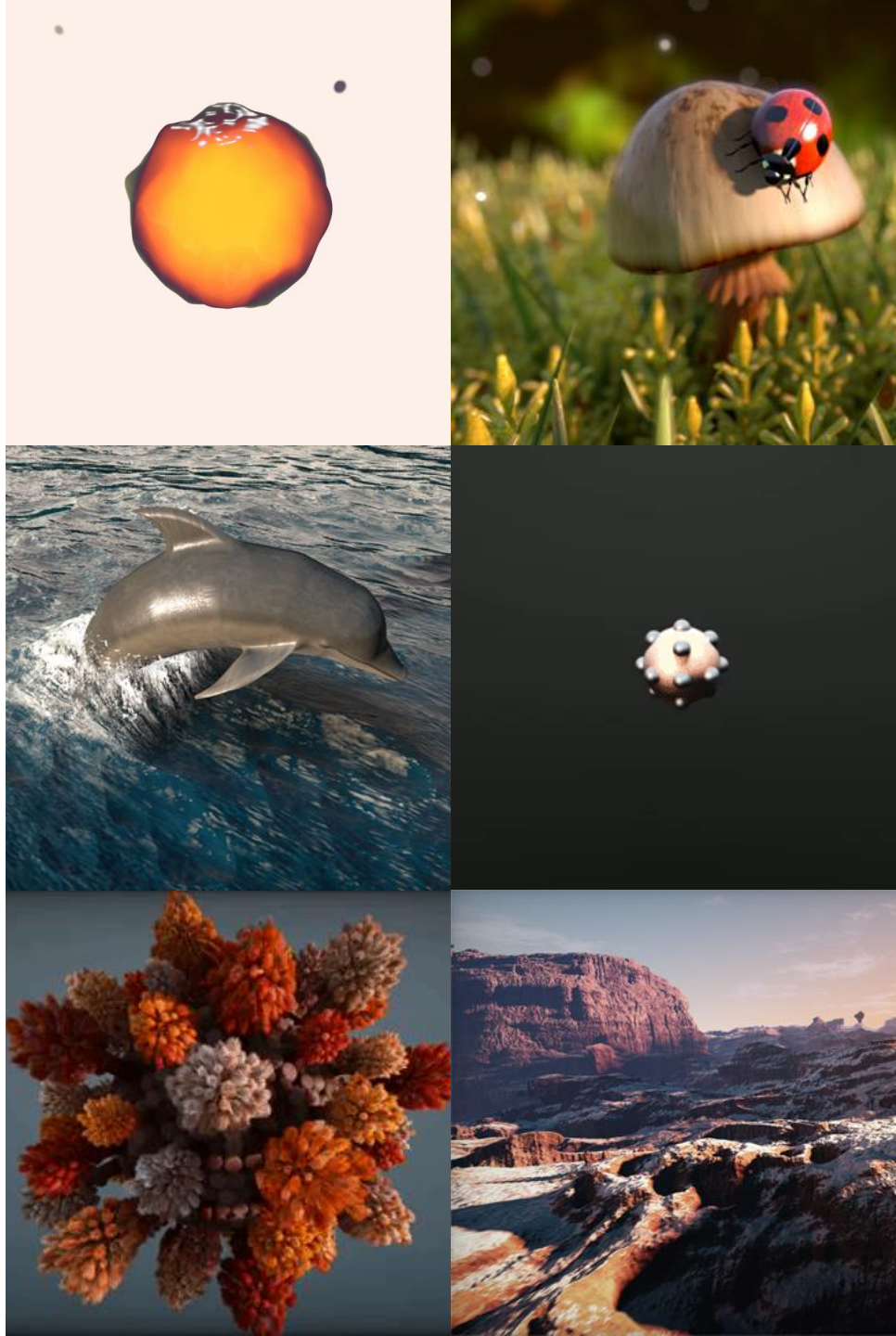
Ray Marching

w/ Signed Distance Functions

What is it?

