CS 319 - Object-Oriented Software Engineering

Second Iteration Final Report

Group 2E:

Ferhat Serdar Atalay

Aylin Çakal

Ali Bulut

İsmail Serdar Taşkafa

Submission: 05.05.18

1. Introduction

The implementation process started right after the first iteration finished. We decided to implement our project with the help of IntelliJ and Eclipse. Each group member connected to GitHub repository which is 2E.Planet-Trip created by Ali Bulut and committed the changes directly to here. Each time when one of us fixed the class, he/she could easily push it to GitHub and merge it with others' codes if necessary.

In terms of the distribution of the tasks, at first we divided the tasks as User Interface classes, Game Management classes and Game Object classes. Each group member tried to do their best for their own task. Ali was responsible for Game Frame and Game Map Manager. Aylin was responsible for User Interface Classes. Serdar Atalay was responsible for Game Engine and Serdar Taşkafa was responsible for Game Object classes and also for business logic for Collision Manager. Then, Ali implemented all specific details for each class and merged all of them. We tried to add all main functional requirements of the game.

Since we have to use the principles of object-oriented programming, we had mostly difficulty in implementation. We wanted to follow the structure of analysis and design reports but sometimes we changed some classes and methods. For example we added StopWatch class for being useful. It helps us to manipulate timer easily and we could get more accurate results. Additionally, we added some functions for creating easy communication over classes.

In general, we have finished all of our promised functional requirements like gameplay, several bonus types and menu selections. Rest of the report will mostly mention these and also some guides to install the game.

2. Design Changes

- In design process, we thought of identifying the object types as integer values. Since we stopped storing object types, getImage() function of the GameEngine class does not take typeOfObject parameter anymore.

- In the function checkBallWallCollision() of the CollisionManager class, we normally take screen width and screen height alongside with ball as parameters. Since we decided that height and width would be constant, those parameters are excluded.

- Since we need to check if the level is cleared or not, we decided on adding a new method called getNoOfDestMeteors() to the GameMapManager class. We use this method in the GameEngine to see if the number of meteors left is equal to zero or not.

- We merged UndestructibleMeteor and DestructibleMeteor classes in an abstract Meteor class.

- Subclasses of DestructibleMeteor and Bonus classes now take xPosition and yPosition as parameters in their constructors.

- StopWatch class added for keeping elapsed time easily. Also FileManager class deleted for being useless.

- GameCanvas class also added for easily displaying and repainting game frame.

3. Lessons Learnt

In the final iteration, we realized the importance of using Github. During the first iteration, our project was not fully implemented. However, we have made a fast progress in the second iteration. Without the use of Github, we would face much more bugs and this would slow us greatly. Also, there would be lots integration problems.

However, our use of Github was not proper and we could have had headaches. We have made the development exclusively on unstable branch and pushed to master only after finishing the whole implementation. This is no different from doing the implementation solely on master branch. Since we are relatively inexperienced with Git, we could have easily altered the whole codebase with a few mistakes. There were a few such incidents but we resolved them quickly. We avoided such issues with using Git with care, by not working on the same files and with a bit of luck. But in the future, we may not be so lucky and we should work on separate branches and merging to main branch only after making sure our code works correctly.

Also, maintaining good communication helped us to stay on the same page. This helped us to do the implementation faster and help each other when we are facing problems. Otherwise, we could not manage to finish the project.

