CISS380: Computer Graphics
Project: FPS

The goal is to write an FPS. That's it.

Demo: Ryan Frappier (start at 9:12)

In more details, the following are the minimum requirements.

1. The World
   - The world is a maze. In other words, you and the other objects reside in a maze.
   - Your maze is built using an n-by-n grid where n can be specified by the user of the program. Each cell is a room. You can go from one room to an adjacent room if there is no wall between the rooms. Instead of a wall that is missing, you can also have an opening in the wall.
   - You must show that you use textures. (Not necessarily for everything.) You must texture a picture of yourself somewhere.
   - [OPTIONAL] You must show some physics. (For instance when things fall.)
   - You do not need to have sound.
2. The View
   - There are 2 views: The view of the first person in the game and an outsider (bird's eye) view.
   - You can switch between the views using a key press.
   - When the outsider view is chosen, the top is not drawn so that one can view the whole n-by-n grid. (You can choose to not have a "top" or ceiling in your game, i.e., you can see the sky in the maze.) You can move the view around while still playing the game. Of course the keys for moving the view is different from the keys controlling your gameplay.
   - The maze should be randomly generated. There are many random maze generation algorithms – you are free to choose any.
3. The Gameplay
   - This is a FPS game. So you must be able to shoot at something/someone - Duh.
   - You move around in the world.
   - The things (robots for instance) can also move around in the world.
   - Basic collision detection must be implemented. For instance you cannot walk into a wall.
   - You cannot move into other objects in the world.
   - You can fire at the objects/robots.
   - The robots can also shoot at you.
   - Your motion need not be too realistics.

4. Models
   - For simple 3D models, you must write your own. For complex models (teapots, etc.), you can use freeglut. For simplicity you can simply use cubes for robots.
   - You must use at least one of the following: client-side vertex arrays, display lists, vertex buffer objects (VBO is optional).
5. Optimizations. The following important optimizations must be present:
   - While in the first person shooter view: Your program must draw the objects (walls, robots, weapons, bullets, etc.) that are reasonably visible from the player. As much as possible, objects not visible should not be drawn. (Example: In a 2D maze, if you are at location (r, c) (row index r and column index c), then if you can facing East and there's an opening on the East side, then South, then West, the room in when you make the East-South-West motion is not visible from (r, c) when facing East:

```
--+----+----+
   EYE->    |
--+----+    +----+----+ EYE facing east -- cannot see walls behind.
     X         Y |      EYE cannot see room X
--+----+----+    +----+ EYE might be able to see East wall of Y
           |  Z |      EYE might be able to see East and South
           +----+      wall of Z but not west wall of Z
```

   - While in the outsider view: You must draw the objects that are reasonably visible from the player. For objects (walls, robots, weapons, etc.) which are not visible from the player, you should do this:
     - For walls - do not draw them. However you should draw lines on the floor of the maze to show connections between the centers of two rooms which are reachable from each other.
     - For anything else that moves - draw a simplified representation of object such as a cube.
   - Your game should run smoothly without freezes or halts. The speed should be reasonable.
   - You know my rules: if your program crashes, you get 0.