

Computer Graphics Cheatsheet

Edoardo Riggio

January 27, 2022

Computer Graphics - SA. 2021
Computer Science
Università della Svizzera Italiana, Lugano

Contents

1	Ray Tracing	2
1.1	Whitted Ray Tracing	2
1.2	Camera / Image Definition	2
1.3	Ray Computation	3
1.4	Ray-Sphere Intersection	4
2	Lighting Models	4
2.1	Illumination Factors	5
2.2	Components of the Phong Lighting Model	5
2.2.1	Diffuse Reflection	5
2.2.2	Ambient Illumination	6
2.2.3	Specular Reflection	6
2.3	Phong Lighting Model	7
2.4	Blinn-Phong Specular Reflection	7
2.5	Light Sources	8
2.5.1	Point Light Sources	8
2.5.2	Spot Light Sources	8
2.6	Distance Attenuation	8
3	Light and Colour	9
3.1	Gamma Correction	9
3.2	Tone Mapping	9
4	Triangle Meshes	9
4.1	Ray-Triangle Intersection	10
5	Transformations	11
5.1	Homogeneous Coordinates	11
5.2	Standard Transformations	12
5.3	Normal Transformation	13
6	Advanced Raytracing	13
6.1	Shadows	13
6.2	Reflection	14
6.3	Snell's Law	14
6.4	Refraction	15
6.5	Fresnel Effect	15
6.6	Space Partitioning	16
6.6.1	Kd-Trees	16

1 Ray Tracing

Ray tracing is a method of graphics rendering that simulates the physical behaviour of light.

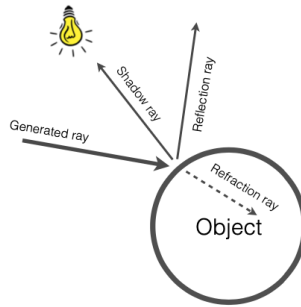
1.1 Whitted Ray Tracing

One ray is traced for every single pixel. These are known as the **primary rays**.

For each of the rays, we need to find their intersection with the scene. After we do so, secondary rays can be generated. Such rays are:

- Shadow Ray
- Reflection Ray
- Refraction Ray

The color of each of the pixels is determined based on the aggregated color of the rays.



The secondary rays are recursively generated. Some termination conditions for this generation are:

- Ray leaves the scene without ever hitting an object
- Maximal recursion depth is reached
- Contribution of the ray to the final color is negligible

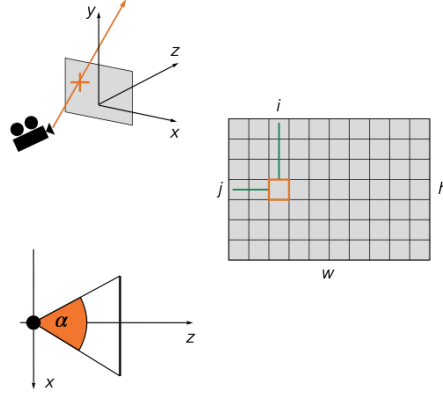
1.2 Camera / Image Definition

We are considering a global coordinate system for the raytracer. Some of the characteristics of the **camera** are:

- It is located at the origin $(0,0,0)$
- It has a horizontal opening α , the FOV (Field Of View)

Some of the characteristics of the **image** are:

- Image plane is located at $z = 1$
- Resolution of the image is of $w \cdot h$ pixels
- Pixels have indices (i, j) and coordinates $p_{ij} = (x_{ij}, y_{ij}, z_{ij})$

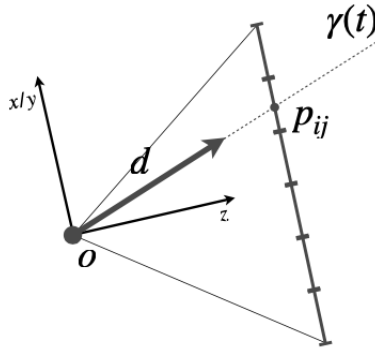


1.3 Ray Computation

The equation of a **ray** is the following:

$$\gamma(t) = o + dt$$

Where t is the distance between the origin and the intersection with an object, o is the point representing the camera position, and d is the vector representing the viewing direction – of magnitude 1.



