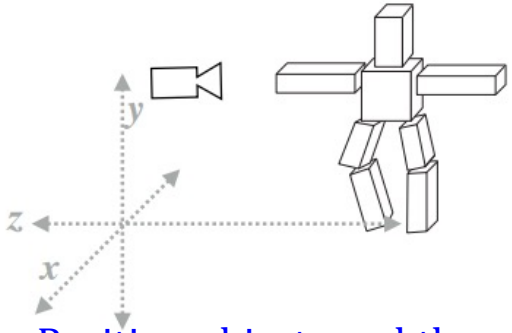
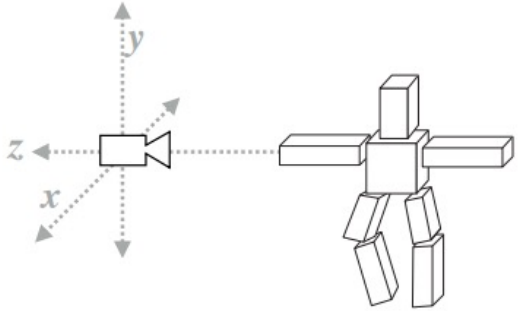
Illumination & Shading

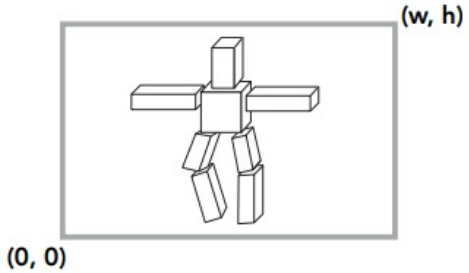
What We've Covered So Far



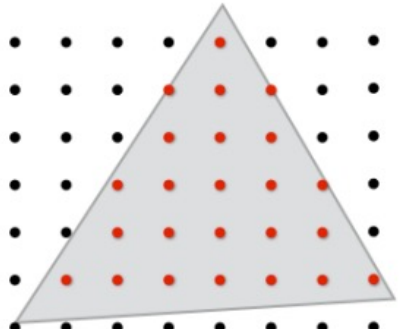
Position objects and the camera in the world



Compute position of objects relative to the camera

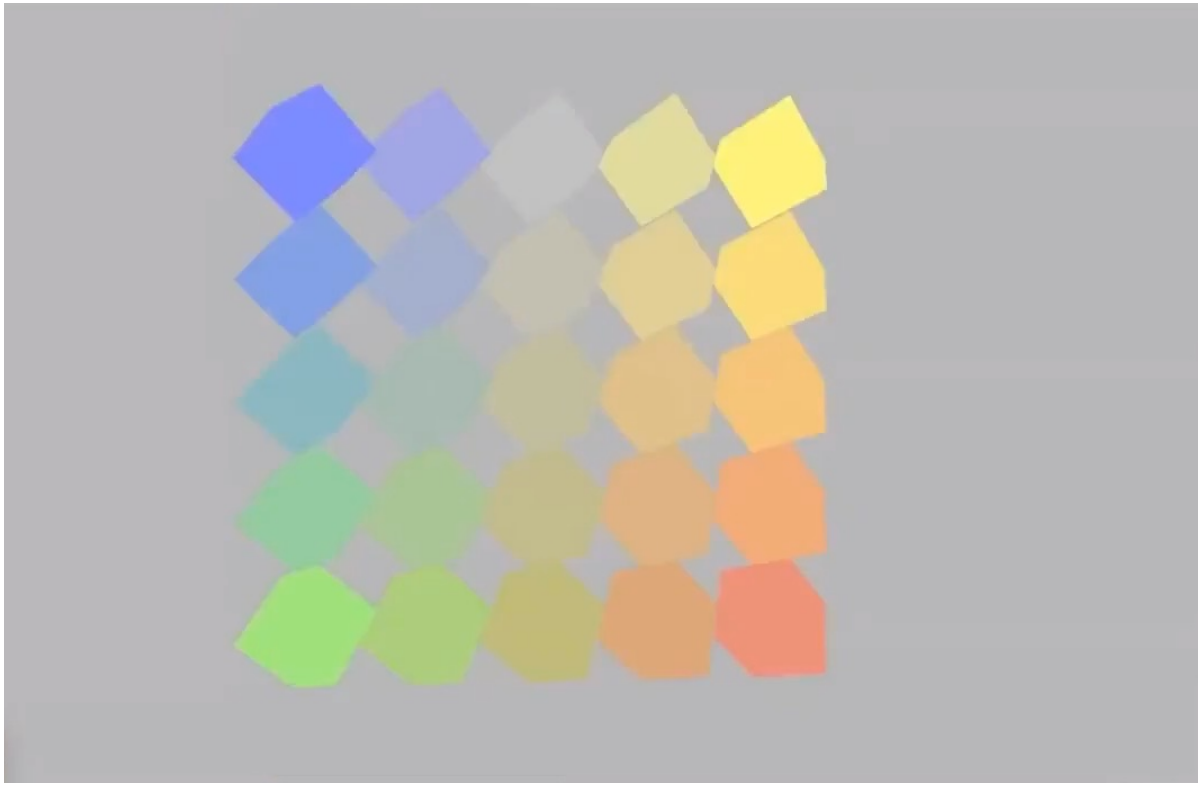


Project objects onto the screen

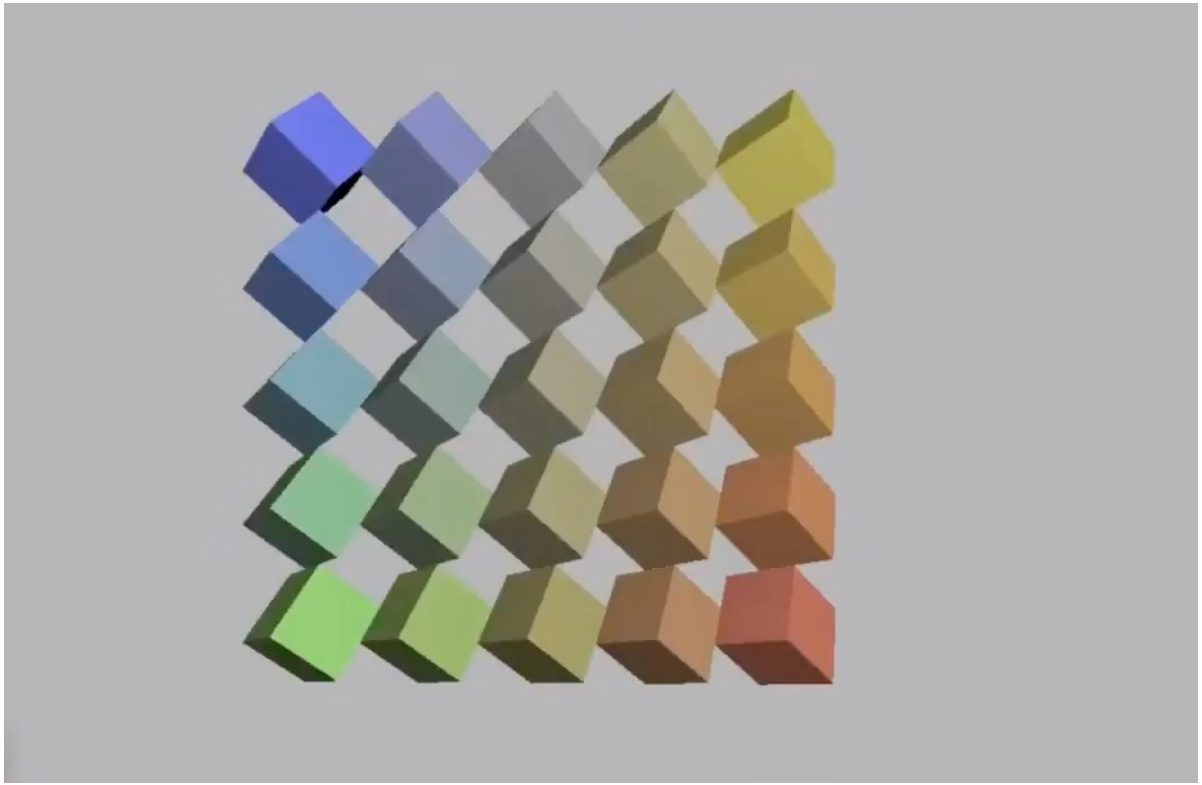


Sample triangle coverage

Rotating Cubes (Now You Can Do)



Rotating Cubes (Now You Can Do)



What Else Are We Missing

Credit: Bertrand Benoit. "Sweet Feast," 2009. [Blender /VRay]

This Time'Topic

- Blinn-Phong Reflectance Model
  - Diffuse
  - Specular
  - Ambient
- Shading Frequencies
- Review of Graphics Pipeline

Shading: Definition

- In Merriam-Webster Dictionary
  - **shad·ing**, noun
  - The darkening or coloring of an illustration or diagram with parallel lines or a block of color.

