Jack Spiratos, 1930720

Robert Vincent Programming Techniques and Paradigms

420-LCW-MS section 1

May 7, 2021

Final Project

**Mini-Manual**

This program is an imitation of the classic Tetris game. To start the game, simply open the python file titled “Tetris” inside the folder provided to the user. Potential problems may occur is user doesn’t have pygame installed or python itself. However, if both these conditions are satisfied, the game will start up and user will be brought to the main menu screen, where one can see the Tetris logo and a text that says “Press any key to begin.” So, do as it says and press any key to start playing Tetris. After key has been pressed the game starts and a random tile is spawned at the top, it is the user’s job to guide the piece by using the rotate function (up arrow), the move function (left and right arrow key), or the descend function (down arrow). The goal of the game is to stack the pieces in a way that allows you to completely fill a row, once a row is filled it will be cleared and points will be attributed to the player. Score varies with the number of rows broken in a single given move. Score is showcased in the right side of the screen next to the next move panel that shows the next tile that will fall, so keep on eye on it to preplan your move. Right underneath is the lines destroyed panel that tells the player how many lines it broke in this run and is a fun little addition for competitive players however unlike the score it will save the highest number achieved to display to others. On the left side you will find another Tetris logo and right underneath your high score. There is also a help panel, that reminds the user of the function of each of the commands. Back to the rules, once you start placing you tiles, you will quickly realise that the game is harder than it seems and you will start to have and endless growing pile of blocks, once a single block makes it to the top of the grid, you will lose the game. A pause menu is implemented for players that must leave but want to make sure not to lose this run, simply press p to pause, however if you exit the game or return to the main menu, **YOUR PROGRESS WON’T BE SAVED**. For more advanced players, a hold feature is in place to not have to spam your keys. That should be all, have fun!

**Design guide**

Explaining the code, the code is separated onto multiple different files; tetris.py, pieces.py, colours.py, and scores.txt.

Scores.txt is simply a text file that holds your high score and is called in tetris.py and is reiterated whenever the score is beaten.

Colours.py houses the class colours, where we keep the value for each colour, we use in our other files to save time searching for the rgb value. It also stores a colour for each piece.

Pieces.py houses the pieces class that stores the matrix design of each peace and uses the get\_shape function to randomise which piece will come out next.

Now for the way to big file for no good reason, Tetris.py, this is where all the magic happens. It starts by importing a bunch of things and should be relatively easy to understand, then we have a list of global variables that help set up the sizes for the screen and blocks. Then a whole bunch of functions and classes.

So for the rogue functions at the start:

Draw\_text\_middle : uses pygames prebuilt font function and its blit function to

Draw\_tetris\_logo: print logo in the middle top of the screen uses the same principles as draw\_text\_middle, but you must load in the image with pygames function .image.load()

Draw\_grid: takes grid predetermined cords and uses it to print out a gray lines for the grid, utilises pygames goes through I and then j because it has to go through column then row to build it one by one.

Now the DrawClass class: serves to draw out the screen when running run()

-Init class to define all the variables

-draw\_window:

Draws all the deferent things on the screen, very similar to the previous draw\_grid(), but has its own class due to being for the run() function instead of the main menu. Nothing real special except when it draws the blocks, it uses another function in the class and does it for the x and y values in range

Then theres draw\_block:

It take the coors of the top left and then the block size and then uses that spot to print in the pixel art I made for each block that follows its respective tile in a index list. I use the word print only cause it makes more sense then saying blit, in actually it blits the image to the screen.

End of the class

More rogue functions yay!:

Convert\_shape\_format

That’s the weird matrix that the shapes are made out of definded in the pieces.py

So it looks for everytime the is a 0 and appends its position in a list with its position shifted -2 from x and +4 from y so that its in the right location

Update\_score

Looks at text file scores.txt and if the value in the text is smaller than the current score rewrite the file with the new score or else keep the high score

Max\_score()

Open the scores.txt and looks for the number, that number is the high\_score

New class – GameClass

Gameclass is the class that defines the game and how it plays for the most part

We start off with the init and put in all our initial values

Create\_grid for every 10 blocks make a row 20 times

Creates grids

Valid\_space :

Looks at shape, if the x and y cords are located outside the grid, (outside their respective cords) or if the block is a full block (has colour, or not black) then that move is wrong therefore false if non of this happens then return true.

