Game Engine Final Exam Practical Document
Andrew Oster 100825248

A Tetromino is spawned every 5 seconds and can be spawned with the W key.

Controls:
A to move left, D to move right
Q to rotate left, E to rotate right (for white; black is the reverse of this)

Command Pattern:

The command pattern was implemented via the object movement and turning system. For this I split up all of the movement functions into commands. These commands are moveLeft, moveRight, turnLeft, and turnRight. The InputHandler stores these commands and then when executed, an Execute function in the command is called that calls a method in the player Controller script. These methods in the player controller script then call methods in the current tetromino on screen. It was done this way to add encapsulation into the game and so that the reverse input turning can be done easier. Depending on the current tetromino on the screen, their rotation is either normal or swapped. This was done using the Command Pattern by first checking what type of tetromino is on screen then calling either the rightTurn or leftTurn method when pressing the equivalent keys.

Object Pooling:

The Object pooling pattern was implemented via the tetromino summoning system. At the start of the game, an object pool is created with 20 tetrominoes in it. These objects can be one of the two tetrominoes in the game and it fills the object pool with a random amount of both objects until 20 total objects have been reached. This was done to save on memory since instantiating and deleting will happen a lot in this game. Once a tetromino has run out of cubes that make up its shape, it gets pushed back into the pool.

Additional Design Implementations:

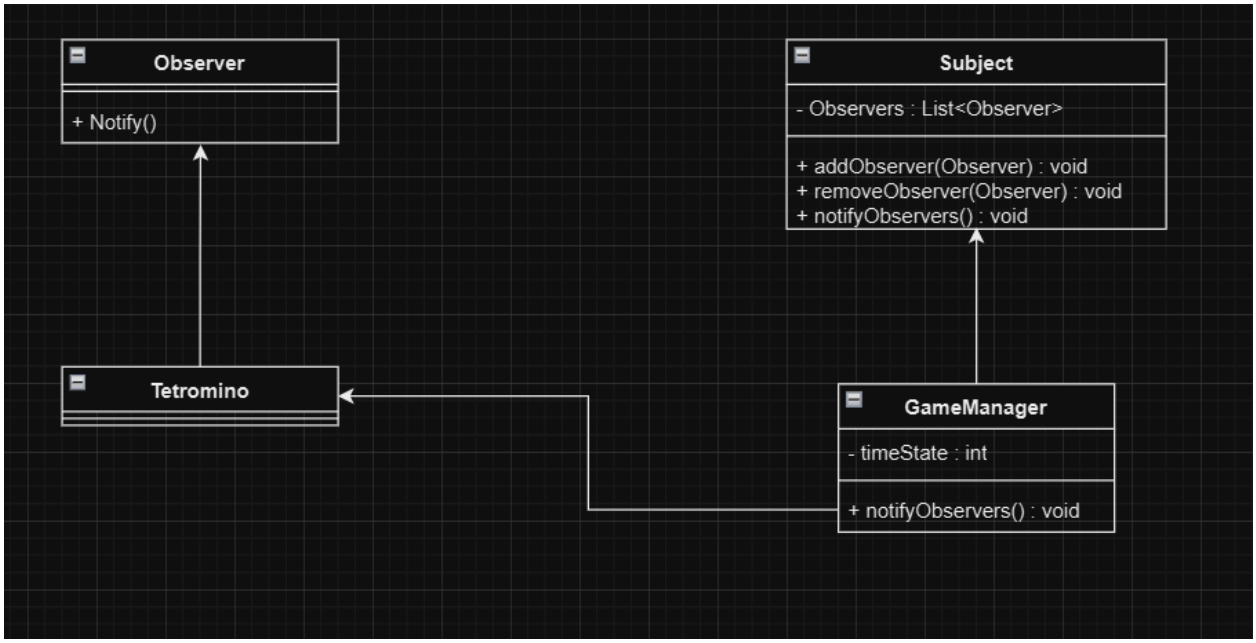
The additional implementation I implemented was the singleton design pattern. I added this onto the Object Pool script for multiple reasons. One reason was so that there is only ever going to be one object pool which will save on memory and prevent issues in the future, but also so that it turns into a global instance. This is needed so that the operations that push and pull the objects from the pool can be called anywhere in the code.

