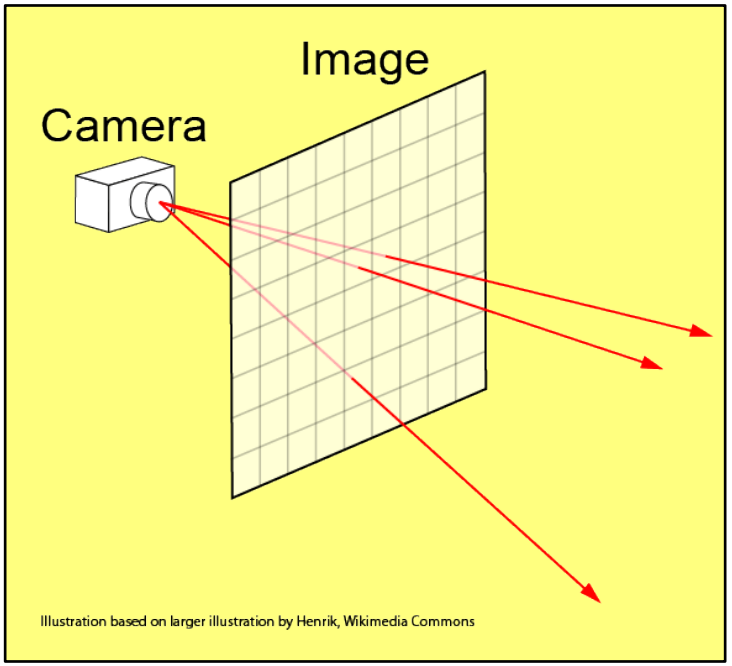
Lecture 9:
Ray Triangle

September 19, 2017

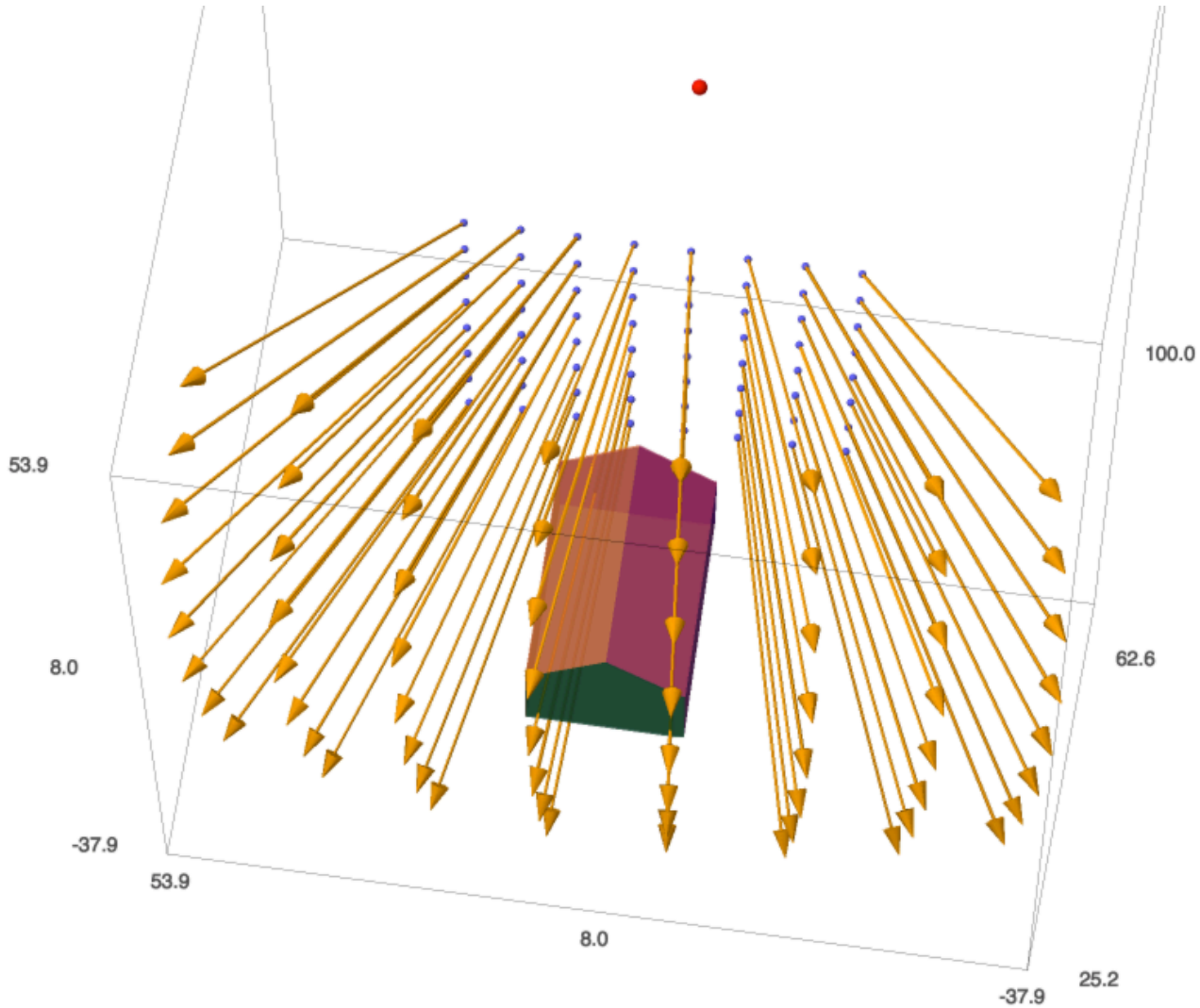
Review: Throwing Rays

- Outer loop of ray tracing:

```
For every pixel {
  Throw ray from pixel;
  Calculate color along ray;
   Fill in pixel;
}
```



Example



Review - Example in Code

Here is an example mapping pixel position (i,j) to a 3D world point:

- The 3D position horizontal away from optical axis: px
- The 3D position vertical away from the optical axis: py
- The 3D position of the camera focal point, the eye: EV
- The 3D direction of the camera z axis (away from scene): WV
- The 3D direction of the camera horizontal axis (unit vector): UV
- The 3D direction of the camera vertical axis (unit vector): VV

```
def pixelPt(i,j):
    px = i/(width-1)*(right-left)+left;
    py = j/(height-1)*(top-bottom)+bottom;
    pixpt = EV + (near * WV) + (px * UV) + (py * VV);
```

