**ACROPOLIS INSTITUTE OF TECHNOLOGY AND RESEARCH INDORE**



**SUBJECT: Computer graphics and multimedia**

**TOPIC: Ray tracing**

**Submitted to: Submitted By:**

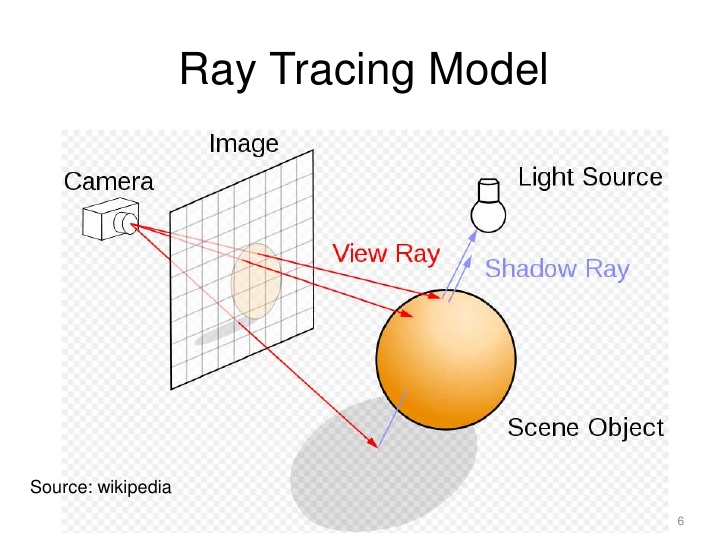
**Mr. Anil Patidar Srishti Agrawal(0827CS161225)**

**Vartika Khaddar(0827CS161239)**

**INTRODUCTION:**

Ray tracing is a technique for presenting three-dimensional (3D) images on a two-dimensional (2D) display by tracing a path of light through pixels on an image plane.

A ray tracing program mathematically identifies and reproduces the path that each light ray follows in reverse direction from the eye back to its point of origin. The path of each light ray consists of multiple straight-line components and nearly always involves reflection, refraction, or shadow effects from points within the scene. In animation, the position and orientation of the straight-line components of each ray constantly vary so each ray is represented by a mathematical equation that defines its path through space as a function of time. Rays are assigned a color based on the pigments the objects in the scene that the ray passes through and each pixel on the display corresponds to a ray.



Ray tracing is a technique for rendering three-dimensional graphics with very complex light interactions. This means you can create pictures full of mirrors, transparent surfaces, and shadows, with stunning results. We discuss ray tracing in this introductory graphics article because it is a very simple method to both understand and implement. It is based on the idea that you can model reflection and refraction by recursively following the path that light takes as it bounces through an environment.

Ray tracing is so named because it tries to simulate the path that light rays take as they bounce around within the world - they are traced through the scene. The objective is to determine the color of each light ray that strikes the view window before reaching the eye. A light ray can best be thought of as a single photon (although this is not strictly accurate because light also has wave properties - but I promised there would be no theory).

The name ``ray tracing'' is a bit misleading because the natural assumption would be that rays are traced starting at their point of origin, the light source, and towards their destination, the eye. This would be an accurate way to do it, but unfortunately it tends to be very difficult due to the sheer numbers involved. Consider tracing one ray in this manner through a scene with one light and one object, such as a table. We begin at the light bulb, but first need to decide how many rays to shoot out from the bulb. Then for each ray we have to decide in what direction it is going. There is really infinity of directions in which it can travel - how do we know which to choose? Let's say we've answered these questions and are now tracing a number of photons. Some will reach the eye directly, others will bounce around some and then reach the eye and many, and many more will probably never hit the eye at all. For all the rays that never reach the eye, the effort tracing them was wasted.

In order to save ourselves this wasted effort, we trace only those rays that are guaranteed to hit the view window and reach the eye. It seems at first that it is impossible to know beforehand which rays reach the eye. After all, any given ray can bounce around the room many times before reaching the eye. However, if we look at the problem backwards, we see that it has a very simple solution. Instead of tracing the rays starting at the light source, we trace them backwards, starting at the eye

Consider any point on the view window whose color we're trying to determine. Its color is given by the color of the light ray that passes through that point on the view window and reaches the eye. We can just as well follow the ray backwards by starting at the eye and passing through the point on its way out into the scene. The two rays will be identical, except for their direction: if the original ray came directly from the light source, then the backwards ray will go directly to the light source; if the original bounced off a table first, the backwards ray will also bounce off the table. You can see this by looking at Figure again and just reversing the directions of the arrows. So the backwards method does the same thing as the original method, except it doesn't waste any effort on rays that never reach the eye.

This, then, is how ray tracing works in computer graphics. For each pixel on the view window, we define a ray that extends from the eye to that point. We follow this ray out into the scene and as it bounces off of different objects. The final color of the ray (and therefore of the corresponding pixel) is given by the colors of the objects hit by the ray as it travels through the scene.

