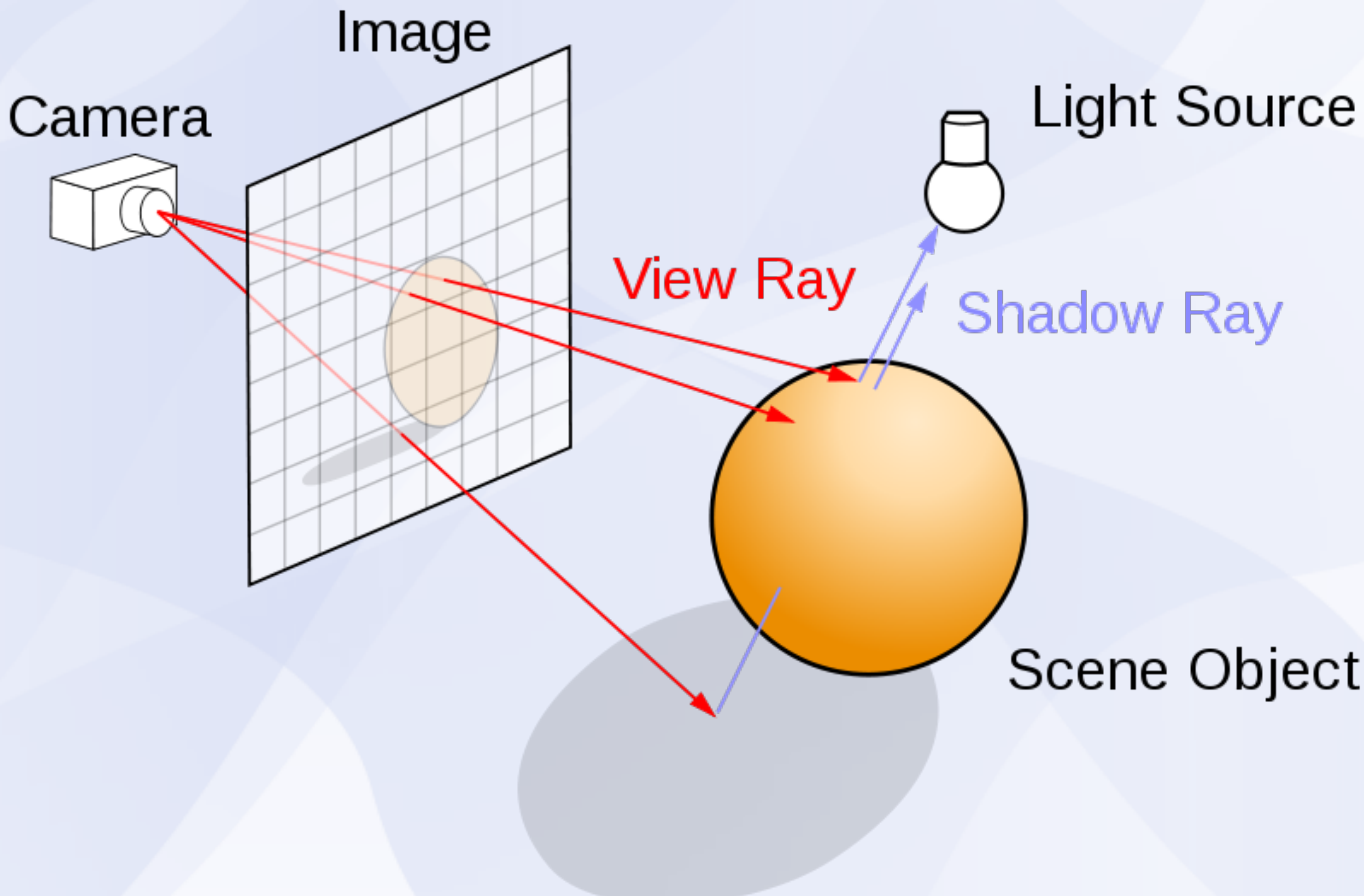
CS420 Assignment 3 Hints

Ray Tracing

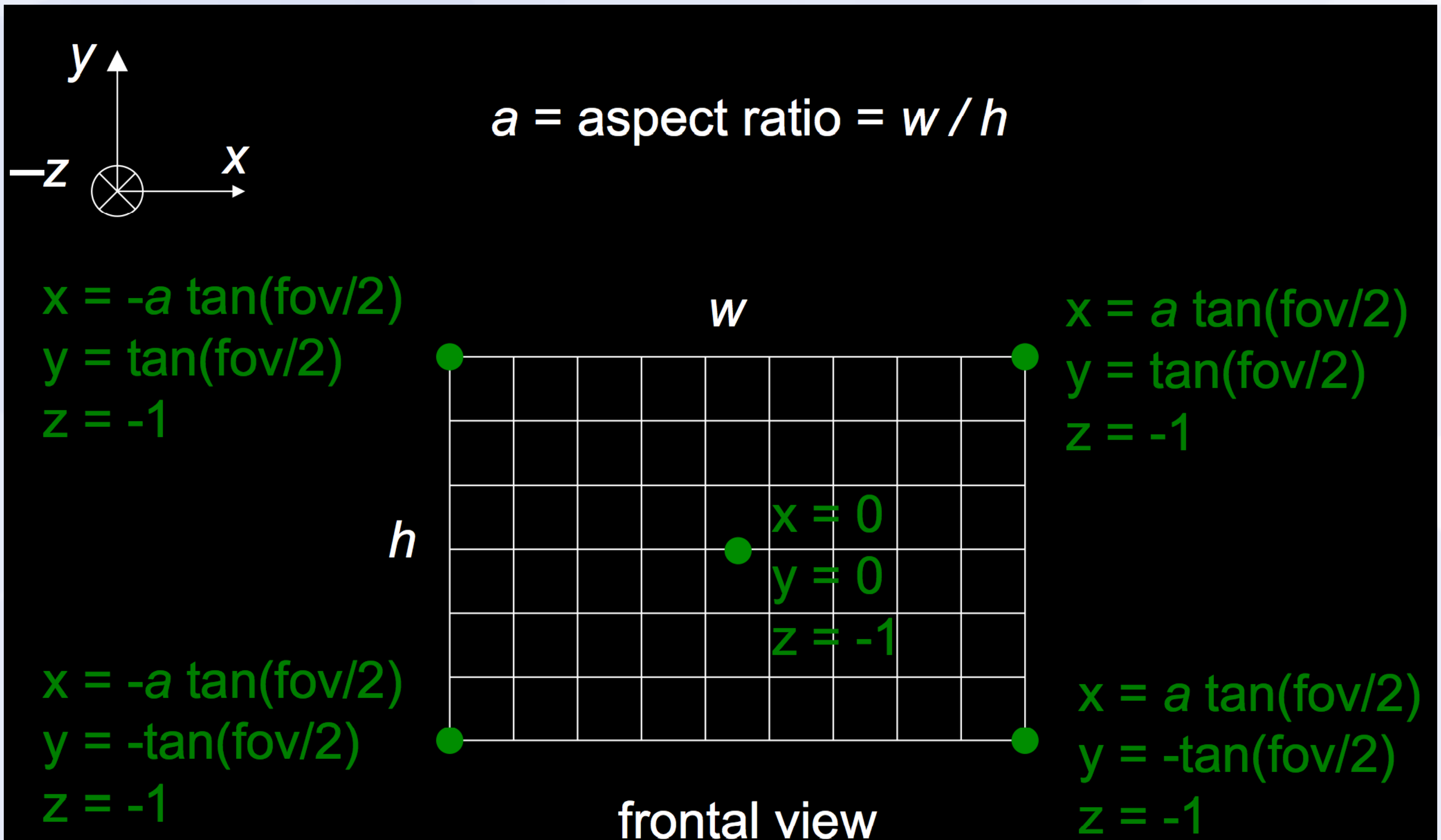
Step 1: send rays



- Send out rays from camera position (0,0,0) pointing to −z
- Image size 640x480
  - For debugging, use smaller size

