CMPU203 Maya T. Hawkins

Feeding Time: TheGame

The game that I have created is called Feeding Time, and it is mentally a pretty simple game. It includes the player, called mousePlayer. It is the player's dream to be eaten by a cat, that is on the other side of the game board. The player's job is to move the mouse over to the cat. The only problem is that random Sticky Paper will spawn on the field, and the player must move around those to get to the cat. If the player touches a Sticky Paper, then the player will die, lose a life and be respawned at the beginning. Yet a dead mouse will be left behind where the player died. If the player touches Sticky Paper five times and loses all five of their lives, then the game is over and the player loses. If the player touches a dead mouse, however, they will lose all of their lives, because they ripped a whole in the space time continuum. The player will then lose. If the player manages to get eaten by the cat, then they will be respawned at the beginning, the cat will move to a different place on the other side of the board, and the board will reset of Sticky Paper and dead mice. Everytime a cat is fed, the Sticky Paper will spawn faster.

All physical items in this game (mouse, stickypaper, deadmouse, cat) are represented by blocks, and the player only controls the movements of the mouse block. All 4 items are represented by different colors: the mouse is blue, the cat is red, the sticky paper is yellow, and the dead mouse is gray. This is shown on a 500X500 background of a magenta color. The mouse only moves by one pixel each time, as shown by the mouseMove function(see lines 35-55).

The StickyPaper and the DeadMouse is represented as arrays of blocks. The stuckHuh function(see lines 79-84) are tests if a mouse has touched an individual stickyPaper, and the lawsOfLife function(see lines 143-148) test if a mouse had touched an individual DeadMouse.

From lines 193 to 206 is the function stuckHuhCheck, which tests an array of stickyPaper. From the lines 208 to 221, the function lawsOfLifeCheck tests an array of DeadMouse.

The cat is an individual block, and the function feedCat (106-110) checks if the mouse has touched, and therefore fed, the cat. To test that the function works properly, I made the tester catFeedTester(see lines 337-352). The function creates a mouse and cat, and will calculate the number of steps needed to call true for feedCat. The function moves the mouse right until it calls true for catFeed, and will then tell the estimated steps and the actual number of steps it took for the mouse to reach the cat. It then moves the mouse back, and makes a new place for the mouse within the sight of the mouse, and tests again 20 times. This tester is effective, because it will give a different output if the counter reaches the same number as the estimated number of moves, which means that it did not see the cat, or started to pass through it. Even though the tester is not currently working, in theory, it should work. The problem with the tester is that it is going through an endless loop without checking each time whether feedCat is true.

The testers for lawsOfLife and stuchHuh are very similar to the catFeedTester. They consist of creating arrays with the said object in front of the mouse and moving the mouse right. If the mouse calls true on the function lawsOfLive and stuckHuh, a printout will say so. Yet since they are similar to the catFeedTester, they also have the same problem and therefore are not in working conditions just yet.

Unfortunately, not everything has been tested just yet. If more testers were to be created, there is one that would test the generation of random StickyPaper and then move the mouse through every position until it reaches the very end of the function or calls stuckHuhCheck, which would mean that a random stickypaper was in fact generated. Another tester that would be useful is a one that would test that the game ends once the lives decrease to zero, which could

be assimilated with continually running the random stickypaper tester until the lives would call zero, and if a worldend is called, then the function would call true. Naturally, this project requires much more work and time than what was given to it so far. With a bit more time, then it would become a full functioning program. Besides that, it was not only a good learning experience, but an eye opener, as programs should be.