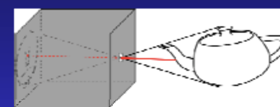
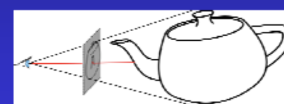
Ray Tracing Geometry

The Camera Model

- Based on a simpile pin-hole camera model
  - Simplest lens model
  - Pure geometric optics – based on similar triangles
  - Perfect image if hole infinitely small
  - Inverted image



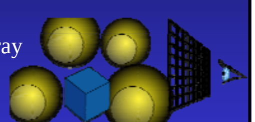
pin-hole camera



simplified pin-hole camera

Basic Ray Tracing Algorithm

```
for every pixel {
   cast a ray from the eye through pixel
   for every object in the scene
       find intersections with the ray
       keep it if closest
   }
   compute color at the intersection point
}
```



Construct a Ray

- 3D parametric line

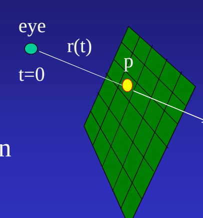
  r(t) = eye + t (p-eye)

  r(t): ray equation

  eye: eye (camera) position

  p: pixel position

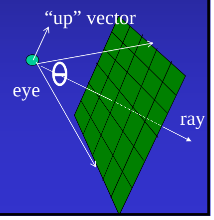
  t: ray parameter



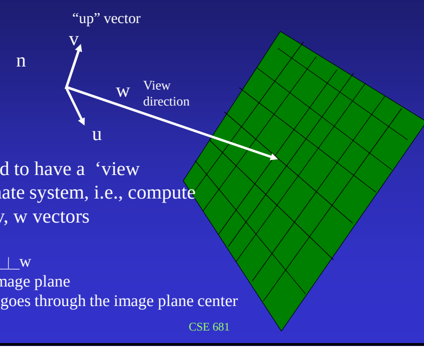
Question: How to calculate the pixel position P?

What are given?

- Camera (eye) position
- View direction or center of interest
- Camera orientation (which way is up?)
  - specified by an "up" vector
- Field of view + aspect ratio
- Distance to the image plane
- Pixel resolutions in x and y



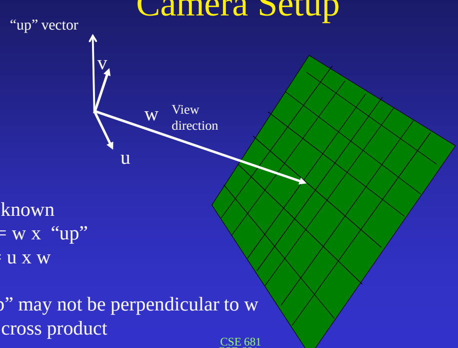
Camera Setup



We need to have a 'view coordinate system, i.e., compute the u, v, w vectors

u ⊥ v ⊥ w
w ⊥ image plane
Eye + w goes through the image plane center

Camera Setup



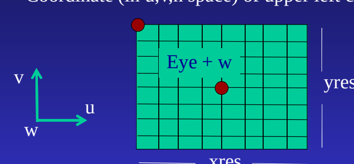
w: known
u = w x "up"
v = u x w

"up" may not be perpendicular to w
x: cross product

CSE 681

Pixel Calculation

Coordinate (in u,v,n space) of upper left corner of screen

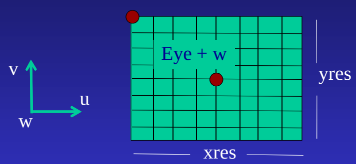


Assume virtual screen is one unit away (D=1) in w direction

Eye + w - (xres/2)*PixelWidth*u + (yres/2)*PixelHeight *v

Pixel Calculation

Coordinate (in u,v,n space) of upper left corner of screen



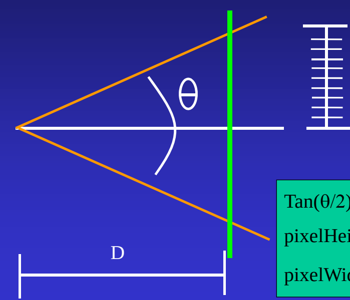
Assume virtual screen is one unit away (D=1) in w direction

How do we calculate PixelWidth and PixelHeight?

Eye + w - (xres/2)*PixelWidth*u + (yres/2)*PixelHeight *v

CSE 681

Camera Setup



$\text{Tan}(\theta/2) = \text{yres*pixelHeight}/2D$

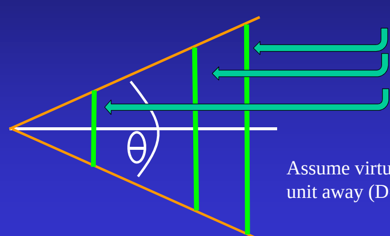
$\text{pixelHeight} = 2*\text{Tan}(\theta_y/2) *D/\text{yres}$

$\text{pixelWidth} = 2*\text{Tan}(\theta_x/2) *D/\text{xres}$

CSE 681

Screen Placement

How do images differ if the resolution doesn't change?



Assume virtual screen is one unit away (D=1) in w direction

CSE 681

Pixel Calculation

$\text{Tan}(\theta/2) = \text{yres*pixelHeight}/2$

$\text{pixelHeight} = 2*\text{Tan}(\theta_y/2) /\text{yres}$

$\text{pixelWidth} = 2*\text{Tan}(\theta_x/2) /\text{xres}$

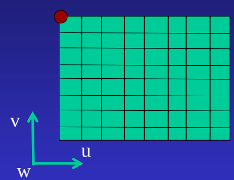
Pixel AspectRatio = pixelWidth/pixelHeight

Coordinate (in xyz space) of upper left corner of screen = ?

CSE 681

Pixel Calculation

Coordinate (in xyz space) of upper left corner of screen = ?



Tan(θ/2) = yres*pixelHeight/2

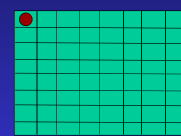
pixelHeight = 2*Tan(θ_y/2) /yres

pixelWidth = 2*Tan(θ_x/2) /xres

Eye + w - (xres/2)*PixelWidth*u + (yres/2)*PixelHeight *v

Pixel Calculation

Coordinate (in xyz space) of upper left pixel center = ?

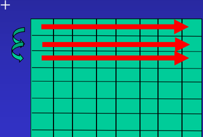


Eye + w - (xres/2)*PixelWidth*u + (yres/2)*PixelHeight *v

+ (pixelWidth/2)*u - (pixelHeight/2)*v

Interate through pixel Centers

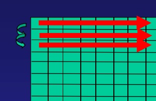
pixelCenter =
scanlineStart = Eye +
w -
(xres/2)*PixelWidth*u +
(yres/2)*PixelHeight *v +
(pixelWidth/2)*u -
(pixelHeight/2)*v



pixelCenter += pixelWidth * u

scanlineStart -= pixelHeight * v

Pixel loops



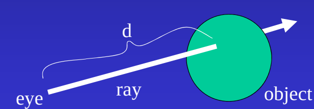
```
ScenlineStart = [from previous slide]
For each scanline {
   pixelCenter = scanlineStart
   For each pixel across {
      form ray from camera through pixel
      ….
      pixelCenter += pixelWidth*u
   }
   scanlineStart -= pixelHeight*v
}
```

Process Objects

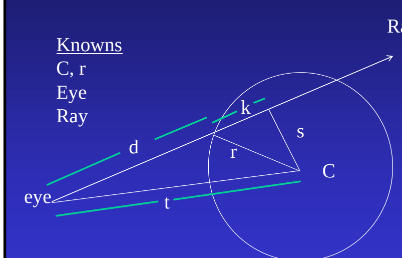
For each pixel {
    Form ray from eye through pixel
    $distance_{min}$ = infinity
    For each object {
        If (distance=intersect(ray,object)) {
            If (distance< $distance_{min}$) {
                closestObject = object
                $distance_{min}$ = distance
            }
        }
    }
    Color pixel according to intersection information
}

After all objects are tested

If ($distance_{min}$ > infinityThreshold) {
    pixelColor = background color
else
    pixelColor = color of object at $distance_{min}$ along ray



Ray-Sphere Intersection - geometric



t= |C-eye|

d+k = (C-eye) · Ray

$t^2 = (k+d)^2 + s^2$

$r^2 = k^2 + s^2$

d = (k+d) - k

Ray-Sphere Intersection - algebraic

$x^2 + y^2 + z^2 = r^2$

P(t) = eye + t*Ray

Substitute definition of p into first equation:

$(eye.x+ t *ray.x)^2 + (eye.y+ t *ray.y)^2 + (eye.z+ t *ray.z)^2 = r^2$

Expand squared terms and collect terms based on powers of u:

$A* t^2 + B* t + C = 0$

Ray-Sphere Intersection (cont'd)

For a sphere with its center at c

A sphere with center c = (xc,yc,zc) and radius R can be represented as:

$(x-xc)^2 + (y-yc)^2 + (z-zc)^2 – R^2 = 0$

For a point p on the sphere, we can write the above in vector form:

$(p-c).(p-c) – R^2 = 0$ (note '.' is a dot product)

Solve p similarly

Quadratic Equation

When solving a quadratic equation

$at^2 + bt + c = 0$

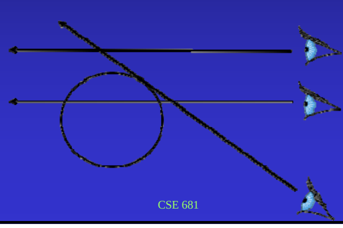
Discriminant: $d = \sqrt{b^2 - 4ac}$

And Solution: $t_{\pm} = \frac{-b \pm d}{2a}$

Ray-Sphere Intersection

$d = \sqrt{b^2 - 4ac}$

$b^2 – 4ac < 0$: No intersection
$b^2 – 4ac > 0$: Two solutions (enter and exit)
$b^2 – 4ac = 0$: One solution (ray grazes the sphere)



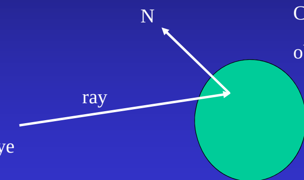
Determine Color

FOR LAB #1

Use z-component of normalized normal vector

Clamp to [0.3..1.0]

objectColor*N_z



What's the normal at a point on the sphere?