Isaac Phillips

CS:4980

Denise Szecsei

12/6/2023

UNO Simulator

For this project I wanted to make one of my childhood favorite games UNO. In the initial pitch I had the vague idea of a UNO simulator based off some of the code used for the blackjack simulator. However, professor was concerned with the time we had for the project and suggested I did color recognition with Cosmo. Yet, I was determined even with the time constraint to get a working UNO simulator.

My initial goals were

I took the time constraint into consideration but by the time the formal presentation came around I had a skeleton of how I wanted the simulator set up, going into Thanksgiving break. All the skeleton really was, was just the beginning works of what would become the deck creation, some of the logic written out, and parts of the blackjack simulator. This includes the imports such as: numpy, pandas, random, and matplotlib. As well as aspect of the simulator and architecture of the data frame used.

Now, for the deck creation, I looked up the cards of UNO and tried to decide what cards I wanted to include in UNO. Previously, I gave some thought about how I would do the reverse, skip, wild and plus 2/ plus 4 cards. I wanted to use all 108 cards since it wouldn’t be UNO without all of the special cards. Each color has one 0, two of each number, numbering 1-9 and 2 of each special action cards. The action cards being the reverse, skip, plus two, wild, and wild plus four. But the use of a reverse and skip would just be matching colors and or in this case symbols to be able to play them. The wild are more just like a free play to get a card out of your hand if you have nothing else to play. However, the wild does not change the color of play, it is only something to play. For how I wanted the cards to be formatted, I thought about the network and how we had the cards for blackjack formatted and decided the split function later on to be able to get both the color and the number or action in two separate indexes in the list. An example of what a card in the deck looks like is “Green;Reverse” and later on after splitting in the list of cards it would look like “[[“Green”, “Reverse”], [“Red”, “7”]…]” and so on for the 7 cards in hand.

After the deck creation I needed to shuffle it. Initially I tried to use the random.shuffle() with the random import, shuffled deck had more greens than any other color. Instead of thinking that it was a problem with the deck creation, I thought it was a problem with the shuffling. So, instead of fixing the deck creation, I decided to make a shuffle function to be able to get the cards shuffle more efficiently so there weren’t as many greens as possible in what was visible to me in the deck. Once the deck was thought to be properly shuffled it was time to deal cards or get ready to deal cards to players. I liked how we did it for the blackjack simulator, but I wasn’t too sure how I would do plus two’s or plus fours with that strategy. Thinking about the function of the game, I wanted a function the I could call that would draw cards for me when it came to the actual running of the game. The drawCards function draws seven cards from the deck and creates a list out of those seven cards for each individual player. In UNO, you are able to have 2-10 players but to simplify things a little for myself I set the player count to 3. I also decided to add an addition to the draw cards function to change the ‘Wild’ into ‘Wild;Wild” so that the split would be much easier and I can figure out if it’s a plus four later on using this. Once the wilds are changed though I split the cards to be able to manipulate them in the next section.

The logic section is where I’ve had the most trouble with the work. Having each player only allowed to play one card or draw one card per turn, while making sure that the card that the player wants to play shares an attribute between the cards. Before any of that though it starts off with splitting the players list down to the first player, then looking at the hand of the first player, iterate through each card to see if any matches occur with the current center color or center value. I then take into account if the center card value is equal to “plus four”, and if so drawCards(4). I then check for plus two and do the same. Now if the center card is a wild, the player is able to play any card from their hand. And as much as I would like to go into the logic of which one is better to play, for simplicity I just play the first card of the hand. For the color and value logic, I first look to see if any cards match the centers color and if a card does it plays it. If the colors don’t match, then we check for if the “value” matches. I say “value” because this can be anything from a number to a reverse, as long as they are matching it will play it. And when I play, I leave my wilds and wild plus fours last to play if nothing else. So, that’s what I did for this as well, it won't play wild plus four if there is a wild in hand, but it won't play a wild unless nothing else matches. Finally, if all else fails, the player draws one.

The simulation section is next, and I didn’t change much to it from the blackjack simulator just few variables were added and changed it from blackjack to UNO basically. In the blackjack simulator it draws a card first for the player, then the dealer and another for the player. However, in UNO, the player starts with seven cards, so we use the drawCards function again to draw the 7 cards for each player. After the players have their cards then the center card is placed. It is also important to note that this is where I shuffle the deck as well; before the players draw, and the center card is drawn. The variables I want back are the color dictionary that counts how many times a color is played, how many cards were picked up, how many cards were played, and what cards where played. I then begin the data frame creation and store it in a csv.

When I first ran the simulator, it ran but immediately came up with errors. The first being the green issue that was previously mentioned. I finally found the real problem to the shuffling. In the deck creation I didn’t iterate through the list of colors properly, so I was getting 3 greens to every blue, red and yellow. But that was an easy fix, what I was focused on is that I was able to see the cards move through the simulation and start to go through the logic. But it has been stuck at the logic ever since. The logical errors that pop up have to do with the amount of card there were in the shoe. I originally had 3 decks of cards, but it always ran out. I kept increasing until I got to 9 decks and even now it still runs out of cards but for better reasons. As I increased the number of decks, I saw a little more with the cards. Now that they were right and there were enough of them, I was able to get to play with the players hands and see how they work with the center card. For the most part it would pop a card from the front of the hand and replace the center card with that. That is where issue number two comes in, and I still have a hard time with getting anything close to the right position. But the .pop(0) takes the first card, and I’ve done something wrong with the variable card\_pos because it is always just a few off and stop the program from running after a few games. So, to be able to get anything, I just pop(0) for now. Sometimes the logic works and pops the right card, but the wild cards are a different story. However, with the pop(0) I’ve been able to get a full game to run but still run out of cards in the deck. I have a while statement that should stop it when the shoe reaches 50 but it just ignores it for now. Yet, it runs the game fully and creates a really messy csv. I haven’t been able to model anything with the data because there are few wins. I’m not sure how this happens, there is only one place if statement to draw cards so I’m not sure what the problem is with that yet.

In conclusion, I was a little ambitious with my original goals and I didn’t really change them. I like to hold myself to a high standard and want to push myself to reach it regardless of how difficult it is. I wish I got into the color recognition a little like professor recommended however, I wanted to do more since this is my last college assignment. I am disappointed I didn’t get to finish it to be able to show you the final product, but it will still make for a cool little project for my resume once it is done. I have provided a few of the pictures of the code and some of the results from this project below.

(PS. Thank you for a good semester. I had you my first semester of college in intro to comp sci so it’s cool to have you in my last semester.)

A screen shot of a computer program

Description automatically generated

This is the deck creation I mentioned at the beginning.

A screen shot of a computer screen

Description automatically generated

This is the draw card and shuffle functions I talked about.

A screen shot of a computer program

Description automatically generated

This is the updated logic from Thursday to Sunday.

A screen shot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

This is a game where only 30 cards were played and the list of the cards above.

A screenshot of a computer screen

Description automatically generated

This screen shot shows that there were 80 cards played in one round and gives the list above of the cards played.

And this is what the center card throughout a game looks like.

A screenshot of a computer

Description automatically generated

Some decent variety, not what really what I’m looking for yet.