

CSE 167:
Introduction to Computer Graphics
Lecture #8: Lights

Jürgen P. Schulze, Ph.D.
University of California, San Diego
Fall Quarter 2020

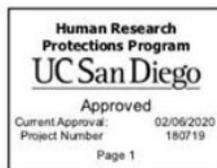
Announcements

- ▶ **Sunday, November 1st at 11:59pm:**
 - ▶ Homework Project 1 late deadline
 - ▶ 25% penalty on score
- ▶ **Tomorrow at 1pm:**
 - ▶ Discussion Project 2
- ▶ **Sunday, November 8th at 11:59pm:**
 - ▶ Homework Project 2 due
- ▶ **Tips for homework submissions**
 - ▶ How to submit source code and video
- ▶ **User study participants needed**
 - ▶ See next two slides or at:
 - ▶ <https://sites.google.com/view/courtneylab/research-studies/vr-studies>

Do you smoke cigarettes or use nicotine products?

Researchers in the Department of Psychiatry at UCSD are seeking individuals ages 18+ who regularly smoke tobacco cigarettes, hookah, or use e-cigs for a study using virtual reality to explore biological markers of craving for various substances. This is NOT a treatment study for nicotine and tobacco use. This research study involves one visit to the lab where you will be asked to complete interviews and questionnaires, and a virtual reality task. You may then be asked to fill-out follow-up questionnaires through your smart phone/device for 1 week. The principal investigator of this study is Dr. Kelly Courtney.

**FOR THE COMPLETION OF THE STUDY,
PARTICIPANTS WILL BE COMPENSATED
UP TO \$120.**

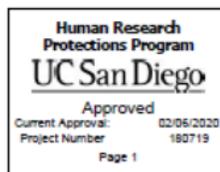


Call 858-534-5157 or
Email vrstudy@health.ucsd.edu
for information

Do you drink alcohol?

Researchers in the Department of Psychiatry at UCSD are seeking individuals ages 18+ who regularly drink alcohol for a study using virtual reality to explore biological markers of craving for various substances. This is NOT a treatment study for substance use. This research study involves one visit to the lab where you will be asked to complete interviews and questionnaires, and a virtual reality task. You may then be asked to fill-out follow-up questionnaires through your smart phone/device for 1 week. The principal investigator of this study is Dr. Kelly Courtney.

**FOR THE COMPLETION OF THE STUDY,
PARTICIPANTS WILL BE COMPENSATED
UP TO \$120.**

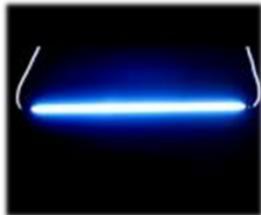


Call 858-534-5157 or
Email vrstudy@health.ucsd.edu
for information

UCSD

Light Sources

- ▶ Real light sources can have complex properties
 - ▶ Geometric area over which light is produced
 - ▶ Anisotropy (directionally dependent)
 - ▶ Reflective surfaces act as light sources (indirect light)



- ▶ Need to use simplified model for real-time rendering

Types of Light Sources

- ▶ At each point on surfaces we need to know
 - ▶ Direction of incoming light (the \mathbf{L} vector)
 - ▶ Intensity of incoming light (the c_l values)
- ▶ Three light types:
 - ▶ **Directional**: from a specific direction
 - ▶ **Point light**: from a specific point
 - ▶ **Spotlight**: from a specific point with intensity that depends on direction

Lecture Overview

- ▶ **Light Sources**
 - ▶ Directional Lights
 - ▶ Point Lights
 - ▶ Spot Lights

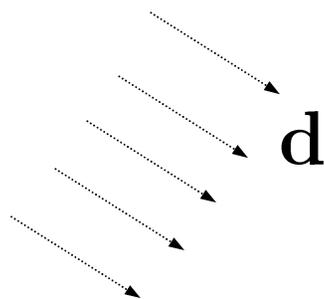
Directional Light

- ▶ Light from a distant source
 - ▶ Light rays are parallel
 - ▶ Direction and intensity are the same everywhere
 - ▶ As if the source were infinitely far away
 - ▶ Good approximation of sunlight
- ▶ Specified by a unit length direction vector, and a color

