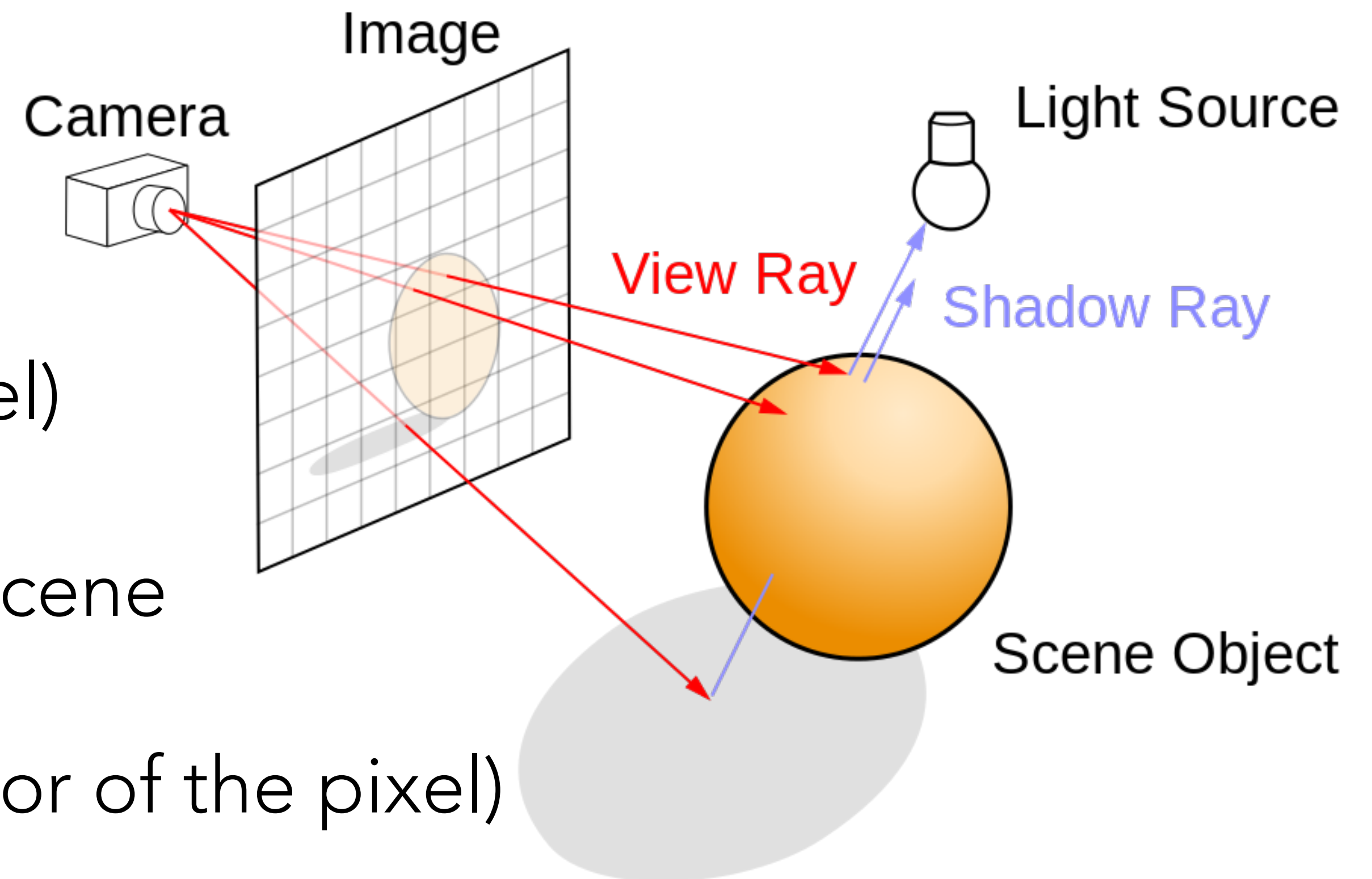# Ray Tracing

University
of Victoria

Computer Science

Basic Raytracing

1. Generation of Rays (one per pixel)

2. Intersection with objects in the scene
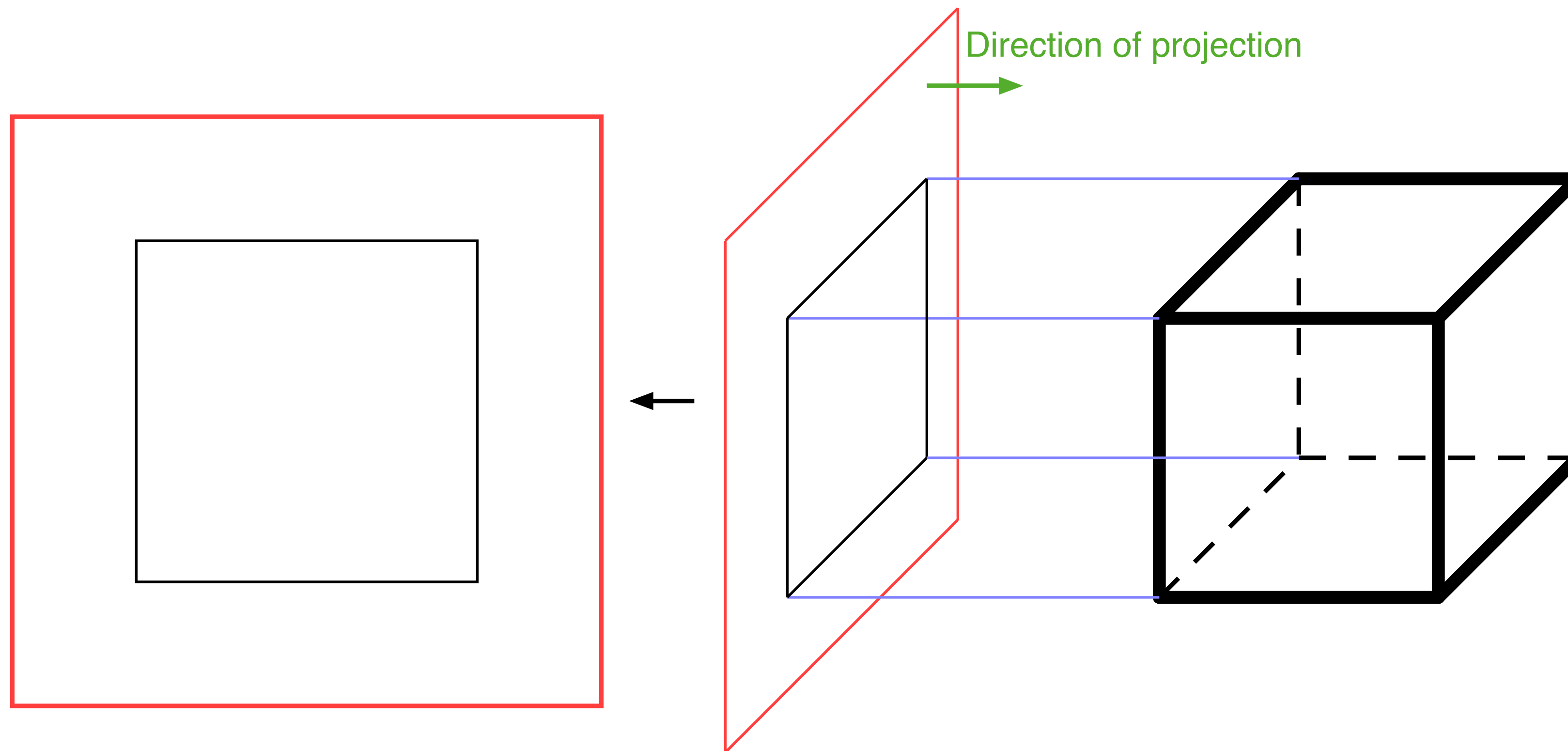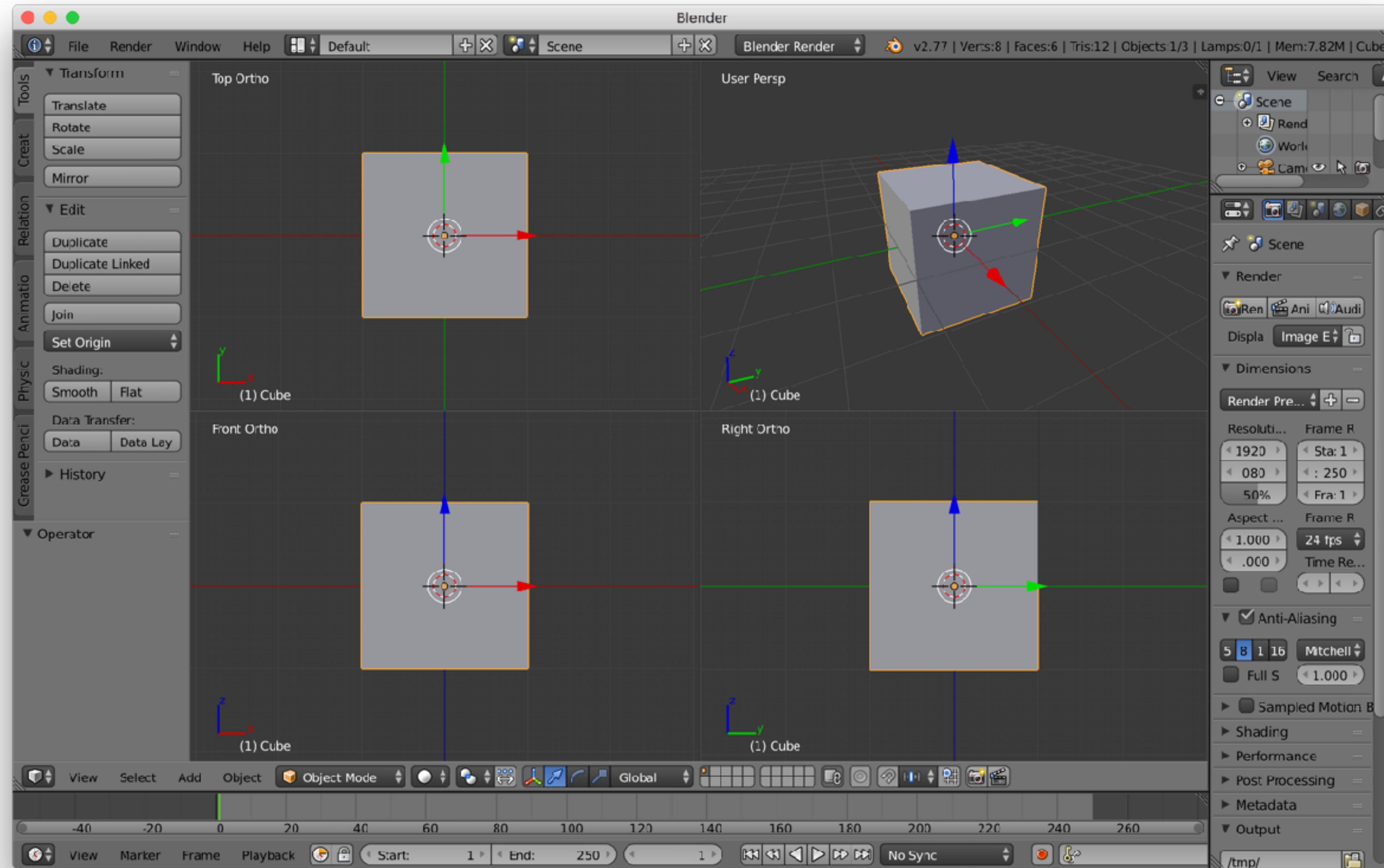
3. Shading (computation of the color of the pixel)

Image
Camera
Light Source
View Ray
Shadow Ray
Scene Object

By Henrik - Own work, GFDL, https://commons.wikimedia.org/w/index.php?curid=3869326

