CS319 – Object Oriented Software Engineering

Final Report

Run Dot Run!

Group 2-H

Beyza Tuğçe BİLGİÇ

Gökalp KÖKSAL

Gökçe ÖZKAN

Emine Ayşe SUNAR

Contents

1. Implementation Flow

We have mentioned that the game will be implemented with a 3-tier architecture in the design report, thus we implemented the code using this architecture. Since we have a 3-tier architecture, we have mainly 3 packages, which are Game Objects, Game Control and GUI. We have implemented the GUI package and the Game Control and Game Objects packages separately, and then combined them both. Since the Game Control and the Game Objects packages were tightly communicating, both of them are implemented together. We have first started by creating the Game Object package. The super class GameObject was firstly created and the subclasses were then implemented. After this, the Game Control package was created. The classes of the Game Object package, which are the main objects in the game, were used in the Game Control package. The Game Control mainly uses the Game Objects and controls them by updating and drawing them. After the main game play was implemented, the GUI was added on top of this; in other words the GUI was attached to the game to create a fully functional game.

2. Changes in the Design

- In our design report, we have planned that the GameManager class, which is the façade class of the Game Logic Subsystem, would extend JPanel, however, after doing some research we have found out that Canvases work more efficiently in 2D games like ours. So, eventually we decided that our GameManager class would extend Canvas. After extending Canvas, a buffer strategy was added to the class to increase efficiency. The buffer strategy is a class which creates the mechanism to organize complex memory. Since our game has to handle many objects at the same time, we decided to add a buffer strategy.
- We have mentioned that that the collision detection would be done in the GameManager class, however to create a more organized code, we decided to add a new class named CollisionDetection which has a method named detectCollision. This method takes the linked list of objects and first specifies the Dot object, which is the main player. Later on it checks whether the other objects in the linked list collide with the Dot. An instance of

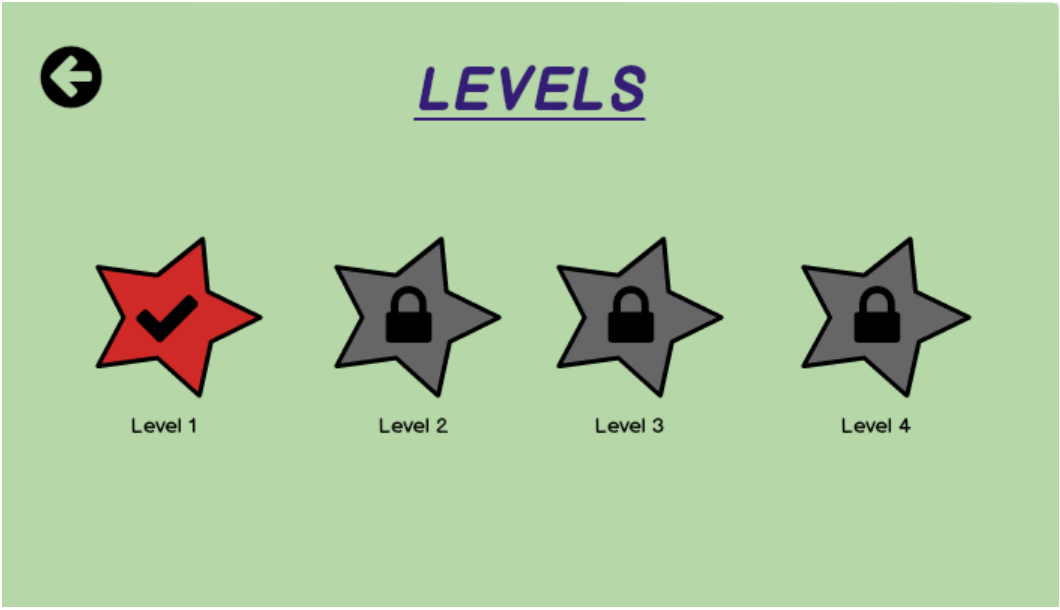
this class is created in the Dot class and the detectCollision method is being called in the update method of the Dot class.

- For our FadingLetterBox class, we mentioned in the analysis report that once the dot collides with these objects, the letter will slowly disappear. Instead of making the letter boxes disappear, we have decided to give them a velocity, so once the dot touches these objects they will start falling down.
- In the Dot class, we added the “lives” attribute; getLives() and setLives(int) methods because we lately added the feature of lives, which refers the the lives (or chances) of the dot and will decrease as the game falls to the ground or collide with obstacles.

