Name: Veer Khosla
Course: CS151
Section: A
Date: 2/22/2022

Project 3 Report:

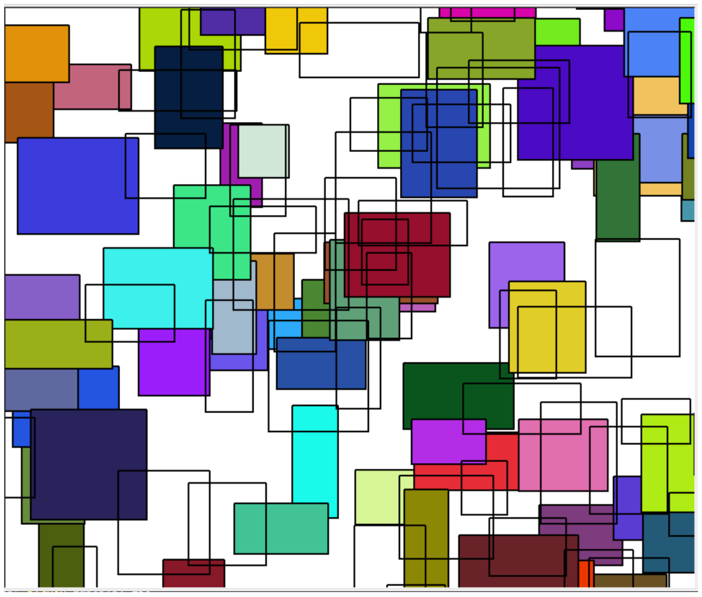
Title: Drawing Scenes within Scenes

Abstract:

- There were many lecture concepts that were relevant to this project. Functions, obviously, were huge. They allowed me to reduce the amount of code I had to write by just repeating some functions. Additionally, parameters were crucial, because there were so many times that I would call a function, but would want some slight alterations. Parameters and arguments made this possible and efficient. Specifically, scaling parameters were really helpful as I would not have to rewrite code for different sizes of the same image. Instead, I could have a variable assigned to a multiplier that would scale a specific function up or down, making it easier to control size. Command Line Arguments were also helpful, especially for the last part of the lab and project. These allow us to run the program directly from the terminal, with specific arguments as well. For example, in the lab, we used the command line to control the number of rectangles in our Mondrian picture. In the project, I used a command line to change the scene outside between day and night. Finally, conditionals and loops were huge in this project. I used them so much, in many functions. One function I made even has 6 for loops. While there is probably a more efficient way to write that for loop, maybe even with a while loop, I was unsure how to do that, but I am sure there is a way that we will discuss soon.
- For this project, I submitted a total of 4 images. The first image is my Mondrian from the lab. For the actual project, I turned the rectangles in this picture into triangles and restricted the colors to a grey palette. My second image is 3 pictures in one scene. The picture used is my scene from project 2, which I scaled and repositioned in 3 different places on the screen as shown below. This was a bit difficult, which is also described in the image description, but I managed to do it. My third and fourth images are scenes inside a home, looking out a window with pictures on the walls. The user has the ability to control whether it is day or night with command line arguments.

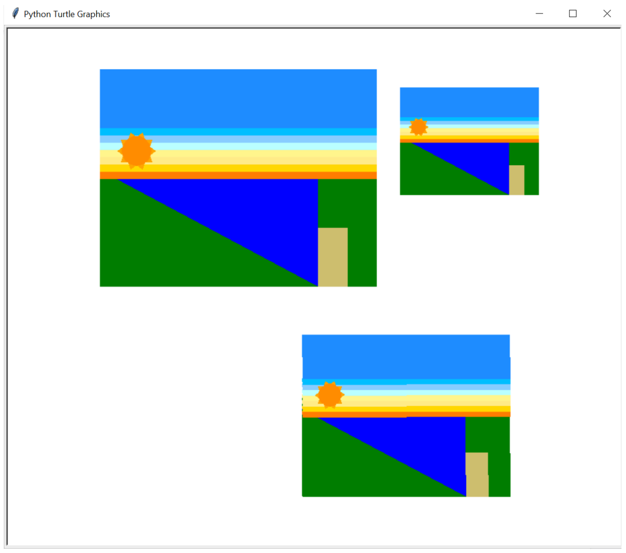
Results:

