mms
Micromouse Simulator

Mack Ward

2015

Table of Contents

1 Introduction

1.1 Summary

This is an application for developing and simulating Micromouse algorithms. For information on the Micromouse competition, see the Micromouse Wikipedia page.

1.2 Motivation

Back when I was in IEEE, as I'm sure is the case now, we procrastinated on mostly all of our projects. This included Micromouse; if it was more than a week before the competition, you could safely assume our robot wasn't built yet. This proved to be problematic for the programmers on the project. They weren't able to start writing code until way too late (read "the night before the competition").

In an attempt to solve the problem of not being able to write and test code before the robot was built, we built this simulator.

The simulator is also meant to be a teaching tool - to help those who may be interested, but not familiar, with programming to take their first few baby steps.

1.3 Contributing Projects

This section isn't finished yet. For now, see the README.md in src/lib.

2 Installation

Regardless of what platform you're planning on simulating on, there are a few dependencies that you must install before you can start. They are as follows:

2.0.1 freeglut

Allows us to create a window and draw things within it.

Note: We're thinking about replacing this with GLFW, a more modern and well kept project that performs a very similar job.

2.0.2 GLEW

Helps with drawing things, and is required by many of the libraries that we used to implement the simulator.

2.1 Linux (Ubuntu)

Questions about Linux installation can be directed at `mward4@buffalo.edu`.

Simply open a terminal and run the following:

```
sudo apt-get install g++
sudo apt-get install freeglut3-dev
sudo apt-get install libglew-dev
```

2.2 Windows

Questions about Windows installation can be directed at `kt49@buffalo.edu` and `tomaszpi@buffalo.edu`.

Beware! You're entering into wild territory! A few people have gotten the simulator to work on Windows, but only with some finicking. If you intend to run the simulator on Windows, just be aware that it may take some extra effort to get things working properly.

2.2.1 Download freeglut and GLEW

See http://www.cs.uregina.ca/Links/class-info/390AN/WWW/Lab1/GLUT/windows.html Unzip them and place them wherever you desire.

2.2.2 Setting up Visual Studio Project

NOTE: These instructions were written for VS 2015

Once freeglut and GLEW are downloaded as per the above instruction you will want to create a project in Visual Studio. To do this automatically:

1. Open VS and select: File → New → Project From Existing Code
2. Select Visual C++ from the dropdown and hit Next

3. Enter the following options on the subsequent screen

   (a) Project file location - The path to **inside** the /src folder (ex. D:\Users\Tomasz\Documents \GitHub\mms\src)

   (b) Project name - The name you wish the project to have (ex. MMS)

   (c) Add files to the project from these folder - Leave Default

   (d) File types to add to the project - Leave Default

   (e) Show all files in Solution Explorer - Checked

   (f) Hit Next

4. Make sure 'Use Visual Studio' is checked and under 'Project Type' it says 'Console Application Project'. Nothing else should be checked

5. Hit Finish and Visual Studio will assemble the project for you

6. (**Encouraged**) Check 'Show all Files' under the 'Project' menu this will make the 'Solution Explorer' pane mirror the directory structure

7. In the 'Solution Explorer' right click the project file (should be the second from the top) and select 'Properties'.

8. Under C/C++ - General - Additional Include Directories: Add the include folders within the freeglut and GLEW folders you downloaded earlier, along with the GL folder within the include folder for GLEW, and the lib folder within /src. For me it looks like this:

   C:\Users\Kyle\Desktop\openGL\glew-1.11.0\include\GL

   C:\Users\Kyle\Documents\GitHub\mms\src\lib

   C:\Users\Kyle\Desktop\openGL\glew-1.11.0\include

   C:\Users\Kyle\Desktop\openGL\freeglut\include

   Click "Apply".

9. Under C/C++ → Output Files → Object File Name:

   Enter '$(IntDir)/%(RelativeDir)/'.

   Click "Apply".

10. Under Linker → General → Additional Library Directories:

    Add the lib folders within the freeglut and GLEW folders. For me it looks like this:

    C:\Users\Kyle\Desktop\openGL\glew-1.11.0\lib

    C:\Users\Kyle\Desktop\openGL\freeglut\lib

    Click "Apply".

11. Under Linker → Input → Additional Dependencies: Add the following:

    freeglut.lib

    glew32.lib

    Click "Apply".

12. Go to the folder where the glew files are held, glew-1.1x.0 (there are different versions: 1.11 and 1.13 are proven to work). Go the lib directory and the following files may or may not be there:

    glew32.lib

    glew32s.lib

    If they are not already in lib, go to Release\Win32 and copy them to the lib directory.

13. Copy the freeglut.dll and glew32.dll files to the src folder, they can be found here:

    ...\freeglut\bin

    ...\glew-1.1x.0\bin\Release\Win32

14. In the 'Solution Explorer' right click the project and select Add → Existing Item. Select mms\res\parameters.xml. This allows for easy access to the parameters file.

15. To build the project select Build → Build Solution or just run it and you will be prompted if you want to build the changed project

16. To run the program select Debug → Start Without Debugging

2.3 Mac

Questions about Mac installation can be directed at `dpclark4@buffalo.edu` and `srsiegar@buffalo.edu`.

You should also bug those guys about putting actual installation instructions here. 'Cause right now, we don't have any :(.

3 Building

3.1 Linux

TODO

3.2 Windows

TODO

3.3 Mac

TODO

4 Running

The simulation can be run by executing the binary in the "bin" directory, as in:

```
./<path-to-bin>/MMSim
```

For example, if you are in the "src" directory, enter the following to run the simulation:

```
../bin/MMSim
```

4.1 Keyboard Commands

During the simulation, a number of keyboard commands are available to the use. They are as follows (though I can't promise that this is 100% updated):

Key	Effect	Discrete Mode Only
p	toggle (p)ause/resume	X
f	make the mouse go (f)aster	X
s	make the mouse go (s)lower	X
l	cycle through the available (l)ayouts	
r	toggle zoomed map (r)otation	
i	zoom (i)n on the zoomed map	
o	zoom (o)ut on the zoomed map	
t	toggle wall (t)ruth visibility	
t	toggle tile (c)olors	
t	toggle wall fo(g)	
x	toggle tile te(x)t	
d	toggle tile (d)istance values	
w	toggle (w)ireframe mode	
q	(q)uit	

4.2 Runtime Parameters

TODO

5 Writing Your Own Algorithms

5.1 Discrete and Continuous Modes

TODO

5.2 IMouseInterface

TODO

5.3 Getting Started

All user-defined mouse algorithms should be placed within the mouse directory, preferably within their own directories. While it is possible to define a single function for the algorithm, a good strategy is to define a class that contains the appropriate functions for executing the algorithm. This will make the algorithm easier to understand, easier to debug, and much more portable. AlgorithmTemplate ".h" and ".cpp" files have been provided to help those who may not know the syntax for defining classes in C++. You can simply copy these, change the approprite identifiers, and provide definitions for the given solve function, as well as you own helper functions.

Additionally, the algorithm class (that you define) should implement the IMouseAlgorithm interface. This ensures that the algorithm is compatible with the simulation utilities already defined.

Lastly, in order to actually execute a user-defined algorithm, the "src/mouse/MouseAlgorithms.cpp" must be modified to include your code.

5.4 Input Buttons

TODO

6 Implementation

6.1 Modern OpenGL

TODO

6.2 Collision Detection

TODO

6.3 Logging

TODO

6.4 Parameters and State

TODO