C++ Labyrinth Game

Marjaana Lahti TPCS2 2020

What the program does

- **Creates a grid** from a file with a string of 0s and 1s
  - 0s appear as corridors, and 1s appear as walls
- The player is able to move **up**, **down**, **left** or **right**
- Movement outside the grid and into wall tiles is **restricted**

Computation

- It all starts with a **string**... "11\n01\n11\n11"
- **\n** determines rows, and **((0s + 1s)/rows)** determines columns (assumes number of columns is even and equal in every row)
- Using a nested for loop, a 2D grid with that amount of rows and columns is painted using QPainter and QBrush
    - Inside this loop, **getLocation()** gets the unique location value of a given coordinate (row, col). It then uses this information to access the char at that location to determine if tile is a wall or not
- Another method called addToVec() uses similar nested for loop logic to add walls to a vector of locations. This is used to prevent players from accessing wall tiles.

(row*row_total + col) = location

0,0	0,1	0,2
1,0	1,1	1,2
2,0	2,1	2,2

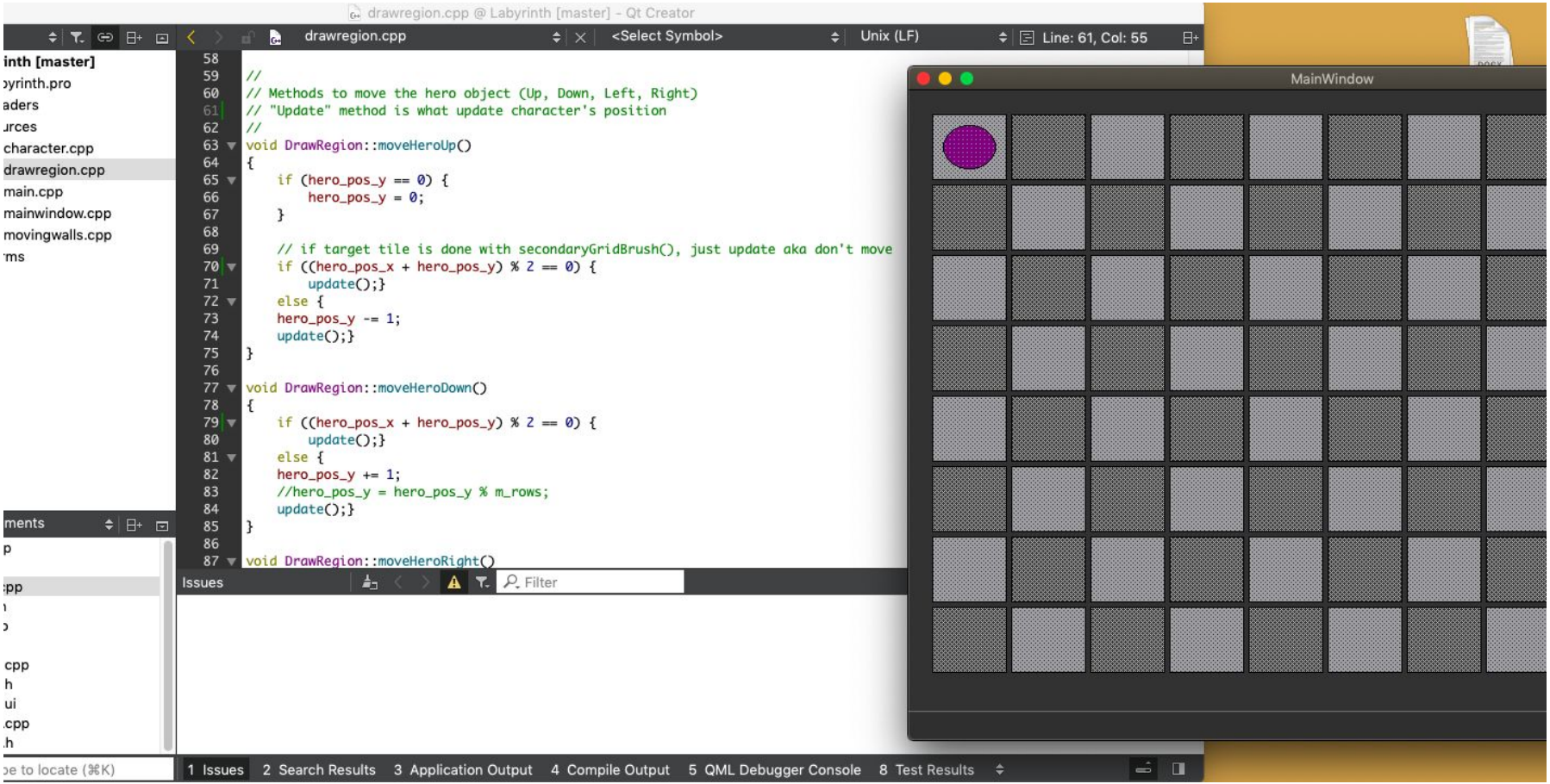
0	1	2
3	4 addToVec()	5
6	7	8

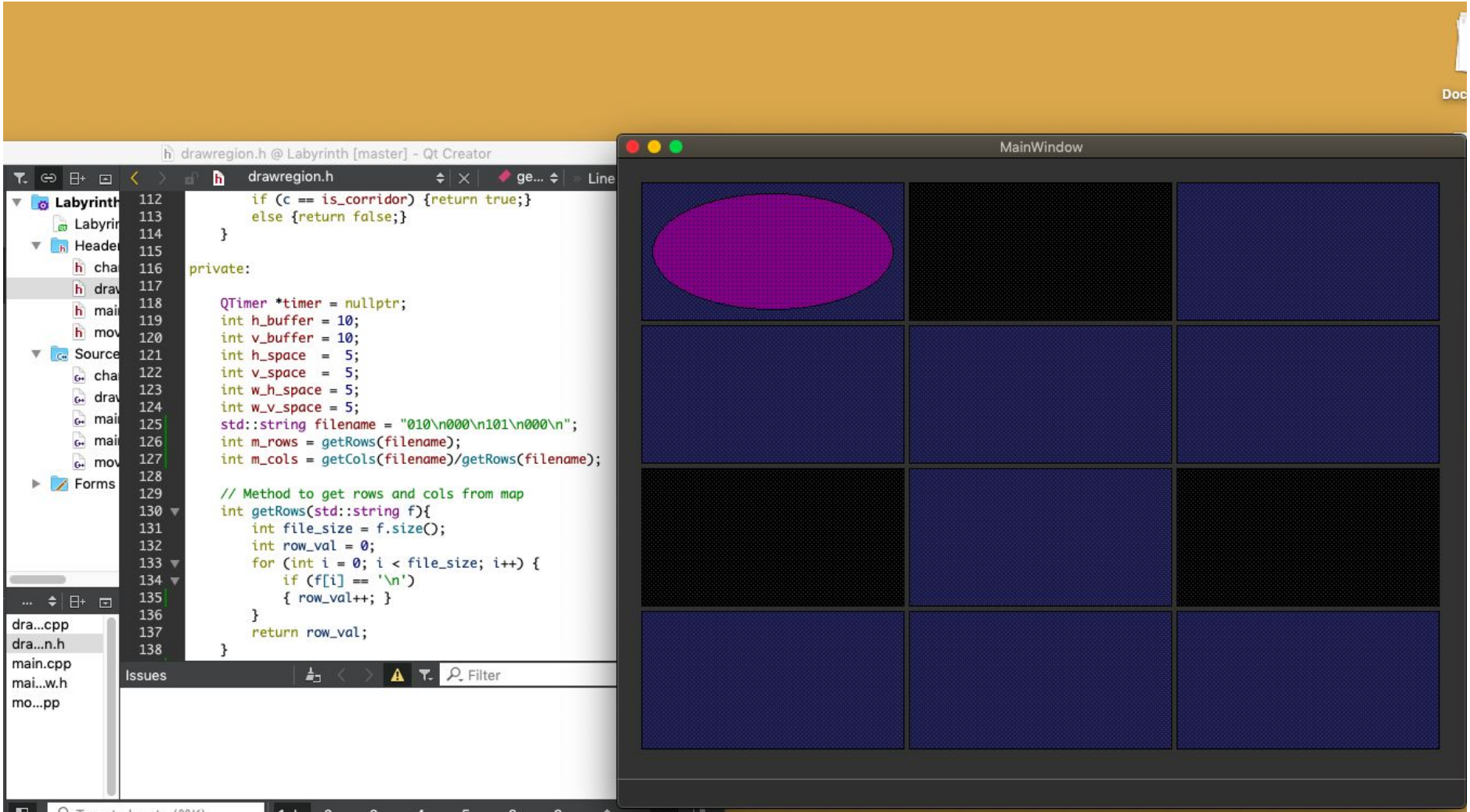
Visualisation

- The User Interface shows a grid. Corridors and Walls are distinguished with their colour.
