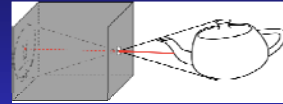


Ray Tracing Geometry

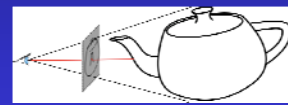
CSE 681

The Camera Model

- Based on a simple 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

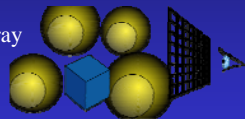


simplified pin-hole camera

CSE 681

Basic Ray Tracing Algorithm

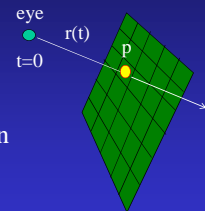
```
for every pixel {  
  cast a ray from the eye  
  for every object in the scene  
    find intersections with the ray  
    keep it if closest  
  }  
  compute color at the intersection point  
}
```



CSE 681

Construct a Ray

- 3D parametric line
 - $r(t) = \text{eye} + t(\text{p} - \text{eye})$
 - $r(t)$: ray equation
 - eye: eye (camera) position
 - p: pixel position
 - t: ray parameter

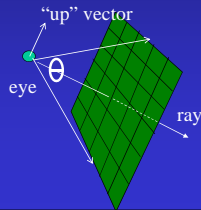


Question: How to calculate the pixel position P?

CSE 681

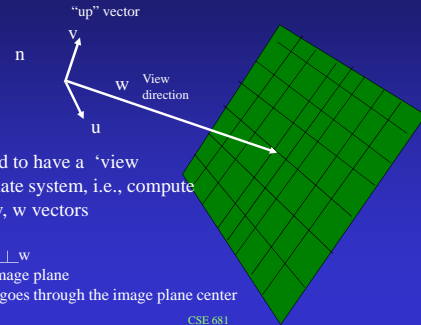
What are given?

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



CSE 681

Camera Setup

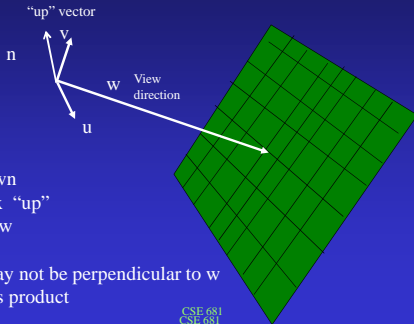


We need to have a 'view coordinate system, i.e., compute the u, v, w vectors

$u \perp v \perp w$
 $w \perp$ image plane
 Eye + w goes through the image plane center

CSE 681

Camera Setup



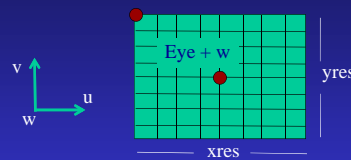
w: known
 $u = w \times \text{“up”}$
 $v = u \times w$

“up” may not be perpendicular to w
 \times : cross product

CSE 681

Pixel Calculation

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



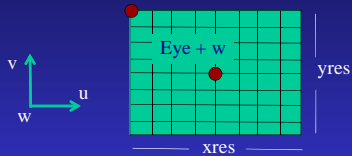
Assume virtual screen is one unit away ($D=1$) in w direction

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

CSE 681

Pixel Calculation

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



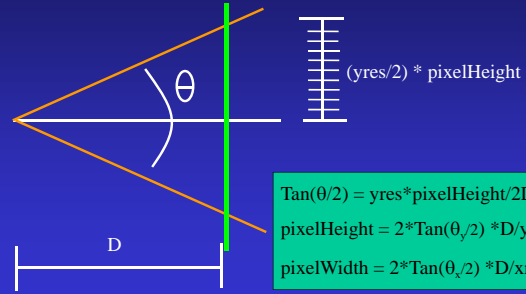
Assume virtual screen is one unit away ($D=1$) in w direction

How do we calculate PixelWidth and PixelHeight?

$$\text{Eye} + w - (xres/2) * \text{PixelWidth} * u + (yres/2) * \text{PixelHeight} * v$$

CSE 681

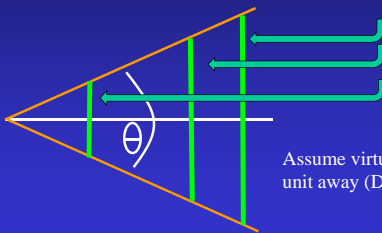
Camera Setup



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

$$\begin{aligned} \text{Tan}(\theta/2) &= yres * \text{pixelHeight} / 2 \\ \text{pixelHeight} &= 2 * \text{Tan}(\theta/2) / yres \\ \text{pixelWidth} &= 2 * \text{Tan}(\theta_x/2) / xres \end{aligned}$$

