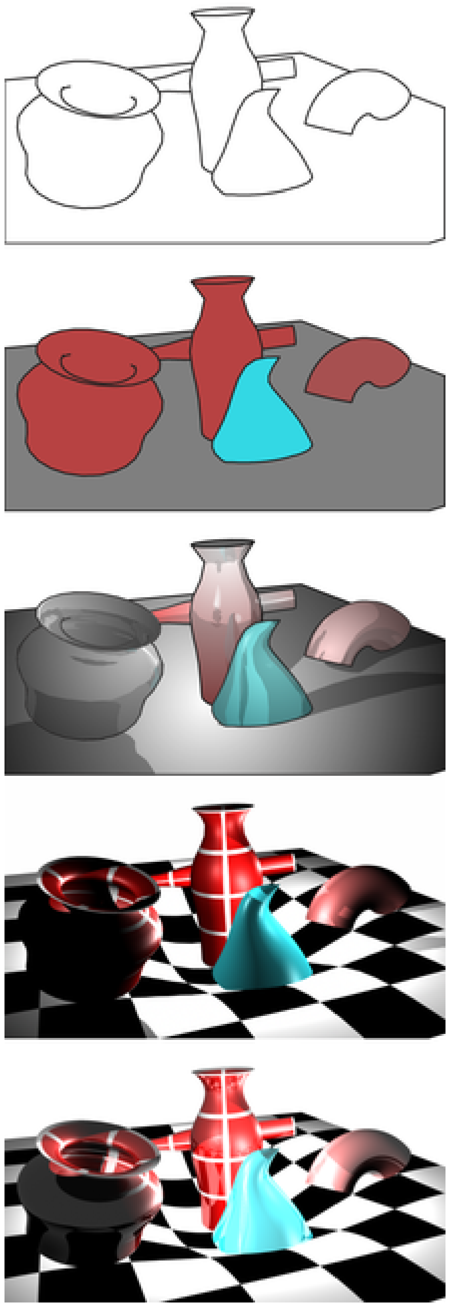
Wikipedia
The Free Encyclopedia

Rendering (computer graphics)

Rendering or **image synthesis** is the process of generating a photorealistic or non-photorealistic image from a 2D or 3D model by means of a computer program. The resulting image is referred to as the **render**. Multiple models can be defined in a *scene file* containing objects in a strictly defined language or data structure. The scene file contains geometry, viewpoint, texture, lighting, and shading information describing the virtual scene. The data contained in the scene file is then passed to a rendering program to be processed and output to a digital image or raster graphics image file. The term "rendering" is analogous to the concept of an artist's impression of a scene. The term "rendering" is also used to describe the process of calculating effects in a video editing program to produce the final video output.


A variety of rendering techniques applied to a single 3D scene

Rendering is one of the major sub-topics of 3D computer graphics, and in practice it is always connected to the others. It is the last major step in the graphics pipeline, giving models and animation their final appearance. With the increasing sophistication of computer graphics since the 1970s, it has become a more distinct subject.

Rendering has uses in architecture, video games, simulators, movie and TV visual effects, and design visualization, each employing a different balance of features and techniques. A wide variety of renderers are available for use. Some are integrated into larger modeling and animation packages, some are stand-alone, and some are free open-source projects. On the inside, a renderer is a carefully engineered program based on multiple disciplines, including light physics, visual perception, mathematics, and software development.

Though the technical details of rendering methods vary, the general challenges to overcome in producing a 2D image on a screen from a 3D representation stored in a scene file are handled by the graphics pipeline in a rendering device such as a GPU. A GPU is a purpose-built device that assists a CPU in performing complex rendering calculations. If a scene is to look relatively realistic and predictable under virtual lighting, the rendering software must solve the rendering equation. The rendering equation doesn't account for all lighting phenomena, but instead acts as a general lighting model for computer-generated imagery.

In the case of 3D graphics, scenes can be pre-rendered or generated in realtime. Pre-rendering is a slow, computationally intensive process that is typically used for movie creation, where scenes can be generated ahead of time, while real-time rendering is often done for 3D video games and other applications that must dynamically create scenes. 3D hardware accelerators can improve realtime rendering performance.

Usage

An image created by using POV-Ray 3.6

When the pre-image (a wireframe sketch usually) is complete, rendering is used, which adds in bitmap textures or procedural textures, lights, bump mapping and relative position to other objects. The result is a completed image the consumer or intended viewer sees.

For movie animations, several images (frames) must be rendered, and stitched together in a program capable of making an animation of this sort. Most 3D image editing programs can do this.

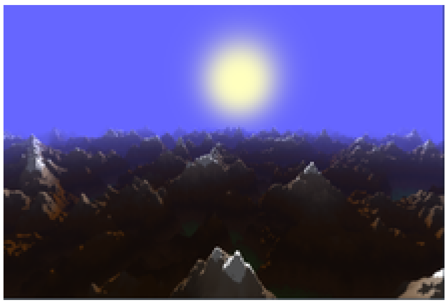
Features

A rendered image can be understood in terms of a number of visible features. Rendering research and development has been largely motivated by finding ways to simulate these efficiently. Some relate directly to particular algorithms and techniques, while others are produced together.

- Shading – how the color and brightness of a surface varies with lighting
- Texture-mapping – a method of applying detail to surfaces
- Bump-mapping – a method of simulating small-scale bumpiness on surfaces
- Fogging/participating medium – how light dims when passing through non-clear atmosphere or air
- Shadows – the effect of obstructing light
- Soft shadows – varying darkness caused by partially obscured light sources
- Reflection – mirror-like or highly glossy reflection
- Transparency (optics), transparency (graphic) or opacity – sharp transmission of light through solid objects
- Translucency – highly scattered transmission of light through solid objects
- Refraction – bending of light associated with transparency
- Diffraction – bending, spreading, and interference of light passing by an object or aperture that disrupts the ray
- Indirect illumination – surfaces illuminated by light reflected off other surfaces, rather than directly from a light source (also known as global illumination)
- Caustics (a form of indirect illumination) – reflection of light off a shiny object, or focusing of light through a transparent object, to produce bright highlights on another object
- Depth of field – objects appear blurry or out of focus when too far in front of or behind the object in focus
- Motion blur – objects appear blurry due to high-speed motion, or the motion of the camera
- Non-photorealistic rendering – rendering of scenes in an artistic style, intended to look like a painting or drawing

Techniques

Many rendering **algorithms** have been researched, and software used for rendering may employ a number of different techniques to obtain a final image.


