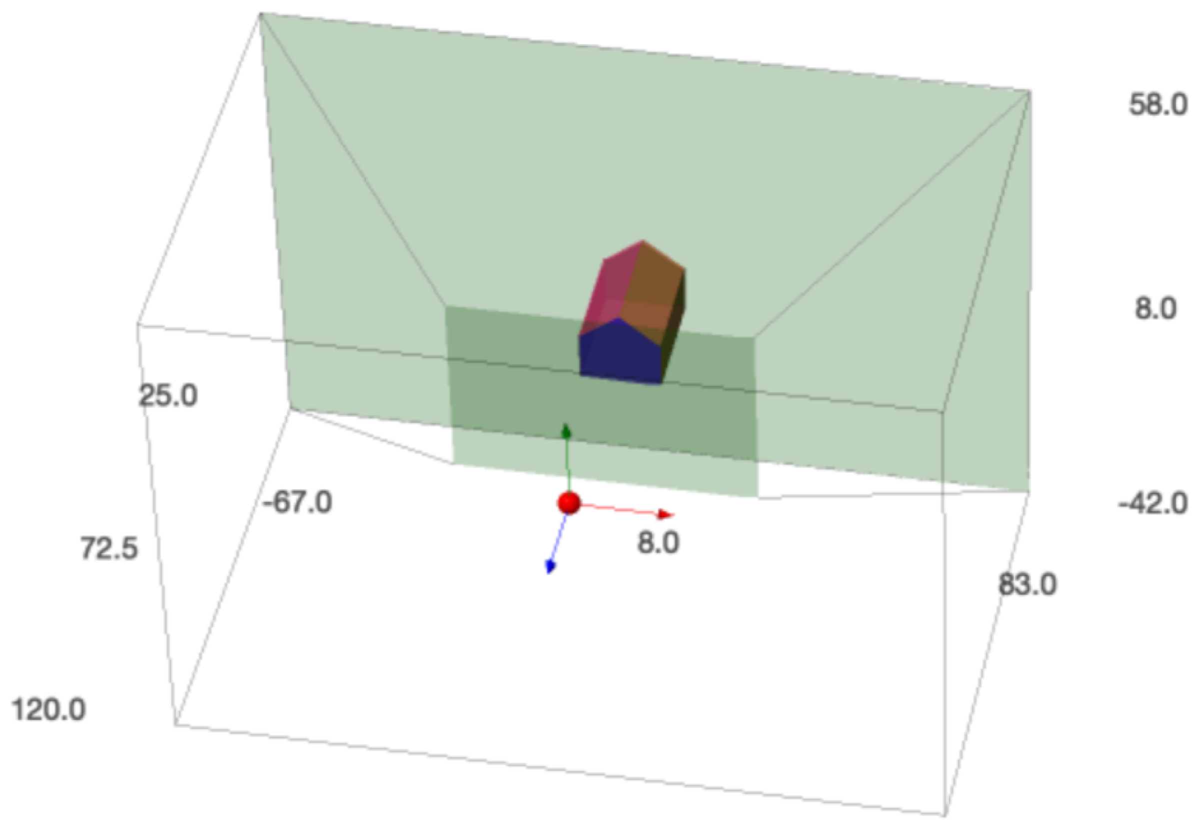
Lecture 05:
Camera Placement

September 10, 2019

PowerPoint Then SageMath

- Begin with overview and motivation.
- Then dive into SageMath Notebook.



Begin: Pinhole Camera Model

en.wikipedia.org/wiki/Pinhole_camera_model

Pinhole camera model - Wikipedia

Not logged in Talk Contributions Create account Log in

WIKIPEDIA
The Free Encyclopedia

Article Talk Read Edit View history Search Wikipedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages

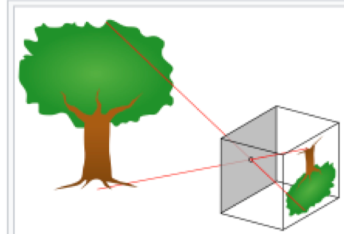
Pinhole camera model

From Wikipedia, the free encyclopedia

For broader coverage of this topic, see Epipolar geometry.

This article includes a list of references, but **its sources remain unclear** because it has **insufficient inline citations**. Please help to improve this article by introducing more precise citations. *(February 2008) (Learn how and when to remove this template message)*

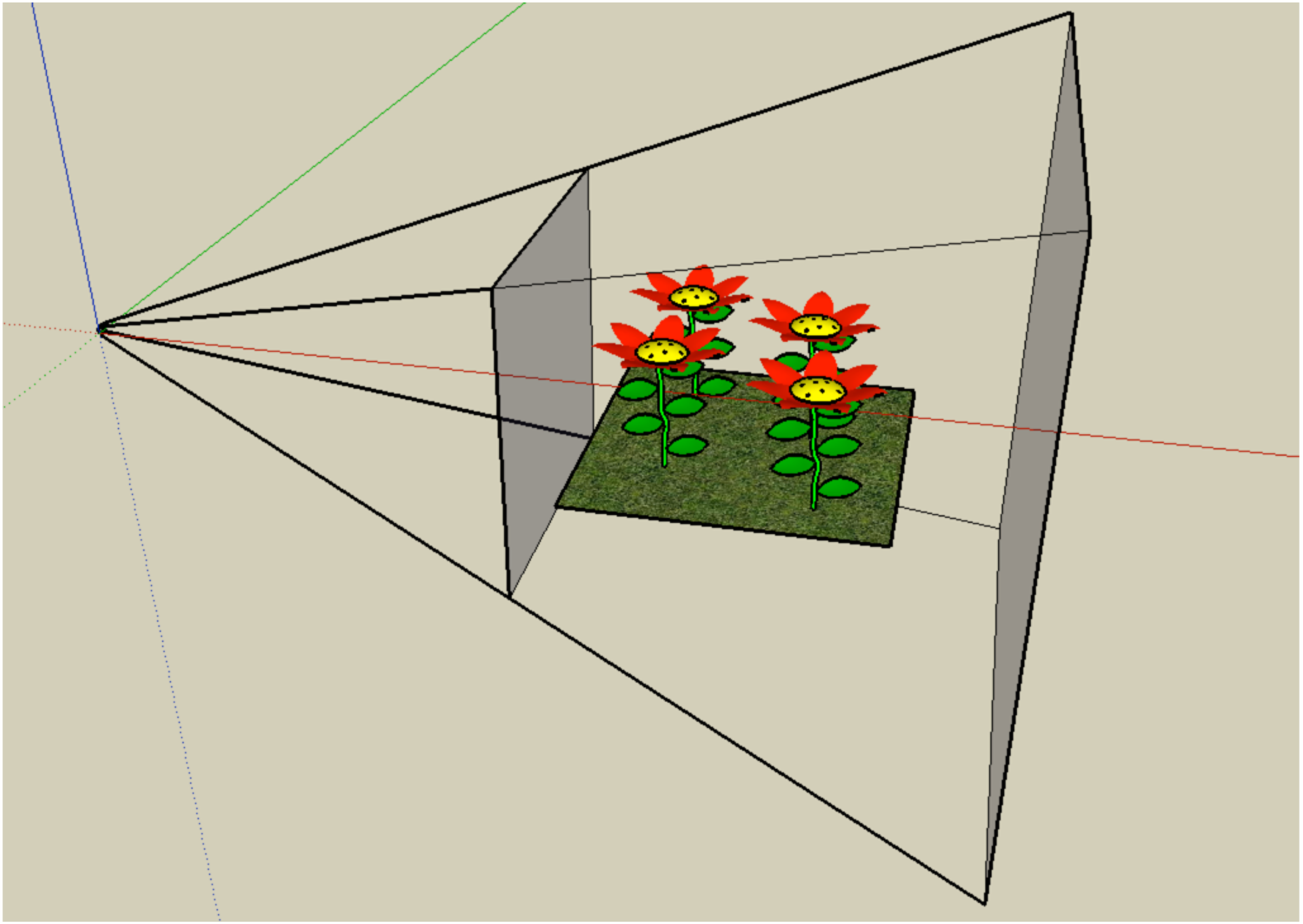
The **pinhole camera model** describes the mathematical relationship between the coordinates of a point in three-dimensional space and its projection onto the image plane of an *ideal* pinhole camera, where the camera aperture is described as a point and no lenses are used to focus light. The model does not include, for example, geometric distortions or blurring of unfocused objects caused by lenses and finite sized apertures. It also does not take into account that most practical cameras have only discrete image coordinates. This means that the pinhole camera model can only be used as a first order approximation of the mapping from a 3D scene to a 2D image. Its validity depends on the quality of the camera and, in general, decreases from the center of the image to the edges as lens distortion effects increase.



