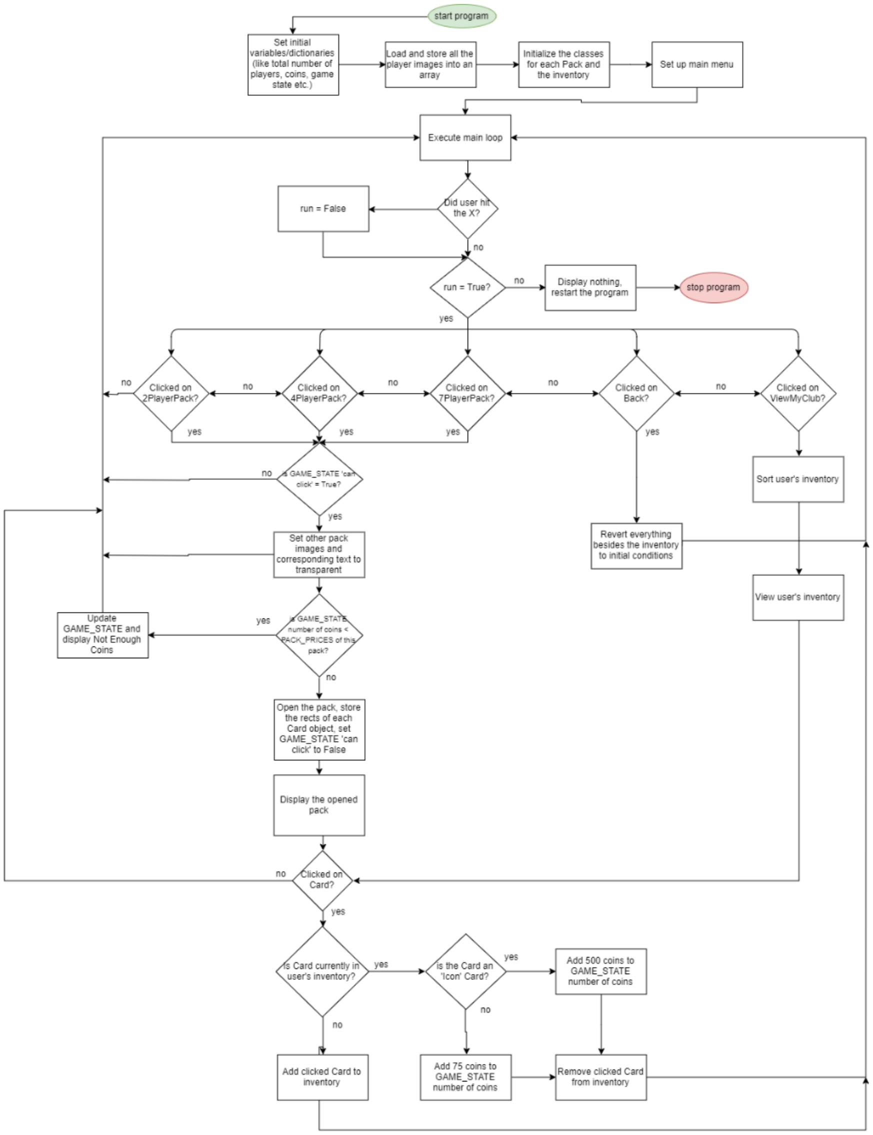
Scott Hunt
CS2021 - Python Programming
Final Project – FIFA Pack Opening Simulator

Introduction:

For the final project of the Python Programming course, I decided to go down the gaming path. As a fan of sports games and more specifically, the FIFA series, I already had something working up in my mind about what I wanted to do. Here's some background information; in FIFA, there is a mode called "Ultimate Team" where the user works from the ground up to manage their own soccer team both offline and online. This mode uses two forms of in-game currency, coins and FIFA points. You earn coins from playing games, and doing various challenges whereas you have to spend real money on FIFA points. The goal is to use these currencies to buy the best players you can in order to form your ultimate team. You can do so by using the in-game player transfer market. This is where other real-life users put their own players up for auction where you, or others, can buy them using coins. The other way to gain players is to open "packs". Packs in FIFA are exactly like a pack of Pokémon cards in real-life. A user spends a set amount of coins or FIFA points to open a pack, and depending on the pack opened, the user can receive different players and/or items to use. The problem arises, for me and a community of other people, when the risk/reward value is just plain too low. It is not worth a user's time to open packs for a very small chance of getting something even half decent. The developers make the prices for packs way too high for users to be receiving some of the worst players/items in the game.

This is where my project comes in. I used a FUT (FIFA Ultimate Team) resource/database website to take the top 300-ish rare gold cards and 60-ish icons (the best of the best players throughout history) to make a Pack Opening Simulator so I don't need to spend any time or money on actual packs but still get the satisfaction of receiving a good player. To accomplish this project, I used Pygame. Pygame helped with loading images, displaying images, event checking, and window displaying, which all played a huge role in completing this project. I used 3 classes to help with objects: Card, Pack, and

MyClubPlayers. The Card class contains variables for a player image, the path of that player image, and the Pygame rect of the image/object. The Pack class contains variables for the name of the pack, the price, the size (# of cards it'll contain), a list of Card objects, and a list of rects for those Cards. The MyClubPlayers class contains a list variable to store Card objects and a corresponding list to store those Card image paths. I used a separate class for that (MyClubPlayers) so I can sort the Card objects and display them differently when viewing the user's club. I tried commenting my code to the best of my ability so there is maximum understanding. Here is a diagram of how the whole thing works:



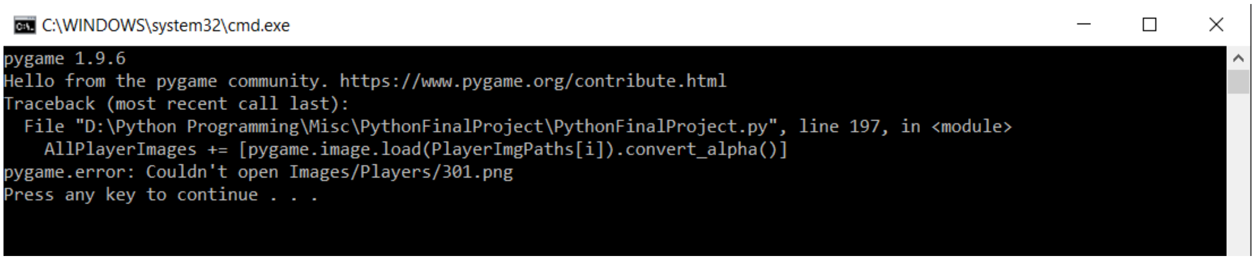
start program
Set initial variables/dictionaries (like total number of players, coins, game state etc.)
Load and store all the player images into an array
Initialize the classes for each Pack and the inventory
Set up main menu
Execute main loop
Did user hit the X?
run = False
no
run = True?
no
Display nothing, restart the program
stop program
yes
Clicked on 2PlayerPack?
Clicked on 4PlayerPack?
Clicked on 7PlayerPack?
Clicked on Back?
Clicked on ViewMyClub?
no
no
no
no
no
yes
yes
yes
yes
is GAME_STATE 'can click' = True?
no
yes
Set other pack images and corresponding text to transparent
Sort user's inventory
Revert everything besides the inventory to initial conditions
View user's inventory
is GAME_STATE number of coins < PACK_PRICES of this pack?
yes
Update GAME_STATE and display Not Enough Coins
no
Open the pack, store the rects of each Card object, set GAME_STATE 'can click' to False
Display the opened pack
Clicked on Card?
no
yes
Is Card currently in user's inventory?
yes
is the Card an 'Icon' Card?
yes
Add 500 coins to GAME_STATE number of coins
no
no
Add clicked Card to inventory
Add 75 coins to GAME_STATE number of coins
Remove clicked Card from inventory

Project Results:

As far as accomplishments, I think I accomplished a good alternative to the heart wrenching tragedy that is opening real FIFA packs. Of course, this project did not go perfectly, there were some more difficult times in comparison to others. Below, is a list of specific problems and successes I had.

