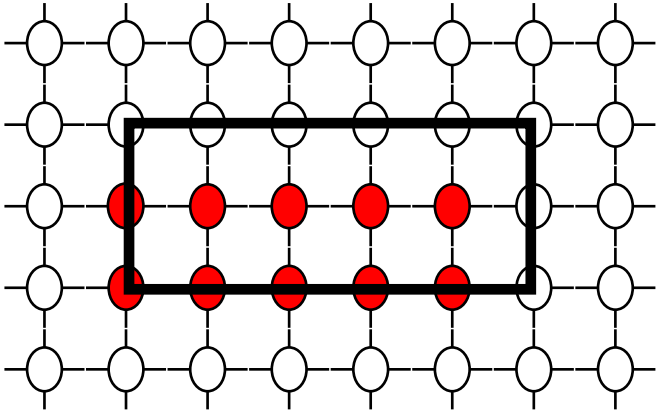
# Lecture 19: Rendering with a Z-buffer

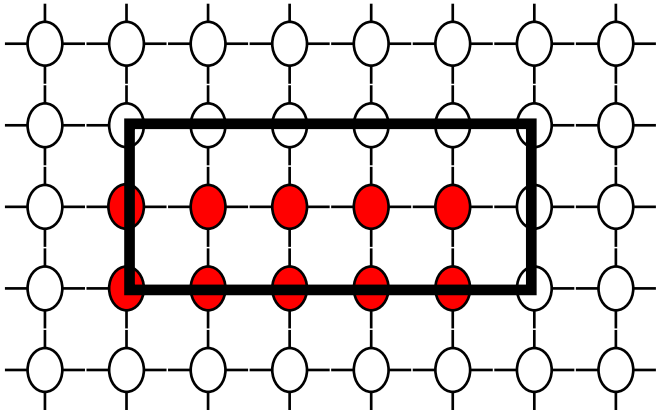
October 30, 2018

# Render this Rectangle

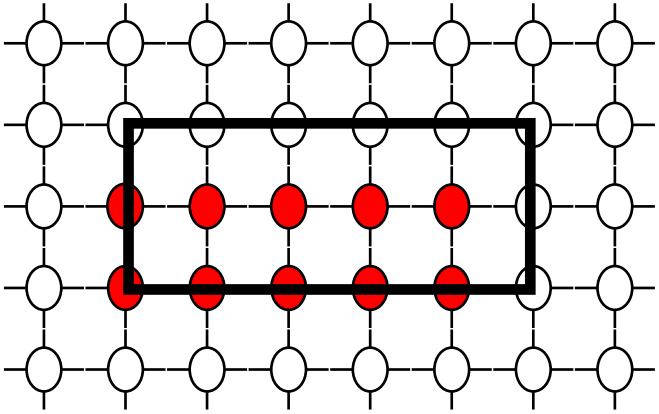
**Z-Buffer**



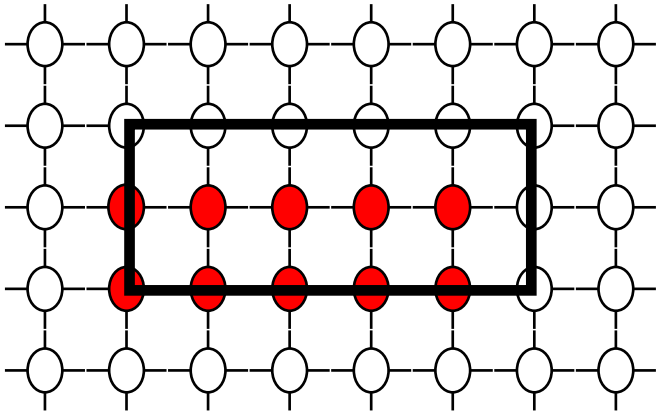
Red Plane



Green Plane



Blue Plane



*What colors and when.*

# Using a Z-Buffer

- Record depth at every vertex
- For every pixel in polygon (previous lecture)
  - Interpolate to get depth at specific pixel.
  - Is depth less then currently recorded?
    - Yes: Record in Z-Buffer and paint pixel
    - No: Move along, nothing to do here
- “Paint” is shorthand for compute the surface illumination for that position on the polygon.