University of Victoria | Computer Science

Projection - Parallel

Direction of projection

University of Victoria | Computer Science

Commonly used in modeling tools

Perspective Projection
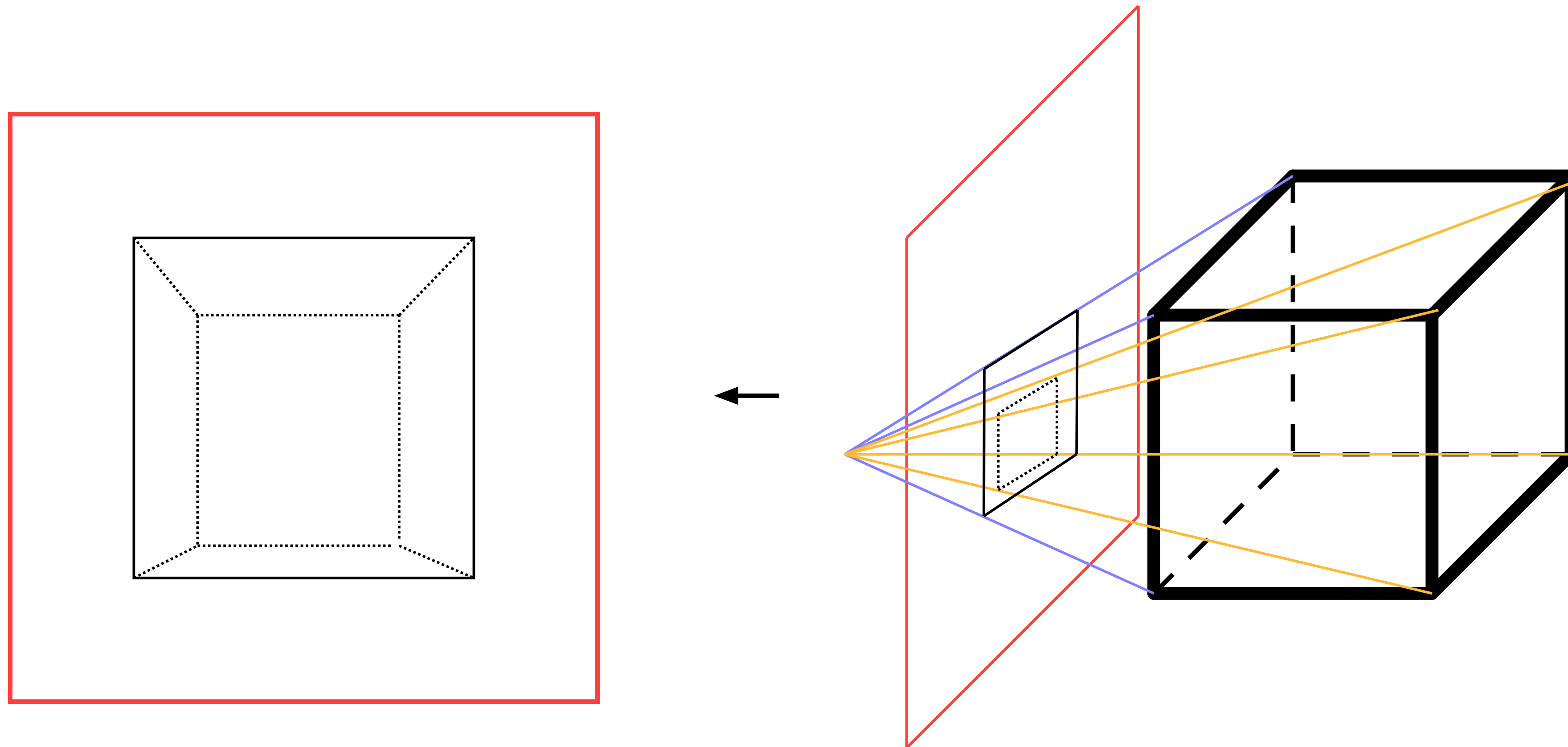
Each ray has a different direction!

University of Victoria | Computer Science

Two and Three Point Perspectives
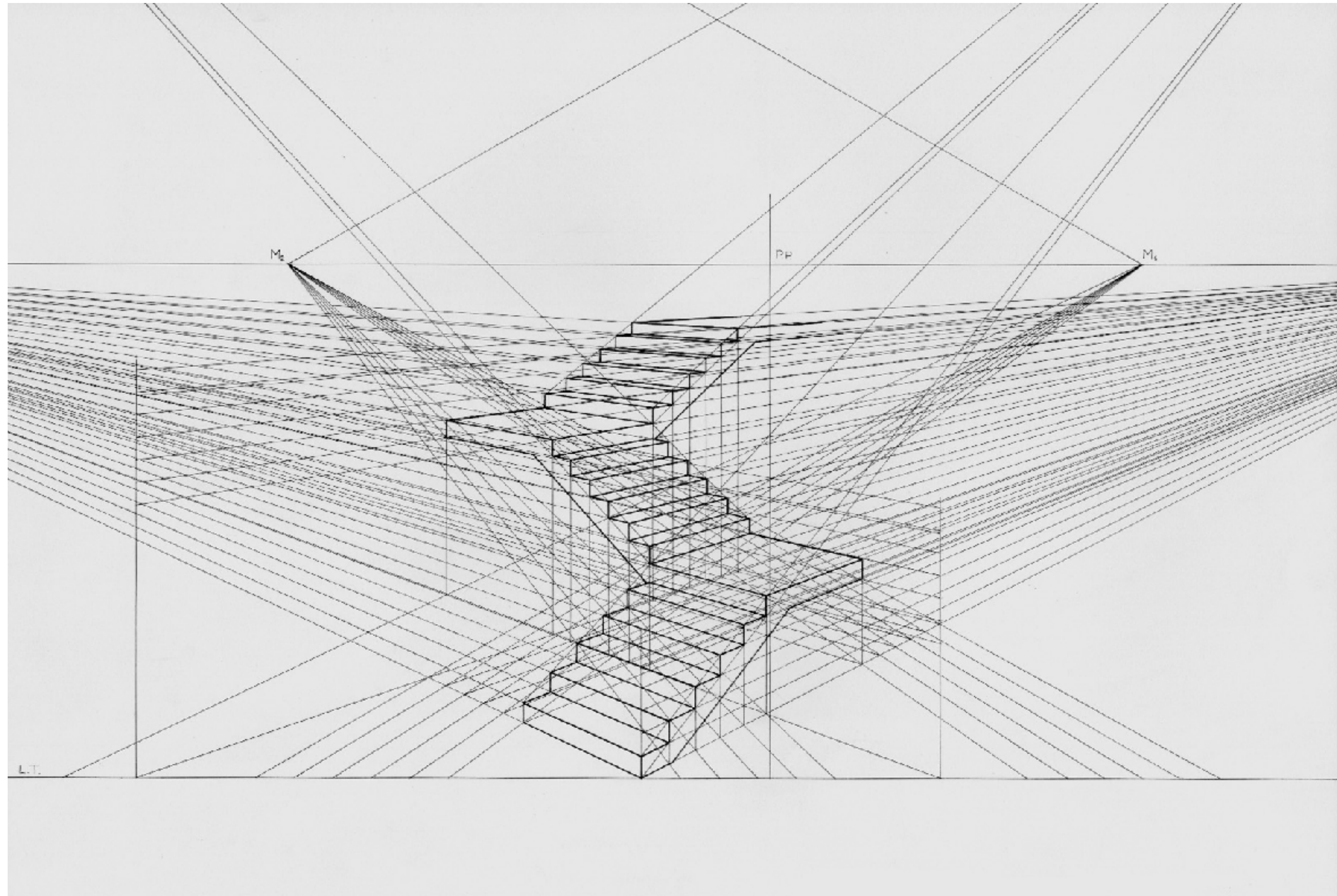
1. Compute Rays
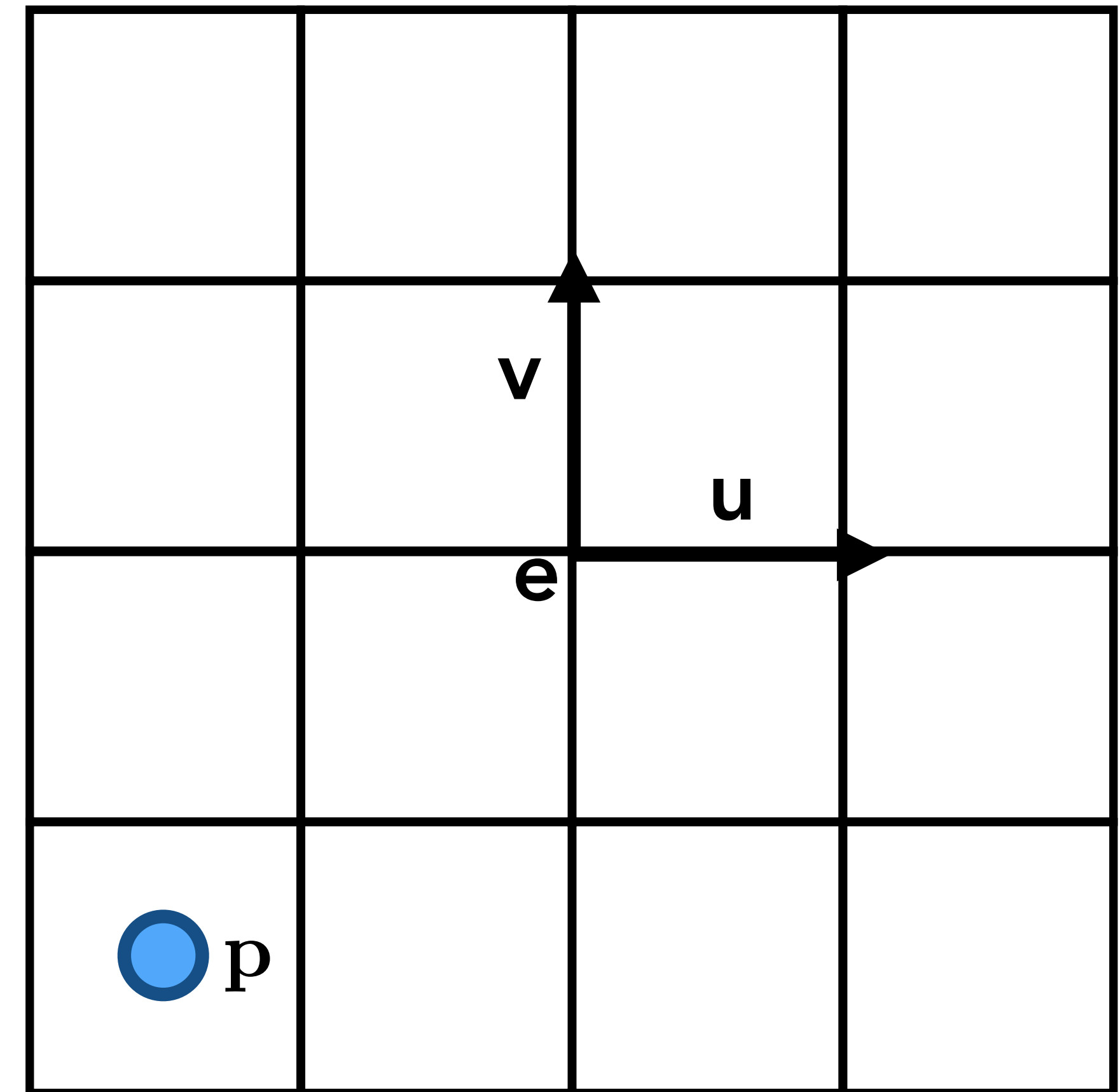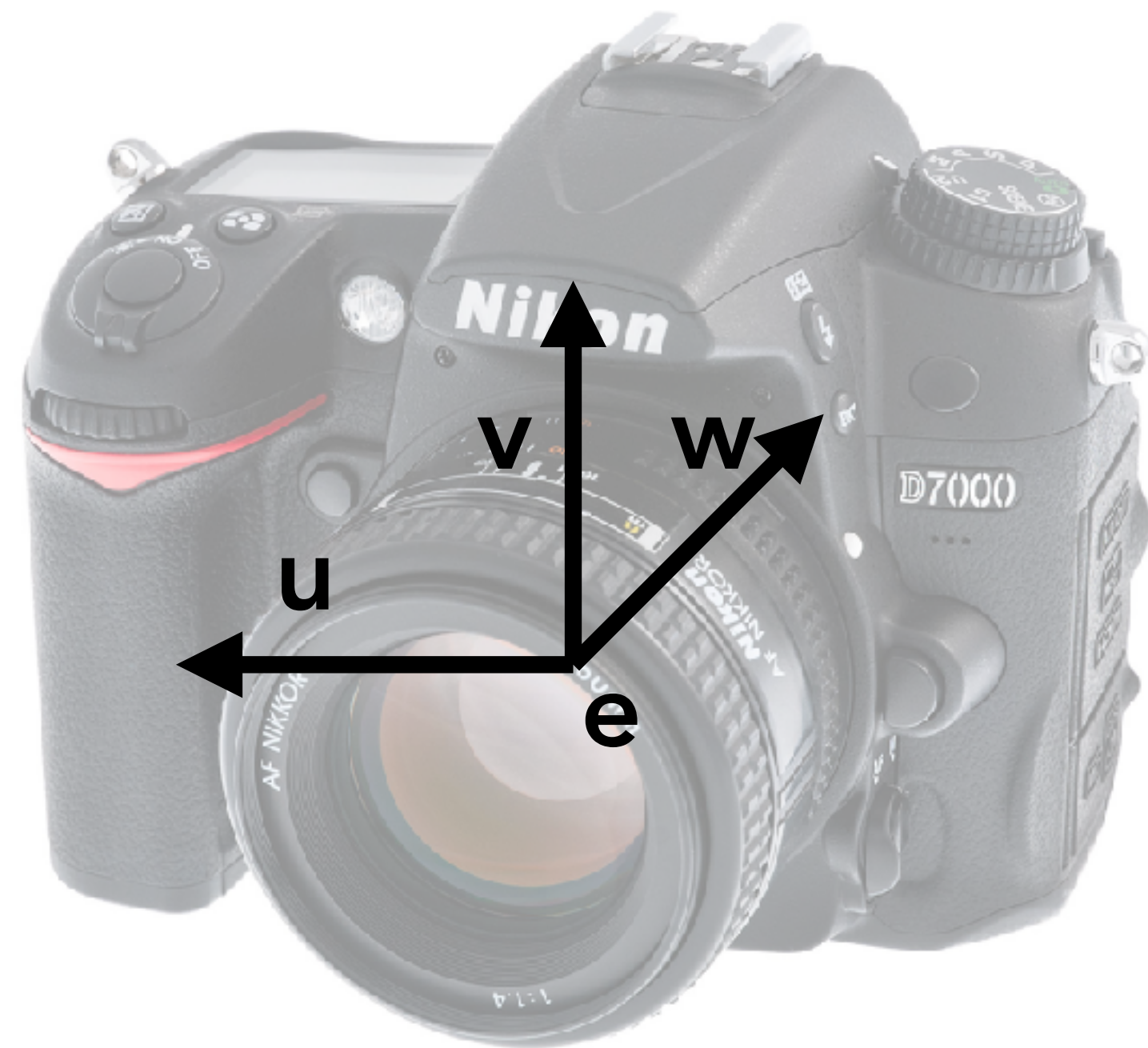
$$\mathbf{p} = \mathbf{e} + u\mathbf{u} + v\mathbf{v} + w\mathbf{w}$$

e is the origin of the reference system
p is the center of the pixel

Orthographic

Direction of projection

v   w

u

e

v

u

e

w

v

u   p
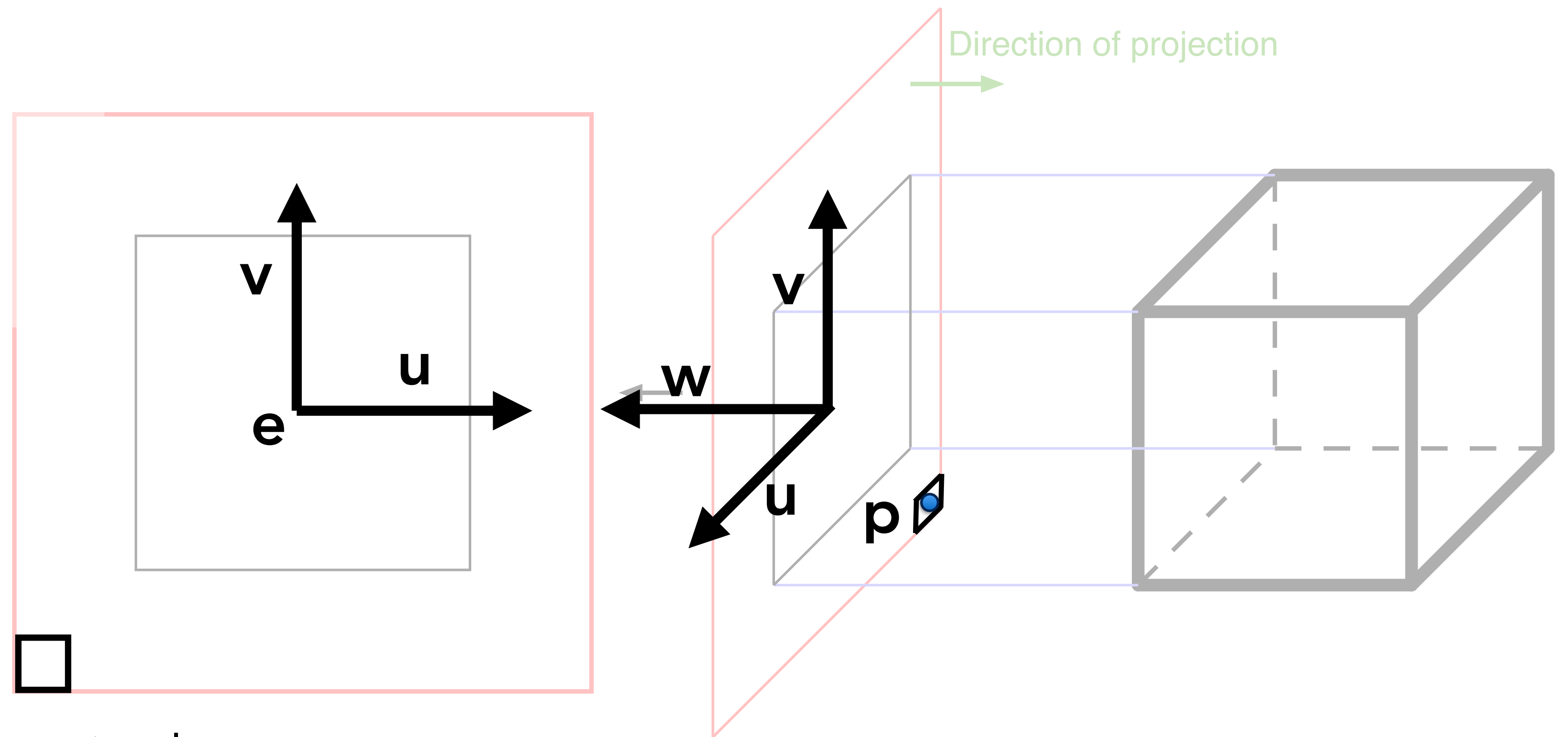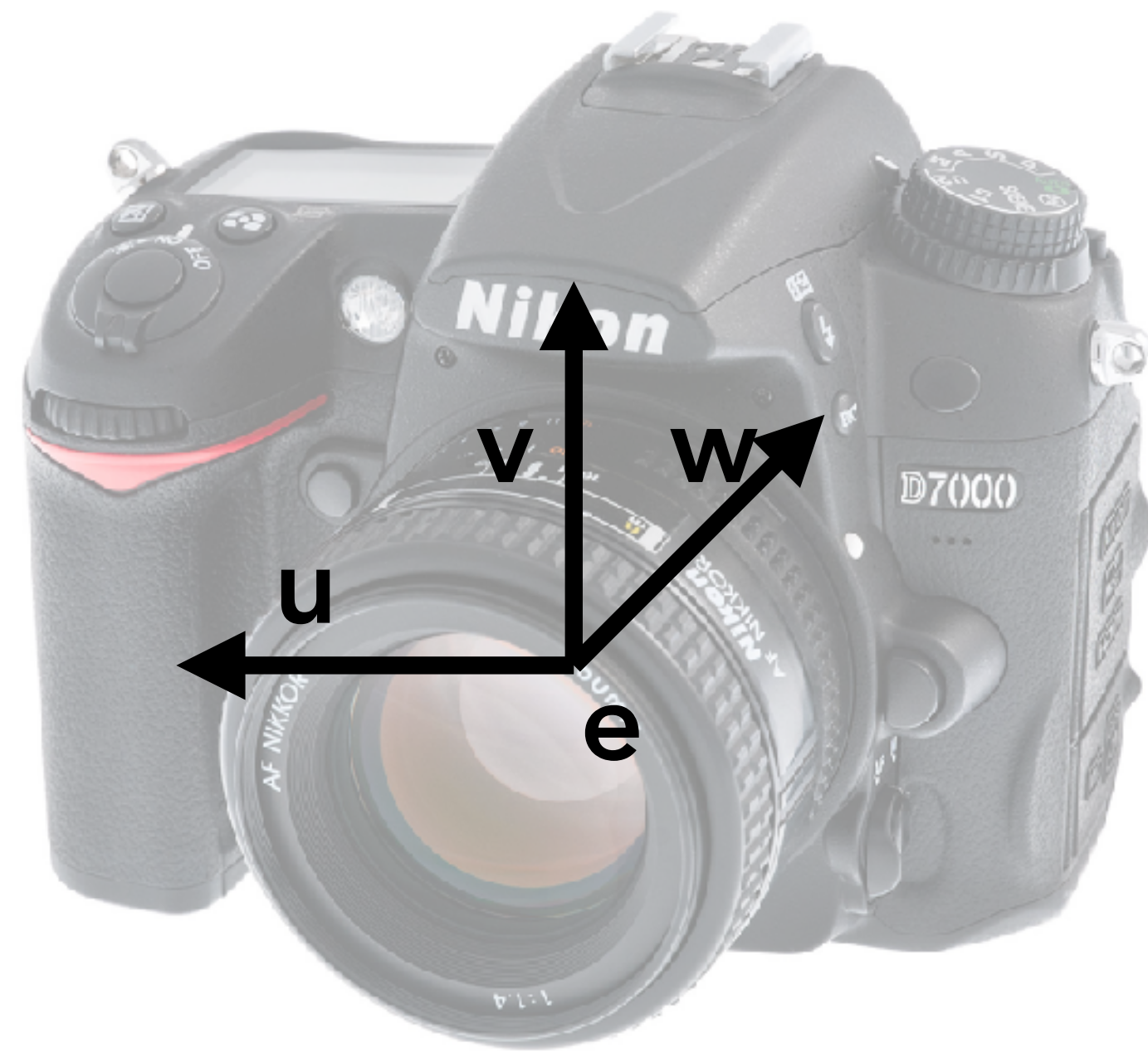
- For the ray assigned to pixel **p**:
  - Origin: **p**
  - Direction: -**w**