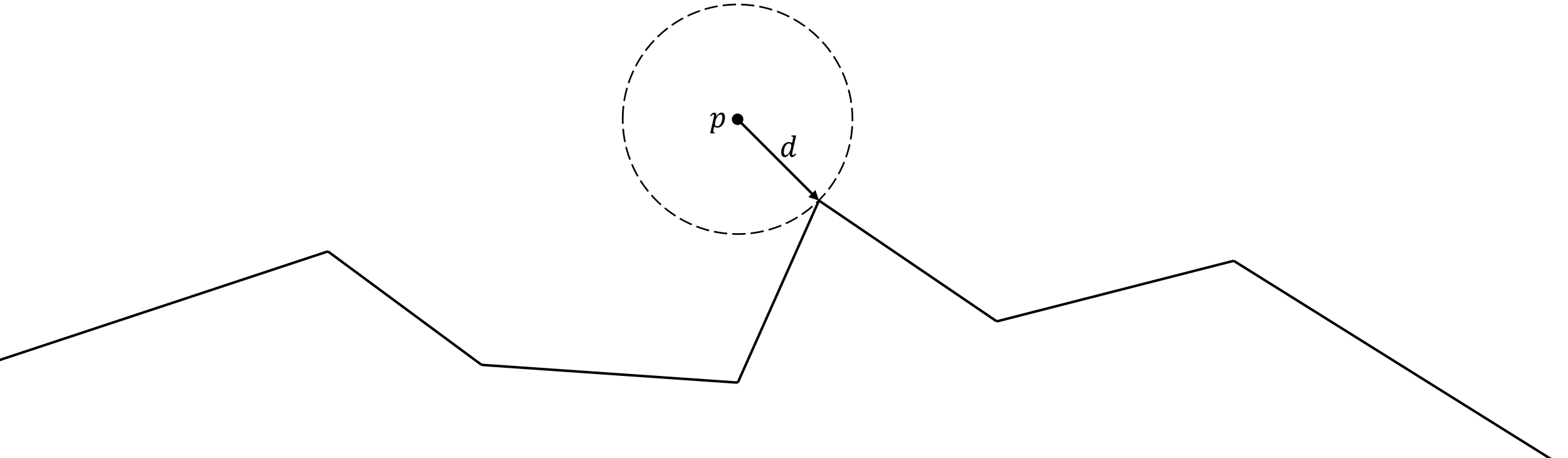
- Ray marching is an algorithm to render surfaces defined by signed distance functions
- Similar to ray tracing
- SDFs can represent extremely complex surfaces



