectangle is covered with the collision effect. |
| 10. Collectable | When the player hits the collectable, 5 points are added to the score. | The collision is detectable but the collectable acts as an enemy and the GUI crashes. | A fix to this was to remove the collectable completely once the player came into contact. |
| 11. Stars | The stars act as a background and does not interact with any of the objects in the arena and there are stars that are stationary in the arena. | The stars do not collide with any of the objects and present as a background. There are some stars that are stationary in the arena. | N/A |

**Evidence for correction of tests that came out wrong:**

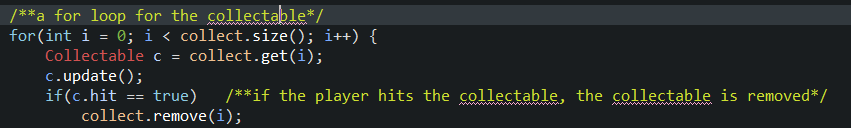
**Test 3 and 4**



**Test 9**



**Test 10**



Conclusion

To conclude, my GUI came along very well however it was not made using the right graphics and imports due to the fact that I had very limited knowledge on javafx and did my best to create something unique from it but was not able to achieve satisfactory results when I was trying to make the simulation during my winter holidays. Moving on, the most interesting part during the making of this project was trying to fix the collision placement of the rectangle enemies. This took a lot of time for me to somewhat perfect it. Most of my time was also spent on aligning the strings in the title screen and creating a hovering effect over the start and exit buttons. I also how to move the player diagonally as well instead of north, south, east and west. This was done by include the dx and dy to the x and y coordinates.