Program Design Methods and Intro to Programming Python Final Project: **UNO!tastic**

Student Information:

Ardelia Shaula Araminta / 2440065163

Class Information:

Class: L1BC
Lecturer's name: Ida Bagus Kerthyayana / D5757

Binus University International
School of Computer Science
2021

Introduction

In my first semester studying Computer Science in Binus International, it was a bit overwhelming and spending most of the days learning and recalling things taught in the class, but it is fascinating to recall some of the lessons that I have learned before, even though I haven't study much about Python. We were taught from the basic programming and explore deeper with different libraries that the Python can be used with. At the end of the semester, we were given a final project assignment where students can explore and create a program with the idea of their own with specific requirements. Since there were so many ideas in my mind, at first, I want to do a Flask Project to create a website but I felt that I was the most comfortable using Tkinter as my external library and I've created some of different program using Tkinter before and it's kind of unique to create a game using it instead of Pygame and my idea is to create UNO card game.

Th project officially started in 18th December 2020. Visual Studio Code was chosen as my IDE for this project. Also, as this project's nature is open source, my initial commit was uploaded to my GitHub account, https://github.com/ardeliaraminta/UNO-tastic .

Project Specification

The purpose of the program is to recreate the Original UNO card game that have a multiplayer mode allowing 2- 4 players to take turns to play in one device. This program will follow the logic of the original UNO card game, which means when the game starts, each of the players will be given 7 cards in their hand and make the players take turns matching a **card** in their hand with the current **card** shown on top of the deck either by color or number (discard pile). Other special cards such as Skip, Reverse, +2 (Draw), +4(Draw), Wildcard are also included in this program. The game ends when one of the players has 0 card left in hand and win the match.

I chose to write this program because I have always enjoy playing UNO card game because of its unique card game mode with additional special cards that I haven't ever seen in other card game. It has always been a good reminder of my childhood, and this is the main reason why I really want to do this.

More Specification on the project:

Input of the program:

- Chosen card in player hands to be placed in the top of the discard pile
- Colour picked in Wildcard
- Draw card from the pile
- End player's turn after drawing a card from the pile

Output:

- The game starts with the numbers of players chosen
- The 7 cards spread in each of the player hands being displayed in the screen
- Winner of the game

Project audience:

- Someone who enjoy playing card game
- Someone who wants to learn Tkinter

Solution Design

1. Players Selection Panel
2. Name Entry Panel
3. Main Game Screen
4. Wildcard colours selection
5. Next Player's Turn panel
6. Results

Players Selection Panel

Players Selection Screen will display the launch of the initial menu of the program which directly ask the player to choose how many people will play the game. There is one button available in the panel “Okay” that will bring the players to the exit the panel enter name panel if the player chooses between 2-4 players, but if they don’t choose, it won’t proceed.

Name Entry Panel

In this panel, players will be able to enter their name. This is a simple panel, with a button “Enter” and bring the players to the main game screen. This information will later be used in the main Game Screen.

Main Game Screen

In this screen, player will start to take turns to play. This is important to know which turn it is during the game. While they play, most likely the 7 cards spread contains special cards that will affect the loop of the game such as:

- Skip – When a player uses this card, the next player’s turn will be passed to the third player, if 1 vs 1 then it will be that player’s turn again.
- Reverse - When a player uses this card, the direction will be reversed.
- +2 Draw - When a player uses this card, the next player will withdraw 2 cards from the draw pile.
- +4 Draw - When a player uses this card, he/she can choose which colour the discard pile would be for the next turn and the next player will withdraw 4 cards from the draw pile.
- Wildcard- When a player uses this card, he/she can choose which colour the discard pile would be for the next turn.

There are four buttons available in this screen, including Draw Card, Left and Right Arrow and End Turn button.

1. **Draw card Button** is used to draw card from the draw pile when none of the card in hand is playable, if the card’s taken can be played, the player can get rid of the card, if not, the player has to end their turn without putting anything on the top of the discard pile.

2. **Left and Right Arrows Buttons** are used to navigate through the page and determine which page of cards displayed in player hands. In each page, it will show 5 cards in hands and since there are 7 in the beginning of the game, the rest of the cards and additional cards (cards drawn later in-game) are displayed in the second page.

3. **End Turn Button** are used end the turn after the player drew a card when they no cards that are playable and decided not to use the cards drawn from the draw pile. However, if the player uses his/her card in hands, it will automatically proceed to the next player’s turn panel.

Wildcard colours selection

This screen displays the colour selection of Wildcard or +4 card which the players have the choice to pick one of the colours between red, yellow, blue, and green to their advantage. If wildcard pops up the player can only pick a colour but if they Draw four (+4) cards is played, not only the player can choose a colour but the next player will draw 4 cards from the draw pile.

Scenario 1:

Player “A” has 2 cards left in her hand, a wildcard and a yellow 0 card, unfortunately the front/top card of the discard pile is blue. She has no choice but to use the wildcard and choose yellow to win the game. The opponent can counter this by placing another 0 with different colour but if they don’t have a card to play or decided to skip, player “A” can simply win with playing the last card.

Next Player’s Turn Panel

This panel is created as a transition panel from one player’s turn to another. Since the program is planned to be implemented in one device, the use of this panel to avoid the opponent to look at the cards in players hands and vice versa. There is one button, “*player name*, it’s your turn” to proceed to the next player.

