God Rays in Modern Gaming

Utsav Jambusaria
Student
D.J.Sanghvi College of Engineering, Mumbai, India

Neerja Katwala
Student
D.J.Sanghvi College of Engineering, Mumbai, India

Khushali Deulkar
Asst. Prof.
D.J.Sanghvi College of Engineering, Mumbai, India

ABSTRACT
The importance of lighting in games is unquestionably tremendous. Not to mention we need to see what we're doing, but to generate a great degree of realism, the lighting too needs to be as realistic. The correct depiction of interaction of light with different surfaces is required. The complexities that arise from trying to render global illumination is multiplied tenfold when the environment itself is interactive, as opposed to it just being there in the background. This is further complicated when the issue of rendering caustics is considered, as this can only be satisfactorily achieved if the light rays (or photons) are traced right back to their sources. The aim of our contribution is to provide a review of the currently used techniques for the real-time rendering of God rays, complete with their advantages and disadvantages to depict exactly how far we have come in this field. And despite all the modern technology available at our fingertips, we shall see that there is a lot of work yet to be done to ensure cheap and accurate representations of stunning visual effects like God rays for the enjoyment of the average gamer.

General Terms
Rendering Algorithms

Keywords
Lighting, caustics, God rays, illumination, games

1. INTRODUCTION
In computer graphics, the articulation of various lighting effects accurately is imperative for achieving a high level of accuracy and authenticity in the eventual illumination of a rendered scene. While there are several different approaches for the same, one must keep in mind that these are simply approximation of the actual equations proven by physics, as we are yet to reach the stage of technological advancement in the field of computer hardware that allows for a cent percent accurate representation of these natural phenomena in real time computer imagery.

For effects like God rays, we predominantly have methods which involve post-processing or shadow maps. This again allows us fairly accurate and visually appealing results in real time at a relatively high, but still achievable cost. That is, even though the methods require a considerable amount of computational power, we do have the hardware (GPU & CPU) capable of providing it.

We can thus see that for the success of any such rendering algorithm, the following two criteria are absolutely imperative:

- It should produce visually appealing and realistic results.
- The computation cost should to as low as possible, to enable rendering in real-time, as is required while playing games.

The greatest challenge we face is the fact that these are to be rendered within a split second, or in real-time as games are highly interactive, and every given input must be instantaneously processed and must result in an appropriate output, without any lag. Thus the screen image could potentially change multiple times in a single second itself. Any delay in this matter is simply unacceptable. For example, it makes no sense if your racing car responds to your commands telling it to turn, but after a few seconds, just to ensure the scene has been rendered to look realistic enough.

2. CAUSTICS
Caustics are patterns formed when light reflects or refracts from some specular or curved surface, and thus focus only on certain areas of the receiving non-specular surface. These are most commonly found on the floors of swimming pools[1] and bath tubs, assuming the water is clean enough to see through. There are two kinds of caustics possible, catacaustics and diacaustic.

Catacaustics are formed by the reflection of light via a curved specular surface. These can be observed at the base of a glass, where the light gets diagonally reflected onto the bottom of the glass.

Diacaustic are the more fascinating. They occur when light refracts through a transparent specular body, like water. These are especially pleasing when the water is not still, and the focus of the light keeps on changing at the receiving surface as a result.

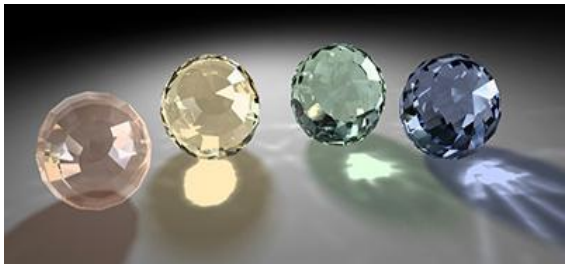

Figure 1: Caustics through glass spheres

3. GOD RAYS
God Rays, or crepuscular rays are the term used for shafts of sunlight that seem to be radiating outward from where the sun is located in the sky[2]. Often, the sun is partly occluded by a shadow inducing object like clouds or trees or buildings, and the sunlight streams through the gaps between or within them. While these beams of light appear to be diverging outward, that is just a matter of perspective as they are, in fact, virtually parallel. This concept is just the same as how railway tracks seem to converge together if we look towards the horizon. Here, the horizon is the sun and the tracks are the shafts of light.

What makes these rays visible as actual 'rays', is the presence of participating media in the medium travelled by the light rays. In this case, it can be the dust or moisture particles present in the air. Light falls on them and reflects, thus making all the particles in its path visible in the form of rays.