- Send out rays from camera position (0,0,0) pointing to −z
- Image size 640x480
    - For debugging, use smaller size

fov: 60 degrees



Step 2: Intersect with scene

- Sphere & triangle
- Analytical solution

Sphere: Analytical Solution

- Sphere equation:
  - $f(\mathbf{q}) = (x - x_c)^2 + (y - y_c)^2 + (z - z_c)^2 - r^2 = 0$
- Ray: $x = x_0 + x_d t, \quad y = y_0 + y_d t, \quad z = z_0 + z_d t$
- Produce:

$$(x_0 + x_d t - x_c)^2 + (y_0 + y_d t - y_c)^2 + (z_0 + z_d t - z_c)^2 = r^2$$

- Simplify to: $at^2 + bt + c = 0$
- $a = x_d^2 + y_d^2 + z_d^2 = 1$
- $b = 2[x_d(x_0 - x_c) + y_d(y_0 - y_c) + z_d(z_0 - z_c)]$
- $c = (x_0 - x_c)^2 + (y_0 - y_c)^2 + (z_0 - z_c)^2 - r^2$

Possible Optimization: precompute *c* and a part of *b* for one start point

- Get t: $$t_{0,1} = \frac{-b \pm \sqrt{b^2 - 4c}}{2}$$
- Calculate $b^2 - 4c$, abort if negative
- Return minimum positive t

Triangle: Intersection

1. find intersection of the ray and the plane which the triangle lies on.
2. determine the ray-plane intersection point is in/out of the triangle in the 2D plane.

Triangle: Analytical Solution

- Plane equation:
  - Implicit form: $ax + by + cz + d = 0$
  - Unit normal: $\mathbf{n} = [a \;\; b \;\; c]^T$ with $a^2 + b^2 + c^2 = 1$
- For triangle ABC,
  - normal direction: $n = \text{normalize}(AB \times AC)$
  - A has coord: (x_a, y_a, z_a)
  - Because A is on the plane:
    - $d = -(ax_a - by_a - cz_a)$

- Ray: $x = x_0 + x_d t, \qquad y = y_0 + y_d t, \qquad z = z_0 + z_d t$
- So: $a(x_0 + x_d t) + b(y_0 + y_d t) + c(z_0 + z_d t) + d = 0$

$$t = \frac{-(ax_0 + by_0 + cz_0 + d)}{ax_d + by_d + cz_d}$$

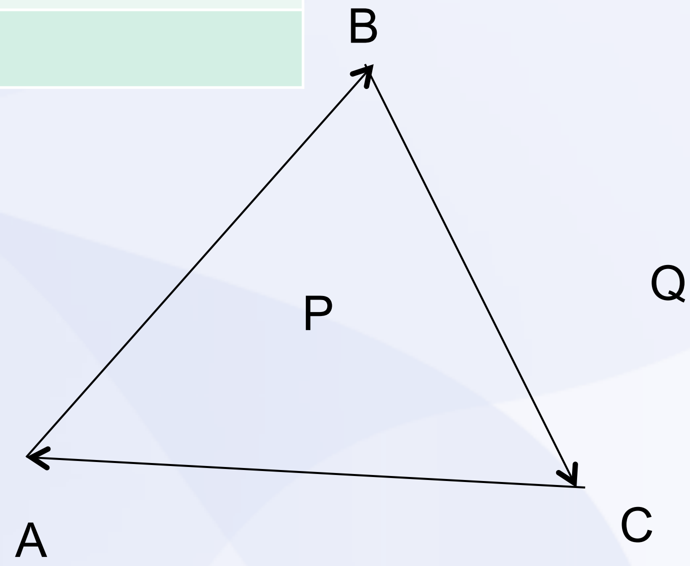
Possible Optimization:
precompute *normal*
and *d*
and *numerator* for one
start point