Signed Distance Functions

Signed Distance Function

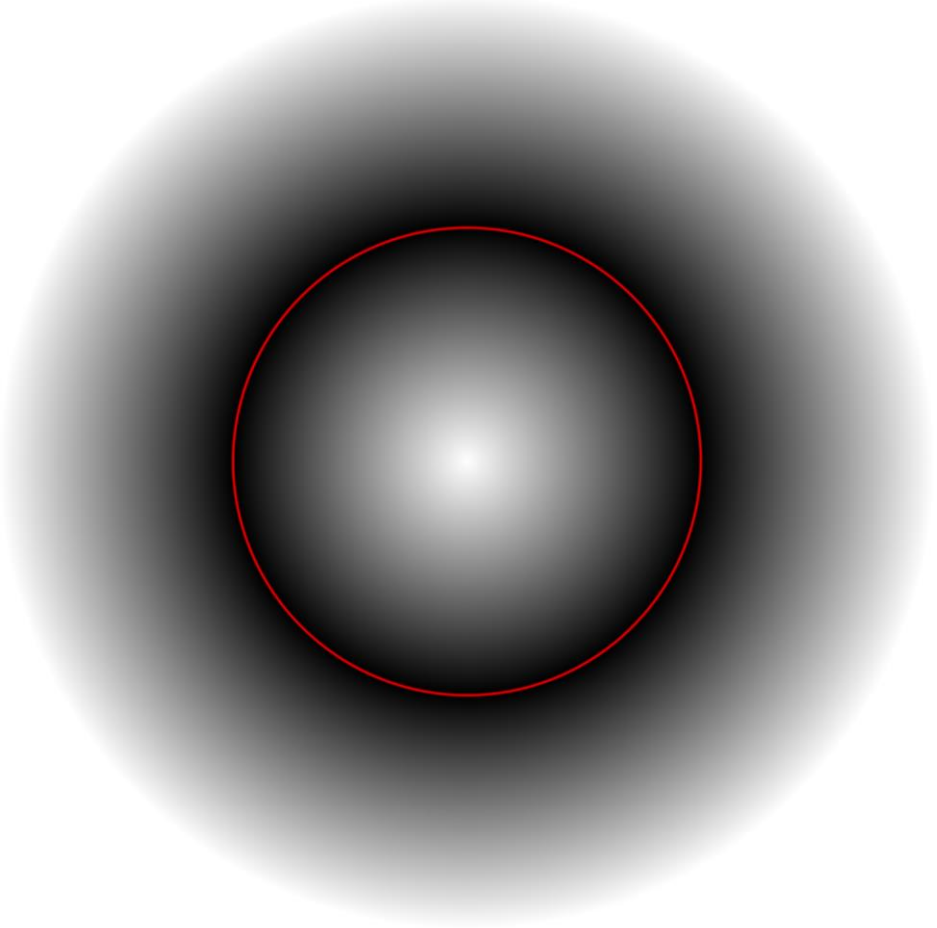
- Given a point p, an SDF returns the distance to the nearest point of a surface
- Positive outside, 0 on the surface, negative inside



Example with a 2D Circle

- Given a point p and a circle defined by center c and radius r
- $SDF(p) = \sqrt{(p_x - c_x)^2 + (p_y - c_y)^2} - r$

Or the distance between point and center minus the radius



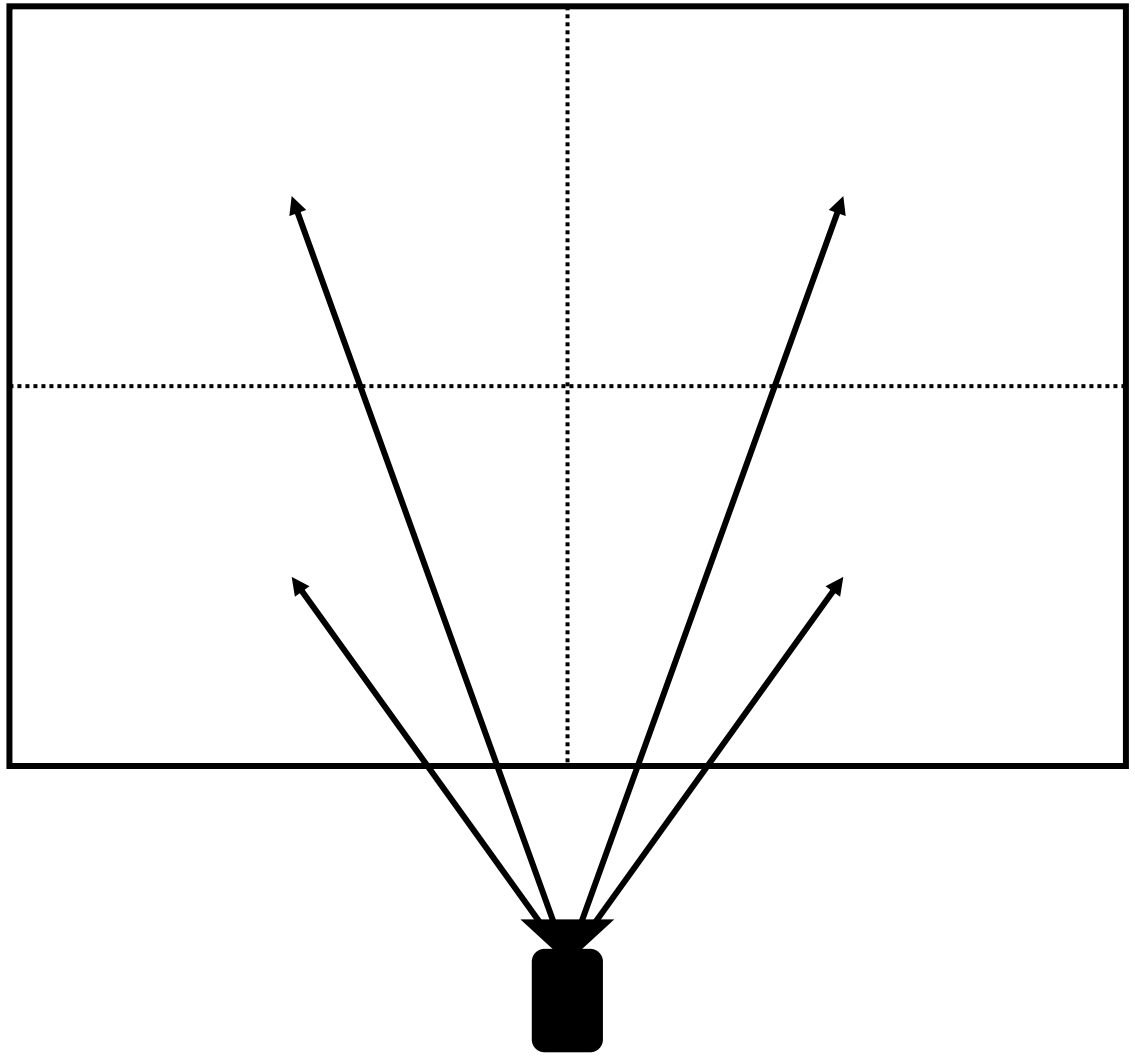
Code in GLSL

```
float sphereSdf(vec3 point, vec3 center, float radius) {
    return length(point - center) - radius;
}
```