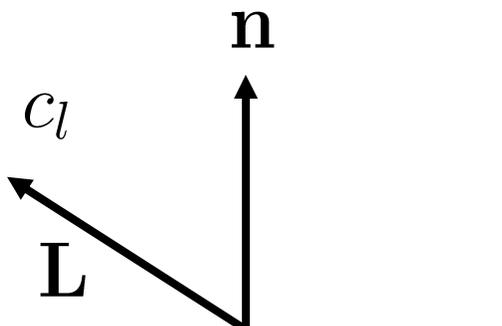


c_{src}

Light source



\mathbf{d}



c_l

\mathbf{L}

\mathbf{n}

Receiving surface

$$\mathbf{L} = -\mathbf{d}$$

$$c_l = c_{src}$$

Lecture Overview

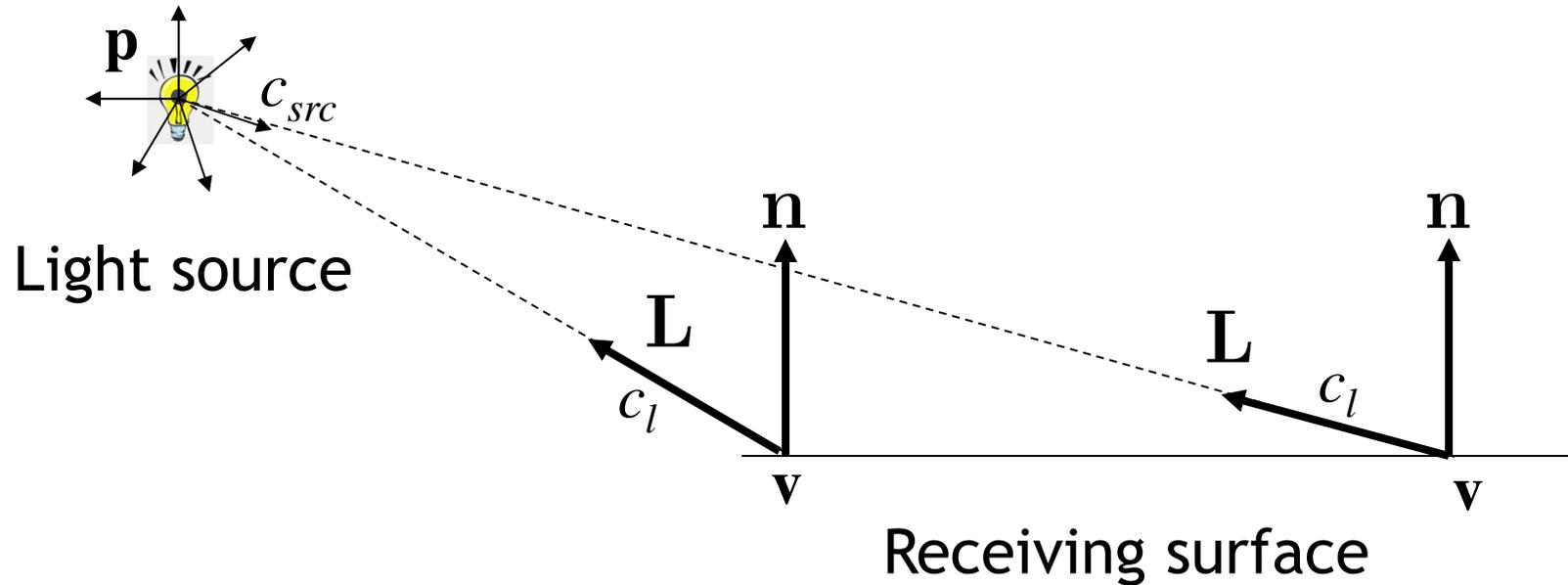
- ▶ **Light Sources**
 - ▶ Directional Lights
 - ▶ **Point Lights**
 - ▶ Spot Lights

Point Lights

- ▶ Similar to light bulbs
- ▶ Infinitely small point radiates light equally in all directions
 - ▶ Light vector varies across receiving surface
 - ▶ What is light intensity over distance proportional to?
 - ▶ Intensity drops off proportionally to the inverse square of the distance from the light
 - ▶ Reason for inverse square falloff:
Surface area A of sphere:
 $A = 4 \pi r^2$



