

Assignment 3

CMSC 740, Spring 2002

Due: 12:30pm Thursday, April 18, 2002

This assignment involves extending your code for Assignment 2 to deal with a virtual walk-through of the AV Williams building.

(a) Read the geometry and image data files. The geometry description of the fourth floor of AV Williams bldg is in the data file `avw.dat`. The data is in the following ASCII format:

- The first line of the data contains one number that indicates how many “layers” are in the file. In our file there are 9 layers, so the number is 9. You can think of each layer as containing similar data (all doors are in one layer, all windows in another, and so on). Then from the second line onwards, the data for each layer appears as discussed next.
- The data for each layer begins with a line indicating the type of that layer. This line begins with character “#”. For instance, the second line of the file is: `#ARWALL`
- The line after the type of the layer contains two numbers, the first one indicates how many quads are in this layer and the second one indicates the number of triangles (in our data file, the second number is always 0, since we have no triangles in this file).
- Then comes the actual geometry data. If the layer contains n quads, then the following n lines will describe the n quads. Each line contains the following information:

$$x_1 y_1 z_1 x_2 y_2 z_2 x_3 y_3 z_3 x_4 y_4 z_4 r g b$$

Note that there is a color for each quad. Also note that there is a layer named `ARTEXT` that contains extrusions of the appropriate room numbers. You are not required to draw it, but you might anyway to help you find your way during the walkthrough.

We are providing four texture map files for elevators, ceiling, door, and floor named `elv.jpg`, `ceiling.jpg`, `door.jpg`, and `ground.jpg`. The assignment web-page also has the PPM versions of these files. If you need any other kind of format, please contact the TA. Alternatively, if you convert them to some other format, make sure that you submit your project with the appropriate format image files. Write the code to read these texture image files. **(3 + 2)**

(b) Use your Assignment 2 code to display the data you have read in part (a). Write code to implement texture mapping for the doors, the ceiling, the floor, and the elevator. For ceiling and ground you should tile the texture and for the doors you should stretch the texture to cover the entire door. For now, you can map the door texture to the windows. There is no texture for the walls. **(5)**

(c) Implement walking around by mouse movements. Start with user at $(54600, 22000, 200)$ and looking along the direction $(0, 1, 0)$. This is close to the fourth floor elevators. Implement walking and looking around as follows:

- When the user presses the left mouse button and drags (moves the mouse with the left mouse button pressed) vertically, translate in the direction given by the current look-at direction and proportional to the amount of vertical mouse movement. Moving up causes forward motion, moving down causes backward motion.
- When the user presses the left mouse button and drags horizontally, rotate the view direction in a plane parallel to the floor and proportional to the amount of horizontal mouse movement. Mouse movement to horizontal right causes the view direction to rotate right (world rotates left) and mouse movement to horizontal left causes the view direction to rotate left.
- When the mouse movement (as discussed above) has both horizontal and vertical components, perform rotation first followed by translation in the new look-at direction.
- When the user presses the right mouse button and drags vertically up, implement translation towards (and above) the ceiling. Similarly when the user drags the mouse vertically down, implement translation towards the ground.
- Implement a reset button on key 'r' to move the user back to the original view point and view direction.
- Incorporate the collision detection code from Assignment 2 to prevent walking through walls. You should allow walking through the doors.

(5)

(d) Assume that your implementation of OpenGL only allows 8 lights. The data file has 199 lights. They are given as quads under the layer LAMPS. Use the nearest visible 8 lamps as OpenGL light sources and display the rest as appropriately colored rectangles. Obviously the selection of the nearest 8 lamps will change dynamically as the user moves about. To find out the 8 nearest visible lights, render the lights in unique colors in the back-buffer, read back the back-buffer, scan it to determine which lights are visible and their depths. This will tell you which lights deserve to be lighted. Now clear the back-buffer and re-render the entire scene (again in the back-buffer) with these 8 lights turned on and then swap the front and back buffers. Use the `glTexEnv(GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE, param)` command to allow textured polygons to be lighted.

(5)