Rendering of a fractal terrain by ray marching

Tracing every particle of light in a scene is nearly always completely impractical and would take a stupendous amount of time. Even tracing a portion large enough to produce an image takes an inordinate amount of time if the sampling is not intelligently restricted.

Therefore, a few loose families of more-efficient light transport modeling techniques have emerged:

- rasterization, including scanline rendering, geometrically projects objects in the scene to an image plane, without advanced optical effects;
- ray casting considers the scene as observed from a specific point of view, calculating the observed image based only on geometry and very basic optical laws of reflection intensity, and perhaps using Monte Carlo techniques to reduce artifacts;
- ray tracing is similar to ray casting, but employs more advanced optical simulation, and usually uses Monte Carlo techniques to obtain more realistic results at a speed that is often orders of magnitude faster.

The fourth type of light transport technique, radiosity is not usually implemented as a rendering technique but instead calculates the passage of light as it leaves the light source and illuminates surfaces. These surfaces are usually rendered to the display using one of the other three techniques.

Most advanced software combines two or more of the techniques to obtain good-enough results at reasonable cost.

Another distinction is between image order algorithms, which iterate over pixels of the image plane, and object order algorithms, which iterate over objects in the scene. Generally object order is more efficient, as there are usually fewer objects in a scene than pixels.

Scanline rendering and rasterization

Rendering of the European Extremely Large Telescope

A high-level representation of an image necessarily contains elements in a different domain from pixels. These elements are referred to as primitives. In a schematic drawing, for instance, line segments and curves might be primitives. In a graphical user interface, windows and buttons might be the primitives. In rendering of 3D models, triangles and polygons in space might be primitives.

If a pixel-by-pixel (image order) approach to rendering is impractical or too slow for some task, then a primitive-by-primitive (object order) approach to rendering may prove useful. Here, one loop through each of the primitives, determines which pixels in the image it affects, and modifies those pixels accordingly. This is called **rasterization**, and is the rendering method used by all current graphics cards.

Rasterization is frequently faster than pixel-by-pixel rendering. First, large areas of the image may be empty of primitives; rasterization will ignore these areas, but pixel-by-pixel rendering must pass through them. Second, rasterization can improve cache coherency and reduce redundant work by taking advantage of the fact that the pixels occupied by a single primitive tend to be contiguous in the image. For these reasons, rasterization is usually the approach of choice when interactive rendering is required; however, the pixel-by-pixel approach can often produce higher-quality images and is more versatile because it does not depend on as many assumptions about the image as rasterization.

The older form of rasterization is characterized by rendering an entire face (primitive) as a single color. Alternatively, rasterization can be done in a more complicated manner by first rendering the vertices of a face and then rendering the pixels of that face as a blending of the vertex colors. This version of rasterization has overtaken the old method as it allows the graphics to flow without complicated textures (a rasterized image when used face by face tends to have a very block-like effect if not covered in complex textures; the faces are not smooth because there is no gradual color change from one primitive to the next). This newer method of rasterization utilizes the graphics card's more taxing shading functions and still achieves better performance because the simpler textures stored in memory use less space. Sometimes designers will use one rasterization method on some faces and the other method on others based on the angle at which that face meets other joined faces, thus increasing speed and not hurting the overall effect.

Ray casting

In **ray casting** the geometry which has been modeled is parsed pixel by pixel, line by line, from the point of view outward, as if casting rays out from the point of view. Where an object is intersected, the color value at the point may be evaluated using several methods. In the simplest, the color value of the object at the point of intersection becomes the value of that pixel. The color may be determined from a texture-map. A more sophisticated method is to modify the color value by an illumination factor, but without calculating the relationship to a simulated light source. To reduce artifacts, a number of rays in slightly different directions may be averaged.

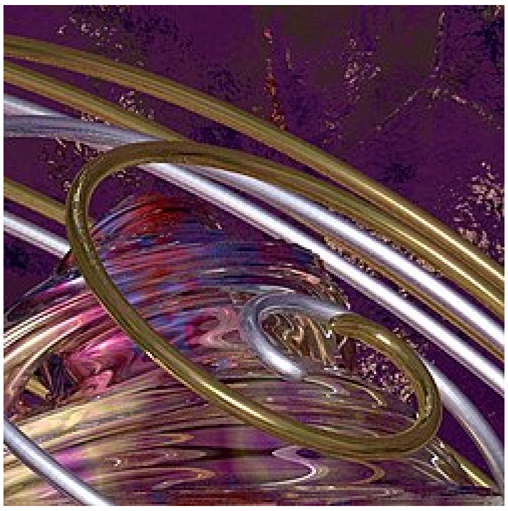
Ray casting involves calculating the "view direction" (from camera position), and incrementally following along that "ray cast" through "solid 3d objects" in the scene, while accumulating the resulting value from each point in 3D space. This is related and similar to "ray tracing" except that the raycast is usually not "bounced" off surfaces (where the "ray tracing" indicates that it is tracing out the lights path including bounces). "Ray casting" implies that the light ray is following a straight path (which may include traveling through semi-transparent objects). The ray cast is a vector that can originate from the camera or from the scene endpoint ("back to front", or "front to back"). Sometimes the final light value is derived from a "transfer function" and sometimes it's used directly.

Rough simulations of optical properties may be additionally employed: a simple calculation of the ray from the object to the point of view is made. Another calculation is made of the angle of incidence of light rays from the light source(s), and from these as well as the specified intensities of the light sources, the value of the pixel is calculated. Another simulation uses illumination plotted from a radiosity algorithm, or a combination of these two.

Ray tracing

Ray tracing aims to simulate the natural flow of light, interpreted as particles. Often, ray tracing methods are utilized to approximate the solution to the rendering equation by applying Monte Carlo methods to it. Some of the most used methods are path tracing, bidirectional path tracing, or Metropolis light transport, but also semi realistic methods are in use, like Whitted Style Ray Tracing, or hybrids. While most implementations let light propagate on straight lines, applications exist to simulate relativistic spacetime effects.[1]

In a final, production quality rendering of a ray traced work, multiple rays are generally shot for each pixel, and traced not just to the first object of intersection, but rather, through a number of sequential 'bounces', using the known laws of optics such as "angle of incidence equals angle of reflection" and more advanced laws that deal with refraction and surface roughness.