3. User Guide

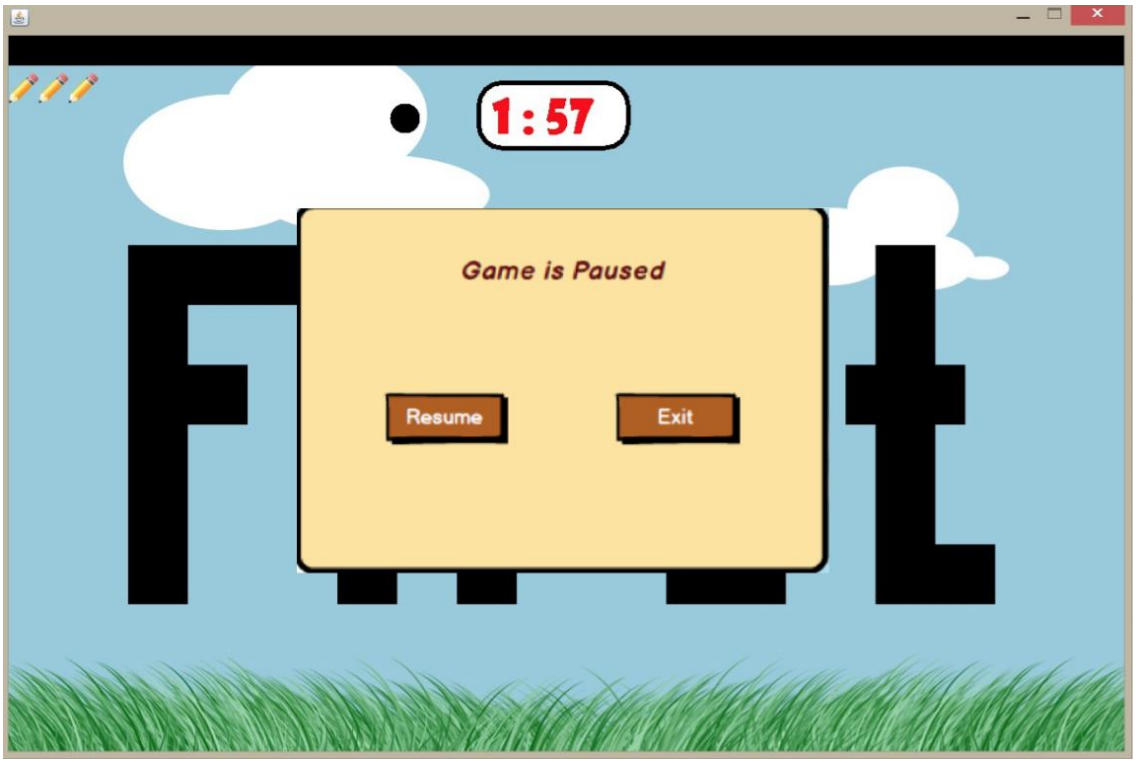
Main Menu Screen (final version)

The game starts with a menu screen offering a variety of options. Before playing the game, the user is able to arrange the volume of the music which will be playing on the background of the game. By clicking the question mark button at the upper right corner of the menu, user can find all necessary information about the rules of the game. Also, credit option provides the user with names of the developers of the game. If the user can exit the game with “Quit” button or display the levels screen with “Play Game” button.