Ray Marching

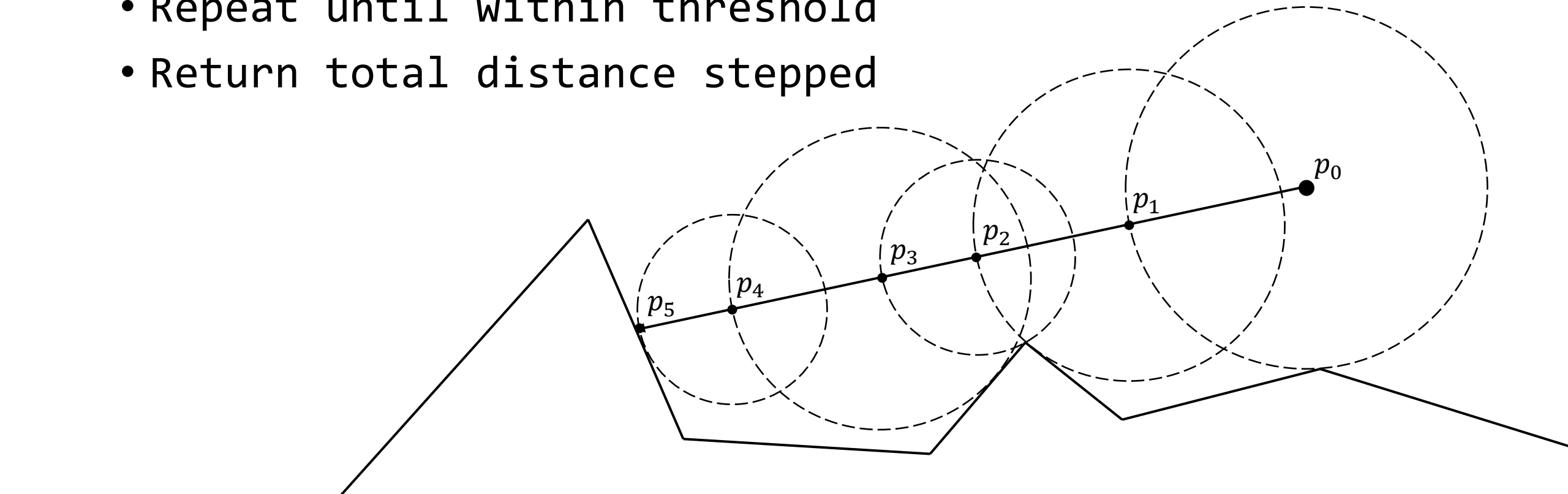
Ray Marching Basics

- For each pixel…
  - Create a ray from the camera through pixel
  - March along the ray to find surface intersection
  - Determine pixel color based on intersection and lights in scene