Spiral Sphere and Julia, Detail, a computer-generated image created by visual artist Robert W. McGregor using only POV-Ray 3.6 and its built-in scene description language

Once the ray either encounters a light source, or more probably once a set limiting number of bounces has been evaluated, then the surface illumination at that final point is evaluated using techniques described above, and the changes along the way through the various bounces evaluated to estimate a value observed at the point of view. This is all repeated for each sample, for each pixel.

In distribution ray tracing, at each point of intersection, multiple rays may be spawned. In path tracing, however, only a single ray or none is fired at each intersection, utilizing the statistical nature of Monte Carlo experiments.

As a brute-force method, ray tracing has been too slow to consider for real-time, and until recently too slow even to consider for short films of any degree of quality, although it has been used for special effects sequences, and in advertising, where a short portion of high quality (perhaps even photorealistic) footage is required.

However, efforts at optimizing to reduce the number of calculations needed in portions of a work where detail is not high or does not depend on ray tracing features have led to a realistic possibility of wider use of ray tracing. There is now some hardware accelerated ray tracing equipment, at least in prototype phase, and some game demos which show use of real-time software or hardware ray tracing.

Neural rendering

Neural rendering is a rendering method using artificial neural networks.[2][3] Neural rendering includes image-based rendering methods that are used to reconstruct 3D models from 2-dimensional images.[2]One of these methods are photogrammetry, which is a method in which a collection of images from multiple angles of an object are turned into a 3D model. There have also been recent developments in generating and rendering 3D models from text and coarse paintings by notably NVIDIA, Google and various other companies.

Radiosity

Radiosity is a method which attempts to simulate the way in which directly illuminated surfaces act as indirect light sources that illuminate other surfaces. This produces more realistic shading and seems to better capture the 'ambience' of an indoor scene. A classic example is a way that shadows 'hug' the corners of rooms.

The optical basis of the simulation is that some diffused light from a given point on a given surface is reflected in a large spectrum of directions and illuminates the area around it.

The simulation technique may vary in complexity. Many renderings have a very rough estimate of radiosity, simply illuminating an entire scene very slightly with a factor known as ambiance. However, when advanced radiosity estimation is coupled with a high quality ray tracing algorithm, images may exhibit convincing realism, particularly for indoor scenes.

In advanced radiosity simulation, recursive, finite-element algorithms 'bounce' light back and forth between surfaces in the model, until some recursion limit is reached. The colouring of one surface in this way influences the colouring of a neighbouring surface, and vice versa. The resulting values of illumination throughout the model (sometimes including for empty spaces) are stored and used as additional inputs when performing calculations in a ray-casting or ray-tracing model.

Due to the iterative/recursive nature of the technique, complex objects are particularly slow to emulate. Prior to the standardization of rapid radiosity calculation, some digital artists used a technique referred to loosely as false radiosity by darkening areas of texture maps corresponding to corners, joints and recesses, and applying them via self-illumination or diffuse mapping for scanline rendering. Even now, advanced radiosity calculations may be reserved for calculating the ambiance of the room, from the light reflecting off walls, floor and ceiling, without examining the contribution that complex objects make to the radiosity – or complex objects may be replaced in the radiosity calculation with simpler objects of similar size and texture.

Radiosity calculations are viewpoint independent which increases the computations involved, but makes them useful for all viewpoints. If there is little rearrangement of radiosity objects in the scene, the same radiosity data may be reused for a number of frames, making radiosity an effective way to improve on the flatness of ray casting, without seriously impacting the overall rendering time-per-frame.

Because of this, radiosity is a prime component of leading real-time rendering methods, and has been used from beginning-to-end to create a large number of well-known recent feature-length animated 3D-cartoon films.

Sampling and filtering

One problem that any rendering system must deal with, no matter which approach it takes, is the **sampling problem**. Essentially, the rendering process tries to depict a continuous function from image space to colors by using a finite number of pixels. As a consequence of the Nyquist–Shannon sampling theorem (or Kotelnikov theorem), any spatial waveform that can be displayed must consist of at least two pixels, which is proportional to image resolution. In simpler terms, this expresses the idea that an image cannot display details, peaks or troughs in color or intensity, that are smaller than one pixel.

If a naive rendering algorithm is used without any filtering, high frequencies in the image function will cause ugly aliasing to be present in the final image. Aliasing typically manifests itself as jaggies, or jagged edges on objects where the pixel grid is visible. In order to remove aliasing, all rendering algorithms (if they are to produce good-looking images) must use some kind of low-pass filter on the image function to remove high frequencies, a process called antialiasing.

Optimization

Due to the large number of calculations, a work in progress is usually only rendered in detail appropriate to the portion of the work being developed at a given time, so in the initial stages of modeling, wireframe and ray casting may be used, even where the target output is ray tracing with radiosity. It is also common to render only parts of the scene at high detail, and to remove objects that are not important to what is currently being developed.

For real-time, it is appropriate to simplify one or more common approximations, and tune to the exact parameters of the scenery in question, which is also tuned to the agreed parameters to get the most 'bang for the buck'.

Academic core

The implementation of a realistic renderer always has some basic element of physical simulation or emulation – some computation which resembles or abstracts a real physical process.

The term "*physically based*" indicates the use of physical models and approximations that are more general and widely accepted outside rendering. A particular set of related techniques have gradually become established in the rendering community.

The basic concepts are moderately straightforward, but intractable to calculate; and a single elegant algorithm or approach has been elusive for more general purpose renderers. In order to meet demands of robustness, accuracy and practicality, an implementation will be a complex combination of different techniques.

Rendering research is concerned with both the adaptation of scientific models and their efficient application.

The rendering equation

This is the key academic/theoretical concept in rendering. It serves as the most abstract formal expression of the non-perceptual aspect of rendering. All more complete algorithms can be seen as solutions to particular formulations of this equation.

$$L_o(x,\vec{w}) = L_e(x,\vec{w}) + \int_\Omega f_r(x,\vec{w}',\vec{w}) L_i(x,\vec{w}')(\vec{w}'\cdot\vec{n})\mathrm{d}\vec{w}'$$

Meaning: at a particular position and direction, the outgoing light (L_o) is the sum of the emitted light (L_e) and the reflected light. The reflected light being the sum of the incoming light (L_i) from all directions, multiplied by the surface reflection and incoming angle. By connecting outward light to inward light, via an interaction point, this equation stands for the whole 'light transport' – all the movement of light – in a scene.