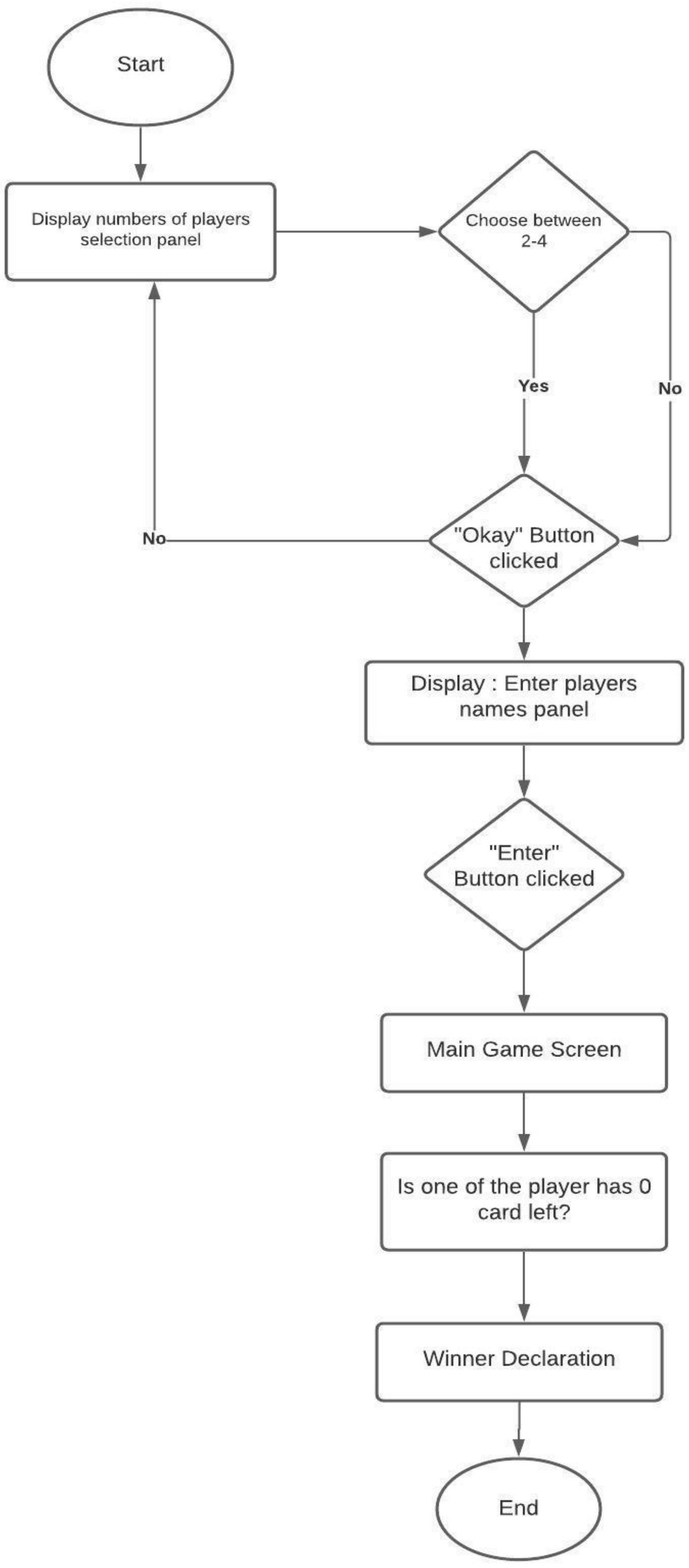
Results

In the results panel, it will delete everything on the screen while not shrinking the window and show the winner amongst the players.

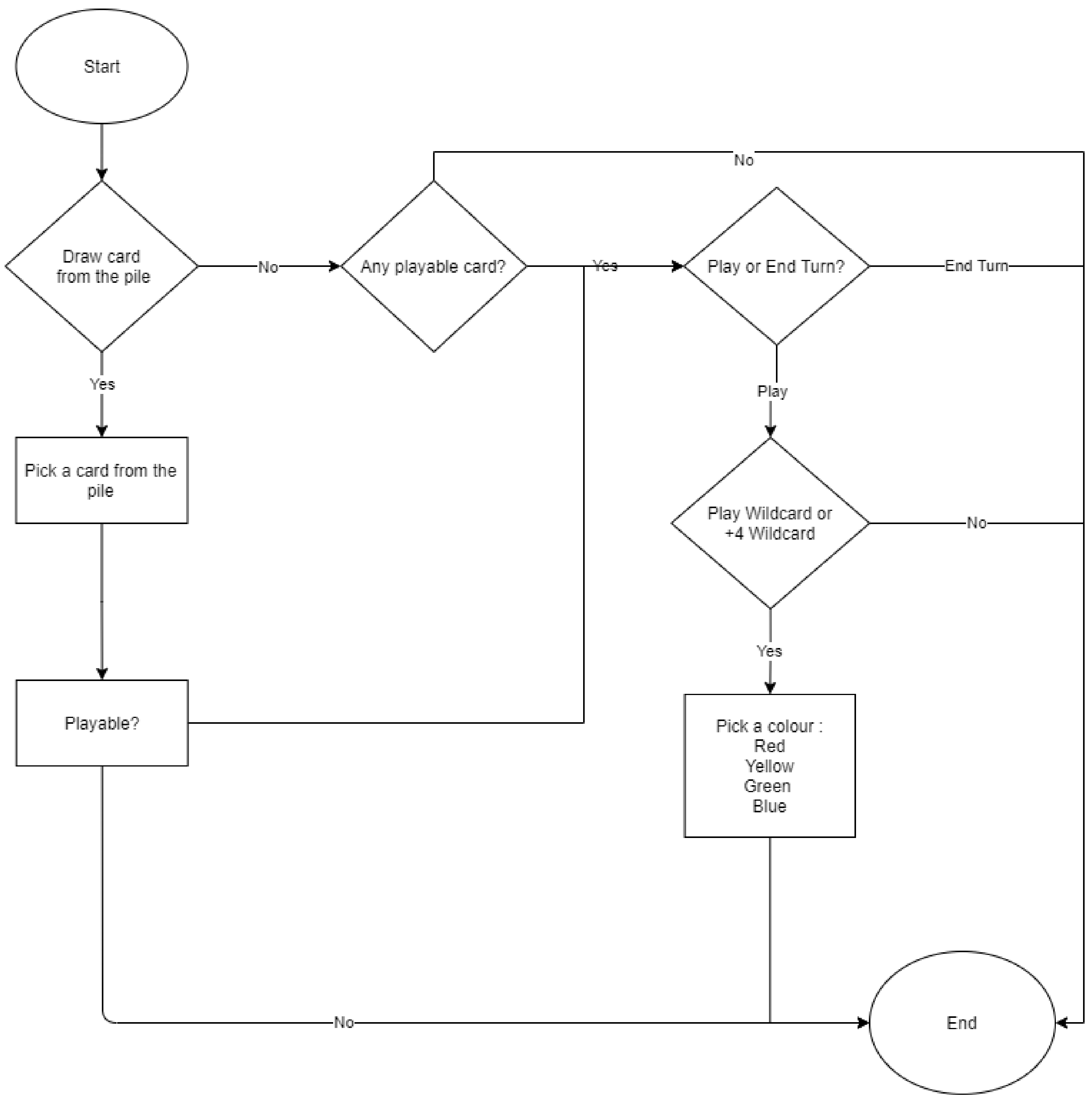
UNO!tastic uses Python 3.7.9 with the following as external library:

