Naomi Chiu and Hadleigh Nunes
Mini Project 4 - Interactive Programming

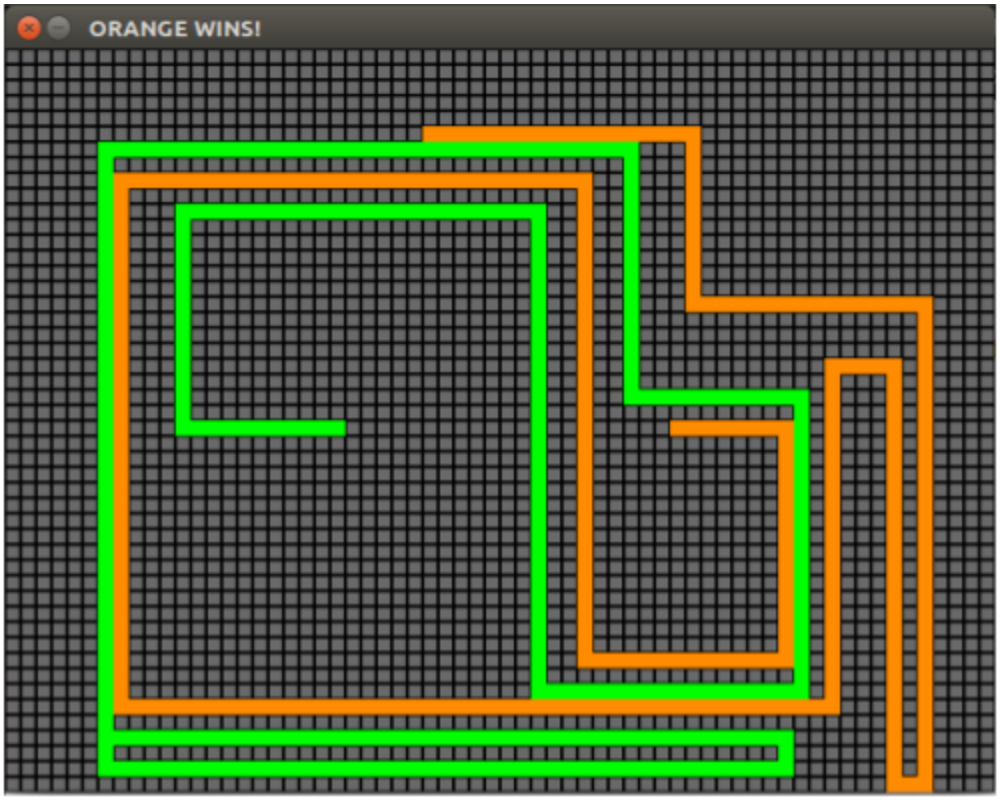
A Tron Remake

Project Overview

For our project, we created a remake of the classic arcade game Tron. Built as a two player survival game, our game incorporates all of the elements found in the original game including the playing field grid, brightly colored players and paths and the quintessential collision detection.

Results

Our final product is a simple but well-polished remake of Tron. While we did not have enough time to implement additional features to modify gameplay, we managed to mimic the original gameplay very well. We were able to create a multiplayer interface that allowed two players to play at the same time on the same console and control their respective avatars. Based on the framework we created, it also would not be a big stretch to add more players to the mix.