Tetromino System:

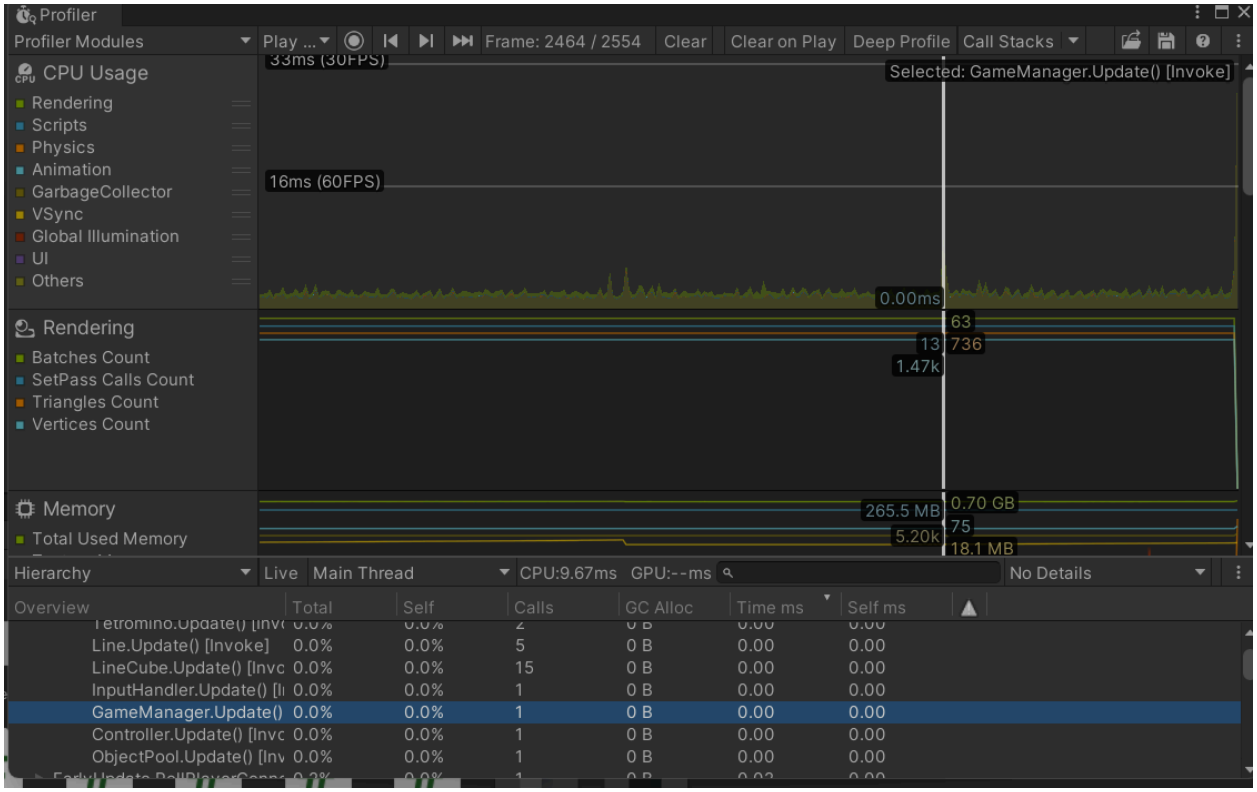
The tetromino system is done by first creating the prefab for the two variants of the tetrominoes. These prefabs are composed of three different cubes that are parented to an empty game object; these objects, when enabled, constantly fall and move using the command pattern described above. When they touch a lane cube, the cube of the tetromino that touches it gets set to be in that cube internally. Once an entire row has these cubes, it sets them as inactive. A Tetromino is called every 5 seconds and can be spawned with W.