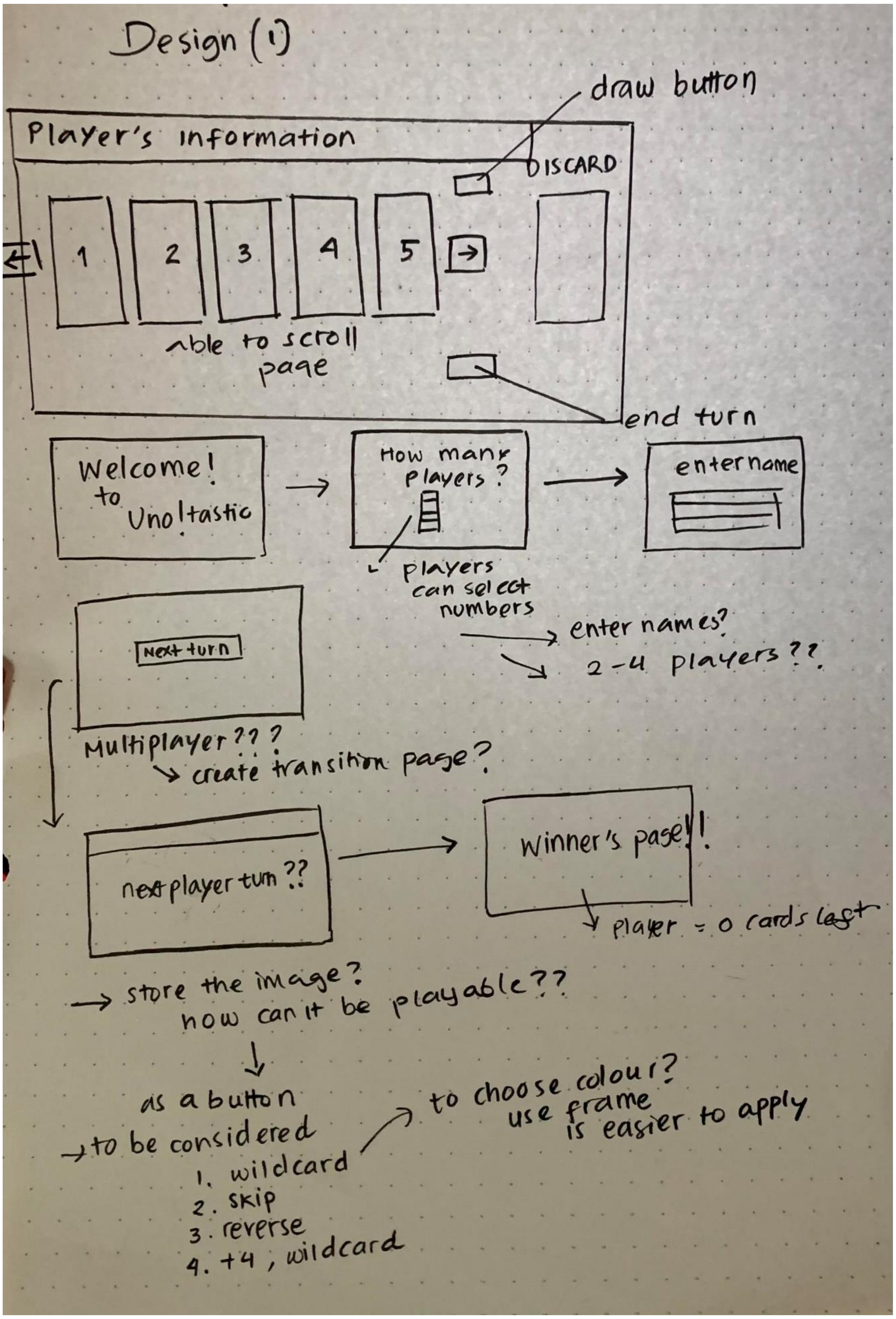
- **Tkinter 8.6**

Project Hierarchy Chart


Start
Display numbers of players selection panel
Choose between 2-4
Yes
No
"Okay" Button clicked
No
Display : Enter players names panel
"Enter" Button clicked
Main Game Screen
Is one of the player has 0 card left?
Winner Declaration
End

How a player takes his/ her turn





My initial design of UNO!Tastic

Lessons I have Learnt

1. **Implementation of Class in Tkinter**

- At first, I was struggling to implement Class in the program because I was so used to using only def functions for creating a project using TkInter.

2. **Useful GUI Widget Classes within TkInter :**

- Label : is a simple widget which displays a short piece of text or an image
- PhotoImage :
- Frame : is a container widget which is placed inside a window, which can have its own border and background
- Button : when the user clicks on a button, something should happen.

3. **The Random Module :**

This model is able to produce random shuffle cards for the seven cards in playerhands and drawpile.

Implementation and Explanation of the Code

Basic Idea

- ➔ Basic buttons to navigate the between the screens

In order to detect the cards based on their colours, numbers and functionality I define a simple way to solve this, in the Class Card. The basic idea is to store it as an integer and according to their number or type of special cards.

```
import random
from tkinter import *
```

import random to shuffle the cards given in each player hand, shuffle draw pile.

from tkinter import * - importing everything from tkinter library

Class Card:

```
class Card:
    def __init__(self, color,cardnum):

        self.color = color #stores color of the card
        self.cardnum = cardnum #stores the card based on their funct
ionality and numbers
        self.img = PhotoImage(file=str(cardnum)+" "+str(color)+".png
") #stores the image of the cards
        self.but= Button(area_toplay,image=self.img, command = self.
useCard, cursor = "heart") #create a button using image of the card

    def useCard(self):
        if self.cardnum == discard.cards[0].cardnum or self.color ==
 discard.cards[0].color or self.color == 4: #if the card can be play
ed
            playCard(self) #play that card
            endClear()
```

Init (self, color, cardnum) function:

Here the properties of card are made, which includes cards color, card number. **self. color** is a property that the card as color as integer, 0 = red, 1 = yellow, 2= green, 3 = blue. Moreover, **self. cardnum** will store the number or type of special cards as 10 = skip, 11 = reverse, 12 = +2, 13 = wildcard, 14 = +4 wildcard.

Additionally, the card images and playing button are defined within the function.

- ➔ ***self.img*** use the color and number of cards to make the Photo Image file from the corresponding .png file of the same name. For example: Red is represented with 0, 2 0.png means red 2 card.
- ➔ ***self.but creates*** a button using the image of the card and created a heart cursor to play.

The **def useCard()** function activated when the card is playable and it will automatically end turn after that particular card is played and lastly it calls out endClear() function which clear the board when player end their turn.

Class Deck:

```
class Deck:

# to make a deck and empty deck if the card is available == 1, create a fu
ll deck and if available == 0, make an empty deck
# 4 colours ( red, yellow, green, blue) except wildcards and make 2 copies
 of each cards
# create deck
    def __init__(self, available):
        self.cards = [] #create an empty deck
        self.available = available
        if self.available == 1: #if available 1, make a full deck. if avai
lable = 0, make an empty deck
            for i in range(4): #for everycard except wildcard
                self.cards.append(Card(i,0)) #make 1 zero card
                for twice in range(2): #make duplicate of card of the card
s below
                    for special_cards in range(0,13): #number cards 0-
9, skip, reverse and +2
                        self.cards.append(Card(i, special_cards))
            for w in range(4): # make 4 wildcards and 4 +4 wildcards
                for wildcard in range(13,15): #13 - wildcard, 14- +4 wildc
ard
                    self.cards.append(Card(4,wildcard))
                random.shuffle(self.cards) #shuffle the deck
```

Init (self, available) function:

- Here the how the deck is made, self.cards = [] will create an empty deck of UNO.
- I use for i in range(4) for each of color except “wild card” make a zero card.
- Duplicate each card from 1-9 and 10-14

Try and except function()

- Removes and returns the value of the cards if card is empty, call out RefreshDrawCards() and returns it.

Function works with buttons

```
def endButton(): # activate when End Turn button is clicked
    global counterturn
    endClear() #clear the screen for next turn
    counterturn = (counterturn + direction)%len(playernames)  # advance to
 the next turn
    showNext()

 # when the Next Turn is clicked
def nextButton():
    nextbut.pack_forget() # remove the Next Turn button
    nextTurn() # create a setup the next turn for the next player
```

- **Def endButton(), Def nextButton() function**

In this particular function, I have used **global function** since **counterturn** variable is defined outside the function, **global variables** are where all the variables needed to be accessed everywhere from the program are stored. The **endButton()** function is to activate the frame for the next player’s transition page, since it is a one device game, there should be an indication that “ this is the next player’s turn” page instead of a direct display of the next players cards in hand by calling out showNext() function.

The def **nextButton()** function is for clear the display of the showNext() and proceed to the next player’s turn setup by calling **nextTurn()** function.

```
def leftArrow(): # when left arrow is clicked
    global pagenum
    pagenum -
= 1 #change the cards in playerhands on the left page by one
    refreshCards()  # create the new page display up


def rightArrow(): # when right arrow is clicked
    global pagenum
    pagenum += 1  #change the cards in playerhands on the right page
 by one if there are more than 5 cards in hand
    refreshCards() # create the new page display up
```

Def leftArrow() and Def rightArrow():

I used global function as pagenum is defined outside the function, and the functionality of these arrows are when the player clicked these buttons, they will get a display refresh the cards on screen depending on which page the player is on.

```
# when the color picker button is clicked
# remove the color choices on the screen
def redClick():
    discard.cards[0].color = 0 #red
    clearChoice()

def yellowClick():
    discard.cards[0].color = 1 #yellow
    clearChoice()

def greenClick():
    discard.cards[0].color = 2 #green
    clearChoice()

def blueClick():
    discard.cards[0].color = 3 ## blue
    clearChoice()

# clear the color choices from the screen
def clearChoice():
    colorinstructions.pack_forget()
    for x in unocolours:
        x.pack_forget()
    showNext()
```

- **The def clearChoice() function**

This function is use to clear the choice of 4 colours in wildcard or +4 wildcard on the screen.

It will remove the **colorinstructions.pack()** and **unocolor.**pack which is widget that ask the players the choices for wildcard, and the choices of the colours and call **showNext()** function to proceed to the next turn.

