3D Engine Notes, pointers, etc.

* Technically, there’s no such thing as 3d graphics
* Trying to represent a 3d shape on screen, but the screen is 2d…
* We’re generating a sequence of 2d shapes with illusion of 3d
* A 3d engine takes 3D geometrical data and converts it into 2D shapes
* For example…
* A cube might be a series of 8 vertices
* All of those points should be grouped into **triangles**
* Triangle is a simple primitive
* We should group them all together
* Its possible to represent any 2d shape using only triangles
* Many algorithms to draw triangles because they consist of straight lines
* Also to fill and shade them in using straight horizontal lines
* Projection
* By normalizing to -1 and +1, we ensure we never see any shape drawn outside of the boundary.
* However, the further away the object is, the larger our field of vision is.
* Narrow FOV is like zooming in, Wide FOV is like zooming out.
* So, we must use a scaling factor. i.e, theta
* One way to think about the factor is by drawing a right triangle right down our FoV
* Tan(theta/2) is relevant.
* However, if we take a point and increase theta, said object gets larger…
* Reducing theta makes it smaller.
* Can make artifacts and introduce displacing.
* **We want the inverse of this…. So - >** 1/(tan(theta/2))
* if we normalize x and y, z should be as well.
* Znear and Zfar
* Zfar / Zfar – Znear
* Will need to be offset later by discrepancy scale
* - Zfar\*Znear / Zfar – Znear
* Final transformed coordinate:
* X and Y should be divided by Z
* Relevant for multiplying vector by matrix etc.