Lab 2 Basics

Contents

Midi Maze

Lab 2 is inspired by what is arguable the very first multiplayer first-person shooter game, Midi Maze, from 1987.

https://www.youtube.com/watch?v=_0g6b-22tNs&t=97s

We won't get first person perspective, our simulation will be top down. It will have smiley face icons and projectiles as best as can be done with ACSII graphics. We will use this simulation for labs 2-4. Lab 2 shows one object at a time and moves it until it hits a wall. Future labs will involve more simultaneous objects and possibly true multiplayer competition.

Your simulation will deal with motion control and keeping track of where an object is and moving it. Starting data comes from various data files that will be provided. Output will either be text with numbers or graphics. Lab 2 is heavily simplified. The lab 2 reference code is about 350 lines (heavy on whitespace, light on comments).

Dates

Prototypes demo by Friday, September 8th at 11:58

Just because the cutoff is midnight does not imply that there will be any office hours at any time near then. Prototypes must be demonstrated to the instructor or any grader in their office hours prior to that cutoff.

Lab Early Saturday September 9th by 11:58 PM

Lab On time Monday September 11th by 11:58 PM

Lab Late cutoff Tuesday September 12th at 11:58 PM

Documents

- Basics – this document
- Simulation & Data
- Input
- Output

No Global Variables! Code Must Compile!

Global variables is a -10 penalty. Errors or warnings in compilation means no credit for the lab, though you might get 2 points if your prototypes are there and compile cleanly.

Zip files and makefiles

Your makefile rules that build your zip file need to self-test the lab (build the lab) or the lab is **late** regardless of when it was turned in.

Consider this fragment of a makefile that should be familiar to you:

```
#self-test what I want graded
make -C install -s lab2
```

The above lines are an example

Files mandated by Lab 2

Your lab 2 **must** be split over various C code files and header files.

C code files:

- lab2.c owns main and only the highest level functions
- you need a file to deal with the bit in the code
- you need a file to deal with simulation and motion
- you need a file that does most of the output
- you need a file that does input
- [this list is not yet certified complete]

Header files for function declarations:

Use **make -r headers** each time you add a function to any of your C code files. Remake the headers if you change the signature of any function as well.

Every C code file you write must include its own header so that the declarations in the header are validated against the definitions in the C code file. In other words, lab2.c must #include lab2.h.

If you get weird errors about your functions, run **make -r headers** again and then rebuild the lab code.

Also check for a missing #include if you get errors about functions that you call in other files.

You will need to #include some of those function declaration header files in more places than the code that generates them. It is likely that lab2.c will need to include the header file for the output functions. The bits header file will be needed in a few places, but bits should not need any outside header files.

Submission

Effectively: Same as lab 1. Your zip file needs to contain README_LAB2, **all** the code to be graded, and a makefile sufficient to build the lab. Do not include any .o files or the lab2 executable. Any file you edited by hand (other than test data) probably needs to be included. Aside from header files, files generated by the compiler should not be included.

All files that you edit by hand must have your name in them.

README_LAB2 text:

THIS IS THE README FILE FOR LAB 2.

BY SUBMITTING THIS FILE TO CARMEN, I CERTIFY THAT I HAVE PERFORMED ALL OF THE WORK TO DETERMINE THE ANSWERS FOUND WITHIN THIS FILE MYSELF WITH NO ASSISTANCE FROM ANY PERSON OTHER THAN THE INSTRUCTOR OF THIS COURSE OR ONE OF OUR UNDERGRADUATE GRADERS.

The readme should contain your **name**, the number of **hours** you worked on the lab, and any comments you want to add. Of particular interest are what was hard or easy about the lab or places where programming reinforced what we went over in class.