The bidirectional reflectance distribution function

The **bidirectional reflectance distribution function** (BRDF) expresses a simple model of light interaction with a surface as follows:

$$f_r(x,\vec{w}',\vec{w}) = \frac{\mathrm{d}L_r(x,\vec{w})}{L_i(x,\vec{w}')(\vec{w}'\cdot\vec{n})\mathrm{d}\vec{w}'}$$

Light interaction is often approximated by the even simpler models: diffuse reflection and specular reflection, although both can ALSO be BRDFs.

Geometric optics

Rendering is practically exclusively concerned with the particle aspect of light physics – known as geometrical optics. Treating light, at its basic level, as particles bouncing around is a simplification, but appropriate: the wave aspects of light are negligible in most scenes, and are significantly more difficult to simulate. Notable wave aspect phenomena include diffraction (as seen in the colours of CDs and DVDs) and polarisation (as seen in LCDs). Both types of effect, if needed, are made by appearance-oriented adjustment of the reflection model.

Visual perception

Though it receives less attention, an understanding of human visual perception is valuable to rendering. This is mainly because image displays and human perception have restricted ranges. A renderer can simulate a wide range of light brightness and color, but current displays – movie screen, computer monitor, etc. – cannot handle so much, and something must be discarded or compressed. Human perception also has limits, and so does not need to be given large-range images to create realism. This can help solve the problem of fitting images into displays, and, furthermore, suggest what short-cuts could be used in the rendering simulation, since certain subtleties won't be noticeable. This related subject is tone mapping.

Mathematics used in rendering includes: linear algebra, calculus, numerical mathematics, signal processing, and Monte Carlo methods.

Rendering for movies often takes place on a network of tightly connected computers known as a render farm.

The current state of the art in 3-D image description for movie creation is the Mental Ray scene description language designed at Mental Images and RenderMan Shading Language designed at Pixar[4] (compare with simpler 3D fileformats such as VRML or APIs such as OpenGL and DirectX tailored for 3D hardware accelerators).

Other renderers (including proprietary ones) can and are sometimes used, but most other renderers tend to miss one or more of the often needed features like good texture filtering, texture caching, programmable shaders, highend geometry types like hair, subdivision or nurbs surfaces with tesselation on demand, geometry caching, raytracing with geometry caching, high quality shadow mapping, speed or patent-free implementations. Other highly sought features these days may include interactive photorealistic rendering (IPR) and hardware rendering/shading.

Chronology of important published ideas

- 1968 *Ray casting*[5]
- 1970 *Scanline rendering*[6]
- 1971 *Gouraud shading*[7]
- 1973 *Phong shading*[8][9]
- 1973 *Phong reflection*[8]
- 1973 *Diffuse reflection*[10]
- 1973 *Specular highlight*[8]
- 1973 *Specular reflection*[8]
- 1974 *Sprites*[11]
- 1974 *Scrolling*[11]
- 1974 *Texture mapping*[12]

Rendering of an ESTCube-1 satellite

- 1974 *Z-buffering*[12]
- 1976 *Environment mapping*[13]
- 1977 *Blinn shading*[14]
- 1977 *Side-scrolling*[15]
- 1977 *Shadow volumes*[16]
- 1978 *Shadow mapping*[17]
- 1978 *Bump mapping*[18]
- 1979 *Tile map*[19]
- 1980 *BSP trees*[20]
- 1980 *Ray tracing*[21]
- 1981 *Parallax scrolling*[22]
- 1981 *Sprite zooming*[23]
- 1981 *Cook shader*[24]
- 1983 *MIP maps*[25]
- 1984 *Octree ray tracing*[26]
- 1984 *Alpha compositing*[27]
- 1984 *Distributed ray tracing*[28]
- 1984 *Radiosity*[29]
- 1985 *Row/column scrolling*[30]
- 1985 *Hemicube radiosity*[31]
- 1986 *Light source tracing*[32]
- 1986 *Rendering equation*[33]
- 1987 *Reyes rendering*[34]
- 1988 *Depth cue*[35]
- 1988 *Distance fog*[35]
- 1988 *Tiled rendering*[35]
- 1991 *Xiaolin Wu line anti-aliasing*[36][37]
- 1991 *Hierarchical radiosity*[38]
- 1993 *Texture filtering*[39]
- 1993 *Perspective correction*[40]
- 1993 *Transform, clipping, and lighting*[41]
- 1993 *Directional lighting*[41]
- 1993 *Trilinear interpolation*[41]
- 1993 *Z-culling*[41]
- 1993 *Oren–Nayar reflectance*[42]
- 1993 *Tone mapping*[43]
- 1993 *Subsurface scattering*[44]
- 1994 *Ambient occlusion*[45]
- 1995 *Hidden-surface determination*[46]
- 1995 *Photon mapping*[47]
- 1996 *Multisample anti-aliasing*[48]

- 1997 *Metropolis light transport*[49]
- 1997 *Instant Radiosity*[50]
- 1998 *Hidden-surface removal*[51]
- 2000 *Pose space deformation*[52]
- 2002 *Precomputed Radiance Transfer*[53]