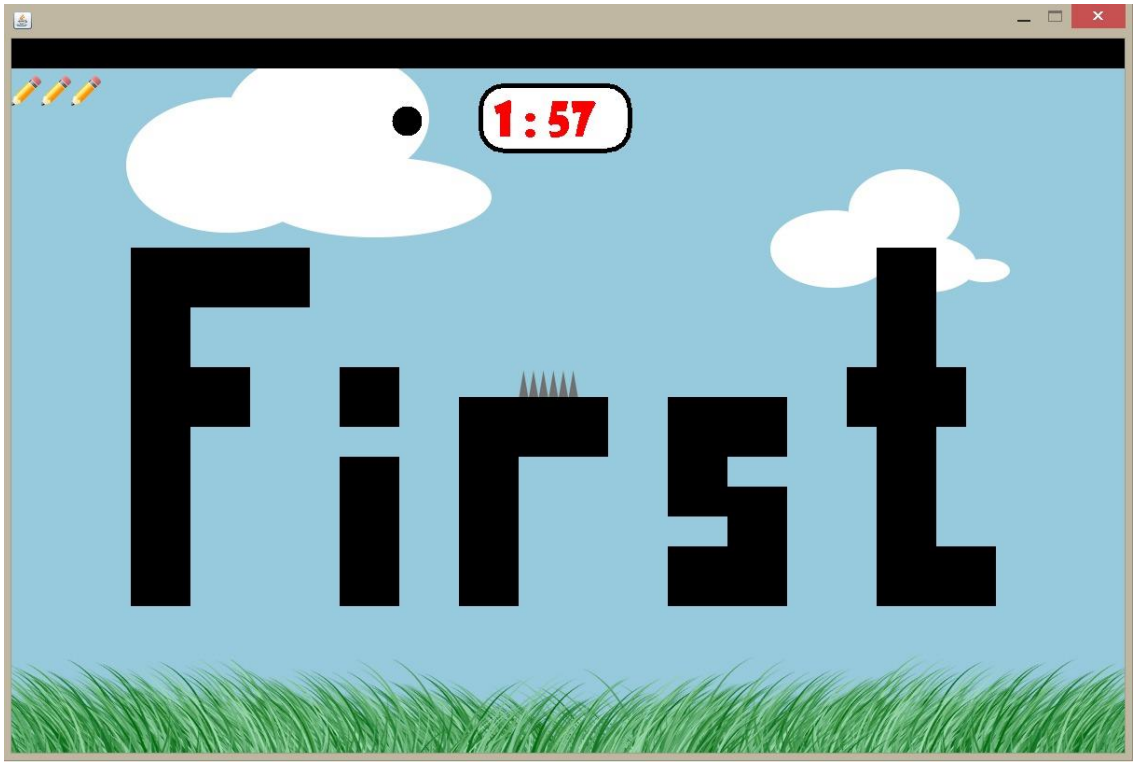
Levels Screen

After selecting the play game button, the levels including different types of difficulties such as spikes or erasers are displayed and the user has to select an unlocked level. After the game starts, there is a counter on the screen then, the user can check the time remaining for finishing the level.



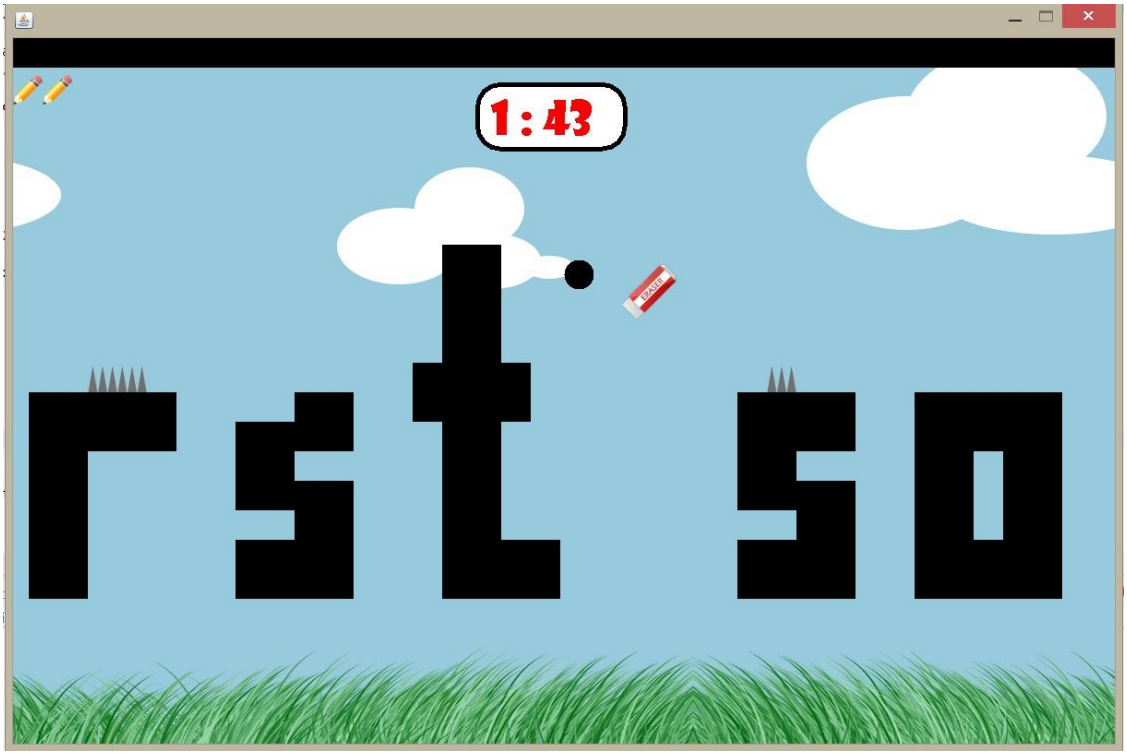
Pause Game Screen

By using keyboard, the user controls his/her movements during the game. The purpose of user is to reach the end of the sentence and to finish the game without falling down. At the same time, the user has to track how much time is left through checking the counter. While playing the game, the user may pause the level and return to menu by selecting exit button or continue with playing this level by selecting resume button.