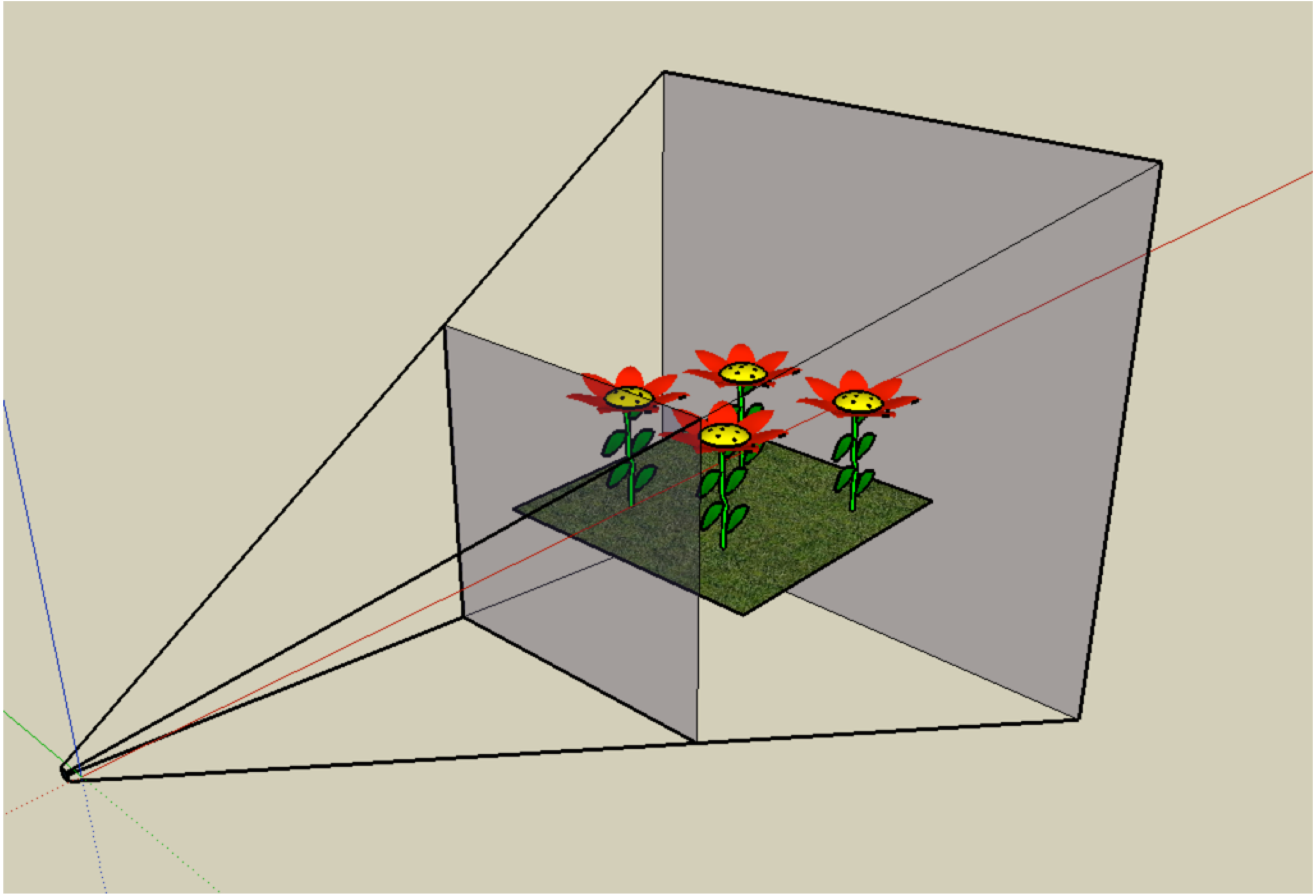
A diagram of a pinhole camera.

Some of the effects that the pinhole camera model does not take into account can be compensated, for example by applying suitable coordinate transformations on the image coordinates; other effects are sufficiently small to be neglected if a high quality camera is used. This means that the pinhole camera model often can be used as a reasonable description of how a camera depicts a 3D scene, for example in computer vision and computer graphics.

