The Ray Tracing Algorithm

January 25, 2010

Logistics

- OptiX update
- Schedule update
- Web site update
- Outstanding procedural questions?

Rays

- Line segment: two points
- Ray: a point and a vector
- Line segment: a ray and a distance
- Line segment: a ray and two distances
- Bounded ray: two ordered points
- Bounded ray: a ray and two distances
- Directed line segment: bounded ray
- Line: any of the above

Rays

- Usyally ray consists of a point and a vector:

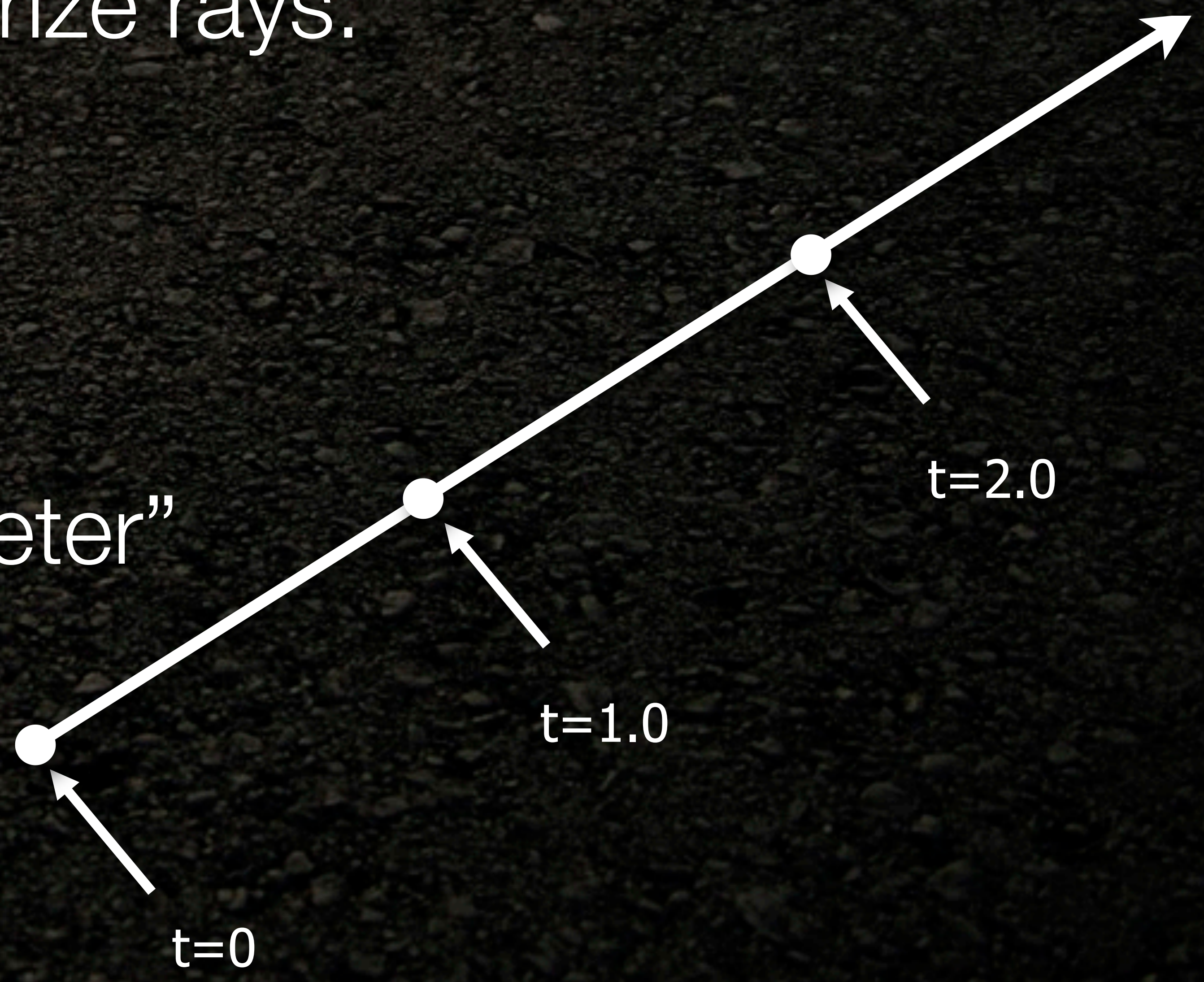
```
Class Ray {
  Point origin;
  Vector direction;
  ...
};
```

Parametric Rays

- We usually parameterize rays:

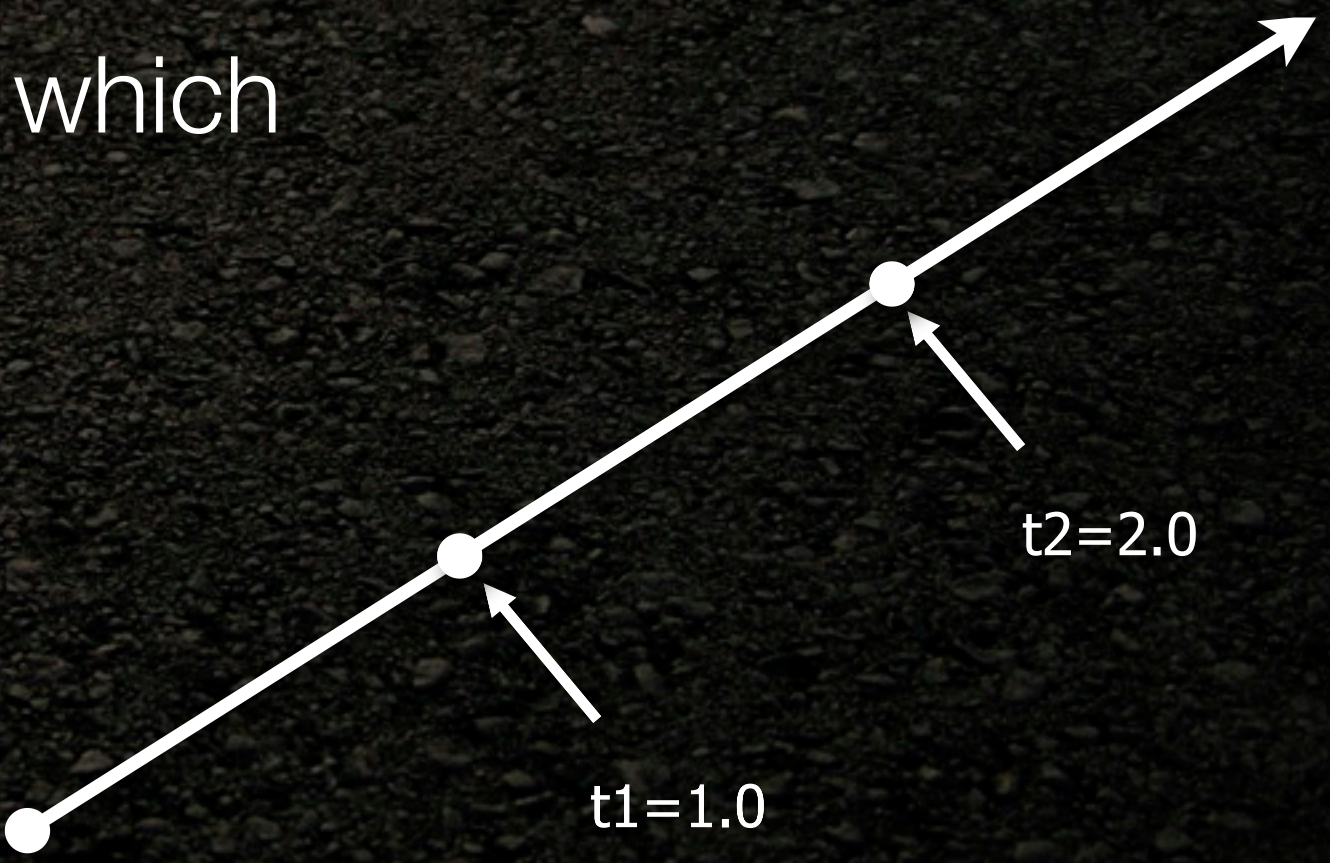
Where O is the origin,
V is direction,
and t is the "ray parameter"

$$\vec{P} = \vec{O} + t\vec{V}$$



Bounded Rays

The interval [t1,t2] says which part of the ray is "live"



$$\overrightarrow{P} = \overrightarrow{O} + t\overrightarrow{V}$$

Ray-Planes

- A implicit equation for a plane can be defined with a Vector (the normal to the plane) and a point on the plane:

$$(P - P0) \bullet N = 0$$

- A parametric ray is : $P(t) = O + tV$

- A ray intersection is an combination of those equations:

$$(O + tV - P0) \bullet N = 0$$

Ray plane 2

- Equation from last slide:

$$(O + tV - P0) \bullet N = 0$$

- Rearrange: $$t(V \bullet N) + (O - P0) \bullet N = 0$$

- Solve for t: $$t = \frac{(P0 - O) \bullet N}{V \bullet N}$$

Colors

- For the purpose of this class, Color is Red, Green, Blue
- Range is 0-1 for LDR and positive (usually) for HDR
- Other color models will be discussed briefly in a few weeks
- Colors should be represented using the “float” datatype - others just don’t make sense
- Define operators that make sense

Image gotchas

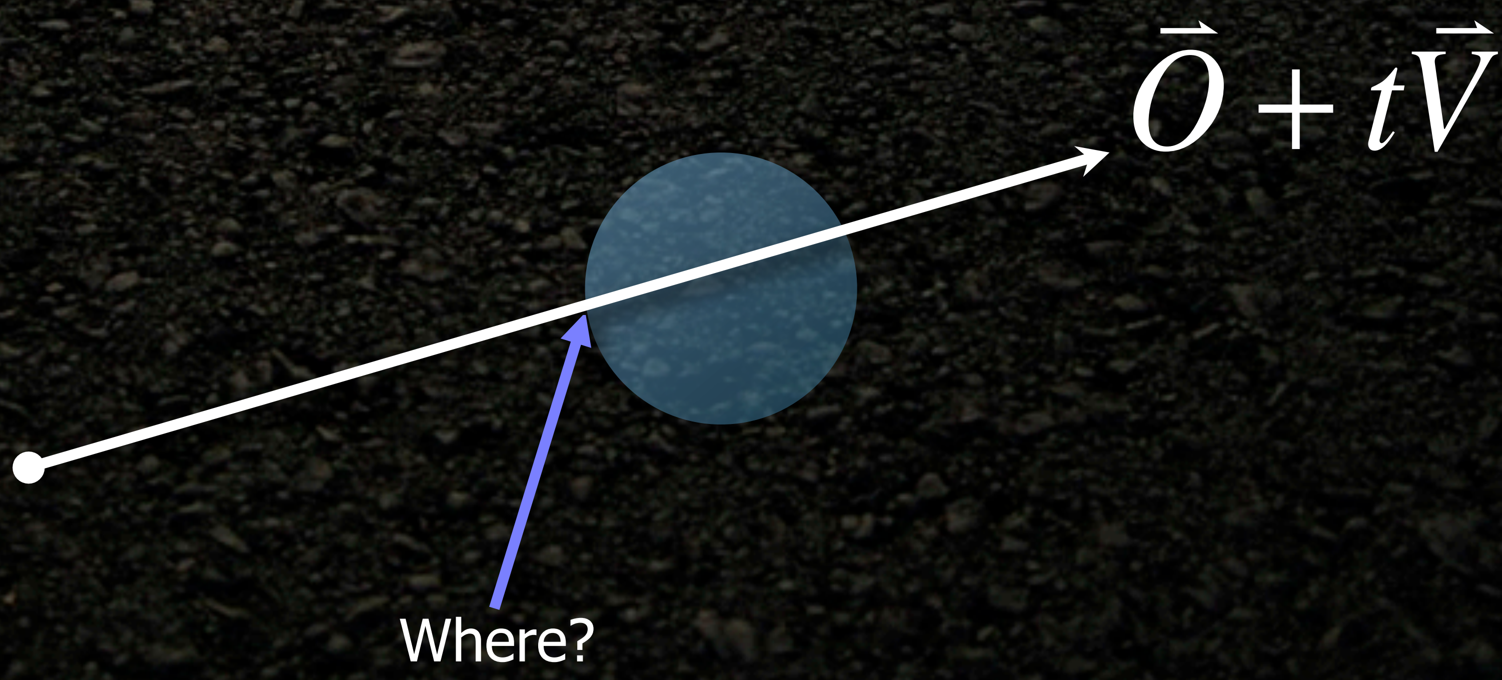
- Be careful - image coordinate system is "upside down"

Real world
Our ray tracer
OpenGL
Taught since 2nd grade

Televisions
Raster Images
Other 1950's technology

Geometric Queries

- Back to the original question:
  What queries can we perform on our virtual geometry?
- Ray tracing: determine if (and where) rays hit an object



Ray plane again

- We have: $t = \dfrac{(P0 - O) \bullet N}{V \bullet N}$

- What does it mean when the denominator is small?

- What does it mean when t is negative?

Ray-sphere intersection

- Points on a sphere are equidistant from the center of the sphere
- Our measure of distance: dot product
- Equation for sphere:

$$\left(\vec{P}-\vec{C}\right)\bullet\left(\vec{P}-\vec{C}\right)-r^2=0$$

$$t^2\vec{V}\bullet\vec{V}+2t\left(\vec{O}-\vec{C}\right)\bullet\vec{V}+\left(\vec{O}-\vec{C}\right)\bullet\left(\vec{O}-\vec{C}\right)-r^2=0$$

Ray-sphere intersection, improved

$$t^2\vec{V}\bullet\vec{V} + 2t\left(\vec{O}-\vec{C}\right)\bullet\vec{V} + \left(\vec{O}-\vec{C}\right)\bullet\left(\vec{O}-\vec{C}\right) - r^2 = 0$$

$$\text{Vector } \vec{O}' = \vec{O} - \vec{C}$$

$$a = \vec{V}\bullet\vec{V}$$

$$b = 2\vec{O}'\bullet\vec{V}$$

$$c = \vec{O}'\bullet\vec{O}' - r^2$$

Solve for the roots the using quadratic equation.

Note that because b has a “2” in it we can dervive some efficiencies.

Ray tracing architecture

The major components in a ray tracer are:

- Camera (Pixels to Rays)
- Objects (Rays to intersection info)
- Materials (Intersection info and light to color)
- Lights
- Background (Rays to Color)

- All together: a Scene

Ray tracing algorithm

Create scene (objects, materials, lights, camera, background)

Preprocess scene

foreach frame

  foreach pixel

    foreach sample

      generate ray

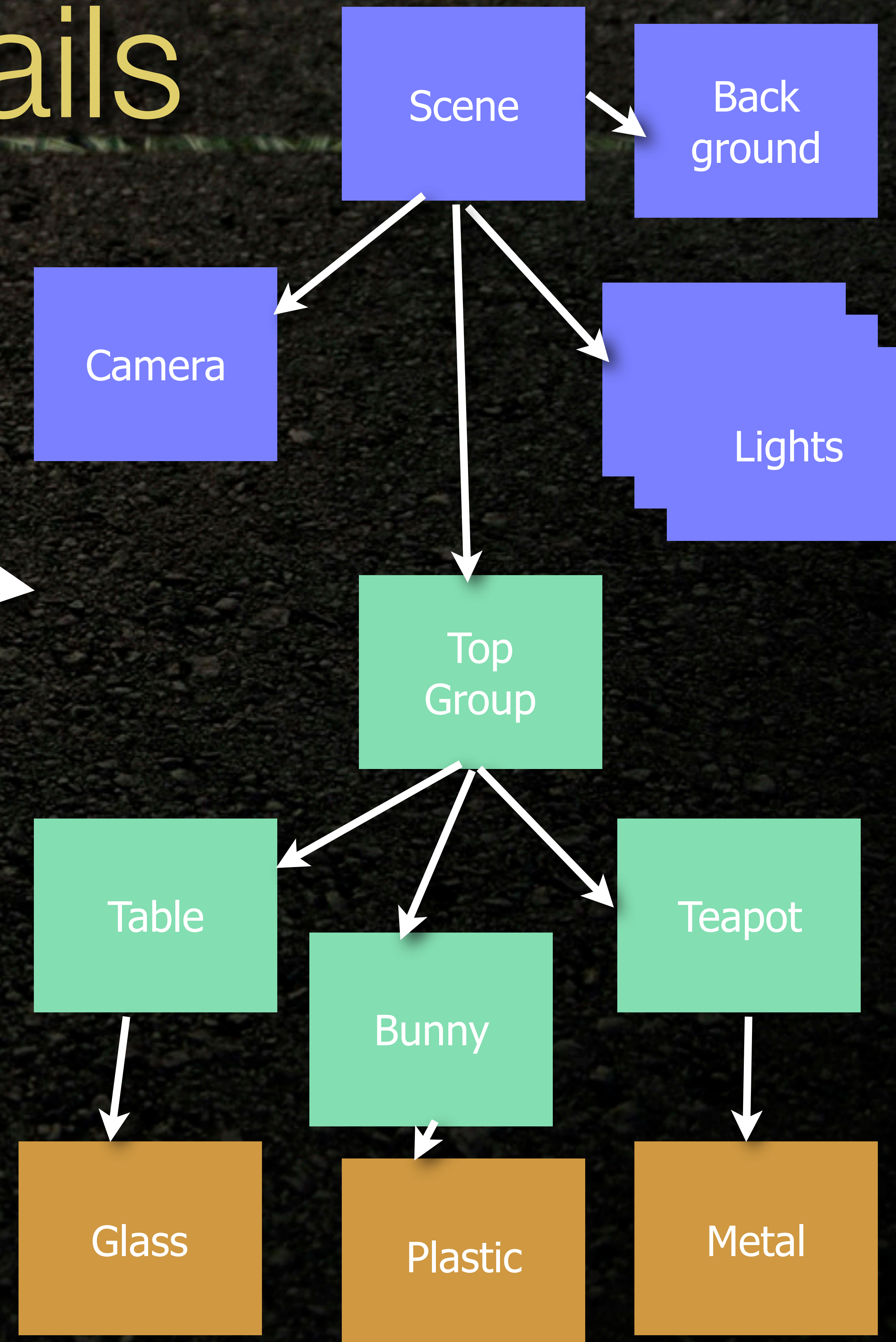
      intersect ray with objects

      find normal of closest object

      shade intersection point

Mutually recursive

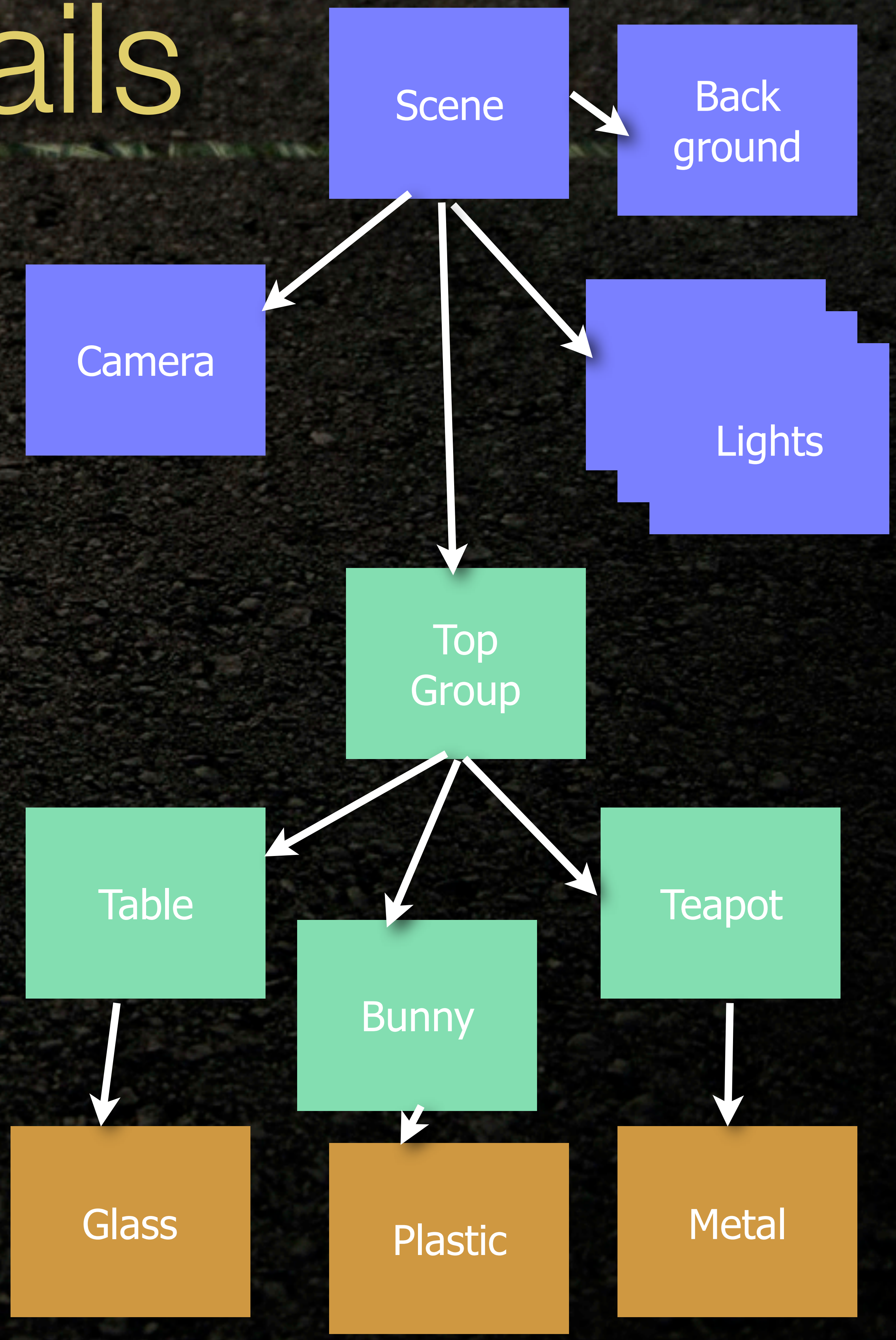
Details


Scene
Back ground
Camera
Lights
Top Group
Table
Bunny
Teapot
Glass
Plastic
Metal

Details

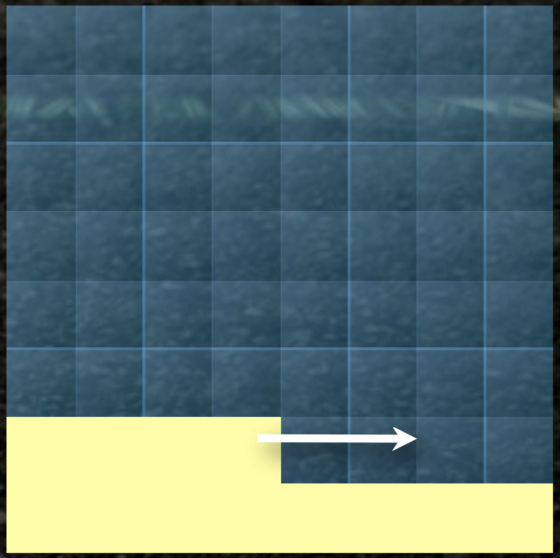
Create scene

Preprocess scene

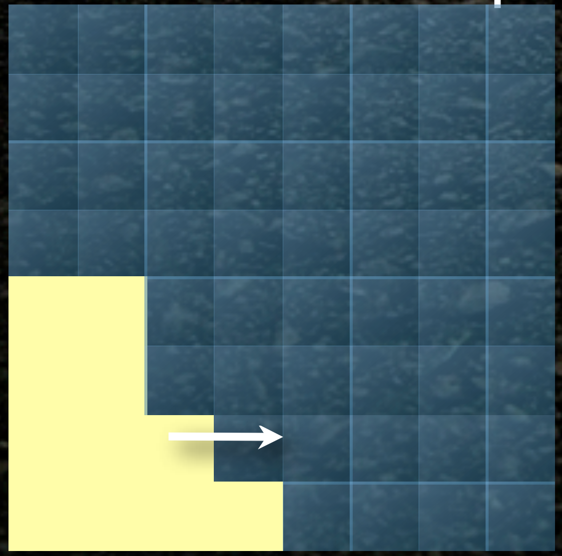


Details

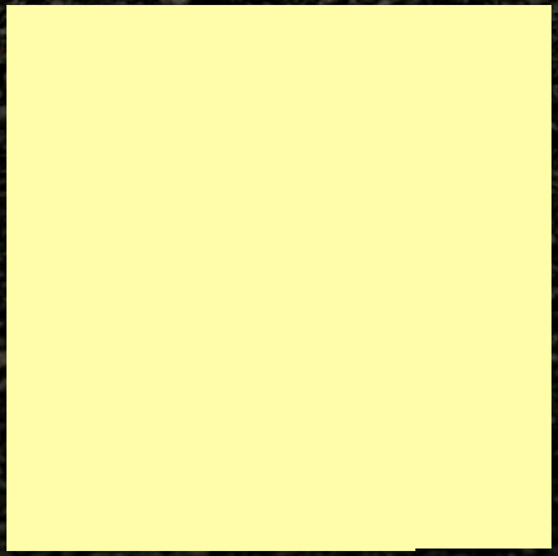
Create scene
Preprocess scene
foreach pixel