Main Functions

```
def Setup():
    spacer_label1.pack(side = TOP)  # spacers to make the screen look nicer
    number_label.pack() #this will ask how many players
    spacer_label2.pack()
    playerselection.pack() # listbox to select how many players playing
    choices = ["2", "3", "4"] # choices for number of users
    for i in range(len(choices)):
        playerselection.insert(i, choices[i])
    spacer_label3.pack()
    numberokaybutton.pack()  #display okay button
```

- **The Setup() function**

This function is the initial start of the GUI that will display and ask how many players are playing the UNO game by given selection from 2 up to 4 players**. (Number_label.pack())**

The choices are in a list and to proceed to the next screen, there is an “Okay” button when and the user has to clicked it.

```
#after clicking okay to a selected amount of players
def Confirm():
    global listofentries, numberofplayers  #make these two variables global
    spacer_label2.pack_forget()
    spacer_label3.pack_forget()
    numberofplayers = [playerselection.get(i) for i in playerselection.curselection()] # the number of players = the number from the listbox
    numberofplayers = int(numberofplayers[0]) #convert from string to int
    listofentries = [] #make a list for the different text entry boxes
    numberokaybutton.pack_forget() #delete all number of user’s pieces
    playerselection.pack_forget()
    number_label.pack_forget()
    name_label.pack(side=TOP) #ask names of users
    for i in range(numberofplayers): #create one text entry for each user
        listofentries.append(0)
        listofentries[i] = Entry(area_toplay)
        listofentries[i].pack()
    confirm_names_button.pack() #enter button
```

- **The Confirm() function**

This function runs after clicking okay button of by deciding selected number of users are playing. The player numbers selection is removed from the screen using .**pack_forget(). In this function ,** I have used **global function** since **listofentries and numbersofplayers** variables are defined outside the function. **Name_label.pack()** ask the name of the user and then create one text entry for each user depending how many players are playing, for example : the 4 users decide to play, the entry box that are available based on the range of numbersofplayers. To convert the numbersofplayers from string to integer I used **int()** and then create an empty list for the entry boxes for names to be entered and created “Enter” to proceed.

```
#confirm numbers of players
def EnterName(): # if the input names are confirmed
    for i in range(numberofplayers): # repeat for as many players there are
        playernames.append(listofentries[i].get()) # add to the list playernam
es each entered name
        listofentries[i].pack_forget()  # delete text entry
    confirm_names_button.pack_forget()  # delete other aspects
    name_label.pack_forget()  # remove everything from the screen
    spacer_label3.pack_forget()
    Begin()
```

- **The EnterName() function**

  This function runs to confirm the input names as well as the number of players entered and get the name of playernames entered and add it to the list. Then, it will delete and remove everything from the screen and call **Begin().**

```
 # starts the game
def Begin():
    SevenCards()
    createDiscard()
    nextTurn()

window = Tk()
```

- **The Begin() function runs to display the seven cards in hand, create the discard pile and next turn button.**

```
def SevenCards():
    for i in range(len(playernames)):  # make an empty deck for each
 player in the game
        playerhands.append(Deck(0))

 # each player will get 7 cards in their hands
    for i in range(7):
        for x in playerhands:
            x.add_front_deck(drawpile.arrange(0))
```

- **The def SevenCards() function**

This function is to spread 7 UNO cards in the players hands in the beginning of the game, first it will check how many people are playing using **for i range(len(playernames))** and create an empty deck for each player and then each of the players will have 7 cards in their hand.

```
 # makes the discard pile at the beginning of the game
def createDiscard():
    while True: #start the game by creating discard pile
        discard.add_front_deck(drawpile.arrange(0)) #continuous loop
 of adding the cards
        if discard.cards[0].cardnum <= 9: #until the 1-
9 number cards is on top the process continues
            break
```

- **The def createDiscard() function**

This function is to create discard pile at the beginning of the game and as the player take turns the discard pile will keep updated, means the played cards on each turn will be place on the front of the discard pile, therefore I use **while-loop** to create a continuous of adding the regular numbers cards of 1-9 if not, it will be terminated using **break statement**.

```
#draw card
def drawCard():
    playerhands[counterturn].cards.append(drawpile.arrange(0)) # adds card
 to end of player's hand
    drawbut.pack_forget() # remove draw button so they can't draw more tha
n one card in a turn
    refreshCards() # refresh the cards in hand so they can access the new
card they have just drawn
    endbut.pack() # show the Next Turn button
```

- **Def drawCard() function**

Append the card that is drawn from the draw pile, this function is to add card to end of the player hands when the player decided to draw card from the discard pile. In this function, draw button is removed after a card is drawn, that leave the player to play the new card or just end their turn, this calls the **end.but pack().**

```
# it will run when a card that can be played is clicked on
# if the player now has no cards in their hand run win()
def playCard(card):
    global direction, counterturn
    cardindex = playerhands[counterturn].cards.index(card)
    endClear()
    discard.add_front_deck(playerhands[counterturn].arrange(cardindex))
    if len(playerhands[counterturn].cards) == 0:
        win()
```

- **The playCard() function**

This function is running when the player is playing a card or decided to get rid of a card from their hand. In this particular function, I have used **global function** since **counterturn and direction** variable is defined outside the function. The card index searches card in player's hand and add that card to the discard pile from the player hands. **endClear() function** is to clear the board of the game. If function here is used to declare a winner, if player has no cards in the hand run the **win()** function.

If, elif and else loop is used here to define the loop of the game when special cards ae played such as skip that will skip the next player's by adding two to the counterturn, reverse will affect the direction of the gameplay so direction * -1 means it will reverse the direction of the turn.

Moreover, if the card is +2, next player will draw 2 cards (**add to the end of the next player's deck**) using append.

If the card is 13 which is wildcard, the player only has to choose the chosen colour to their advantages and this calls out colorPicker()

If the card is 14 which is +4 wildcard, the player chooses the colour and the next player will draw 4 cards (**add to the end of the next player's deck**).

```
# this is the function of special cards that defne the loop of the game
    if card.cardnum == 10: # skip
        counterturn = (counterturn+(2*direction))%len(playernames)
        showNext()
    elif card.cardnum == 11: # reverse
        direction = direction * (-1) # reverse the direction of the turns
        counterturn = (counterturn + direction)%len(playernames)
        showNext()
    elif card.cardnum == 12: # + 2
        for i in range(2):
            playerhands[(counterturn+direction)%len(playernames)].cards.ap
pend(drawpile.arrange(0)) # next player draws 2 cards
        counterturn = (counterturn+(2*direction))%len(playernames) # skip
that player
        showNext()
    elif card.cardnum == 13: # wildcard
        counterturn = (counterturn + direction)%len(playernames)
        colorPicker() # allow player to pick the color of their wildcard
    elif card.cardnum == 14: # + 4
        for i in range(4):
            playerhands[(counterturn+direction)%len(playernames)].cards.ap
pend(drawpile.arrange(0))
        counterturn = (counterturn+(2*direction))%len(playernames)
        colorPicker() # gets player to pick the color of their wild draw4
card
    else: # number cards
```

```
       counterturn = (counterturn + direction)%len(playernames)
       showNext()
#transition of players' turns and so the next player can start their
 turn
def showNext():
    nextbut.config(text = playernames[counterturn] + " , please clic
k to take your turn.")
    nextbut.pack()
```

- **The showNext() function**

This function basically the transition button that allow the players to know when it's their turn or not by showing their name taken from input and **nextbut.pack()** when clicked will direct them to the playboard of the next player.

```
# runs when next button is clicked
def nextTurn():
    refreshInformation()
    refreshBoard()
```

- **The nextTurn() function**

This function calls **refreshInformation() and refreshBoard()** and run when the next button is clicked and prepare the board and information of the board ready for the next player.

```
def colorPicker():
    colorinstructions.pack(side=TOP)
    for x in unocolours:
        x.pack(side=TOP)
```