Point Light Math



At any point \mathbf{v} on the surface:

$$\mathbf{L} = \frac{\mathbf{p} - \mathbf{v}}{\|\mathbf{p} - \mathbf{v}\|}$$

Attenuation:

$$c_l = \frac{c_{src}}{\|\mathbf{p} - \mathbf{v}\|^2}$$

Light Attenuation

- ▶ Adding constant factor k to denominator for better control
- ▶ Quadratic attenuation: $k*(p-v)^2$
 - ▶ Most computationally expensive, most physically correct
- ▶ Linear attenuation: $k*(p-v)$
 - ▶ Less expensive, less accurate
- ▶ Constant attenuation: k
 - ▶ Fastest computation, least accurate

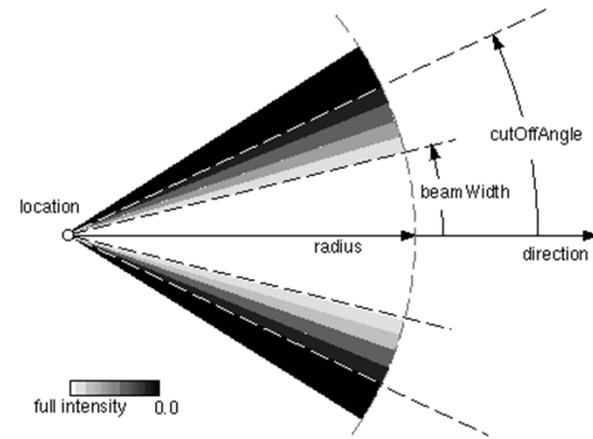
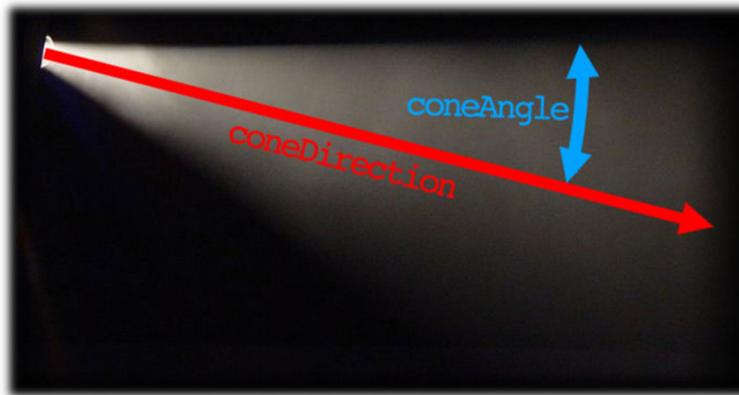
Lecture Overview

- ▶ **Light Sources**
 - ▶ Directional Lights
 - ▶ Point Lights
 - ▶ **Spot Lights**



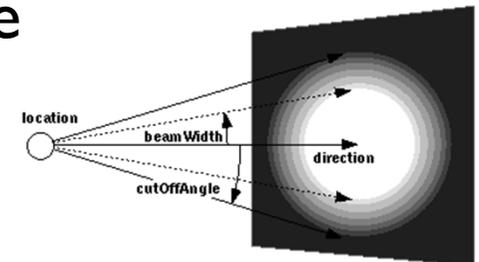
Spotlights

- ▶ Like point light, but intensity depends on direction

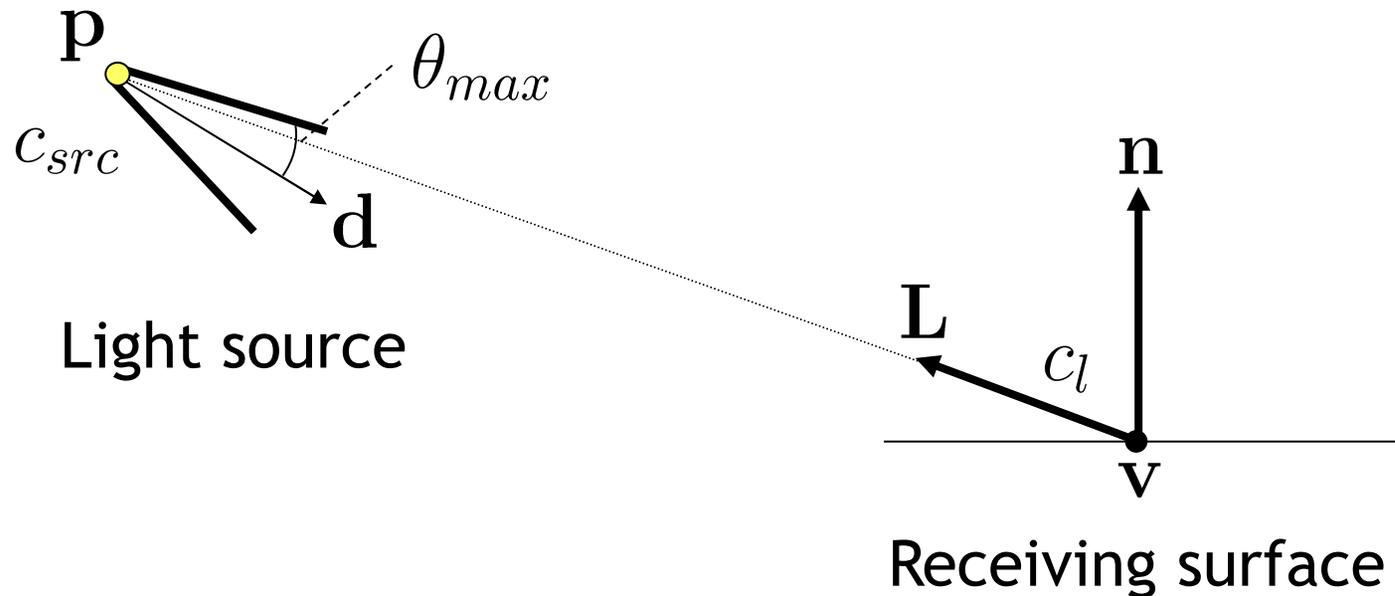


Parameters

- ▶ Position: location of light source
- ▶ Cone direction d : center axis of light source
- ▶ Intensity falloff:
 - ▶ Beam width (cone angle θ_{max})
 - ▶ The way the light tapers off at the edges of the beam (cosine exponent f)



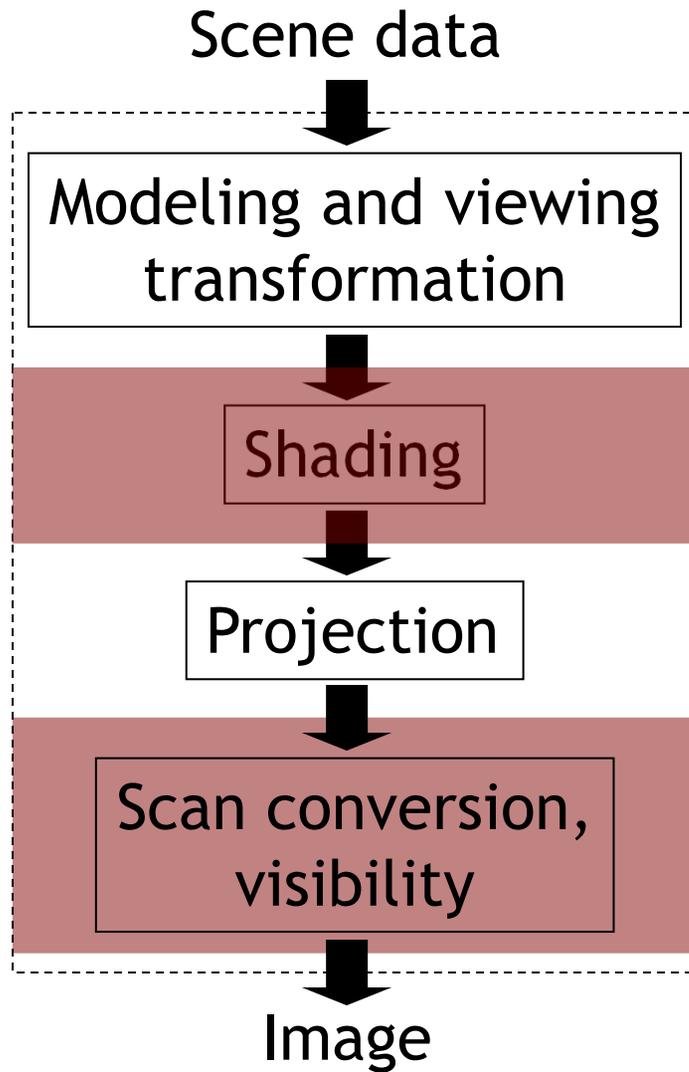
Spotlights



$$\mathbf{L} = \frac{\mathbf{p} - \mathbf{v}}{\|\mathbf{p} - \mathbf{v}\|}$$

$$c_l = \begin{cases} 0 & \text{if } -\mathbf{L} \cdot \mathbf{d} \leq \cos(\theta_{max}) \\ c_{src} (-\mathbf{L} \cdot \mathbf{d})^f & \text{otherwise} \end{cases}$$