- abort if $ax_d + by_d + cz_d == 0$

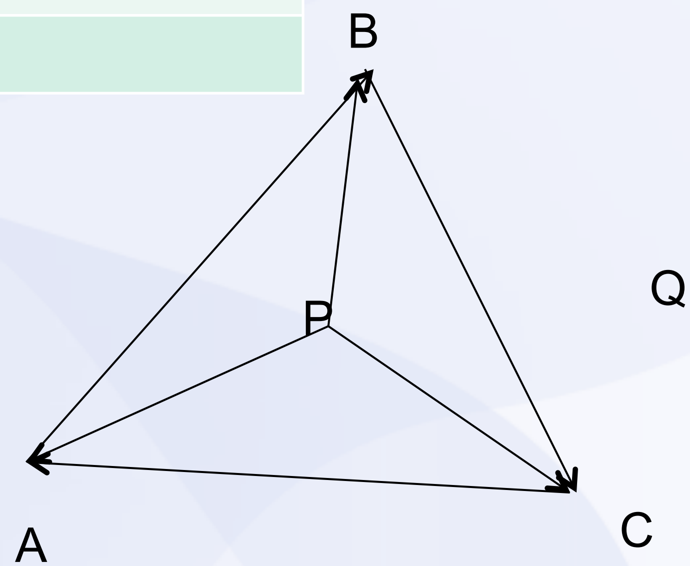
In/Out Test for Triangle

- determine intersection point *p* in/out of triangle ABC
- project to 2D
  - e.g. if n = (a,b,c), |a| > |b| && |a| > |c| (|*a*| is biggest)
  - project to the plane x = 0

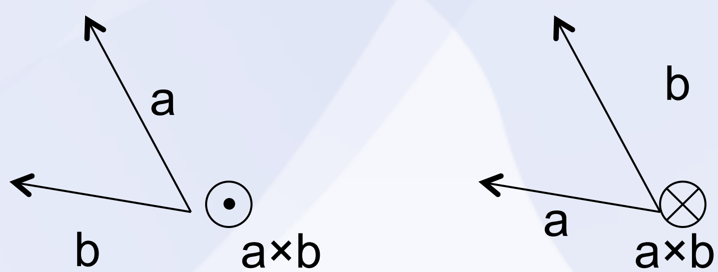
Directed Edge	Side which P lies	Side which Q lies
AB	right	right
BC	right	left
CA	right	right



DirEdge XY	PX×PY	QX×QY
AB	in	in
BC	in	out
CA	in	in



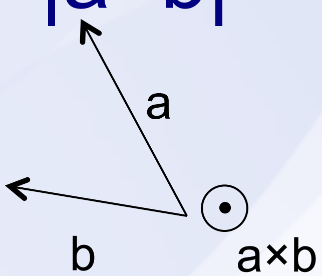
- $|a \times b| = |a||b|\sin(\theta)$

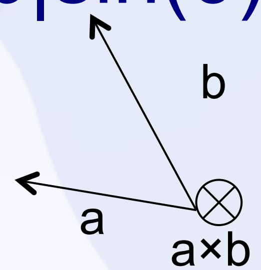


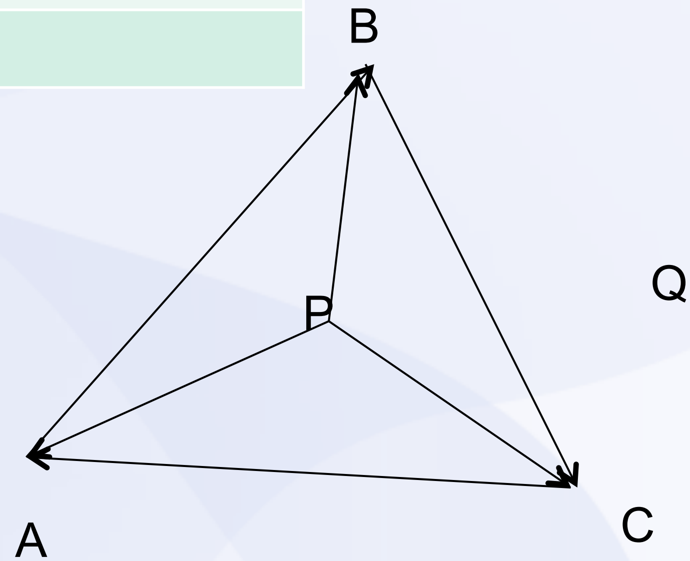
Cross Product

DirEdge XY	SignedArea(PXY)	SingedArea(QXY)
AB	-	-
BC	-	+
CA	-	-

- Area sign:
  - clockwise (-)
  - anti-clockwise (+)
- $|a \times b| = |a||b|\sin(\theta)$





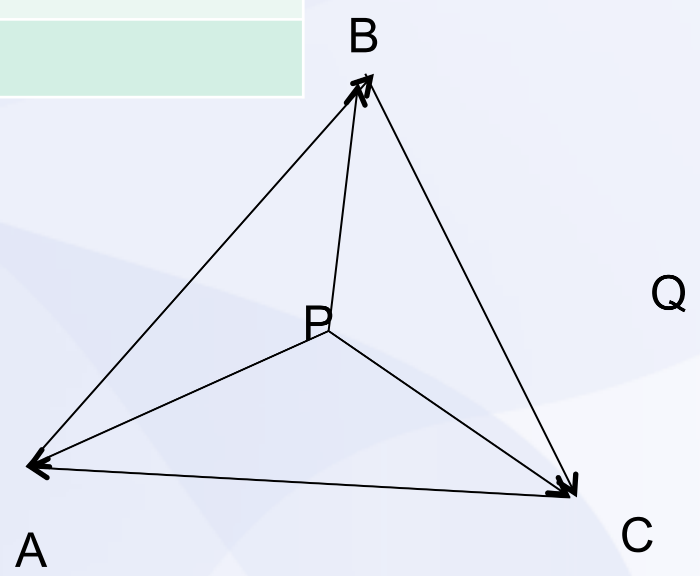


$$|S_{PAB}| = |PA||PB|\sin(\angle APB) / 2$$
$$= |PA \times PB| / 2$$

Cross Product

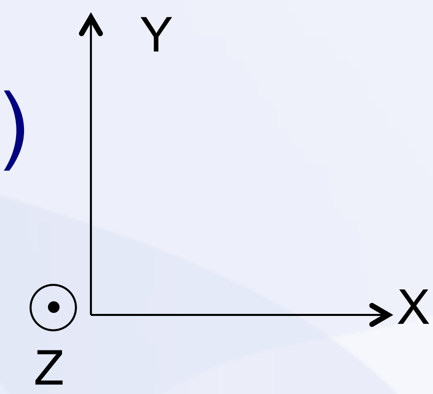
DirEdge XY	P's BaryCen. on Z	Q's BaryCe. on Z
AB	-	-
BC	-	+
CA	-	-

- Barycentric coord.
  - $P = \alpha A + \beta B + \gamma C$
  - $\alpha+\beta+\gamma = 1$
  - $\alpha:\beta:\gamma = S_{PBC}:S_{PCA}:S_{PAB}$
- $|a\times b| = |a||b|\sin(\theta)$



