

Assignment 1

CMSC 740, Spring 2002

Due: 12:30pm Thursday, February 28, 2002

This assignment has two pages. Make sure you have both of them.

This assignment involves using OpenGL programming API with the GLUT toolkit. The goal is to implement a simple maze game. The goal of the game is for the player to move a game piece in the shape of a face (we will refer to it as Gump) from its start position at the lower left of the screen to its ending position at the top right of the window. Your program should allow the player to use the keyboard to control Gump's movement.

You can find details about using OpenGL and GLUT on various platforms, links to download OpenGL and GLUT on your PC or UNIX workstation, and Assignment 1 startup source code and data at <http://www.cs.umd.edu/class/spring2002/cmsc740/assg1/>. Given the various possible platforms and their configurations as well as our limited resources we would not be able to help you install OpenGL and GLUT on your personal systems. Please make sure that your final program works on machines in one of the following labs: Microsoft/WAM Lab (AVW 3452) or CSD Junkfood Lab (AVW 3457). Before submitting your assignment you should create a README file that gives details about which location your program works. You

Copy files from the assignment web-page. Compile and run them after linking with OpenGL and GLUT. A window should pop-up with a blue background and a tiny yellow point in the window that is actually a 4×4 square of pixels. The function that determines where a point is drawn on the screen is `glVertex2i()` and is being called from `display_func()` in file `ogl.c`.

(a) Open a window should of size 512×512 , with blue background. The walls of the maze are specified in a file `mazedata.txt`, in the following format:

$$\begin{array}{cccc} n & & & \\ x_{11} & y_{11} & x_{12} & y_{12} \\ x_{21} & y_{21} & x_{22} & y_{22} \\ & \vdots & & \\ x_{n1} & y_{n1} & x_{n2} & y_{n2} \end{array}$$

Here n is the number of line segments denoting the n walls of the maze, and (x_{i1}, y_{i1}) and (x_{i2}, y_{i2}) represent the start and end points of the i^{th} wall. Write the code to read the above data file and display the walls of the maze as yellow lines that are three pixels thick. (5)

(b) Use the image-reading code at the assignment web-page to read the image of a house `house.dt`. If you prefer your own image readers, you may use GIF, TIFF, JPG formats of this image available from this assignment's web page. Display the image near the exit of the maze at the top right corner of the window.

Also write the code to display Gump's face at the lower left corner of the maze and inside the maze. You should create it using OpenGL triangles or polygons. (5)

(c) Write code to enable the player to use the keyboard's four arrow keys to control Gump's movements in the maze as follows:

- If Gump is headed in the same direction as the arrow key that is pressed, Gump is advanced by 50 pixels in that direction.
- If Gump is headed in a direction different from the direction of the arrow key that is pressed, rotate Gump so that it is now pointing in the correct direction. Obviously, you should design Gump so that it is not rotationally symmetric.
- Gump cannot move across the walls of the maze. Implement a simple collision detection scheme. You may assume that the walls of the maze can be only horizontal and vertical, but not at arbitrary angles. Designing your Gump with a rectangular outline will similarly be helpful for this part of the assignment.

(7)

(d) Implement Gump to have animated expressions as follows:

- If Gump reaches the end-point at the top right corner of the window, draw it as being happy.
- If Gump hits a wall it starts crying. Draw the tears on Gump gradually falling across its face using the idle function. Gump should stop weeping when it can start moving normally.
- When Gump is moving about normally, draw it in the third state that is neither happy nor sad.

(3)

Special Note: Refer to the demo version of our sample implementation of this game to resolve any ambiguities that might have remained in the above description.