**3D Graphics Engine - (C#)**

Libraries Used:

* math.NET
  + Used for matrix operations
* GtkSharp
  + Used for drawing on 2d canvas

The Math

**Part 1:** Finding & Projecting the Viewport:

Find a 2-dimensional subspace of ℝ3 that will be the viewport. Storing position in a 3-d vector, and rotation in a 2d vector (as tilt is not necessary). This position vector will be stored in the typical fashion, using the typical orthonormal bases of ℝ3. This rotation will be stored using common spherical coordinate system (ϕ = horizontal rotation; θ = vertical rotation; r = 1). Using these metrics to find the vector orthogonal to the viewport, we can find the 2-d subspace of ℝ3 that will represent the viewport:

vorthogonal = [sin(ϕ)cos(θ), sin(ϕ)sin(θ), cos(ϕ)].

The plane orthogonal to this vector is described by two orthonormal basis vectors; b1 and b2, where:

b1 = [sin(ϕ - π/2)cos(θ), sin(ϕ - π/2)sin(θ), cos(ϕ - π/2)].

b2 = vorthogonal × b1

From these basis vectors, we can create a matrix A:

With this matrix A, we can calculate the projection onto the viewport for any vector x in ℝ3

using the projection matrix P = A(ATA)-1AT.

We then need to transform this projected vector onto the x-y plane in ℝ2 to be able to draw the projection.

In order to do this, we need to consider the matrix B:

Inverting the matrix B gives us the matrix needed to transform our projected vector x into the x-y plane in ℝ3. We then simply multiply by the matrix T, lowering the vector into ℝ2:

T =

To get our final vector f:

-1

f = A(ATA)-1AT

x

**Part 2:** Drawing Edges: