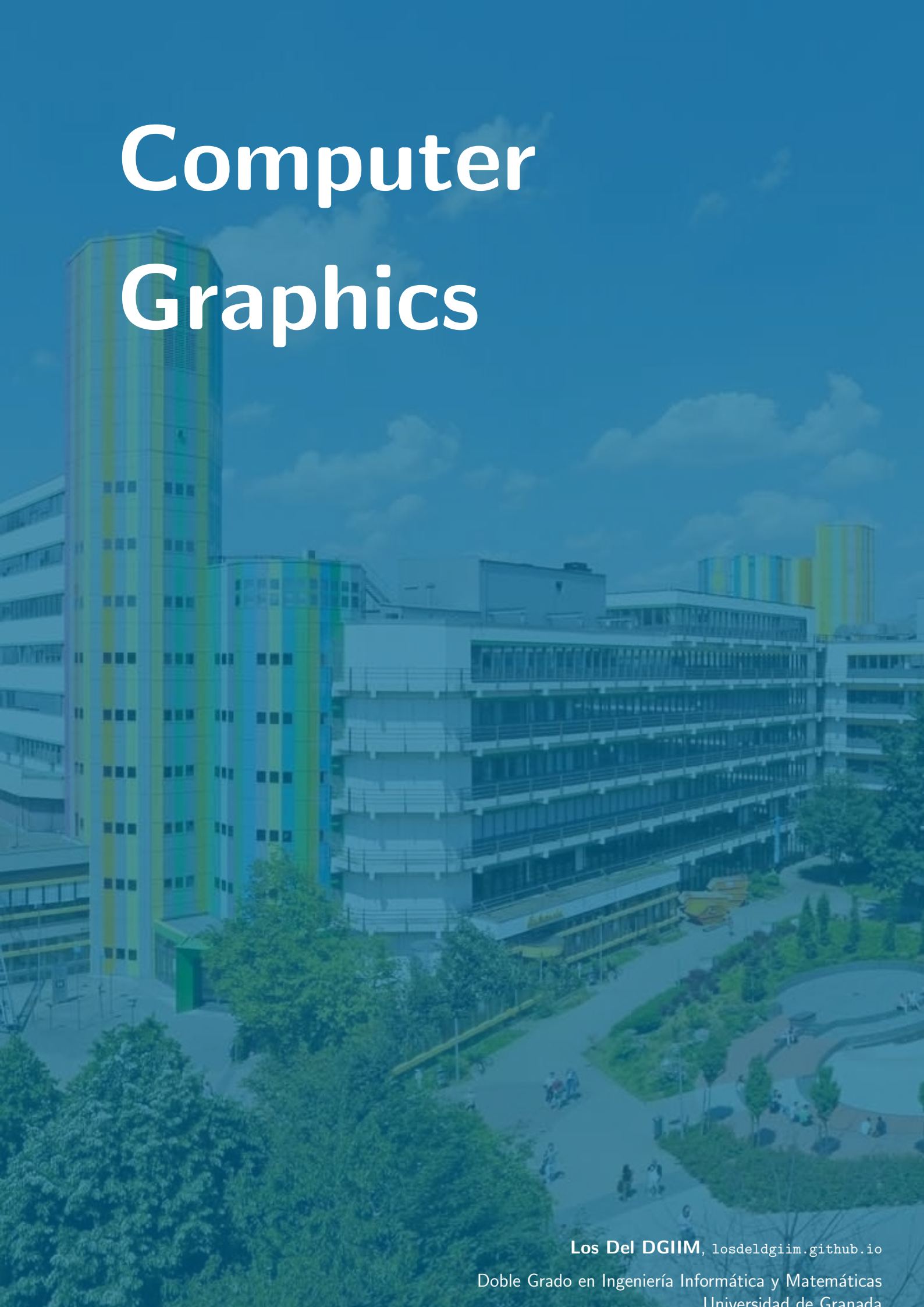
Computer Graphics



Los Del DGIIM, losdeldgiim.github.io

Doble Grado en Ingeniería Informática y Matemáticas

Universidad de Granada

Computer Graphics

Los Del DGIIM, losdeldgiim.github.io

Arturo Olivares Martos

Granada, 2026

Índice general

1. Ray and Path Tracing

During this chapter, we will cover the basics of ray tracing and path tracing, two important techniques in computer graphics for rendering images. In order to introduce them, some previous concepts are required.

Definición 1.1 (Light Ray)**.** A *light ray* is a straight line along which light energy (photons) travels. It is defined by its origin and direction (which can also be given as a second point).

Definición 1.2 (Light Path)**.** A *light path* is a sequence of light rays that represent the journey of light from its source to the observer, including any interactions with objects in the scene (such as reflection, refraction, or absorption).