Just as in the light-source-to-eye method it might take a very large number of bounces before the ray ever hits the eye, in backwards method it might take many bounces before the ray every hits the light. Since we need to establish some limit on the number of bounces to follow the ray on, we make the following approximation: every time a ray hits an object, we follow a single new ray from the point of intersection directly towards the light source.

**Recursive Ray Tracing:**

The next important research breakthrough came from [Turner Whitted](https://en.wikipedia.org/wiki/Turner_Whitted) in 1979. Previous algorithms traced rays from the eye into the scene until they hit an object, but determined the ray color without recursively tracing more rays. Whitted continued the process. When a ray hits a surface, it can generate up to three new types of rays: reflection, refraction, and shadow. A reflection ray is traced in the mirror-reflection direction. The closest object it intersects is what will be seen in the reflection. Refraction rays traveling through transparent material work similarly, with the addition that a refractive ray could be entering or exiting a material. A shadow ray is traced toward each light. If any opaque object is found between the surface and the light, the surface is in shadow and the light does not illuminate it. This recursive ray tracing added more realism to ray traced images.

**HISTORY:**

Ray tracing was first developed in the 1960s by scientists at an organization known as Mathematical Applications Group. Ray tracing is used extensively in computer gaming and animation, television and DVD programming and movie production.

**ALGORITHM:**

Color raytrace( Ray r, int depth, Scene world, vector <Light\*> lights )

{ Ray \*refl, \*tran;

Color color\_r, color\_t, color\_l;

// Terminate if maximum recursion depth has been reached.

if ( depth > MAX\_DEPTH ) return backgroundColor;

// Intersect ray with scene and keep nearest intersection point

int hits = findClosestIntersection(r, world);

if ( hits == 0 ) return backgroundColor;

// Apply local illumination model, including shadows

color\_l = calculateLocalColor(r, lights, world);

// Trace reflected and refracted rays according to material properties

if (r->isect->surface->material->k\_refl > 0)

{ refl = calculateReflection(r);

color\_r = raytrace(refl, depth+1, world, lights);

}

if (r->isect->surface->material->k\_refr > 0)

{ tran = calculateRefraction(r);

color\_t = raytrace(tran, depth+1, world, lights);

}

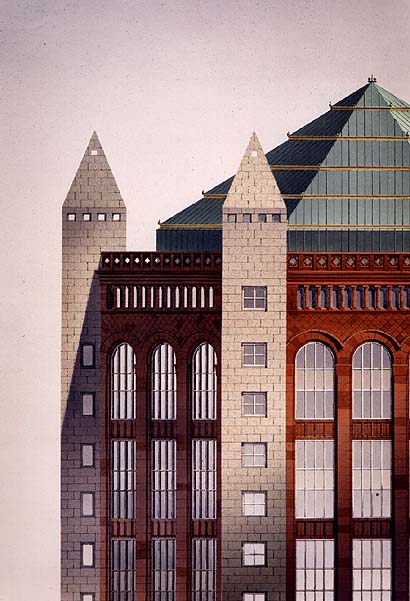
return color\_l + color\_r + color\_t;

}

**APPLICATIONS**

* **Ray Tracing in Architecture**

When presenting design proposals to clients, it is very important that architects have realistic renderings accompanying their designs.  In the past, architects relied on hand-done drawings created with ink and water colours.  Unfortunately, it is extremely difficult to produce realistic illumination effects using the traditional rendering methods.  Even many computer aided design (CAD) programs which accurately model objects are unable to model light.   
    Some modeling programs even allow architects to introduce light into a picture, specifying the light's location, orientation, color, and distribution.  These features help designers to create effects such as shadows and specular highlights, however, they still fail to model the physical reality that objects tracing with light.  
    Architects are primarily interested in creating visually realistic images. Backwards ray tracing, combined with radiosity techniques, is often the most useful method for architectural rendering. Because modeling light is so essential to the field, the development of programs that incorporate ray tracing has been a dream come true for many architects.  Global illumination technology is succeeding in bringing architectural design to life.

    
  Examples of Ray Traced Architectural Images 

## Ray Tracing in Theater and Television Lighting Design

## Because ray tracing allows visually realistic modeling of light, the technology can be usefully applied in the areas of theater and television lighting.  Without the ability to model physically correct images, stage lighting setups can take extreme amounts of effort.  Many stage and television productions require hundreds of individual lights that must be positioned, aimed, and filtered.  It is also necessary for lights to be switched, redirected, and dimmed while a production is in progress.  Ray tracing allows set and lighting designers, actors, and directors to develop and visualize complex lighting setups months before a production ever opens.

## Ray Tracing as a Tool for Engineers

Ray tracing can consider all the light in a given environment (termed global illumination*).*Global illumination is a physically correct model, which accurately simulates light's behavior in a real physical environment.  This proves to be extremely useful to lighting designers, solar energy researchers, and mechanical engineers.  They can use ray tracing to do much more than render photorealistic pictures.  Engineers use the technology to predict illumination levels, luminance gradients, and visual performance criteria.  Global illumination is a valuable engineering tool in that it allows us to quantitatively analyse the distribution and directionality of light and research radiant heat transfer.  This is helping us to progress in everything from lighting and heating rooms more efficiently, to creating solar energy concentrators for aerospace applications.