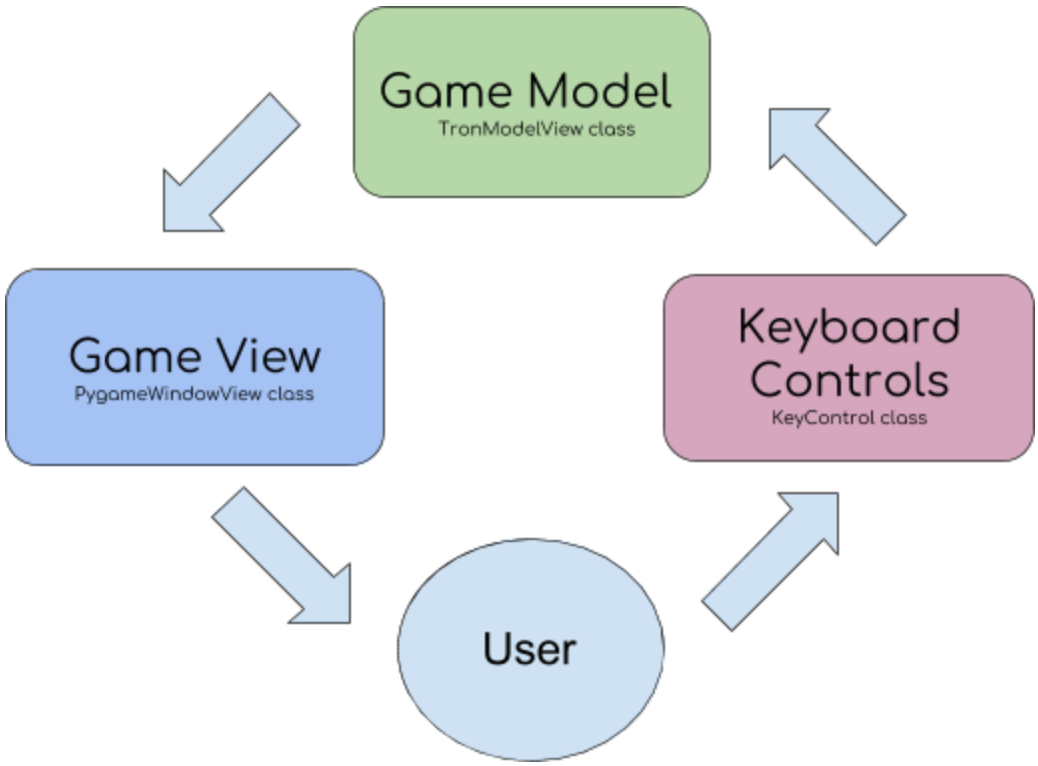


A screenshot of our game after a completed match. In this particular round, the orange player won as the green player crashed into their own path in the lower left hand corner. When a match ends, the pygame window caption displays who won.

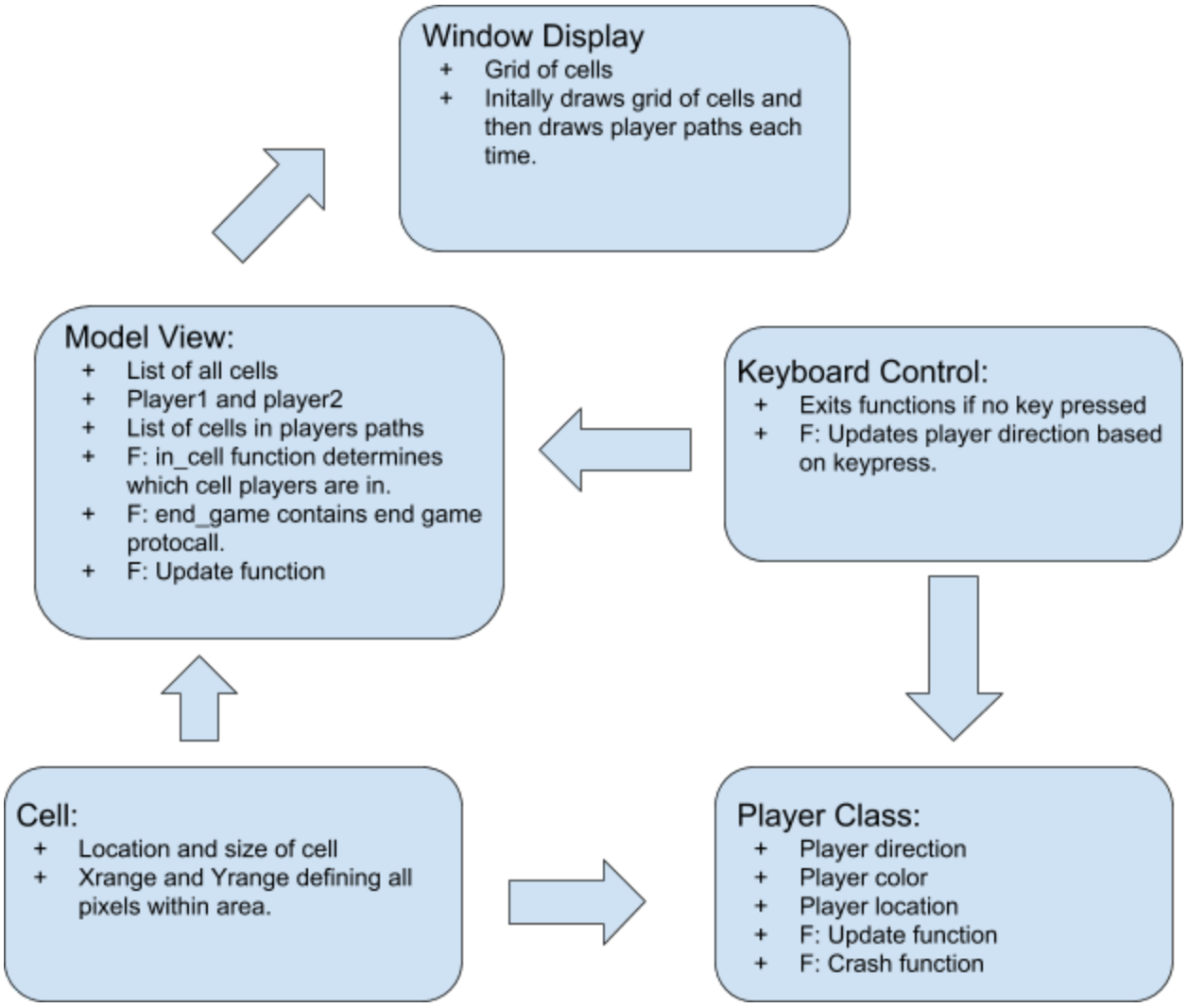
True to original gameplay, we also achieved crash detection that would sense if a player hit the walls surrounding the play area or if they ran into one of the paths left behind by the players. If a player crashed, the game would stop and their opponent would be declared the winner. We also achieved a restart loop that would allow players to start a new game after one ended by simply pressing the spacebar rather than closing the pygame window and rerunning the code.