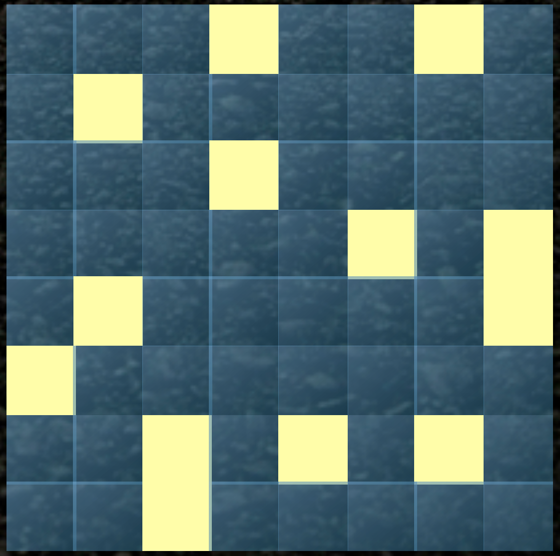
Row-major order



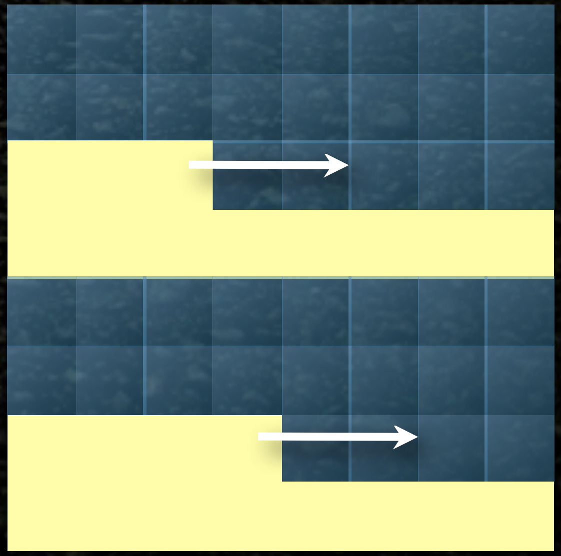
Tiled



Progressive



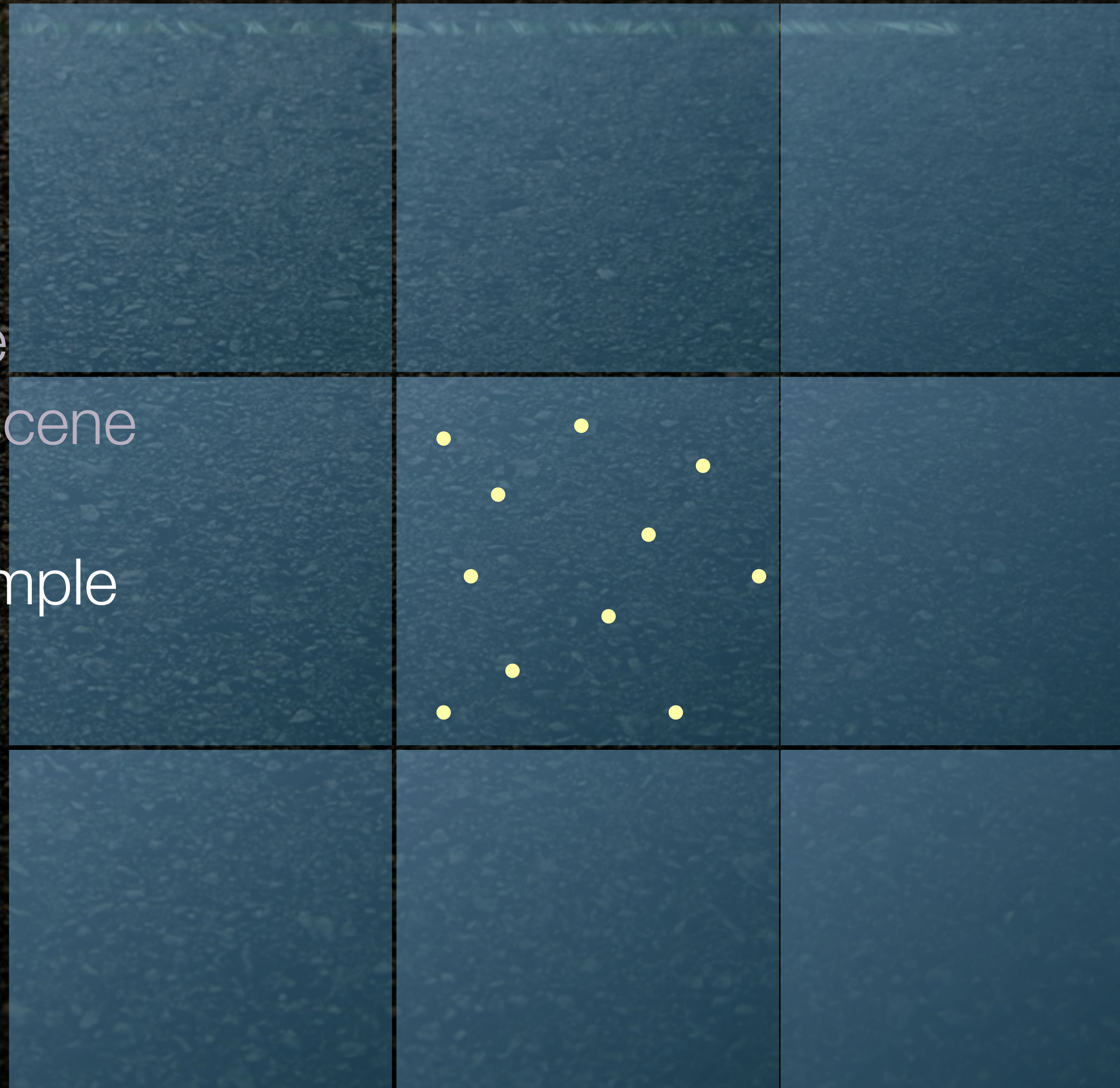
Frameless rendering



Parallel

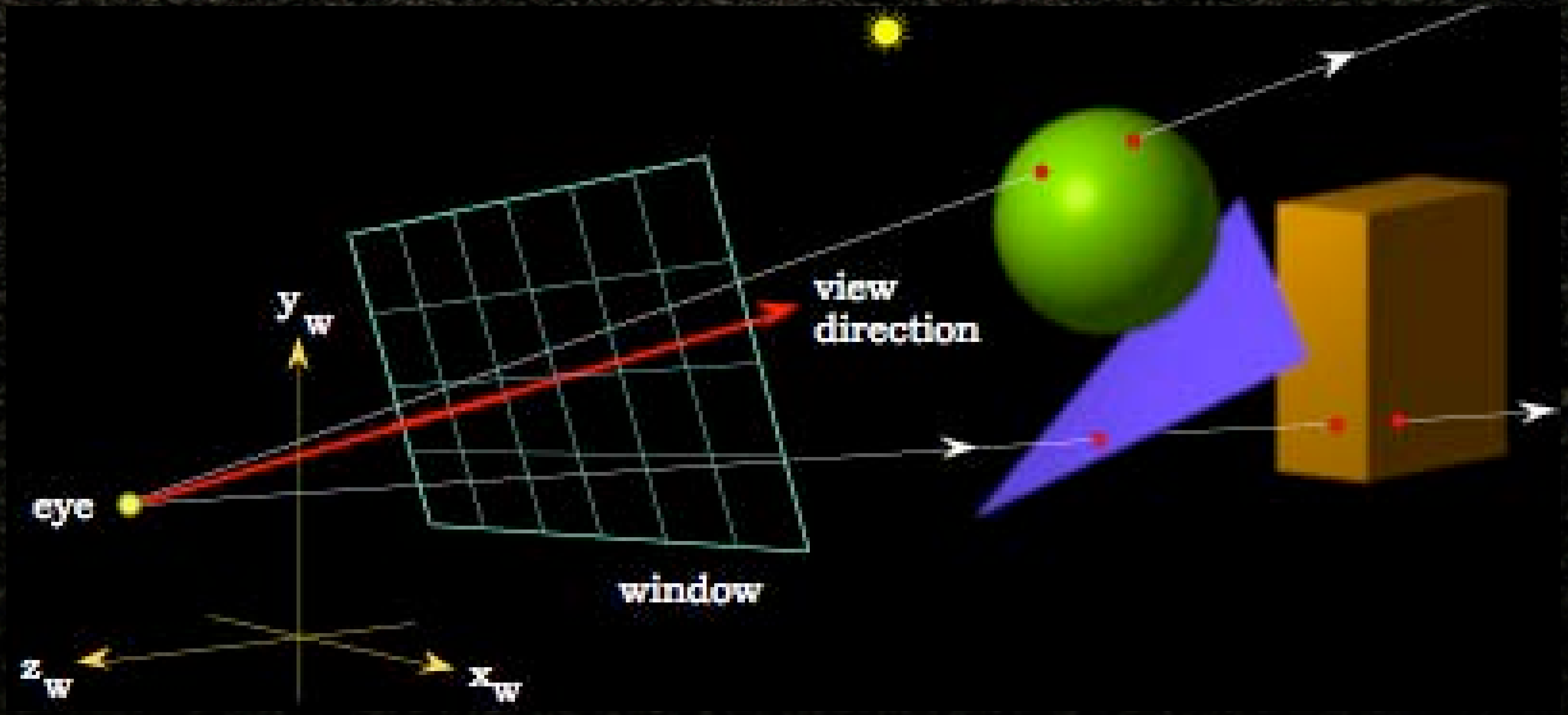
Details

Create scene
Preprocess scene
foreach pixel
    foreach sample



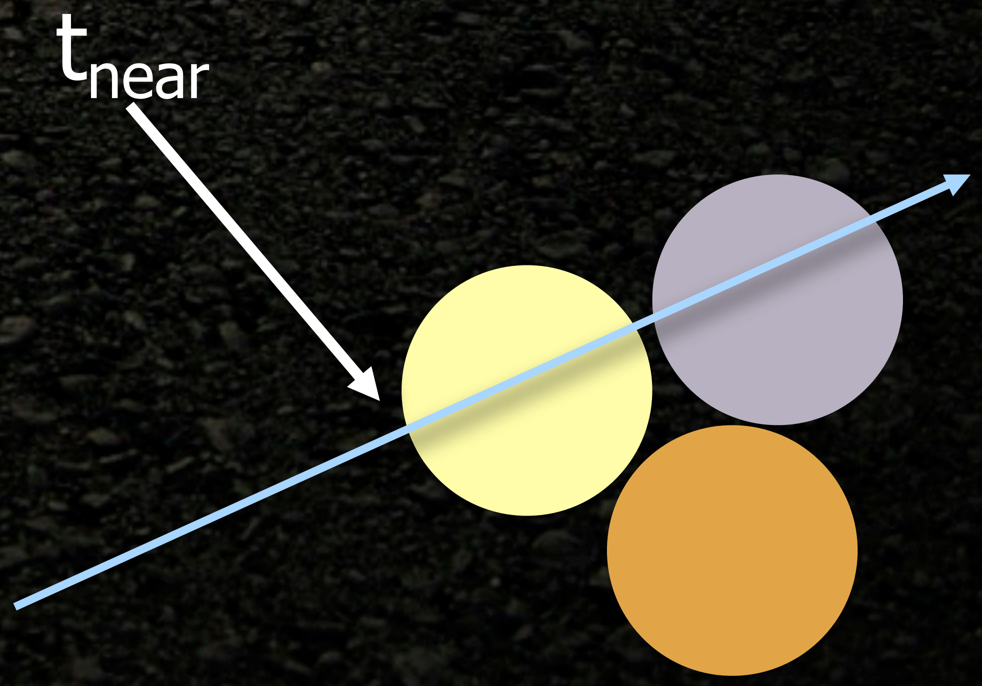
Details

Create scene
Preprocess scene
foreach pixel
  foreach sample
    generate ray



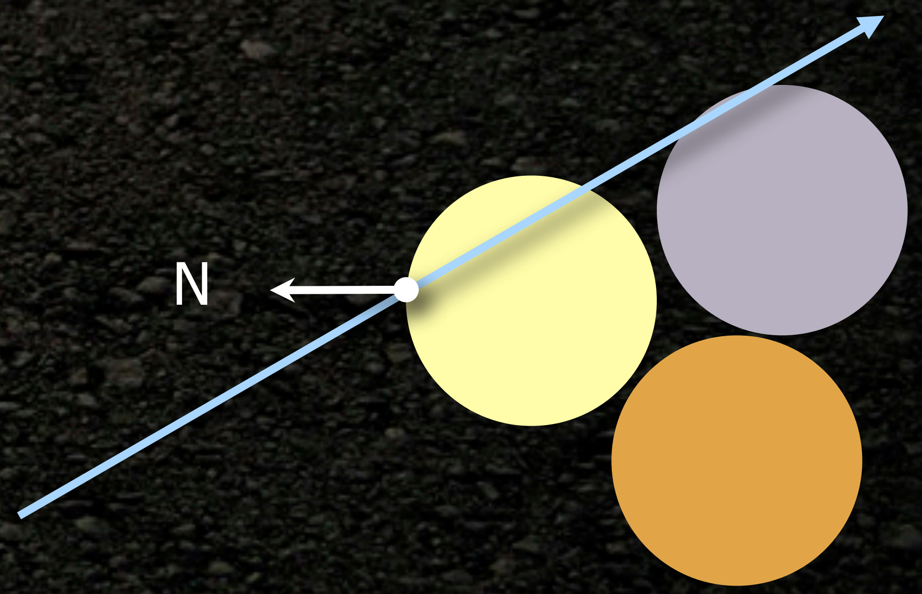
Details

Create scene
Preprocess scene
foreach pixel
    foreach sample
        generate ray
        intersect ray with objects



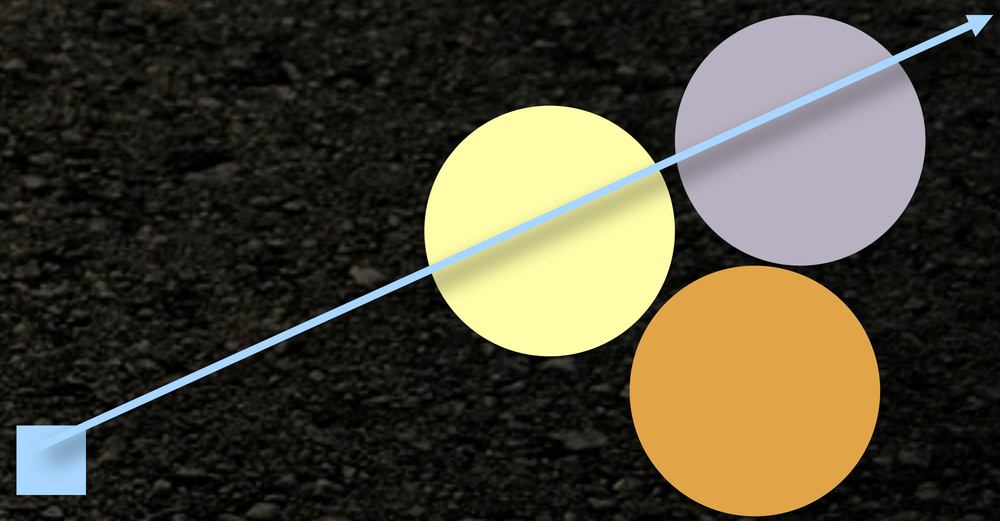
Details

Create scene
Preprocess scene
foreach pixel
    foreach sample
        generate ray
        intersect ray with objects
        find normal of closest object



Details

Create scene
Preprocess scene
foreach pixel
    foreach sample
        generate ray
        intersect ray with objects
        find normal of closest object
        shade intersection point



Ray tracing algorithm

Create scene (objects, materials, lights, camera, background)

Preprocess scene

foreach frame

  foreach pixel

    foreach sample

      generate ray

      intersect ray with objects

      find normal of closest object

      shade intersection point

Mutually recursive

Ray tracing architecture

