**Manual:**

Game allows a user to run through a randomly generated maze.

A user may fly upward if they wish to using arrow keys to see the full maze otherwise “WASD” allow the user to walk through it without cheats.

A user may turn on no clip if they get stuck or want to see other parts of the maze by pressing the “N” key, pressing it again toggles it back off.

Game has a camera controlled through mouse movement, but the full, up down pitch control is limited to stop users from peeping over walls.

**Code Build:** (Note maze creation is a library we found required porting to vmath but other than that almost untouched)

To build code clone from GitHub. From PowerShell terminal run “./ProjectRefresh.bat” (may need to copy some include and lib folders from another project”. Run “mingw32-make main”. Run ./bin/main.exe and then enjoy.

**Objects:**

A picture containing graphical user interface

Description automatically generatedA picture containing text

Description automatically generatedBedrock is used to mark the floor of the maze, while Bricks are used to denote walls.

A picture containing radar chart

Description automatically generatedA picture containing chart

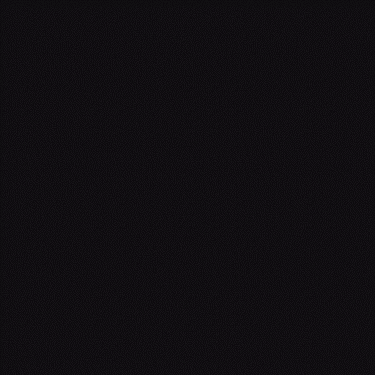
Description automatically generatedDiamond is used to denote the end of the maze and stone is used to denote the beginning.

**A picture containing text

Description automatically generatedA group of boats on a body of water at night

Description automatically generated with low confidenceA picture containing dark

Description automatically generatedA picture containing dark

Description automatically generatedA black rectangle with a black background

Description automatically generated with low confidence Skybox pictures**

**Maze Generation** (Note maze creation is a library we found required porting to vmath but other than that almost untouched)

Maze is generated using recursive backtracking.

1. Choose a starting point in the maze.
2. Randomly choose a wall at that point and carve a passage through to the adjacent cell, but only if the adjacent cell has not been visited yet. This becomes the new current cell.
3. If all adjacent cells have been visited, back up to the last cell that has uncarved walls and repeat.
4. The algorithm ends when the process has backed all the way up to the starting point.

**Parts/Fitting it together:**

To start with the maze sits in a 3d skybox that is a picture of the end from Minecraft, the maze is then constructed starting with the floor made of blocks approximately 2X2 in size and then after the walls are generated using the same blocks just scaled to be thinner and smaller. Each box has a bounding box that interacts with the cameras bounding box to prevent players from running through the walls and floors. The floors are given a different texture from the walls to help distinguish them and also the start and end are different than each other and the floor as well to allow users to know when finished.