Marching along a Ray

- Starting at p_0, sample SDF
- Step that far along ray
- Repeat until within threshold
- Return total distance stepped



Code in GLSL

```
float marchRay(vec3 origin, vec3 ray) {
    float t = 0.0;
    for(int i = 0; i < 100; i++) {   // 100 is an arbitrary limit
        vec3 p = origin + t * ray;
        float dist = sdf(p);
        if (dist < 0.0001) return t;
        t += dist;
        if (t > 20.0) return infinity;
    }
    return t;
}
```

Demo

- Basic ray marcher
- RP logo

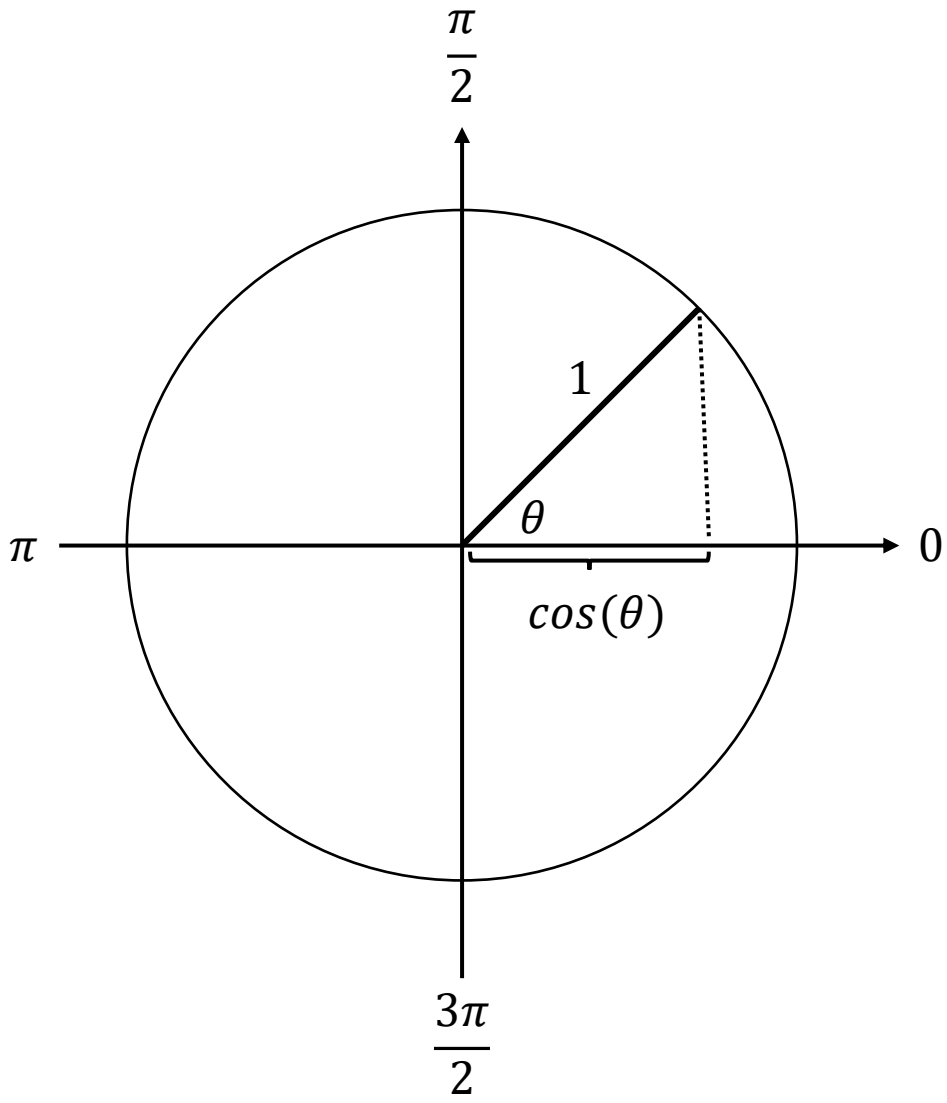
Appendix: Math Refresher

Vector

- $\langle x, y, z \rangle$
- Has a length/magnitude and direction
- Does not have a position or unit of measure
- A “unit vector” is a vector whose length is 1

Cosine

- $\cos(0) = 1$
- $\cos\left(\frac{\pi}{2}\right) = 0$



Dot Product

- If a and b are vectors, then