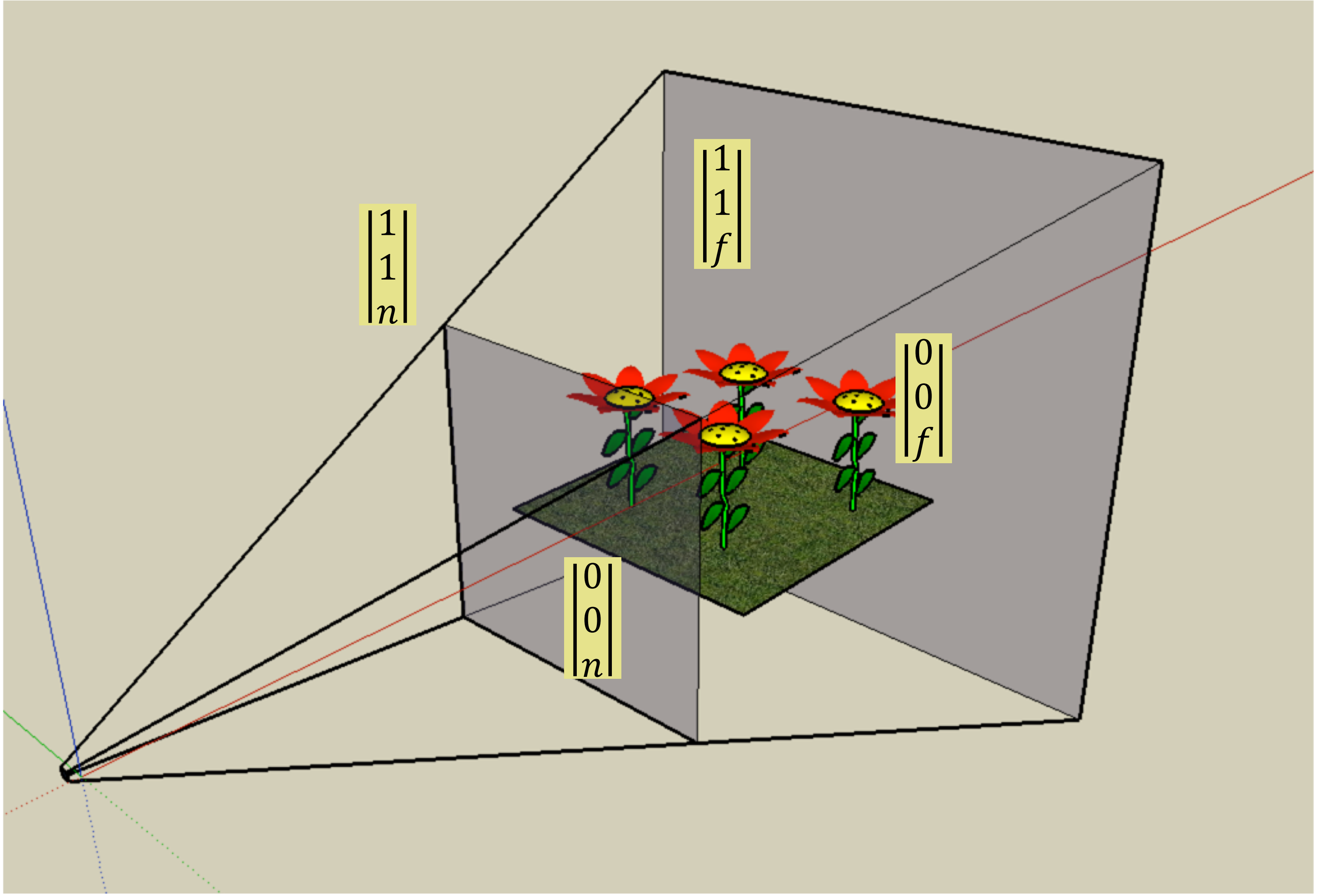
Visualize View Volume (View 1)



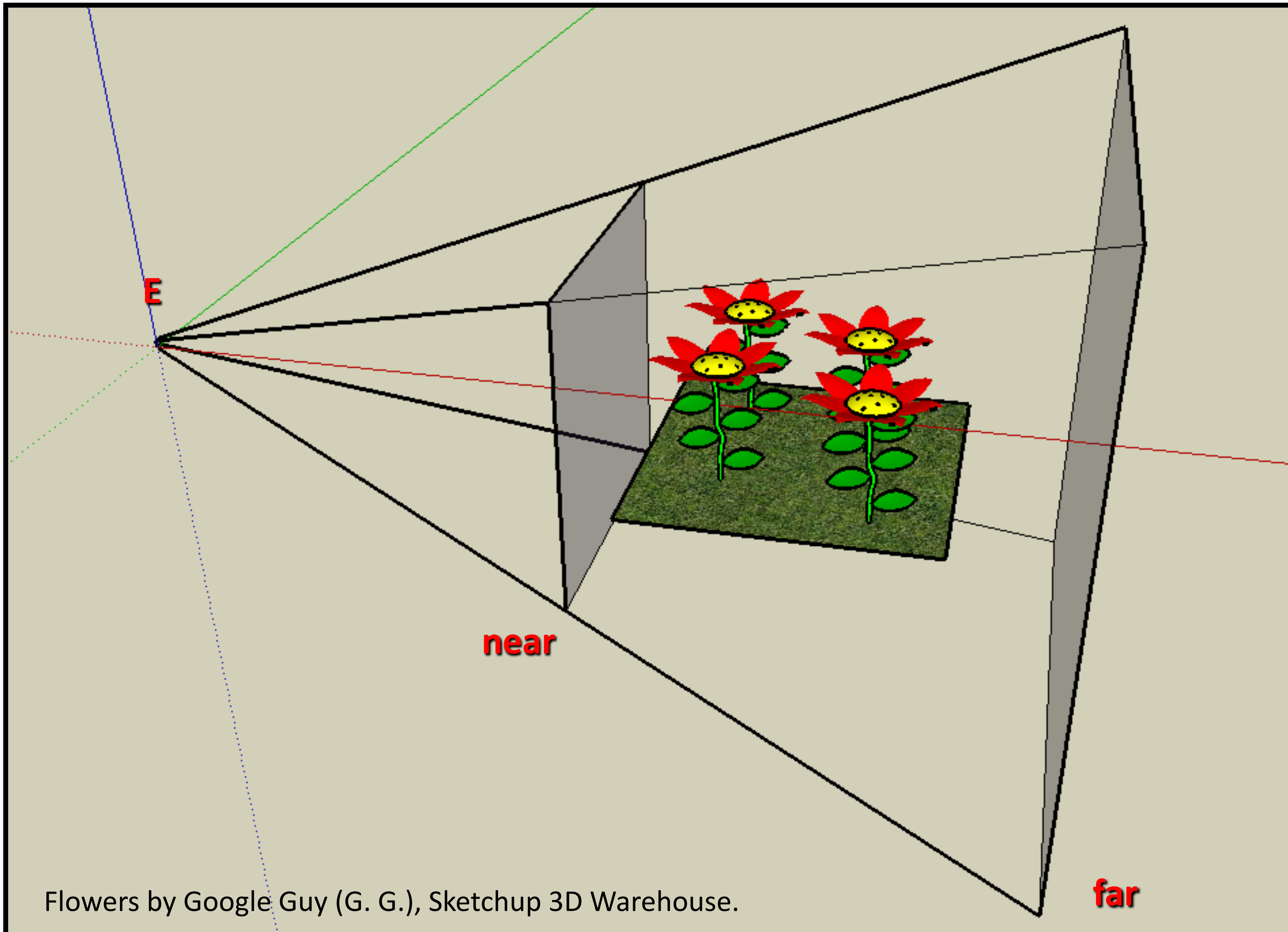
Visualize View Volume (View 2)