See also

- 2D computer graphics – Computer-based generation of digital images
- 3D computer graphics – Graphics that use a three-dimensional representation of geometric data
- 3D rendering – Process of converting 3D scenes into 2D images
- Artistic rendering – Style of rendering
- Architectural rendering – creating two-dimensional images or animations showing the attributes of a proposed architectural design
- Chromatic aberration – Failure of a lens to focus all colors on the same point
- Displacement mapping – Computer graphics technique
- Font rasterization – Process of converting text from vector to raster
- Global illumination – Group of rendering algorithms used in 3D computer graphics
- Graphics pipeline – Procedure to convert 3D scenes to 2D images
- Heightmap – Type of raster image in computer graphics
- High-dynamic-range rendering – Rendering of computer graphics scenes by using lighting calculations done in high-dynamic-range
- Image-based modeling and rendering
- Motion blur – Photography artifact from moving objects
- Non-photorealistic rendering – Style of rendering
- Normal mapping – Texture mapping technique
- Painter's algorithm – Algorithm for visible surface determination in 3D graphics
- Per-pixel lighting
- Physically based rendering – Computer graphics technique
- Pre-rendering
- Raster image processor – component used in a printing system which produces a raster image also known as a bitmap
- Radiosity – Computer graphics rendering method using diffuse reflection
- Ray tracing – Rendering method
- Real-time computer graphics – Sub-field of computer graphics
- Reyes – Computer software architecture in 3D computer graphics
- Scanline rendering/Scanline algorithm – 3D computer graphics image rendering method
- Software rendering
- Sprite (computer graphics) – 2D bitmap displayed on top of a larger scene
- Unbiased rendering – Type of rendering in computer graphics
- Vector graphics – Computer graphics images defined by points, lines and curves
- VirtualGL
- Virtual model – Form of computer-aided engineering
- Virtual studio – Technology for television and film production
- Volume rendering – Representing a 3D-modeled object or dataset as a 2D projection

- Z-buffer algorithms – Type of data buffer in computer graphics

References

1. "Relativistic Ray-Tracing: Simulating the Visual Appearance of Rapidly Moving Objects". 1995. CiteSeerX 10.1.1.56.830 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.56.830).
2. Tewari, A.; Fried, O.; Thies, J.; Sitzmann, V.; Lombardi, S.; Sunkavalli, K.; Martin-Brualla, R.; Simon, T.; Saragih, J.; Nießner, M.; Pandey, R.; Fanello, S.; Wetzstein, G.; Zhu, J.-Y.; Theobalt, C.; Agrawala, M.; Shechtman, E.; Goldman, D. B.; Zollhöfer, M. (2020). "State of the Art on Neural Rendering" (https://onlinelibrary.wiley.com/doi/am-pdf/10.1111/cgf.14022). *Computer Graphics Forum*. **39** (2): 701–727. arXiv:2004.03805 (https://arxiv.org/abs/2004.03805). doi:10.1111/cgf.14022 (https://doi.org/10.1111%2Fcgf.14022). S2CID 215416317 (https://api.semanticscholar.org/CorpusID:215416317).
3. Knight, Will. "A New Trick Lets Artificial Intelligence See in 3D" (https://www.wired.com/story/new-way-ai-see-3d/). *Wired*. ISSN 1059-1028 (https://www.worldcat.org/issn/1059-1028). Retrieved 2022-02-08.
4. Raghavachary, Saty (30 July 2006). "A brief introduction to RenderMan" (http://dl.acm.org/citation.cfm?id=1185657.1185817). *ACM SIGGRAPH 2006 Courses on - SIGGRAPH '06*. ACM. p. 2. doi:10.1145/1185657.1185817 (https://doi.org/10.1145%2F1185657.1185817). ISBN 978-1595933645. S2CID 34496605 (https://api.semanticscholar.org/CorpusID:34496605). Retrieved 7 May 2018 – via dl.acm.org.
5. Appel, A. (1968). "Some techniques for shading machine renderings of solids" (http://graphics.stanford.edu/courses/Appel.pdf) (PDF). *Proceedings of the Spring Joint Computer Conference*. Vol. 32. pp. 37–49. Archived (https://web.archive.org/web/20120313214110/http://graphics.stanford.edu/courses/Appel.pdf) (PDF) from the original on 2012-03-13.
6. Bouknight, W. J. (1970). "A procedure for generation of three-dimensional half-tone computer graphics presentations". *Communications of the ACM*. **13** (9): 527–536. doi:10.1145/362736.362739 (https://doi.org/10.1145%2F362736.362739). S2CID 15941472 (https://api.semanticscholar.org/CorpusID:15941472).
7. Gouraud, H. (1971). "Continuous shading of curved surfaces" (https://web.archive.org/web/20100702012343/http://www.cs.uiowa.edu/~cwyman/classes/spring05-22C251/papers/ContinuousShadingOfCurvedSurfaces.pdf) (PDF). *IEEE Transactions on Computers*. **20** (6): 623–629. doi:10.1109/t-c.1971.223313 (https://doi.org/10.1109%2Ft-c.1971.223313). S2CID 123827991 (https://api.semanticscholar.org/CorpusID:123827991). Archived from the original (http://www.cs.uiowa.edu/~cwyman/classes/spring05-22C251/papers/ContinuousShadingOfCurvedSurfaces.pdf) (PDF) on 2010-07-02.
8. "History | School of Computing" (https://www.cs.utah.edu/about/history/). Archived (https://web.archive.org/web/20131203035242/http://www.cs.utah.edu/dept/history/) from the original on 2013-12-03. Retrieved 2021-11-22.
9. Phong, B-T (1975). "Illumination for computer generated pictures" (https://web.archive.org/web/20120327165141/http://jesper.kalliope.org/blog/library/p311-phong.pdf) (PDF). *Communications of the ACM*. **18** (6): 311–316. CiteSeerX 10.1.1.330.4718 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.330.4718). doi:10.1145/360825.360839 (https://doi.org/10.1145%2F360825.360839). S2CID 1439868 (https://api.semanticscholar.org/CorpusID:1439868). Archived from the original (http://jesper.kalliope.org/blog/library/p311-phong.pdf) (PDF) on 2012-03-27.
10. Bui Tuong Phong, Illumination for computer generated pictures (http://www.cs.northwestern.edu/~ago820/cs395/Papers/Phong_1975.pdf) Archived (https://web.archive.org/web/20160320142825/http://www.cs.northwestern.edu/~ago820/cs395/Papers/Phong_1975.pdf) 2016-03-20 at the Wayback Machine, Communications of ACM 18 (1975), no. 6, 311–317.

11. Putas. "The way to home 3d" (http://vintage3d.org/history.php). *vintage3d.org*. Archived (https://web.archive.org/web/20171215001253/http://vintage3d.org/history.php) from the original on 15 December 2017. Retrieved 7 May 2018.
12. Catmull, E. (1974). *A subdivision algorithm for computer display of curved surfaces* (https://web.archive.org/web/20141114173944/http://www.pixartouchbook.com/storage/catmull_thesis.pdf) (PDF) (PhD thesis). University of Utah. Archived from the original (http://www.pixartouchbook.com/storage/catmull_thesis.pdf) (PDF) on 2014-11-14. Retrieved 2011-07-15.
13. Blinn, J.F.; Newell, M.E. (1976). "Texture and reflection in computer generated images". *Communications of the ACM*. **19** (10): 542–546. CiteSeerX 10.1.1.87.8903 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.87.8903). doi:10.1145/360349.360353 (https://doi.org/10.1145%2F360349.360353). S2CID 408793 (https://api.semanticscholar.org/CorpusID:408793).
14. Blinn, James F. (20 July 1977). "Models of light reflection for computer synthesized pictures" (https://doi.org/10.1145%2F965141.563893). *ACM SIGGRAPH Computer Graphics*. **11** (2): 192–198. doi:10.1145/965141.563893 (https://doi.org/10.1145%2F965141.563893) – via dl.acm.org.
15. "Bomber - Videogame by Sega" (https://web.archive.org/web/20171017194023/https://www.arcade-museum.com/game_detail.php?game_id=12797). *www.arcade-museum.com*. Archived from the original (http://www.arcade-museum.com/game_detail.php?game_id=12797) on 17 October 2017. Retrieved 7 May 2018.
16. Crow, F.C. (1977). "Shadow algorithms for computer graphics" (https://web.archive.org/web/20120113074712/https://design.osu.edu/carlson/history/PDFs/crow-shadows.pdf) (PDF). *Computer Graphics (Proceedings of SIGGRAPH 1977)*. Vol. 11. pp. 242–248. Archived from the original (http://design.osu.edu/carlson/history/PDFs/crow-shadows.pdf) (PDF) on 2012-01-13. Retrieved 2011-07-15.
17. Williams, L. (1978). "Casting curved shadows on curved surfaces". *Computer Graphics (Proceedings of SIGGRAPH 1978)*. Vol. 12. pp. 270–274. CiteSeerX 10.1.1.134.8225 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.134.8225).
18. Blinn, J.F. (1978). *Simulation of wrinkled surfaces* (http://research.microsoft.com/pubs/73939/p286-blinn.pdf) (PDF). Computer Graphics (Proceedings of SIGGRAPH 1978). Vol. 12. pp. 286–292. Archived (https://web.archive.org/web/20120121143525/http://research.microsoft.com/pubs/73939/p286-blinn.pdf) (PDF) from the original on 2012-01-21.
19. Wolf, Mark J. P. (15 June 2012). *Before the Crash: Early Video Game History* (https://books.google.com/books?id=oK3D4i5ldKgC&pg=PA173). Wayne State University Press. ISBN 978-0814337226. Archived (https://web.archive.org/web/20190502053839/https://books.google.com/books?id=oK3D4i5ldKgC&pg=PA173) from the original on 2 May 2019. Retrieved 7 May 2018 – via Google Books.
20. Fuchs, H.; Kedem, Z.M.; Naylor, B.F. (1980). *On visible surface generation by a priori tree structures*. Computer Graphics (Proceedings of SIGGRAPH 1980). Vol. 14. pp. 124–133. CiteSeerX 10.1.1.112.4406 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.112.4406).
21. Whitted, T. (1980). "An improved illumination model for shaded display". *Communications of the ACM*. **23** (6): 343–349. CiteSeerX 10.1.1.114.7629 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.114.7629). doi:10.1145/358876.358882 (https://doi.org/10.1145%2F358876.358882). S2CID 9524504 (https://api.semanticscholar.org/CorpusID:9524504).
22. Purcaru, Bogdan Ion (13 March 2014). "Games vs. Hardware. The History of PC video games: The 80's" (https://books.google.com/books?id=lB4PAwAAQBAJ&pg=PA181). Purcaru Ion Bogdan. Archived (https://web.archive.org/web/20210430132633/https://books.google.com/books?id=lB4PAwAAQBAJ&pg=PA181) from the original on 30 April 2021. Retrieved 7 May 2018 – via Google Books.

23. "System 16 - Sega VCO Object Hardware (Sega)" (https://web.archive.org/web/20160405061411/http://www.system16.com/hardware.php?id=690). *www.system16.com*. Archived from the original (http://www.system16.com/hardware.php?id=690) on 5 April 2016. Retrieved 7 May 2018.
24. Cook, R.L.; Torrance, K.E. (1981). *A reflectance model for computer graphics*. Computer Graphics (Proceedings of SIGGRAPH 1981). Vol. 15. pp. 307–316. CiteSeerX 10.1.1.88.7796 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.88.7796).
25. Williams, L. (1983). *Pyramidal parametrics*. Computer Graphics (Proceedings of SIGGRAPH 1983). Vol. 17. pp. 1–11. CiteSeerX 10.1.1.163.6298 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.163.6298).
26. Glassner, A.S. (1984). "Space subdivision for fast ray tracing". *IEEE Computer Graphics & Applications*. **4** (10): 15–22. doi:10.1109/mcg.1984.6429331 (https://doi.org/10.1109%2Fmcg.1984.6429331). S2CID 16965964 (https://api.semanticscholar.org/CorpusID:16965964).
27. Porter, T.; Duff, T. (1984). *Compositing digital images* (http://keithp.com/~keithp/porterduff/p253-porter.pdf) (PDF). Computer Graphics (Proceedings of SIGGRAPH 1984). Vol. 18. pp. 253–259. Archived (https://web.archive.org/web/20150216062332/http://keithp.com/~keithp/porterduff/p253-porter.pdf) (PDF) from the original on 2015-02-16.
28. Cook, R.L.; Porter, T.; Carpenter, L. (1984). *Distributed ray tracing* (http://www.cs.rutgers.edu/~nealen/teaching/cs428_fall09/readings/cook84.pdf) (PDF). Computer Graphics (Proceedings of SIGGRAPH 1984). Vol. 18. pp. 137–145.
29. Goral, C.; Torrance, K.E.; Greenberg, D.P.; Battaile, B. (1984). *Modeling the interaction of light between diffuse surfaces*. Computer Graphics (Proceedings of SIGGRAPH 1984). Vol. 18. pp. 213–222. CiteSeerX 10.1.1.112.356 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.112.356).
30. "Archived copy" (https://web.archive.org/web/20160304034742/http://cgfm2.emuviews.com/txt/s16tech.txt). Archived from the original (http://cgfm2.emuviews.com/txt/s16tech.txt) on 2016-03-04. Retrieved 2016-08-08.
31. Cohen, M.F.; Greenberg, D.P. (1985). *The hemi-cube: a radiosity solution for complex environments* (https://web.archive.org/web/20140424063155/http://arnetminer.org/dev.do?m=downloadpdf&url=http%3A%2F%2Farnetminer.org%2Fpdf%2FPDFFiles2%2F--g---g-Index1255026826706%2FThe%2520hemi-cube%2520%2520a%2520radiosity%2520solution%2520for%2520complex%2520environments1255058011060.pdf) (PDF). Computer Graphics (Proceedings of SIGGRAPH 1985). Vol. 19. pp. 31–40. doi:10.1145/325165.325171 (https://doi.org/10.1145%2F325165.325171). Archived from the original (http://www.arnetminer.org/dev.do?m=downloadpdf&url=http://arnetminer.org/pdf/PDFFiles2/--g---g-Index1255026826706/The%20hemi-cube%20%20a%20radiosity%20solution%20for%20complex%20environments1255058011060.pdf) (PDF) on 2014-04-24. Retrieved 2020-03-25.
32. Arvo, J. (1986). *Backward ray tracing*. SIGGRAPH 1986 Developments in Ray Tracing course notes. CiteSeerX 10.1.1.31.581 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.31.581).
33. Kajiya, J. (1986). *The rendering equation*. Computer Graphics (Proceedings of SIGGRAPH 1986). Vol. 20. pp. 143–150. CiteSeerX 10.1.1.63.1402 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.63.1402).
34. Cook, R.L.; Carpenter, L.; Catmull, E. (1987). *The Reyes image rendering architecture* (http://graphics.pixar.com/library/Reyes/paper.pdf) (PDF). Computer Graphics (Proceedings of SIGGRAPH 1987). Vol. 21. pp. 95–102. Archived (https://web.archive.org/web/20110715090353/http://graphics.pixar.com/library/Reyes/paper.pdf) (PDF) from the original on 2011-07-15.