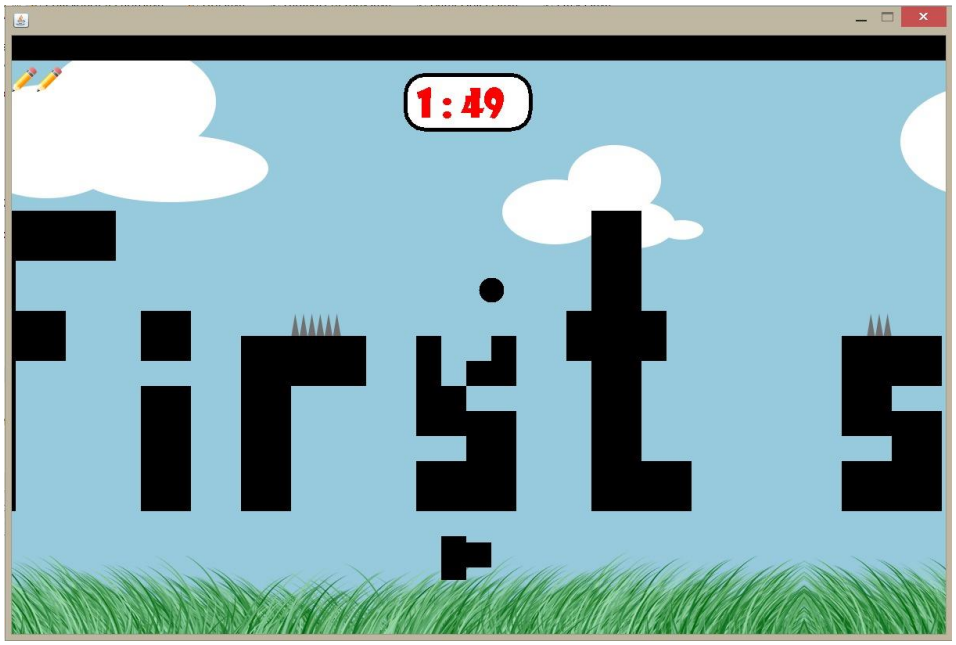
Game Screen with Spike (Level 3)

The user encounter diverse obstacles during the game. For instance, the user should not hit the spikes because there will be a decrease in the number of “lives” and the user will return to the beginning of the level in case of hitting spikes.



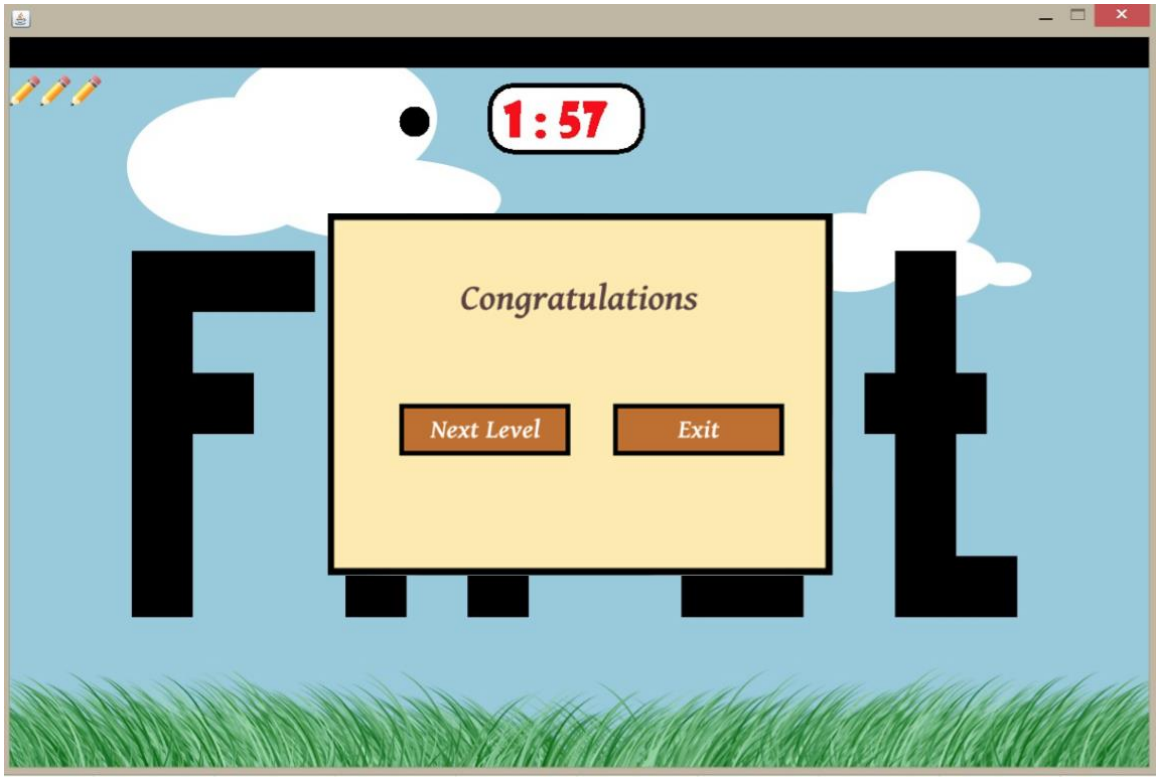
Game Screen with Eraser (Level 3)

