Amber Duevel

Sidney Williams

Jordan Persinger

Monopoly Final

One goal for this project was to break down the game of Monopoly into four different components. With these four different components we added someone new to see if it would affect the amount a times a player landed on a position. In our first round we just worried about the dice. This was the simplest version of Monopoly. We found out that each position on the board was equally likely to be landed on compared to the other spots. We only ran the game for one hundred rounds so some spaces were landed on more frequently than others. We didn’t run into any problems with this round of code. In out second component of the project, we added in the double rule. With this rule players could roll doubles up to three times. If they rolled doubles three times in a row they were sent to jail. Due to us changing the position of the player now we suspected that jail would be landed on more frequently than other spots. We also thought that there would be a higher count for each space because players can now roll more than once in a row. We only ran the program for one hundred rounds so there were a couple changes with the spaces but not drastically. This code ran smoothly because we rewrote the dice function to account for doubles and going to jail. The rest of the code stayed the same and we just called a different dice function. The only problem that we faced when running this code was calling functions within functions. In the dice function we had to call get out of jail and double check function. Connecting all these pieces together was a little difficult but do able.

2. A screenshot of a computer

Description automatically generated with low confidence 3. A screenshot of a computer

Description automatically generated with low confidence

With our next components of the project, we added cards and money. In the graph we just added the community chess and chance cards to the player function. If a player landed on a chance or community chess card, they would randomly select a card and then do what the card said. In this part we only focused on the moving cards. If a player picked up a card that had them move to a new position, then we had a function that would change their location to that position. Connecting the cards function to the player turn and move function worked pretty well. The problem that we ran into was selecting a card. We ended up creating a new data frame that only had what was written on the cards. This had to occur because we couldn’t use the select command with the original data frame that we had. Then we were able to match those cards to a position and write a bunch of if statements to move the players. With the last component of the project, we introduced money. This function would add or subtract the amount of money based on a community chess card, chance card, or a space on the board. This is where we ran into the most trouble with the project. We had a bunch of function declared and called them based on if statements. When calling the money function, it was subtracting or adding an amount of money but wasn’t updating the player data frame that held all the data. Somewhere in the program it was getting reset to $1500 at the end of all the rounds. We solved this problem by splitting up the money function, so it was in different parts. This way we didn’t have a bunch of nested if statements in one function. By fixing this problem we were able to update the money amount in the data frame for each player.

4. A screenshot of a computer

Description automatically generated with medium confidence 5. A screenshot of a computer

Description automatically generated with low confidence