In order to compute the vector d , we need to do the following:

```

1 s = (2 * tan(alpha / 2)) / 2;
2 X = (-w * s) / 2;
3 Y = (h * s) / 2;
4
5 for (int i = 0; i++; i < w) {
6     for (int j = 0; j++; j < h) {
7         dx = X + (i * s) + (0.5 * s);
8         dy = Y - (j * s) - (0.5 * s);
9         dz = 1;
10
11         d = glm::vec3(dx, dy, dz);
12         d = glm::vec3.normalize(d)
13     }
14 }

```

1.4 Ray-Sphere Intersection

In order to define a **sphere**, we need to set the centre and the radius. Once that is done, we need to check if the sphere intersects with the generated ray. To do so we need to check if there is some t such that:

$$\| \gamma(t) - c \| = r$$

In order to find the t , we use the following procedure.

$$\begin{aligned}
 a &= \langle c, d \rangle \\
 D &= \sqrt{\| c \|^2 - \langle c, d \rangle^2} \\
 t_{1,2} &= \langle c, d \rangle \pm \sqrt{r^2 - D^2}
 \end{aligned}$$

If the ray intersects the sphere and the camera is not inside of the sphere, then the pixel corresponding to the intersection is painted.

2 Lighting Models

In order to compute the color values on an object's surface we can use rules based on the laws of physics – radiometry and photometry. These rules model the effect of:

- **Light Sources**

Position, intensity and colour.

- **Object Surface**

Geometry and reflective properties.

2.1 Illumination Factors

The intensity of a color depends on several different factors:

- The colour and reflective properties of the object
- The position and intensity of the light sources
- The position of the viewer
- The normal of the surface at point p
- The distance of point p to the light source

2.2 Components of the Phong Lighting Model

These components are the **diffuse reflection**, the **ambient illumination**, and the **specular highlight**.

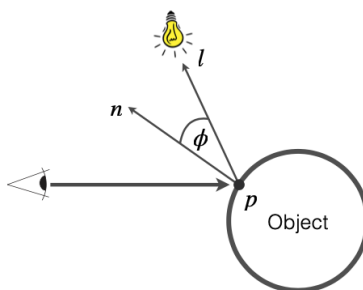
2.2.1 Diffuse Reflection

This type of reflection simulates Lambertian surfaces – i.e. matte surfaces. Here the reflected light is dispersed evenly in all directions.

There is also a material-dependent reflection constant $\rho_d \leq 1$. This is called the **diffuse coefficient**.



The intensity I_d of a surface coming from the diffuse reflection is proportional to the cosine of the angle between the surface normal n and the direction of the light source l . This is known as the **Lambertian Cosine Rule**.



The proper computation of the diffuse reflection is the following – in which the Cosine Rule is applied:

$$\begin{aligned} I_d &= \rho_d \cdot \cos \phi \cdot I \\ &= \rho_d \cdot \langle n, l \rangle \cdot I \end{aligned}$$

Where ρ_d is the diffuse coefficient, n is the normal of the surface at point p , l is the vector pointing towards the light source, and I is the intensity of the light.

An important thing to always check is that $\langle n, l \rangle > 0$.

2.2.2 Ambient Illumination

Ambient illumination is used in order to simulate indirect lighting. This refers to the effect of multiple inter-reflections of light between all objects, and to the isotropy and independence of light sources and viewpoints.

There is a global constant I , and a material-dependent reflection constant $\rho_a \leq 1$. This is called the **ambient coefficient**.

The ambient illumination is given by the following formula:

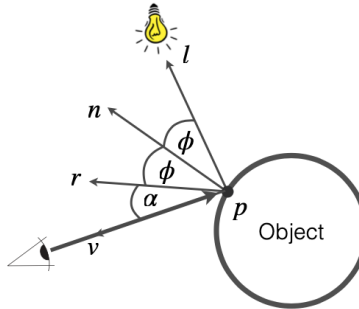
$$I_a = \rho_a \cdot I$$

Where ρ_a is the ambient coefficient, and I is the intensity of the ambient lighting.

2.2.3 Specular Reflection

Specular reflection is used in order to simulate shiny surfaces. The incoming light is reflected in exactly one reflect direction.

The specular reflection makes use of something known as the reflection vector.



The proper formulation of the specular reflection is the following:

$$I_s = \rho_s \cdot \langle r, v \rangle^k \cdot I$$

Where ρ_s is the specular coefficient of the object, r is the reflection ray given by:

$$r = 2n \cdot \langle n, l \rangle - l$$

Furthermore, v is the viewing direction of the camera, $k \geq 1$ is the shininess coefficient, and I is the light intensity.

