Ray tracing technologies

# Introduction

In the last three decades the quality of interactive computer graphics has increased drastically and there is still a demand for higher quality. However, the standard method of computing images, called rasterization, does not allow for advanced effects such as reflections and shadows. For this reason, the development of an alternative method which would allow this has started. The main goal is to investigate whether it can replace rasterization.

Ray tracing is a brand-new image rendering technique. The image is created by tracing the path of light by each ray individually. That allows to produce a much higher degree of visual realism, though it has much greater computational costs.

Ray tracing is capable of simulating a wide variety of optical effects, such as reflection and refraction, scattering, and dispersion phenomena (such as chromatic aberration). The implementation of these effects on modern graphics hardware is complex and non-intuitive and in many cases the implementation of such effects on modern graphics hardware can only be an approximation due to the limits of the rendering method.

# How light works in real life

In nature, each light source emits a ray or multiple rays of light which travels, eventually, to a surface that interrupts its progress. The "ray" can be imagined as a stream of photons traveling along the same path. In a perfect vacuum this ray will be a straight line (ignoring relativistic effects).

Every possible combination of four things might happen with this light ray: absorption, reflection, refraction and fluorescence.

If a light ray is absorbed by a surface, it results in a loss of intensity of the reflected or refracted light rays. The surface might also reflect all or part of the light ray, in one or more directions. If the surface has any transparent or translucent properties, it refracts a portion of the light beam into itself in a different direction while absorbing some (or all) of the spectrum (and possibly altering the color). Less commonly, a surface may absorb some portion of the light and fluorescently re-emit the light at a longer wavelength color in a random direction, though this is rare enough that it can be discounted from most rendering applications.

In each case described earlier, all of the incoming light must be accounted Between absorption, reflection, refraction and fluorescence.

All the reflected and/or refracted rays may strike other surfaces, where their absorptive, refractive, reflective and fluorescent properties again affect the progress of the incoming rays. Some of these rays travel in such a way that they hit our eye, causing us to see the scene and so contribute to the final rendered image.

# What ray-tracing does

The current standard rendering method, know as rasterization, is a local illumination rendering method. This means that only the light that comes directly from a light source is taken into account. Light that does not come directly from a light source, such as light reflected by a mirror, does not contribute to the image.

In contrast, ray tracing is a global illumination rendering method. This means that light that is reflected from other surfaces, for example a mirror, is also taken into account. This is essential for advanced effects such as reflection and shadows.

For example, if we want to model a water surface reflecting the scene correctly we need a global illumination rendering method. With a local illumination rendering method the light from the water surface can only be determined by the light falling directly on it, not the light from the rest of the scene and thus we will see no reflections.

# Algorythm description

Optical ray tracing works by tracing a path from an imaginary eye through each pixel in a virtual screen, and calculating the color of the object visible through it.

Typically, each ray must be tested for intersection with some subset of all the objects in the scene. Once the nearest object has been identified, the algorithm will estimate the incoming light at the point of intersection, examine the material properties of the object, and combine this information to calculate the final color of the pixel. Certain illumination algorithms and reflective or translucent materials may require more rays to be re-cast into the scene.

Some people doubted the idea of sending rays away from the camera, rather than into it (as actual light does in reality), but doing so is many orders of magnitude more efficient. Since the overwhelming majority of light rays from a given light source do not make it directly into the viewer's eye, a "forward" simulation could potentially waste a tremendous amount of computation on light paths that are never recorded. Therefore, after either a maximum number of reflections or a ray traveling a certain distance without intersection, the ray ceases to travel and the pixel's value is updated.

# Hardware requirements

Using such computationally difficult algorithms requires certain hardware. In fact, ray-tracing has completely changed the market of GPUs and other computer parts.

Ray tracing requires from the computer such qualities as:

- Parallel calculations

Ray tracing is known is as being “embarrassingly parallel”. This is because rays are independent from each other. Because of this, it would be very efficient if the hardware can trace rays in parallel.

- Large amount of floating-point operations

The computation of the intersections of the rays and, when the rays hit an object, the computation of the colour of the surface requires a lot of floating point operations.

- Complex flow control

When a ray hits an object other rays may be traced recursively and it is always uncertain how many rays deep this will go and how many branches the calculation will have. This means the hardware needs to be able to handle complex flow control like recursion.

- A lot of memory access

Every ray may intersect with any point in the scene and the appropriate data should be fetched for computation. This means a lot of memory access.

# Motion support

For static scenes this spatial index structure can be computed in advance, and used for each frame. In this way the performance is increased – the environment cost does not effect the frame rate.

However, this approach does not allow scenes to be truly interactive because the user can walk or fly trough a scene but cannot interact with the environment. It is impossible to move an object or have animations in the scene without also changing the spatial index structure. Therefore, to handle motion, ray-tracing algorithms have to me modified.

# Types of motion and matching calculational strategies

Keeping the spatial index structure up to date is done differently for different types of motion. includes .

The animation is divided into four categories:

- Static motion (for objects which don’t change)

- Hierarchical motion (for a set of primitives with the same transformation)

All the transformations are usually stored in a tree called the scene tree, hence the name hierarchical motion. For example, when a whole object is moved trough a scene.

- Continuous Dynamic motion includes transforming the triangles of the surface of the object.

- Unstructured motion is when a set of primitives move in an unstructured way, without relation to each other. For example triangles animated by a particle system.