- In Computer Graphics
  - The process of applying a material to an object

A Simple Shading Model (Blinn-Phong Reflectance Model)

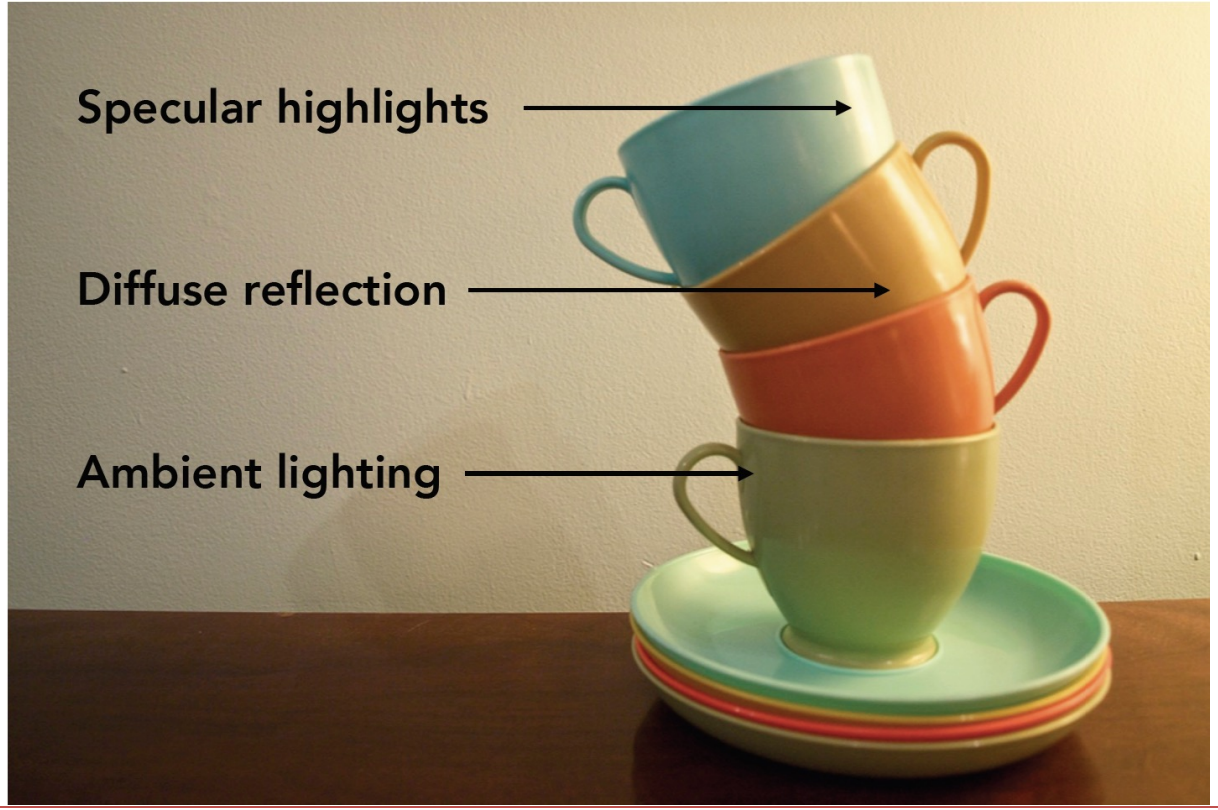
A local illumination model

one bounce: light → surface → viewer

Simple Shading vs Realistic Lighting & Materials

- What we will cover today
  - A local shading model: simple, per-pixel, fast
  - Based on perceptual observations, not physics
- What we will cover later in the course
  - Physics-based lighting and material representations
  - Global light transport simulation

Perceptual Observations

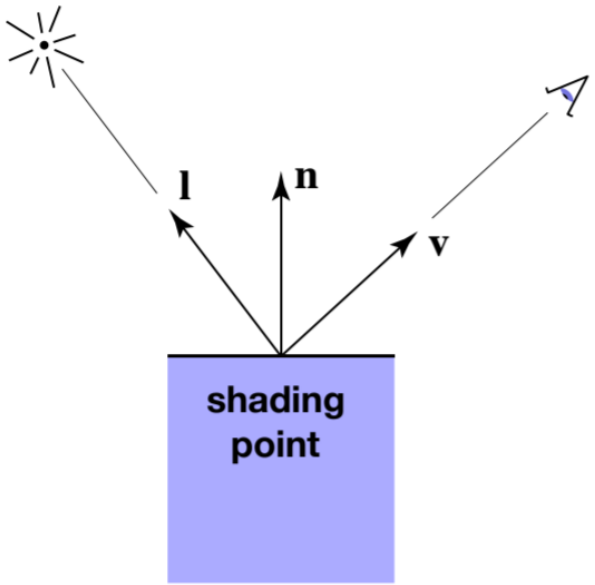


Shading is Local

- Compute light reflected toward camera at a specific shading point

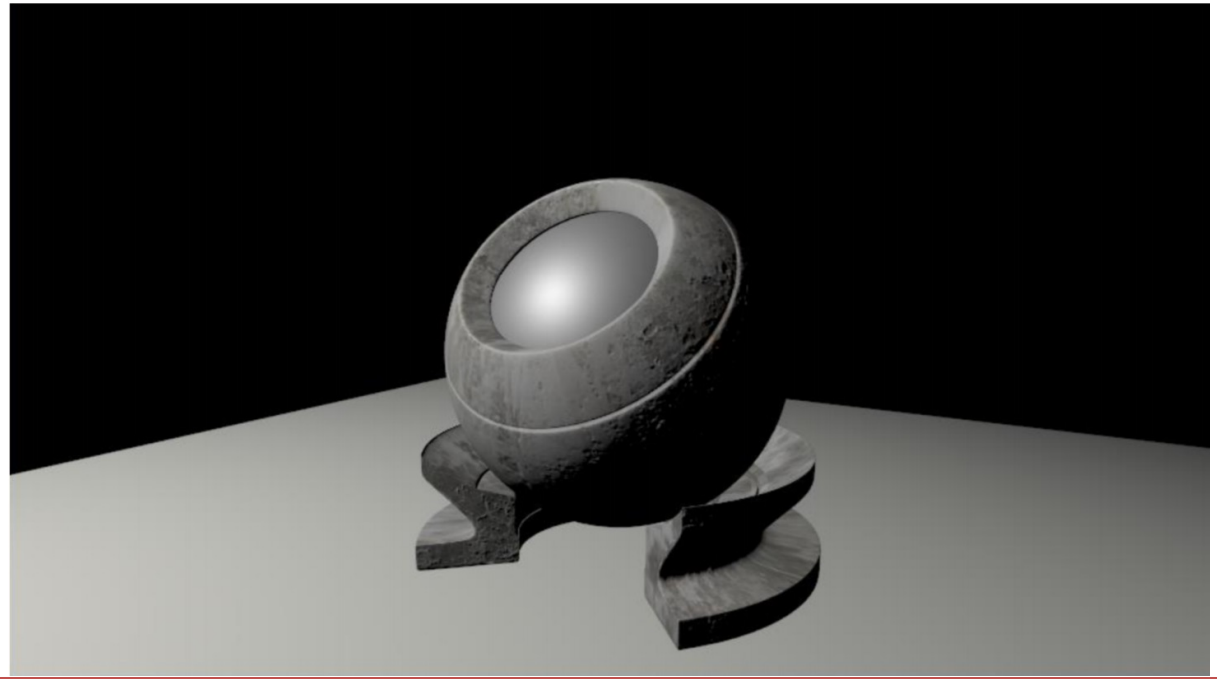
Inputs:

- Viewer direction, v
- Surface normal, n
- Light direction, l
  (for each of many lights)
- Surface parameters
  (color, shininess, …)



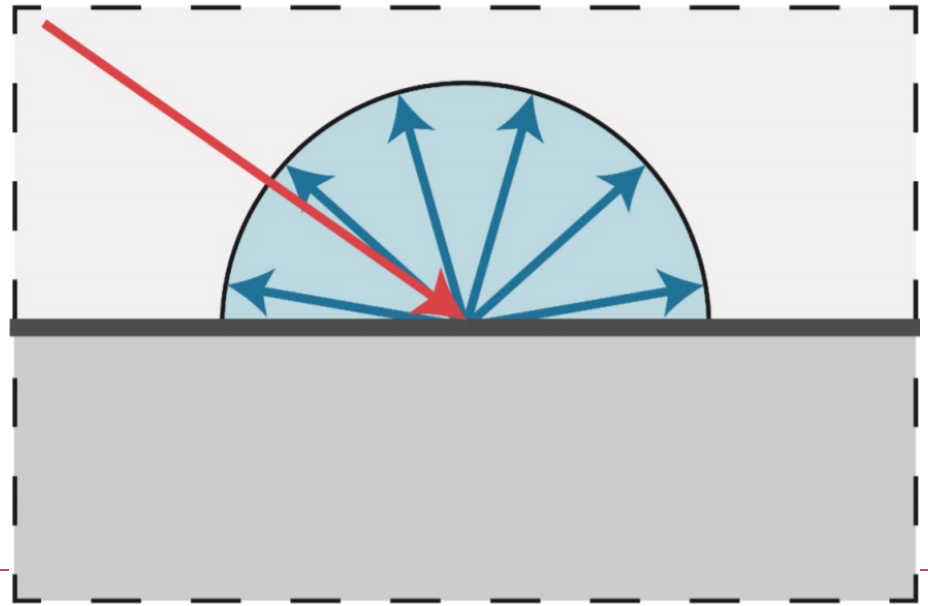
Shading is Local

- No shadows will be generated! (**shading ≠ shadow**)

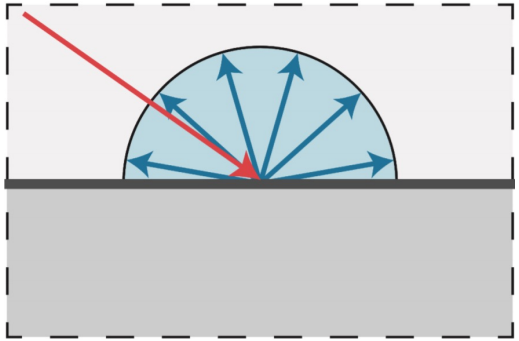


Diffuse Reflection

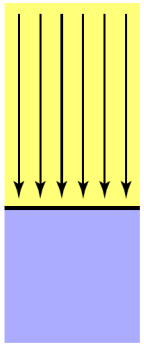
- Light is scattered uniformly in all directions
  - Surface color is the same for all viewing directions



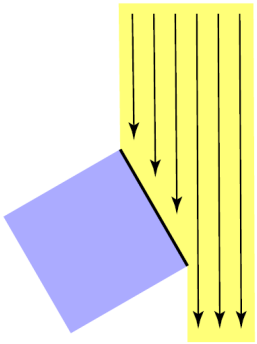
Diffuse Reflection



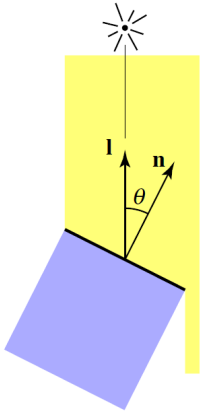
- Light is scattered uniformly in all directions
  - Surface color is the same for all viewing directions
- Lambert's cosine law