2.3 Phong Lighting Model

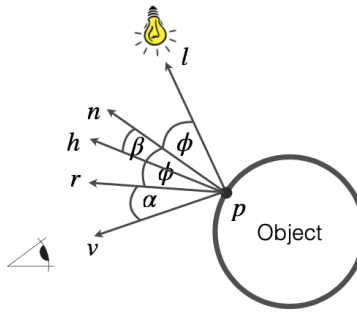
The Phong lighting model is given by the superposition of ambient, diffuse, and specular terms for each light source. The final formula is the following:

$$I = I_e + \rho_a \cdot I_a + \sum_{j=1}^n (\rho_d \cdot \langle n, l_j \rangle + \rho_s \cdot \langle r_j, v \rangle^k) \cdot I_j$$

Where I_e is the self-emitting intensity, ρ_a is the diffuse coefficient, I_a is the ambient intensity, ρ_d is the diffuse coefficient, n is the normal vector, l is the vector pointing to the light source, ρ_s is the specular coefficient, r is the reflection ray, v is the direction vector, k is the shininess, and I_j is the intensity of the j -th light source.

2.4 Blinn-Phong Specular Reflection

Here we use h instead of r in order to compute the specular reflection of an object.



The proper formulation of the Blinn-Phong specular reflection is the following:

$$I_s = \rho_s \cdot \langle n, h \rangle^{4k} \cdot I$$

Where ρ_s is the specular coefficient, n is the normal, k is the shininess coefficient, I is the light intensity, and h is defined as:

$$\frac{1}{2}(l + v)$$

Where l is the light vector, and v is the viewing direction.

2.5 Light Sources

2.5.1 Point Light Sources

Point light sources are a type of source that is isotropic – radiates evenly in all directions, it is specified by a position, and has intensity I .

Directional light sources are a special case of light sources which are only specified by direction. This is because it is an infinite set of rays that all have the same direction.

2.5.2 Spot Light Sources

This type of light source generates a light cone in a specified direction. In order to define the spot light source, we need to specify its position p , its direction d , and its opening angle Θ_L .

The intensity of the light is either maximal in the direction of d , or decreases following the formula:

$$I'(\Theta) = \cos^k \Theta \cdot I$$

And it's 0 when $\Theta > \Theta_L$.

2.6 Distance Attenuation

The more the object is distant from the light source, the less it will be illuminated by the source. Its formal definition is:

$$att(r) = \frac{1}{a_1 + a_2 r + a_3 r^2}$$

Where r is the ray of light, and a_1 , a_2 , and a_3 are constant values.

3 Light and Colour

Light is composed of electrically charged particles undergoing acceleration, and emitting electromagnetic radiation.

The eye has two different receptors in order to process such light:

- **Rods**

They perceive the intensity of light.

- **Cones**

They perceive the colors. Three types of cones exist, one for the short wavelengths, one for the medium wavelengths, and one for the long wavelengths.

3.1 Gamma Correction

The relation between the display input and the intensity shown on the screen is non-linear. In order to account for this fact, we need to apply inverse gamma correction:

$$I_{in} = I^{\frac{1}{\gamma}}$$

Where I_{in} is the input for our display, I is the intensity computed by our ray-tracer, and γ is a constant between 1.8 and 2.4.

By doing so, we are basically assigning more bits to darker regions to which we are more sensitive.

3.2 Tone Mapping

It is impossible to reproduce the real luminance range onto a screen. For this reason we need to implement tone mapping.

$$I_{in} = \max((\alpha \cdot I^\beta)^{\frac{1}{\gamma}}, 1.0)$$

Where I_{in} is the input for our display, I is the intensity computed by our raytracer, α and β are the tone mapping coefficients, and γ is the gamma coefficient.

4 Triangle Meshes

Complex surfaces can be approximated with triangle meshes. A triangle mesh is composed by the following elements:

- **Vertices**

Each vertex has three coordinates – x , y , and z .

- **Edges**

- **Triangles/Faces**

These are triplets of vertices which are displayed in counter-clockwise order.

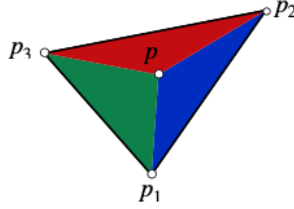
4.1 Ray-Triangle Intersection

In order to compute the intersection between the ray and a triangle, we first need to compute the intersection between the ray and the triangle's plane. After, we do the following:

1. Find a t such that $p = \gamma(t)$ is coplanar with p_1 , p_2 and p_3
2. Check if $t > 0$ and that p is inside of the triangle
3. If the above conditions are all true, compute the lighting at point p . The normal vector at point p is computed as:

$$n = \langle (p_2 - p_1), (p_3 - p_1) \rangle$$

In order to test whether the point is inside of the triangle, we use the **Barycentric Coordinates**.