35. "MAME | SRC/Mame/Drivers/Namcos21.c" (https://web.archive.org/web/20141003022000/http://mamedev.org/source/src/mame/drivers/namcos21.c.html). Archived from the original (http://mamedev.org/source/src/mame/drivers/namcos21.c.html) on 2014-10-03. Retrieved 2014-10-02.
36. Wu, Xiaolin (July 1991). *An efficient antialiasing technique* (http://portal.acm.org/citation.cfm?id=122734). *Computer Graphics*. Vol. 25. pp. 143–152. doi:10.1145/127719.122734 (https://doi.org/10.1145%2F127719.122734). ISBN 978-0-89791-436-9.
37. Wu, Xiaolin (1991). "Fast Anti-Aliased Circle Generation". In James Arvo (ed.). *Graphics Gems II*. San Francisco: Morgan Kaufmann. pp. 446–450. ISBN 978-0-12-064480-3.
38. Hanrahan, P.; Salzman, D.; Aupperle, L. (1991). *A rapid hierarchical radiosity algorithm*. Computer Graphics (Proceedings of SIGGRAPH 1991). Vol. 25. pp. 197–206. CiteSeerX 10.1.1.93.5694 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.93.5694).
39. "IGN Presents the History of SEGA" (http://ign.com/articles/2009/04/21/ign-presents-the-history-of-sega?page=8). *ign.com*. 21 April 2009. Archived (https://web.archive.org/web/20180316075945/http://www.ign.com/articles/2009/04/21/ign-presents-the-history-of-sega?page=8) from the original on 16 March 2018. Retrieved 7 May 2018.
40. "System 16 - Sega Model 2 Hardware (Sega)" (http://www.system16.com/hardware.php?id=713). *www.system16.com*. Archived (https://web.archive.org/web/20101221001009/http://system16.com/hardware.php?id=713) from the original on 21 December 2010. Retrieved 7 May 2018.
41. "System 16 - Namco Magic Edge Hornet Simulator Hardware (Namco)" (http://www.system16.com/hardware.php?id=832). *www.system16.com*. Archived (https://web.archive.org/web/20140912000953/http://www.system16.com/hardware.php?id=832) from the original on 12 September 2014. Retrieved 7 May 2018.
42. M. Oren and S.K. Nayar, "Generalization of Lambert's Reflectance Model (http://www1.cs.columbia.edu/CAVE/publications/pdfs/Oren_SIGGRAPH94.pdf) Archived (https://web.archive.org/web/20100215153120/http://www1.cs.columbia.edu/CAVE/publications/pdfs/Oren_SIGGRAPH94.pdf) 2010-02-15 at the Wayback Machine". SIGGRAPH. pp.239-246, Jul, 1994
43. Tumblin, J.; Rushmeier, H.E. (1993). "Tone reproduction for realistic computer generated images" (http://smartech.gatech.edu/bitstream/handle/1853/3686/92-31.pdf?sequence=1) (PDF). *IEEE Computer Graphics & Applications*. **13** (6): 42–48. doi:10.1109/38.252554 (https://doi.org/10.1109%2F38.252554). S2CID 6459836 (https://api.semanticscholar.org/CorpusID:6459836). Archived (https://web.archive.org/web/20111208231341/http://smartech.gatech.edu/bitstream/handle/1853/3686/92-31.pdf?sequence=1) (PDF) from the original on 2011-12-08.
44. Hanrahan, P.; Krueger, W. (1993). *Reflection from layered surfaces due to subsurface scattering*. Computer Graphics (Proceedings of SIGGRAPH 1993). Vol. 27. pp. 165–174. CiteSeerX 10.1.1.57.9761 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.57.9761).
45. Miller, Gavin (24 July 1994). "Efficient algorithms for local and global accessibility shading" (http://dl.acm.org/citation.cfm?id=192161.192244). *Proceedings of the 21st annual conference on Computer graphics and interactive techniques - SIGGRAPH '94*. ACM. pp. 319–326. doi:10.1145/192161.192244 (https://doi.org/10.1145%2F192161.192244). ISBN 978-0897916677. S2CID 15271113 (https://api.semanticscholar.org/CorpusID:15271113). Archived (https://web.archive.org/web/20211122155805/https://dl.acm.org/doi/10.1145/192161.192244) from the original on 22 November 2021. Retrieved 7 May 2018 – via dl.acm.org.