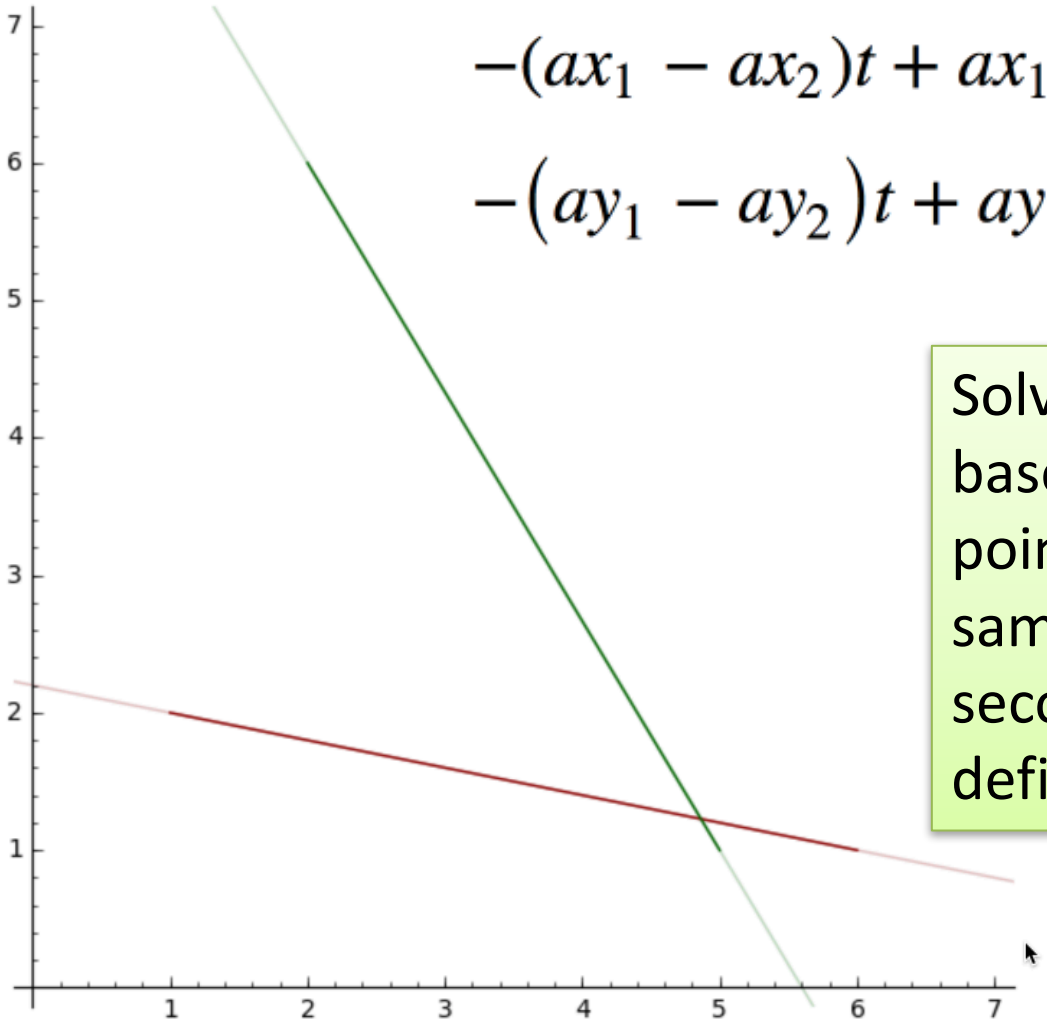
Be Careful! In this example WV points away from the scene AND near is specified as a negative number.

Representing a Ray

- There is a point of origination: L
- There is a unit vector denoting direction: D
- There is a parameter moving along ray: t

$$R(t) = L + tD = \begin{bmatrix} lx \\ ly \\ lz \end{bmatrix} + t* \begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} = \begin{bmatrix} dxt + lx \\ dyt + ly \\ dzt + lz \end{bmatrix}$$