- **The colorPicker() function**

This function basically adds the label of explaining of what to do in the screen and the choices of colours that the players need to pick and place it in the top .

```
# clears the screen at the end of a player's turn
def endClear():
    global pagenum
    pagenum = 0 #
    drawbut.pack_forget()
    endbut.pack_forget()
    topcard.pack_forget()
    discardlabel.pack_forget()
    leftbut.pack_forget()
    rightbut.pack_forget()
    wildlabel.pack_forget()
    for x in playerhands[counterturn].cards:
        x.but.pack_forget()
```

- **The endClear() function**

This function clears the play board at the end of the player's turn as mentioned before. , I have used **global function** since **pagenum** variable is defined outside the function **pagenum = 0** means that it resets the page number back to 0 for the next player (most left page of their hand).

```
 # refreshes the board
def refreshBoard():
    refreshCards() # shows their first page of cards
    refreshDiscard() # updates top of discard pile
    drawbut.pack() # adds the draw button
```

- **The def refreshBoard () function**

This function is use for refresh the board of the player turn after ending a turn, refreshing the players card whether it's less or more than previous because of drawing a card from the draw pile or getting skipped, or even getting a draw card of +2 or +4 this calls **refreshCards(). It also called out refreshDiscard() function** to update the discard pile**, and drawbut.pack() that** display the draw button.

```
# Refreshes the information at the top of the in-
game screen when the next player goes
def refreshInformation():
    for i in range(len(playernames)):
        infolist[i].pack_forget() #clear the information of the current pl
ayer
        # Changes the text of each of the information labels
        if i == counterturn: # if it is the player's turn
            infolist[i].config(text = playernames[i] + ". It's your turn!"
)
        else: #  if it isn't the player's turn then show how many cards le
ft in hand
            infolist[i].config(text = playernames[i] + ". Cards in hand:"
+ str(len(playerhands[i].cards)))
    infolist[4].pack_forget() # remove the draw pile label
    infolist[4].config(text = "Cards left in draw pile: " + str(len(drawpi
le.cards))) # refresh the draw pile display
```

```
    for i in range(len(playernames)): # add the number of cards in the the
 other playerhand
        infolist[i].pack(fill = Y, side=LEFT) # add each player's informat
ion
    infolist[4].pack(fill = Y) # add draw pile display back
```

- **The def refreshInformation() function**

  This function is use to show the information on the upper left of the main game screen or area of play, the player's names that are inputted are shown. If it's the player's turn, it will display "[player's name], it's your turn!" as well as shown how many cards left in the opponent's hand(s). I used **if else** function, to show information when the player takes turns. For instance, if it's player A turn it will show "A, it's your turn!", if not only show numbers of cards they have left.

```
# refreshes the player's cards in hand
def refreshCards():
    for x in playerhands[counterturn].cards:  #remove every card in
the player's hand from the screen
        x.but.pack_forget()
    leftbut.pack_forget() # remove the left arrow
    rightbut.pack_forget() # remove the right arrow
    if pagenum > 0: #if not in the first page
        leftbut.pack(side=LEFT) # add left arrow so the player can s
croll left of page
    for x in playerhands[counterturn].cards[pagenum:pagenum+5]:
        x.but.pack(side=LEFT) ## Show 5 cards (or fewer) in the play
er's hand depending on which page you are on
    if pagenum < len(playerhands[counterturn].cards)-5:
        rightbut.pack(side=LEFT)
```

- **The def refreshCards() function**

This function is to refresh the cards in player hands whenever they want to shift their page of cards display to left or right depending on which page they are in since the maximum cards that can be in one page are 5.

For example : Player A has 8 cards in hands, the discard pile colour is blue, he is currently in page one and couldn't find blue, so he decided to see if he has blue, in the rest three cards that he couldn't see in the first page, so he shifts to the right until he finally found blue card with three clicks of right arrow which is his last page of cards.

How the code works:

- It will remove every card in the player hands as well as the right and left arrow button.
- If the page is not 0 (first page) then it will add left arrow, therefore the player can shift their page to the left calling **leftbut.pack()** but if the cards in hands are 5 or less there will be no arrows available.

- If the player is not in last page of cards, show the right arrow so that the players can shift to the right page by calling **righbut.pack()**

```
def refreshDiscard(): # update the discard
    discardlabel.pack(side = TOP) # add a label to show that this is
 the top card of the discard pile
    if discard.cards[0].cardnum == 13 or discard.cards[0].cardnum ==
 14: # if the front of the card is wildcard or +4 wildcard
        wildlabel.config(text = "This wild card is \n" + colorlist[d
iscard.cards[0].color], fg = colorlist[discard.cards[0].color])
        wildlabel.pack() #add colour text to indicate the wildcard c
hosen
    topcard.config(image = discard.cards[0].img) # update the image
of the discard pile
    topcard.pack(side=RIGHT) # put the topcard right in
```

- **The def refreshDiscard() function**

This function is particularly to update the top card of the discard pile by adding label to show that by calling **.pack** which is declared in #creating label and buttons. If the card is regular number cards and special cards from 0-13, it will just show the top card.

On the other hand, if the card is wildcard or +4 it will show the chosen wildcard color by taking it from **colorlist** list and the font will follow the colour as well and update the topcard of the discard file and placed in on the right using **.config(image =) and .pack(side = RIGHT) functionality is to place the desired image on the right side .**

```

def refreshDraw():
    discardlength = len(discard.cards) #if the drawpile ran out, eve
ry card will reshuffle from the discard pile but not the top card
    for i in range(discardlength-1):
        drawpile.add_front_deck(discard.arrange(-
1)) # move every card except the top front card of deck, starting wi
th the bottom card, of the discard pile to the draw pile
    random.shuffle(drawpile.cards) # shuffle the new draw pile
```

- **The def Refresh Draw() function**

  This function is for reshuffling the draw pile from the discard pile, if a scenario happens where players are still playing and the draw pile ran out but not the top card.

  Functions that are used here is **len(), for in range()**

```
# remove everything on the main in-game screen
def win():
    information.pack_forget()
    leftbox.pack_forget()
    area_toplay.pack_forget()
    rightbox.pack_forget()
    butbox.pack_forget()
    discardbox.pack_forget()
    winbox.pack()
    winbox.pack_propagate(0)
    winlabel.config(text = playernames[counterturn] + " wins!") #show the
winner
    winlabel.pack(fill=BOTH)
```

