Angel Checo and Daniel Segarra

CSCI 492.01 - Functional Programming in OCaml

Professor Alexey Nikolaev

OGalaga Term Project Paper

OGalaga

Both Angel and I have an affinity for visuals and decided that a video game would be a suitable task for us. We initially were working on two separate projects and realized it would be easier for us to work together. I initially tried using the OCaml graphics library, with some starter code provided by our professor. Angel planned on using a few ReasonML/Reprocessing projects and articles as guidance for his own video game. After some reflecting we figured it would be best to follow some existing projects as guidance for maintaining our own game. Our main resources would become the ReasonML and Reprocessing documentation as well as an archived livestream of two seasoned Reason developers replicating a renowned mobile game, Flappy Bird. After spending a lot of time with these resources and technologies, we realized we could accomplish something we would be proud of and enjoy creating. Thus, our first game development experience in OCaml began.

Although the project seemed feasible we quickly encountered a set of strange issues. We attempted to follow documentation in order to bind key presses with game events. When running it on Angel's computer it seemed to work perfectly fine. The ship was now moving left and right. Daniel attempted to test it on his machine but received a nasty error. It stated that there was an issue with the remft parser. Confused we did what all confused programmers do, a quick Google search. Unfortunately nothing was returned from the query. We referenced the "Flappy Bird" video we had watched and we had nothing out of the ordinary in our code. After much frustration

and creative forms of debugging we found that the issue was in the indentation of the code. This error forced us to be cautious of our formatting but inspired an appreciation for clean and clear code (Sidenote: OCaml looks really pretty when formatted properly). Something else we realized as well was that Google was probably not going to be able to provide us with much guidance for the development of this project.

The next feature we implemented was displaying enemies. This was nothing special, just different art of a ship facing downwards. We figured the only thing we needed to know was who was an enemy and who was a player. In addition, the only thing we needed to know about these individual ships were their coordinates. Since we already had the player stored as a y-axis coordinate, we decided to store the enemies as a `List` of enemies. This appeared to be the best choice in order to retain the functional style of OCaml as well as the benefits of functional code. We then easily displayed three ships onto the screen at the top of the screen with random y-axis points.

Our next task was to allow our player to be able to shoot at enemy ships to ensure their safety. Upon agreeing that the most visually appealing choice would be to load an image for the bullet, as opposed to a traveling pixel, we decided to tackle our next issue. We needed to allow the user to fire a bullet at the click of a button, literally. We went with a gaming standard and binded the shooting event to the spacebar. After successfully implementing the logic to display a bullet after hitting the spacebar, we started to update the bullets position by moving it upwards a certain amount per frame. Soon after creating a working bullet we realized that we would have to keep track of several bullets, assuming they would attempt to shoot more than once. We attempted the same approach we did with the enemy ships and stored all fired bullets in a `List`

as well. On every spacebar hit we store a new bullet in our `List` and on every frame we offset the y-axis position of the bullets by a certain amount upwards on the screen.

With enemies and bullets on the screen now, we needed to be able to detect collisions. This proved difficult to accomplish, due to our actors being images and not physically drawn images with areas that reflect the actors shapes. Our Macgyvered remedy was to test for collisions each frame by testing the positions of all bullets and enemies and placing rectangles behind all of them. These rectangles allowed us to define just how much of the image can be considered the actual enemy ship and providing us some control over the difficulty of the game. We then used these rectangles to test for collisions with a Reprocessing function. If there is an intersection between enemy rectangles and bullet rectangles then we can safely remove those ships and reward the player with a point. The same logic would later be applied later on to detect collisions between the enemy ships and the player.

When attempting to display the score of the player we also encountered a set back. We displayed the correct points of the player based on how many enemies were destroyed but since our background was a dark blue we could barely see it. Because of this we decided on making the text white. After researching the docs we found that this was impossible to do. The correct way to display text of different colors is to import a ".fnt" file. This isn't necessarily a font file. This file matches the coordinates of a section in a sprite sheet with their corresponding ascii character value. This way, when the program requests the letter 'A' it opens this file and displays the sub image of the sprite sheet which corresponds to letter 'A.' After going through the process of matching the coordinates with their appropriate ascii character code we found that there was a tool for this. It was frustrating given the work we had already done in order to export the fonts

from the spite sheet but it was also a relief. Because of this we managed to easily interchange fonts much quicker and find the best that suited the visual theme of the application.

Overall despite these sets of frustrations along with many more, it was a very interesting and eye opening experience. Often times as students we rely on internet to help us overcome confusing errors. This time it was different. It is very fortunate that we had the chance to work together as it produced the best possible result. We are very proud and have found a new appreciation not only for game development but also for teamwork. We hope if possible to somehow give back to the community at some point in the future so other students who take this class experience the journey we did as well.