University of Victoria | Computer Science

Perspective
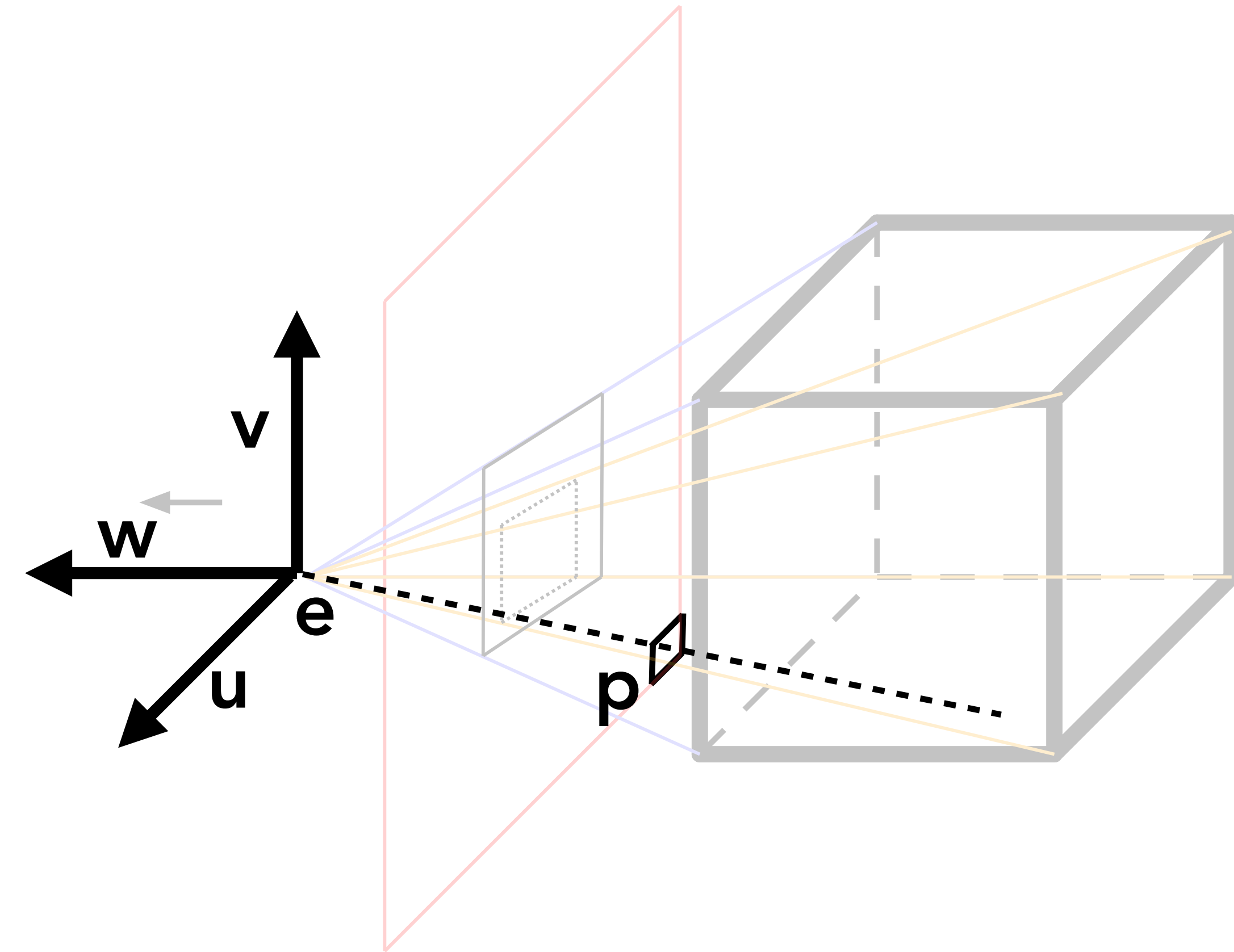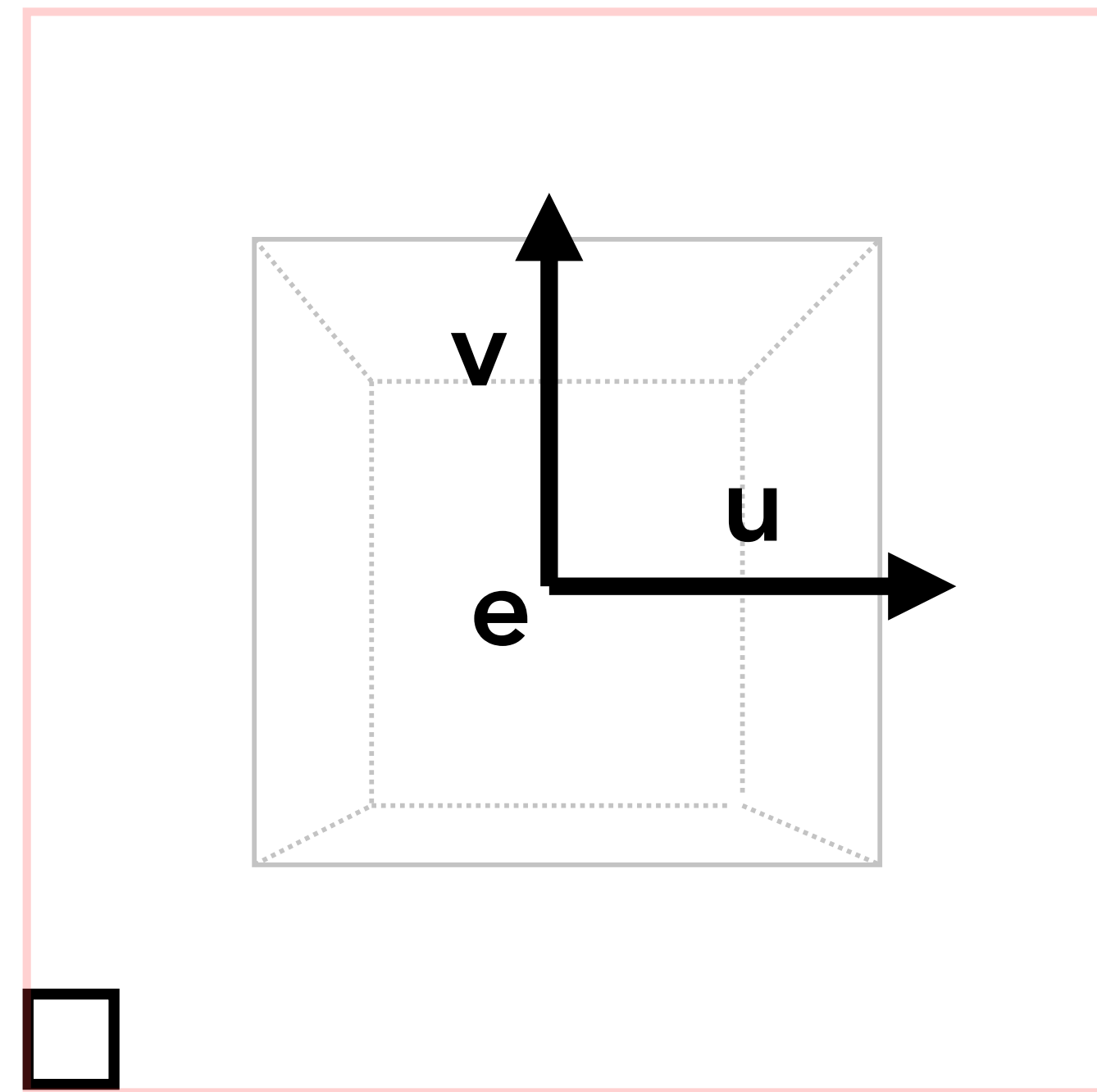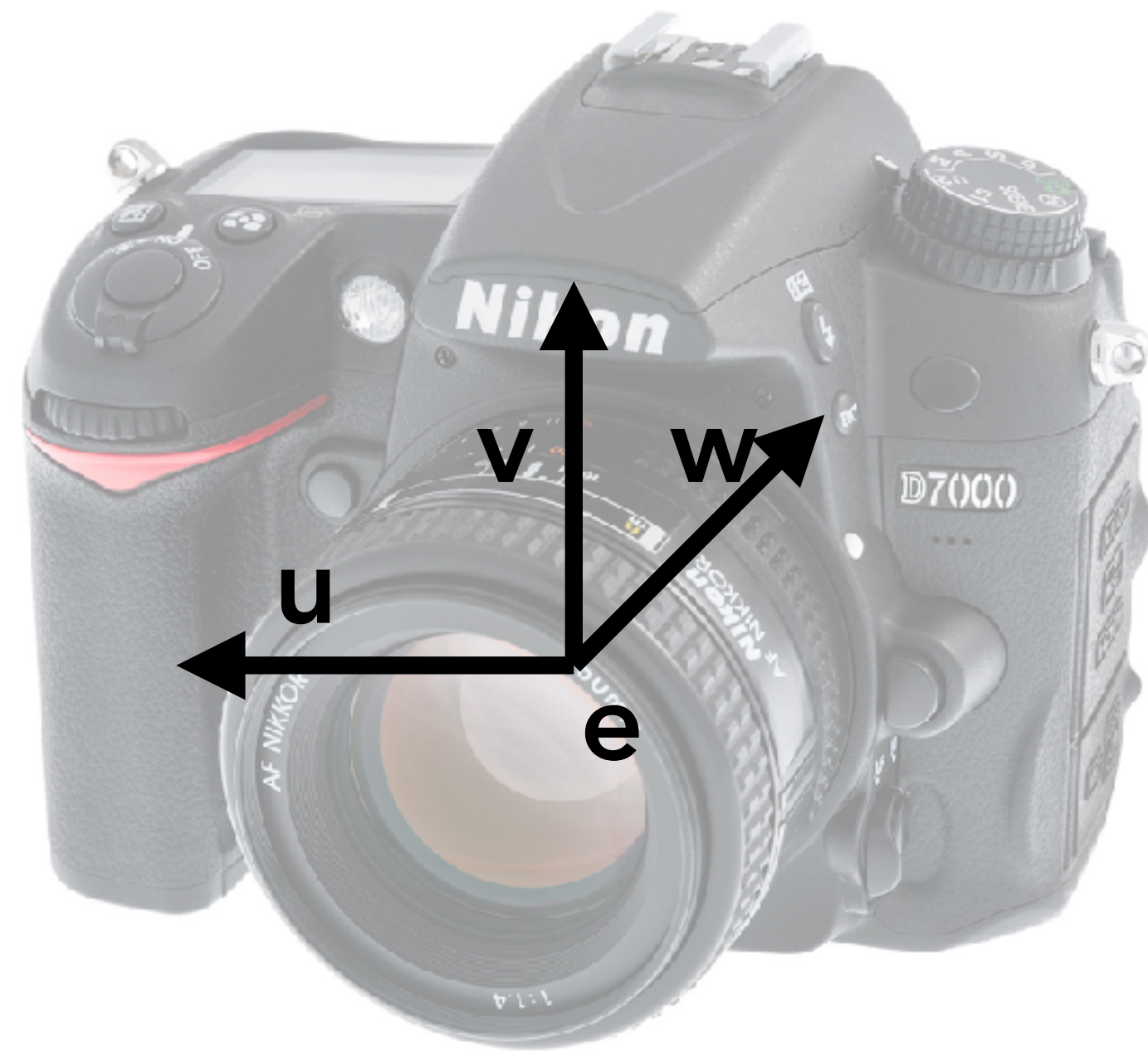
- For the ray assigned to pixel **p**:
  - Origin: **e**
  - Direction: **p - e**

University of Victoria | Computer Science

Basic Raytracing

1. Generation of rays (one per pixel)

2. Intersection with objects in the scene

3. Shading (computation of the color of the pixel)

By Henrik - Own work, GFDL, https://commons.wikimedia.org/w/index.php?curid=3869326

University of Victoria | Computer Science
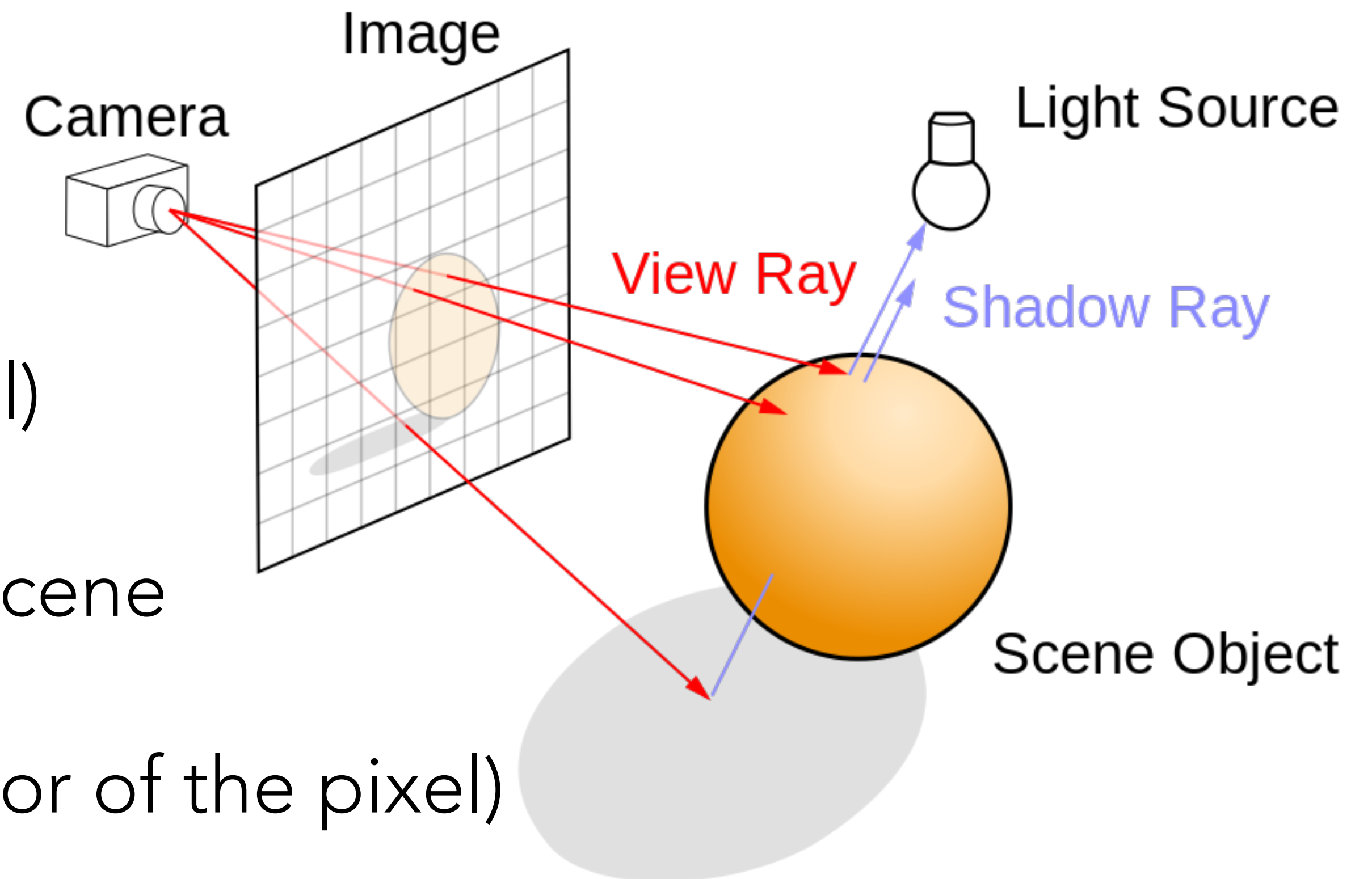
Intersections