Recall Previous Lecture

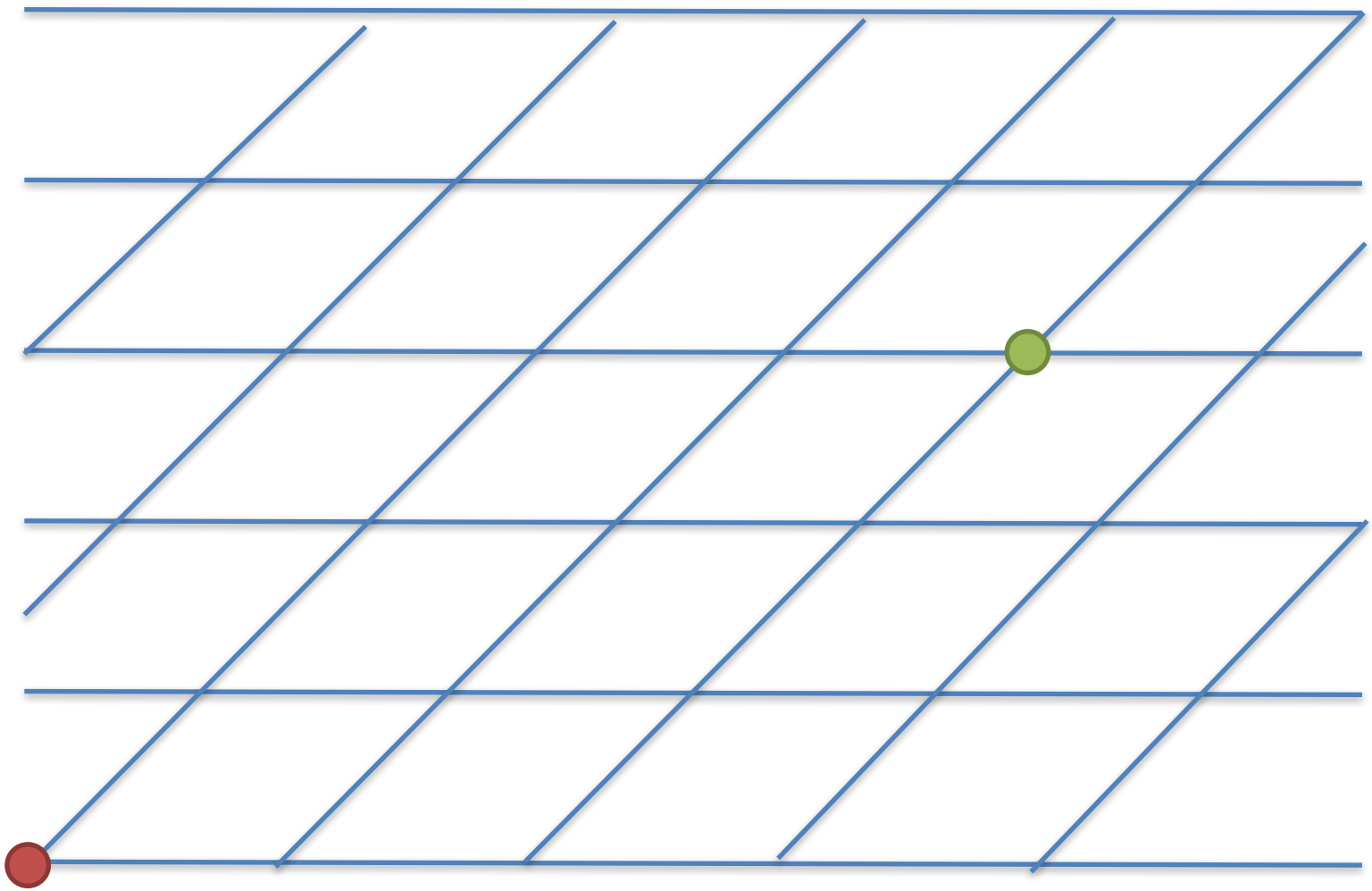


$$-(ax_1 - ax_2)t + ax_1 = -(bx_1 - bx_2)s + bx_1$$

$$-(ay_1 - ay_2)t + ay_1 = -(by_1 - by_2)s + by_1$$

Solve for two parameters based on knowledge that the point on the first line is the same as the point on the second line – the very definition of intersection.

Triangle Warm-up



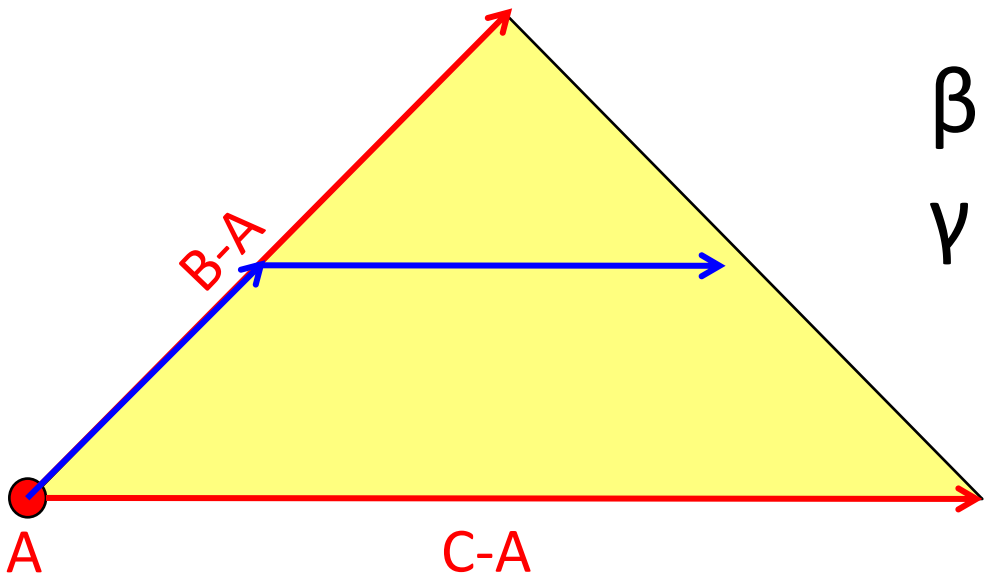
How do you ‘drive’ from the red to the green point?