## Ray Tracing in Animation

Questions surrounding animation have interested computer scientists for several decades.  Advancements in computer graphics including developments in ray tracing have opened world of possibility in the field.  Traditionally, individual frames of animated works were drawn by hand.  Movement was simulated through a complex series frame adjustment step.  There is still a strong sense of nostalgia for traditional methods of animation, however, computer graphics are playing a stronger and stronger role in the process.   
    Computer animation is executed on model worlds before they are rendered by a ray tracer. This technique can be highly optimized.   
    Ray tracing can be used to add "fancy" effects such as reflection and shadowing that are often difficult and time consuming for traditional artists to produce. Graphical technology is also capable of rendering photorealistic images that would be nearly impossible to produce without computerized ray tracing.   
    Examples of computer graphics and ray tracing in modern animation include   
 advanced reflection, shadowing, and secularity

**ADVANTAGES AND DISADVANTAGES**

## Advantages

1. Conceptually simple. Same basic ray-object intersection code handlers
   1. shadows
   2. reflections
   3. transparency
2. Automatic hidden surface removal (always finds closest intersection)
3. Ray-object intersection is fundamental in computer graphics:
   1. line of sight (visibility testing)
   2. tracing light - indirect energy transfer

### **Disadvantages**

A serious disadvantage of ray tracing is performance (though it can in theory be faster than traditional scanline rendering depending on scene complexity vs. number of pixels on-screen). Scanline algorithms and other algorithms use data coherence to share computations between pixels, while ray tracing normally starts the process anew, treating each eye ray separately. However, this separation offers other advantages, such as the ability to shoot more rays as needed to perform [spatial anti-aliasing](https://en.wikipedia.org/wiki/Spatial_anti-aliasing) and improve image quality where needed.

Although it does handle interreflection and optical effects such as refraction accurately, traditional ray tracing is also not necessarily photorealistic. True photorealism occurs when the [rendering equation](https://en.wikipedia.org/wiki/Rendering_equation) is closely approximated or fully implemented. Implementing the rendering equation gives true photorealism, as the equation describes every physical effect of light flow. However, this is usually infeasible given the computing resources required.

The realism of all rendering methods can be evaluated as an approximation to the equation. Ray tracing, if it is limited to Whitter’s algorithm, is not necessarily the most realistic. Methods that trace rays, but include additional techniques ([photon mapping](https://en.wikipedia.org/wiki/Photon_mapping), [path tracing](https://en.wikipedia.org/wiki/Path_tracing)), give far more accurate simulation of real-world lighting.

**FUTURE SCOPE:**

Ray tracing has long been gaming's holy grail. A method of creating hyper-realistic lighting and graphics, for years ray tracing has been promised as the technology that will take games the next step closer to total realism. Ray tracing has perennially been just on the horizon, but at GDC 2018, both NVIDIA and Microsoft showed off technology that could make real-time ray tracing a reality.

Typical graphics technology, struggle with how light works. Most games used rasterization, which draws a frame almost the same way someone paints a picture, one bit at a time, and with a lot of approximation. Ray tracing hews closer to how light works in the real world, by modeling millions of beams of light, and calculating how they'd bounce around a scene.

This is actually the technique that movie studios use to make modern special effects look so good. But those film studios have render farms of dozens of computers running for hours or even days to ray trace each individual frame in a special effect shot. A game needs to be able to crank out at least 30 frames per second on your one lowly computer. There just hasn't been enough computational power in the average computer to run ray-traced graphics.

But here's where there's some interesting news. At GDC 2018, Microsoft and NVIDIA demoed new technology that supports ray-tracing in games. Microsoft's tech, called DXR, adds software support for ray tracing to DirectX 12, the toolset that underpins most Windows games. NVIDIA announced support for DXR as well as hardware-acceleration for real-time raytracing for its Volta line of graphics cards. AMD also announced driver support for DXR, and some updates to its own ray tracing technology called Radeon Rays.

NVIDIA's RTX is potentially exciting. Hardware acceleration is part of what makes things like playing back HD video so smooth on most modern computers, and it's probably crucial to real-time ray tracing. NVIDIA, Microsoft, Epic Games, Remedy and a few other studios have shown off incredible demos of ultra-high-quality game footage using ray tracing and being rendered in real-time.

It's a huge technical achievement to get footage like this rendering in real time, but there are a few reasons why it may still be a while before you see ray tracing in your favorite games.

**REFERENCES**

<https://en.wikipedia.org/wiki/Ray_tracing_(graphics)>

https://whatis.techtarget.com/definition/ray-tracing-raytracing-ray-tracing-or-ray-casting

<http://www2.cs.man.ac.uk/~raym8/comp37111/main/node43.html>

<https://cs.stanford.edu/people/eroberts/courses/soco/projects/1997-98/raytracing/applications.html>

<https://www.engadget.com/2018/04/16/the-future-of-ray-tracing-explainer/>