Johnathan Picek

John Rinderer

CPSC 24500-003 – Object Oriented Programming

October 25, 2021

Project Proposal Summary

Group Members:

Just me, Johnathan Picek

For my Project, I will build the card game UNO with a JavaFX GUI. I will implement the basic UNO rules found here: <https://www.ultraboardgames.com/uno/index.php>, <https://www.unorules.com/>, and <https://www.ultraboardgames.com/uno/game-rules.php>

Using an Object-Oriented Programming approach should help organize the design process, enhance security through data hiding and encapsulation, and reduce code complexity and limit duplicated code through inheritance and polymorphism.

I have described the number of cards in the table the table below:

|  |  |  |
| --- | --- | --- |
| **Card** | **Qty per 4 Colors** | **Qty Total** |
| 0 | 1 | 4 |
| 1 through 9 | 2 | 72 |
| Skip | 2 | 8 |
| Reverse | 2 | 8 |
| Draw Two | 2 | 8 |
| Wild |  | 4 |
| Wild Draw Four |  | 4 |
|  | Total | 108 |

I intend to do it as a 4-player game versus 3 computer-controlled opponents that follow a very simple strategy. If they have no playable cards, they draw 1 card and play it if it is playable. Whether the drawn card is or isn’t playable, this ends their turn. Otherwise (if they had playable cards to begin with) play a playable card, excluding Wilds, at random. Only use Wild Draw Four according to the rules, which is when no other cards are playable. Illegal Play of Wild Draw Four challenges will not be developed. Human player will be prevented from playing the Wild Draw Four illegally, so the computer opponents also won’t need to challenge them. The somewhat-rare Regular Wild could either be saved to be played last, to win the game, or included in improved AI. Color changes will be picked by remaining cards in hand (or random if no colors remain).

This AI can probably be improved somewhat. Randomly playing one of many playable cards probably isn’t a great strategy. Each card in a hand could be scored with a relative weight and the most optimal card is played. This could be optimized further but devising a strong opponent might not be preferred. I want the player to be able to win. Choosing to have the computer play worse might be better... Anyway, the game could eventually have difficulty options.

I will use an interface class named Playable with a method advanceToNextTurn() that each subclass of the Card class would implement. I think I can have NumberCard, ReverseCard, SkipCard, DrawTwoCard, WildCard, and WildDrawFourCard all extend Card class. For most cards, advanceToNextTurn() would just advance 1 Player (+1 or –1 depending on current turn direction). For Reverse, negate the turn direction variable, then advance 1 turn. For Skip, advance \* 2. Draw Two and Wild Draw Four Cards are essentially more powerful Skips since the player forfeits their turn, with the added effect of having them Draw as well. The Wilds both additionally allow the Player to change the “playable color” variable which the Human player will need a UI elements for and the computer players will need AI designed for.

Human player can attempt to play any card, but illegal moves will be rejected. Human player will also need to hit a separate highly visible button that says “UNO” upon playing their last card. If they fail to do so, every subsequent turn by a computer player will have… oh, maybe a 30% chance of that player “noticing” and catching the player, which will penalize the human player, making them draw two cards. Computer players will automatically say UNO properly (to never incur the two card penalty and remove the need for additional UI elements and controls to “catch” computer opponents).