Top face of cube receives a certain amount of light

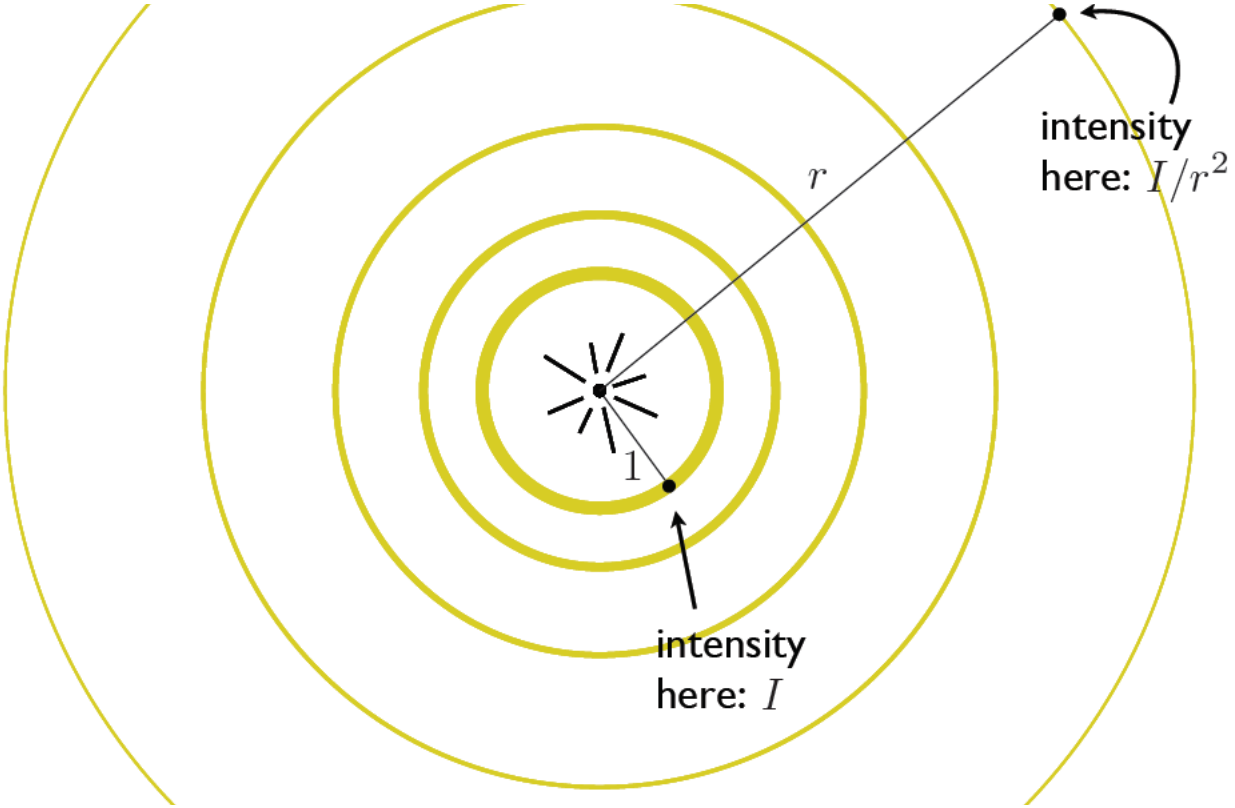


Top face of 60° rotated cube intercepts half the light



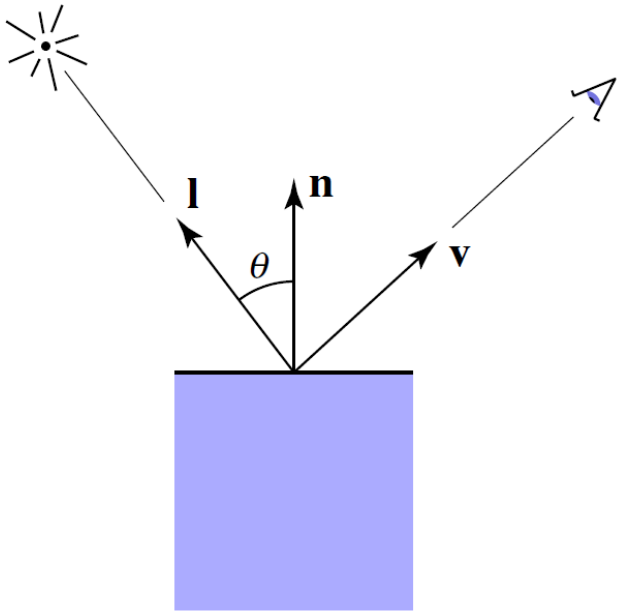
In general, light per unit area is proportional to $\cos\theta = \mathbf{l} \cdot \mathbf{n}$

Light Falloff



Lambertian (Diffuse) Shading

- Shading independent of view direction



illumination from source

$$L_d = k_d \, (I/r^2) \max(0, \mathbf{n} \cdot \mathbf{l})$$

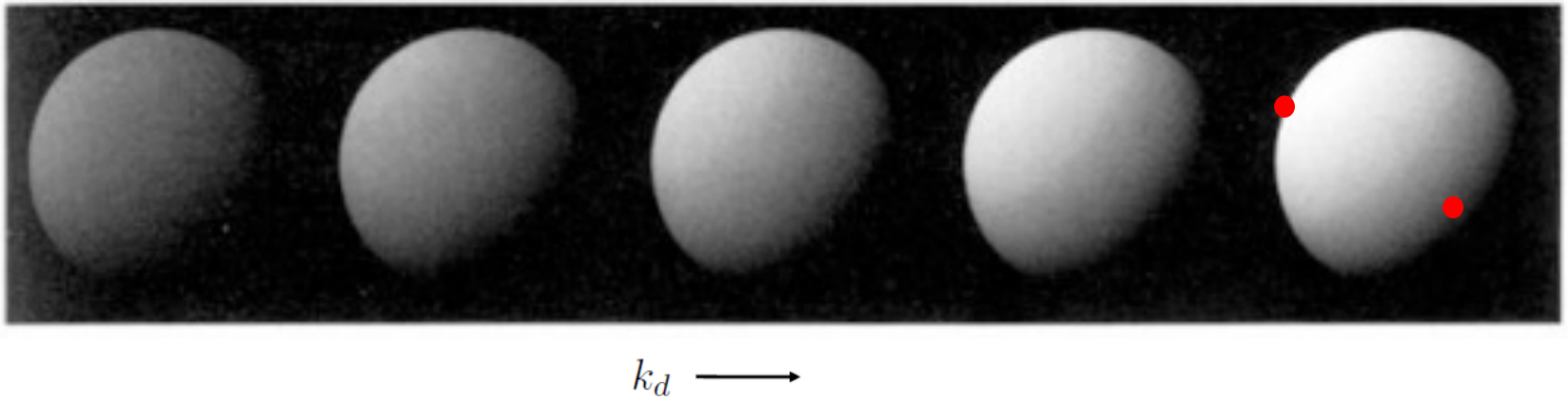
diffusely reflected light

diffuse coefficient

energy received by the shading point

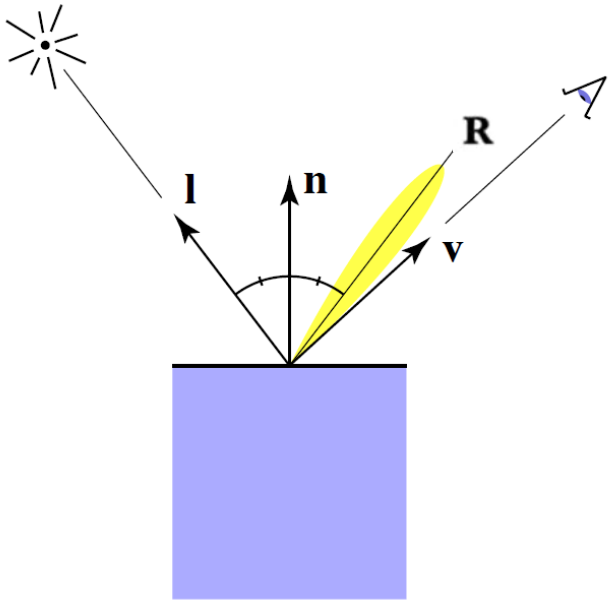
Lambertian (Diffuse) Shading

- Produces diffuse appearance



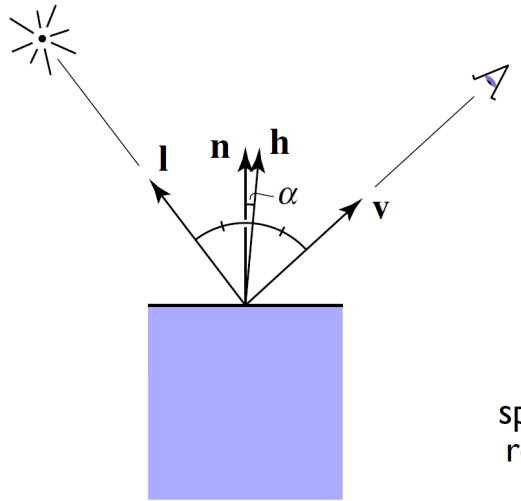
Specular Shading (Blinn-Phong)

- Intensity depends on view direction
  - Bright near mirror reflection direction



Specular Shading (Blinn-Phong)