# About depth: the z-value

- Z-buffering based upon pseudo-depth is key to modern polygon rendering.
- Depth already revealed in SageMath notebook on the Canonical View Volume.
- Here let us briefly dive into the calculation of pseudo-depth using essentially that example.

# SageMath Notebook

- Emphasize the z coordinate of transform

localhost:8888/notebooks/CS410%20Fall%202018/lectures/cs410lec1

cs410lec19n01

jupyter cs410lec19n01 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted SageMath 8.3

Run Code

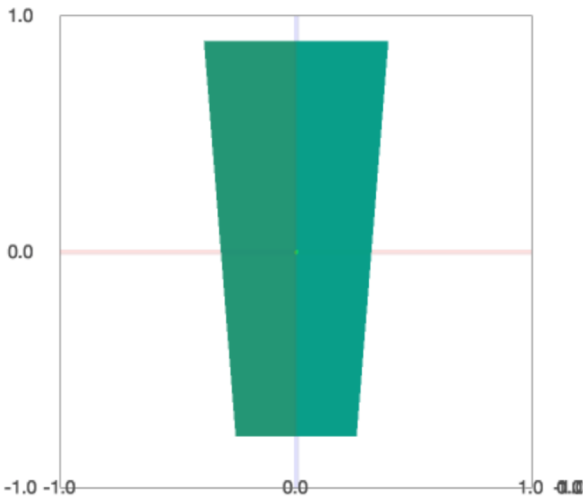
$$\begin{vmatrix} -\frac{umax+umin}{umax-umin} \\ -\frac{vmax+vmin}{vmax-vmin} \\ \frac{\frac{2\,farnear}{far-near}-\frac{(far+near)z}{far-near}}{z} \\ 1 \end{vmatrix} = \begin{vmatrix} \frac{2\,near}{umax-umin} & 0 & -\frac{umax+umin}{umax-umin} & 0 \\ 0 & \frac{2\,near}{vmax-vmin} & -\frac{vmax+vmin}{vmax-vmin} & 0 \\ 0 & 0 & -\frac{far+near}{far-near} & \frac{2\,farnear}{far-near} \\ 0 & 0 & 1 & 0 \end{vmatrix} \begin{vmatrix} 0 \\ 0 \\ z \\ 1 \end{vmatrix}$$

$$P_{cc} = \begin{vmatrix} -\frac{umax+umin}{umax-umin} \\ -\frac{vmax+vmin}{vmax-vmin} \\ \frac{\frac{2\,farnear}{far-near}-\frac{(far+near)z}{far-near}}{z} \\ 1 \end{vmatrix} \quad \text{and the z term only} \quad pz = \frac{\frac{2\,farnear}{far-near}-\frac{(far+near)z}{far-near}}{z}$$