The algorithm is the following:

1. Consider a planar triangle made up of points (p_1, p_2, p_3) and a point p
2. Compute the area W of the triangle

$$p_i = (x_i, y_i, z_i)$$

$$n = \langle (p_2 - p_1), (p_3 - p_1) \rangle$$

$$2W = \|n\|$$

3. Compute the areas w_1 , w_2 and w_3 of the triangles

$$p = (x, y, z)$$

$$n_i = \langle (p_{i+1} - p), (p_{i-1} - p) \rangle$$

$$2w_i = \|n_i\| \cdot \text{sign}(\langle n_i, n \rangle)$$

4. Divide all of these areas by W . These values are $\lambda_1(p)$, $\lambda_2(p)$ and $\lambda_3(p)$ – i.e. barycentric coordinates

$$\lambda_1(p) = \frac{w_1}{W} \quad \lambda_2(p) = \frac{w_2}{W} \quad \lambda_3(p) = \frac{w_3}{W}$$

There are two properties for barycentric coordinates:

- **Partition of Unity**

The sum of all barycentric coordinates is equal to 1.

- **Non-Negativity**

All of the barycentric coordinates are greater or equal to 0.

5 Transformations

There are two different ways of moving an object in the scene:

- Moving the camera coordinates – i.e. global coordinates
- Moving the object coordinates – i.e. local coordinates

Each object is initially positioned at the origin of the scene. Here the local coordinates are the same as the global coordinates. Afterwards, the object is moved around using local coordinate system transformations.

5.1 Homogeneous Coordinates

In order to combine different transformations together, we need to work in four dimensions. Any point will be now written as:

$$p = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

And each displacement – i.e. vector – will be now written as:

$$\vec{v} = \begin{bmatrix} x \\ y \\ z \\ 0 \end{bmatrix}$$

Such that:

- $p + d = 1 + 0 = 1 = \text{position}$
- $p - p = 1 - 1 = 0 = \text{displacement}$
- $d + d = 0 + 0 = 0 = \text{displacement}$
- $p + p = 1 + 1 = 0 = \text{displacement}$

5.2 Standard Transformations

Some of the standard transformations we discussed are:

- **Translation** by $t = (t_x, t_y, t_z)$

$$T = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- **Shearing** yz -plane, xz -plane and xy -plane

$$S_{yz} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ d_y & 1 & 0 & 0 \\ d_z & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad S_{xz} = \begin{bmatrix} 1 & d_x & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & d_z & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad S_{xy} = \begin{bmatrix} 1 & 0 & d_x & 0 \\ 0 & 1 & d_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- **Scaling** by factors s_x, s_y, s_z

$$S = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- **Rotating** about axes x, y, z

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha & 0 \\ 0 & \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_y = \begin{bmatrix} \cos \alpha & 0 & \sin \alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \alpha & 0 & \cos \alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_z = \begin{bmatrix} \cos \alpha & -\sin \alpha & 0 & 0 \\ \sin \alpha & \cos \alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

These transformations are also known as **affine transformations**. The kind of transformations preserve line, planes parallelisms and ratios. Distances and angles are only preserved if

$$\det(A) = \pm 1$$

Transformations can also be combined. For example:

$$p \mapsto T \cdot R \cdot p$$

Where R is a rotation and is performed as the first transformation, and T is a translation and is performed as the second transformation.

5.3 Normal Transformation

Normals need to be transformed too. For any transformation m that preserves angles – i.e. $\det(A) = \pm 1$, we compute the the new normals as follows:

$$\vec{n}_{new} = M \vec{n}$$

Otherwise, for a more general case, we can use the following:

$$\vec{n}_{new} = (M^{-1})^T \vec{n}$$

6 Advanced Raytracing

6.1 Shadows

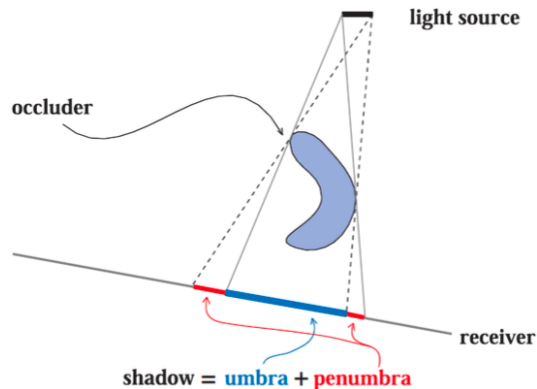
Shadows are darks spots for which the light is occluded. A shadow is composed of two main parts:

- **Umbra**

This is the complete shadow.

- **Penumbra**

This is the partial shadow.



