COMP 356: Computer Graphics
Homework 4: Object-order renderinge

PROBLEM 1. *Implement an interactive 3-D maze-solving application. My solution, which you should use as a guide, is posted on the website. For full credit on this assignment:*

- *You must use OpenGL's lighting and material functions for all surfaces. Choose something that looks nice.*
- *You must use transforms appropriately for drawing all walls. In particular, you should only have a couple of functions for drawing a couple of basic shapes in some fixed position; everything in your world is drawn by invoking these functions after appropriately modifying the model-view transform. For example, my solution has two functions: one draws a square of side-length 2 in the xy-plane centered at the origin; the other draws a rectangular solid of length 1, width .25, and height 1 along the x-axis centered at the origin.*
- *For setting the camera transform, you must use appropriate change-of-frame transformations. You may not use the utility functions in the GLU library (e.g., `gluLookAt()`). You may (and are encouraged to) use functions like `glTranslate` and `glRotate`.*
- *You must implement some sort of more-or-less reasonable collision detection. It is difficult to make this perfect, but at the very least the player should not be able to walk straight through a wall that is parallel to the viewing plane (corners are harder to get right).*
- *You must implement a "jump" function that is toggled by the space key; a "jump" gives a bird's-eye view of (much of) the maze. While jumping, you may use functions like `gluLookAt` for setting the camera transform. All movement must be disabled when jumping.*
- *As the player navigates the maze, "breadcrumbs" must be dropped on visited cells—i.e., some sort of marker to let the player know that s/he has been to that cell already.*
- *You must print the player's position and heading.*

Although not required, it would be great to implement some nicer game-play than my sample application; for example:

- *Jumping should be animated.*
- *Something should happen when the player makes it to the end cell.*
- *There should be some visual feedback to represent collisions.*

1. Grading criteria

I will be grading your submission based on the following criteria:

- Completeness. All questions are answered.
- Correctness. All required algorithms are implemented correctly.
- Efficiency. There are no obvious inefficiencies such as unnecessary loops, repeated calculations, etc.
- Code structure. Functions defined as appropriate, code organized into files appropriately, appropriate memory management.
- Style. Code well-written, formatted, and documented.