Bilkent University

Department of Computer Engineering

Object Oriented Software Engineering Project

Project short-name: Curve Fever

Final Report

Barış Poyraz, Yunus Ölez, Zeynep Delal Mutlu

Course Instructor: Bora Güngören

Final Report
December 24, 2010

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Object Oriented Software Engineering Project CS319.

Table Of Contents

1.Introduction

Our project is a replica of the games “Curve Fever 2”, “Tron: Evolution”. It is a Java Desktop Application. In this game, each player is given a distinct color randomly and allowed changes in the color. When the game starts, the colors to move by their head and leave a pattern behind themselves while giving some gaps. The objective of the game is not the cross each other’s color. The players are allowed to move their color, either left or right. During gameplay, there pops some popups in the screen which can either give player a boost or harden the gameplay for his/her. Modern games with this kind of gameplay include “Slither io”, “Curve Fever io” and Rockstar Games’s GTA V has a dlc based on this kind of games.

This report is the final report of the project. It includes user’s guide, changes to implementation and general API documentation.

2. User’s Guide

Since this game is a Java Desktop Application, the users must have a Java environment to play our game. In order to run the game, the user should download the game from our github repository (https://github.com/ZeynepDelalMutlu/Bilkent_CS319-1). User can view the source codes and run the program using an IDE. We will create the .jar file for the project later.

3.Changes to Implementation

We tried our best to keep the design as same with the implementation. The mockups we draw for the previous reports are used to implement the game, but some changes made for implementation.

For example for view package, in order to go through each, first we created string variables to pass through each panel. But while we were doing some research on the project, we found that CardLayout, built in Java is the thing we should have done. So we changed implementation of passing through panels with the help of CardLayout. In adding player screen, at first we thought that getting color selection from user by simply asking them to enter some integers between 0 – 255 to satisfy r g b numbers, then dynamically show the color they choose. We thought that this can be frustrating for user, then we found that JColorChooser of Java allows us to do this in a better way. Simply, when clicked on the button, JColorChooser shows up, and here user can select his/her color for the game.

One of the biggest problems we had was about gameplay. In the gameplay, though it is completely playable, there happen some performance issues based on the number of players. Basically, as the number of players increases, game speed decreases due to our implementation mistakes in the gameplay. When we learnt how to solve this issue, we realized that, it was too late to change the design due to shortage of time. What we learnt from this is that, while doing the designs, the working mechanism of programming language should be considered.

We followed our design pattern and object model as possible as it can be but realized that some classes can work better with different implementations and designs.

4. Complications During Implementation

One of the biggest issues was integrating our code because of lack of communication. What we should do was simply, pull the code from most recent version and push it after we made some changes. The gameplay itself was the hardest part of this project, because there were multiple key events occurring and there were power ups spawning randomly on different places of the game panel, making this game event-driven game. Other issue was to get key events from the user, though the keys are assigned to players with object oriented style, the issue was occurred because of keycodes. We converted the typed key into key code but at first had some problems converting it back into KeyEvent.

5. General API Documentation

Our program follows MVC pattern. Codes of this patterns can be found on our github page within subsystems Model, View, Controller. Other than main panel which uses CardLayout as discussed before, rest of the panels use AbsoluteLayout, a layout in which everything is placed in the exact places, pixel by pixel.

6. Conclusion

To sum up, we learned general principles and details of object oriented software engineering. We realized that there can occur some mismatches between design and implementations. We think that we improved ourselves significantly over the course itself and while doing the project.