- V Close to mirror direction ⇔ **half vector** near normal
  - Measure "near" by dot product of unit vectors



$$\mathbf{h} = \text{bisector}(\mathbf{v}, \mathbf{l}) \quad \textbf{(半程向量)}$$

$$= \frac{\mathbf{v} + \mathbf{l}}{\|\mathbf{v} + \mathbf{l}\|}$$

$$L_s = k_s \, (I/r^2) \max(0, \cos\alpha)^p$$

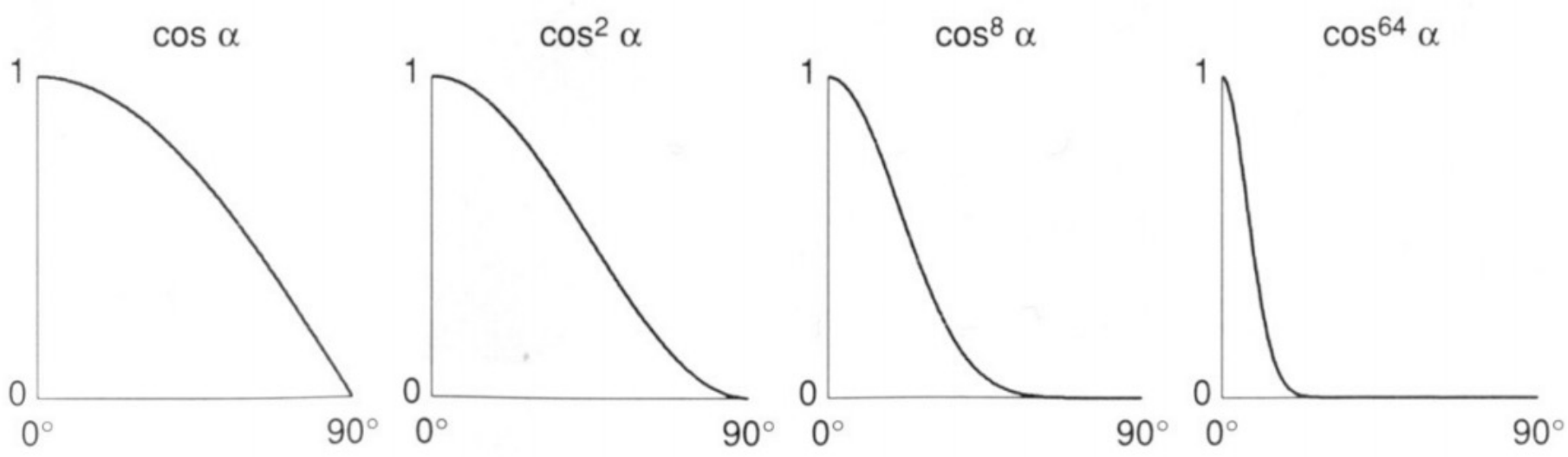
$$= k_s \, (I/r^2) \max(0, \mathbf{n} \cdot \mathbf{h})^p$$

specularly reflected light

specular coefficient

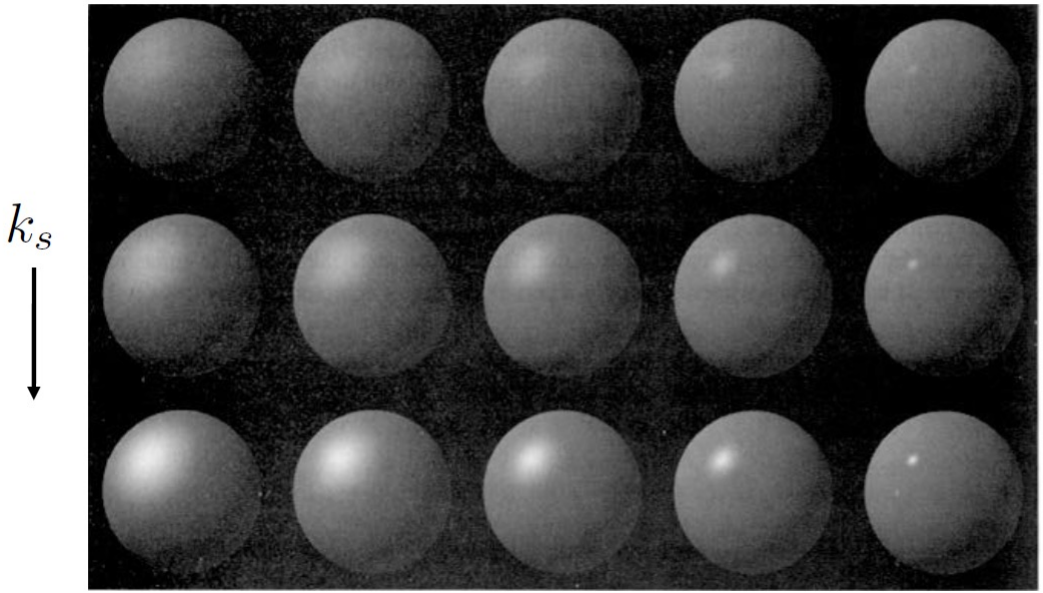
Cosine Power Plots

- Increasing p narrows the reflection lobe



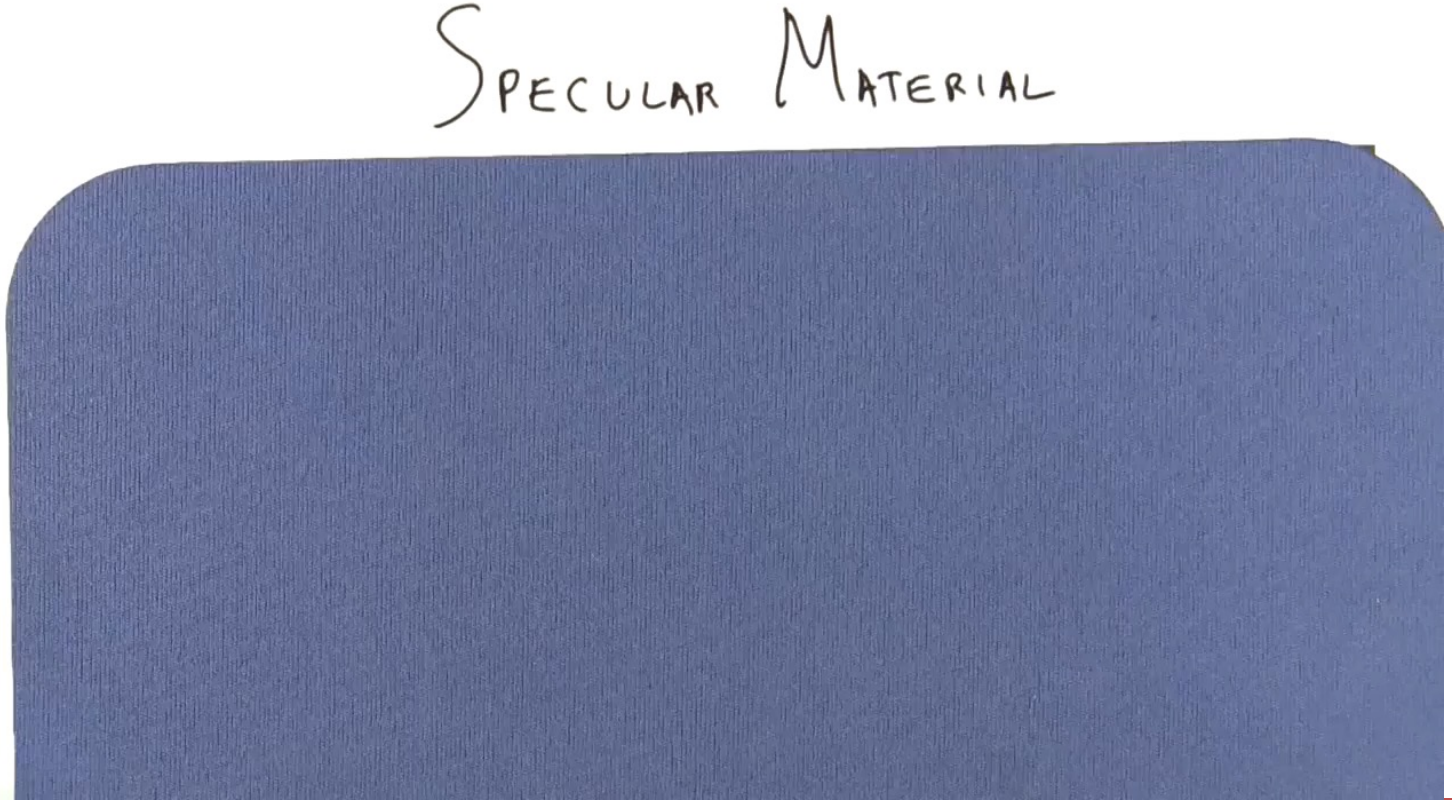
Specular Shading (Blinn-Phong)

$$L_s = k_s \, (I/r^2) \max(0, \mathbf{n} \cdot \mathbf{h})^p$$



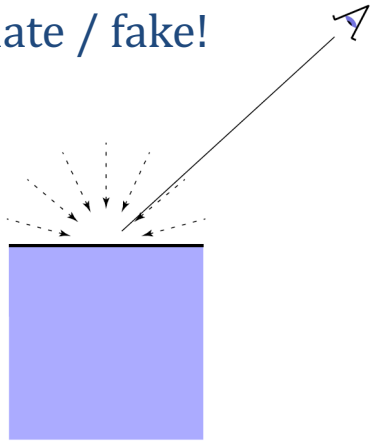
Specular Material (Video)

https://www.youtube.com/watch?v=2xWDoA8sbFk



Ambient Shading

- Shading that does not depend on anything
  - Add constant color to account for disregarded illumination and fill in black shadows
  - This is approximate / fake!

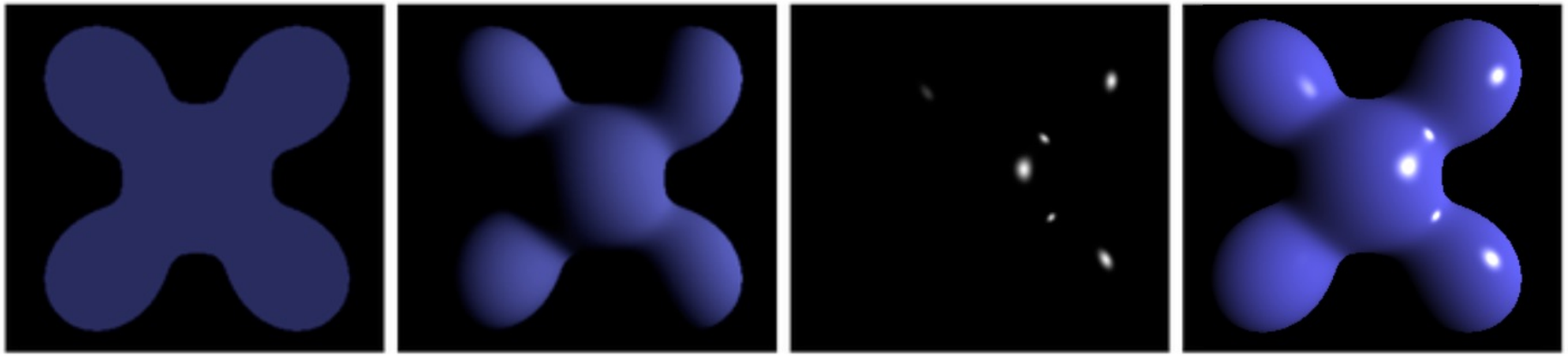


$$L_a = k_a I_a$$

ambient coefficient

reflected ambient light

Blinn-Phong Reflection Model



Ambient + Diffuse + Specular = Phong Reflection

$$L = L_a + L_d + L_s$$
$$= k_a\, I_a + k_d\, (I/r^2) \max(0, \mathbf{n} \cdot \mathbf{l}) + k_s\, (I/r^2) \max(0, \mathbf{n} \cdot \mathbf{h})^p$$

Questions?

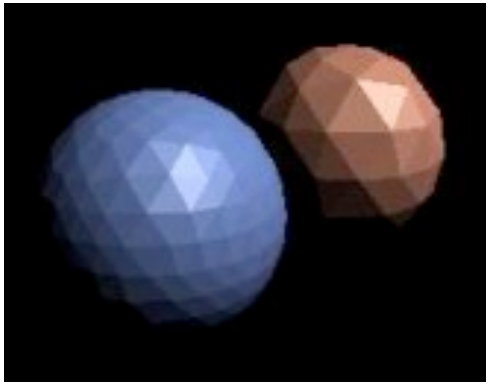
Shading Frequencies

What caused the shading difference?

What caused the shading difference?

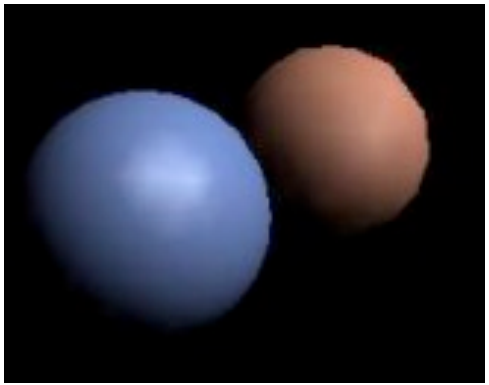
Shade each triangle/face (Flat shading)

- Flat shading
  - Shade each triangle/face
  - Triangle face is flat — one normal vector
  - Not good for smooth surfaces



Shade each vertex (Gouraud shading)

- Gouraud shading
  - Shade each vertex
  - Interpolate colors from vertices across triangle
  - Each vertex has a normal vector (how?)



Shade each pixel (Phong shading)

- Phong shading
  - Shade each pixel
  - Interpolate normal vectors across each triangle
  - Compute full shading model at each pixel
  - **Not the Blinn-Phong Reflectance Model**

Shading Frequency: Face, Vertex or Pixel

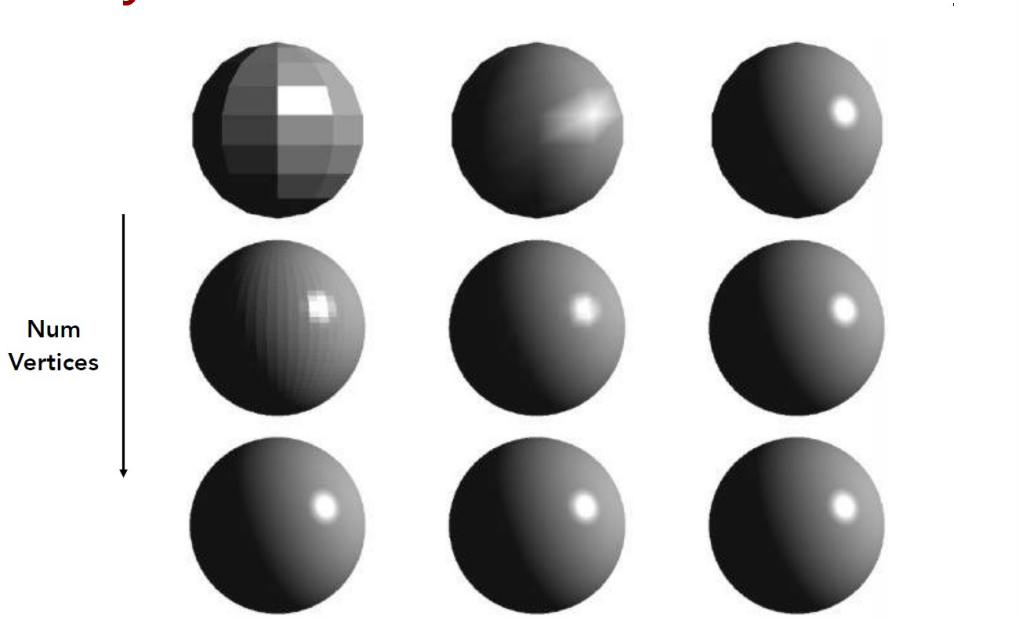
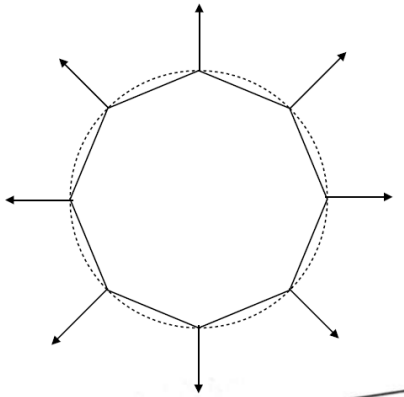


Image credit: Happyman, http://cg2010studio.com/

Defining Per-Vertex Normal Vectors

Best to get vertex normals from the underlying geometry

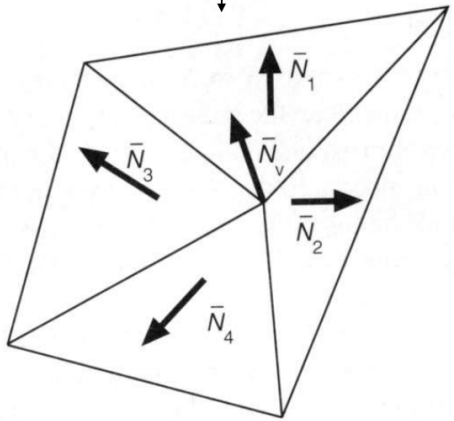
- e.g. consider a sphere



Otherwise have to infer vertex normals from triangle faces

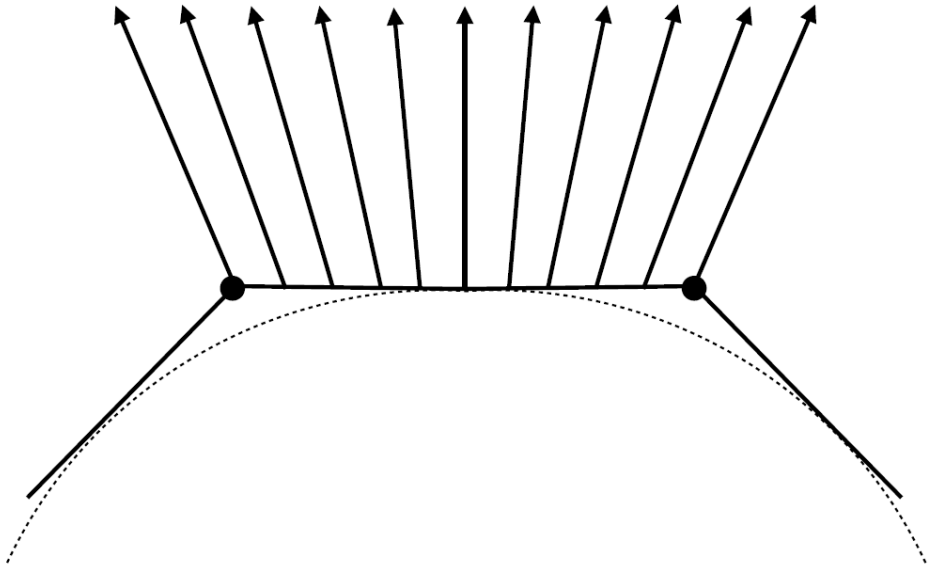
- Simple scheme: **average surrounding face normals**

$$N_v = \frac{\sum_i N_i}{\| \sum_i N_i \|}$$



Defining Per-Pixel Normal Vectors

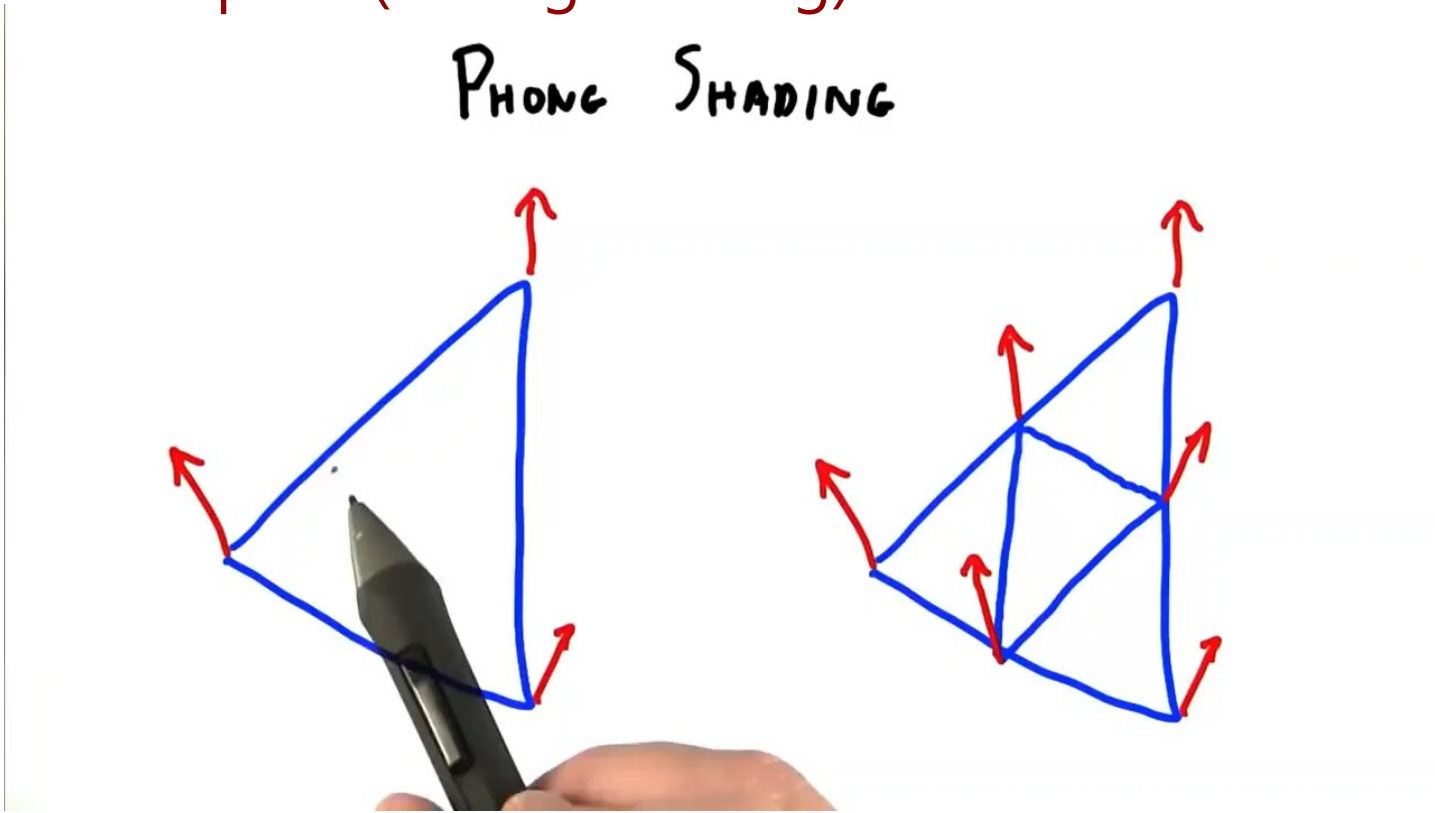
Barycentric interpolation of vertex normals



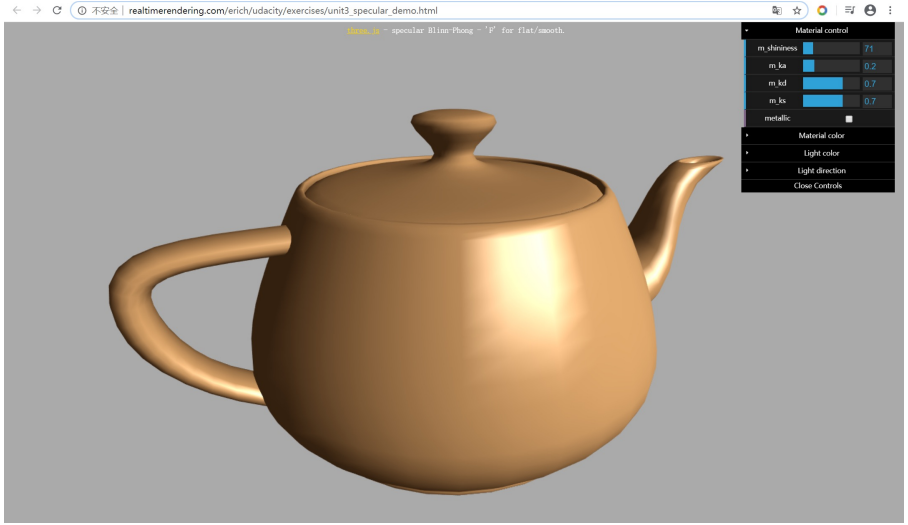
Problem: length of vectors?

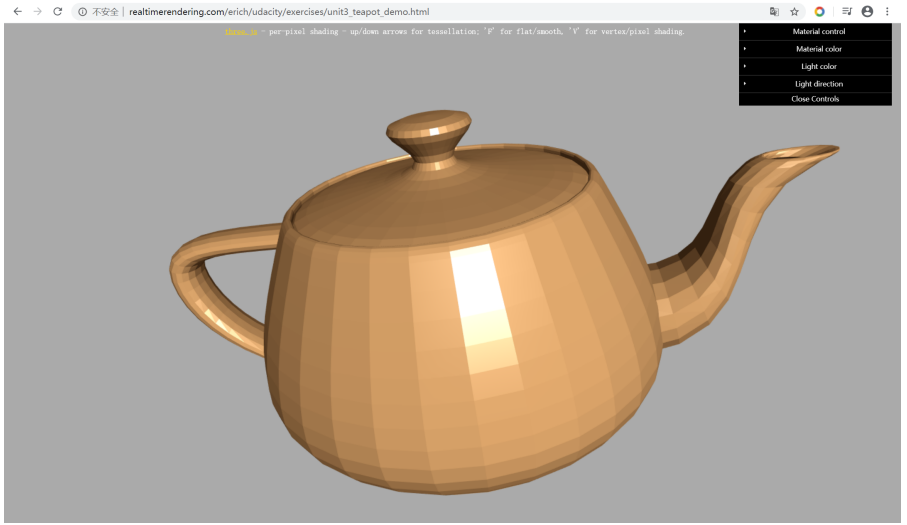
https://www.youtube.com/watch?v=jWaeoKI4cwY

Shade each pixel (Phong shading)



Try Online Exercise!



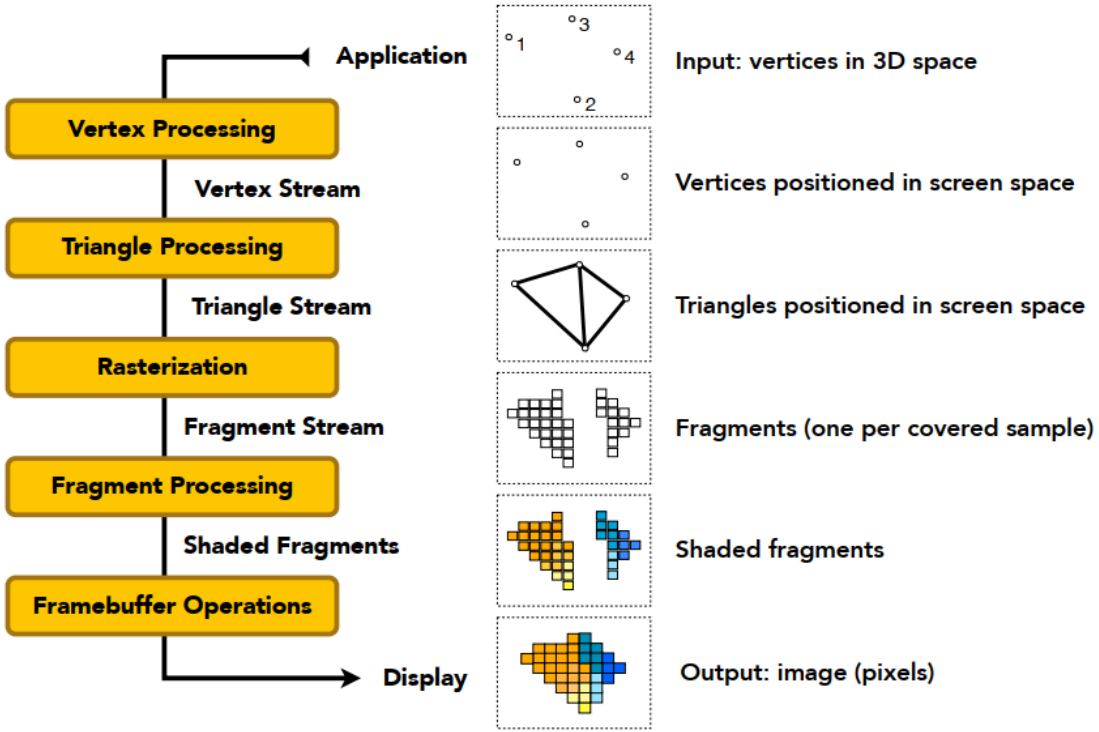


http://www.realtimerendering.com/erich/udacity/exercises/unit3_specular_demo.html

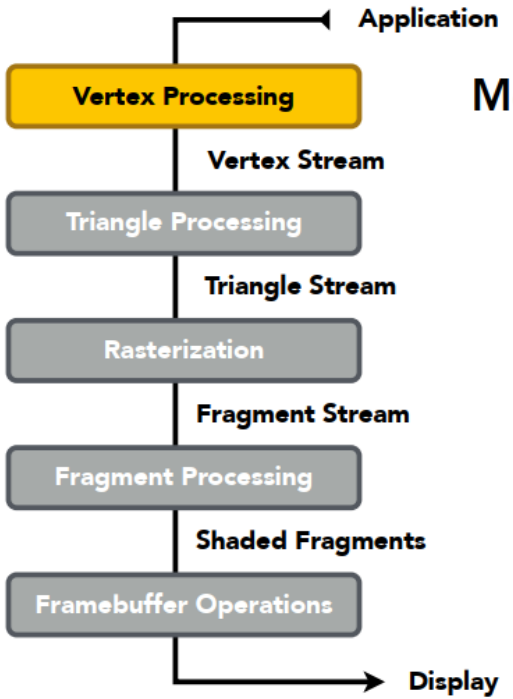
http://www.realtimerendering.com/erich/udacity/exercises/unit3_teapot_demo.html

Recap of The Graphics Pipeline

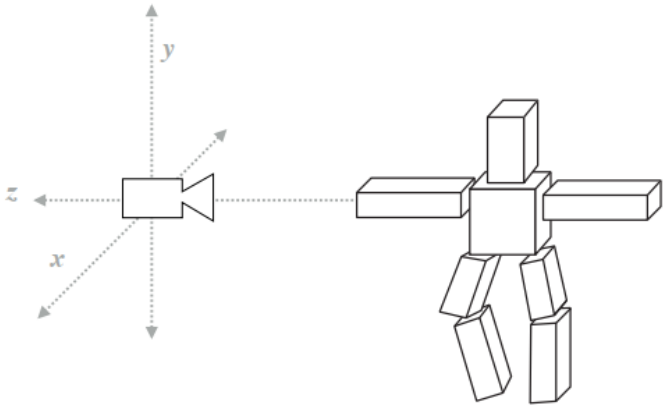
Graphics Pipeline



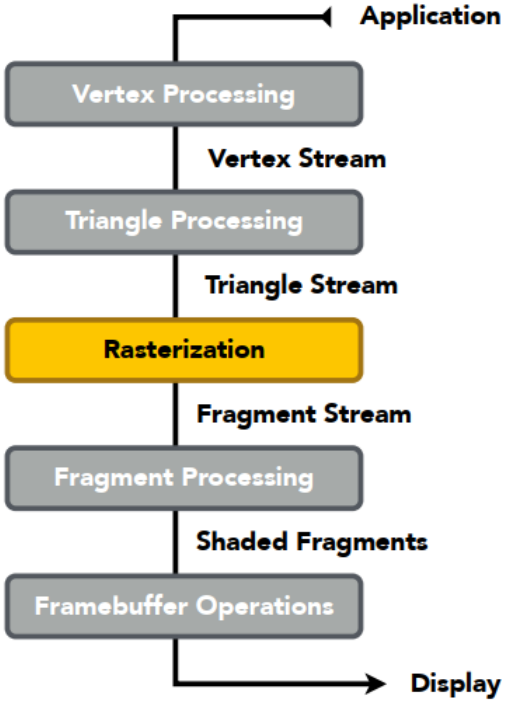
Graphics Pipeline



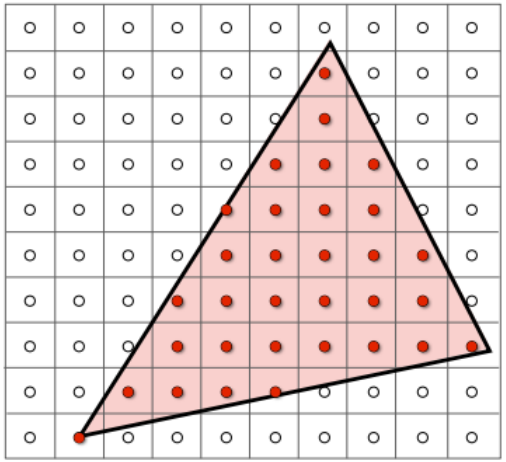
Model, View, Projection transforms

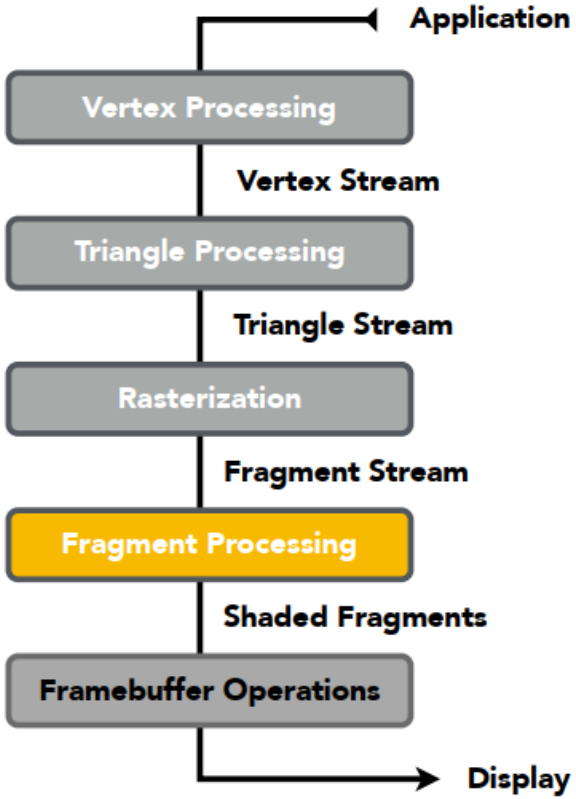


Graphics Pipeline



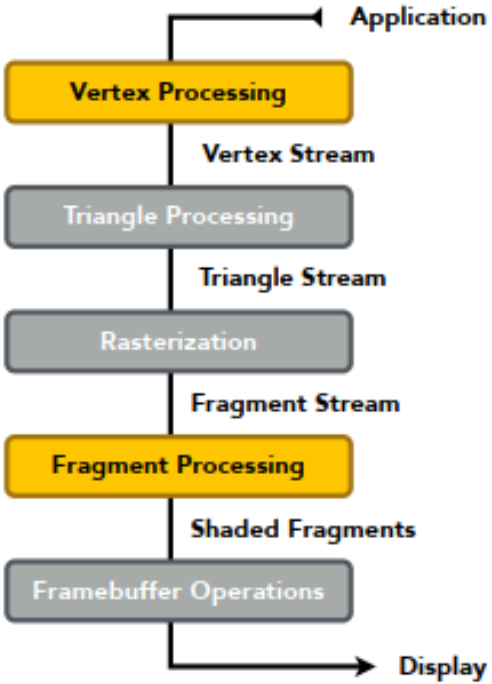
Sampling triangle coverage



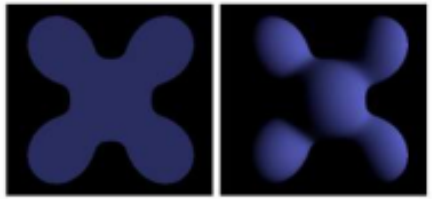


Z-Buffer Visibility Tests

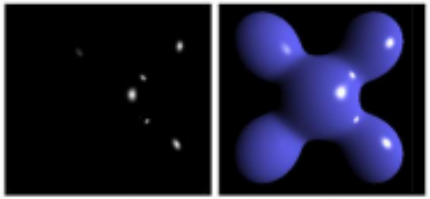
Graphics Pipeline



Shading

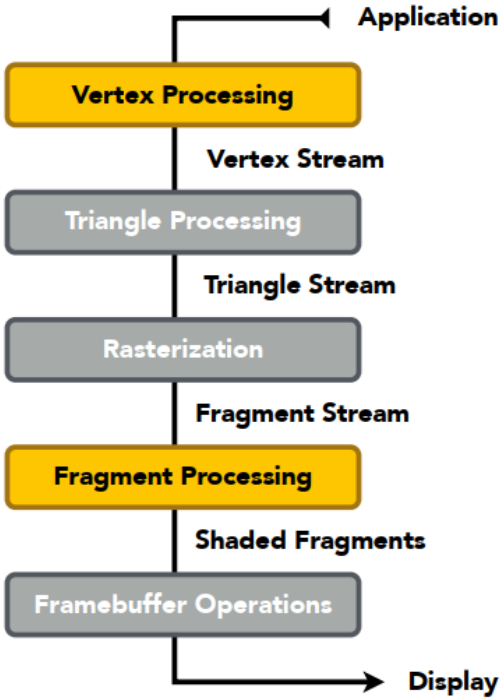


Ambient + Diffuse



+ Specular = Blinn-Phong Reflectance Model

Graphics Pipeline



Texture mapping
(introducing soon)

Shader Programs

- Program vertex and fragment processing stages
- Describe operation on a single vertex (or fragment)

Example GLSL fragment shader program

```
uniform sampler2D myTexture;
uniform vec3 lightDir;
varying vec2 uv;
varying vec3 norm;

void diffuseShader()
{
  vec3 kd;
  kd = texture2d(myTexture, uv);
  kd *= clamp(dot(–lightDir, norm), 0.0, 1.0);
  gl_FragColor = vec4(kd, 1.0);
}
```

- **Shader function executes once per fragment.**
- **Outputs color of surface at the current fragment's screen sample position.**
- **This shader performs a texture lookup to obtain the surface's material color at this point, then performs a diffuse lighting calculation.**

Shader Programs

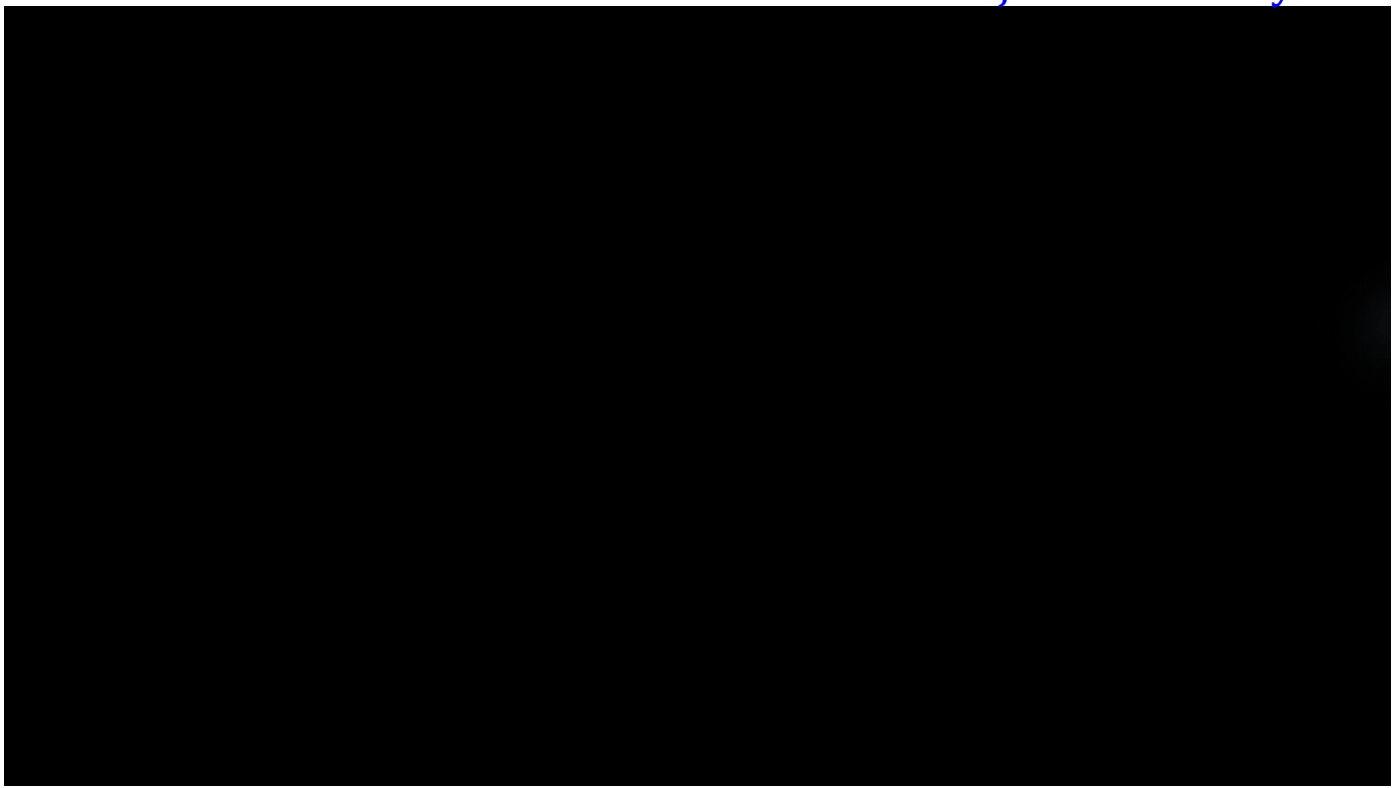
- Program vertex and fragment processing stages
- Describe operation on a single vertex (or fragment)