Clear\_rows:

Whenever a row has no more black blocks, add a number to rows to delete then for that many numbers delete the row and create that many new ones on top

Each time you do this add numbers to the score counter and lines deleted counter

User\_input:

Uses pygame, basically whenever a key is pressed if that key corresponds to any of the if clauses preform what it says to do.

Important to note that it checks if a move is valid and if the key is being held there is an inbuilt cooldown functions that limits the ticks per second so the key is repeated hundreds of times per frame

On\_cooldown:

Uses pygame time set-up to collect ticks in milliseconds whenever the current ticks are bigger then the cooldown + our set limit then your move is valid and can repeat itself if not then your move returns false and doesn’t happen just yet

Update\_ticks

Makes it so it counts the number of ticks in relation to pygames clock.rawtime feature, this is useful for a later function that looks at when the blocks should fall by themselves.

Update:

When the time is bigger then the fall speed that means its time to move and the block goes down by one y position until it hits a non-valid space

Change\_piece:

Looks when a move is done, so if the move stops at an unvalid location that means you’ve lost, if not the grid becomes the current pieces colour

Then your new piece becomes the next piece which is determined by the get\_shape function and we call the clear function to see if there are any rows to be deleted.

Lose:

Whenever you are sent to the lose function text appears on the middle of the screen which you update then the music stops and it stay like this for 2 seconds.

Then it looks at your score and ends your game by calling game\_over

Run():

When the game is still not over, update your frames consistently and look for player imput. This is how the game runs it also calls the draw function to draw the game.

Draw:

Calls the draw\_class sets up the game. Then for every x,y of a piece draw in the colours and custom pixel art, I keep the colours here just because the game was already set up to look for non black squares and need the colours behind the pixel art so as for no black square to exist behind and not have to change all the code.

Since run calls draw we use it to stop the screen whenever pause is called and contently update the screen to keep it paused until p us pressed again to cancel it

Main\_menu :

The interface before the start of the game, uses the 2 draw functions from the start to draw the screen and then quits then game when you close it or starts the game when you press a key

The end code is basically the code the program runs when you open the file, so it initialise all the font (text) and the music and then the game functions then opens the display and starts up the main menu which you can then start the game from.

Sources:

<http://inventwithpython.com/pygame/chapter7.html>

<https://www.youtube.com/watch?v=Et--T7SKHnk&t=282s>

<https://www.youtube.com/watch?v=juSH7hmYUGA&t=88s>

<https://www.youtube.com/watch?v=YOCt8nsQqEo>

<https://www.youtube.com/watch?v=2bnhEVPTq5g&t=437s>

<https://www.youtube.com/watch?v=NmCCQxVBfyM>

mainly used this for the skeleton:

<https://www.youtube.com/watch?v=zfvxp7PgQ6c&t=4951s>

I built most my code with this video to have a base for the game and understand how the game should work, then went through it and start fixing all the problems, which there were a lot and added all the features you see in the game such as score and music and instructions etc… there were many problems with the clear function that needed changes and the files needed some reorganising though I could’ve done a lot more of that, but what I did do was separate the things into classes which was not done in the original video. I also added my own art to the blocks which was fun. The code also wasn’t structured in a way I liked so the main for example became the run and was turned from 100+ lines to 10 and whatever was in the 100 lines became its own functions to have a clearer graphs of what everything does.

And way to many reddit and git hub links that I cant even sort them all out, I think I left some links in the code itself

most of the time I wasn’t able to find anything good but it did greatly inspire me.

And most importantly:

https://www.pygame.org/news