Problems:

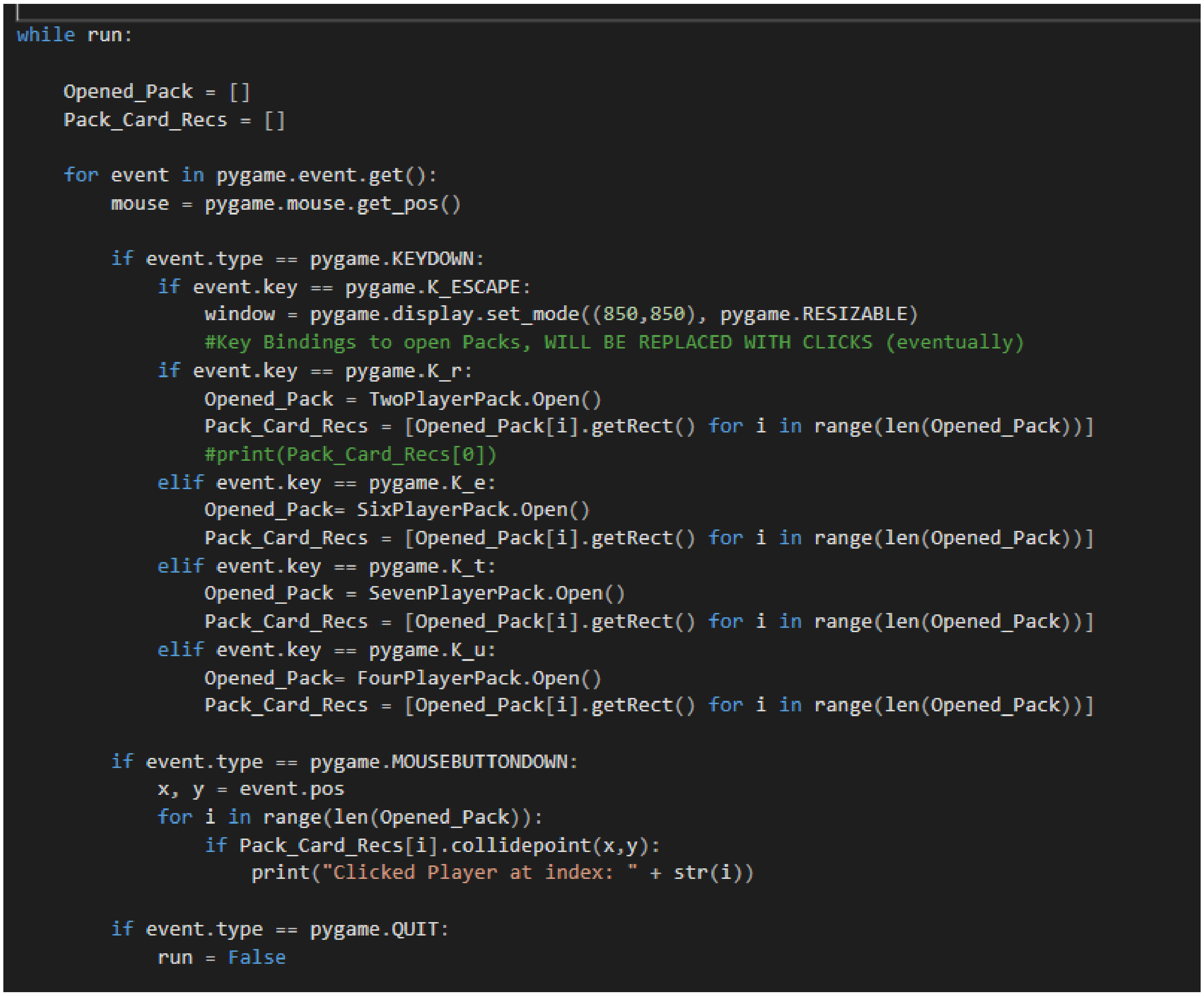
-About half way from the finished version, I switched from IDLE to Visual Studio. Everything was going fine until I tried to add more images from the database to the project. It gave a "couldn't open image" error for seemingly no reason:



```
pygame 1.9.6
Hello from the pygame community. https://www.pygame.org/contribute.html
Traceback (most recent call last):
  File "D:\Python Programming\Misc\PythonFinalProject\PythonFinalProject.py", line 197, in <module>
    AllPlayerImages += [pygame.image.load(PlayerImgPaths[i]).convert_alpha()]
pygame.error: Couldn't open Images/Players/301.png
Press any key to continue . . .
```