The major components in a ray tracer are:

- Camera (Pixels to Rays)
- Objects (Rays to intersection info)
- Materials (Intersection info and light to color)
- Lights
- Background (Rays to Color)

- All together: a Scene

Ray tracing algorithm

Create scene (objects, materials, lights, camera, background)

Preprocess scene

foreach frame

  foreach pixel

    foreach sample

      generate ray

      intersect ray with objects

      find normal of closest object

      shade intersection point

Mutually recursive

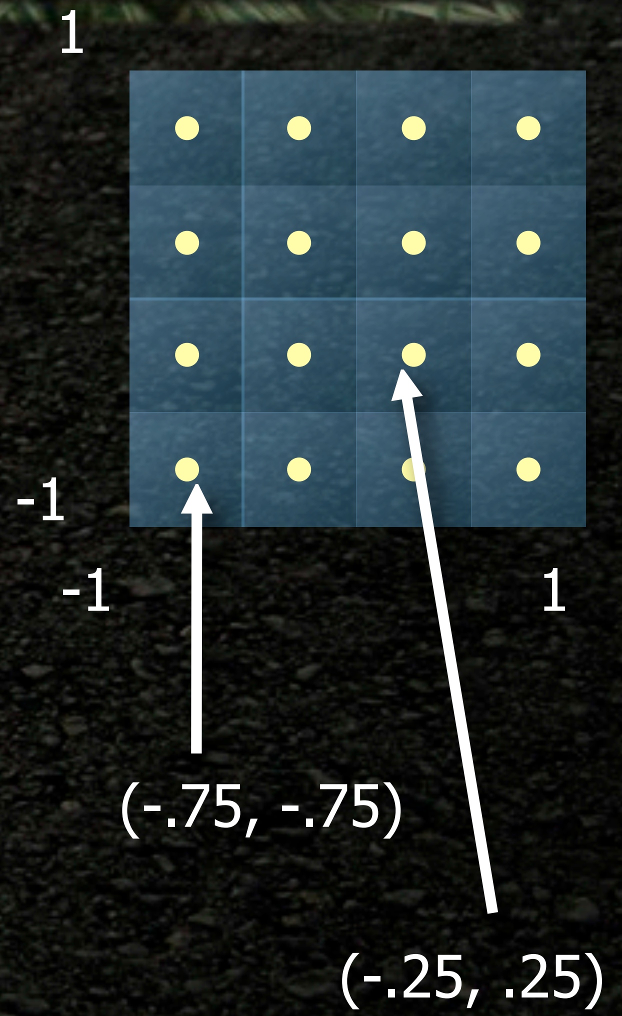
Camera models

- The camera maps pixels to rays
- What kind of camera models might we want?

Camera models

- Typical:
  - Orthographic
  - Pinhole (perspective)
- Advanced:
  - Depth of field (thin lens approximation)
  - Sophisticated lenses ("A realistic camera model for computer graphics," Kolh, Mitchell, Hanrahan)
  - Fish-eye lens
  - Arbitrary distortions

Camera models

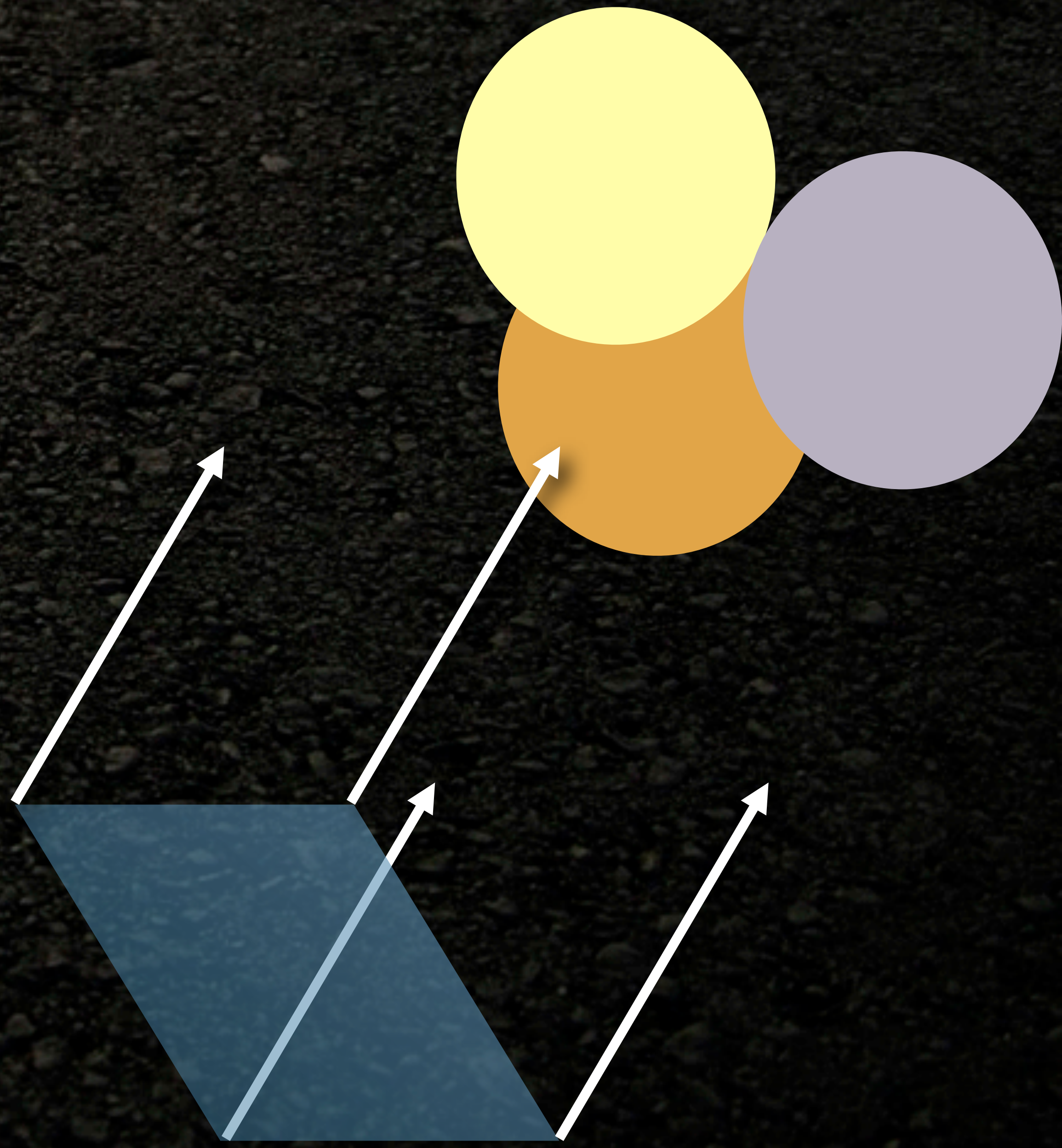


- Map pixel coordinates -1 to 1
- Pay careful attention to pixel centers

- Non-square images
  - Longest dimension is -1 to 1, shorter is smaller (still centered at 0)
  - Or camera knows about aspect ratio

Orthographic projection

- “Film” is just a rectangle in space
- Rays are parallel (same direction)



Orthographic projection

- Specify with center (P) and two vectors (u, v)

