Object Oriented Programming
CS 319

Final Report
Section 01
Group 1G

Ali Altaf Salemwala - 21500413
Hüseyin Eren Çalık - 21402338
Bora Bardük - 21401029
Kıvanç Gümüş - 21401767

Introduction

This document is the final report for our semester project. It contains a description of our application system, details of our implementation stage, changes we made to our original design during our implementation of the project, challenges we faced during the semester and how we overcame them, a small manual for users on how to play the game, and a final word on what we learned in this course.

Description of the System

EmojiStrike is a 2-D computer game inspired by the classic arcade game Worms. The game is designed so that two friends can have a short light-hearted competition; for these reasons we have set certain game features, i.e. two players can play, the game does not take very long, and the game is slightly competitive, but not highly enough for it to become intense. Users select emojis to represent themselves in the game, and play by shooting others with weapons provided to them. The object of the game is to kill all the other players' emojis. Players can enhance gameplay by getting power-ups, which increase ammo, replenish health, and provide more weapons. Players can also select the maps they play on. To ensure that the game is not too long, each player has only 30 seconds for a single turn.

Implementation Details

This game was planned using the Model-View-Controller design pattern. The views aspect involves the GUI of the project, and this is handled by Unity after we have created it. The model was made by us, and it involves the underlying workings of the game, and is completely coded and maintained by us. The controller involves the link between the model and the view, and this is implemented partially by us in custom features, and partly by Unity in the methods it provides as interfaces between the UI and the model we designed.

We implemented this game in Unity3D, which was a huge windfall for us as we could design the GUI very easily. It greatly changed our plans for the implementation stage, since we had previously expected to have to code the UI in Swing and this would entail not only a lot of trial and error, but would require us to practice and refamiliarize ourselves with Java's GUI packages. In Unity, while we had to change our coding language to C#, we were still able to maintain the object structure we had planned earlier, with a few changes. Our work went by much faster, and we were able to design a much smoother and user-friendly user interface since we could see immediately what was working and what was not.

There was a learning curve involved with Unity, since most of us had never used a similar tool before, but we were able to help each other and soon developed quick skills with the application. We did not know how the scripts, assets, and screens were linked together and spent a lot of time understanding how they were used in combination with each other, but once we learned that, the game began being developed very quickly. To adequately use Unity, we learned C#, but the plethora of online resources made it much easier for us to learn the language.
Since Unity is such a powerful tool, most of our implementation problems were greatly simplified by it, and in many cases, such as the firing of bullets, where we were very concerned with the coding of the physics and the animation of the objects, we found Unity offered large

compensations where we felt we would be lacking. The physics engine of Unity is very extensive, and supported us in cases ranging from the shooting of guns, helping us handle the firing of bullets and their contact with other players, to the simple clicking of buttons (the Collider objects were very useful here). The Destructible 2D objects in Unity were probably the most useful things we discovered, and the most important detail in our implementation after the Unity tool itself. We had come up with many ideas about how to implement destructible maps, and had finally settled on using sprites of images as maps, and recording changes on individual bits to represent destruction. While the idea seemed like it would work, the implementation was very daunting and we were strongly considering dropping the feature from our project. Destructible 2D objects, however, convert images into bitmaps and then record damage and adjust the damage accordingly. Thus, we could focus on the object-oriented aspect of the implementation of our project without having to concern ourselves with the much more technical aspects, like conversion to bitmaps, an translation of these bitmaps.
Saving and loading the games, while possible, was very difficult and we ended up with not enough time to implement it, both in terms of the relatively complex GUI required, and the very complex mechanisms involved in storing, retrieving, and accurately interpreting the data. We did manage to create a feasible strategy that we believe could have worked had we had enough time to implement it.

Challenges Faced During the Semester

The biggest challenge we faced during the semester was overcoming the psychological aspect of the project. We planned the project relatively quickly, but it built up a large façade before us and we were a bit intimidated about starting the actual implementation. Once we started, however, this problem quickly disappeared as the group momentum quickly picked up.

Another big problem was that all of us were very busy over the course of the semester and had trouble organizing meeting times that were appropriate for all of us. We eventually each allocated mutually acceptable timings in our schedules and committed to those timings. Additionally, if any of us could not reach at the decided time, we learned how to make up for the missing person's work so that the group would not lag behind, but also ensured that no person had an unreasonable number of absences.
Since not all of us had the same strengths, we had to quickly learn each other's strengths and weaknesses and adapt to them. For example, stronger writers spent more time on the reports and stronger coders spent more time on the actual coding, and those with weaker laptops teamed up with those with stronger laptops so they could use Unity together. We quickly learned how to boost each other up.
The learning curve of Unity was a problem for us initially, especially since some of us were sharing the computer with each other. Additionally, since all of us did not know C# or JavaScript properly, we had to spend a lot of time learning how they work. Finally, some of us had a lot of misunderstandings about Unity and how it works, and none of us recognized them until much later into the project, so they did hamper our progress a bit. We quickly adapted to the new technologies, however, and are reasonable proficient with them now.
We had some timing and pacing issues in creating the final version of the game. The most major things we were unable to implement were the save and load game facilities. This was a challenge

throughout the semester, as we had to meet many deadlines, and although we did improve with time, we were unable to completely fix this issue.