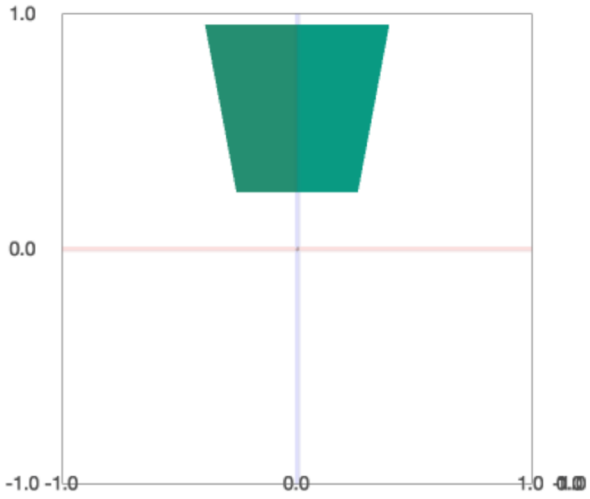
```
In [68]: if (case != 'sym') :
```

# First the Symptom

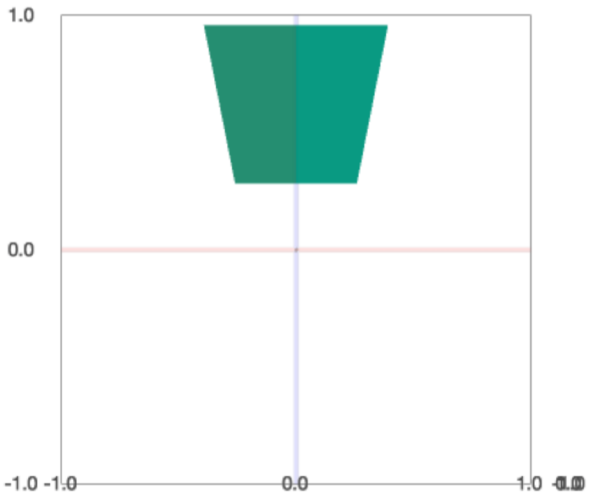
Near = -25
Far = -75




Near = -25
Far = -750




Near = -25
Far = -7500




Remember, the house lies between z of 30 and 54 in world coordinates.

Even pushing the far clipping plane 2 orders of magnitude further back from -75 still results in the house occupying most of the pseudo-depth range between 0 and 1.

# Back to the Math

- Camera at origin no world cam. rotation

$$\begin{vmatrix} -\frac{(umax+umin)z}{umax-umin} \\ -\frac{(vmax+vmin)z}{vmax-vmin} \\ \frac{2\,farnear}{far-near} - \frac{(far+near)z}{far-near} \\ z \end{vmatrix} = \begin{vmatrix} \frac{2\,near}{umax-umin} & 0 & -\frac{umax+umin}{umax-umin} & 0 \\ 0 & \frac{2\,near}{vmax-vmin} & -\frac{vmax+vmin}{vmax-vmin} & 0 \\ 0 & 0 & -\frac{far+near}{far-near} & \frac{2\,farnear}{far-near} \\ 0 & 0 & 1 & 0 \end{vmatrix} \begin{vmatrix} 0 \\ 0 \\ z \\ 1 \end{vmatrix}$$

$$P_{cc} = \begin{vmatrix} -\frac{umax+umin}{umax-umin} \\ -\frac{vmax+vmin}{vmax-vmin} \\ \frac{\frac{2\,farnear}{far-near} - \frac{(far+near)z}{far-near}}{z} \\ 1 \end{vmatrix}$$

and the z term only $pz = \frac{\frac{2\,farnear}{far-near} - \frac{(far+near)z}{far-near}}{z}$

Pseudo-depth

# $pz$ At near and far

- Equation: $$pz = \frac{2 * far * near}{(far - near) * z} - \frac{(far + near)}{(far - near)}$$

**Let $z$ equal $near$**

$$pz = \frac{2 * far * near}{(far - near) * near} - \frac{(far + near)}{(far - near)}$$

