Project Report

First meeting:

The first thing we started doing was initializing one level, one scene, and one view in the main, in addition to reading the board data from the text file we have done previously. All of these were in the main function. We then made a class called "constants," we did its header file, where we initialized all the constants we needed to use throughout the whole project to make it easier in the long run. In that class, we initialized the directory of the files we will use for our GUI, in addition to the dimensions of the screen, board, tiles, enemies, power pellets, and bullets. Moreover, we added the UI basic format in that class as well. Moving on, we made four classes the bullet, enemy, power pellet, and player classes, each having its own header and CPP files. For the bullet, enemy, and power pellet classes, we set the image and position of each. We did the same thing in the player class; however, we added the implementation of the movement of the player, which takes its input from the keyboard and checks the board data to see if the movement is possible or not. We also made a handle collisions function, where we checked if the player collided with a bullet or power pellet in removed the item from the screen. If it were a power pellet, the god mode would be true. If it were a bullet attack, the function would be called. Finally, if it were an enemy, the health of the player would decrease by 1, and the enemy would still be there. We finally added the enemies' random movement to the main. In the end, we changed the scene, view, and level from the main to the game class. In the game class we added 4 functions, the load level, load resources, show, and finally, watch functions. The load level function loaded the level by reading the text file "board data". The load resources loaded all the items of the game to the board by checking the board data; for instance, if it was -1 then that would be the wall image "obstacle". This function also set the player in the right position. The show function set the scene and showed it. We changed the enemies' movement from the main to the watch function. However, it still does not work properly, although our logic does make sense.

Second meeting:

In the game class, we added multiple functions to load each item separately, and then we added a function called “display game over window,” which is called when the player’s health is equal to zero, called in the watch. We send a Qstring “GAME OVER,” where we disable all the items in the scene and display the text Game over. Then we decided to make a new class called “game window” to replace the game class. We made 1 view and multiple scenes and levels in this class, which allowed us to have multiple levels and update the scene accordingly and according to whether the player wins or loses. We implemented Slots and signals to be able to update and change between the different scenes using buttons which we did a class for. The button class simply has logic for the shape and settings of the buttons we used. Back to the game window class, we implemented a function called “game over,” which gave the user a choice between two options using the buttons either to retry or return to the levels screen. Finally, we were able in this class to read the levels automatically from the folders. Because we did the “game window” class to replace the “game” class, we needed another class, “level,” to load the images and data separate from the game window class as we have multiple levels and scenes now. We updated the textures.

Third meeting:

This time we updated the textures again hopefully, for the last time, we decided to make the game home-alone themed. We figured out the delay function, and we discovered that this function is executed in a different thread, which will allow us to have the function work in parallel to other functions of the program. We used this to our advantage by using it in the watch function, which is supposed to be responsible for the enemies' random movement. However, this function is still not working, and we are stuck on it. We have no idea we it does not work. The watch function is in the level class.

Fourth and final meeting:

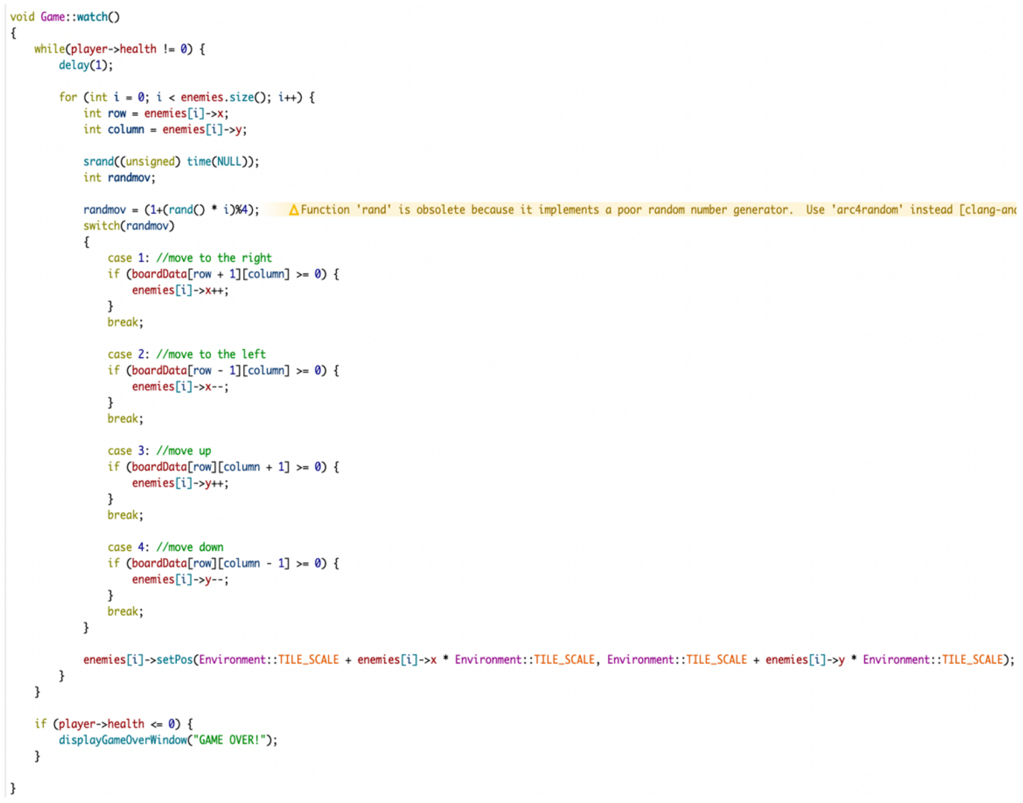
Today was basically a full day of work. We finally figured out the watch function. Our error was switching the rows and columns. That's why our enemies were behaving weirdly. We did a function called "change app" in the player class to enable us to change the image of the player when he collided with the weapon. We also modified the delay function to work with milliseconds instead of seconds as it did not allow for "double" and we wanted some delays to be faster than a second. We implemented the logic of the attack function which takes a list of pointers of enemy and loop on the number of enemies to get their Euclidian distance and damage the enemy closer to the player. Damage is a function in the enemy class which decreases the life of the enemy by 1, the enemy has to lives. We called this function in the handle collisions one. We did a function in the enemy class to change its appearance when damaged similar to the one in the player class discussed above. We also updated the GUI to show the three lives of the player. We did both the winning and losing screens, implementing them in the game window class. We changed the watch function into two functions watch and handle enemies.
We organized the code.

Others:

We did multiple lists for the bullets enemies, etc. to be able to make more levels in the future.
P.S: in the constants class the directories we made did not work on a mac device and they needed to be changed to the paths of the file. (provided below screenshots for elaboration)

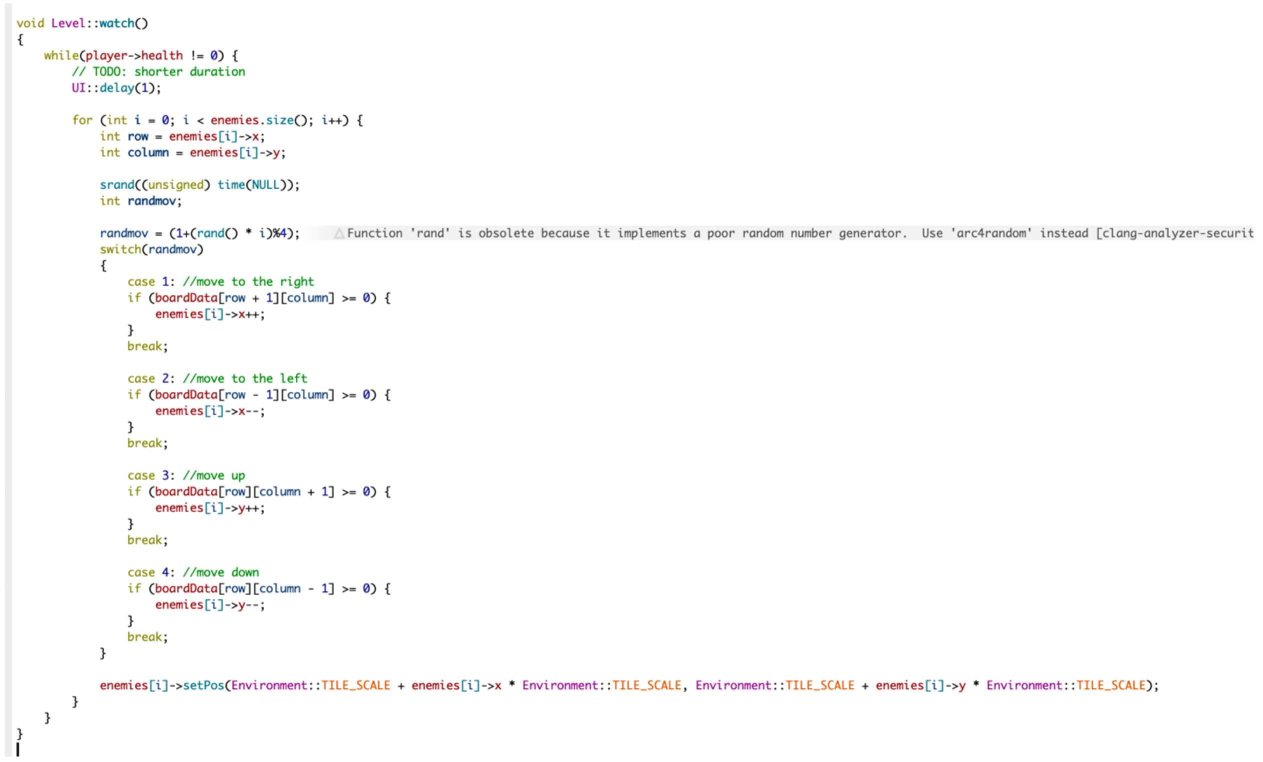
Screenshots:

Watch func() in game class



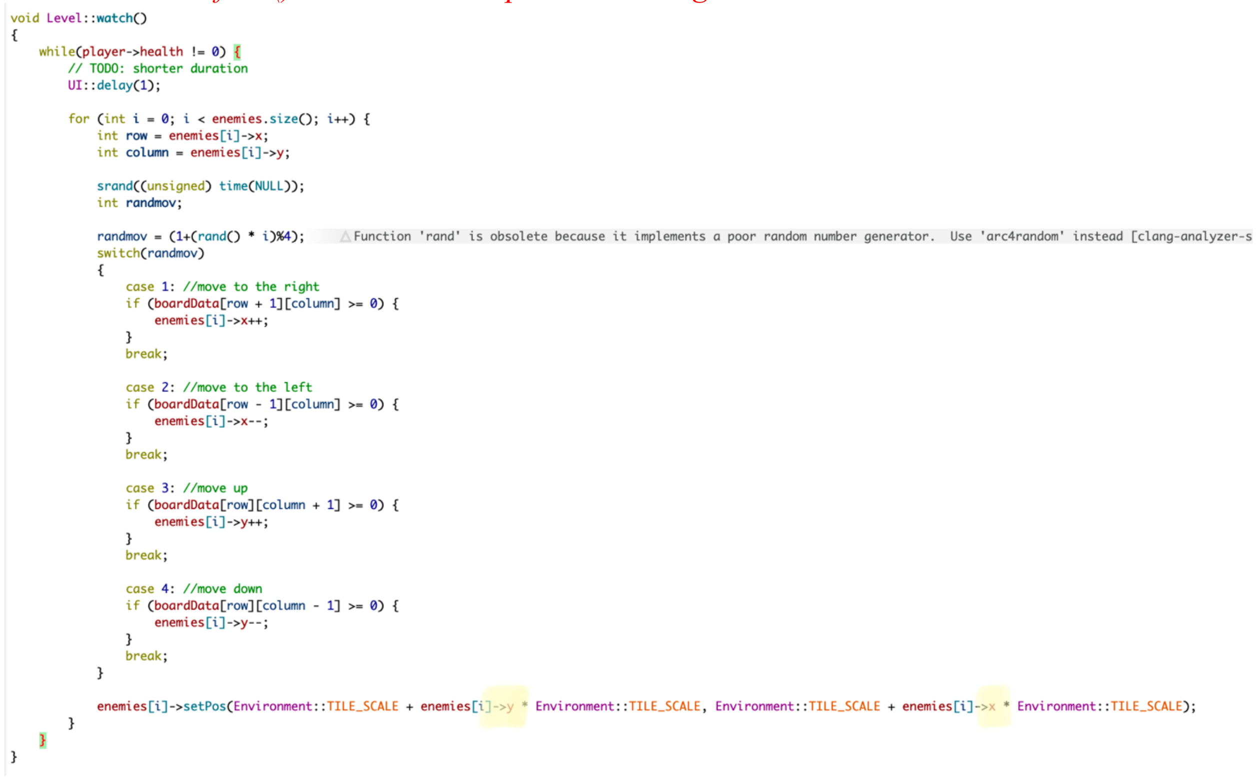
```
void Game::watch()
{
    while(player->health != 0) {
        delay(1);

        for (int i = 0; i < enemies.size(); i++) {
            int row = enemies[i]->x;
            int column = enemies[i]->y;

            srand((unsigned) time(NULL));
            int randmov;

            randmov = (1+(rand() * i)%4);    ⚠Function 'rand' is obsolete because it implements a poor random number generator.  Use 'arc4random' instead [clang-anc
            switch(randmov)
            {
                case 1: //move to the right
                if (boardData[row + 1][column] >= 0) {
                    enemies[i]->x++;
                }
                break;

                case 2: //move to the left
                if (boardData[row - 1][column] >= 0) {
                    enemies[i]->x--;
                }
                break;

                case 3: //move up
                if (boardData[row][column + 1] >= 0) {
                    enemies[i]->y++;
                }
                break;

                case 4: //move down
                if (boardData[row][column - 1] >= 0) {
                    enemies[i]->y--;
                }
                break;
            }

            enemies[i]->setPos(Environment::TILE_SCALE + enemies[i]->x * Environment::TILE_SCALE, Environment::TILE_SCALE + enemies[i]->y * Environment::TILE_SCALE);
        }
    }

    if (player->health <= 0) {
        displayGameOverWindow("GAME OVER!");
    }

}
```