Example GLSL fragment shader program

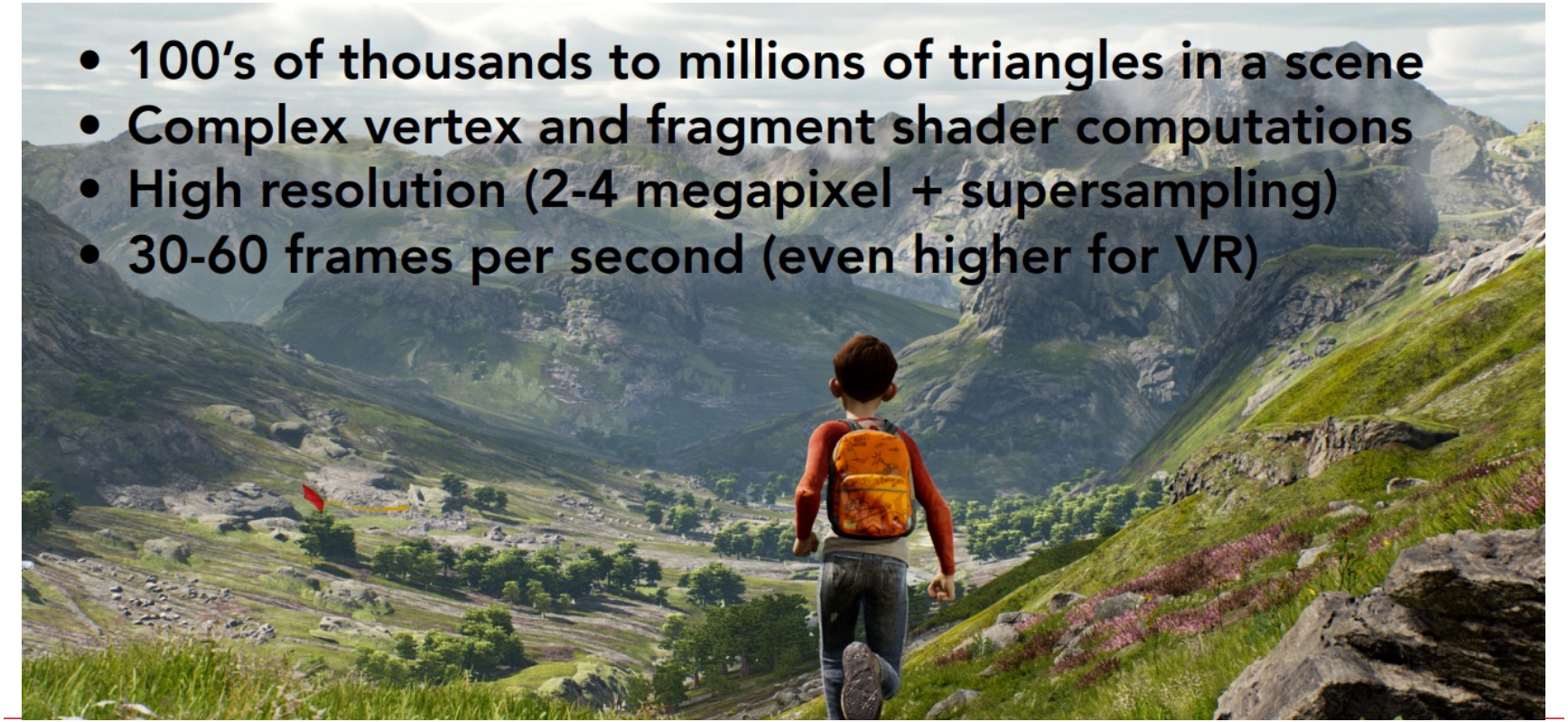
```
uniform sampler2D myTexture;      // program parameter
uniform vec3 lightDir;            // program parameter
varying vec2 uv;                  // per fragment value (interp. by rasterizer)
varying vec3 norm;                // per fragment value (interp. by rasterizer)

void diffuseShader()
{
  vec3 kd;
  kd = texture2d(myTexture, uv);                  // material color from texture
  kd *= clamp(dot(–lightDir, norm), 0.0, 1.0);    // Lambertian shading model
  gl_FragColor = vec4(kd, 1.0);                   // output fragment color
}
```

Snail Shader Program(Video)

https://www.youtube.com/watch?v=XuSnLbB1j6E&feature=youtu.be



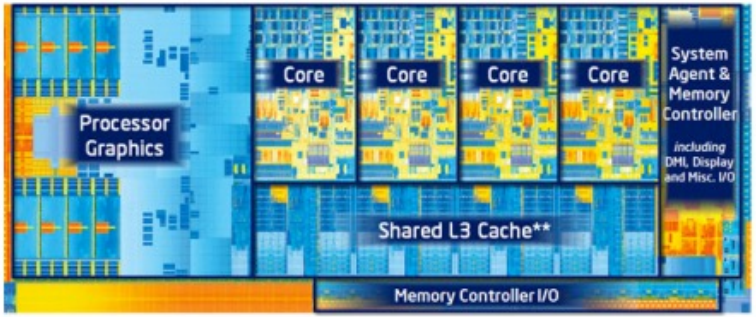
Goal: Highly Complex 3D Scenes in Realtime



Graphics Pipeline Implementation: GPUs

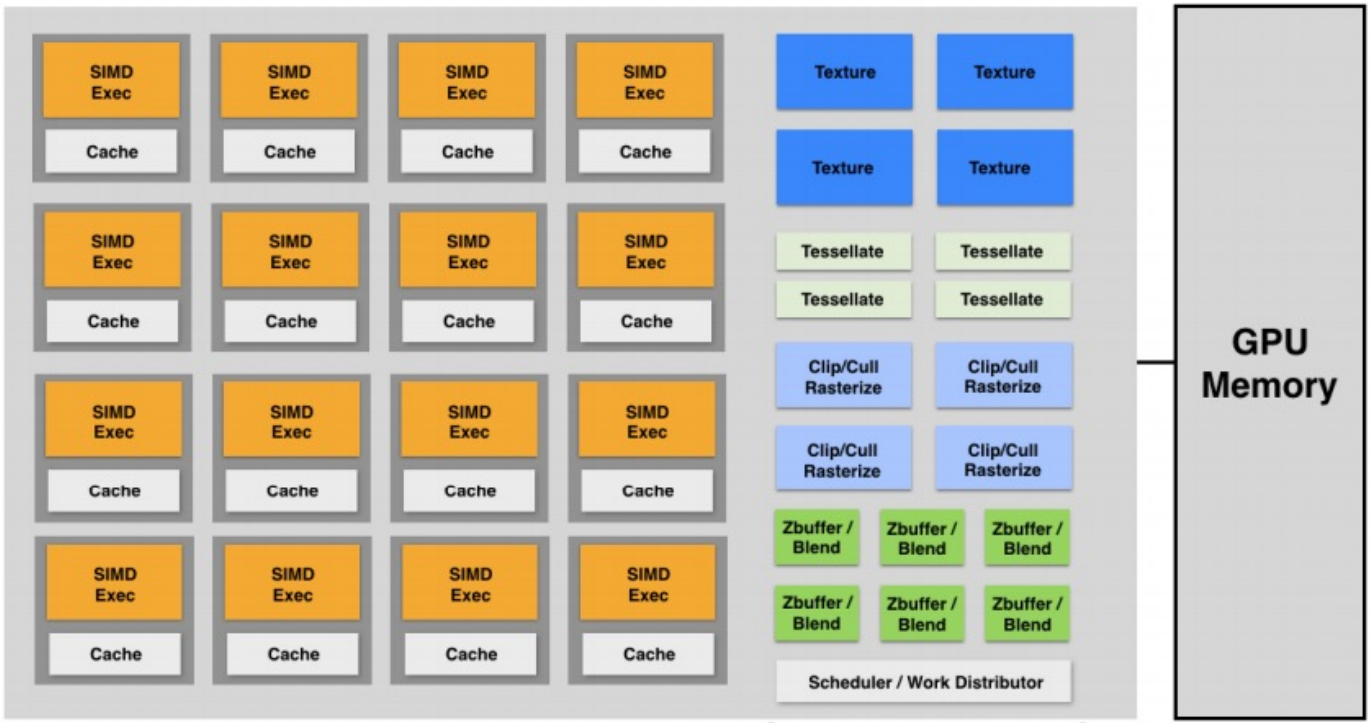
Specialized processors for executing graphics pipeline computations

Discrete GPU Card
(NVIDIA GeForce Titan X)



Integrated GPU:
(Part of Intel CPU die)

GPU: Heterogeneous, Multi-Core Processor



Thank you!