- This is an expensive operation

- There is a large literature, a good overview is here: http://www.realtimerendering.com/intersections.html

- We will study two very useful cases:

  - Spheres

  - Triangles (by combining many triangles you can approximate complex surfaces)

University of Victoria | Computer Science

Ray-Sphere Intersection

- We have a ray in explicit form:

$$\mathbf{p}(t) = \mathbf{e} + t\mathbf{d}$$

- and a sphere of radius r and center **c** in implicit form

$$f(\mathbf{p}) = (\mathbf{p} - \mathbf{c}) \cdot (\mathbf{p} - \mathbf{c}) - R^2 = 0$$

- To find the intersection we need to find the solutions of

$$f(\mathbf{p}(t)) = 0$$

University of Victoria | Computer Science

2. Ray-Triangle Intersection

- Explicit parametrization of a triangle with vertices **a**,**b**,**c**:

$$\mathbf{f}(u, v) = \mathbf{a} + u(\mathbf{b} - \mathbf{a}) + v(\mathbf{c} - \mathbf{a})$$

- Explicit ray:

$$\mathbf{p}(t) = \mathbf{e} + t\mathbf{d}$$

- The ray intersects the triangle if a t,u,v exist s.t.:

$$\mathbf{f}(u, v) = \mathbf{p}(t)$$

$$t > 0 \quad 0 \leq u, v \quad u + v \leq 1$$

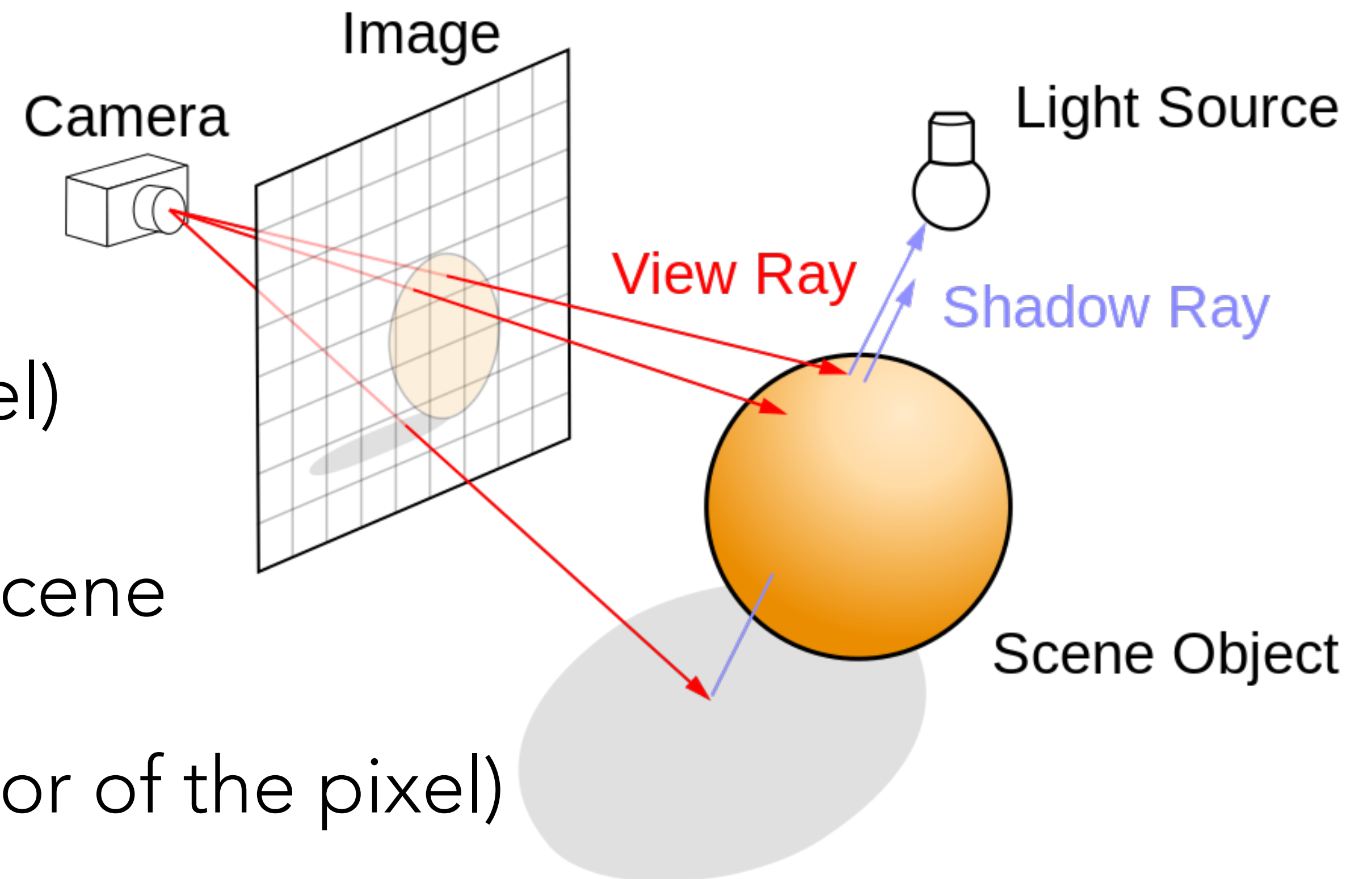University of Victoria | Computer Science

Multiple Objects

- It is simple, intersect it with all of them and only keep the closest intersection

- To speed up computation, you can use a spatial data structure to prune the number of collisions that you need to check

University of Victoria | Computer Science

Basic Raytracing

1. Generation of Rays (one per pixel)

2. Intersection with objects in the scene

3. Shading (computation of the color of the pixel)

By Henrik - Own work, GFDL, https://commons.wikimedia.org/w/index.php?curid=3869326

University of Victoria | Computer Science

Shading

- Modeling accurately the behavior of light is difficult and computationally expensive

- We will use an approximation that is simple and efficient. It is divided in 3 parts:

  - Diffuse (Lambertian) Shading

  - Specular (Blinn-Phong) Shading

  - Ambient Shading

- The three terms will be summed together to obtain the final color

University of Victoria | Computer Science

Diffuse and Specular

Copyright: Marsette Vona

University of Victoria | Computer Science
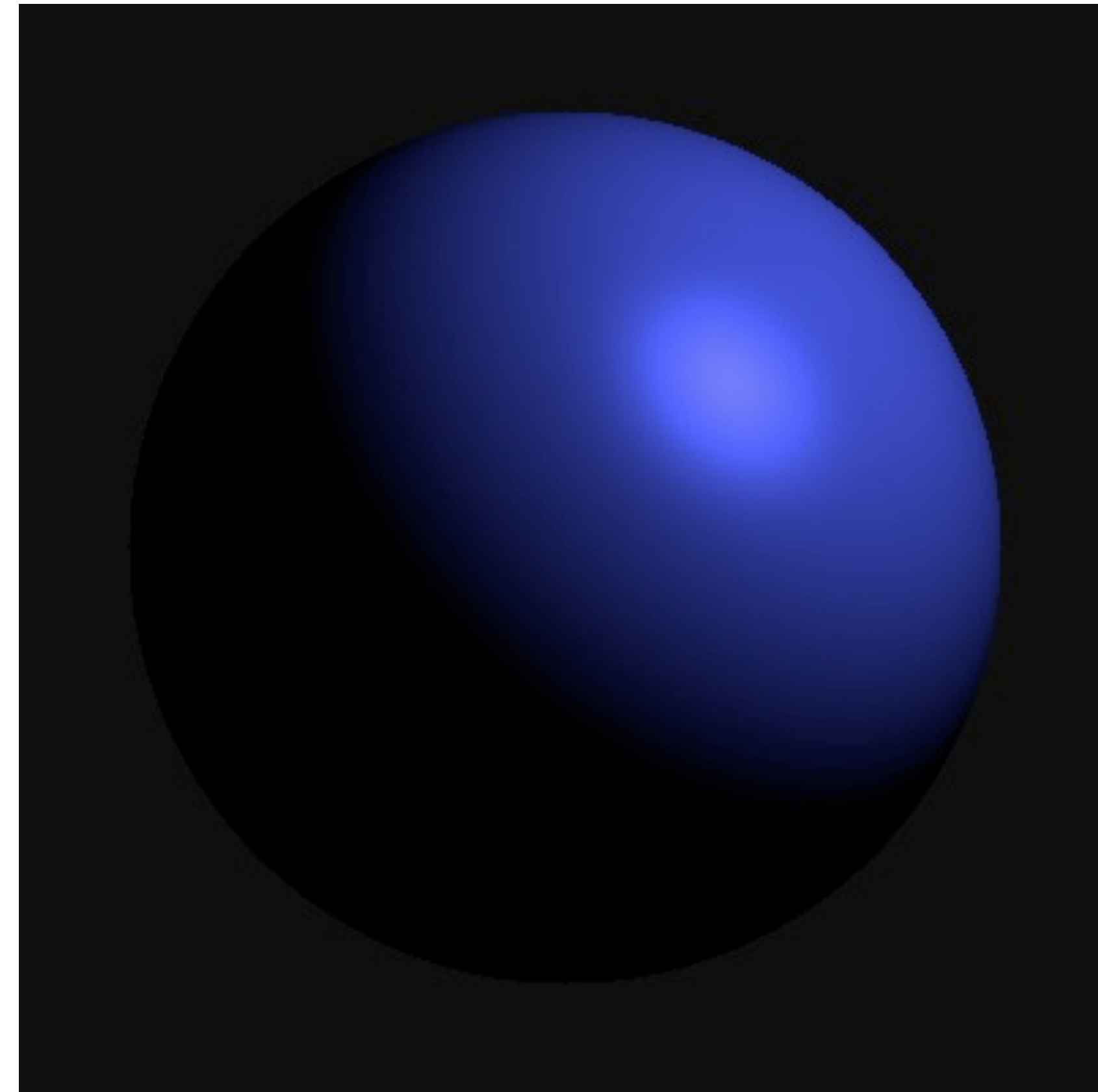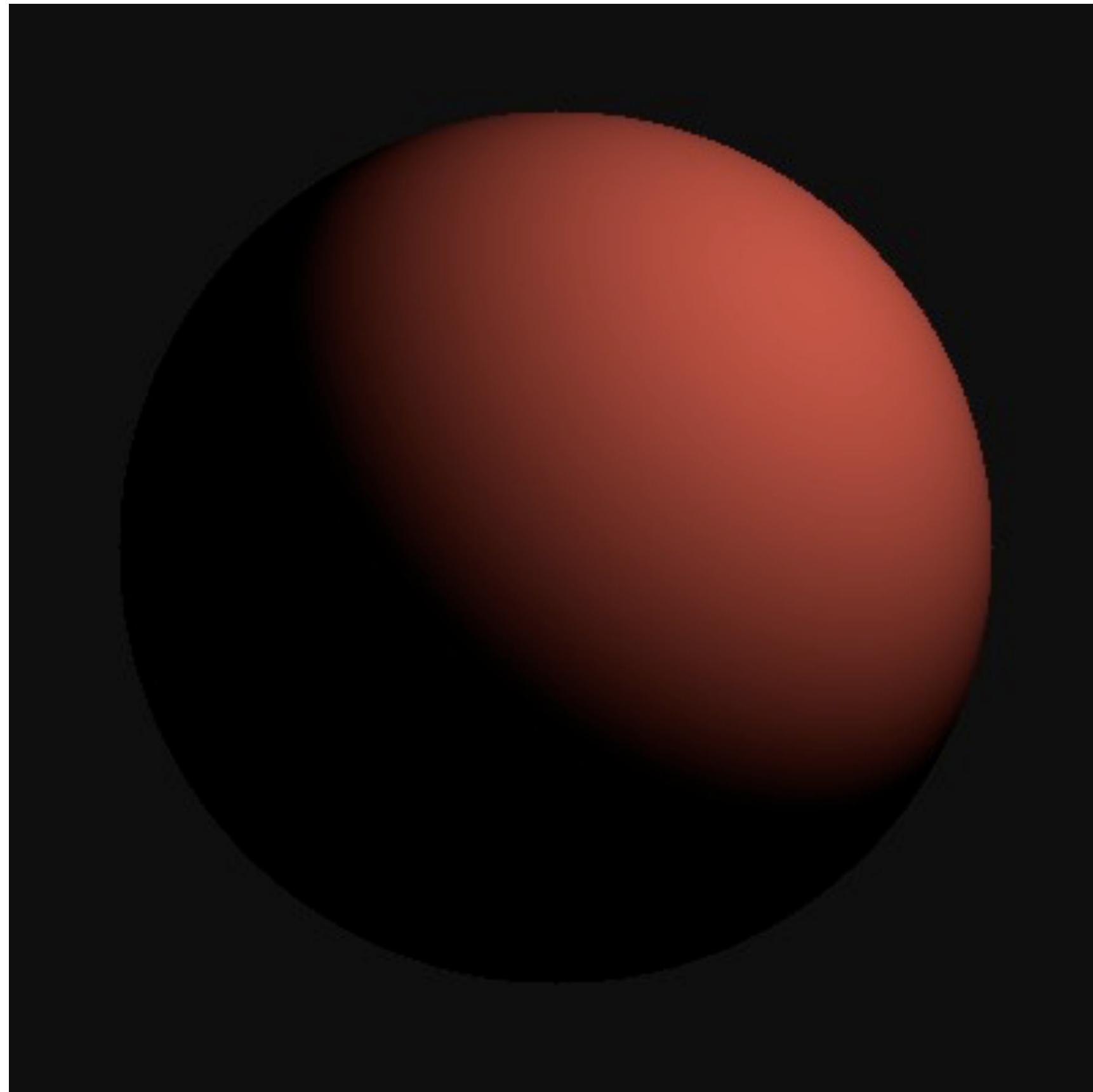
Shading Variables

