**Ideas**

This article will house ideas that are not ready to be included in official documentation. It is to include any thought on how to fix an error, optimize code, improve design, alter set up of room, or even completely change our approach to the goal. Ideas should begin on separate pages and, if possible, be categorized.

If this document becomes too cluttered or congested, it may be replaced by a subfolder of Documents/ in which each Idea is a separate file.

Links to design images, proof of concept code, and any other resources are encouraged where applicable.

I don’t really know what to do with this space. First idea on next page.

**Representing 3D Lines by 2D Lines:**

**Forewarning:** I have little confidence that this will work. This is not the approach to finding 3d line intersections that I have seen online and when I asked my calculus teacher about it, he seemed skeptical. Nevertheless, I fail to see why it wouldn’t work, so I’ll share.

Projectile trajectories in two dimensions are represented by two functions of time- horizontal motion and vertical motion. I argue that this concept can be extended to 3D lines. The y and z coordinates can be expressed as functions of the x coordinate because, being a line, there is only one y and z value for every x value. You can think of each function as giving you the 2D coordinates of the line, when looking at it from a certain angle. (e.g.: The y function gives you the 2D graph of the line when looking directly at the XZ plane)

To write the y and z functions of a line, two points (x,y,z) are needed. Let’s call these (a,b,c) and (d,e,f). We will start with point-slope form (y – y1 = m (x – x1)) and then convert to slope-intercept form (y = mx + b). Focus first on the y function.

First, we need the slope of y(x). This is given by:

(e-b)/(d-a)

Then we plug in the coordinates of either point. This gives us:

y – b = ((e-b)/(d-a)) (x-a)

Reformatting results in:

y = ((e-b)(x-a)/(d-a)) + b

(Remember that a, b, d, and e are all constants, so it isn’t as ugly with actual numbers.

The same steps are used to find z(x):

z = ((f-c)(x-a)/(d-a)) + c

Once you have a y and z function for both lines. You can focus on their intersection. Because coordinates are only used once per line ((1,2,3) and (7,2,5) cannot fall on the same line), you need only to find one coordinate of the intersection. To keep things easy, solve for x. Set either your two y functions or your two z functions equal and find x. I have created a C++ method for finding intersections of 2D. It solves basic “x on both side” equations. I cannot link to it for some reason but it is “Line Intersection Locator.cpp” under Trial/Proof of Concept Code/.

Once you have the x coordinate of intersection, plug it into both y(x) and z(x) to find the 3 coordinates (x,y,z)

This may not be the proper place to write this, but most solutions to finding the intersection of 3D lines include vector math. [This](http://tutorial.math.lamar.edu/Classes/CalcIII/EqnsOfLines.aspx) web page had a very good explanation of vector math and how it relates to equations of lines.