IMAGE 1



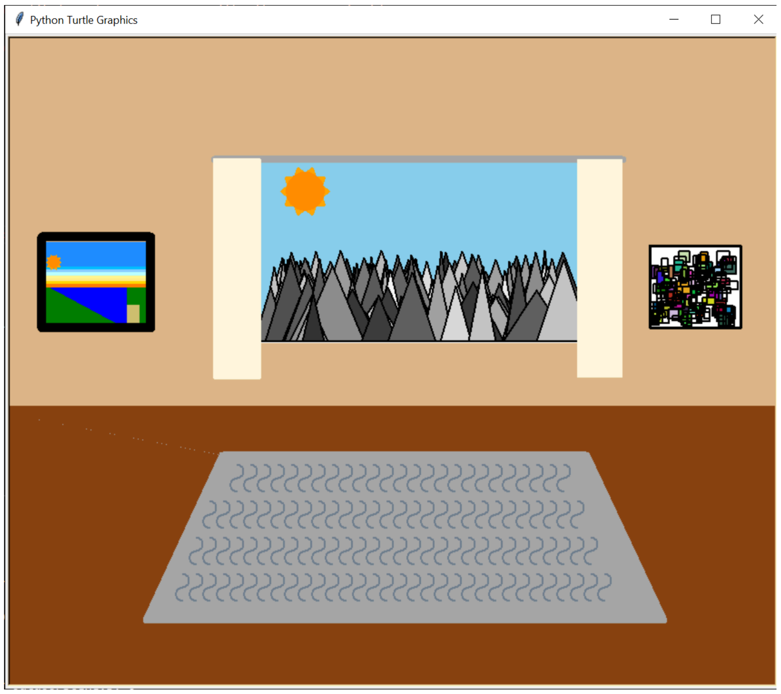
I made this image in the lab and the afternoon following. I had a function that drew a rectangle with coordinates, size, and color and pen parameters. If the fill was True, the rectangles would be any color. If false, it would be empty, as seen. Another function, main, has a for loop which repeats as many times as input in the terminal with the command line prompt. When i is less than 80, the rectangles are given any color. Anything else, there is no fill on the rectangles. With the command line prompt, I can input however many rectangles I want. I chose 150. This is displayed above in Image 1.

IMAGE 2



This function took me hours to make. Once adding a scale to all the size parameters in my function from Project 2, it seemed to be working; however, the exception to this was the sky. For some reason (I think because it was totally separate from the other functions within this scene function), the sky was constantly offset from the rest of the image. However, I believe I got it working correctly by messing around with the sky function. I also nested sky this function within the whole scene function so it would not be independent. Eventually, I got it working, and scaling the scene would, in fact, scale the entire image as seen above. Then, all I had to do was choose different points to place my image to create this. It is under the function pattern_outdoorScene.

IMAGE 3



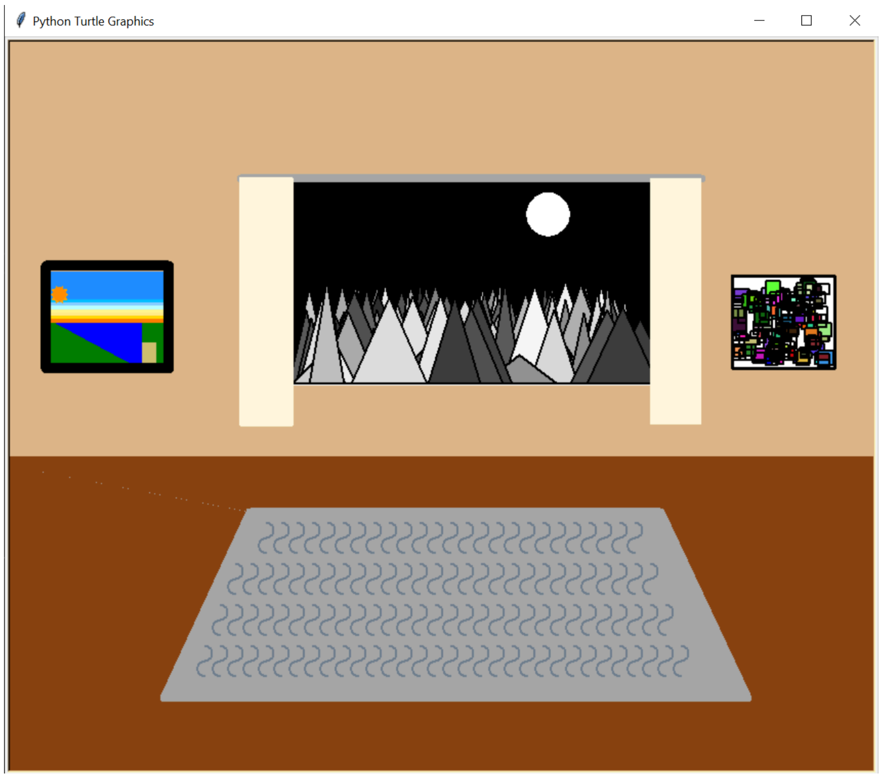
I decided to make my scene within a scene image inside a home. My previous scene from project 2, as well as my rectangular Mondrian from Lab 3, are both paintings on the wall, and my triangular Mondrian pattern through the window is a mountain range. This image is daytime. To get this, in the command line, type: ‘python OutdoorScene.py 0’. 0 here corresponds to daytime (This is for Windows controls. On OSX, I believe you just have to type python3 instead of python).