- The shading depends on the entire scene, the light can bounce, reflect and be absorbed by anything that it encounters

- We will simplify it so that it depends only on:

  - The light direction **l** (a *unit* vector pointing to the light source)

  - The view direction **v** (a *unit* vector pointing toward the camera)

  - The surface normal **n** (a vector perpendicular to the surface at the point of intersection)

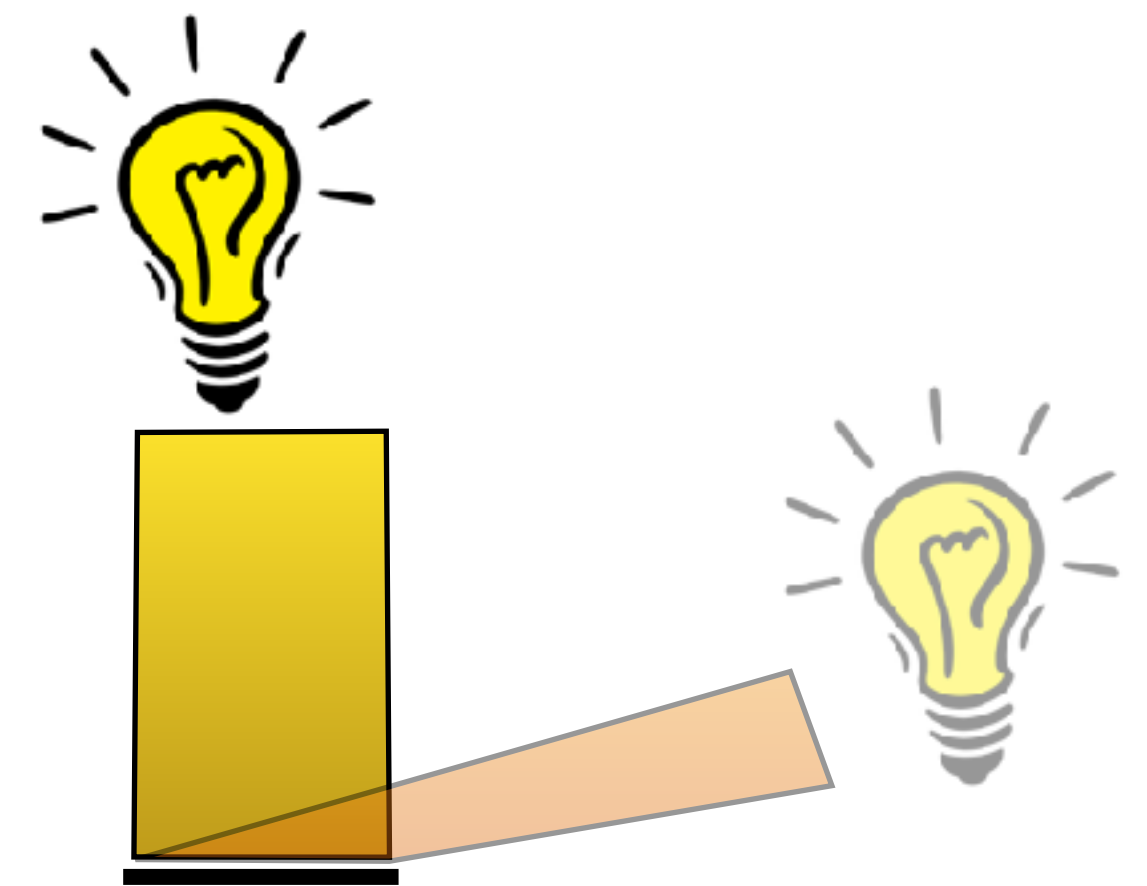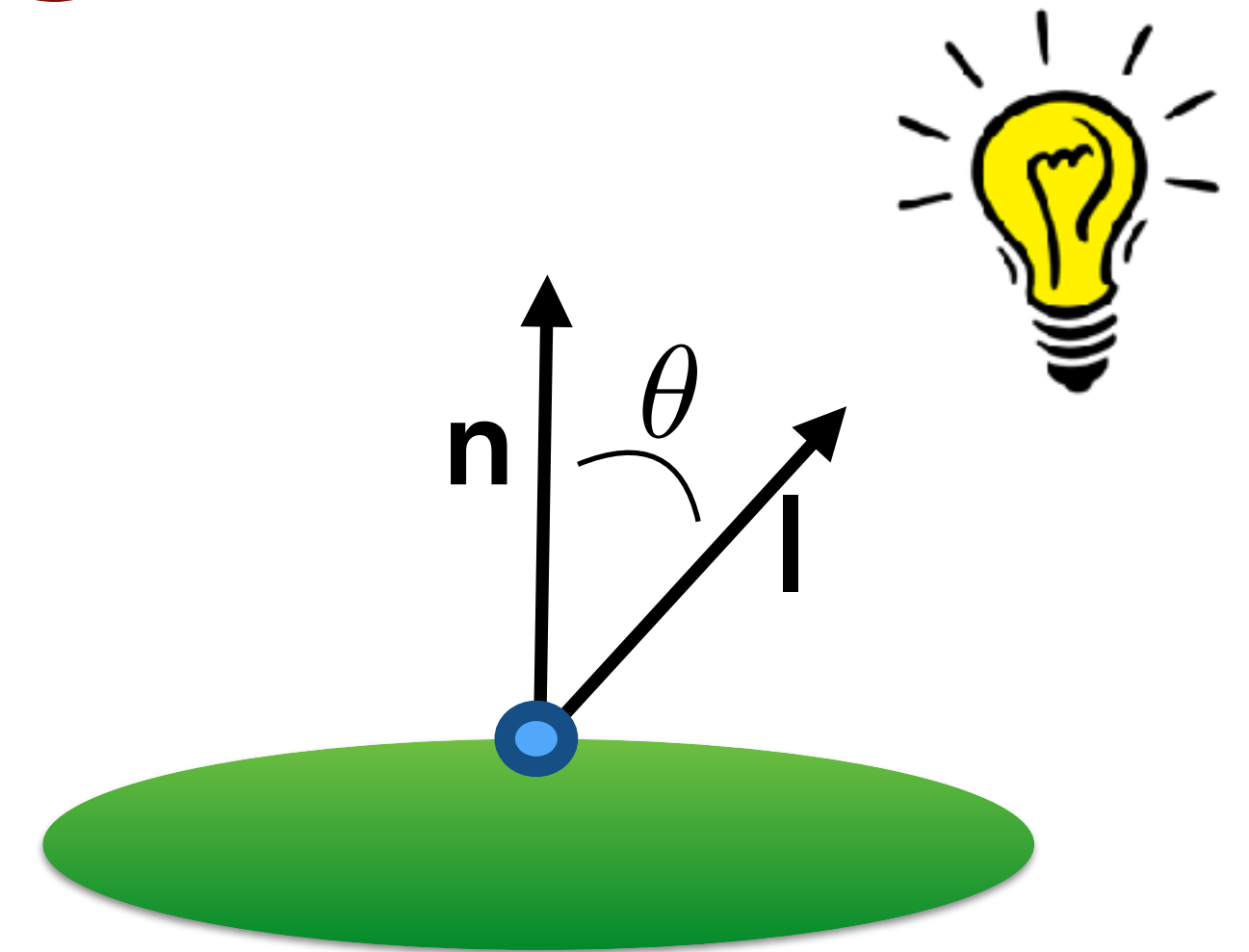University of Victoria | Computer Science
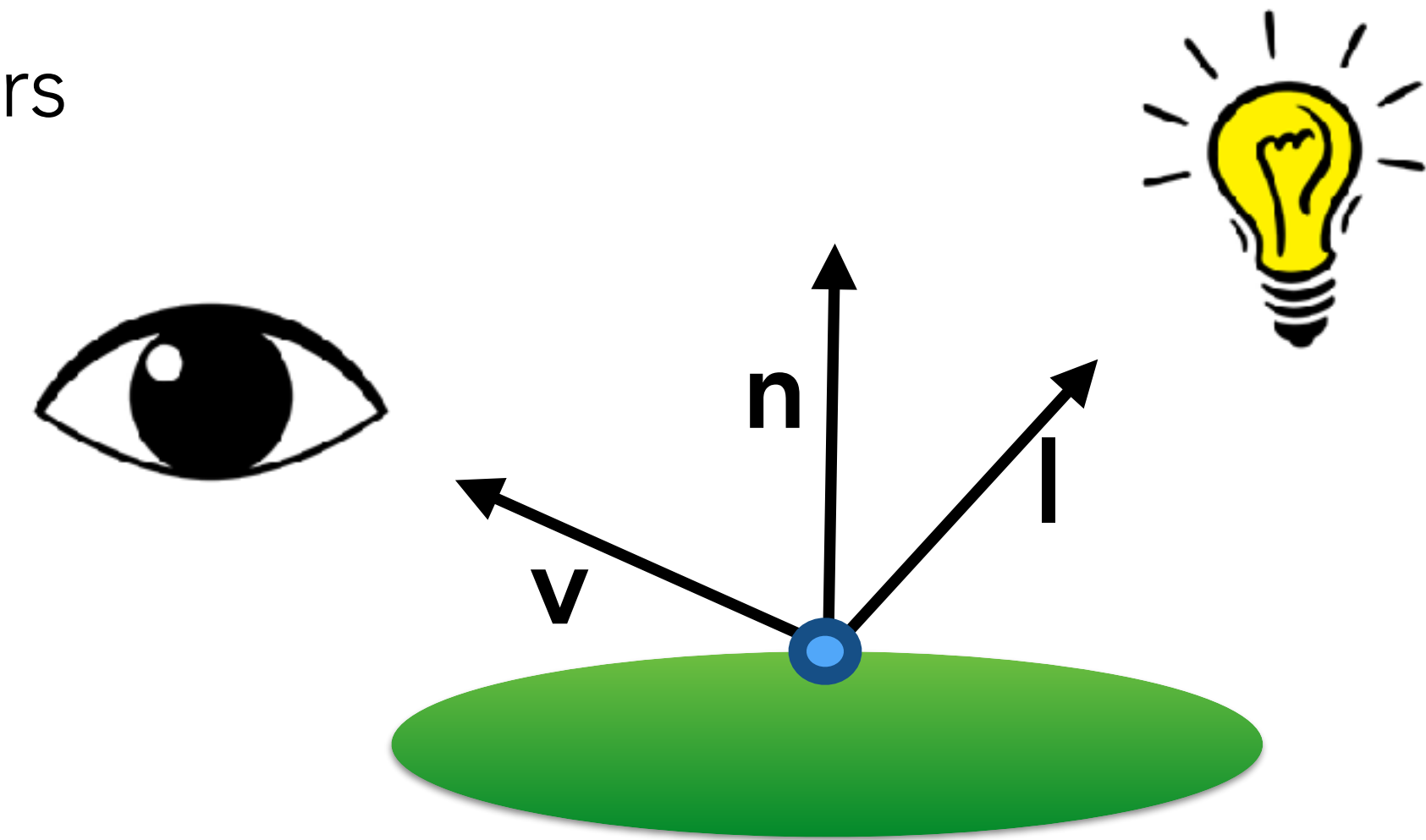
Diffuse Shading

- Lambert (18th century) observed that the amount of energy from a light source that falls on an area of surface depends on the angle of the surface to the light

- To model it, we make the amount of light proportional to the angle θ between the **n** and **l**

diffuse coefficient

$$L = k_d I \max(0, \mathbf{n} \cdot \mathbf{l})$$

intensity of the light

The position of the camera is not used!

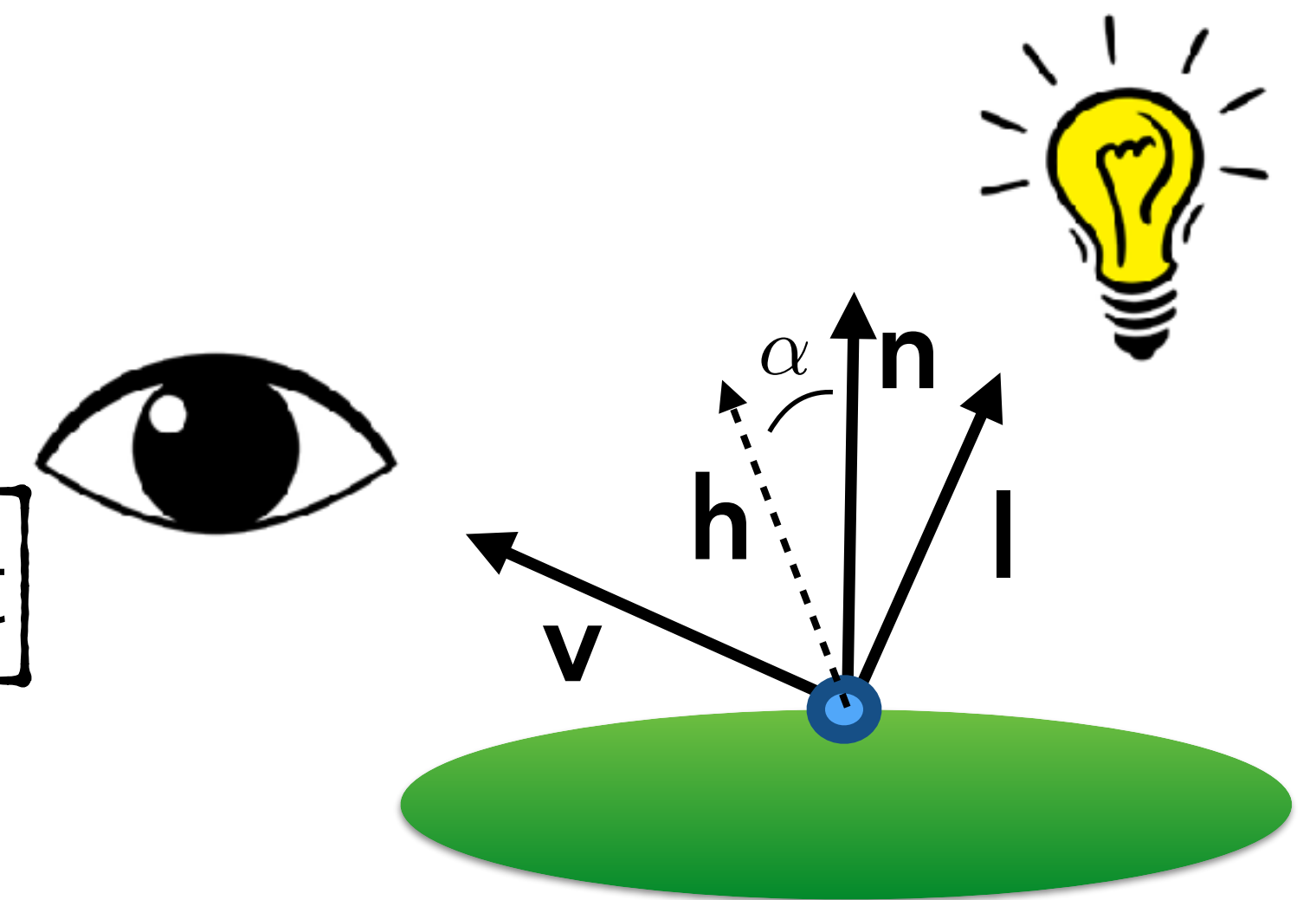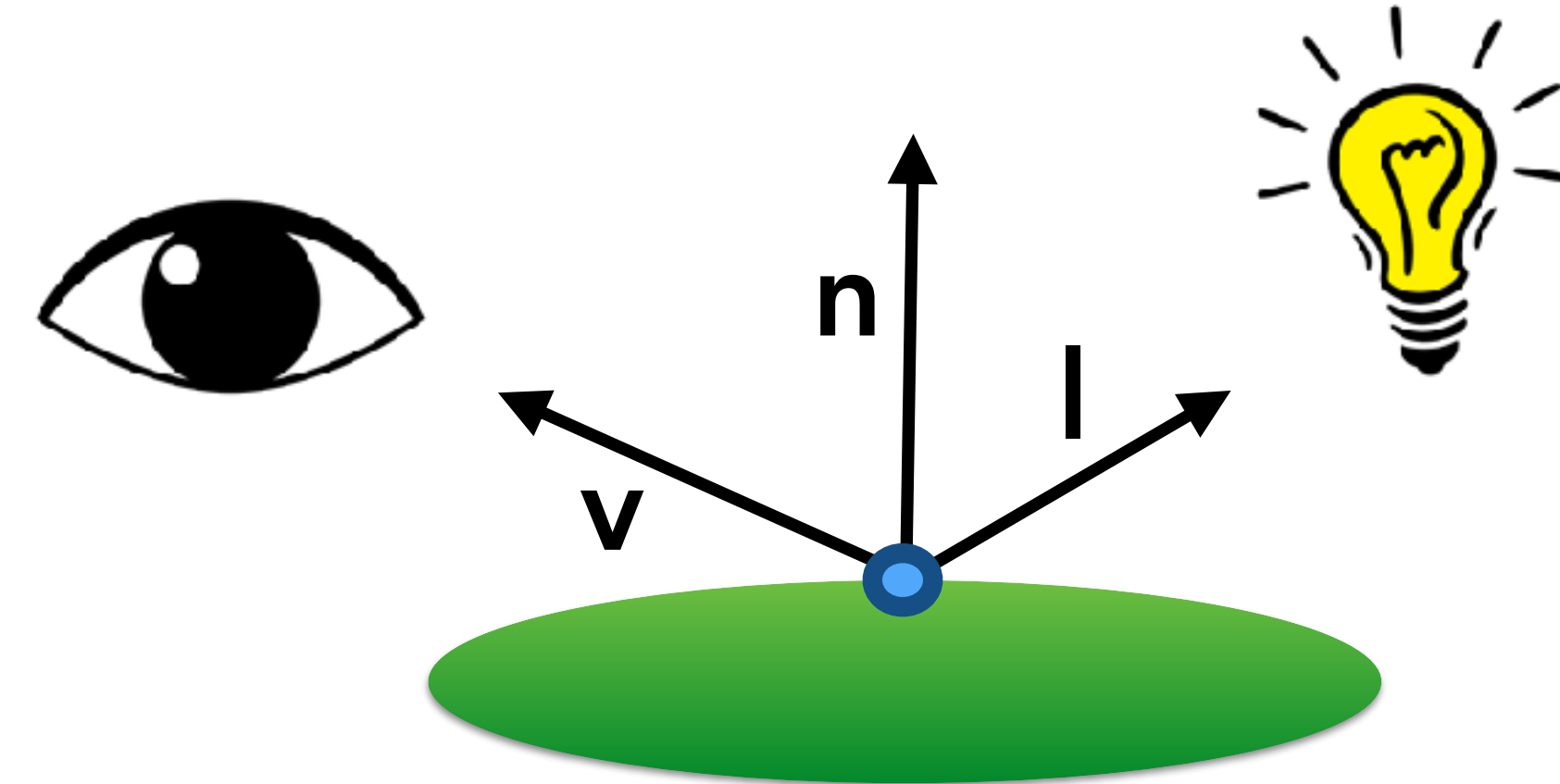University of Victoria | Computer Science

Specular Shading

- Specular highlights depend on the position of the viewer

- A simple and effective model to model them has been proposed by Phong (1975) and refined by Blinn (1976)

- The idea is to produce a reflection that is bright if **v** and **l** are symmetric wrt **n**

- To measure the asymmetry we measure the angle between **h** (the bisector of **v** and **l**) and **n**

