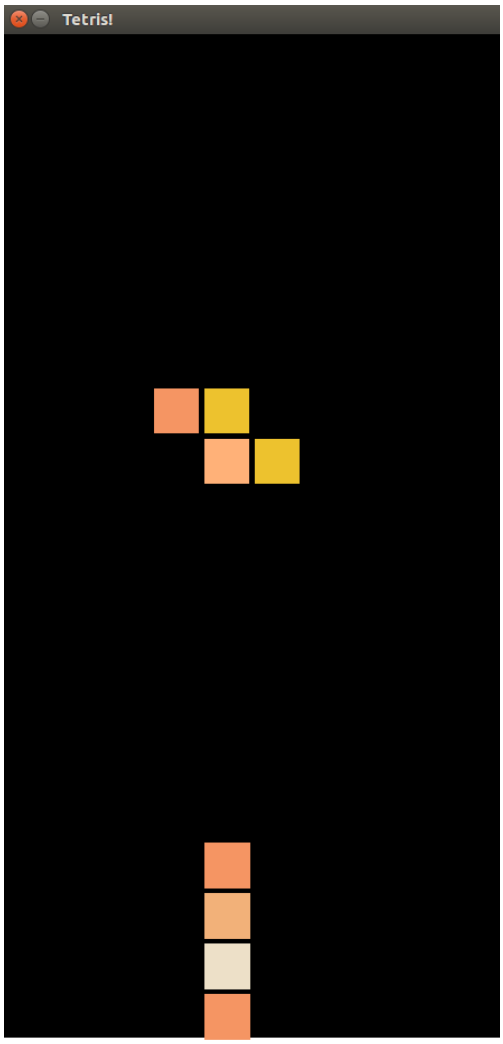
Tetris – Emma Pan, Andrew Schnurr

Project Overview:

For this project, we decided to try and combine two popular video games, Tetris and 2048, into one game using pygame. We had the idea that we would start by creating Tetris, and from there implement the 2048 aspect of combining blocks of equal value. In the end, we decided to just try and create Tetris as it proved more challenging than we had thought to get pieces to move on a board, and thus our final product is a version of Tetris.

Results:

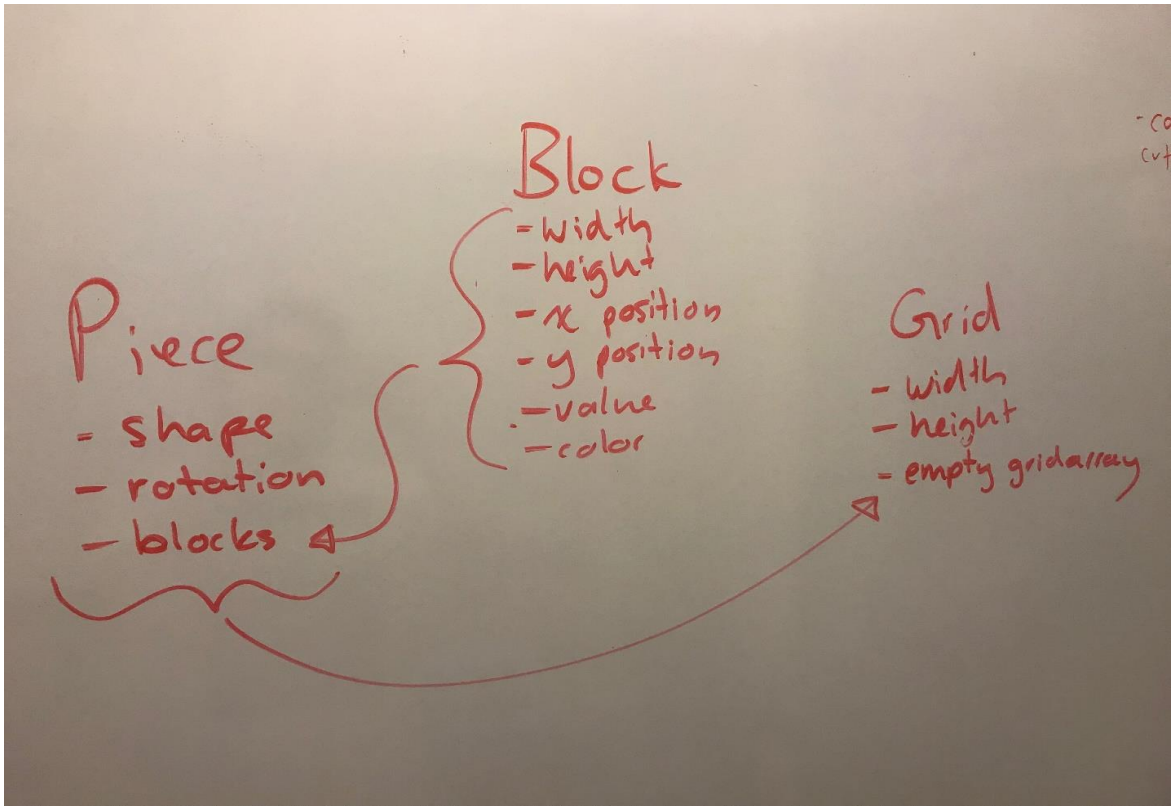
We ended up creating a Tetris simulation, where randomized blocks fall, and are then stacked upon each other to make it more difficult to find spaces to put blocks. The final product turned out well, and in the image below you can see the final design of the game



Implementation:

Our goal for this project was to utilize Class objects to more easily and efficiently write a program that has pieces that fall down the screen, and can be moved left, right, and downwards. To do this, we created three main classes, Block, Piece, and Grid. The Block class is the most fundamental class in our program, and it describes the individual blocks that make up pieces, and give the dimensions that create the grid that is played on. When a block is created it is given a width, height, x coordinate, y coordinate, value, and color. The value of a block is either a 1 or a 0, and blocks that 'take up space' are assigned a value of 1, whereas other spaces have a value of 0.

The blocks are used to create the Pieces class, and the Pieces objects are simply a certain



arrangement of blocks. We created a dictionary of several shapes, and within that a list of orientations that those shapes can have. A Piece object has the attributes shape, rotation, and blocks.

Finally we have the Grid class. This class is used to define the grid that the pieces fall downwards on, and has the attributes width, height, and also includes an empty grid array that contains the shapes and occupied blocks. One decision we had to make was how to rotate the pieces. We could either apply a rotation matrix to the coordinates of each piece, or create a different set of assigned blocks that the falling block can switch between. We ended up choosing the second option because operating on coordinates with matrices seemed very difficult, and it was relatively easy to just hard code the other orientations in.

Reflection:

One of the biggest issues with our project was the scope. We had assumed at the beginning that it wouldn't be very difficult to get a version of Tetris up and running, and from there we assumed that it wouldn't be that hard to assign values to each block and essentially we assumed that we could just adapt a simple Tetris program. However, we soon realized that creating a version of Tetris with pygame was not as easy as it sounded. We got caught up in trying to make a program that worked, without really testing it along the way.

One improvement could be spending more time initially really making sure that we knew what we were getting into, and scoping the project appropriately. The team process was good, we did almost all of the work through pair programming, and this was very effective because we caught many errors before they became bigger problems. Something that would be helpful to do next time is to plan out a full scaffolding of our code before we started coding.