Implementation

For our game we implemented a Model View Controller system for user interface. For each of these components, we created a new class with the model containing all of the happenings of the game, the view displaying what happens within the model and the keyboard controls affecting what happens within the model. In our implementation, we started with the model and the view and controller called upon the model to influence then display its contents.



Within the game model, we created additional classes for the players, the cell display and the cells themselves. The player class stored the position of the player as well as methods for displaying the player, changing the player's direction and detecting if the player crashed into the wall. The cell view class was used for initializing displaying the grid while the cell class was to define the cells that was used for storing the paths of the players.



When implementing the loop to allow repeated plays without closing the window, we chose to not put the main loop within the model as a method. While we had originally implemented it as a method, it prevented us from running a nested while loop that would allow for repeated plays. For that reason we chose to code in the main loop outside of the classes although it looked less elegant.

Reflection

Though we didn't go much above and beyond our MVP for this project, we are very happy with our final product as well as the manner in which we accomplished it. In the beginning we were very confused about how to begin to implement everything that we

wanted but we were able to figure it out through research and help. We also feel that the project was extremely successful in terms of our learning goals. Hadleigh really felt like she gained a grasp of pair project planning and program implementation that she might not have had time to really build had the project felt more rushed. For Naomi, she is very excited about the success of the pair programming as we figured out a method that worked for the two of us very well. In the beginning, we had tried to work on the code at the same time together but oftentimes it ended up that we generated two versions of the same thing, with only one of them working and the other one being scrapped. Neither of us felt that that approach was particularly helpful and after some analysis on our individual working methods, we decided to work separately and pass the code back and forth between the two of us. As Hadleigh worked better during the day, we would discuss what we wished to accomplish and Hadleigh would begin the code for it. After she did some work, she would push the code to Naomi who would look over what Hadleigh had done and either continue to build on it or edit it to work more smoothly. We also worked separately on different aspects and then combined our code. For example, Hadleigh worked on collision detection while Naomi wrote the code to allow for repeat plays. Both of us feel that the project was well scoped which allowed us to take advantage of the opportunity for self directed learning.

Looking back, we think it would have been cool if we could have developed an CPU to play with a single player but we just did not have the time for it. In terms of improvement, we think that we could have worked more on graphics to create a more visually appealing game but we are satisfied with the simple arcade-like display. Going forward, we would like to explore more inheritance which we did not use in this project.