Figure 2: A classic example of God Rays

4. POST-PROCESSING PIPELINE TO RENDER GOD RAYS

This method can be used to render light beams caused by directly glancing at a partly occluded light source. In other words, this method fails if the source does not lie within the image borders.

The fundamental principle of their approach is to bloom and blur parts of the scene around the position of the sun to simulate overexposure.[3]This post-processing is done in seven steps and three render targets- the final image, Temp0 and Temp1, which contain the same image with one-sixteenth the resolution of the original image.

4.1 Algorithm

- In the first step the scene is rendered into the main render target.
- The main render target is down-sampled 4 times into Temp0.
- Temp0 is horizontally blurred into Temp1, in which Bloom-Effect, Star-Effect and tone-mapping are added.
- Temp1 is vertically blurred into Temp0.
- Steps five and six are the main steps of the method. In step five a Radial Glow Mask[4] is calculated. The purpose of this mask is the simulation of pixel glowing, at which pixel farther away from the sun exhibit little to no glowing. The mask is applied to the blurred image of Temp0 by positioning a greyscale gradient texture on the blurred image at the sun's position. The texture's and Temp0's pixels are then multiplied and saved in Temp1. By scaling the texture's size or intensity different appearances of the final effect can be achieved. The Radial Glow Mask is implemented with a vertex and pixel shader.
- Subsequently we calculate the Radial Glow Illumination from Temp1 and save it in Temp0. This is done by means of a gather operation. For each pixel, a line connecting it to the sun's screen-space position is computed. Along this line n samples are placed. These samples are weighted according to the sun's distance. The highest value is at the sun's centre, it decreases with increasing distance from the sun. The output of this operation is the weighted sum of the radial glow mask's texels at the sample points. This way the pixel intensities are blurred from the sun's centre outward.
- To get the final image Temp0 has to be added to the main render target.

The main advantage of this method is that a complex scene has no effect of the rendering time as his method involves only post-processing of the output frame.

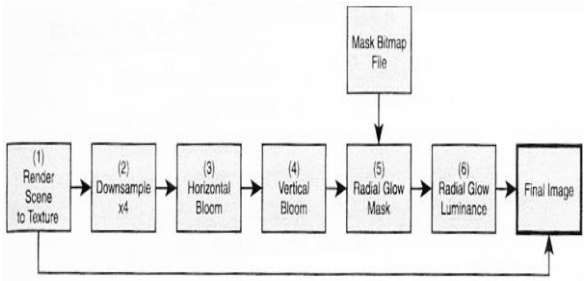

Figure 3: Post Processing Pipelining

5. GOD RAYS USING SHADOW MAPS

A method was introduced to render god rays in real-time by using shadow maps to detect the occlusion in the light beams and interleaved sampling to reduce the computation cost[5]. It is derived from the classic Ray Marching algorithm[6],along with further extensions.

5.1 Algorithm

- Ray Casting: A ray is cast through the volume from the viewer, for each pixel of the final image.
- Sampling: Along the part of the ray that lies within the volume, equidistant sampling points or samples are selected. The sample will generally be in between voxels, or volume pixels, hence it is necessary to interpolate the values of the samples from the surrounding voxels.
- Shading: For each sampling point, a gradient of illumination values is computed. The samples are then shaded (coloured and lit) according to the surface orientation and the position of the light source in the scene.
- Compositing: Once all samples have been shaded, they are composited along the ray of sight, resulting in the final colour value at that pixel. Computation works back to front, i.e. it starts with the sample farthest away from the viewer and ends at the one closest to him. This ensures that the masked parts of the volume do not affect the resulting pixel.

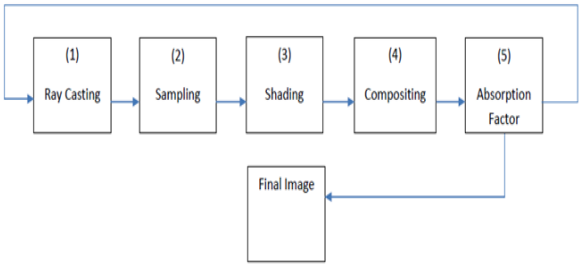

Figure 4: God Rays using Shadow Maps

This does not consider the participating media, which causes a decrease in the radiance between the surface and the eye. For this, the absorption factor $e^{-t(s-l_n)}$for each segment is computed and the final illumination is composed by the sum of their products.