Ray/Triangle Intersections

- Ray/Triangle intersections are efficient and can be computed directly in 3D
  - No need for ray/plane intersection
- Solution relies on the following implicit definition of a triangle:

$$P = A + \beta(B - A) + \gamma(C - A)$$
$$\beta \geq 0, \gamma \geq 0, \beta + \gamma \leq 1$$

Implicit Triangles



$\beta = 0.5$

$\gamma = 0.48$

Triangle Parametric Form

- There are two free parameters
- They move inside triangle

  ... And outside it as well!

$$P(\beta, \gamma) = A + \beta(B - A) + \gamma(C - A)$$

$$P(\beta, \gamma) = = \begin{bmatrix} -(ax - bx)\beta - (ax - cx)\gamma + ax \\ -(ay - by)\beta - (ay - cy)\gamma + ay \\ -(az - bz)\beta - (az - cz)\gamma + az \end{bmatrix}$$

Find Ray Plane Intersection

- If they intersect, there is a solution to:

$$\begin{bmatrix} dxt + lx \\ dyt + ly \\ dzt + lz \end{bmatrix} = \begin{bmatrix} -(ax - bx)\beta - (ax - cx)\gamma + ax \\ -(ay - by)\beta - (ay - cy)\gamma + ay \\ -(az - bz)\beta - (az - cz)\gamma + az \end{bmatrix}$$

$$\begin{bmatrix} dxt + lx \\ dyt + ly \\ dzt + lz \end{bmatrix} + \begin{bmatrix} (ax - bx)\beta + (ax - cx)\gamma - ax \\ (ay - by)\beta + (ay - cy)\gamma - ay \\ (az - bz)\beta + (az - cz)\gamma - az \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Standard 3x3 Linear System

- Rearranging terms we have a standard:

$$M\ X = Y$$

- Expanded this is …

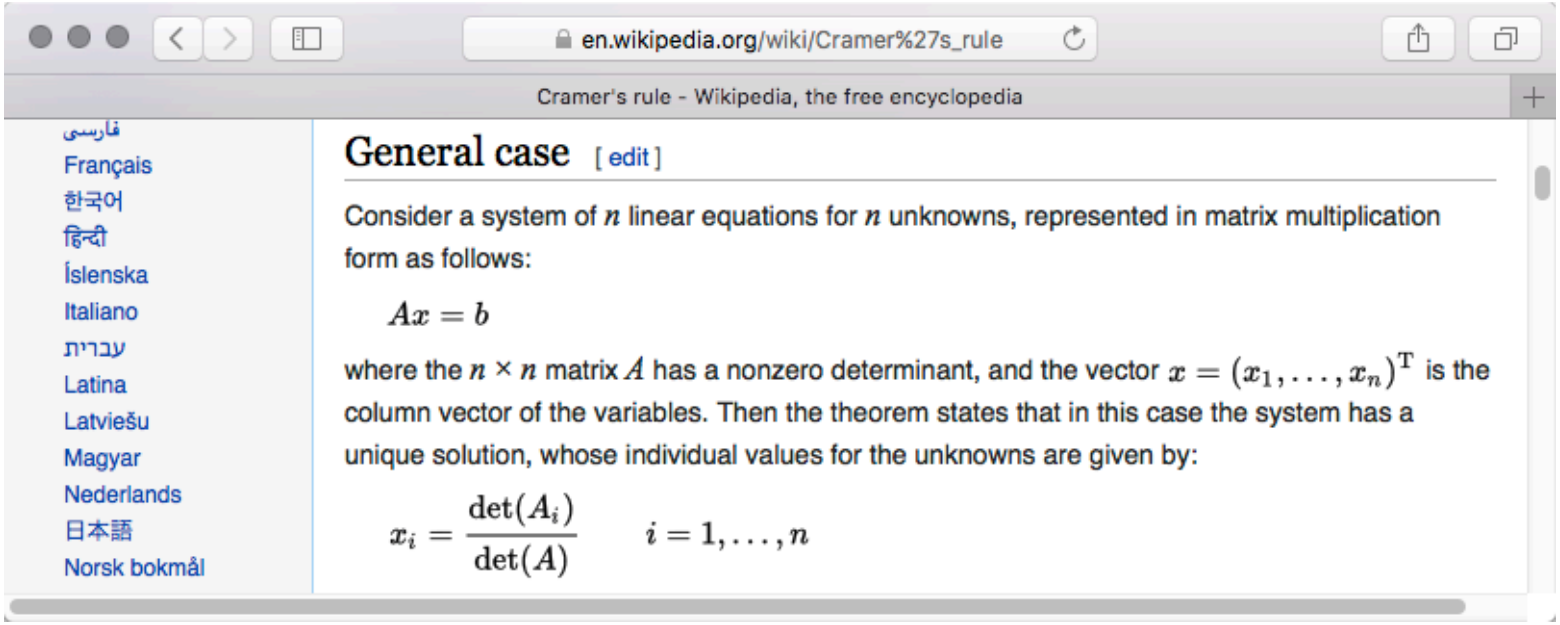
$$\begin{bmatrix} ax - bx & ax - cx & dx \\ ay - by & ay - cy & dy \\ az - bz & az - cz & dz \end{bmatrix} * \begin{bmatrix} \beta \\ \gamma \\ t \end{bmatrix} = \begin{bmatrix} ax - lx \\ ay - ly \\ az - lz \end{bmatrix}$$

Solve for Intersection

- Using favorite linear system method
  - More on this soon
- What if the matrix is singular?
  - then the plane doesn't intersect the ray
- The point is inside the triangle and in front of the camera if and only if
  - $\beta \geq 0$
  - $\gamma \geq 0$
  - $\beta + \gamma \leq 1$
  - $t > 0$
  - Note: Knowing t yields the point of intersection

Using Cramer's Rule

- One approach users Cramer's Rule



The efficiency of this approach compared to a numerical package depends upon details, including the care taken implementing the actual code. For example, using early exit strategies.

Here In Full Glory

$$\mathtt{M} = \begin{bmatrix} ax-bx & ax-cx & dx \\ ay-by & ay-cy & dy \\ az-bz & az-cz & dz \end{bmatrix}$$

$$M_1 = \begin{bmatrix} ax-lx & ax-cx & dx \\ ay-ly & ay-cy & dy \\ az-lz & az-cz & dz \end{bmatrix}, \quad M_2 = \begin{bmatrix} ax-bx & ax-lx & dx \\ ay-by & ay-ly & dy \\ az-bz & az-lz & dz \end{bmatrix}, \quad M_3 = \begin{bmatrix} ax-bx & ax-cx & ax-lx \\ ay-by & ay-cy & ay-ly \\ az-bz & az-cz & az-lz \end{bmatrix}$$

$$\beta = \frac{|M_1|}{|M|} = \frac{((az-cz)dy-(ay-cy)dz)(ax-lx)-((az-cz)dx-(ax-cx)dz)(ay-ly)+((ay-cy)dx-(ax-cx)dy)(az-lz)}{((az-cz)dy-(ay-cy)dz)(ax-bx)-((az-cz)dx-(ax-cx)dz)(ay-by)+((ay-cy)dx-(ax-cx)dy)(az-bz)}$$