watch func() in level class:



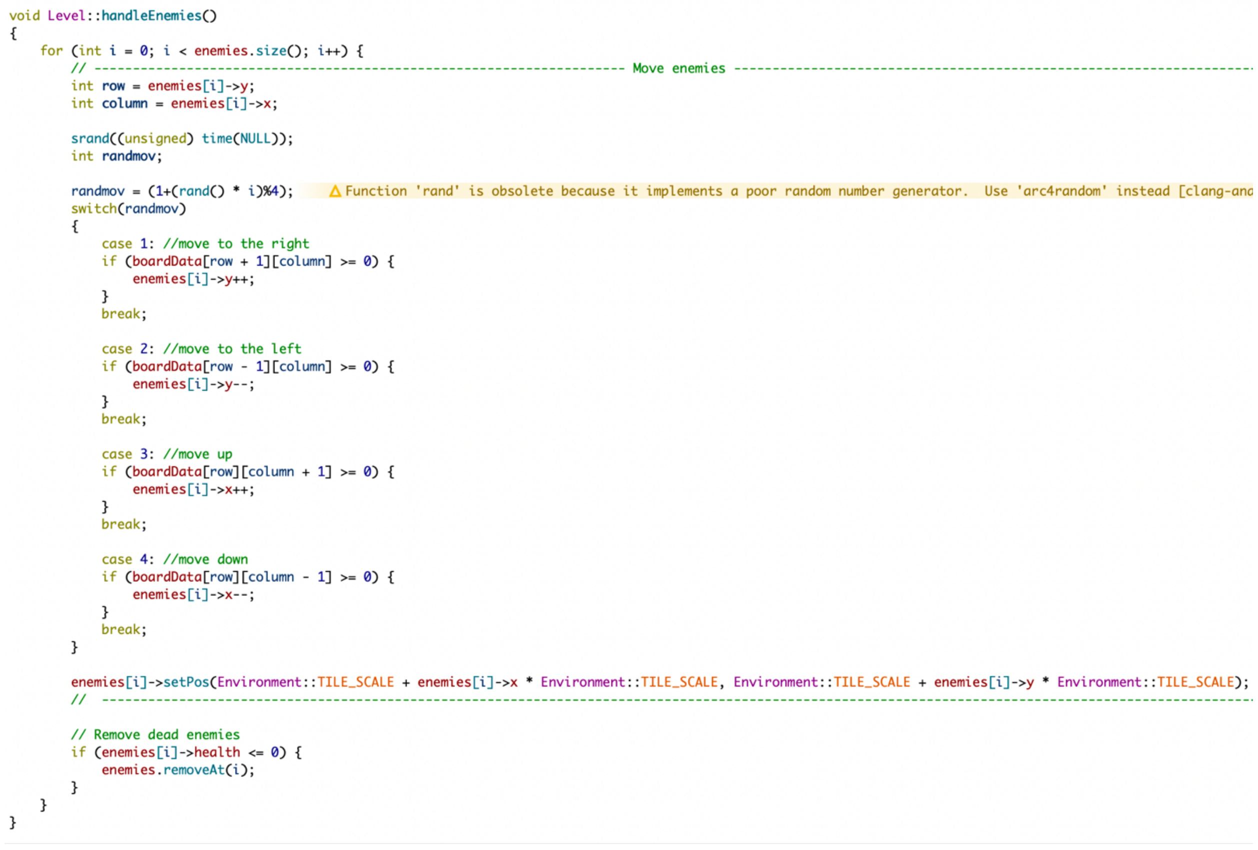
```
void Level::watch()
{
    while(player->health != 0) {
        // TODO: shorter duration
        UI::delay(1);

        for (int i = 0; i < enemies.size(); i++) {
            int row = enemies[i]->x;
            int column = enemies[i]->y;

            srand((unsigned) time(NULL));
            int randmov;

            randmov = (1+(rand() * i)%4);    ⚠Function 'rand' is obsolete because it implements a poor random number generator.  Use 'arc4random' instead [clang-analyzer-securit
            switch(randmov)
            {
                case 1: //move to the right
                if (boardData[row + 1][column] >= 0) {
                    enemies[i]->x++;
                }
                break;

                case 2: //move to the left
                if (boardData[row - 1][column] >= 0) {
                    enemies[i]->x--;
                }
                break;

                case 3: //move up
                if (boardData[row][column + 1] >= 0) {
                    enemies[i]->y++;
                }
                break;

                case 4: //move down
                if (boardData[row][column - 1] >= 0) {
                    enemies[i]->y--;
                }
                break;
            }

            enemies[i]->setPos(Environment::TILE_SCALE + enemies[i]->x * Environment::TILE_SCALE, Environment::TILE_SCALE + enemies[i]->y * Environment::TILE_SCALE);
        }
    }
}
```