The number of steps N along a ray is proportional to the computing time. Too many steps cause slow rendering, however, too few samples cause image artefacts. A compromise must be found. A good way to solve this problem

is to use interleaved sampling[7] for the ray marching algorithm. This is possible because the surface and the scattering are similar in the neighbouring pixels.

6. COMPARISON

Table 1. Comparison

Post Processing	Shadow Mapping
No computational overhead, because processing happens only after the final scene is rendered.	Significant computation overhead, as the scene involves every ray to be mapped one at a time.
May produce effects that are unnatural.	Gives considerably accurate and natural results.
Can only be used if light source is within the image borders and fails in all other scenarios	Can be used in any scenario. Does not need to have the light source within the image borders.
Doesn't consider shadows within god rays in final rendering of the scene.	Considers shadows first and then renders the final image with god rays.
Quicker, cheaper, but less accurate.	Slower, costlier, and hence more accurate.

7. PROPOSED SOLUTIONS

After having extensively studied the above methods for the real-time rendering of God Rays in interactive gaming applications and gauging their advantages and disadvantages over each other, we have come to the conclusion that while approximating the rendering equations as closely as possible is the right way forward, implementing the same in today's hardware leaves a lot to be desired.

As a result of this, we propose the following solutions:

7.1 Solution 1 - Best of Both Worlds

We appreciate both, the Post-Processing approach and the Shadow Map implementation, and feel that if both of them were merged to form a new algorithm, it could give quicker and more accurate results in all environments.

The Post-Processing approach is excellent with respect to its very low computation cost, but its greatest drawback is that it cannot work if the light source is outside the image borders. To overcome this, we can assume the location of the source to be within the image boundary by taking the global illumination of the scene into account, just for the purpose of calculation.

After this, we can calculate the Shadow Map of the scene as in the second approach, thus saving on computation time for the occluded parts.

Finally, the image is post processed, taking into account the occluded parts.

This ensures that the process can work in every scene, unlike the Post-Processing approach at only a marginal time increase that is necessary for the Shadow Map calculation.

7.2 Solution 2 - Improved Hardware Performance

As discussed earlier, the algorithms for approximating the rendering equations give highly accurate results, and do not need to be worked upon. But a noticeable increase in the hardware performance and efficiency (both, CPU & GPU) will result in a considerable improvement in the scope of the implementation of the above algorithms.

Also considering that what we are using now is just an approximation and not the actual equation. And there is nothing better for rendering reality, than rendering reality itself. That should be our final aim. To use the most accurate algorithms and still manage to efficiently imbibe them into the most fascinating and scintillating computer graphics the world has ever gazed upon.

Because eventually, as software and hardware go hand in hand. As newer software algorithms make it possible for us to incorporate more realistic and visually appealing elements in our project, newer and more powerful hardware should be developed to make it possible for us to practically implement them and turn what was a dream yesterday, into a reality today.

8. CONCLUSION

Lighting effects such as caustics, god rays and light shafts are very important to render realistically appearing scenes. They add to the mood of a scene and to its impression on the viewer. The best results are obtained by sticking close to the physical correct descriptions of light and its behaviour. Also the physical correct specification of the participating media adds to the authenticity of a scene. We introduced several methods which can handle light characteristics, which are normally hard to obtain, such as anisotropic scattering, multiple scattering, colour bleeding, god rays and light radiating through non-homogeneous media. The price of this high accuracy is high computing times and high memory costs. That's why physically correct methods are only used for offline rendering. Nowadays most 3D graphic applications, for instance games and visualizations, require real-time rendering. This is always accomplished by simplifying and approximating the lighting model. By implementing the methods on the GPU more speed can be gained.

9. FUTURE SCOPE

In the future the main task will be the realization of better approximations and the improvement of the computing capabilities of the GPU, to generate physical accurate lighting effects in real-time.

Even further, the aim should be developing the hardware that is capable of rendering not just the approximations, but the actual equations themselves and reproduce nature in its complete beauty on computer screens.

10. REFERENCES

[1] Arvo, J. 1986. Backward ray tracing. Siggraph '86, 259–263.

[2] Kajiya, J. T. 1986. *The rendering equation*, Siggraph '86, 143–150.

[3] Wand, M., Strasser, W. 2003. Real-time caustics. Computer Graphics Forum 22, 3, 611–620.

[4] www.Wikipedia.com/Crepuscular_Rays

[5] B Fruhstuck, A., Prast, S.,Caustics, Light Shafts, God Rays, 2013.

[6] T´oth, B., Umenhoffer, T. 2009. Real-time Volumetric Lighting in Participating Media. Eurographics 2009.

[7] www.Wikipedia.com/Ray_Marching