I was able to do this by making a ‘time’ variable that was equal to the integer of the system.argv input. I made day == 0, and night == 1. Then, all I had to do was make an if statement:

```
    If time == day, execute code for blue fill and sun
    Elif time == night, execute code for black fill and moon
```

Additionally, I added some other elements to make it look more realistic, such as the carpet, wood flooring, wall, and curtains in front of the window. I made a function called frame, which makes uses parameters to choose a position and size of the rectangle. I also gave a parameter for color, so I would be able to reuse the frame function to make my window. Finally, The carpet is a trapezoid to make the room look and feel more 3-dimensional. I also put a cool little pattern on it by making semicircles in opposite directions.

IMAGE 4



This image is the same as Image 3, just the only difference is that outside the window, it is nighttime. For this image, type: 'python OutdoorScene.py 1'. 1, in this case, 1 corresponds to nighttime. Additionally, typing a value that is not 0 or 1 will result in an error statement telling the user to input either 0 or 1 in the command line argument. It will also close the turtle window.

Reflection:

1. A conditional statement is a statement that allows us to take different actions and paths, depending on whether a condition is met or not. Conditional statements allow us to control the flow of functions by executing differently depending on the condition. For example, we could have r.randint(0, 10) generate a random number between 0 and 10. We could use conditionals to say, if the number generated is less than or equal to 5, say hello. If it's above 5, say goodbye. This could be done as:

```
        num = random.randint(0,10)
```

```
If num <= 5:
        print('Hello!')
Else:
        print('Goodbye!')
```

As seen, depending on the condition, we can control what will follow. In this case, it is very simple with just a print statement. However, we can make this much more complex, and completely change the direction of the entire program following the conditional statement.

2. The parts of a for loop are:
    - Index: variable to which we can assign any name
    - Sequence: how many times the loop will repeat; often range
    - Body: what is actually being repeated by the loop.

   The range is a list. For example, range(10) is really the list [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]. In total, this list contains 10 integers. Putting this in a for loop will tell the for loop to run 10 times, going to the next integer in the list with each iteration. Once it reaches the last value in the list, the program recognizes this as the end and ends the for loop. The program will then proceed to the following code.

Sources, imported libraries, and collaborators are cited, **or a note is included indicating that none were referenced:** This ensures you are properly crediting the people and sources who help you achieve your results. Not listing them in the report is considered plagiarism or stealing. Please code with honor.

Math Module Documentation: https://docs.python.org/3/library/math.html
Turtle Module Documentation: https://docs.python.org/3/library/turtle.html
Random Module Documentation: https://docs.python.org/3/library/random.html
System Module Documentation: https://docs.python.org/3/library/sys.html

Greyscale Pallete:
https://dev.to/finnhvman/grayscale-color-palette-with-equal-contrast-ratios-2pgl
Grey Color Pallete: https://calcolor.co/palette/942409461
Turtle Color List: https://cs111.wellesley.edu/labs/lab02/colors

TA Hours - Maya Purohit (and her friend who was stopping by - did not their catch name)
CS151A Google Classroom - Lab 3 PowerPoint and Lecture Slides and turtle_wave.py