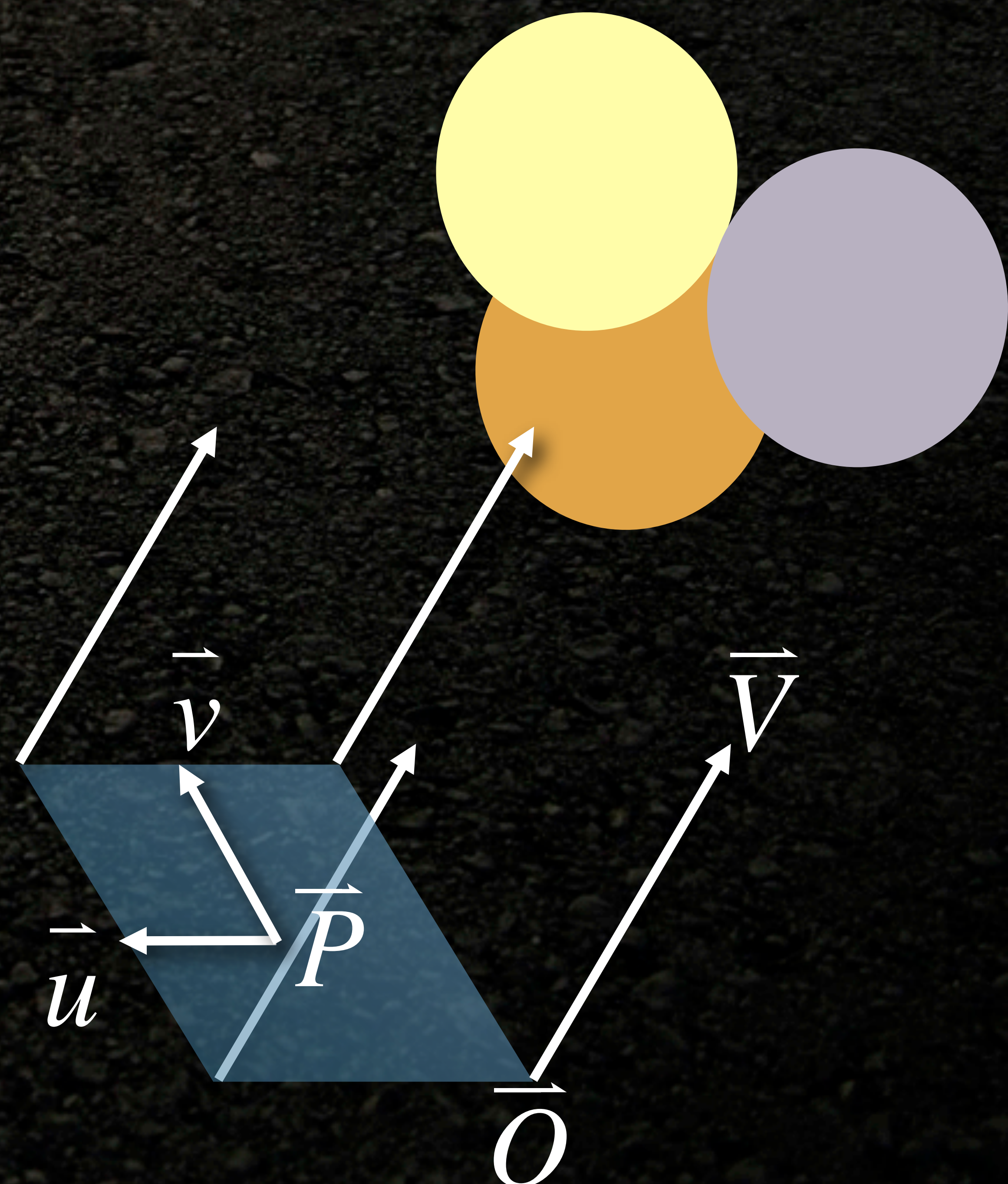
$$\vec{O} = \vec{P} + x\vec{u} + y\vec{v}$$

$$\vec{V} = \vec{u} \times \vec{v}$$

$$\|\vec{u}\|, \|\vec{v}\| : \text{ image size}$$

$$\frac{\|\vec{u}\|}{\|\vec{v}\|} = aspect\ ratio$$

$$\text{square image: } \vec{u} \cdot \vec{v} = 0$$

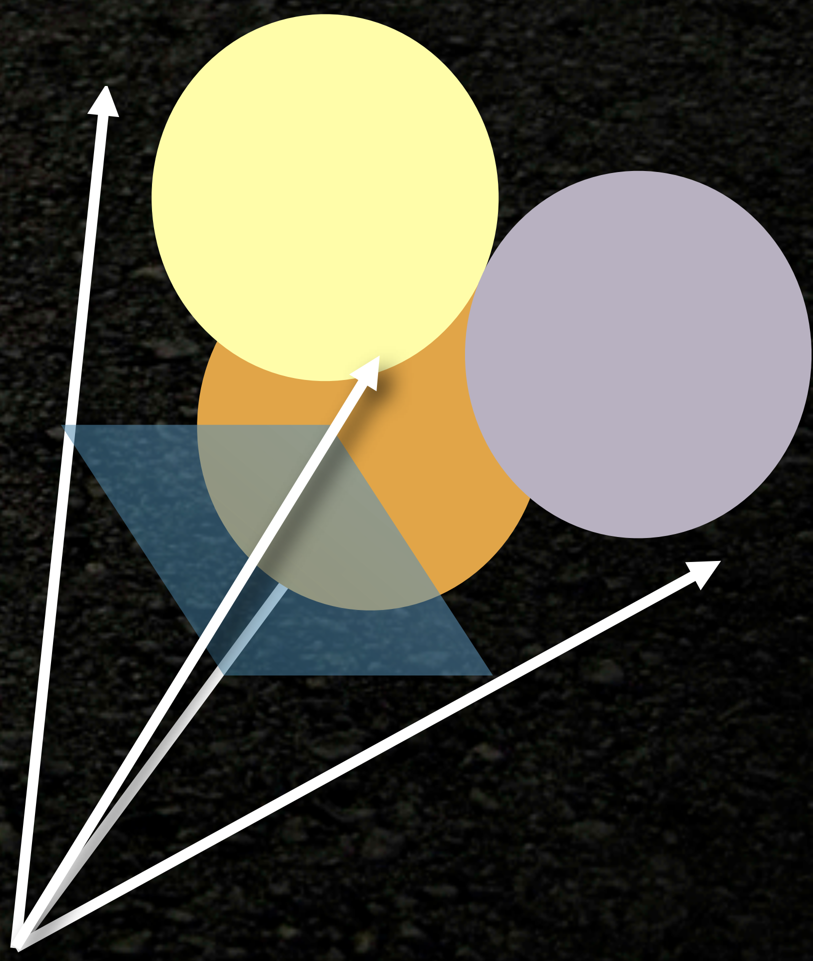


Pinhole camera

- Most common model for ray tracing
- Image is projected upside down onto image plane

Pinhole camera

- Easier to think about rightside up
- Focal point is also called the eye point



Pinhole camera

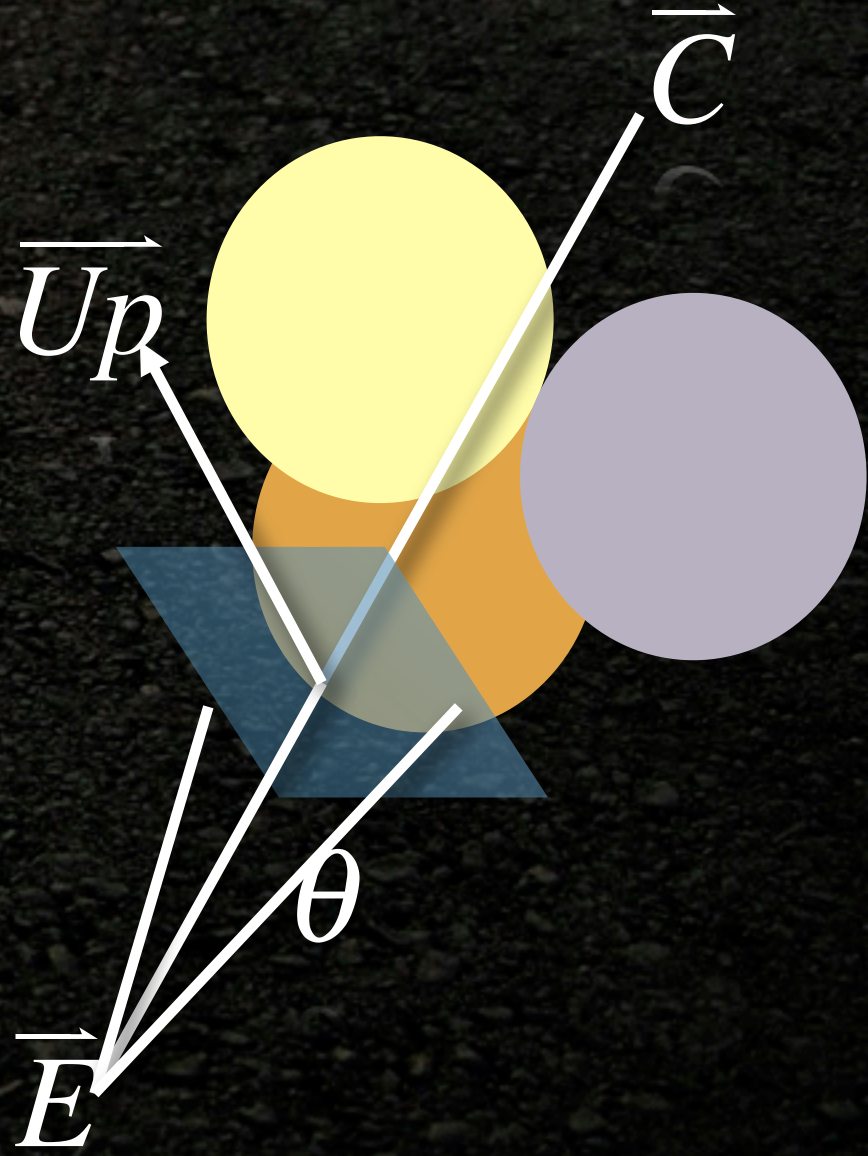
- Parameters:

$\vec{E}$: Eye point (focal point)

$\vec{C}$: Lookat point

$\overrightarrow{Up}$: Up vector

θ: Field of view



Pinhole camera

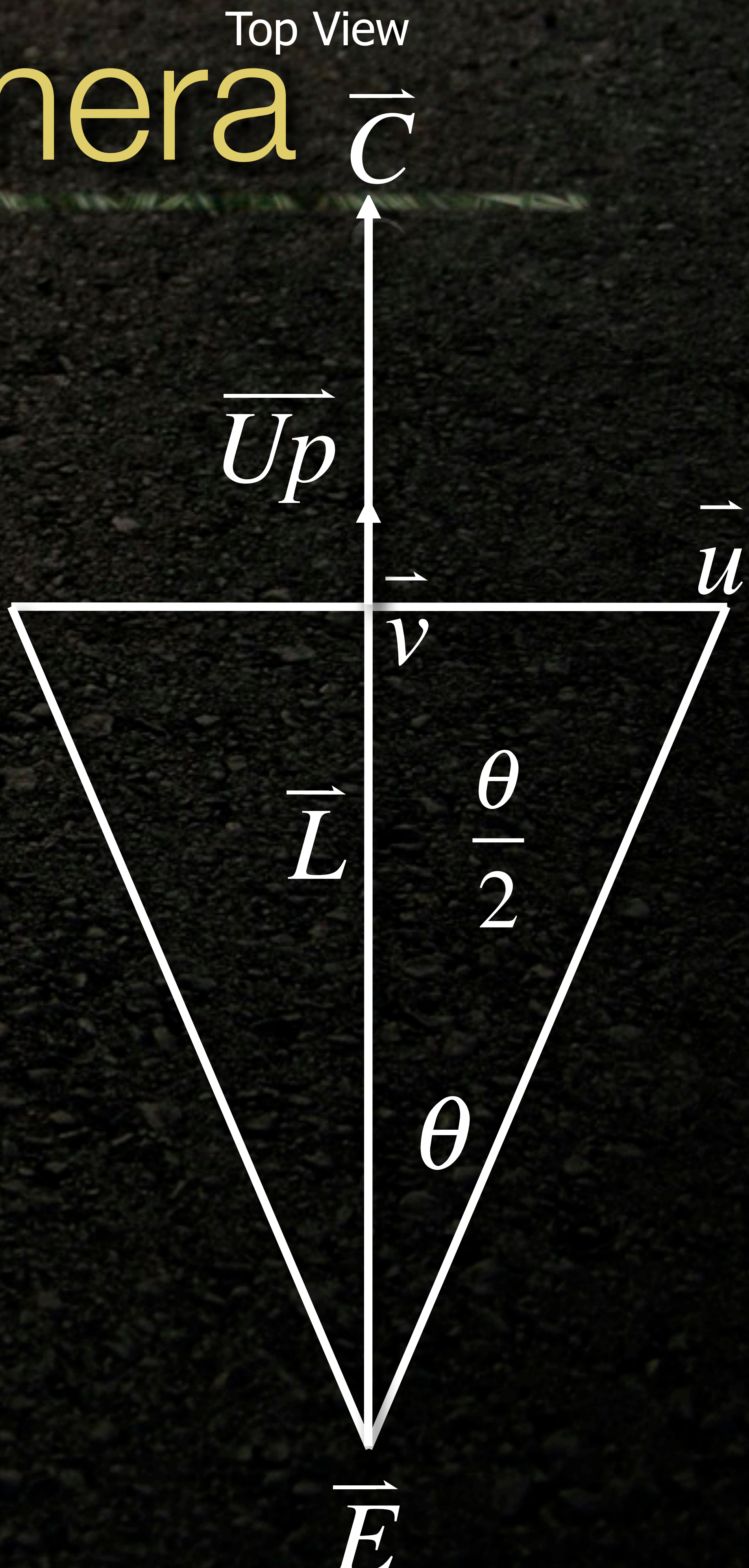
- Construction:

$$\vec{L} = \vec{C} - \vec{E} \quad \text{(look or gaze direction)}$$

$$\vec{L_n} = \frac{\vec{L}}{\|L\|}$$

$$\overrightarrow{u_{tmp}} = \vec{L_n} \times \overrightarrow{Up}$$

$$\overrightarrow{v_{tmp}} = \overrightarrow{u_{tmp}} \times \vec{L_n}$$



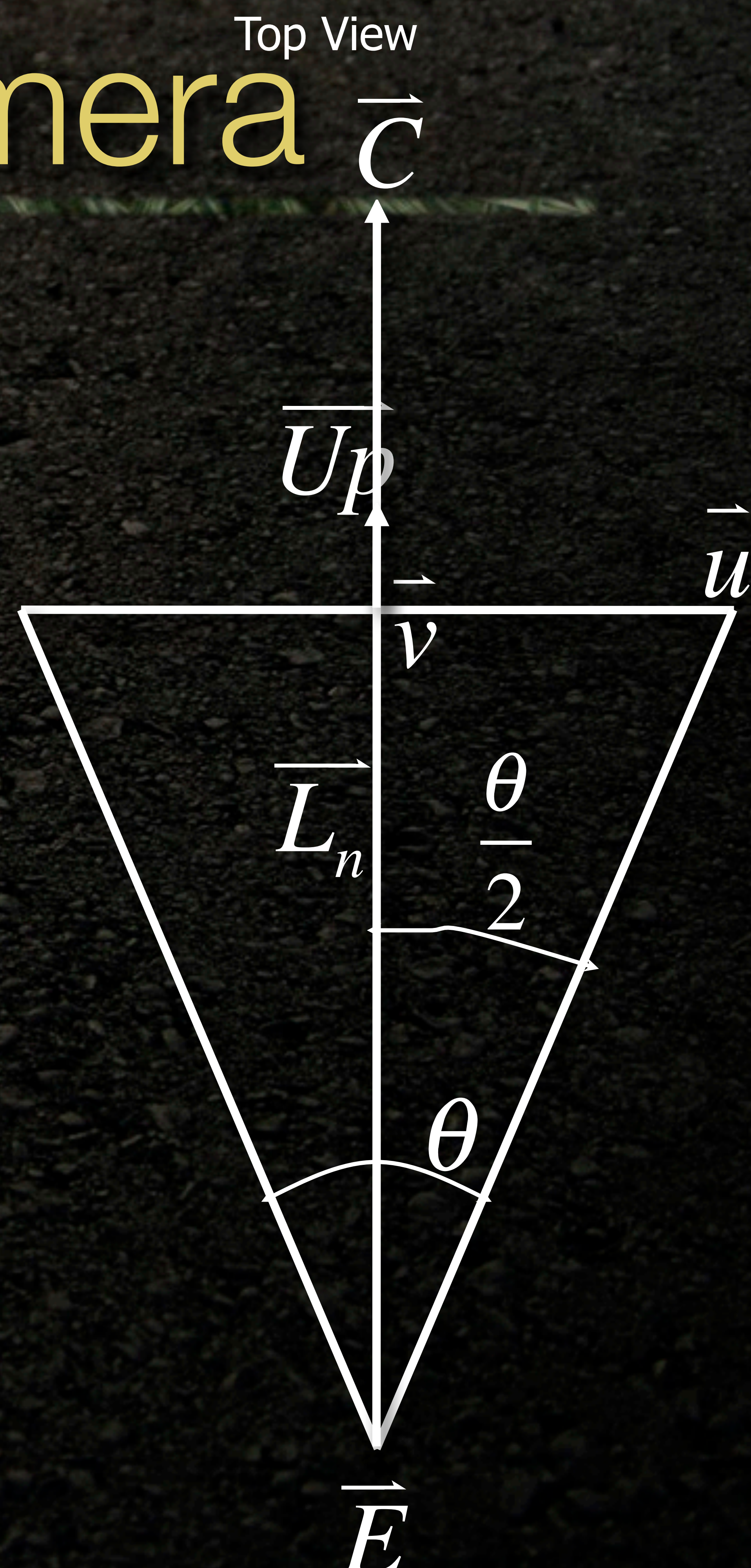
Pinhole camera

- How do we get the lengths of u/v?

$$\tan\frac{\theta}{2} = \frac{\|\vec{u}\|}{\|\vec{L_n}\|}$$

$$\|\vec{u}\| = \tan\frac{\theta}{2}$$

$$\vec{u} = \frac{\overrightarrow{u_{tmp}}}{\|\overrightarrow{u_{tmp}}\|}\tan\frac{\theta}{2}$$



Pinhole camera

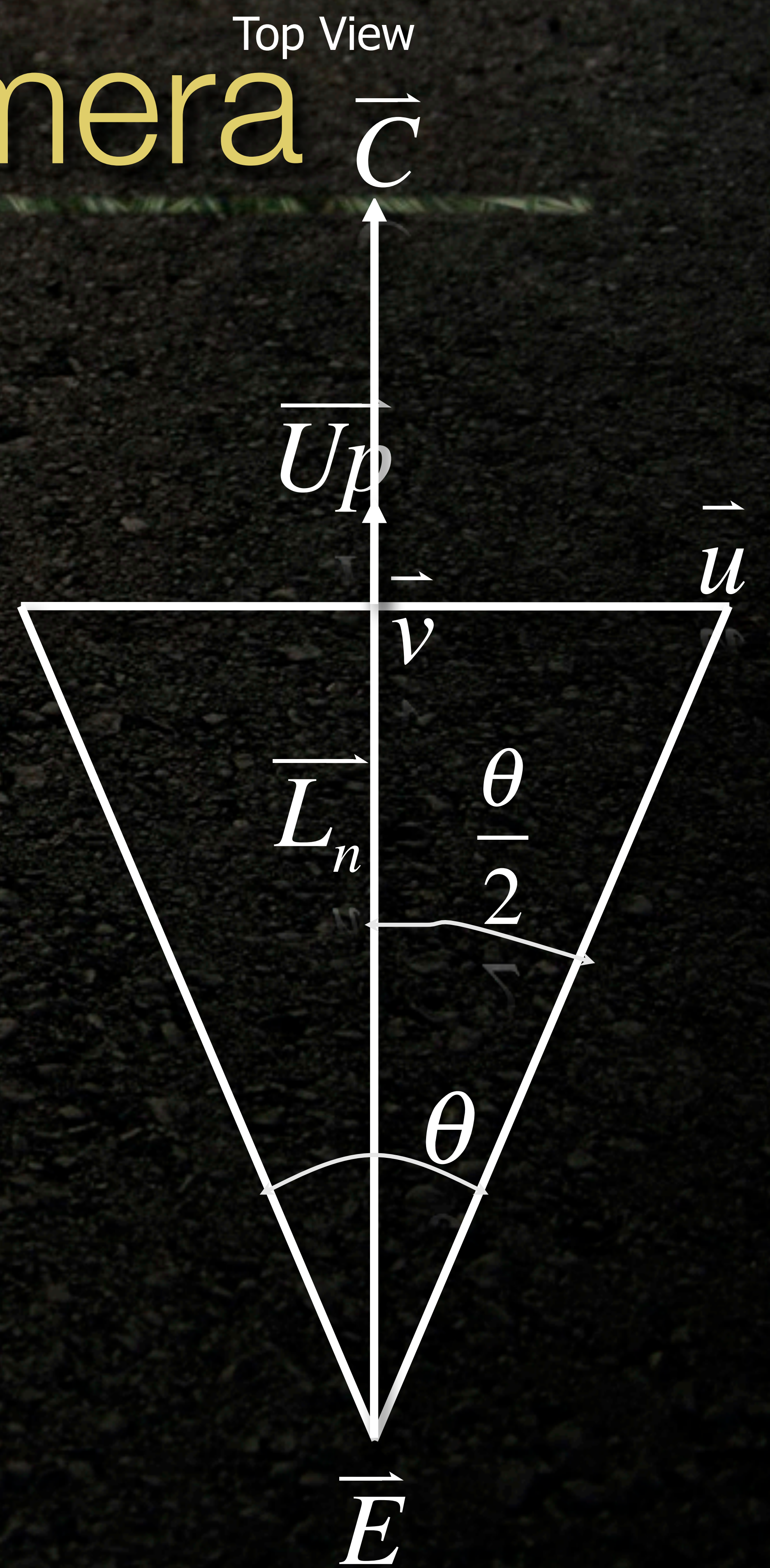
- What about v?

$$\text{aspect ratio} = a = \frac{\|\vec{u}\|}{\|\vec{v}\|}$$

$$\|\vec{u}\| = \tan\frac{\theta}{2}$$

$$\|\vec{v}\| = \frac{\tan\frac{\theta}{2}}{a}$$

$$\vec{v} = \frac{\overrightarrow{v_{tmp}}}{\|\overrightarrow{v_{tmp}}\|}\frac{\tan\frac{\theta}{2}}{a}$$