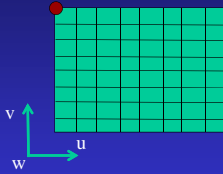
$$\text{Pixel AspectRatio} = \text{pixelWidth} / \text{pixelHeight}$$

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

CSE 681

Pixel Calculation

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



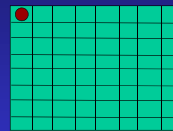
$$\begin{aligned} \tan(\theta/2) &= \text{yres} * \text{pixelHeight} / 2 \\ \text{pixelHeight} &= 2 * \tan(\theta/2) / \text{yres} \\ \text{pixelWidth} &= 2 * \tan(\theta/2) / \text{xres} \end{aligned}$$

$$\text{Eye} + w - (\text{xres}/2) * \text{PixelWidth} * u + (\text{yres}/2) * \text{PixelHeight} * v$$

CSE 681

Pixel Calculation

Coordinate (in u,v,n space) of upper left pixel center = ?

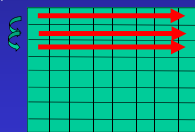


$$\begin{aligned} \text{Eye} + w - (\text{xres}/2) * \text{PixelWidth} * u + (\text{yres}/2) * \text{PixelHeight} * v \\ + (\text{pixelWidth}/2) * u - (\text{pixelHeight}/2) * v \end{aligned}$$

CSE 681

Iterate through pixel Centers

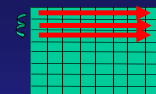
```
pixelCenter =
scanlineStart = Eye +
    w -
    (xres/2)*PixelWidth*u +
    (yres/2)*PixelHeight*v +
    (pixelWidth/2)*u -
    (pixelHeight/2)*v
```



```
pixelCenter += pixelWidth * u
scanlineStart -= pixelHeight * v
```

CSE 681

Pixel loops



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

CSE 681

Process Objects

```

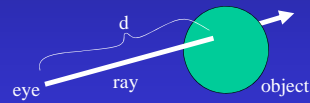
For each pixel {
  Form ray from eye through pixel
  distancemin = infinity
  For each object {
    If (distance=intersect(ray,object)) {
      If (distance < distancemin) {
        closestObject = object
        distancemin = distance
      }
    }
  }
  Color pixel according to intersection information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

CSE 681

After all objects are tested

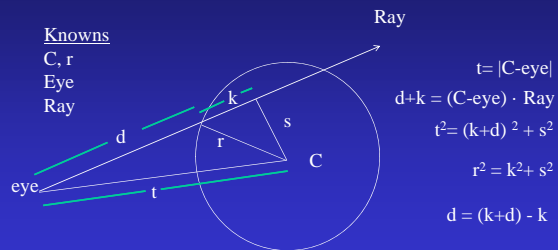
```

If (distancemin > infinityThreshold) {
  pixelColor = background color
} else {
  pixelColor = color of object at distancemin along ray
}
    
```



CSE 681

Ray-Sphere Intersection - geometric



CSE 681

Ray-Sphere Intersection - algebraic

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

$$P(t) = \text{eye} + t * \text{Ray}$$

Substitute definition of p into first equation:

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

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

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

CSE 681

Ray-Sphere Intersection (cont'd)

For a sphere with its center at c

A sphere with center $c = (x_c, y_c, z_c)$ and radius R can be represented as:

$$(x-x_c)^2 + (y-y_c)^2 + (z-z_c)^2 - R^2 = 0$$

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

$$(p-c) \cdot (p-c) - R^2 = 0 \quad (\text{note '.' is a dot product})$$

Solve p similarly

CSE 681

Quadratic Equation

When solving a quadratic equation

$$at^2 + bt + c = 0$$

Discriminant:

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

And Solution:

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

CSE 681

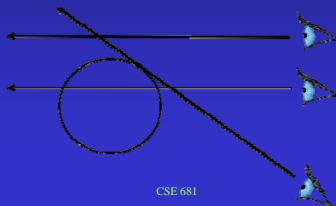
Ray-Sphere Intersection

$b^2 - 4ac < 0$: No intersection

$b^2 - 4ac > 0$: Two solutions (enter and exit)

$b^2 - 4ac = 0$: One solution (ray grazes the sphere)

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



CSE 681

Determine Color

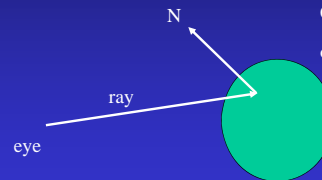
FOR LAB #1

Use z-component of normalized normal vector

Clamp to [0.3..1.0]

$\text{objectColor} * N_z$

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



CSE 681