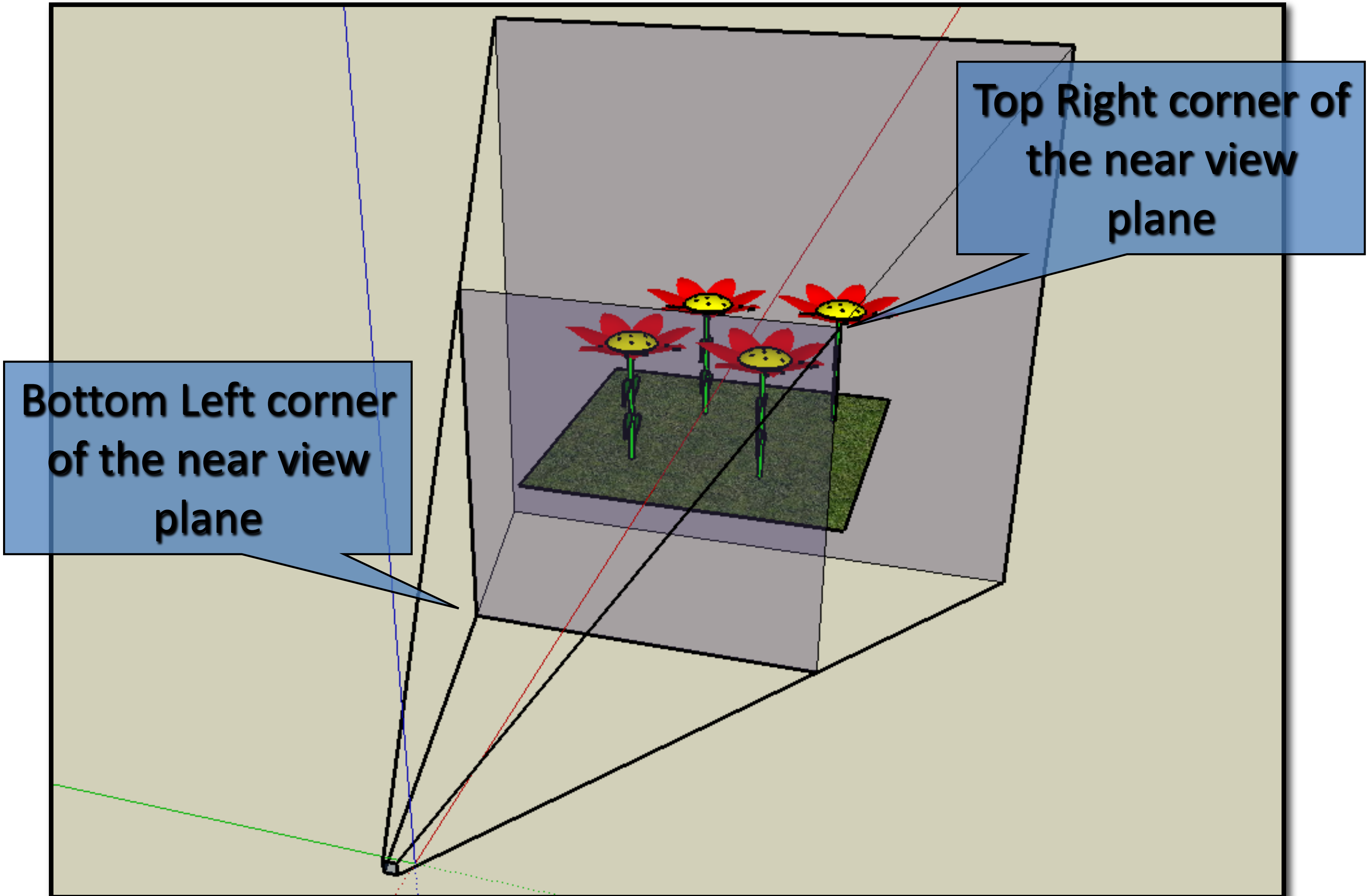
Consider Some Key Points



View Volume - Frustum



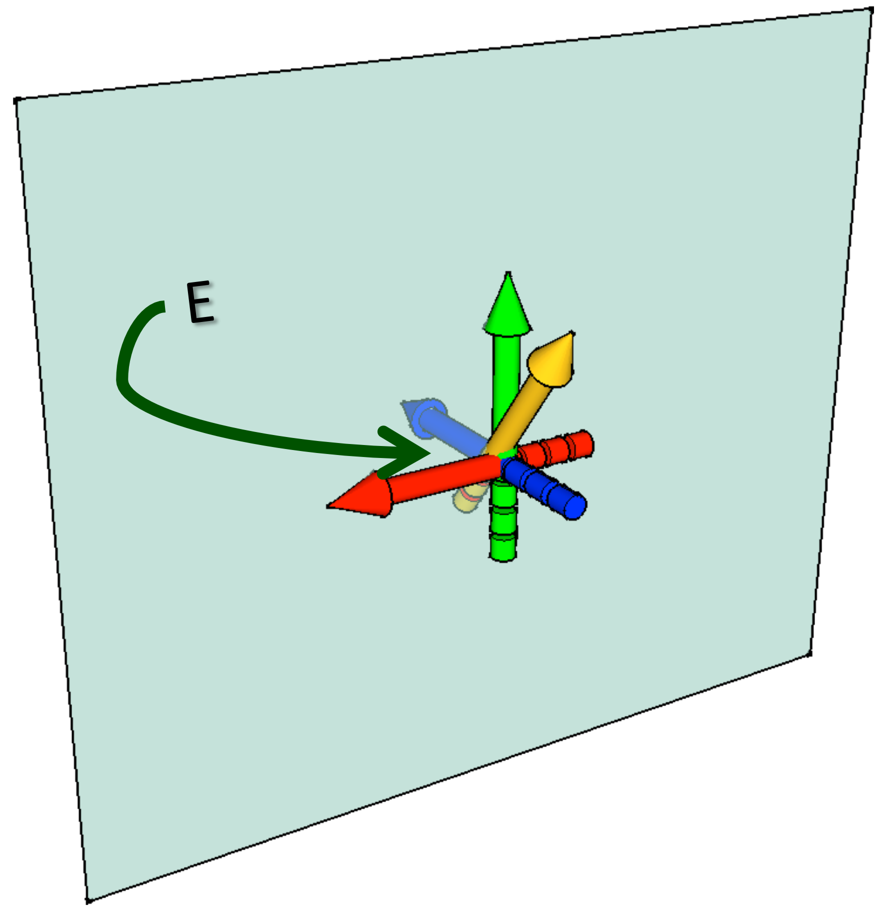
Frustum Continued



Camera Coordinate System

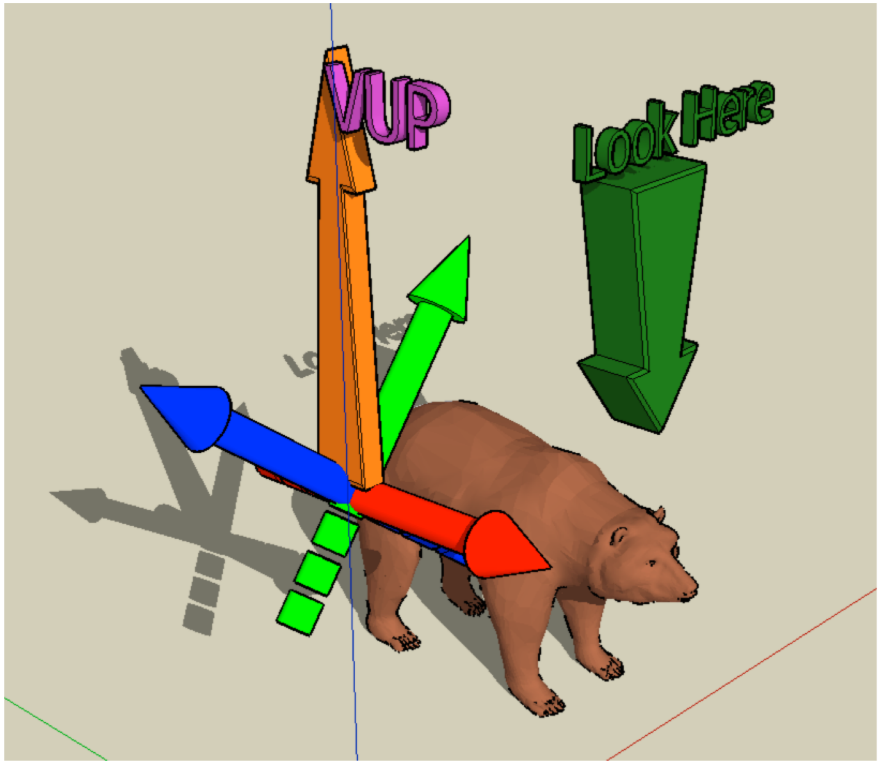
Formally, the view reference coordinate system

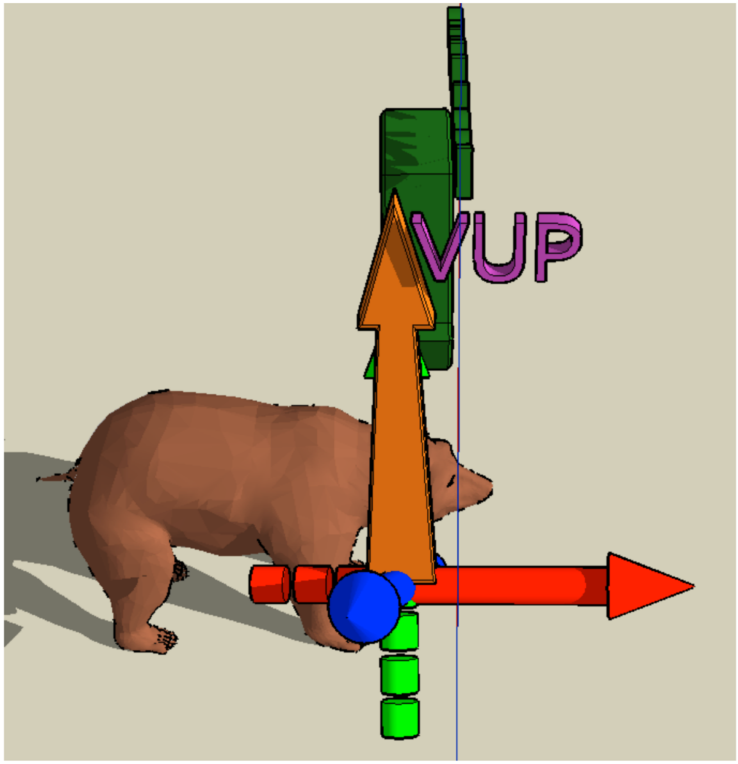
- Eye point E,
  - aka. Focal point, PRP, …
- Image u is red
- Image v is green
- VUP is yellow
- Camera w(z) is blue



Need to Orient the Camera

- Define a “look at” point L. Points E and L define Gaze G.
- Solution for rotation R now similar to axis in axis-angle.
- VUP defines which way is up.