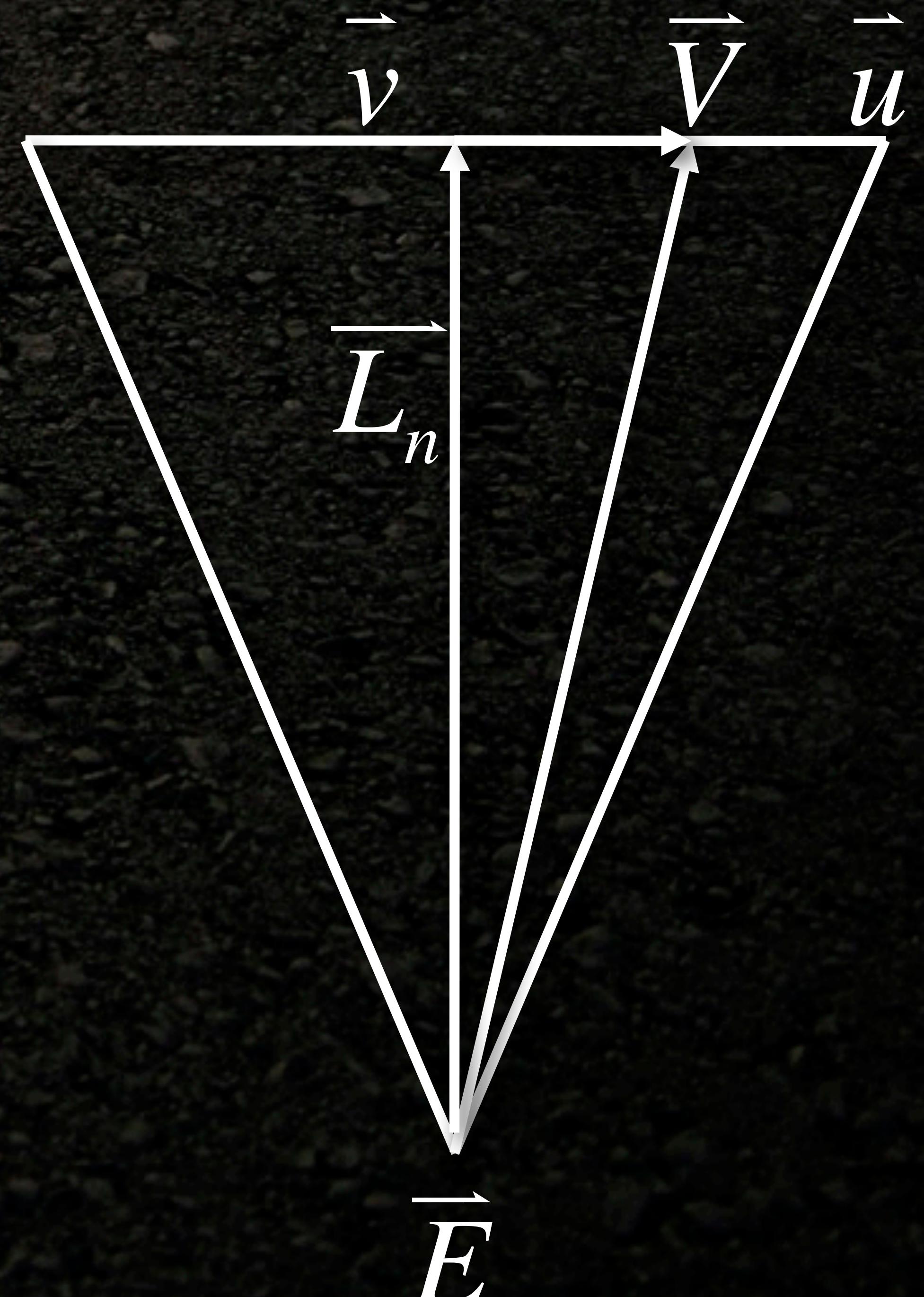
Pinhole camera

Top View

- Finally

$$\vec{O} = \vec{E}$$

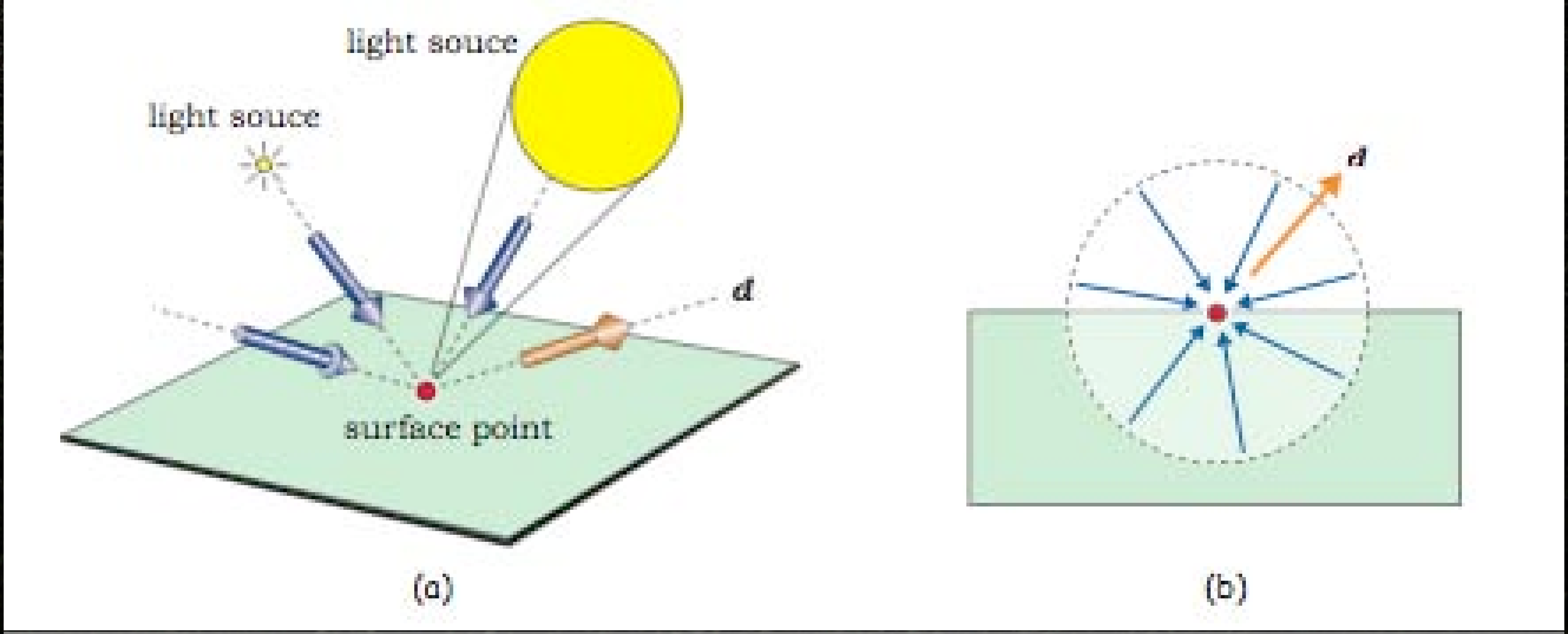
$$\vec{V} = \vec{L_n} + x\vec{u} + y\vec{v}$$



Intersection

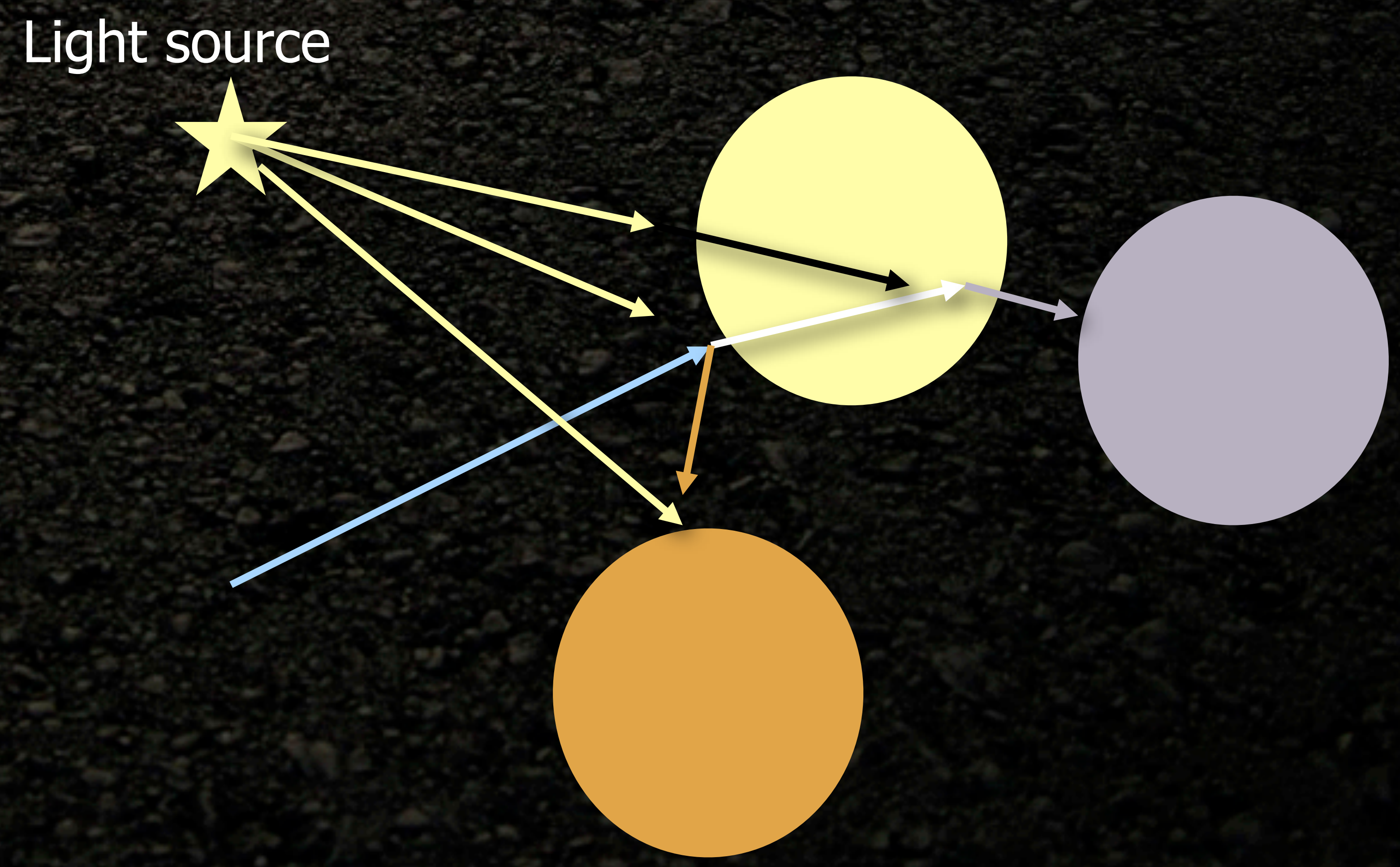
- Use the sphere algorithm from earlier
- Loop over spheres to find minimum t value

Shading



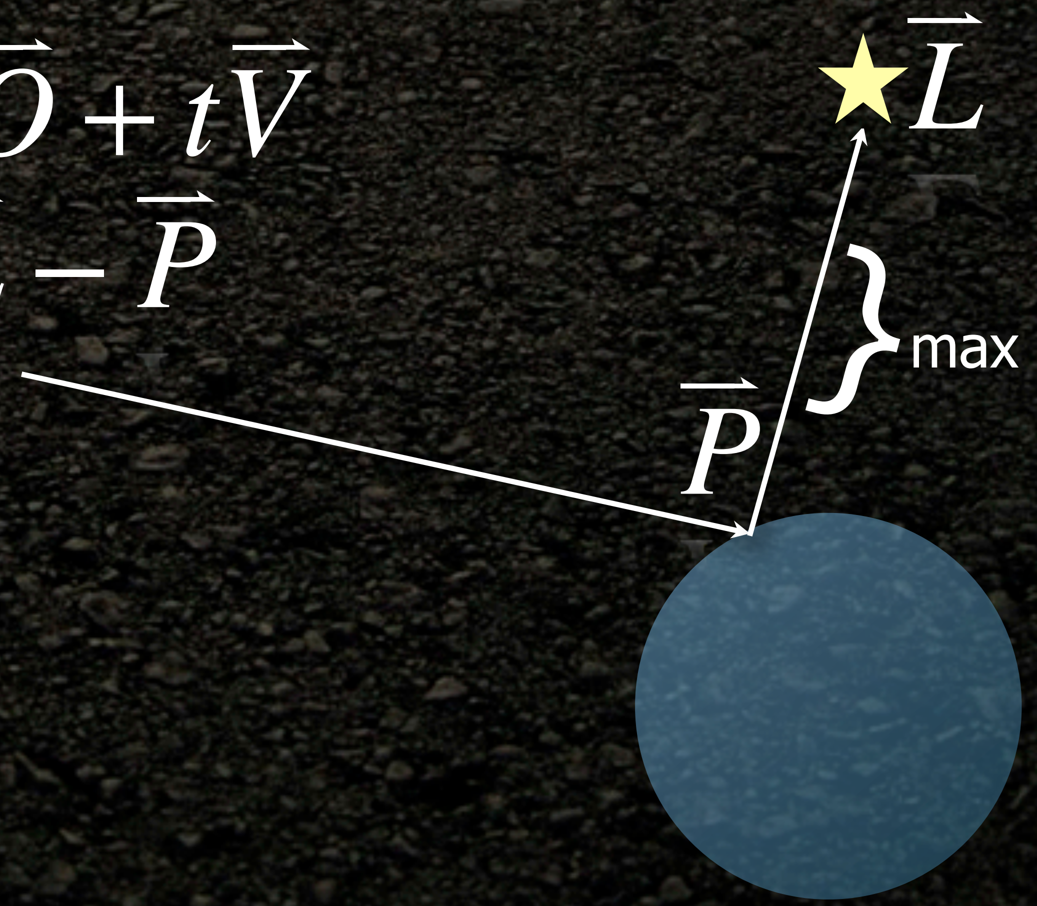
Shading

- Path tracing: consider light from all directions
- Whitted Ray tracing: consider the dominant directions:
  - direct (unobstructed from light source)
  - reflection
  - refraction