- $a \cdot b = a_x b_x + a_y b_y + a_z\, b_z$

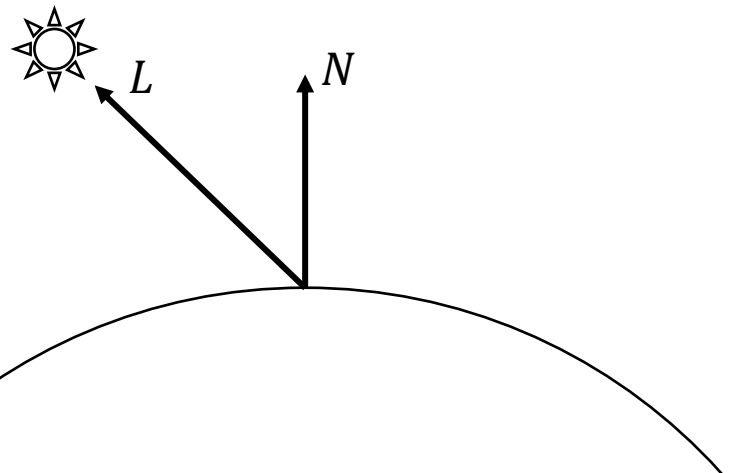
- If a and b are unit vectors, then
- $a \cdot b = \cos(\theta)$

- Parallel unit vectors give dot product of 1
- Perpendicular unit vectors give dot product of 0

Appendix: Lighting

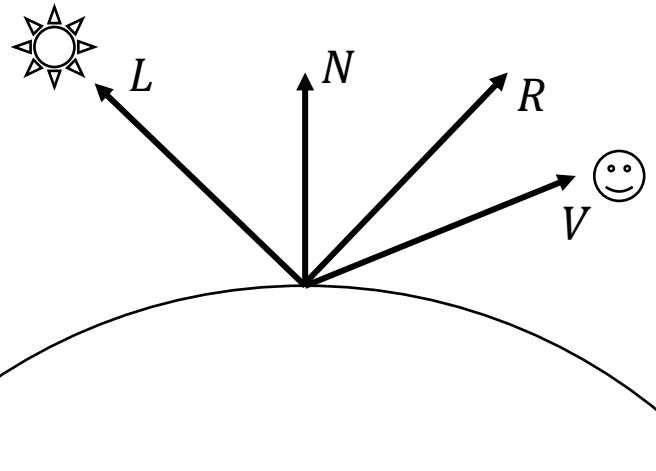
Lambert Diffuse

- $diffuse = (L \cdot N) * C * I_L$
- Where…
  - L is the direction of the light
  - N is the normal of the surface
  - C is the albedo (color) of the material
  - I_L is the intensity of the light



Phong Specular Highlight

- $specular = (R \cdot V)^{\alpha}$
- Where…
  - R is the incoming light direction ($-L$) reflected using the surface normal N
  - V is the direction of the camera
  - α is the intensity of the reflection (larger numbers mean more mirror-like surfaces with smaller highlight)



Complete Lighting

- $light = diffuse + specular$