When a light ray interacts with an object, it can be absorved or its direction can be changed due to reflection or refraction. In order to render images, there are two main approaches:

- Going forward: rays are traced forward from the light source, so the entire scene would be perfectly illuminated. However, most of the rays will not reach the observer, so this method is inefficient.
- Going backward: rays are traced backward from the observer, so only the rays that reach the observer are considered. A ray is traced for determining the color of each pixel in the image.

Given the inefficiency of the forward approach, the backward approach is the one used in ray tracing and path tracing.

1.1. Ray Tracing

Ray tracing[1] is a rendering technique that simulates the way light interacts with objects in a scene to produce realistic images. For each pixel in the image, a *primary ray* is traced from the observer's eye through the pixel into the scene. Once the ray intersects with an object, three type of *secondary rays* can be generated:

- Shadow rays: For each light source, a shadow ray is traced from the intersection point to the light source to determine if the point is in shadow or not. If the shadow ray intersects another object before reaching the light source, the point is in shadow and does not receive direct illumination from that light source.

[1] It is also kown as Whitted Ray Tracing, as Turner Whitted was one of its original developpers.

- Reflection rays: If the surface is reflective, a reflection ray is traced in the direction of the reflected ray, which is calculated based on the angle of incidence and the surface normal.
- Refraction rays: If the surface is translucid, a refraction ray is traced in the direction of the refracted ray, which is calculated based on Snell's law.

Therefore, for each intersection $N + 2$ rays are traced, where N is the number of light sources in the scene. The reflection and refraction rays are recursively considered as primary rays, so they can generate more secondary rays if they intersect with other objects. This process continues until once of the following conditions is met:

- A ray does not intersect with any object, in which case it contributes the background color to the pixel.
- A ray reaches a predefined maximum recursion depth, in which case it does not generate any more rays and contributes just its local illumination to the pixel.

The color of each pixel is determined by the color of its first intersection point. For each intersection point, the color is calculated as:

$$I = k_d I_{\text{direct}} + k_s I_{\text{reflection}} + k_t I_{\text{refraction}}$$

where $k_d, k_s, k_t \in \mathbb{R}_0^+$ such that $k_d + k_s + k_t = 1$ are three coefficients that determine the contribution of each type of ray to the final color, and I_{direct} is the color contribution from local illumination calculated using phong shading, $I_{\text{reflection}}$ is the color contribution from the intersection point of the reflection ray, and $I_{\text{refraction}}$ is the color contribution from the intersection point of the refraction ray.

Observación. As the reader may have noticed, the complexity of ray tracing increases exponentially, but the contribution of rays decreases exponentially, so in practice, a maximum recursion depth of 4 or 5 is usually sufficient for producing high-quality images.

1.1.1. Diffuse surfaces

There are two main types of surfaces:

- Diffuse surfaces: they reflect light uniformly in all directions.
- Specular surfaces: they reflect light in a specific direction, following the law of reflection.

As the reader may notice, ray tracing just considers specular surfaces. In order to render diffuse surfaces, another reason to stop the recursion is included:

- A ray hits a diffuse surface, in which case it does not generate any more rays and contributes just its local illumination to the pixel.

This way, between the eye and the light source as many specular surfaces as we want can be included, but only one diffuse surface can be included just before the light source.

1.2. Path Tracing

Path tracing is also a rendering technique that simulates the way light interacts with objects in a scene to produce realistic images, similar to ray tracing. For each pixel in the image, N rays are traced from the observer's eye through the pixel into the scene, where N is a predefined number of samples per pixel. Once a ray intersects with an object, its direction is randomly changed according to the material properties of the surface, and a new ray is traced in that direction, generating therefore a path. This process continues until one of the following conditions is met:

- A ray does not intersect with any object, in which case it contributes the background color to the pixel.
- A ray reaches a predefined maximum depth, in which case it contributes just its local illumination to the pixel.

The color of each pixel is determined by the average color contribution of all the rays traced for that pixel. It should however be noted that "randomly" does not mean "uniformly". Depending on the material properties of the surface, from an specific direction, some directions will be more likely to be chosen than others.

- For perfect diffuse surfaces, the new direction is chosen uniformly in the hemisphere above the surface.
- For perfect specular surfaces, the new direction is the reflection direction.

2. Acceleration Structures

One of the main problems of ray tracing is that for each ray, we need to calculate its first intersection point with the objects in the scene. This is $O(mp)$, where m is the number of objects in the scene and p is the number of pixels in the image (as many rays as pixels are traced). If there are n polygons in total in the scene, the complexity of ray tracing is $O(np)$. In order to reduce this complexity, acceleration structures are used. In the following sections, we will cover the three most common acceleration methods.