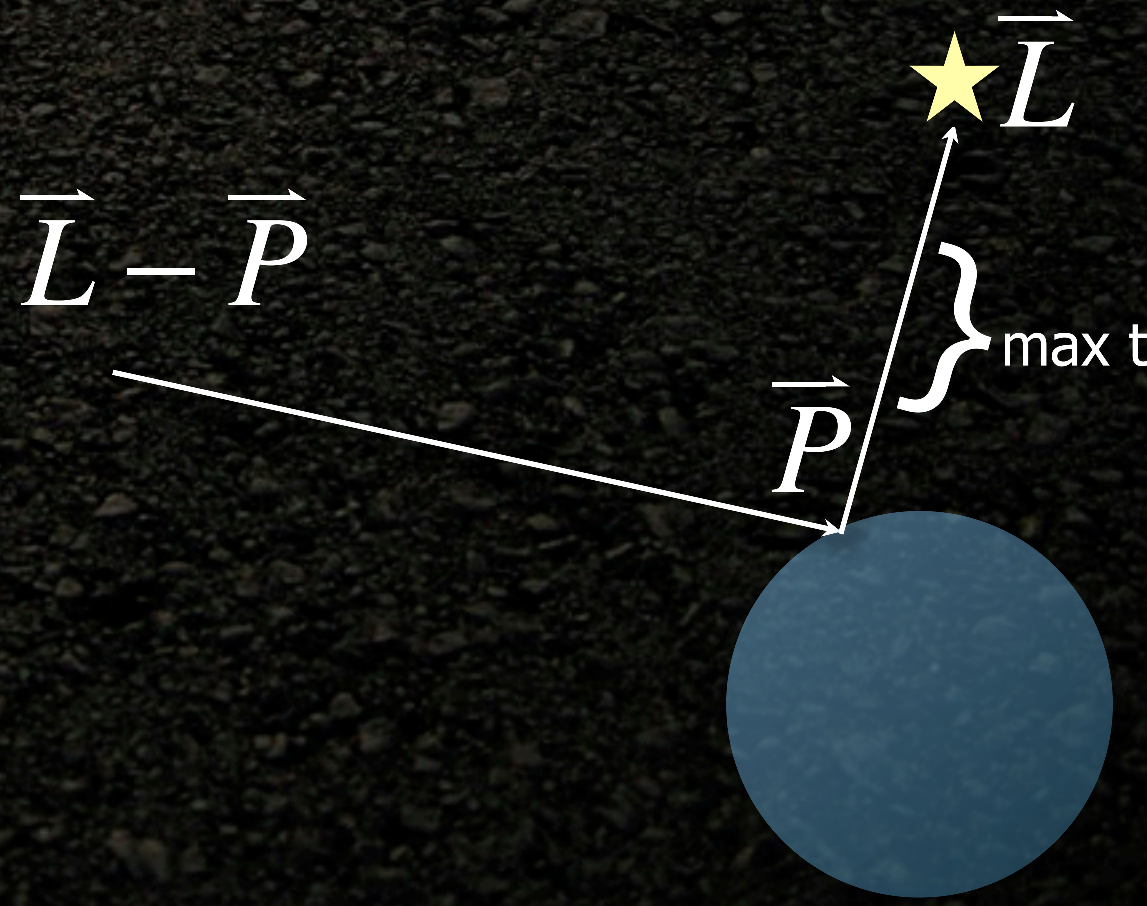
Shadow rays

- Shadows are computed by tracing rays from (to) the light source
- Intersection point:
- Origin: $\vec{P} = \vec{O} + t\vec{V}$
- Direction: $\vec{L} - \vec{P}$
- max t: 1.0



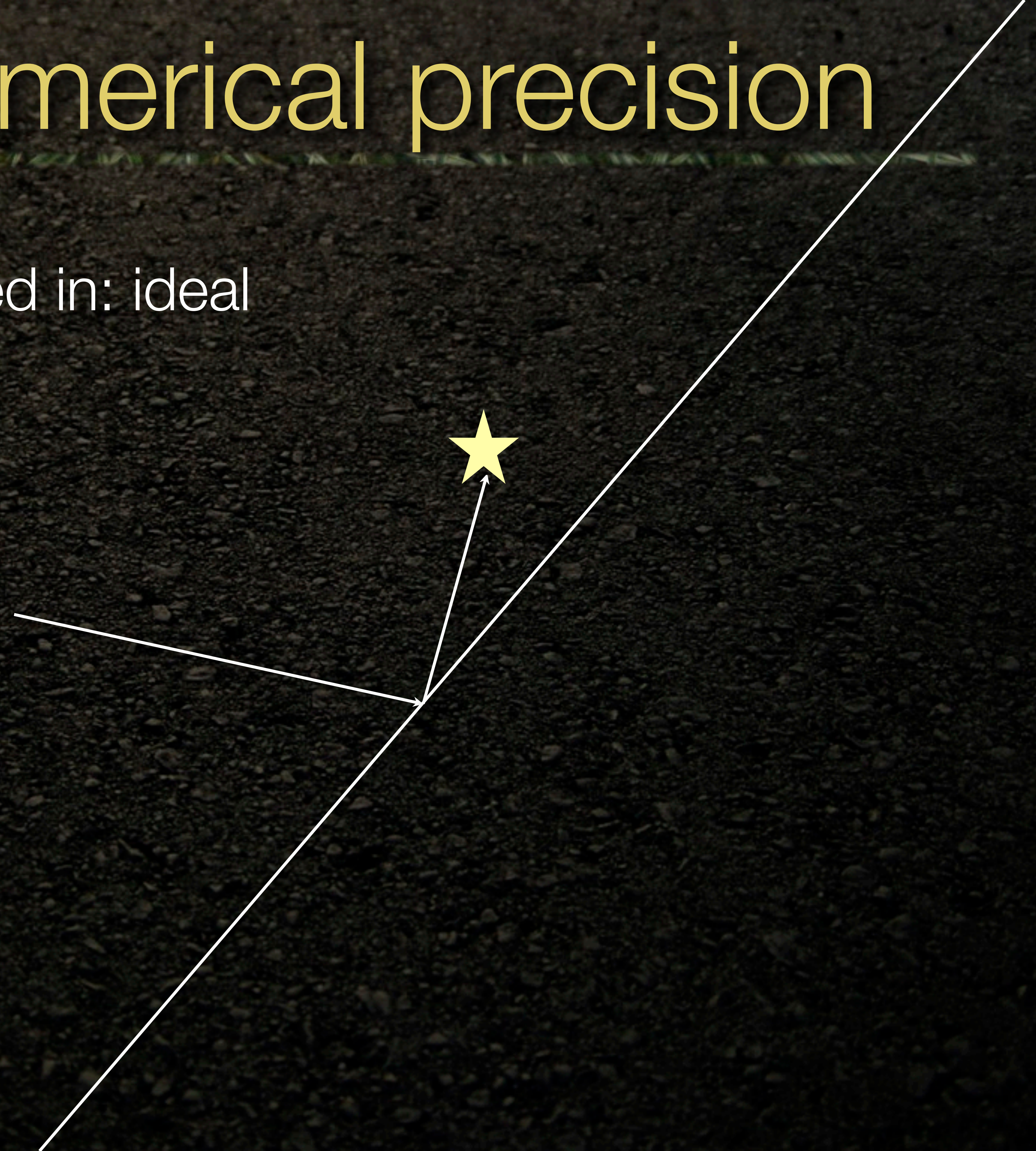
Shadow ray bugs

- Two bugs might show up when you do this:
  - False shadows (considering rays <0 or > 1)
  - Freckles (considering rays == 0)



Numerical precision

- Zoomed in: ideal



Numerical precision

- Zoomed in (numerical roundoff)

Numerical precision

- Zoomed in (numerical roundoff)

Solutions:

- Only consider intersections where t>small_num
- Offset ray in normal direction: P +=N*small_num
- Offset ray in light source direction: P+=(L-P)*small_num

- small_num = 1.e-6

Reflection

$$\vec{S} = \vec{V} + \vec{N}\cos\theta_i$$

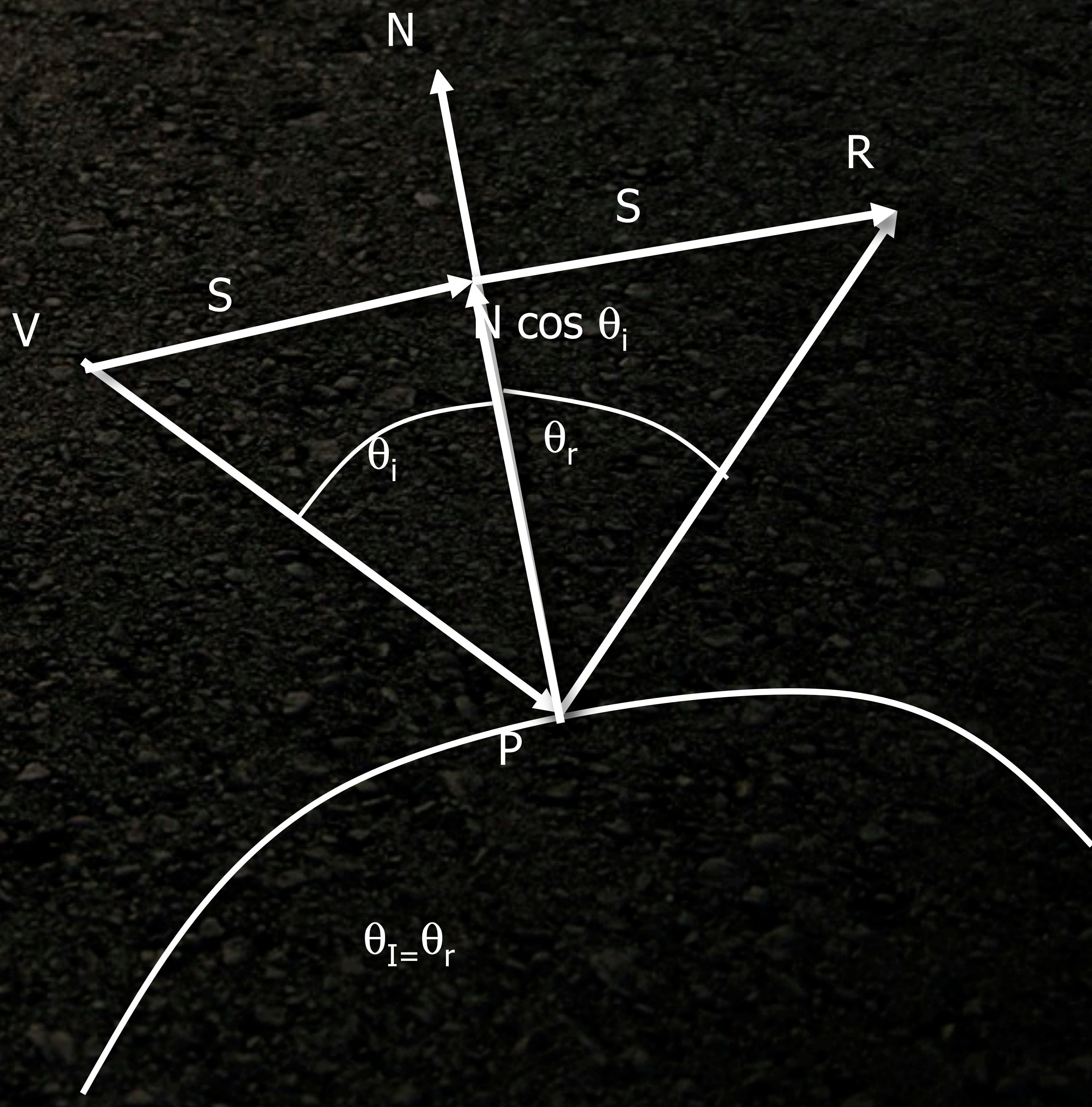
$$\vec{R} = \vec{N}\cos\theta_i + \vec{S}$$

$$\vec{R} = \vec{N}\cos\theta_i + \vec{V} + \vec{N}\cos\theta_i$$

$$\vec{R} = 2\vec{N}\cos\theta_i + \vec{V}$$

$$\vec{R} = 2\vec{N}\left(-\vec{N}\cdot\vec{V}\right) + \vec{V}$$

$$\vec{R} = \vec{V} - 2\left(\vec{N}\cdot\vec{V}\right)\vec{N}$$



Review

- The guts of a ray tracer have ray generation, ray intersection, and shading
- Most ray tracers use RGB color
- Now that ray tracers are interactive, the order in which you generate the rays may matter

Questions?