Observer Pattern:

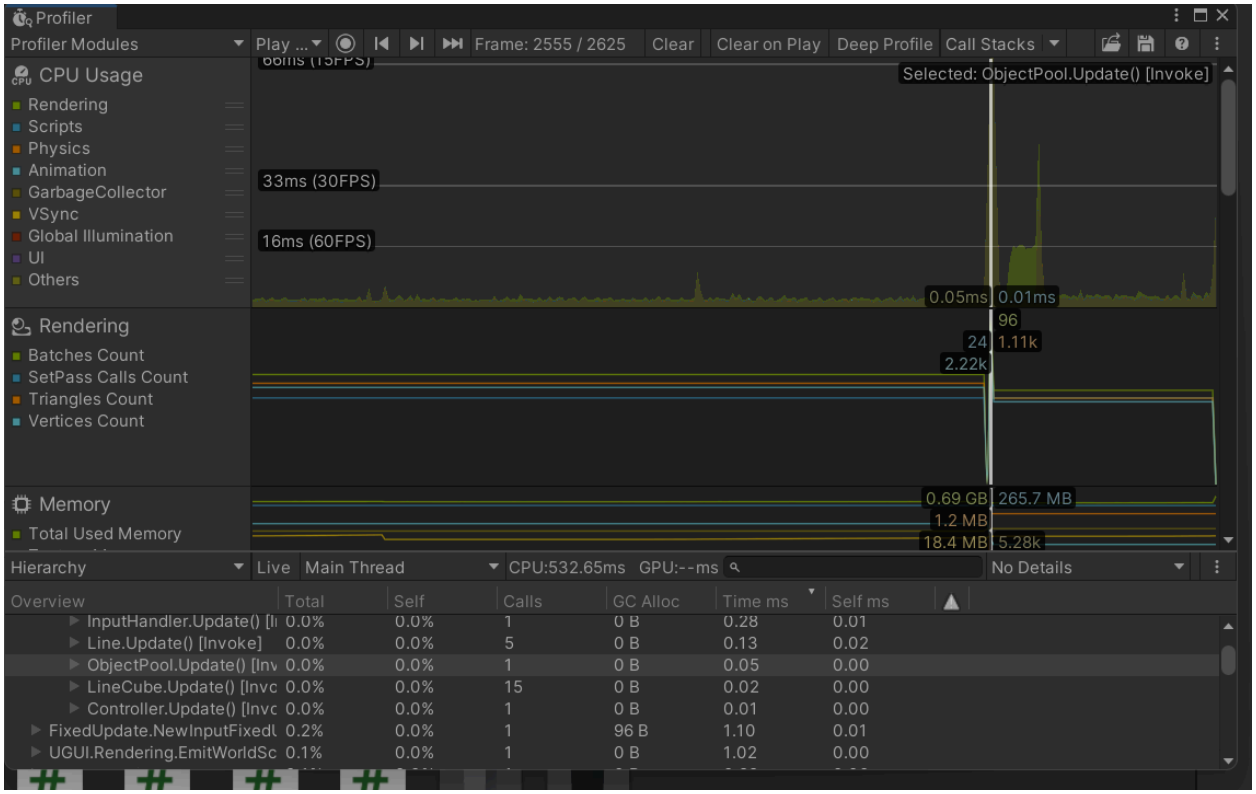
I did not implement the observer design pattern, however the Professor said it was fine to describe how I would implement it. To implement this design pattern I could do multiple things. I could make every tetromino an observer and then the script that handles the time can be the subject. The subject would notify the observers at different time intervals. By notifying the observers of this, a system can be implemented where every 10, 20, 30, etc. seconds the tetrominoes would move faster.



Performance Profiling:



This screenshot shows what happens when a tetromino is called from the object pool. It does not take up any memory for Garbage Allocation or any time in ms.



Even when the game is started and it initializes the object pool, it takes up a very small amount of memory and time.