$$\gamma = \frac{|M_2|}{|M|} = \frac{((az-lz)dy-(ay-ly)dz)(ax-bx)-((az-lz)dx-(ax-lx)dz)(ay-by)+((ay-ly)dx-(ax-lx)dy)(az-bz)}{((az-cz)dy-(ay-cy)dz)(ax-bx)-((az-cz)dx-(ax-cx)dz)(ay-by)+((ay-cy)dx-(ax-cx)dy)(az-bz)}$$

$$t = \frac{|M_3|}{|M|} = \frac{((ay-ly)(az-cz)-(ay-cy)(az-lz))(ax-bx)-((ax-lx)(az-cz)-(ax-cx)(az-lz))(ay-by)+((ax-lx)(ay-cy)-(ax-cx)(ay-ly))(az-bz)}{((az-cz)dy-(ay-cy)dz)(ax-bx)-((az-cz)dx-(ax-cx)dz)(ay-by)+((ay-cy)dx-(ax-cx)dy)(az-bz)}$$

Without serious effort to collect terms this solution will run slower than a numerical solver.

However, collecting terms is not that difficult.

Sage Notebook

localhost:8080/home/admin/21/

Starting, please wait... | rayTriangle -- Sage

The Sage Notebook
Version 7.3

admin Toggle | Home | Published | Log | Settings | Help | Report a Problem | Sign out

rayTriangle

last edited Oct 11, 2016, 12:08:41 PM by admin

Save | Save & quit | Discard & quit

File... | Action. | Data... | sage | ☑ Typeset ☐ Load 3-D Live ☐ Use java for 3-D

Print | Worksheet | Edit | Text | Revisions | Share | Publish

Here is a solution to the ray-triangle intersection problem that does not require explicitly intersecting the ray with a full 3D plane. This approach instead takes advantage of a simple property of triangles, namely that any point in a triangle may be expressed as some offset from a key vertex in directions determined by vectors derived from this key vertex and the remaining two vertices.

In this example the actual scalar variables are being made explicit, hence the next setup block is rather large in so much as it declares all the necessary symbolic scalar variables.

```
latex.matrix_delimiters("[", "]")
var('ax', 'ay', 'az', 'bx', 'by', 'bz', 'cx', 'cy', 'cz');
var('lx', 'ly', 'lz', 'dx', 'dy', 'dz');
var('t', 'beta', 'gamma');
Av = vector(SR, 3, ('ax', 'ay', 'az'));
Bv = vector(SR, 3, ('bx', 'by', 'bz'));
Cv = vector(SR, 3, ('cx', 'cy', 'cz'));
Lv = vector(SR, 3, ('lx', 'ly', 'lz'));
Dv = vector(SR, 3, ('dx', 'dy', 'dz'));
A = matrix(SR, 3,1, Av); B = matrix(SR, 3,1, Bv); C = matrix(SR, 3,1, Cv);
L = matrix(SR, 3,1, Lv); D = matrix(SR, 3,1, Dv);
```