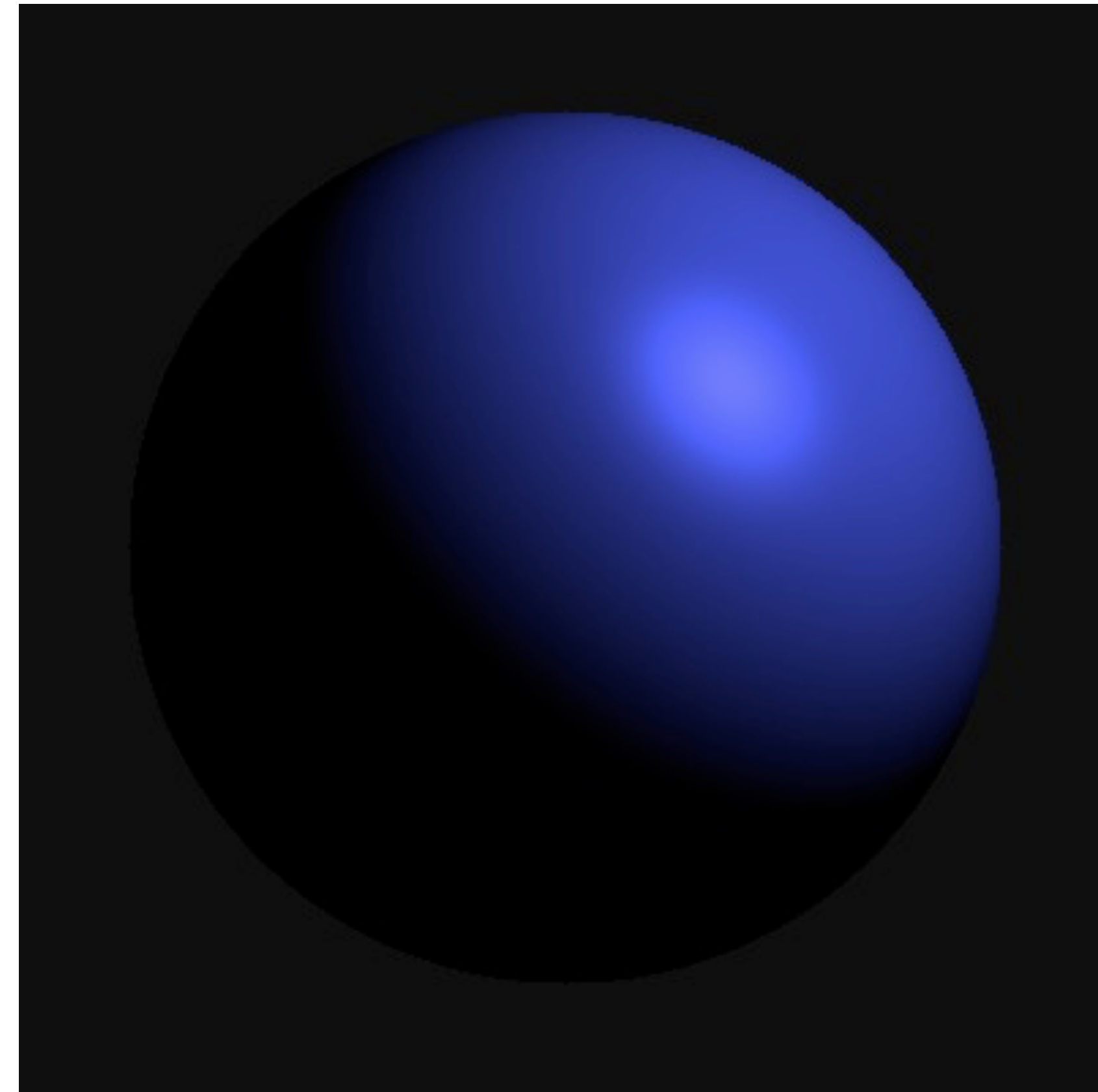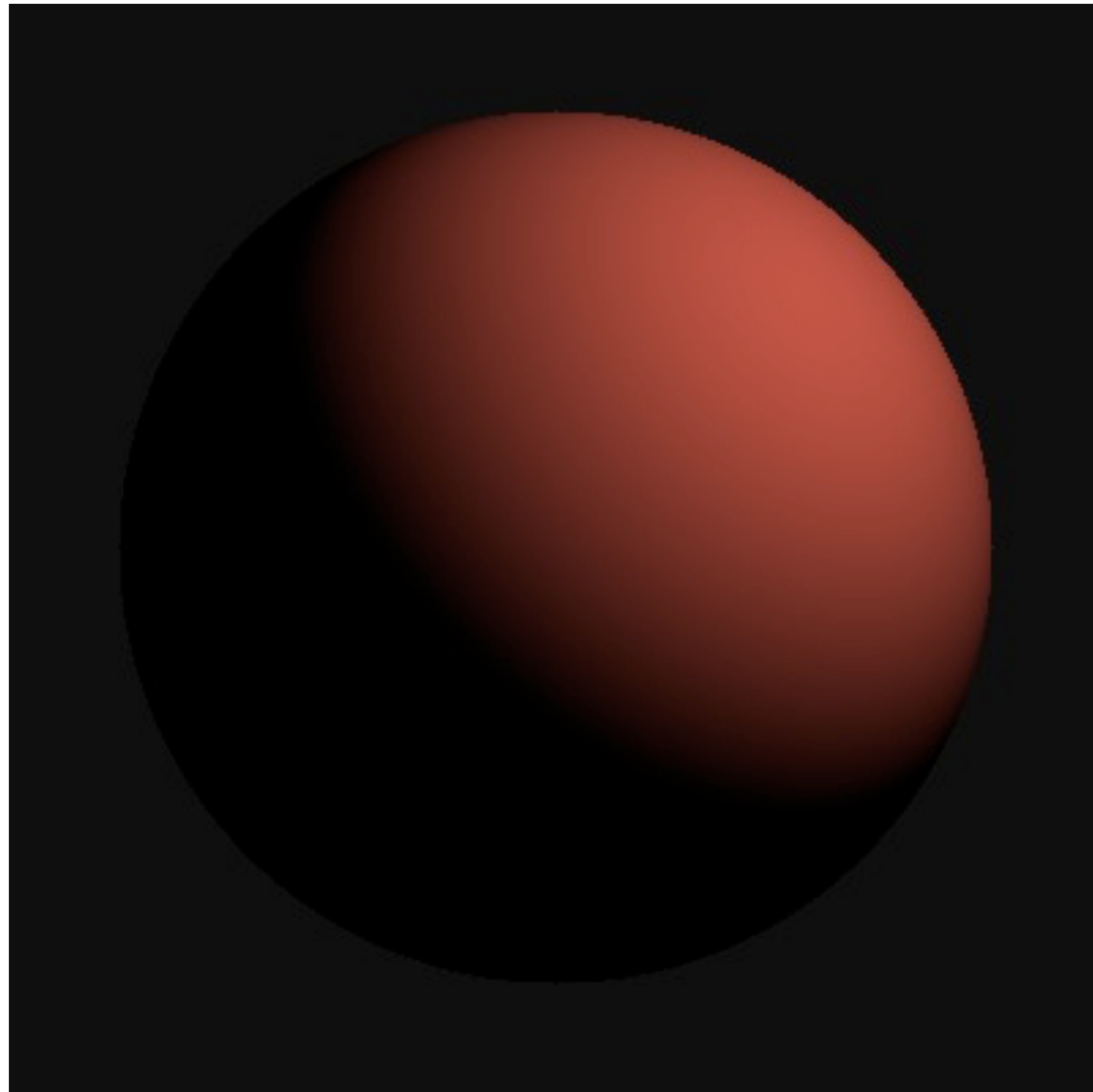$$\mathbf{h} = \frac{\mathbf{v} + \mathbf{l}}{||\mathbf{v} + \mathbf{l}||}$$

Phong exponent

$$L = k_d I \max(0, \mathbf{n} \cdot \mathbf{l}) + k_s I \max(0, \mathbf{n} \cdot \mathbf{h})^p$$

specular coefficient

University of Victoria | Computer Science

Diffuse and Specular

Copyright: Marsette Vona

University of Victoria | Computer Science

Final Shading Equation

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

Ambient — Diffuse — Specular

If you have multiple lights, simply sum them up all together. Note that the ambience light should be considered only once.

University of Victoria | Computer Science

Shadows

- The blue point does not receive light, while the orange one does

- To check it, cast a ray from each point to the light — if you intersect something (before reaching the light) then it is in a shadow area, and the light should not contribute to its color

- These rays are usually called **shadow rays**
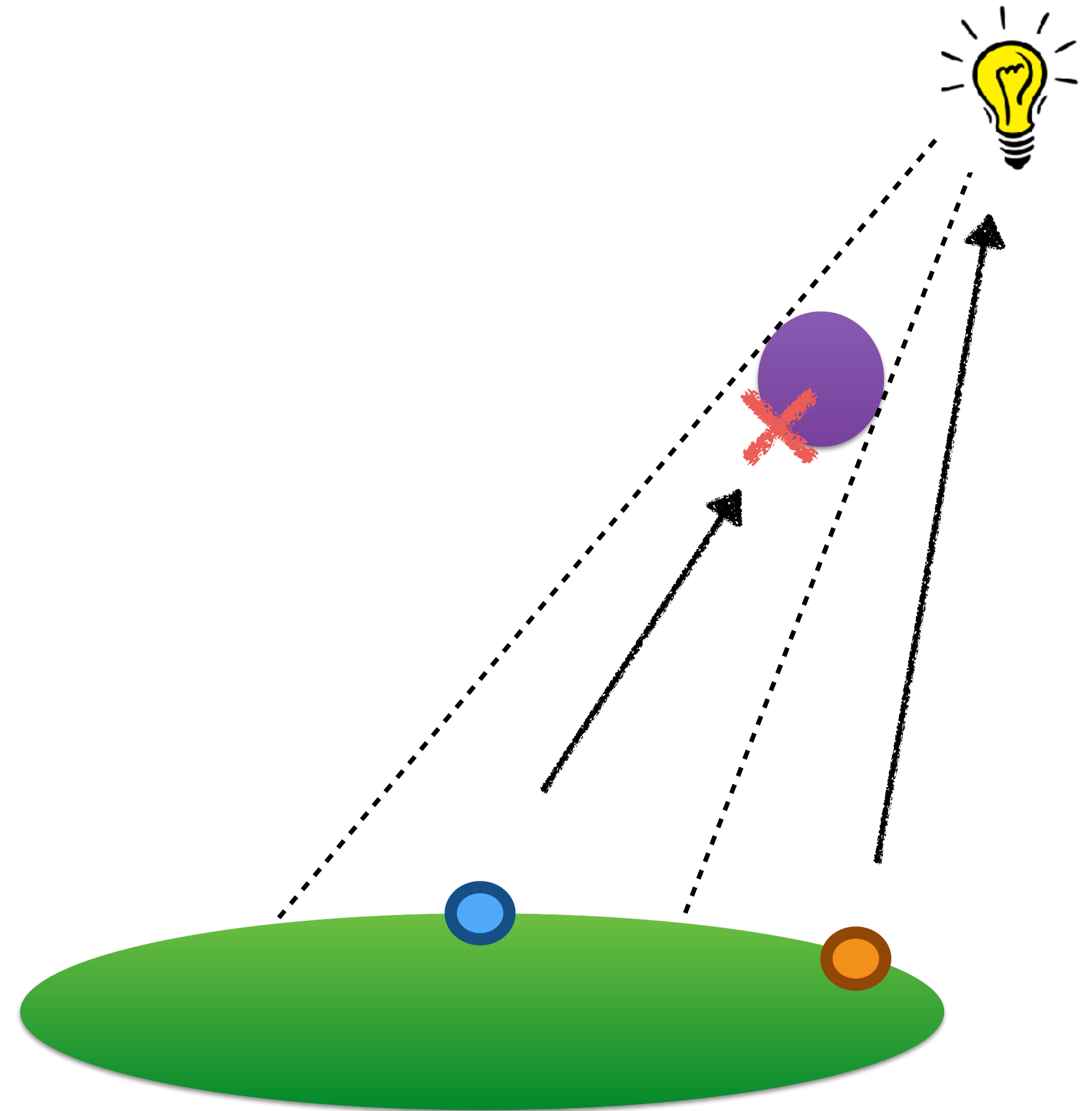
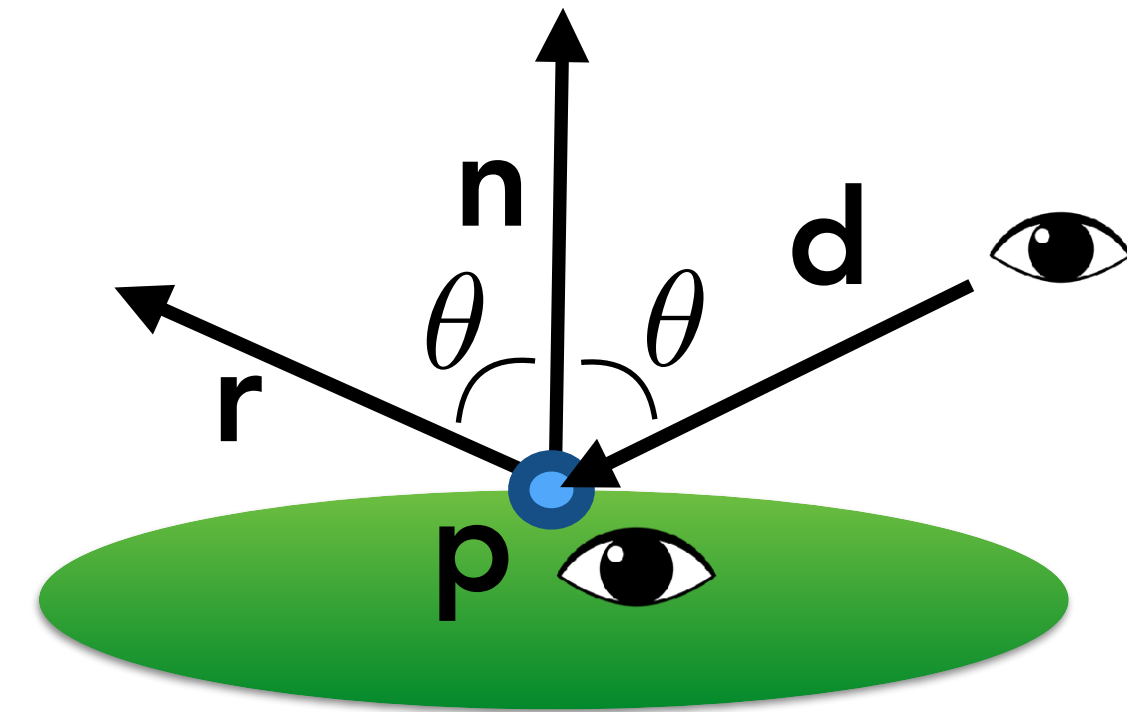The shadow rays should be casted an epsilon away from the source

University of Victoria | Computer Science

Ideal Reflections

- It is easy to add ideal reflections (also called mirror reflections) to your ray tracing program

$$\mathbf{r} = \mathbf{d} - 2(\mathbf{d} \cdot \mathbf{n})\mathbf{n}$$

$$\text{color } c = c + k_m \text{raycolor}(\mathbf{p} + s\mathbf{r}), \epsilon, \infty)$$

Reflection color

By Prabhu B - http://www.flickr.com/photos/kshathriya/851429608/sizes/l/, CC BY 2.0, https://commons.wikimedia.org/w/index.php?curid=5779465

University of Victoria | Computer Science

A simple ray-tracing program

- The source code of Assignment 2 is a simple ray tracer

University of Victoria | Computer Science

Conclusions

- Ray tracing is an effective way to render images

  - Specular reflection and Shadows are straightforward to implement

  - It is ubiquitously used even in rasterization pipelines to "pick" objects

University of Victoria | Computer Science

References

Fundamentals of Computer Graphics, Fourth Edition
4th Edition **by Steve Marschner, Peter Shirley**

Chapters 4, 10, 13

University of Victoria | Computer Science