2.1. Fewer rays

Two main techniques can be used for reducing the number of rays traced:

- Adaptive tree-depth control: The maximum recursion depth is not fixed, but it is determined adaptively based on the contribution of each ray to the final image. If a ray contributes very little to the final image, it is not traced further.
- Stochastic sampling: As seen in path tracing, only one secondary ray is traced for each intersection point, whose direction is randomly chosen according to the material properties of the surface.

However, there are some cases where these techniques are not effective. In these cases, other acceleration methods are required.

2.2. Faster ray-object intersections

In order to speed up the ray-object intersection tests, the concept of "barycentric interpolation" in a triangle can be used.

2.2.1. Barycentric interpolation

Barycentric interpolation is a method of interpolating values at the vertices of a triangle to find the value at any point inside the triangle. It uses the concept of barycentric coordinates, which are a set of three numbers that represent the relative position of a point with respect to the vertices of the triangle. The barycentric coordinates of a point P with respect to a triangle defined by vertices P_1, P_2, and P_3 are given by:

$$\lambda_1 = \frac{\Delta(P, P_2, P_3)}{\Delta(P_1, P_2, P_3)}, \quad \lambda_2 = \frac{\Delta(P, P_3, P_1)}{\Delta(P_1, P_2, P_3)}, \quad \lambda_3 = \frac{\Delta(P, P_1, P_2)}{\Delta(P_1, P_2, P_3)}$$

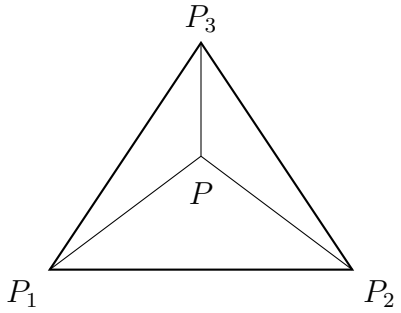


Figura 2.1: Barycentric coordinates of a point P with respect to a triangle defined by vertices P_1, P_2, and P_3.

where $\Delta(A, B, C)$ is the area of the triangle defined by points A, B, and C, as shown in Figure 2.1. They satisfy the following properties:

- $\lambda_1 + \lambda_2 + \lambda_3 = 1$
- $\lambda_i \geqslant 0$ for $i = 1, 2, 3$ if P is inside the triangle.

Once the barycentric coordinates of a point P are calculated, we can use them to interpolate any value defined at the vertices of the triangle. Given the values f_1, f_2, and f_3 at the vertices P_1, P_2, and P_3 respectively, the interpolated value at point P with barycentric coordinates λ_1, λ_2, and λ_3 is given by:

$$f(P) = \lambda_1 f_1 + \lambda_2 f_2 + \lambda_3 f_3$$

Two things should be noted about barycentric interpolation:

- It also interpolates the x and y coordinates of the point P in the triangle. These two conditions and the fact that $\lambda_1 + \lambda_2 + \lambda_3 = 1$ are sufficient to determine the barycentric coordinates of any point in the triangle.
- It can be proven that this interpolation is linear; and given that there is a unique linear interpolation for any set of values at the vertices of the triangle, it can be concluded that barycentric interpolation is just another way of performing linear interpolation.

2.2.2. Ray-triangle intersection

Any point P in a triangle defined by vertices P_1, P_2, and P_3 can be expressed as a linear combination of the vertices:

$$P = \lambda_1 P_1 + \lambda_2 P_2 + \lambda_3 P_3$$

where λ_1, λ_2, and λ_3 are the barycentric coordinates of P with respect to the triangle. In addition, any point P' on a ray defined by an origin P_S and a direction $\overrightarrow{D}$ can be expressed as:

$$P' = P_S + t\overrightarrow{D} \qquad t \in \mathbb{R}_0^+$$

Therefore, determining the intersection point of a ray with a triangle is equivalent to finding the values of λ_1, λ_2, λ_3, and t that satisfy the following system of equations:

$$\begin{cases} \lambda_1 P_1 + \lambda_2 P_2 + \lambda_3 P_3 = P_S + t\overrightarrow{D} \\ \lambda_1 + \lambda_2 + \lambda_3 = 1 \end{cases}$$

It should be noted that the first equation represents three equations, one for each coordinate. Therefore, we have a system of four equations with four unknowns. That system can be fastly solved using Cramer's rule and the triple product of vectors to calculate the determinants. After solving this system, we have to check if the solution is valid:

- $t \in \mathbb{R}_0^+$, as we are only interested in the intersection points that are in the direction of the ray (not behind the ray origin).
- $\lambda_i \geqslant 0$ for $i = 1, 2, 3$, as we are only interested in the intersection points that are inside the triangle (not in the whole plane defined by the triangle).

2.3. Fewer ray-object intersections