I checked to make sure the image existed and it did. No matter what I changed in the code, it didn't want to work. I tried renaming all the images, renaming the image that threw the error, I tried doing so much that ended up not working. After a day or two, I copy/pasted to code back into IDLE and sure enough it worked fine. So, I ran it a couple times in IDLE, made some small changes and pasted it back into Visual Studio and it worked again, and has worked ever since.

-Another huge setback I had was checking for clicks on a list of Card objects. Unfortunately, this was the hardest problem I faced during this project for many reasons.

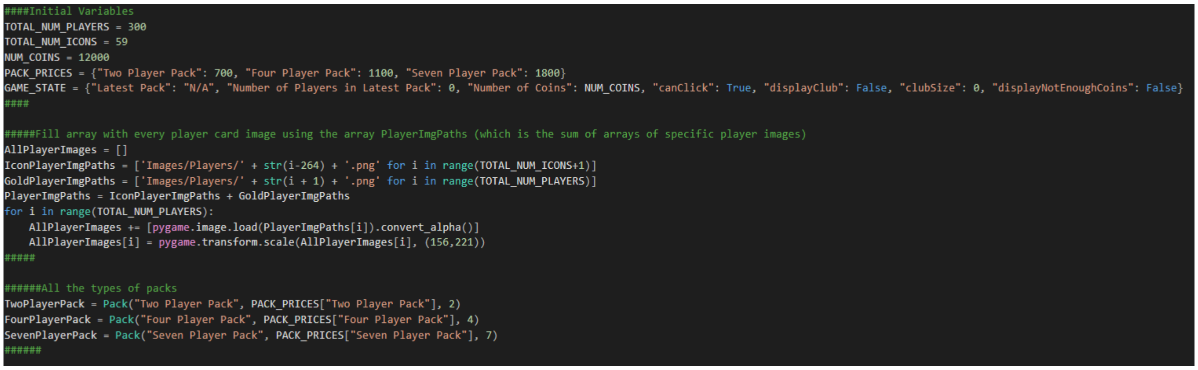


```
while run:

    Opened_Pack = []
    Pack_Card_Recs = []

    for event in pygame.event.get():
        mouse = pygame.mouse.get_pos()

        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_ESCAPE:
                window = pygame.display.set_mode((850,850), pygame.RESIZABLE)
                #Key Bindings to open Packs, WILL BE REPLACED WITH CLICKS (eventually)
            if event.key == pygame.K_r:
                Opened_Pack = TwoPlayerPack.Open()
                Pack_Card_Recs = [Opened_Pack[i].getRect() for i in range(len(Opened_Pack))]
                #print(Pack_Card_Recs[0])
            elif event.key == pygame.K_e:
                Opened_Pack= SixPlayerPack.Open()
                Pack_Card_Recs = [Opened_Pack[i].getRect() for i in range(len(Opened_Pack))]
            elif event.key == pygame.K_t:
                Opened_Pack = SevenPlayerPack.Open()
                Pack_Card_Recs = [Opened_Pack[i].getRect() for i in range(len(Opened_Pack))]
            elif event.key == pygame.K_u:
                Opened_Pack= FourPlayerPack.Open()
                Pack_Card_Recs = [Opened_Pack[i].getRect() for i in range(len(Opened_Pack))]

        if event.type == pygame.MOUSEBUTTONDOWN:
            x, y = event.pos
            for i in range(len(Opened_Pack)):
                if Pack_Card_Recs[i].collidepoint(x,y):
                    print("Clicked Player at index: " + str(i))

        if event.type == pygame.QUIT:
            run = False
```

