Abstract
The purpose of this project is to produce a 3D game that depends on graph theory, procedural world generation, and artificial intelligence. The game should also be a good playing experience, as measured by surveys of users playing the beta. If time permits, I would also like to blog about my learning and implementation process and produce a tutorial of sorts for other technical people looking to get into game development.

High-level Game Overview
My vision for the game is a maze-based world that the player will have to explore in first person, all the while contending with the demands and map alterations of an unfriendly, powerful being (the game AI) in control of the situation. Communication with the AI will occur via a text interface (allowing me to avoid the need for voice actors).

The player starts the game in an unfamiliar grey room with 1-4 doors leading out to hallways. The hallways lead to identical rooms – these hallways and doors constitute the maze, and will be designed with a scientific simplicity and blandness in mind. As the game progresses, the player will have to decide whether or not to comply with the increasingly strange demands issued by a being that seems to be in control of this place. If the player complies, the game will become noticeably easier, with the being providing helpful hints and shortcuts. If the player does not comply, the being will become increasingly hostile, thwarting the player at every turn: locking the player in a room for a few minutes, spinning the maze to confuse orientation, moving hallways in the maze if the player is on the right path, and lengthening hallways.

The game should leave the player with the sense that they must choose between their own agency and winning the game – they can comply in order to win, with the downside that the being will mock them as weak for doing so (and the ending will be less satisfying).

Technical Challenges
• The maze. The maze will present multiple challenges in the areas of graph theory and procedural map generation.
  o I plan on representing the maze as an undirected grid graph in either 2 or 3 dimensions (3 dimensions only if there is time to add this feature). This grid will be turned into a randomized tree using some spanning tree algorithm such as Kruskal’s or Prim’s
  o Multiple algorithms will be tested to see which creates the most convincing maze – a full spanning tree may end up feeling too predictable, for example.
  o I will then use Unity to generate the physical maze from this graph, generating hallways for edges and rooms for nodes.
  o I may add some variation to the maze by changing the appearance of each room / hallway based on some random weights on the nodes / edges. The complexity of this generation will depend on how much time I end up having to complete the project.
  o References and resources:
The being controlling the maze. The AI controller for the game will present numerous artificial intelligence challenges.

- Since I’ve never programmed a game AI before, I plan on using one of the more straightforward high-level architectures: A finite state machine. The actions of the player will cause the AI to transition states (each can be thought of as a degree of antipathy or sympathy towards the player), which will change the actions it will take and the probability of each action.
- The AI will have to track the player’s movements and compute shortest paths to the end of the maze in order to thwart or help him.
- The AI will have to be able to communicate with the player to relay demands and get responses. This may require some simple NLP work on the player’s text input and the ability to register the player’s precise movements (e.g. walking in circles). It would be nice to add some randomness to these demands to switch things up, but this will require more work than simple hard-coded rules (since the program has to understand the request and the conditions to fulfill it).
- The AI will have to be able to modify the maze on the fly in a clever way to help the player (e.g. adding an edge to allow the player a shorter path through the maze). This will require significant work on pathfinding algorithms.
- References and resources:
  - [http://www.gamedev.net/page/resources/_/technical/artificial-intelligence/a-practical-guide-to-building-a-complete-game-a-r784](http://www.gamedev.net/page/resources/_/technical/artificial-intelligence/a-practical-guide-to-building-a-complete-game-a-r784)

Tools and Languages

- C# and the Unity game development platform.

Deliverables

- The game produces a convincing, random maze on each playthrough
- The maze generation is high performance (it should not take more than a moment to start up)
- **If time permits** The maze is 3D (i.e. paths can go straight up or down)
- The controller AI communicates with the player correctly (i.e. understands correct vs. incorrect responses).
- The controller AI changes the map in a variety of ways without breaking it (spinning, moving edges, adding edges, etc.)
- The controller AI responds correctly (more hostile the more you don’t listen and vice versa).
- **If time permits** The controller AI generates a wide range of possible randomized requests for the player
- The game produces the desired effects in players (more simply: it’s a good game!)
  - This will be measured through beta testing on real people.
- **If time permits** A blog about the experience interspersed with tutorial for others following in my footsteps.
Development Timeline (approximate)

- 2/19/16 – Complete physical maze primitives. Have a scene that generates a random version of the maze that a player can guide the camera through and chooses a good start and end point.
- 3/4/16 – Complete simple AI communication mechanism (text to text, hardcoded). AI should be able to give and understand responses to commands. Most of the script should be done at this point.
- 3/18/16 – Complete and test the state machine and decision architecture for the AI.
- 4/1/16 – Add most of the actions the AI can take on the maze.
- 4/15/16 – Add any missing actions, polish the feel of the game. Get some beta testing done.
- 4/29/16 – Work on stretch goals, update game based on feedback. Compile blog if enough material has been produced.