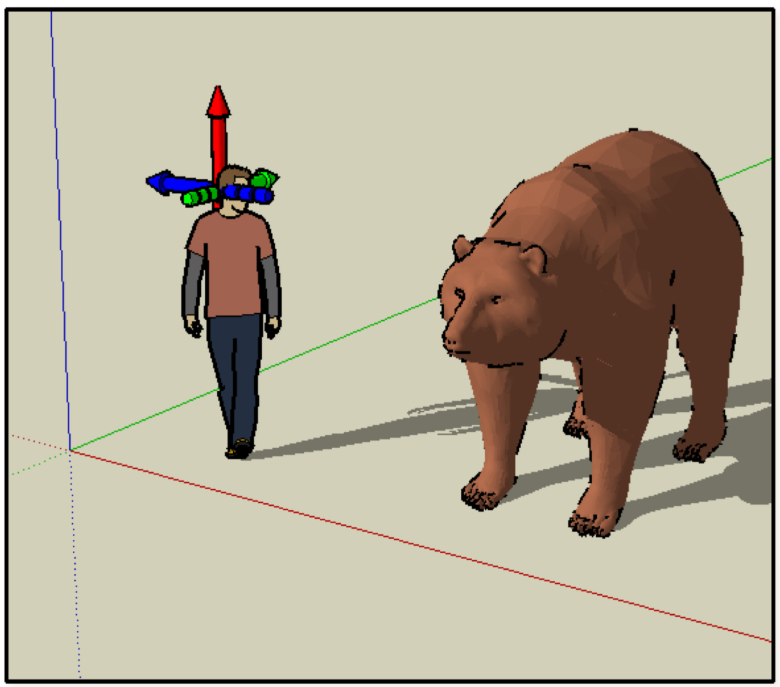


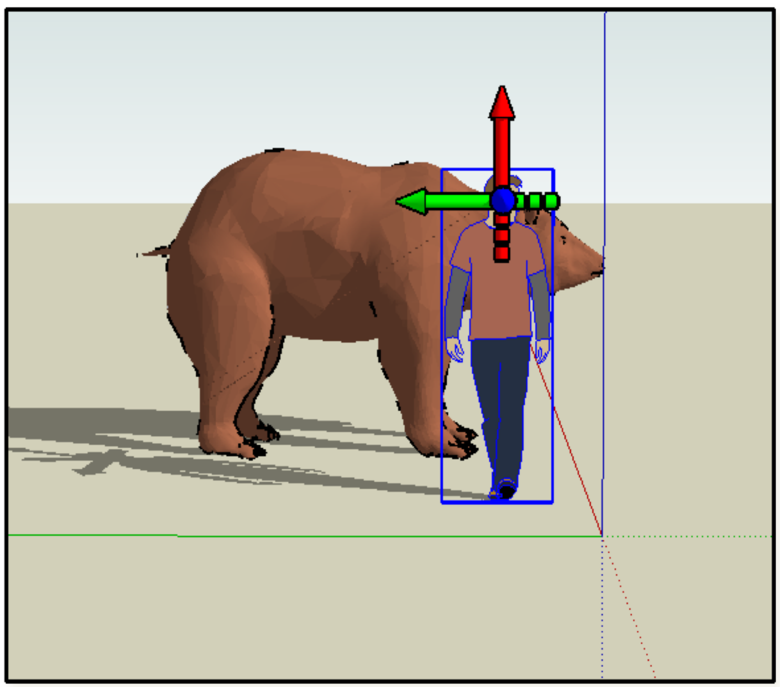


Color coded camera axes: red for u, green for v, blue for w.

Point the Z-Axis away.

- Somewhat counter intuitive at first.
- Standard convention
  - camera looks down the negative z-axis.
- Away from look-at point





Gaze Direction

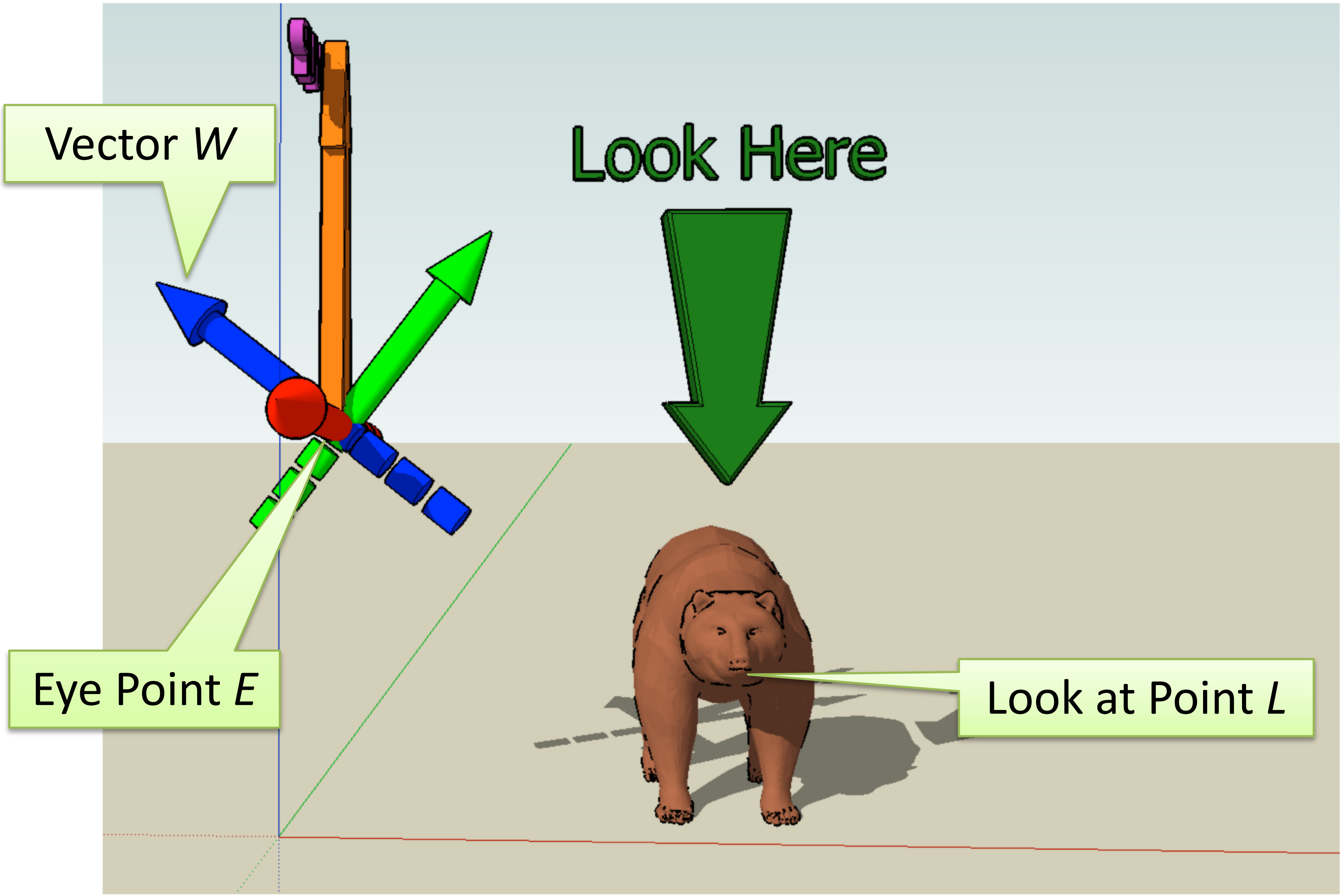
- We have to points in 3D
  - E is the position of the eye given in world.
  - L is the position of the look at point in world.
  - G is the vector indicating gaze direction.
  - Therefore:

$$G = L - E$$

- So, the Z axis of the camera is defined as:

$$W = \frac{E - L}{\|E - L\|}$$

Visualize *E*, *L* and *W*



One of 3 Rows Defined

- Similar to first step in axis angle formulation.
- We have a vector pointing in the Z direction.