Rendering Pipeline



- Place object in 3D space
- Determine colors of vertices
 - Per vertex shading
- Map triangles to 2D
- Draw triangles
 - Per pixel shading

Vertex Shader

```
#version 150

uniform mat4 camera;
uniform mat4 model;

in vec3 vert;
in vec2 vertTexCoord;
in vec3 vertNormal;

out vec3 fragVert;
out vec2 fragTexCoord;
out vec3 fragNormal;

void main()
{
    // Pass some variables to the fragment shader
    fragTexCoord = vertTexCoord;
    fragNormal = vertNormal;
    fragVert = vert;

    // Apply all matrix transformations to vert
    gl_Position = camera * model * vec4(vert, 1);
}
```

Fragment Shader for Diffuse Reflection

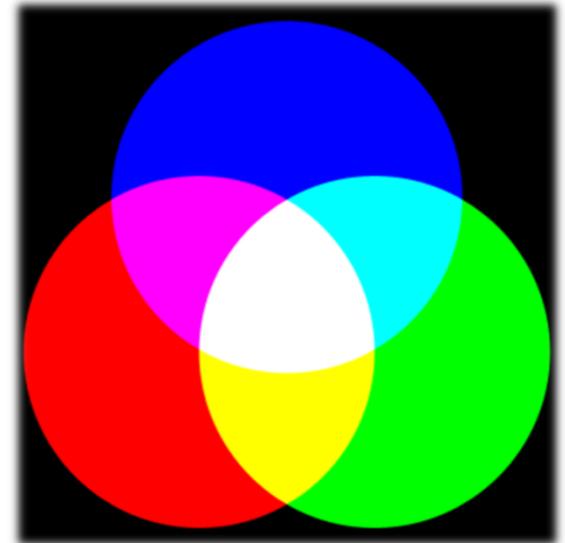
```
#version 150

uniform mat4 model;
uniform sampler2D tex;

uniform struct Light
{
    vec4 position; // if w component=0 it's directional
    vec3 intensities; // a.k.a the color of the light
    float attenuation; // only needed for point and spotlights
    float ambientCoefficient;
    float coneAngle; // only needed for spotlights
    vec3 coneDirection; // only needed for spotlights
    float exponent; // cosine exponent for how light tapers off
} light;

in vec2 fragTexCoord;
in vec3 fragNormal;
in vec3 fragVert;

out vec4 finalColor;
```



Fragment Shader Part 2

```
void main()
{
    // calculate normal in world coordinates
    mat3 normalMatrix = transpose(inverse(mat3(model)));
    vec3 normal = normalize(normalMatrix * fragNormal);

    // calculate the location of this fragment (pixel) in world coordinates
    vec3 fragPosition = vec3(model * vec4(fragVert, 1));

    // calculate the vector from this pixels surface to the light source
    vec3 surfaceToLight = light.position - fragPosition;

    // calculate the cosine of the angle of incidence
    float brightness = dot(normal, surfaceToLight) / (length(surfaceToLight) * length(normal));
    brightness = clamp(brightness, 0, 1);

    // calculate final color of the pixel, based on:
    // 1. The angle of incidence: brightness
    // 2. The color/intensities of the light: light.intensities
    // 3. The texture and texture coord: texture(tex, fragTexCoord)
    vec4 surfaceColor = texture(tex, fragTexCoord);
    finalColor = vec4(brightness * light.intensities * surfaceColor.rgb, surfaceColor.a);
}
```

Lighting with GLSL

- ▶ Tutorial for diffuse lighting with a point light
 - ▶ <http://www.tomdalling.com/blog/modern-opengl/06-diffuse-point-lighting/>