*Watch func() in level class **positions changed:*



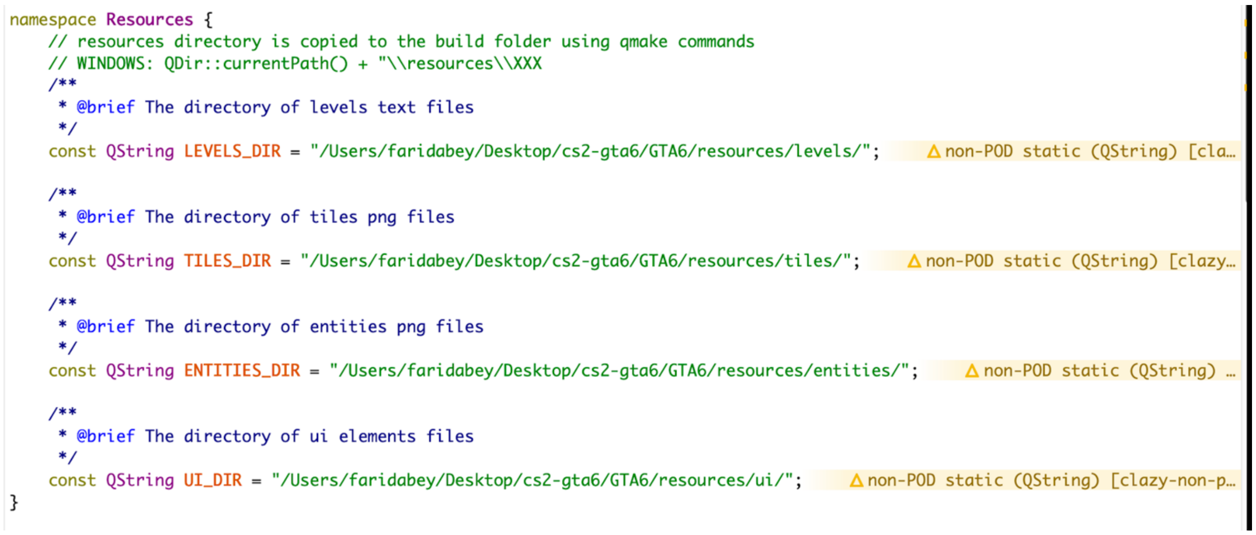
```
void Level::watch()
{
    while(player->health != 0) {
        // TODO: shorter duration
        UI::delay(1);

        for (int i = 0; i < enemies.size(); i++) {
            int row = enemies[i]->x;
            int column = enemies[i]->y;

            srand((unsigned) time(NULL));
            int randmov;

            randmov = (1+(rand() * i)%4);    ⚠Function 'rand' is obsolete because it implements a poor random number generator.  Use 'arc4random' instead [clang-analyzer-s
            switch(randmov)
            {
                case 1: //move to the right
                if (boardData[row + 1][column] >= 0) {
                    enemies[i]->x++;
                }
                break;

                case 2: //move to the left
                if (boardData[row - 1][column] >= 0) {
                    enemies[i]->x--;
                }
                break;

                case 3: //move up
                if (boardData[row][column + 1] >= 0) {
                    enemies[i]->y++;
                }
                break;

                case 4: //move down
                if (boardData[row][column - 1] >= 0) {
                    enemies[i]->y--;
                }
                break;
            }

            enemies[i]->setPos(Environment::TILE_SCALE + enemies[i]->y * Environment::TILE_SCALE, Environment::TILE_SCALE + enemies[i]->x * Environment::TILE_SCALE);
        }
    }
}
```

*Handle enemies in level class **positions changed final "the working one:*



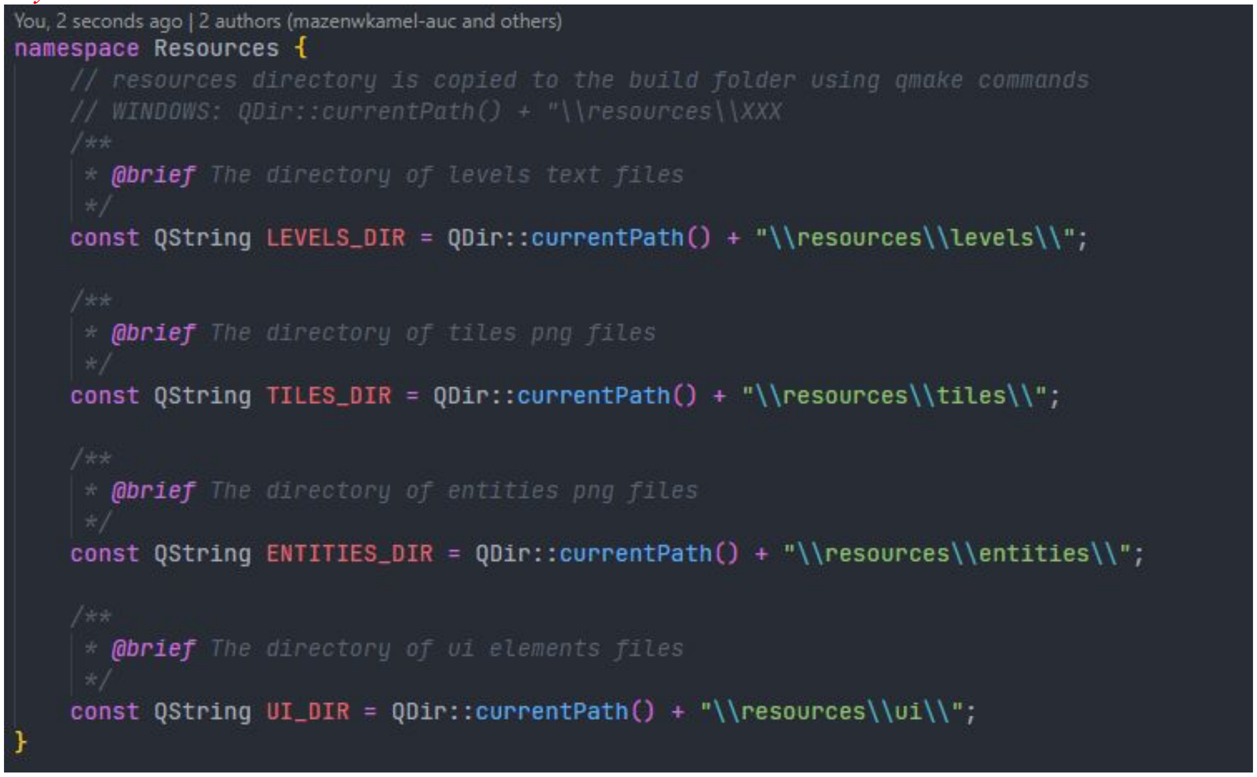
```