- **The def win() function**

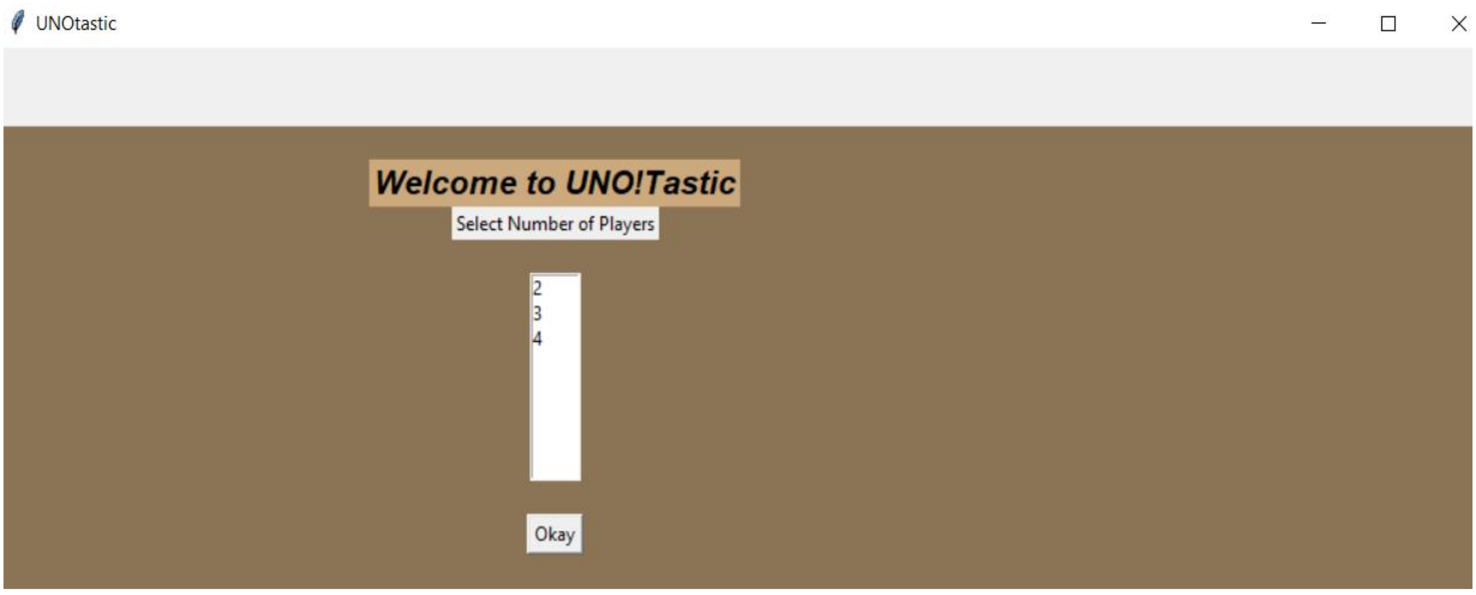
function is for showing the players who's the winner of the game and eventually end of the game, **.pack_forget()** is to remove and clear the layout and display of the in game set up and information display.

```

# Setting up the main frames in the game
#propagate - stop the window from sinking
information = Frame(window, width = 1000, height = 50) # show information  at
the top
information.pack_propagate(0)
information.pack()
leftbox = Frame(window, width = 64, height = 300, bg = "burlywood4") # box for
 left arrow
leftbox.pack_propagate(0)
leftbox.pack(side=LEFT)
area_toplay = Frame(window, width = 620, height = 300, bg = "burlywood4") # cr
eate area of the game to play with the cards
area_toplay.pack_propagate(0)
area_toplay.pack(side=LEFT)
rightbox = Frame(window, width = 64, height = 300, bg = "burlywood4") # box fo
r right arrow
rightbox.pack_propagate(0)
rightbox.pack(side=LEFT)
butbox = Frame(window, width = 128, height = 300, bg =  "burlywood4") #box for
 button
butbox.pack_propagate(0)
butbox.pack(side=LEFT)
discardbox = Frame(window, width = 124, height = 300, bg =  "burlywood4") # bo
x for top of discard pile
discardbox.pack_propagate(0)
discardbox.pack(side=LEFT)
winbox = Frame(window, width = 1000, height = 350) # box when a player wins. G
ets packed at the end
```

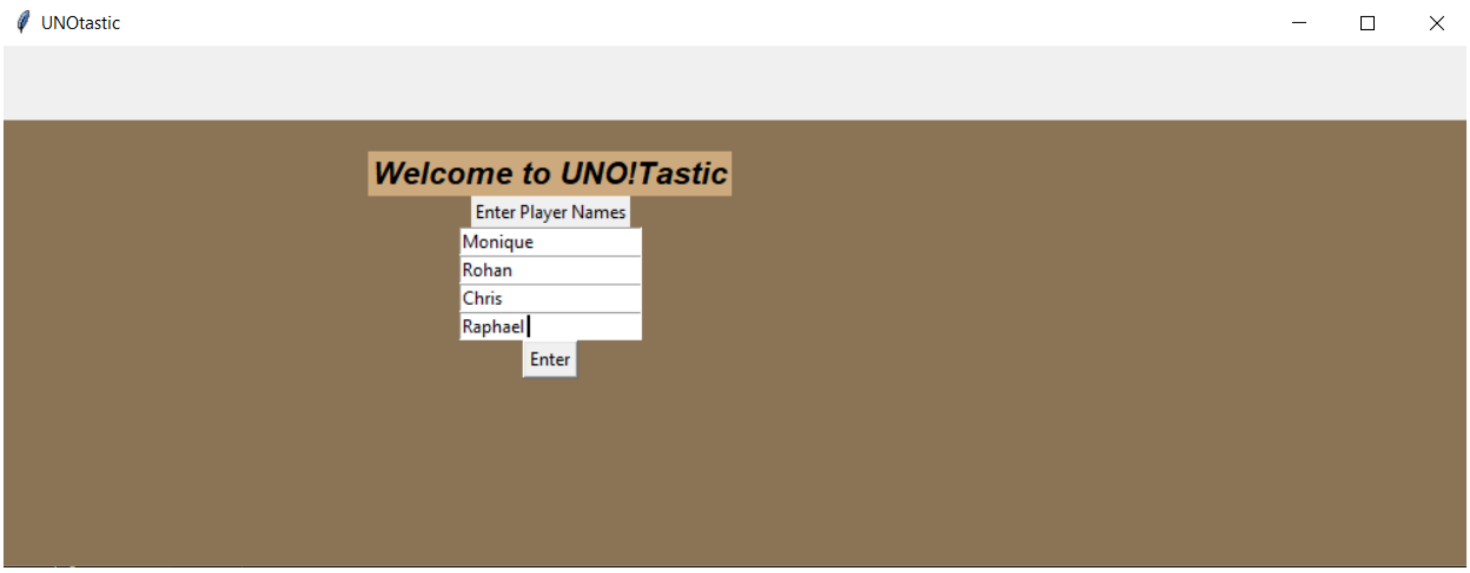
```
# Creating labels & buttons
topcard = Label(discardbox) # displays image of top of discard pile
discardlabel = Label(discardbox, text = "Game Discard Pile ", bg = "black", fg
 = "burlywood3", width = 18, height = 2) # Shows that this card is the discard
 pile
winlabel = Label(winbox, font=("Courier", 60))
drawbut = Button(butbox, text = "Draw Card", command = drawCard)
endbut = Button(butbox, text = "End Turn", command = endButton)
nextbut = Button(area_toplay, text = "Next Player, please click this button to
 begin your turn.", command = nextButton)
leftimage = PhotoImage(file="leftarrow1.png")
leftbut = Button(leftbox, image = leftimage, command = leftArrow)
rightimage = PhotoImage(file="rightarrow1.png")
rightbut = Button(rightbox, image = rightimage, command = rightArrow)

# Creating variables & objects
playernames = [] #
playerhands = [] # this list will be used to call on each player hands
drawpile = Deck(1) # create a full deck
discard = Deck(0) # create an initial empty discard pile
direction = 1 # direction flow of the game but can be reversed with a special
card : reverse card
counterturn = 0 # corresponds to the index of the player whose turn it is
pagenum = 0 #determines which page of hards in the player's hand gets displaye
d
infolist = [Label(information, bg = "antique white"), Label(information, bg =
"lightblue"), Label(information, bg = "thistle"),Label(information, bg = "Ligh
tPink2"),Label(information)]
unocolours = [Button(butbox, text = "Red", fg = "red", command = redClick), Bu
tton(butbox,  text = "Yellow", fg = "orange",  command = yellowClick),Button(b
utbox, text = "Green", fg = "green", command = greenClick),Button(butbox, text
 = "Blue", fg = "blue", command = blueClick)]
colorinstructions = Label(area_toplay, text = "Choose a color you would like y
our wild card to be.", bg = "white", fg = "black")
wildlabel = Label(discardbox, bg = "black")  # displays the color of wild card
colorlist = ["red","yellow","green","blue"]  # used to convert color from numb
ers to strings
playerselection = Listbox(area_toplay, selectmode = BROWSE, width =5, height=8
) #this is the listbox for selecting the number of players
confirm_names_button = Button(area_toplay, text = "Enter", command = EnterName
) #this is the button to confirm the names inputted
name_label = Label(area_toplay, text = 'Enter Player Names')
number_label = Label(area_toplay, text = 'Select Number of Players')
numberokaybutton = Button(area_toplay, text = "Okay", command = Confirm) #this
 is the button to confirm number of players selection
```

```
spacer_label1 = Label(area_toplay, bg = "black")
spacer_label2 = Label(area_toplay, bg = "black")
spacer_label3 = Label(area_toplay, bg = "black")

Setup() #begin the setup !!!!!!!

window.title("UNO!tastic ")
window.mainloop()
```

