Marcos Remy

GLUT and OpenGL were used to create the Pacman game using c++.

I created classes for the Ghost and Pacman

```
struct Ghost
{
    int row; //(row, col) = position on the grid
    int col;
    int speed; //ghost speed percentage per frame (should always be le
    int row_percentage; //position percentage between two cells (value
    int col_percentage;
    int exit_counter;
    Direction dir; //the direction the ghost is traveling; set to NUM_
    RGB color; // the color of the ghost when in the "normal" state
    bool frightened; //false indicates the ghost is after the player t
    std::string name; //the name of the ghost (blinky, clyde, ect...)
    GhostAI ai; //the AI that this ghost uses

    std::mutex* gMutex;
    std::condition_variable* gCV;
    bool ready;

    Ghost();

    void ghostAI(GameWorld& gw);
};

#endif // GHOST_H
```

What's interesting here is how I added a mutex lock and condition variable so that when each Ghost is on its own thread they won't access one another at the same time which would cause a crash.

```
10  Object data structures
11  ass PacMan
12
13    private:
14        bool keys[NUM_DIRECTIONS];
15
16    public:
17        PacMan();
18
19        int row; //(row, col) = position on the grid
20        int col; //note that if the row is set to be 3.5 it means that pacman is between row 3 and row 4
21        int speed; //pacman speed percentage per frame (should always be less than 100 but greater than 0)
22        int squares_traveled; //the number of squares that pacman has traveled
23
24        int row_percentage;
25        int col_percentage;
26
27        Direction dir; //direction pacman faces
28        Direction requested_dir; //the direction requested by the player
29        bool isAnimated; //constrols whether or not PacMan is animating or now
30        int isLive; //determines if the thread is running or not
31
32        std::mutex pacMutex;
33        std::condition_variable pacCV;
34        bool pacReady;
35
36        void key_update(unsigned char key, bool keydown); //used to change the direction of pacman with keyboard
37        void checkUserInput();
38
39        void update(GameWorld& gw); //this function is called every update cycle
40
41
42
```

```
/**
 * @brief update function for pacman and the ghosts
 */

void updatePacAndGhosts(void)
{

    //Pacman Threading:
    {

        std::lock_guard lk(gameWorld.pacman->pacMutex);
        gameWorld.pacman->pacReady = true;
        gameWorld.pacman->pacCV.notify_one();

    }

    //Ghost Threading:
    {

        for (int i = 0; i < gameWorld.ghost_count; i++) {

            std::lock_guard<std::mutex> lk( *gameWorld.ghost_array[i].gMutex );
            gameWorld.ghost_array[i].ready = true;
            gameWorld.ghost_array[i].gCV->notify_one();

        }

    }

    checkLoadNext();

    //Check Game Over:
    if (running == false) {
```

Here is how I handled each thread, as the ghost must acquire a lock it'll notify it's condition variable it's ready and once that condition variable is done operating it notifies the next one.

```
void updateGhosts(int k) { //Ghost Thread Function

    while (running == true) {

        std::unique_lock<std::mutex> lk( *gameWorld.ghost_array[k].gMutex );
        gameWorld.ghost_array[k].gCV->wait(lk, [&]() {return gameWorld.ghost_array[k].ready;});

        //If Ghost is hit while frightened return back to center
        if (gameWorld.pacman->row == gameWorld.ghost_array[k].row && gameWorld.pacman->col == gameWorld.ghost_a

            if (gameWorld.ghost_array[k].frightened == true) {

                gameWorld.ghost_array[k].row = 13;
                gameWorld.ghost_array[k].col = 9;
                gameWorld.ghost_array[k].frightened = false;

            } else {//if not frightned return to center and subtract from lives

                gameWorld.ghost_array[k].row = 13;
                gameWorld.ghost_array[k].col = 9;
                pacman_lives--;

            }

        }

        gameWorld.ghost_array[k].ghostAI(gameWorld);

        //Ghost has been updated now unlock and wait for render function to do work.
        gameWorld.ghost_array[k].ready = false;
        lk.unlock();

        gameWorld.ghost_array[k].gCV->notify_one();

    }
```

Here's what's being run by each of the ghost threads to move ghosts around and change their data.