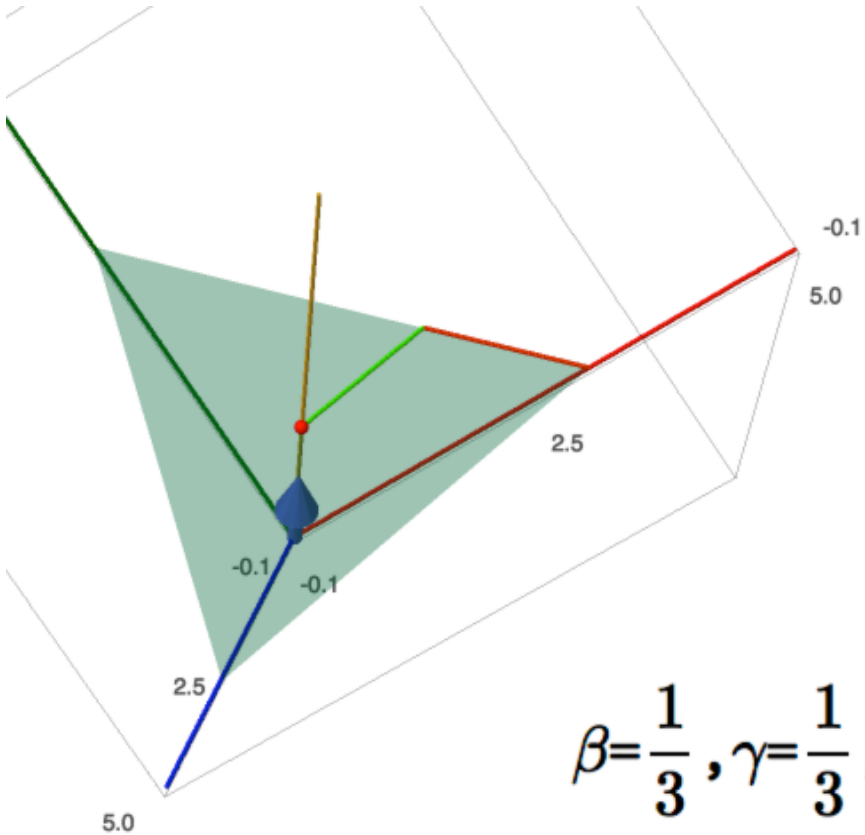
Problem / Opportunity

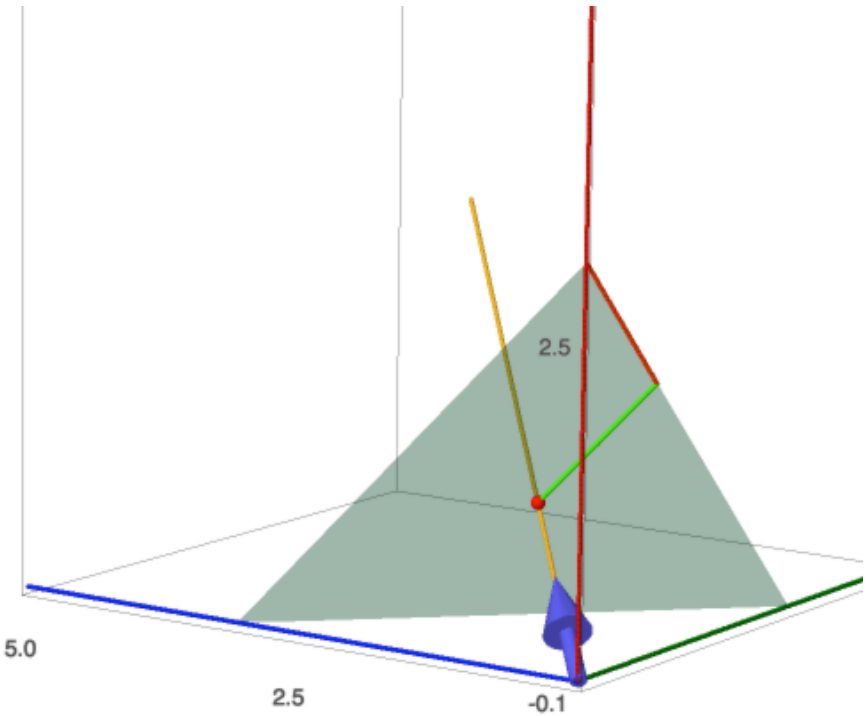
- You have two ways to compute the same three values.
- One, at least, is rife with chances to make errors when being converted to code.
- But, that same option holds promise of efficiency through early termination.

How would one write code to confidently debug and test a relatively complicated geometric computation …

Example 1 Visualization

Ray from origin in direction (1,1,1) with triangle pinned at 3 out each of the X, Y and Z axes.





$$\beta=\frac{1}{3}, \gamma=\frac{1}{3}, t=\sqrt{3}$$