Screenshot of Working Program

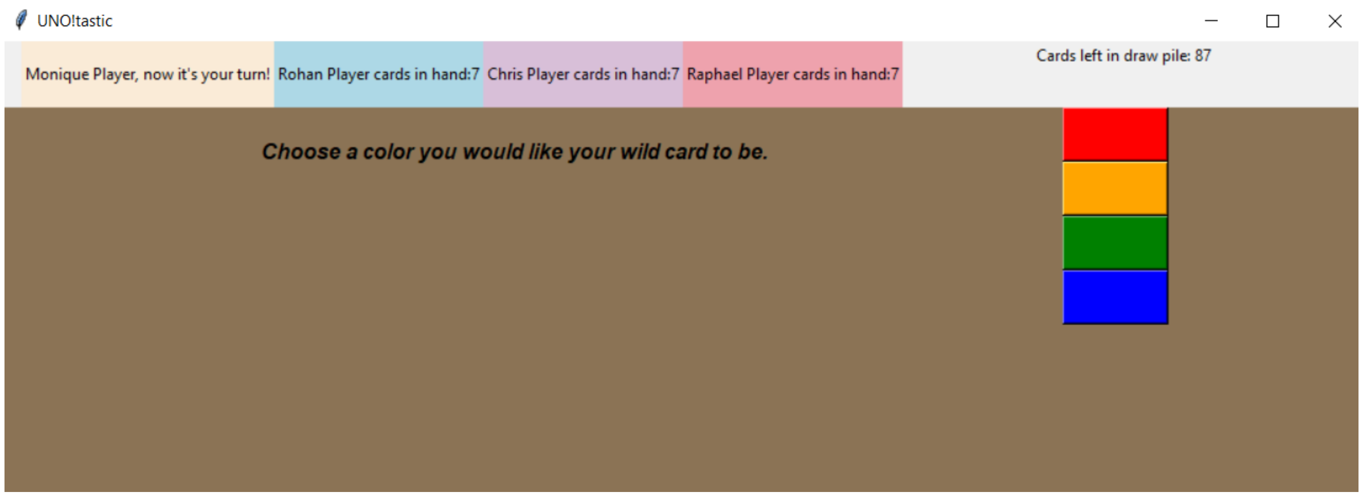


First Window : The program asked how many people are playing the game



Monique
Rohan
Chris
Raphael

Players are able to input their names and shown in the game screen



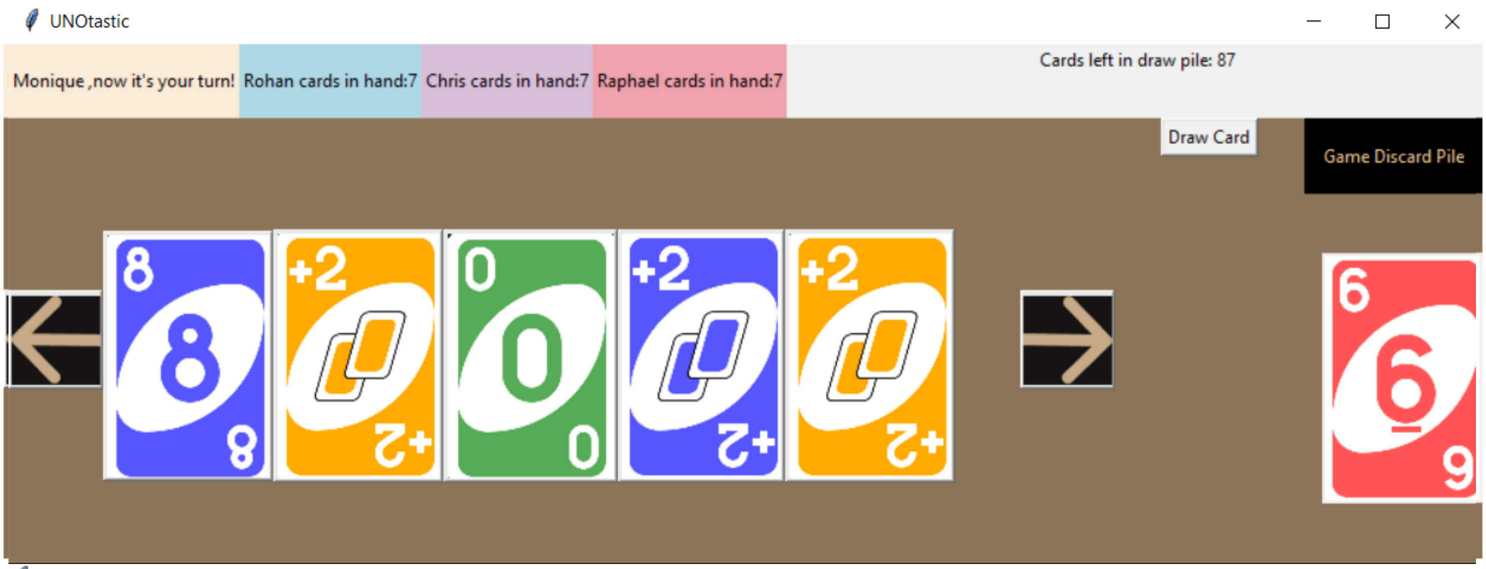
UNO!tastic

Monique Player, now it's your turn! | Rohan Player cards in hand:7 | Chris Player cards in hand:7 | Raphael Player cards in hand:7

Cards left in draw pile: 87

Choose a color you would like your wild card to be.

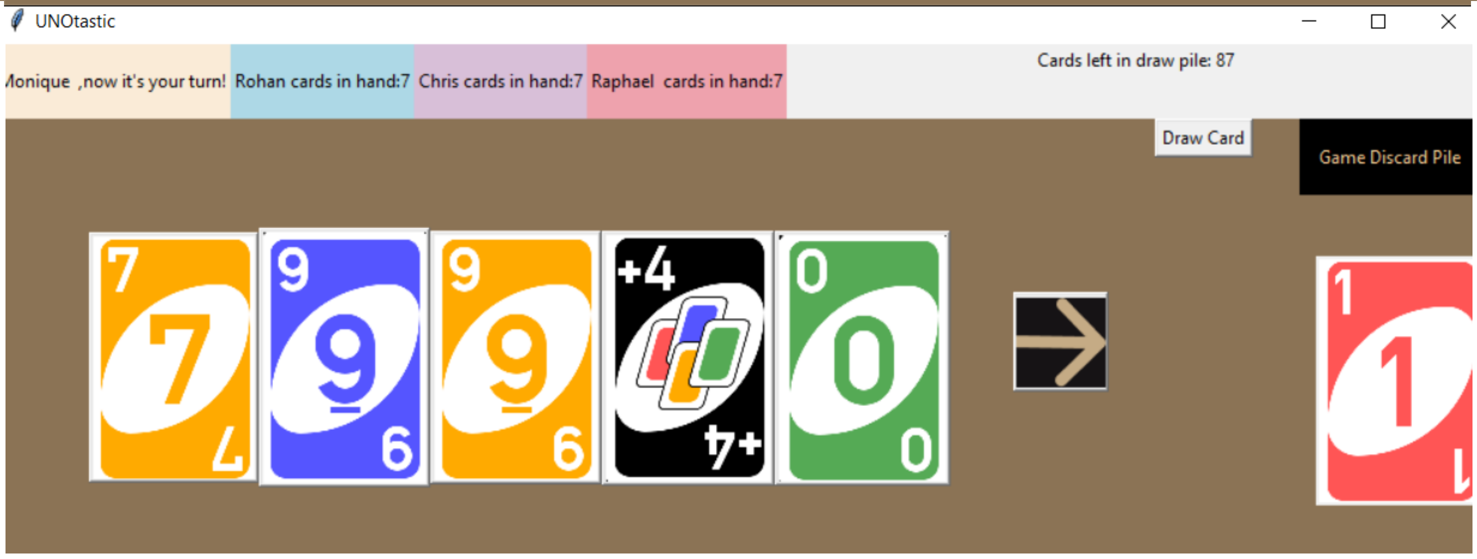
When a player uses their wildcard / +4 wildcard they can choose what colour they desire.