46. "Archived copy" (http://www.hotchips.org/wp-content/uploads/hc_archives/hc07/3_Tue/HC7.S5/HC7.5.1.pdf) (PDF). Archived (https://web.archive.org/web/20161011194640/http://www.hotchips.org/wp-content/uploads/hc_archives/hc07/3_Tue/HC7.S5/HC7.5.1.pdf) (PDF) from the original on 2016-10-11. Retrieved 2016-08-08.
47. Jensen, H.W.; Christensen, N.J. (1995). "Photon maps in bidirectional monte carlo ray tracing of complex objects". *Computers & Graphics*. **19** (2): 215–224. CiteSeerX 10.1.1.97.2724 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.97.2724). doi:10.1016/0097-8493(94)00145-o (https://doi.org/10.1016%2F0097-8493%2894%290145-o).
48. "System 16 - Sega Model 3 Step 1.0 Hardware (Sega)" (http://www.system16.com/hardware.php?id=717). *www.system16.com*. Archived (https://web.archive.org/web/20141006125510/http://www.system16.com/hardware.php?id=717) from the original on 6 October 2014. Retrieved 7 May 2018.
49. Veach, E.; Guibas, L. (1997). *Metropolis light transport*. Computer Graphics (Proceedings of SIGGRAPH 1997). Vol. 16. pp. 65–76. CiteSeerX 10.1.1.88.944 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.88.944).
50. Keller, A. (1997). *Instant Radiosity*. Computer Graphics (Proceedings of SIGGRAPH 1997). Vol. 24. pp. 49–56. CiteSeerX 10.1.1.15.240 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.15.240).
51. "Hardware Review: Neon 250 Specs & Features" (https://web.archive.org/web/20070807151257/http://www3.sharkyextreme.com/hardware/reviews/video/neon250/2.shtml). *sharkyextreme.com*. Archived from the original (http://www3.sharkyextreme.com/hardware/reviews/video/neon250/2.shtml) on 2007-08-07. Retrieved 2021-11-22.
52. Lewis, J. P.; Cordner, Matt; Fong, Nickson (1 July 2000). "Pose space deformation". *Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation*. ACM Press/Addison-Wesley Publishing Co. pp. 165–172. doi:10.1145/344779.344862 (https://doi.org/10.1145%2F344779.344862). ISBN 978-1581132083. S2CID 12672235 (https://api.semanticscholar.org/CorpusID:12672235) – via dl.acm.org.
53. Sloan, P.; Kautz, J.; Snyder, J. (2002). *Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low Frequency Lighting Environments* (https://web.archive.org/web/20110724151631/http://www.mpi-inf.mpg.de/~jnkautz/projects/prt/prtSIG02.pdf) (PDF). Computer Graphics (Proceedings of SIGGRAPH 2002). Vol. 29. pp. 527–536. Archived from the original (http://www.mpi-inf.mpg.de/~jnkautz/projects/prt/prtSIG02.pdf) (PDF) on 2011-07-24.

Further reading

- Akenine-Möller, Tomas; Haines, Eric (2004). *Real-time rendering* (https://archive.org/details/isbn_9781568811826) (2 ed.). Natick, Mass.: AK Peters. ISBN 978-1-56881-182-6.
- Blinn, Jim (1996). *Jim Blinn's corner : a trip down the graphics pipeline*. San Francisco, Calif.: Morgan Kaufmann Publishers. ISBN 978-1-55860-387-5.
- Cohen, Michael F.; Wallace, John R. (1998). *Radiosity and realistic image synthesis* (3 ed.). Boston, Mass. [u.a.]: Academic Press Professional. ISBN 978-0-12-178270-2.
- Philip Dutré; Bekaert, Philippe; Bala, Kavita (2003). *Advanced global illumination* ([Online-Ausg.] ed.). Natick, Mass.: A K Peters. ISBN 978-1-56881-177-2.
- Foley, James D.; Van Dam; Feiner; Hughes (1990). *Computer graphics : principles and practice* (https://archive.org/details/computergraphics00fole) (2 ed.). Reading, Mass.: Addison-Wesley. ISBN 978-0-201-12110-0.

- Andrew S. Glassner, ed. (1989). *An introduction to ray tracing* (3 ed.). London [u.a.]: Acad. Press. ISBN 978-0-12-286160-4.
- Glassner, Andrew S. (2004). *Principles of digital image synthesis* (2 ed.). San Francisco, Calif.: Kaufmann. ISBN 978-1-55860-276-2.
- Gooch, Bruce; Gooch, Amy (2001). *Non-photorealistic rendering*. Natick, Mass.: A K Peters. ISBN 978-1-56881-133-8.
- Jensen, Henrik Wann (2001). *Realistic image synthesis using photon mapping* ([Nachdr.] ed.). Natick, Mass.: AK Peters. ISBN 978-1-56881-147-5.
- Pharr, Matt; Humphreys, Greg (2004). *Physically based rendering from theory to implementation*. Amsterdam: Elsevier/Morgan Kaufmann. ISBN 978-0-12-553180-1.
- Shirley, Peter; Morley, R. Keith (2003). *Realistic ray tracing* (2 ed.). Natick, Mass.: AK Peters. ISBN 978-1-56881-198-7.
- Strothotte, Thomas; Schlechtweg, Stefan (2002). *Non-photorealistic computer graphics modeling, rendering, and animation* (2 ed.). San Francisco, CA: Morgan Kaufmann. ISBN 978-1-55860-787-3.
- Ward, Gregory J. (July 1994). "The RADIANCE Lighting Simulation and Rendering System" (http://radsite.lbl.gov/radiance/papers/sg94.1/). *Siggraph 94*: 459–72. doi:10.1145/192161.192286 (https://doi.org/10.1145%2F192161.192286). ISBN 0897916670. S2CID 2487835 (https://api.semanticscholar.org/CorpusID:2487835).

External links

- *GPU Rendering Magazine (http://www.gpurendering.com)*, online CGI magazine about advantages of GPU rendering
- SIGGRAPH (https://web.archive.org/web/19961221040900/http://siggraph.org/) – the ACMs special interest group in graphics – the largest academic and professional association and conference
- List of links to (recent, as of 2004) siggraph papers (and some others) on the web (https://web.archive.org/web/20040923075327/http://www.cs.brown.edu/~tor/)