$$R = \begin{vmatrix} ? & ? & ? & 0 \\ ? & ? & ? & 0 \\ x_w & y_w & z_w & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

Where recall … $W = \begin{vmatrix} x_w \\ y_w \\ z_w \end{vmatrix} = \dfrac{E - L}{\|E - L\|}$

Resolving *U* and *V*

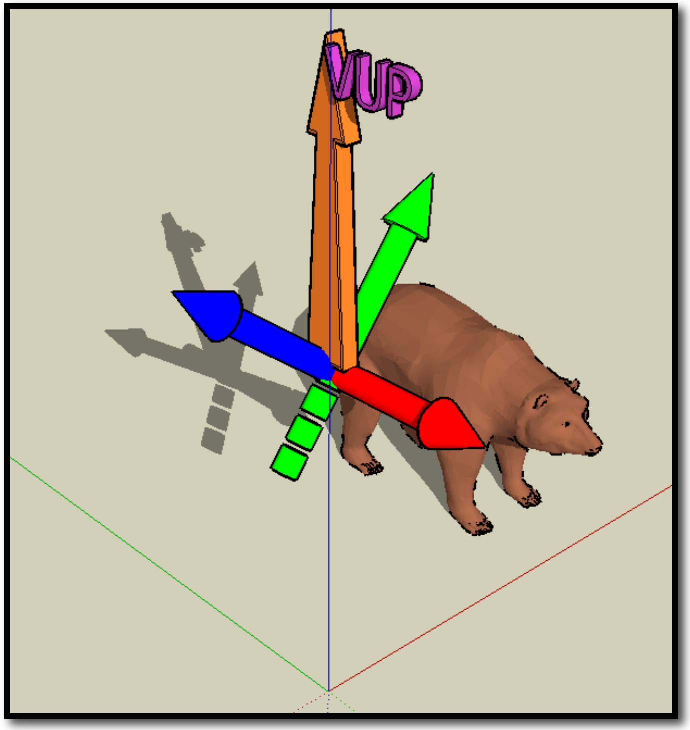
- Consider life in a world with Gravity.
- Gravity means there is an "up".
- Photographers keep their cameras level.
- Which of these looks right to you ….

W & VUP Define Horizontal

- The horizontal axis u is perpendicular to
- … a plane defined by the W and VUP.

$$U = \frac{VUP \times W}{\|VUP \times W\|}$$

$$R = \begin{vmatrix} x_u & y_u & z_u & 0 \\ ? & ? & ? & 0 \\ x_w & y_w & z_w & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$



Last Axis Must Be …

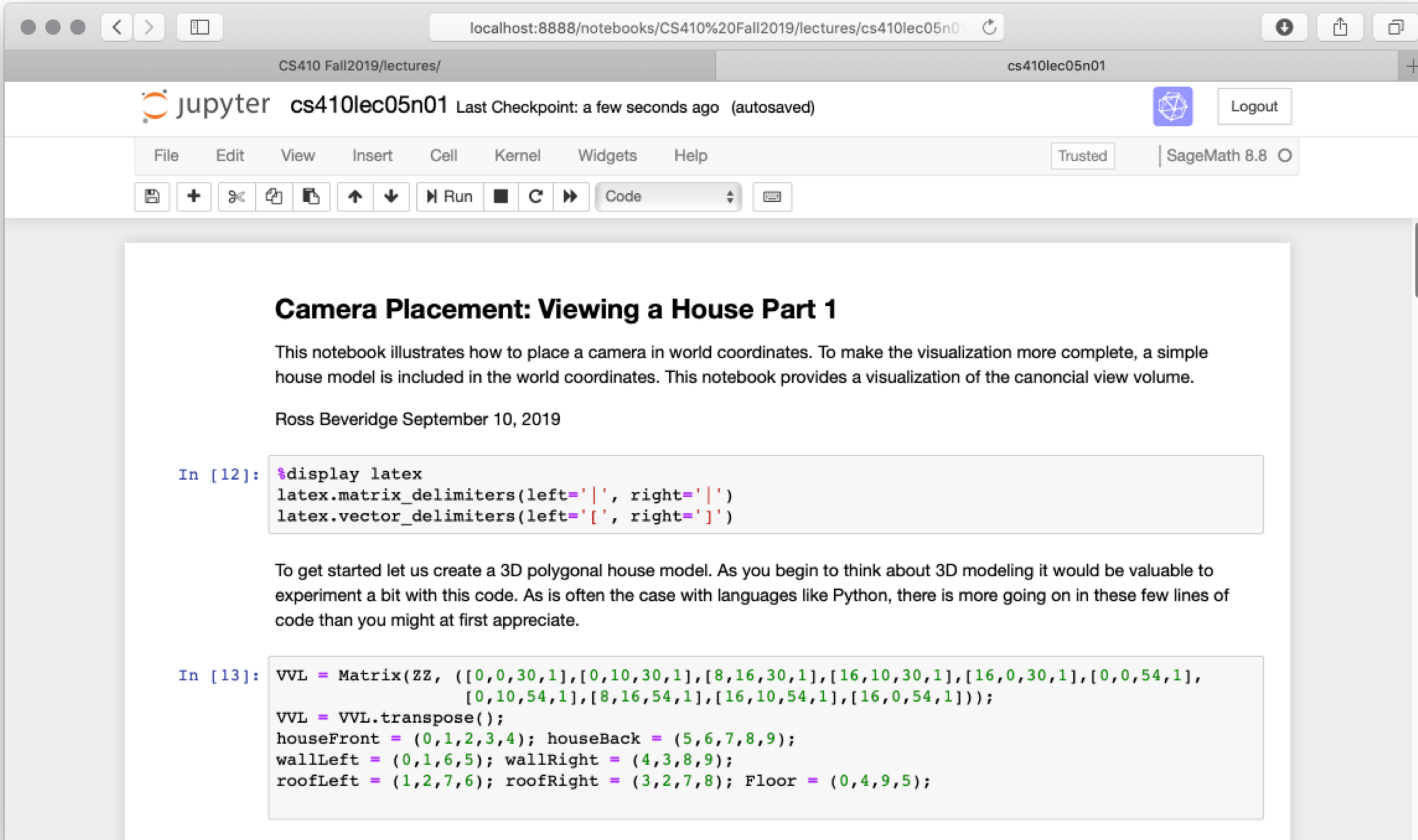
- Given the first two axis, the third is

$$V = W \times U$$

- There is no need to normalize *V*

$$R = \begin{vmatrix} x_u & y_u & z_u & 0 \\ x_v & y_v & z_v & 0 \\ x_w & y_w & z_w & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

Now SageMath …



localhost:8888/notebooks/CS410%20Fall2019/lectures/cs410lec05n0

CS410 Fall2019/lectures/ cs410lec05n01

jupyter cs410lec05n01 Last Checkpoint: a few seconds ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted SageMath 8.8

Run Code

Camera Placement: Viewing a House Part 1

This notebook illustrates how to place a camera in world coordinates. To make the visualization more complete, a simple house model is included in the world coordinates. This notebook provides a visualization of the canoncial view volume.