void Level::handleEnemies()
{
    for (int i = 0; i < enemies.size(); i++) {
        // ---------------------------------------------------------------------- Move enemies ----------------------------------------------------------------------
        int row = enemies[i]->y;
        int column = enemies[i]->x;

        srand((unsigned) time(NULL));
        int randmov;

        randmov = (1+(rand() * i)%4);    ⚠Function 'rand' is obsolete because it implements a poor random number generator.  Use 'arc4random' instead [clang-ana
        switch(randmov)
        {
            case 1: //move to the right
            if (boardData[row + 1][column] >= 0) {
                enemies[i]->y++;
            }
            break;

            case 2: //move to the left
            if (boardData[row - 1][column] >= 0) {
                enemies[i]->y--;
            }
            break;

            case 3: //move up
            if (boardData[row][column + 1] >= 0) {
                enemies[i]->x++;
            }
            break;

            case 4: //move down
            if (boardData[row][column - 1] >= 0) {
                enemies[i]->x--;
            }
            break;
        }

        enemies[i]->setPos(Environment::TILE_SCALE + enemies[i]->x * Environment::TILE_SCALE, Environment::TILE_SCALE + enemies[i]->y * Environment::TILE_SCALE);
        // ----------------------------------------------------------------------------------------------------------------------------------------------------------

        // Remove dead enemies
        if (enemies[i]->health <= 0) {
            enemies.removeAt(i);
        }
    }
}
```