This is the only screenshot I have of the code in question when it was causing problems. If i iterated through the Opened_Pack list using a for loop and try to click on it in-game, then nothing happens. When I tried it using an index ([0], [1], etc) then there was an Index Out of Range Error, no matter what number I used or how big the list is. I fixed this particular screenshot of code simply my moving the two lists outside to main loop, as it was resetting them with each loop. It was an extremely stupid mistake to be making. But I still had the problem of nothing happening when I click on a Card. This was when I had to change the Card class to have the rect of the image to be part of the class so I could use it here. Because of that change, I had to change how the Open method of the Pack class worked and so on until it works how it does now.

<u>Successes</u>

I've had some successes while working on the project as well. A notable one is this snippet of code:



```
####Initial Variables
TOTAL_NUM_PLAYERS = 300
TOTAL_NUM_ICONS = 59
NUM_COINS = 12000
PACK_PRICES = {"Two Player Pack": 700, "Four Player Pack": 1100, "Seven Player Pack": 1800}
GAME_STATE = {"Latest Pack": "N/A", "Number of Players in Latest Pack": 0, "Number of Coins": NUM_COINS, "canClick": True, "displayClub": False, "clubSize": 0, "displayNotEnoughCoins": False}
####

#####Fill array with every player card image using the array PlayerImgPaths (which is the sum of arrays of specific player images)
AllPlayerImages = []
IconPlayerImgPaths = ['Images/Players/' + str(i-264) + '.png' for i in range(TOTAL_NUM_ICONS+1)]
GoldPlayerImgPaths = ['Images/Players/' + str(i + 1) + '.png' for i in range(TOTAL_NUM_PLAYERS)]
PlayerImgPaths = IconPlayerImgPaths + GoldPlayerImgPaths
for i in range(TOTAL_NUM_PLAYERS):
    AllPlayerImages += [pygame.image.load(PlayerImgPaths[i]).convert_alpha()]
    AllPlayerImages[i] = pygame.transform.scale(AllPlayerImages[i], (156,221))
#####

######All the types of packs
TwoPlayerPack = Pack("Two Player Pack", PACK_PRICES["Two Player Pack"], 2)
FourPlayerPack = Pack("Four Player Pack", PACK_PRICES["Four Player Pack"], 4)
SevenPlayerPack = Pack("Seven Player Pack", PACK_PRICES["Seven Player Pack"], 7)
######
```

The Initial Variables section is something I had since the beginning. I've only had to change what was needed with each major update the code had. The most successful part of this though was how I load all the images to the AllPlayerImages list. This was one of the first things I attempted and it evolved as the project went on until I did it the most efficient way I see possible, even now. Those 7 lines of code can load in as many images as you could imagine (theoretically) and it only takes 7 lines. Not only that, but it also scales each and every one of them to the correct size. This is something I'm personally proud of.

Division of Work:

This project was done entirely by me and I started from scratch. I've made this entire project from the ground up. I used some websites to help when I struggled but I contributed every line of code in the project. For documentation, I commented every section of code so it should be easier to follow and/or understand.

Bibliography:

GeeksForGeeks, Remove Duplicates [function] https://www.geeksforgeeks.org/python-remove-duplicates-list/