UNOtastic

Monique ,now it's your turn! | Rohan cards in hand:7 | Chris cards in hand:7 | Raphael cards in hand:7

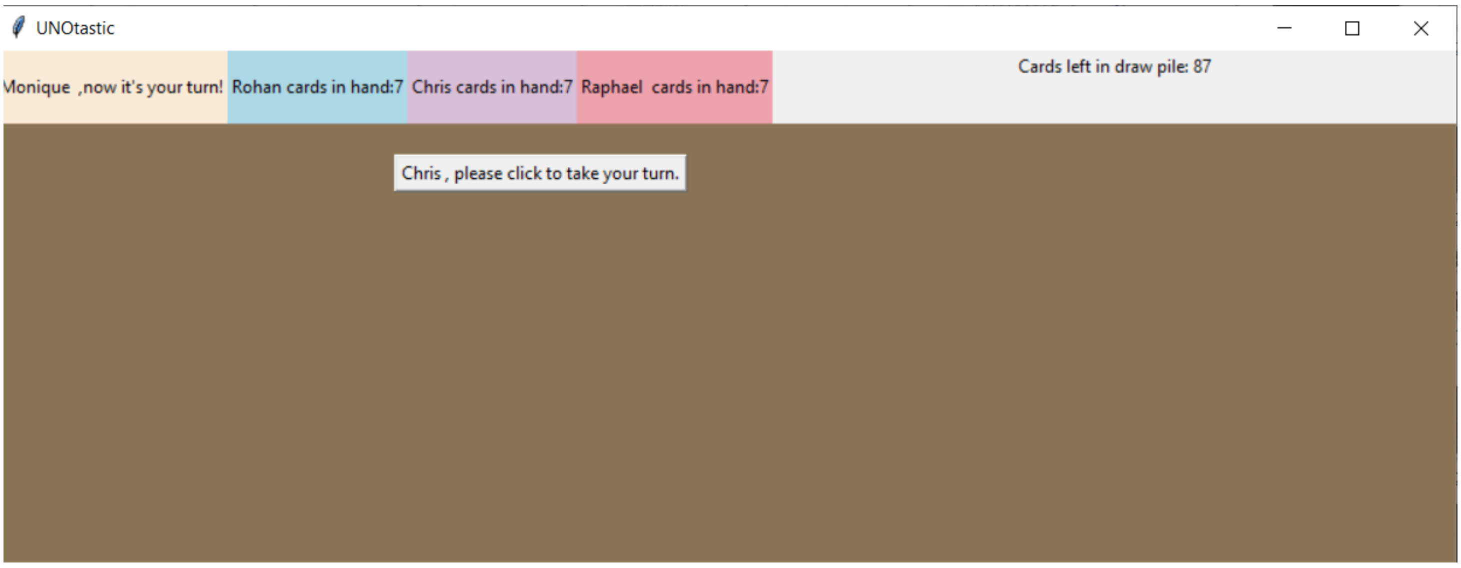
Cards left in draw pile: 87



UNOtastic

Monique ,now it's your turn! | Rohan cards in hand:7 | Chris cards in hand:7 | Raphael cards in hand:7

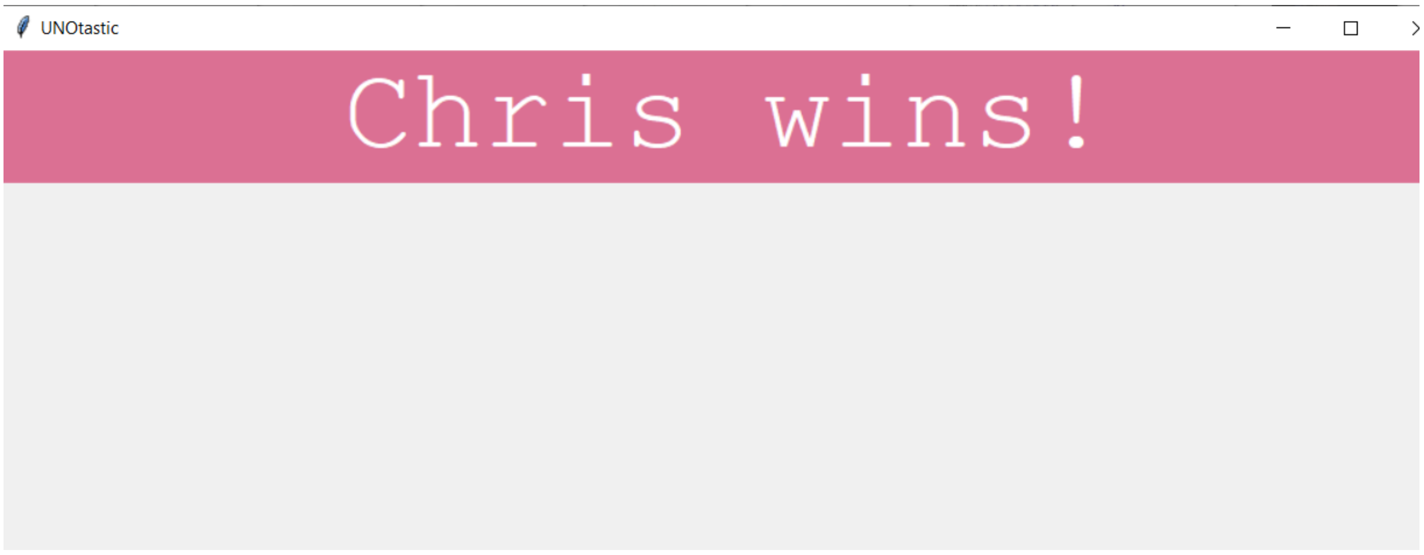
Cards left in draw pile: 87


UNOtastic

Monique ,now it's your turn! | Rohan cards in hand:7 | Chris cards in hand:7 | Raphael cards in hand:7

Cards left in draw pile: 87

Transition page to know it's the next player's turn


UNOtastic

Chris wins!

Winner Declaration

Source of Code :

https://github.com/ardeliaraminta/UNO-tastic

References and Sources :

http://www.science.smith.edu/dftwiki/index.php/Color_Charts_for_TKinter

https://python-textbok.readthedocs.io/en/1.0/Introduction_to_GUI_Programming.html

https://commons.wikimedia.org/wiki/File:UNO_cards_deck.svg

https://www.geeksforgeeks.org/introduction-to-tkinter/?ref=lbp

https://stackoverflow.com/questions/14247709/how-do-i-change-button-size-in-python