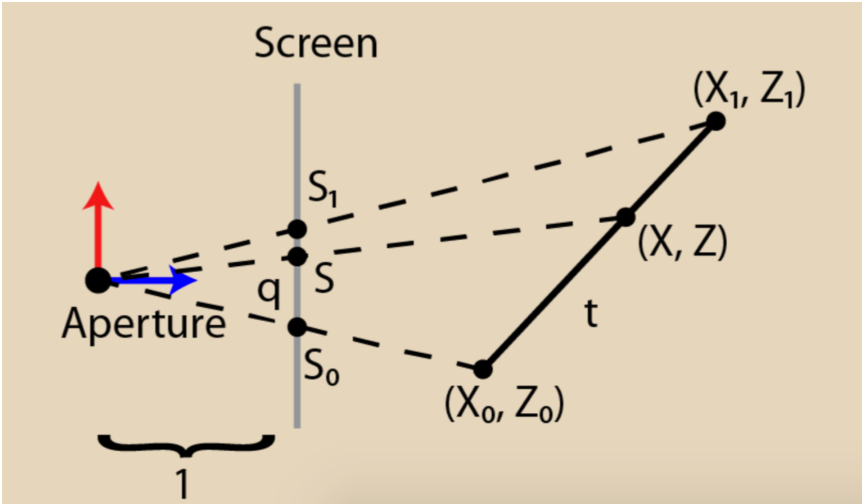
Q1:
(7 pts) Why does interpolating attributes of 3D triangle vertices in screen space lead to incorrect visualization? How can one correctly interpolate these vertex attributes?

A1:
Because the distances between its vertices is likely to change, so the proportion of line segment could also be changed, interpolation on these distorted vertices will provide inaccurate results.



Based on this picture from the lecture slides, we could compute X coordinate as X = X0 + t * (X1 - X0), Z coordinate as 1/Z = 1/Z0 * (1-q) + 1/Z1 * q or Z = Z0 + t * (Z1 - Z0), where S1 and S0 are the projecting points of (X1, Z1) and (X0, Z0) on the image screen, q is the proportion of SS0 to S0S1 and t = qZ0/(q*Z0 + (1-q) * Z1). In this way we solve the value of X and Z.

Q2:
(8 pts) Describe in detail the sequence of transformations that must occur in order to project a set of triangles given in 3D world space into the coordinate system of the pixelized screen.

A2:
1. We use back-face culling to ignore triangles that have a surface normal facing away from the camera, which we'll never be able to see.
2. Scale W by Z to perform perspective divide, where W is an extra coordinate to enable translation via matrix multiplication
3. Scale Z to [0, 1] range, between clipping planes(Perspective Frustum)
4. Scale XY to [-1, 1] range based on field of view and aspect ratio, where the lower-left corner of screen as [-1, -1] and the upper-right corner of screen as [1, 1].
5. Obtain the View Orientation Matrix and View Translation matrix in the form of

- $O = \begin{matrix} R_x & R_y & R_z & 0 \\ U_x & U_y & U_z & 0 \\ F_x & F_y & F_z & 0 \\ 0 & 0 & 0 & 1 \end{matrix}$
- $T = \begin{matrix} 1 & 0 & 0 & -eye_x \\ 0 & 1 & 0 & -eye_y \\ 0 & 0 & 1 & -eye_z \\ 0 & 0 & 0 & 1 \end{matrix}$

Then View matrix = O * T.

6. Concatenate two matrices with geometry to view the scene, which also converts our geometry from world space to normalized device coordinate.

Scene = Projection matrix * View matrix * geometry;

7. Final step is to convert NDC value to Pixel value using the following equatuion:

    Pixel.x = (NDC.x + 1) * Width / 2;

    Pixel.y = (1 - NDC.y) * Height / 2;

Then we’re all set.