**Stuff to do:**

* **display your own health bar (or one hit you're dead?)**
  + **Screen turns red as you're hit, health regenerates?**
* **COLLISIONS**
  + **Will his laser kill his own protection?**
  + **Bullet collisions on whatever the protective stuff will be**

**Change boss color on successive hits**

* **Diagonal rotations**
* **Regeneration of the protection layers (for level 2)**
* **Better meshes for rotation layers, boss, laser, bullets**
* **LIGHTING**
* ***2 Levels*/winning conditions**
* **sounds**

**Minor Stuff:**

* **Splash screen (intro, beat level, beat game)**
* **Score**
* Peer evals
* One-page report about who did what, tools used for graphics and sound, appraisal of project (what’s good and what needs more work)
* Filled-out rubric sheet
* **Video**
* **1 mesh&texture/person, advanced techniques (billboarding, lighting)**

**maybe don’t have the laser destroying the layers for this increment, but do that for the final. Because I don’t know how to get good collision on it.**

**Idea for player damage:** have a health value between 0 and 100, if the two phi and theta values are close (player is being hit), decrease the health. We’ll have a red quad in front of the camera and set its *transparency* according to the health value. Have health regenerate when not being hit? (might make it too easy)

Layer class that contains a vector of GameObjects (will be a different type later when ethan makes one).

* Rotation matrix and spinAmount
* Update() function that updates that
* addShell()
* removeShell() when hit (or perhaps just set it to inactive so that you can shoot through it?)
* Layer.draw() will run through Box.draw for all boxes in its vector (or whatever type that will be)

**Where will we check for collisions? In layer.update()?**

We'll then have an array of Layers

**THE SCALING FOR GAMEOBJECT MIGHT BE A LITTLE MESSED UP since I added dimensions for it**

**IDEA for rotating layers:**

* Have a bunch of really small flat squares whose position is at the origin, but then rotate them by small incremental amounts and then translate them out, so it will look kinda like an actual long thin wall rotating around it