- A shape is shown in the top left corner of the grid. This is moved around by the player using key event controls. Before entering a tile, the move method checks if the target tile is a wall (**isWall() bool**), or if the current tile is a grid edge. If either of these two is true, the **shape is not moved**. If they aren't, the shape moves in the chosen direction.

Development cycle

- IDEA: A computer game based on the board game Labyrinth. A player is placed in a maze, can move around in corridors but not access walls. The player can move one wall per turn to the edge of either the same row or column. The aim is to collect treasure.
- DESIGN: Determining the realistic **scope** of the project. Starting from making a grid and learning to make a user interface in Qt.
- PROGRAM: First making a pre-determined **grid** and a **player** that can be moved. Creating simple code to **prevent access to walls**.
- EVALUATE: Determining what can be added. Reprogramming **source for tiles** and **tile access.**


drawregion.cpp @ Labyrinth [master] - Qt Creator
drawregion.cpp
<Select Symbol>
Unix (LF)
Line: 61, Col: 55
inth [master]
yrinth.pro
aders
urces
character.cpp
drawregion.cpp
main.cpp
mainwindow.cpp
movingwalls.cpp
ms
58
59 //
60 // Methods to move the hero object (Up, Down, Left, Right)
61 // "Update" method is what update character's position
62 //
63 void DrawRegion::moveHeroUp()
64 {
65 if (hero_pos_y == 0) {
66 hero_pos_y = 0;
67 }
68
69 // if target tile is done with secondaryGridBrush(), just update aka don't move
70 if ((hero_pos_x + hero_pos_y) % 2 == 0) {
71 update();}
72 else {
73 hero_pos_y -= 1;
74 update();}
75 }
76
77 void DrawRegion::moveHeroDown()
78 {
79 if ((hero_pos_x + hero_pos_y) % 2 == 0) {
80 update();}
81 else {
82 hero_pos_y += 1;
83 //hero_pos_y = hero_pos_y % m_rows;
84 update();}
85 }
86
87 void DrawRegion::moveHeroRight()
ments
p
pp
cpp
h
ui
.cpp
.h
Issues
Filter
e to locate (⌘K)
1 Issues
2 Search Results
3 Application Output
4 Compile Output
5 QML Debugger Console
8 Test Results
MainWindow


drawregion.h @ Labyrinth [master] - Qt Creator
drawregion.h
MainWindow
Issues
Filter
Type to locate (⌘K)

Next steps...

- Fixing file read
- Adding mechanism to move walls. This will require modifying the original string, the location value of the tile, and hence the vector that stores the location values.
- ***But ideally*** would write a **class which manages Tiles**.
  - String elements would be stored in a vector object
  - Universal method to get x and y coordinates
  - Bool that determines if tile can be walked on
- Make code more **general and organised**

Thank you for your attention!