Ross Beveridge September 10, 2019

In [12]:
```
%display latex
latex.matrix_delimiters(left='|', right='|')
latex.vector_delimiters(left='[', right=']')
```

To get started let us create a 3D polygonal house model. As you begin to think about 3D modeling it would be valuable to experiment a bit with this code. As is often the case with languages like Python, there is more going on in these few lines of code than you might at first appreciate.

In [13]:
```
VVL = Matrix(ZZ, ([0,0,30,1],[0,10,30,1],[8,16,30,1],[16,10,30,1],[16,0,30,1],[0,0,54,1],
                  [0,10,54,1],[8,16,54,1],[16,10,54,1],[16,0,54,1]));
VVL = VVL.transpose();
houseFront = (0,1,2,3,4); houseBack = (5,6,7,8,9);
wallLeft = (0,1,6,5); wallRight = (4,3,8,9);
roofLeft = (1,2,7,6); roofRight = (3,2,7,8); Floor = (0,4,9,5);
```

First Major Aside: The House

- 3D Example needs something to 'lookat'

```
VVL = Matrix(ZZ, ([0,0,30,1],[0,10,30,1],[8,16,30,1],[16,10,30,1],[16,0,30,1],[0,0,54,1],
                  [0,10,54,1],[8,16,54,1],[16,10,54,1],[16,0,54,1]));
VVL = VVL.transpose();
houseFront = (0,1,2,3,4); houseBack = (5,6,7,8,9);
wallLeft = (0,1,6,5); wallRight = (4,3,8,9);
roofLeft = (1,2,7,6); roofRight = (3,2,7,8); Floor = (0,4,9,5);
```

An array of vertices

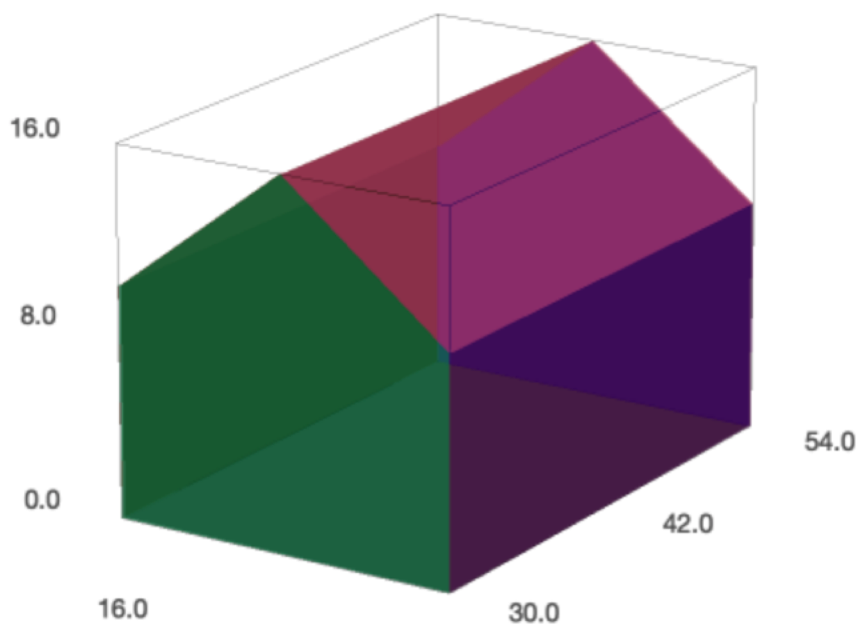
Perhaps the first thing to notice about this example is the way in which vertices are expressed. Namely, in a 4 x N matrix where N is the number of vertices; N = 10 for the house.

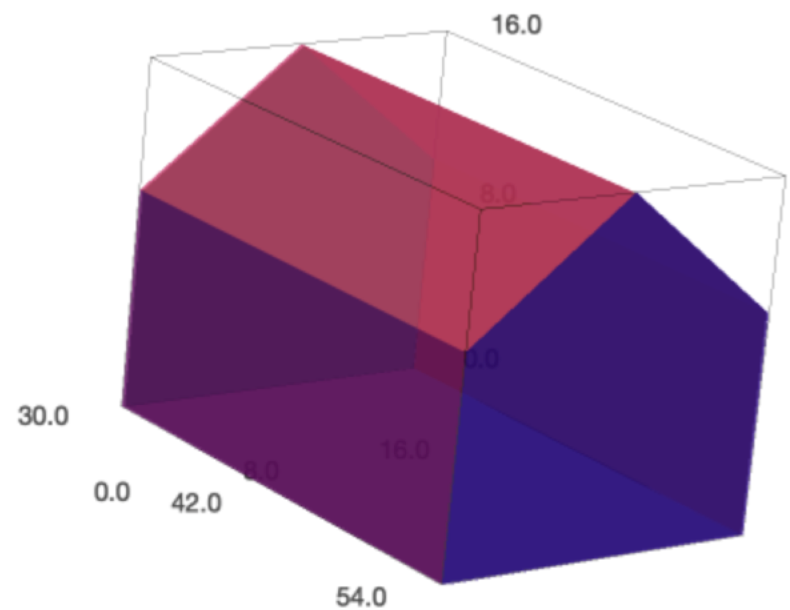
```
pretty_print("VVL = ", VVL)
```

$$\mathrm{VVL} = \left|\begin{array}{cccccccccc} 0 & 0 & 8 & 16 & 16 & 0 & 0 & 8 & 16 & 16 \\ 0 & 10 & 16 & 10 & 0 & 0 & 10 & 16 & 10 & 0 \\ 30 & 30 & 30 & 30 & 30 & 54 & 54 & 54 & 54 & 54 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right|$$

SageMath 3D Drawing of House

- Pay attention to structure, axes, colors …





Configuring a Camera

- Interact with SageMath to see different camera placements and view volumes.