$|S_{PAB}| = |PA||PB|\sin(\angle APB) / 2$
$= |PA \times PB| / 2$

- Compute PA×PB, PB×PC, PC×PA
  - They can be scaled to barycentric coord.
  - if have same sign: P is in
- In 2D PA = (x_1,y_1), PB = (x_2,y_2)
  - PA×PB = $(x_1y_2 – y_1x_2)$
  - PA×PB > 0: points outward, Z
  - PA×PB < 0: points inward, -Z



$P = \alpha A + \beta B + \gamma C$

$\alpha+\beta+\gamma = 1$

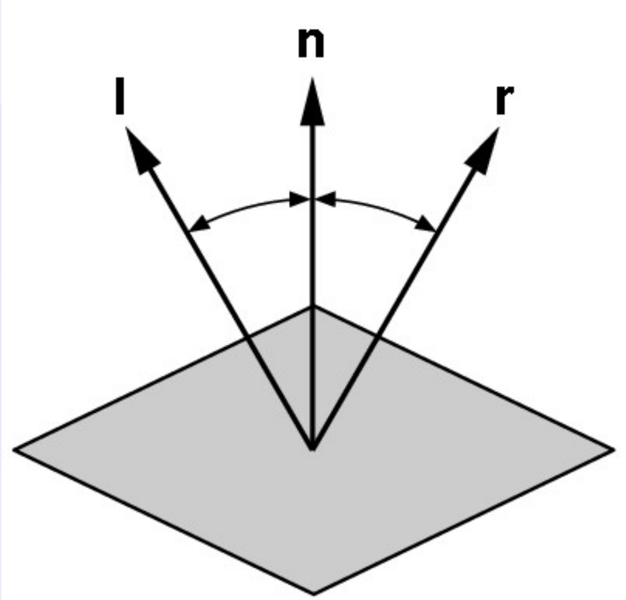
- Alternative:
- Compute barycentric coord. in 3D using same method
- more computation, but no need to projection

Phong Model

- Clamp dot product to 0-1

$$I = L\left(k_d(l \cdot n) + k_s(r \cdot v)^{\alpha}\right)$$

- L: light coefficient
- l: dirToLight, n: normal
- r: reflectDir = 2(**l** • **n**) **n** – **l**
- v: dirToCamera



Compute Normal

- Sphere:

$$n = \frac{1}{r}\left[(x_i - x_c) \quad (y_i - y_c) \quad (z_i - z_c)\right]^T$$

- Triangle:
  - Interpolate vertex normals using barycentric coord.
  - Interpolate diffuse, specular and shininess as well

$P = \alpha A + \beta B + \gamma C$
$\alpha+\beta+\gamma = 1$
$\alpha:\beta:\gamma = PB \times PC : PC \times PA : PA \times PB$

Debugging

- Do step by step
  - Intersect with sphere, test code
  - Intersect with triangle, test code
  - Compute sphere color, test code
  - Compute triangle color, test code