Changes to the Design

As we began implementing our system, we realized that some aspects of our design were poorly considered, or could be improved with a different pattern. Additionally, some features promised to be part of the final system were difficult to accomplish. On the whole, our system is mostly similar to our initial plans, but new developments in our design and our decision to use Unity resulted in a few changes to the final product, and drastic changes to our planned implementation process.

We have abandoned projectile motion in our guns, and have instead provided only facilities for linear shooting. The destructions which resulted from projectile shots looked very odd and did not make for a good user experience, so we decided to choose a slightly lower level of functionality for a more attractive user design. Since there is no projectile shooting, we allowed direct front and back shooting and to do this we added an extra command whereby pressing “F” shoots in the direction opposite the user’s character.

The UI is the same as we had planned it originally. Changes in the functionality, such as no projectile shooting, resulted in changes in the UI since we no longer have projectile arrows indicating where the weapon is aimed. These minor changes are almost indistinguishable form our original imagining of the game.

In terms of functionality, a major aspect we did not include was the ability to save and load previous games. We lagged behind a bit during the semester and could not catch up until the end, so we were sadly forced to abandon this functionality in the end. This is probably the most major aspect of the originally planned project that we did not include.

User Manual

This is a very simple system, intended for all sorts of computer users, including those who are not very computer literate. The machine should have the Java 8 SDK so that it may run the .jar files with which the system runs. Internet, high amounts of RAM, memory, or processor power are not necessary for our game.

When a player first begins the game, they will be presented with two options, to begin playing a new game, or to open the tutorial. The player can navigate with the keyboard by pressing the "P" key for a new game, or "T" for tutorial. The tutorial page opens in a new screen and the player can return to the home page by pressing "R".

The new game screen occurs in stages. First, there is a stage, where the users must cycle through a list of maps and select the one they want to play on. They can cycle using the left and right arrow keys on the keyboard. To move to the next stage, users can either press the Enter key.

The second state contains a screen where users can choose which emoji they want to use to represent themselves, and what they want to name themselves. Here users can use the mouse to focus on the input fields and cycle to the desired emoji, or use the Enter key on the keyboard to navigate to different fields on the screen. For example, users will first be in the first player's name selection box by default and can type in the name here, and then press enter to move to that player's emoji selection and use the left and right keyboard keys to choose the desired emoji, and

press enter to move to the next player's selection section and so on, until pressing Enter will move the screen to the game screen.
In the game, users can play using the arrow keys to navigate, the "F" key to switch direction, and the Space key to shoot. The weapon is changed using the "W" key. If the bullets from the weapon hit the terrain, the map also gets damaged. Players lose health if they are hit with bullets and lose the game if their health goes to zero. Players can navigate their emoji to the position of Power-Ups, which fall from the sky. If they catch a Power-Up, players experience various benefits, such as increased health, a special weapon, or an ammo boost. The benefit depends on the specific Power-Up caught. Players have 30 seconds to play their turn before it switches to the next player in line, and there is a small box in the corner displaying the timer. The game is over when only one player remains.
Players can pause and un-pause the game by hitting the "P" button, and go back to the main menu screen by hitting the "R" button.

What We Learned in this Course

This was a very new and unique course for us. While all of us had prior experience in doing term projects, this was our first time with a project that required such a substantial amount of planning and report-writing. In previous courses, we would not give much importance to reports or planning, and we did many things on the fly. This group was also the first for all of us where we were all strangers, and we each had our own different styles of studying and working, so were expecting a lot of conflict. Because of this wariness, perhaps, we were careful to cooperate and ended up having the best out of almost all of our project experiences so far, and we all learned a lot about cooperating with others in groups.
We learned that time management is of the utmost importance, as we were forced to abandon an exciting functionality of our game that we ourselves were looking forward very much to implementing, but could not due to poor planning and time management. We learned some scheduling and time and task analysis techniques to better handle these problems to get back on track, but we were forced to abandon some functionality regardless. These skills and the lesson that came with them will stay with us for a long time, however.
We also learned a lot of planning techniques, along with an innumerable number of design and implementation patterns. It was a big, but very important, jump from our earlier rag-tag styles of coding. We have all come a step closer to being software engineers thanks to this, and feel we are all more prepared not just for future senior projects, but also for future internships and jobs.