$$pz = \frac{2 * far - far - near}{(far - near)}$$

$$pz = \frac{far - near}{(far - near)}$$

$$pz = 1$$

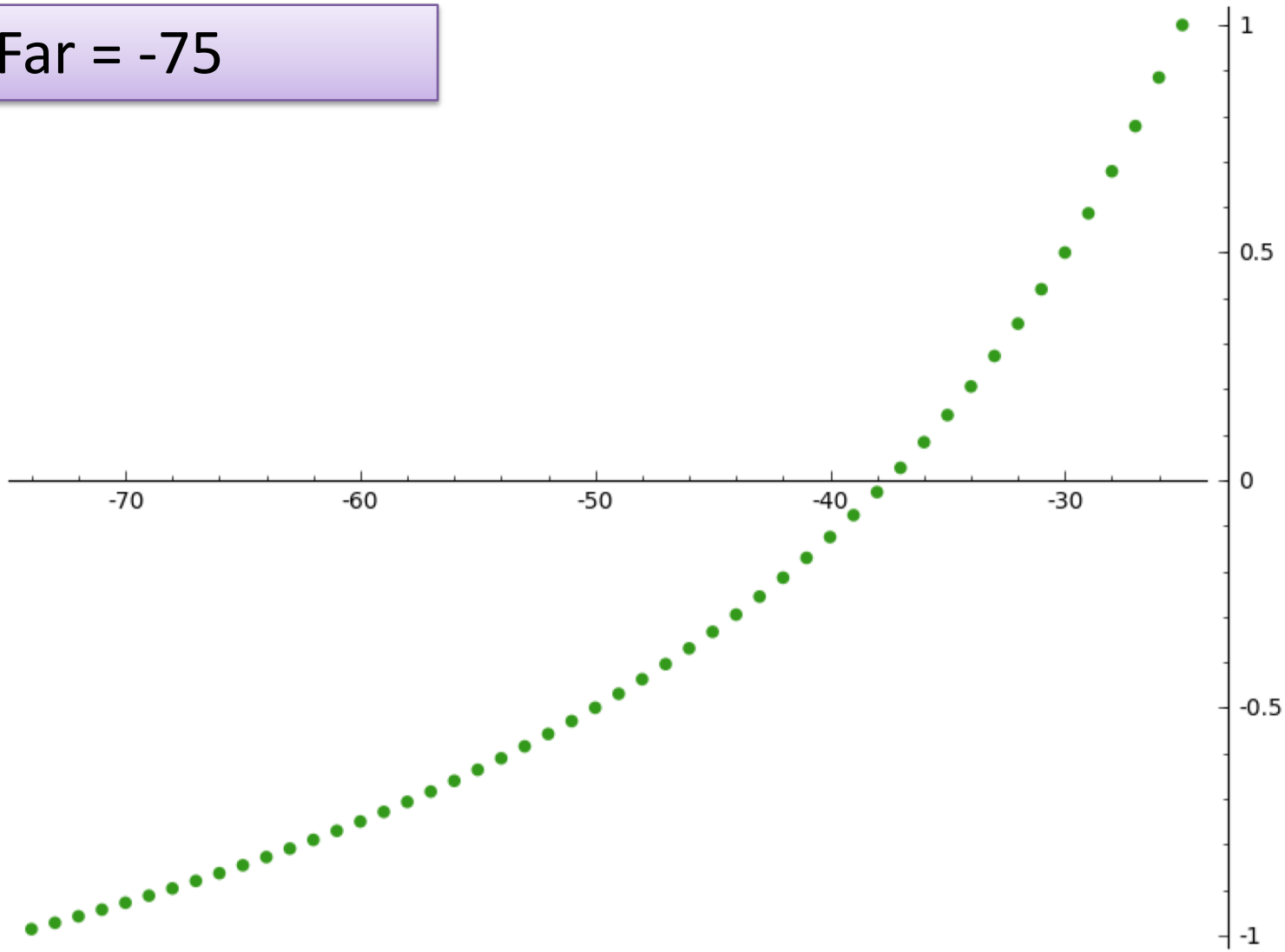
Similarly …

**Let $z$ equal $far$**

$$pz = -1$$

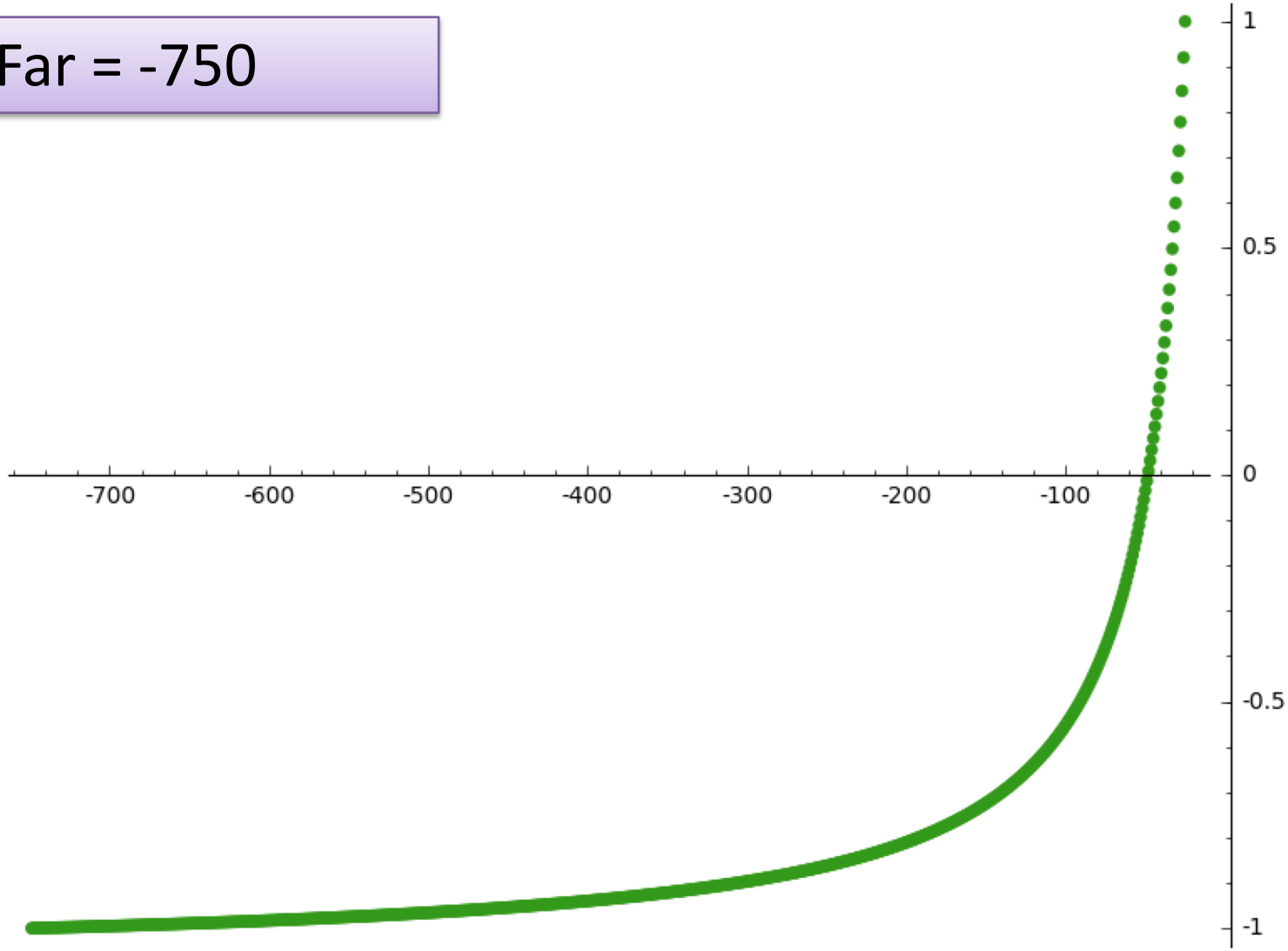
# Plot actual Depth to Pseudo-depth

Far = -75



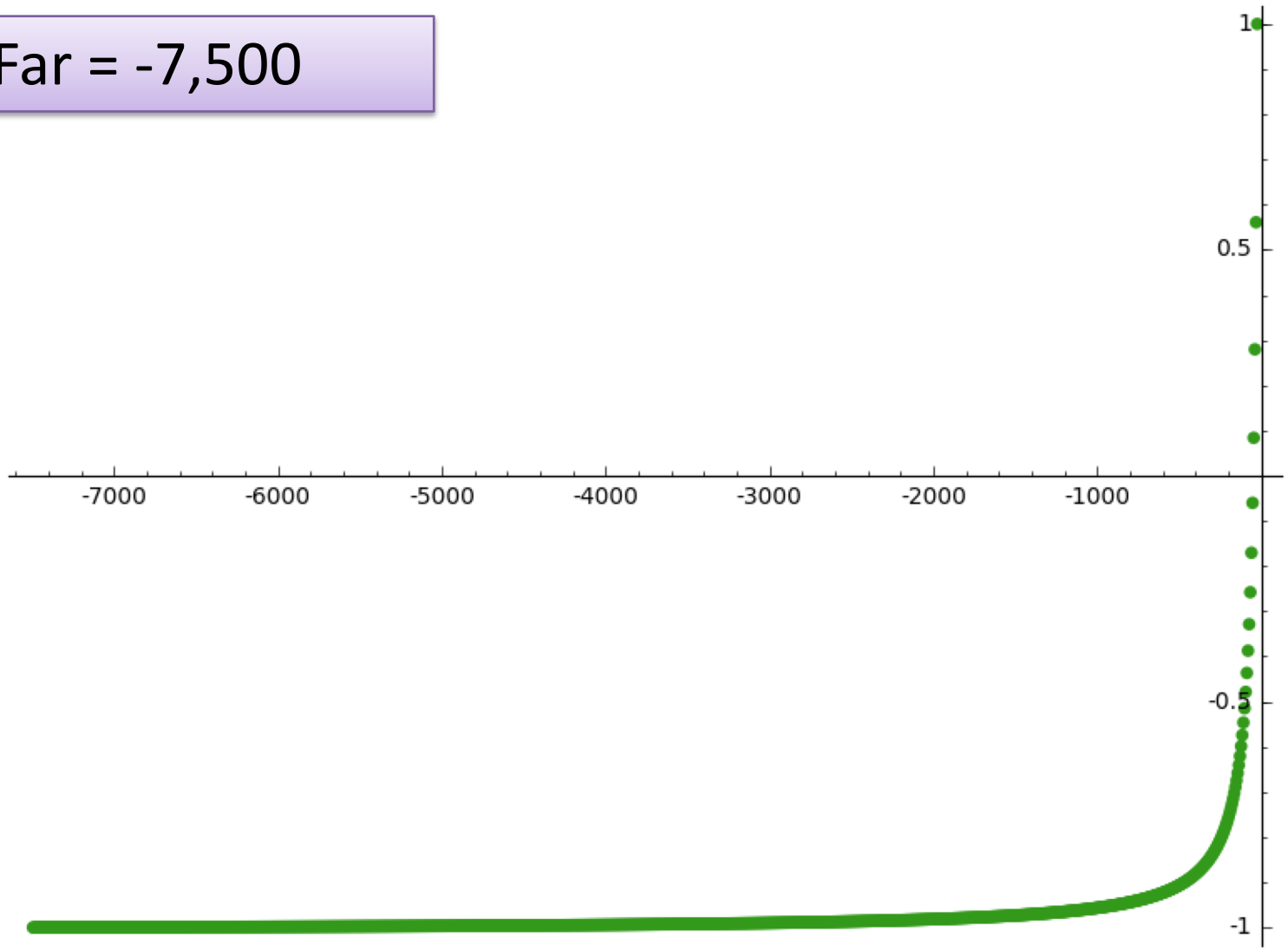
# Plot actual Depth to Pseudo-depth

Far = -750