mac constants version:



```
namespace Resources {
    // resources directory is copied to the build folder using qmake commands
    // WINDOWS: QDir::currentPath() + "\\resources\\XXX
    /**
     * @brief The directory of levels text files
     */
    const QString LEVELS_DIR = "/Users/faridabey/Desktop/cs2-gta6/GTA6/resources/levels/";    ⚠ non-POD static (QString) [cla…

    /**
     * @brief The directory of tiles png files
     */
    const QString TILES_DIR = "/Users/faridabey/Desktop/cs2-gta6/GTA6/resources/tiles/";    ⚠ non-POD static (QString) [clazy…

    /**
     * @brief The directory of entities png files
     */
    const QString ENTITIES_DIR = "/Users/faridabey/Desktop/cs2-gta6/GTA6/resources/entities/";    ⚠ non-POD static (QString) …

    /**
     * @brief The directory of ui elements files
     */
    const QString UI_DIR = "/Users/faridabey/Desktop/cs2-gta6/GTA6/resources/ui/";    ⚠ non-POD static (QString) [clazy-non-p…
}
```

any other devices version:



```
You, 2 seconds ago | 2 authors (mazenwkamel-auc and others)
namespace Resources {
    // resources directory is copied to the build folder using qmake commands
    // WINDOWS: QDir::currentPath() + "\\resources\\XXX
    /**
     * @brief The directory of levels text files
     */
    const QString LEVELS_DIR = QDir::currentPath() + "\\resources\\levels\\";

    /**
     * @brief The directory of tiles png files
     */
    const QString TILES_DIR = QDir::currentPath() + "\\resources\\tiles\\";

    /**
     * @brief The directory of entities png files
     */
    const QString ENTITIES_DIR = QDir::currentPath() + "\\resources\\entities\\";

    /**
     * @brief The directory of ui elements files
     */
    const QString UI_DIR = QDir::currentPath() + "\\resources\\ui\\";
}
```