Additionally, the user has to face with “erasers” which is another obstacle of the game after he/she manages to reach level 3. These eraser objects comes from the sky or the ground.



Game Screen with Fading Letters (Level 4)

If the user can unlock the level 4, the game starts with “fading letters” which are able to disappear in 1-2 seconds when the dot hit these letters.



Next Lexel Pop-Up

If the user manages to finish any level successfully then, there will be a shown pop-up asking the user for deciding whether he/she wants to play next level or return to main menu. In case of falling down or hitting an obstacle, the user returns the beginning of the level without resetting the counter.

4. What has been left in the implementation?

The main game has been mostly implemented, except for some details. For the first iteration of the implementation, we wanted to give a feel of how our game will be and how it will be played. So, we created an example level, which we added most of our properties in. Normally, during the game, as the player unlocks different levels, new obstacles would be introduced in the

game, however as mentioned above, a sample level has been created to show all different properties in the game.

What has not been implemented yet?

- Once the main menu is opened, there will be some music playing in the background. The player will be able to change the volume of this music from the main menu. However this property hasn't been implemented yet.
- As mentioned in the analysis and design reports, our game will have multiple levels and the player will be able to select the levels that are unlocked. The levels haven't been created yet, however the infrastructure for creating the levels, which is our levelImageLoader class has been implemented. The levelImageLoader class provides random level images according to the selected level. The properties and methods of the levelImageLoader class have been implemented, however the images of the levels haven't been provided in the code. These images will be drawn manually by us.

  There will be a file that would keep the data of the last unlocked level and according to this data, the levels will be shown as locked or unlocked in the LevelPanel. However, since the levels haven't been created yet, the file and the related methods to this process haven't been implemented yet.
- In the first iteration version of our game, we haven't added curvy letters yet. The letters the dot has to over jump are letters which have sharp edges, thus the dot does not slide from the edges.

5. What is next?

After completing the parts left for the first iteration, we are going to make different additions to the original concept of "Run Dot Run".

In some panels, such as main menu, we will add some animations (e.g there will be a jumping dot on the background of Pause panel, Main Menu panel etc.).

In the second version of the game, in each level, there will be several check points placed on the path of the dot, so that, if the dot cannot pass the obstacles or falls to the ground, it will go back to the last check point's position to continue the game. This will help the player to finish the level in the given period.

We are going to add time bonus to the game in the next iteration. Time bonuses will be placed on the letters or a bit higher. If the dot can get the bonus, depending on how much time that bonus includes, the time left for the game will increase. Also, with the same logic, there will be some penalties placed on or higher the letters. If the dot touches them, the time left will decrease depending on the time that penalty point holds.

There will be bonus life reward added in the levels. If the dot can get that reward, which are also placed on or higher than the letters like the time bonuses, it will gain a new chance of life.

6. Conclusion

In this project there are many things we have learnt to get use to the real life projects since the main aim of this course is to enable us to get some experience on working in a group of people that you do not know in real jobs.

Experiences:

- **Planning, Designing the Project**

We saw that planning and designing the project before getting into coding has huge effects on the quality of the work. It enabled us to see some of the problems that might occur and to correct them beforehand. Also, by looking at the designs it is possible to see everything that are needed in a chart instead of searching in a pile of code.

- **Project Management**

We learned how to use Github to build our project with other teammates. With Github, we can share our work with our team easily. It gives us a place to load our work and save them in instead of having several copies of work done.

- **Time Management**

Thanks to Bilkent's tough conditions of working, we had to learn to deal with the deadlines to experience time management.

- **Team Work**

This project made us to see how the work is being done in real job's projects since we were assigned randomly in our projects. We had to learn how to work with other people that we do not know to proceed on the project.