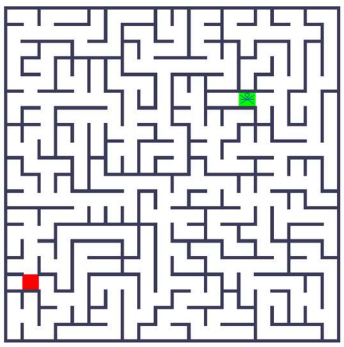


24-780 B—Engineering Computation

Assigned: Mon. Oct. 30, 2023
Due: Tues. Nov. 7, 2023, 11:59pm

Problem Set 7: Ortho Maze

This assignment has two main tasks, which can be done in any order since you do not need to finish one before the other can be tested. We will discuss the algorithms in the lecture and then you can implement them within the context of the given code. I am giving you a LOT of code that already works so spend your precious time on the requirements themselves.

Task 0 (Make it yours)

Before you begin, take a few minutes to become familiar with how the maze software works, getting the feel for how the user sees the application. Note that you can load two of the mazes just by typing "a" or "b" in the filename prompt, and you can omit file extension (.map). See how zoom works and how you can change the start and end locations. Navigate the maze and try to reach the end. Also take time to see how the maze is modeled, although you should only be interacting with the maze through the class functions without getting down into the nitty-gritty.

Also, feel free to make changes to my code when you want to make an improvement (in clarity and/or process), but I recommend against making too many changes that do not get you closer to completing the assignment. You certainly will want to change the window name, for example, but don't spend too much time changing the little spider 😊.

Btw, the easiest way to make this code your own is to create a new project and just copy my files into it. This will allow you to provide a good name for the project. You can start on this even before lecture.

Task 1 (Create Random Maze)

You are to add a functionality to the maze application so that the user can create a new maze from scratch. Mainly, the user provides the size and the Maze object itself generates a random maze.

In the startup code, we have identified that you need to flesh out the function MazeManager::createNewMaze() which should take less than 20 minutes. The bulk of your effort should be in implementing Maze::createRandom(), also identified in the code. To keep it simple, you are to use the Aldous-Broder algorithm which we will discuss in lecture (not very complicated).

Task 2 (Find Shortest Path)

The ability to find the shortest path will be implemented in Entity class. This will allow for multiple entities in some unrealized future 😊. The function Entity::findShortestPath() is identified in the code for you to implement. The auxiliary arrays (visited and parent) have been included as member variables. We will discuss the BFS algorithm in the lecture.

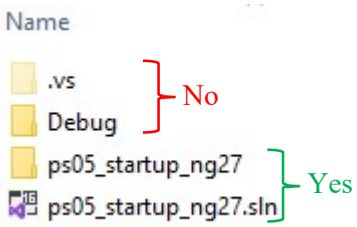
Deliverables

Several files (zipped together):

MazeManager.h and .cpp
Maze.h and .cpp
Entity.h and .cpp

Note that you don't need to include all the "library files" (StringPlus, DrawingUtilNG, GraphicFont, etc.) unless you made some changes. Upload the zip file to the class Canvas page before the deadline (Tuesday, Nov. 7, 11:59pm).

Alternatively, if you are using Visual Studio, it may be easier to submit your entire solution rather than a collection of files. To do this, create a *zip file* of the whole project (the .sln file and the associated folder), being careful NOT to include the hidden folder called ".vs". This folder is used only to manage the IDE and is typically huge (100MB). Erasing or omitting it will just force Visual Studio to rebuild it when needed. The Debug folder should be kept out of the zip file too to avoid including executable files that some firewalls may disallow. *The name of the project should include your AndrewID*



Learning Objectives

Use of classes and objects in C++.

Use of "large" dynamically allocated data models.

Increasing understanding of graphical data representation.

Implementing algorithms developed by others.