Sprint 1 Code Review

Authored by Dennis Kovarik

<u>Bugs</u>

- In tankTesting.cpp
    - In TEST_CASE(“Test Tank Upward movement”)
        - In SECTION(“ Moving tank up 1 position. Expected: true, Y = 3, Direction = 0 ")
            - Bug: should check testTank7.getY()==3 not testTank7.getY()==2.
    - The catch tests for the death stat need to be replaced since the health stat is being used instead.

- In gameTesting.cpp
    - In TEST_CASE("Testing get and set dimensions sane limits")
        - SECTION("Testing invalid size Y, 20 x 11. Expected result: setting false, getting true, y != 11 ")
            - Bug: This test was checking the X value instead of the Y value causing a false positive result. Now that it is checking the Y value the test is failing with Y being set to 11.
- In classGame.cpp
    - Destructor for gameSetting class would only free 2D array if “map == NULL”. It should instead check that map != Null.

<u>To Do</u>

- While logging in there was a message about Auto DevOps being used if CI wasn't configured. DevOps needs to follow up with this message to see how it affects our current pipeline/CI tests and if it is something we need to look into using

<u>Request For Change</u>

- Swap the deaths with health as discussed in the group meeting. We would now have health points instead of deaths and we would just go backwards in theory from what we have up currently.
- In classTank
    - Add a default value of 1 damage so that takeDamage can be called by either takeDamage() or takeDamage(1).
- In tankTesting.cpp
    - TEST_CASE("Test tank Direction")
        - SECTION(" Setting direction to 0. Expected: true, direction = 0")

            - Might want to add a check on testTank.setdirection(2) to make sure that step doesn't fail, or if 2 fails don't test the rest of change direction that depend on it. Same for other occurances of "Only valid if ..."
    - TEST_CASE("Test Tank Down movement")
        - SECTION(" Trying to move off the map. Expected: false ")
            - Does not check for the expected tank position after the invalid move has been made.
    - TEST_CASE("Test Tank Left movement")
        - SECTION(" Trying to move off grid by no value. Expected: false ")
            - Does not check for the expected tank position after the invalid move has been made.
- In default.map
    - Since the map size is now fixed, the map files no longer need to include the map dimensions.
- In classGame.cpp
    - In bool gameSetting::extractGameInfo (ifstream &map_file)
        - Since the map size is going to be fixed, this function no longer needs to extract the maps size dimensions.

<u>Feature Requests</u>

- It may be helpful for there to one place to store the fixed map dimensions. This way classes and functions can use this to reference instead of these values being hard-coded in. This way if the size of the map ever changes, then there will be less code to re-write.
- tankTesting.cpp
    - If fixed map size ever becomes larger than 20x10, then the sections "Checking out of range X value, too high: 21" and "Checking out of range Y value, too high: 11" in test case "Create a tank with starting inputs" will no longer be valid.

<u>Unlabeled</u>

- tankTesting.cpp
- "Checking negative X Value: -10" and "Checking out of range Y/X Value: 200". For the out of range values does the tanks class set them equal to something thats in range? Just a couple comments would make it easier for people to understand what you expect to happen.
- Function headers might help, although most section titles have a good description, a couple could use something like "expected values to return to default"
    - In TEST_CASE("Test Tank Speed")
        - SECTION("Setting speed to negative value. (expect default 1)")
            - The setSpeed(-1) statement will set the tank's speed to 0 which the getSpeed() function will return. The secton checks that the return value for getSpeed() == 0, but this contradicts the section heading.
    - In TEST_CASE("Test Tank colors")

        - Checking setting Colors to 255, 255, 255" is redundant? It seems like this test can be covered by just "Checking all values stayed the same: 255, 255, 255" or "Checking no values changed: 255, 255, 255"
- Catch testing files
    - Need to improve the comments inside of testing files to better follow the teams selected comment format and improve readability when looking at the code.
- classTank.cpp
    - tank::getColors - What happens if the colors were never set?
    - tank::setPos - Upper bounds? What if I set x to 2000?
    - Tank Movement - when the tank moves, shouldn't the direction change with it?
    - May make more sense to rename "death" variables and functions to "lives"
    - documentation on moves the tank left by subtracting x from the vertical position.. Left should be replaced with "horizontal position"
    - tank direction should change when using the move__ commands?
- classGame.cpp
    - In class game where array would freed on empty arrays only
- gameTesting.cpp
    - It may be helpful to run tests of passing a turn, resetting the turns, and then passing two turns to make sure those two are working together

Code Review Assignments:

Jonathan Mckee: Catch testing files,

Samuel Backes: gameTesting.cpp

William Doering: Tank Class

JD Pessoa: gameSetting class

Aidan Anderson: startTanks.cpp

Adesh Kumar: Tank Class

Wesely Adams: tankTesting.cpp

Levi Butts: classGame.cpp and classGame.h

Gwyn Kardelis: classTank.cpp

Dennis Kovarik: gameTesting.cpp, tankTesting.cpp, authored "Sprint 1 Code Review"