4. User's Guide

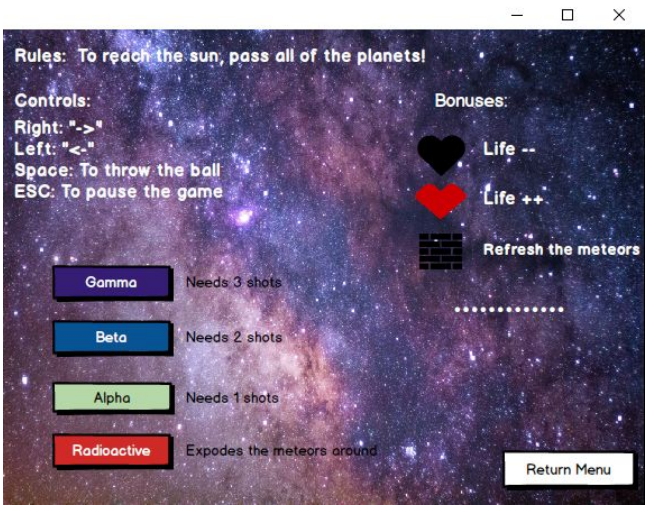
4.1. System Requirements & Installation

Since it runs with an executable jar file, Planet Trip will be played without installation and on any computer with a Java distribution.

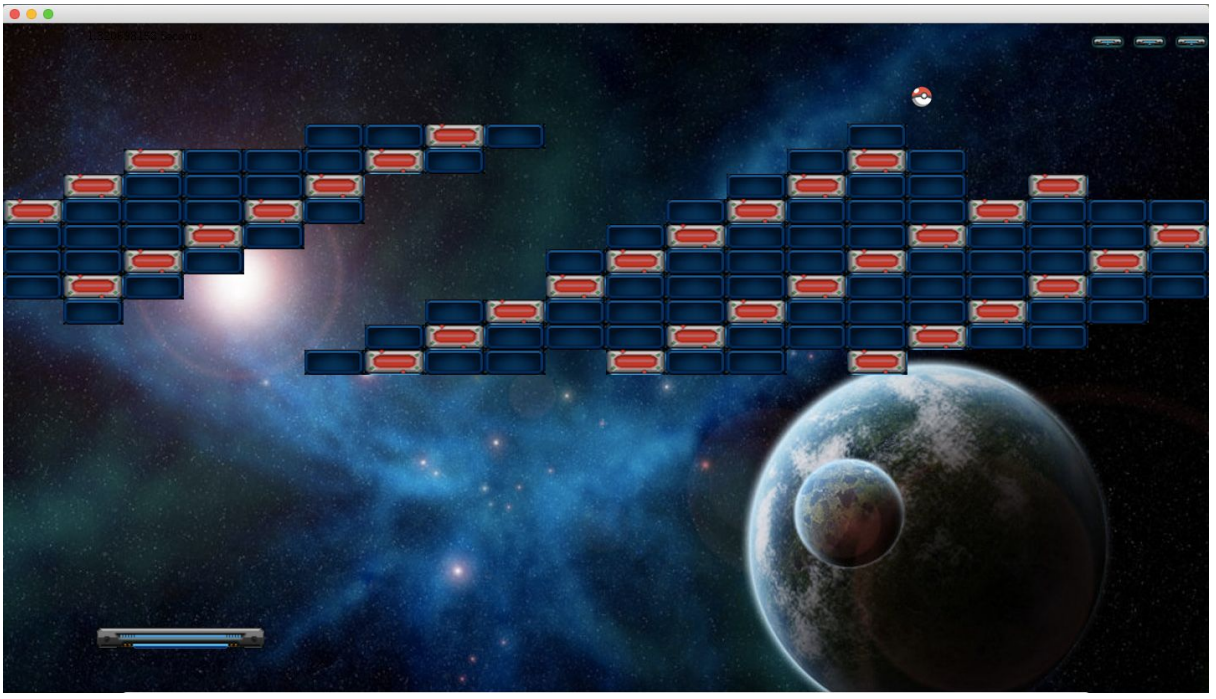
4.2. How to Use

When the user runs the game, main menu is opened. The user may start the game by clicking Play Game and selecting desired the level to be played. The game is closed by Exit button.

The user learns controls of the game and icons to be used in the game by clicking How to Play button.



After clicking Play Game button, game is started. For the first iteration, there will be a paddle to show its movement horizontally and some meteors. The paddle will be controlled via using arrows on keyboard.



The user can adjust the volume by clicking Options button. By the Apply button, all changes are saved.

By the Credits button, the user is informed about the game developers and may return back to the main menu.