Furthermore, shadows convey 3D information of an object, such as:

- Relative position
- Depth information
- Position of the light sources

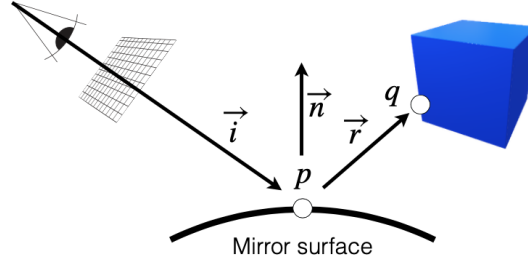
Shadows can be computed by extending the Phong model as follows:

$$I = I_e + \rho_a \cdot I_a + \sum_{j=1}^n (\rho_d \cdot \langle n, l_j \rangle + \rho_s \cdot \langle r_j, v \rangle^k) \cdot I_j \cdot \text{attr}(t) \cdot s_j(p)$$

Where $s_j(p)$ is either 0 or 1. It is 1 when the secondary ray reaches the light source, 0 if the secondary ray hits another object before reaching the light source.

6.2 Reflection

Reflection happens when a ray hits a reflective object.



In order to compute the reflected vector r , we use the following formula:

$$r = i - 2n \cdot \langle n, i \rangle$$

In order to colour the mirror surface, we first need to find the intersection at q . The color at point q is then computed and used for p as well.

In order to determine how reflective a surface is, we need to use the constant

$$\alpha_{\text{reflect}} \in [0, 1]$$

6.3 Snell's Law

This law says that the ratio of the sines of the two angles is equal to the ratio of the indices of refraction. This is also equal to the ratio of the velocities in the mediums.

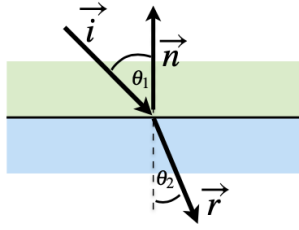
$$\frac{\sin \Theta_1}{\sin \Theta_2} = \frac{\delta_1}{\delta_2} = \frac{v_1}{v_2}$$

We also have that:

$$\frac{\delta_1}{\delta_2} \cdot \sin \Theta_1 \begin{cases} < 1 & \text{refraction} \\ = 1 & \text{critical angle} \\ > 1 & \text{total internal reflection} \end{cases}$$

6.4 Refraction

Refraction happens whenever a ray passes the boundary between two transparent materials.



The refracted ray r can be found by using the following algorithm:

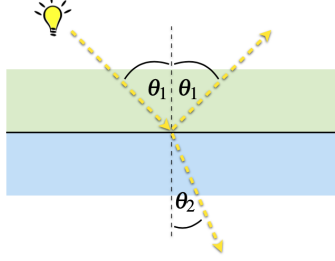
$$\begin{aligned} a &= n \cdot \langle n, d \rangle \\ b &= i - a \\ \beta &= \frac{\delta_1}{\delta_2} \\ \alpha &= \sqrt{1 + (1 - \beta^2) \frac{\|b\|^2}{\|A\|^2}} \end{aligned}$$

Where δ_1 and δ_2 are the two indices of refraction of the two materials. And finally

$$r = \alpha a + \beta b$$

6.5 Fresnel Effect

The amount of light that is reflected or refracted on a surface depends on the viewing angle.



The reflected and refracted rays of the Fresnel effect is computed as:

$$F_{\text{Ri}} = \frac{1}{2} \left(\left(\frac{\delta_2 \cos \Theta_1 - \delta_1 \cos \Theta_2}{\delta_2 \cos \Theta_1 + \delta_1 \cos \Theta_2} \right)^2 + \left(\frac{\delta_1 \cos \Theta_1 - \delta_2 \cos \Theta_2}{\delta_1 \cos \Theta_1 + \delta_2 \cos \Theta_2} \right)^2 \right)$$

$$F_{\text{Rf}} = 1 - F_{\text{Ri}}$$

6.6 Space Partitioning

Octrees are adptive grid subdivisions. They are difficult to traverse.

In **Bounding Volume Hierarchy** (BVM), neighbouring objects are gathered using simple bounding primitives. the gathering difficulty is optimal. When we traverse we start from the biggest primitives.

Binary Space Partitioning trees (BSPt) recursively divide the scene with planes. It has a running time of $O(\log n)$, but it is complicated to traverse.

6.6.1 Kd-Trees

Here we recursively divide the scene into planes. Differently from BSpT, the planes are axis-aligned.

In order to traverse a kd-tree, we use the parametric ray equation

$$\gamma(t) = o + td$$

And we recursively consider the active ray segment

$$[t_{\min}, t_{\max}]$$

This algorithm runs in $O(\log n)$.