Notice

- Ensure B != 0 when doing A / B
- Before call sqrt(…), make sure

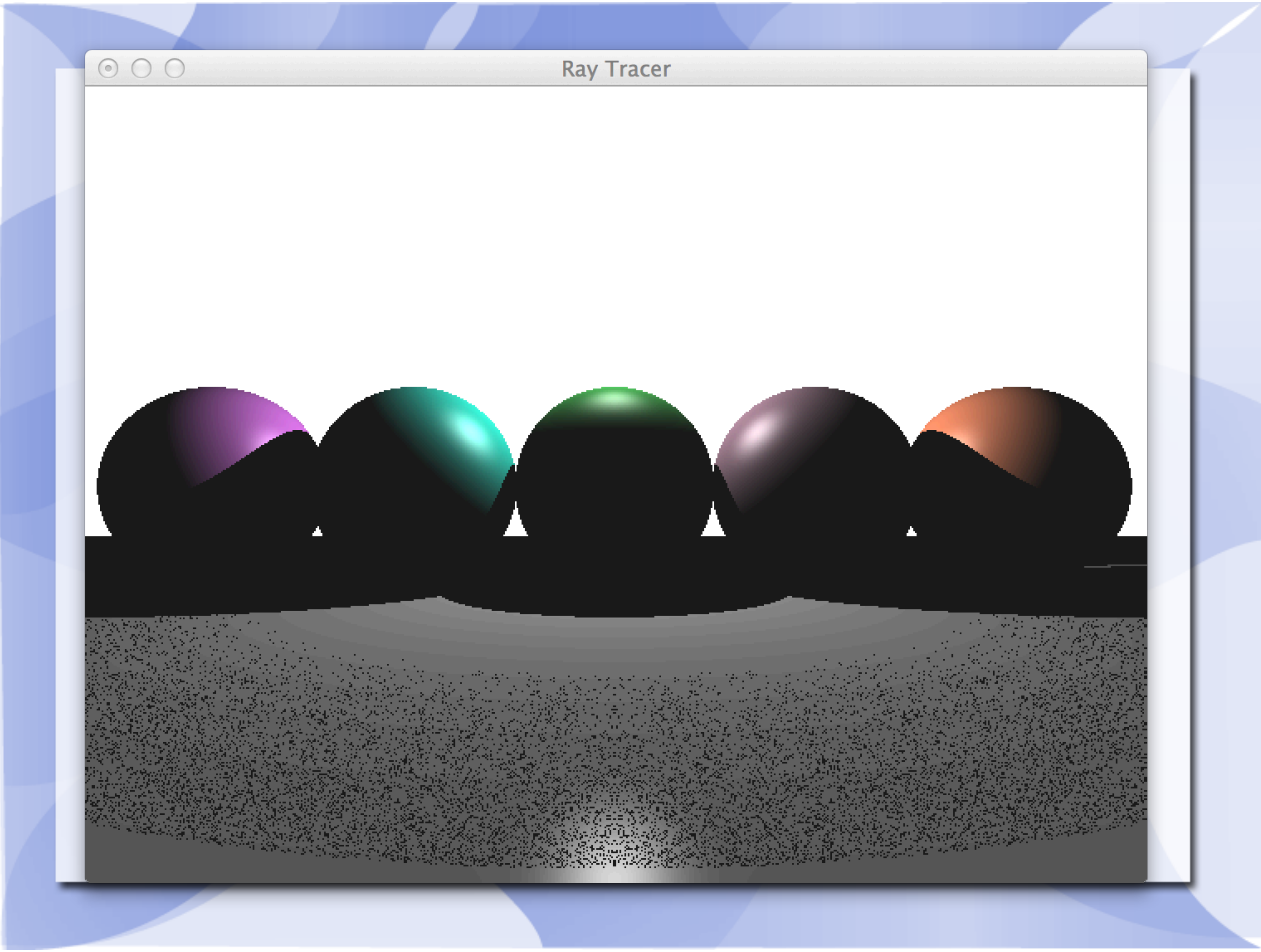
  parameter >= 0
- Remember to normalize direction vector. Remember to check len(dir) != 0 before dir.normalize()

Notice(cont'd)

- Distinguish between normals:
  - normal of a triangle
  - vertex normal
  - normal interpolated from vertex normals

Notice(cont'd)

- Floating−point operations not accurate:
  - When computing shadow rays:
  - distanceFromLightToFirstObject < distanceFromlightToTargetSurface – smallValue
  - Otherwise… (see next image)


Ray Tracer

Extra Credits

- Super sampling
  - anti-aliazing
  - can do adaptively: if some region is smooth, no need to super sampling
- Real ray tracing
  - (1-ks) localPhongColor + ks colorOfReflectedRay
  - You can also add refraction ray component

Extra Credit (Cont'd)

- Animation
- Soft shadow
- parallel computing to accelerate
  - openmp: utilize multi-core
  - cuda: use GPU to do parallel computing

Thanks!

Please email me any errors in the slides.