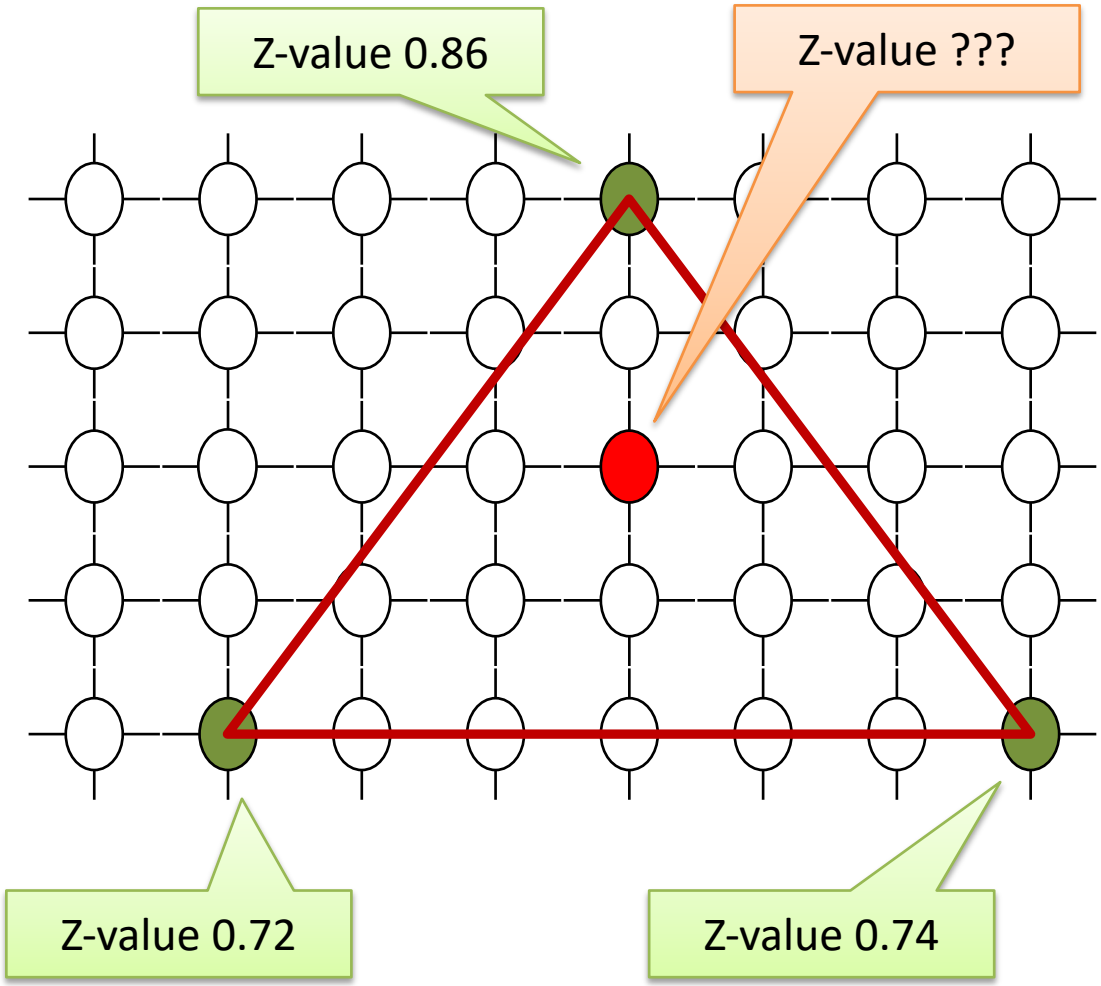


# Plot actual Depth to Pseudo-depth

Far = -7,500



# Interpolate Z-value




There are various ways to interpolate in order to arrive at an estimated z-value for a interior point on any given triangle.

Common is to first interpolate up the sides and then to interpolate across.

# Z-Buffer Summary

- A Z-buffer is an array of doubles
- Size of the frame buffer / image
- Initialized to -1.0, i.e. far clipping plane
- Now consider a specific triangle
- For each pixel to be filled
  - Interpolate pixels z-value
  - Test if larger then what is in the Z-buffer
  - If yes then “paint” that pixel for that triangle

# What if you Want Depth?

- Mapping may be inverted.

$$pz = \frac{2 * far * near}{(far - near) * z} - \frac{(far + near)}{(far - near)}$$

$$z = \frac{2 * far * near}{(far - near) * pz + far + near}$$

```
In [1]: var('y','near','far','z')
        eq = y == (2*far*near)/((far-near)*z) - (far + near)/(far - near)

In [2]: eq

Out[2]: y == -(far + near)/(far - near) + 2*far*near/((far - near)*z)

In [3]: solve(eq,z)

Out[3]: [z == 2*far*near/((far - near)*y + far + near)]
```

There are worse things then checking your work in a symbolic math package.