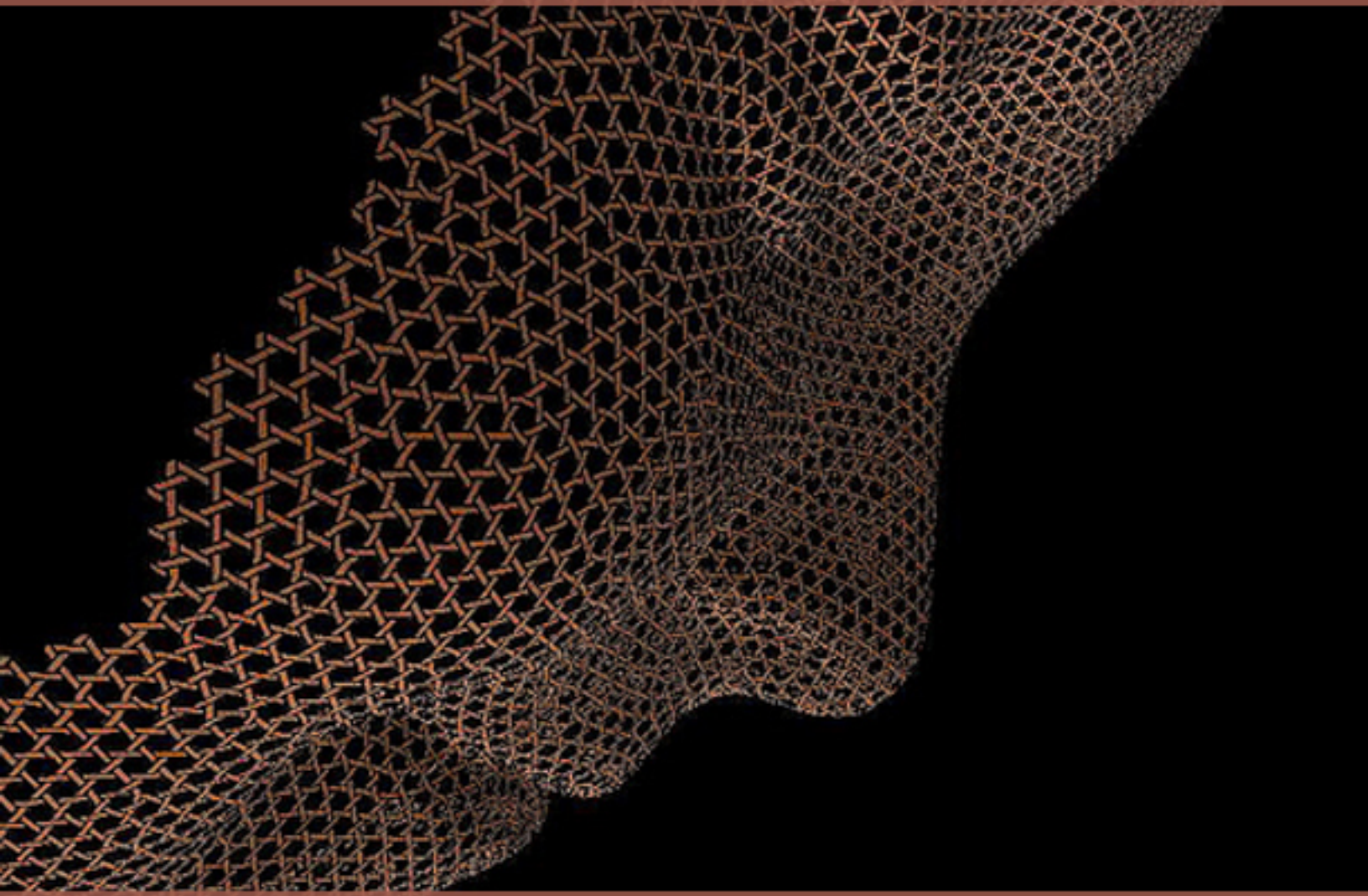
Fundamentals, Concepts and Applications of Computer Graphics



Lynell Soriano
Vincenza Yates

First Edition, 2012

ISBN 978-81-323-1273-4

Published by:
College Publishing House
4735/22 Prakashdeep Bldg,
Ansari Road, Darya Ganj,
Delhi - 110002
Email: info@wtbooks.com

Table of Contents

Chapter 1

Computer Graphics

The development of **computer graphics** has made computers easier to interact with, and better for understanding and interpreting many types of data. Developments in computer graphics have had a profound impact on many types of media and have revolutionized animation, movies and the video game industry.

Overview

The term computer graphics has been used in a broad sense to describe "almost everything on computers that is not text or sound". Typically, the term *computer graphics* refers to several different things:

- the representation and manipulation of image data by a computer
- the various technologies used to create and manipulate images
- the images so produced, and
- the sub-field of computer science which studies methods for digitally synthesizing and manipulating visual content.

Today, computers and computer-generated images touch many aspects of daily life. Computer imagery is found on television, in newspapers, for example in weather reports, or for example in all kinds of medical investigation and surgical procedures. A well-constructed graph can present complex statistics in a form that is easier to understand and interpret. In the media "such graphs are used to illustrate papers, reports, thesis", and other presentation material.

Many powerful tools have been developed to visualize data. Computer generated imagery can be categorized into several different types: 2D, 3D, 4D, 7D, and animated graphics. As technology has improved, 3D computer graphics have become more common, but 2D computer graphics are still widely used. Computer graphics has emerged as a sub-field of computer science which studies methods for digitally synthesizing and manipulating visual content. Over the past decade, other specialized fields have been developed like information visualization, and scientific visualization more concerned with "the visualization of three dimensional phenomena (architectural, meteorological, medical, biological, etc.), where the emphasis is on realistic renderings of volumes, surfaces, illumination sources, and so forth, perhaps with a dynamic (time) component".

Initial Development of Computer Graphics

The advance in computer graphics was to come from one MIT student, Ivan Sutherland. In 1961 Sutherland created another computer drawing program called Sketchpad. Using a light pen, Sketchpad allowed one to draw simple shapes on the computer screen, save them and even recall them later. The light pen itself had a small photoelectric cell in its tip. This cell emitted an electronic pulse whenever it was placed in front of a computer screen and the screen's electron gun fired directly at it. By simply timing the electronic pulse with the current location of the electron gun, it was easy to pinpoint exactly where the pen was on the screen at any given moment. Once that was determined, the computer could then draw a cursor at that location.

Sutherland seemed to find the perfect solution for many of the graphics problems he faced. Even today, many standards of computer graphics interfaces got their start with this early Sketchpad program. One example of this is in drawing constraints. If one wants to draw a square for example, s/he doesn't have to worry about drawing four lines perfectly to form the edges of the box. One can simply specify that s/he wants to draw a box, and then specify the location and size of the box. The software will then construct a perfect box, with the right dimensions and at the right location. Another example is that Sutherland's software modeled objects - not just a picture of objects. In other words, with a model of a car, one could change the size of the tires without affecting the rest of the car. It could stretch the body of the car without deforming the tires.

These early computer graphics were Vector graphics, composed of thin lines whereas modern day graphics are Raster based using pixels. The difference between vector graphics and raster graphics can be illustrated with a shipwrecked sailor. He creates an SOS sign in the sand by arranging rocks in the shape of the letters "SOS." He also has some brightly colored rope, with which he makes a second "SOS" sign by arranging the rope in the shapes of the letters. The rock SOS sign is similar to raster graphics. Every pixel has to be individually accounted for. The rope SOS sign is equivalent to vector graphics. The computer simply sets the starting point and ending point for the line and perhaps bend it a little between the two end points. The disadvantages to vector files are that they cannot represent continuous tone images and they are limited in the number of colors available. Raster formats on the other hand work well for continuous tone images and can reproduce as many colors as needed.

Also in 1961 another student at MIT, Steve Russell, created the first video game, Spacewar. Written for the DEC PDP-1, Spacewar was an instant success and copies started flowing to other PDP-1 owners and eventually even DEC got a copy. The engineers at DEC used it as a diagnostic program on every new PDP-1 before shipping it. The sales force picked up on this quickly enough and when installing new units, would run the world's first video game for their new customers.

E. E. Zajac, a scientist at Bell Telephone Laboratory (BTL), created a film called "Simulation of a two-giro gravity attitude control system" in 1963. In this computer generated film, Zajac showed how the attitude of a satellite could be altered as it orbits

the Earth. He created the animation on an IBM 7090 mainframe computer. Also at BTL, Ken Knowlton, Frank Sindon and Michael Noll started working in the computer graphics field. Sindon created a film called Force, Mass and Motion illustrating Newton's laws of motion in operation. Around the same time, other scientists were creating computer graphics to illustrate their research. At Lawrence Radiation Laboratory, Nelson Max created the films, "Flow of a Viscous Fluid" and "Propagation of Shock Waves in a Solid Form." Boeing Aircraft created a film called "Vibration of an Aircraft."

It wasn't long before major corporations started taking an interest in computer graphics. TRW, Lockheed-Georgia, General Electric and Sperry Rand are among the many companies that were getting started in computer graphics by the mid 1960's. IBM was quick to respond to this interest by releasing the IBM 2250 graphics terminal, the first commercially available graphics computer.

Ralph Baer, a supervising engineer at Sanders Associates, came up with a home video game in 1966 that was later licensed to Magnavox and called the Odyssey. While very simplistic, and requiring fairly inexpensive electronic parts, it allowed the player to move points of light around on a screen. It was the first consumer computer graphics product.

Also in 1966, Sutherland at MIT invented the first computer controlled head-mounted display (HMD). Called the Sword of Damocles because of the hardware required for support, it displayed two separate wireframe images, one for each eye. This allowed the viewer to see the computer scene in stereoscopic 3D. After receiving his Ph.D. from MIT, Sutherland became Director of Information Processing at ARPA (Advanced Research Projects Agency), and later became a professor at Harvard.

Dave Evans was director of engineering at Bendix Corporation's computer division from 1953 to 1962, after which he worked for the next five years as a visiting professor at Berkeley. There he continued his interest in computers and how they interfaced with people. In 1968 the University of Utah recruited Evans to form a computer science program, and computer graphics quickly became his primary interest. This new department would become the world's primary research center for computer graphics.

In 1967 Sutherland was recruited by Evans to join the computer science program at the University of Utah. There he perfected his HMD. Twenty years later, NASA would re-discover his techniques in their virtual reality research. At Utah, Sutherland and Evans were highly sought after consultants by large companies but they were frustrated at the lack of graphics hardware available at the time so they started formulating a plan to start their own company.

A student by the name of Edwin Catmull started at the University of Utah in 1970 and signed up for Sutherland's computer graphics class. Catmull had just come from The Boeing Company and had been working on his degree in physics. Growing up on Disney, Catmull loved animation yet quickly discovered that he didn't have the talent for drawing. Now Catmull (along with many others) saw computers as the natural progression of animation and they wanted to be part of the revolution. The first animation that Catmull

saw was his own. He created an animation of his hand opening and closing. It became one of his goals to produce a feature length motion picture using computer graphics. In the same class, Fred Parke created an animation of his wife's face. Because of Evan's and Sutherland's presence, UU was gaining quite a reputation as the place to be for computer graphics research so Catmull went there to learn 3D animation.

As the UU computer graphics laboratory was attracting people from all over, John Warnock was one of those early pioneers; he would later found Adobe Systems and create a revolution in the publishing world with his PostScript page description language. Tom Stockham led the image processing group at UU which worked closely with the computer graphics lab. Jim Clark was also there; he would later found Silicon Graphics, Inc.

The first major advance in 3D computer graphics was created at UU by these early pioneers, the hidden-surface algorithm. In order to draw a representation of a 3D object on the screen, the computer must determine which surfaces are "behind" the object from the viewer's perspective, and thus should be "hidden" when the computer creates (or renders) the image.

Image types

2D computer graphics

Raster graphic sprites (left) and masks (right)

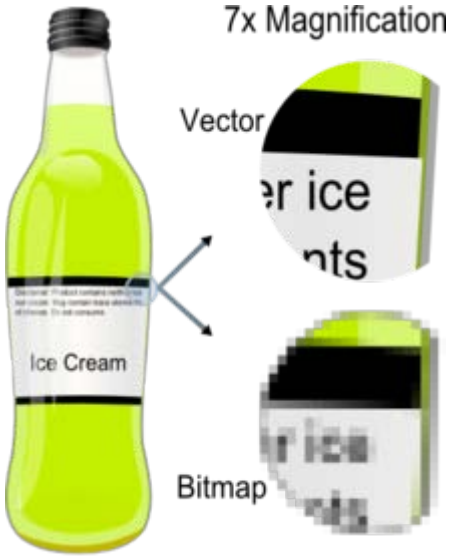
2D computer graphics are the computer-based generation of digital images—mostly from two-dimensional models, such as 2D geometric models, text, and digital images, and by techniques specific to them.

2D computer graphics are mainly used in applications that were originally developed upon traditional printing and drawing technologies, such as typography, cartography, technical drawing, advertising, etc.. In those applications, the two-dimensional image is not just a representation of a real-world object, but an independent artifact with added semantic value; two-dimensional models are therefore preferred, because they give more direct control of the image than 3D computer graphics, whose approach is more akin to photography than to typography.

Pixel art

Pixel art is a form of digital art, created through the use of raster graphics software, where images are edited on the pixel level. Graphics in most old (or relatively limited) computer and video games, graphing calculator games, and many mobile phone games are mostly pixel art.

Vector graphics



Example showing effect of vector graphics versus raster (bitmap) graphics

Vector graphics formats are complementary to raster graphics, which is the representation of images as an array of pixels, as it is typically used for the representation of photographic images. There are instances when working with vector tools and formats is best practice, and instances when working with raster tools and formats is best practice. There are times when both formats come together. An understanding of the advantages and limitations of each technology and the relationship between them is most likely to result in efficient and effective use of tools.

3D computer graphics

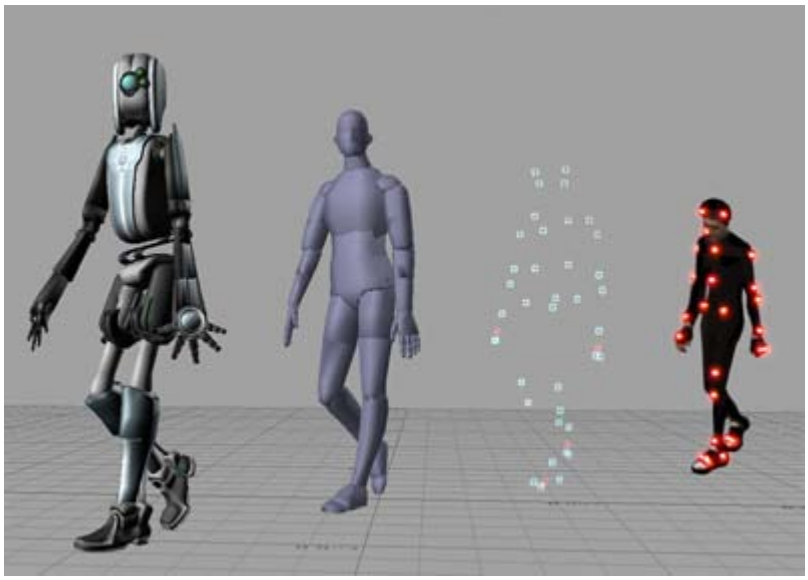
3D computer graphics in contrast to 2D computer graphics are graphics that use a three-dimensional representation of geometric data that is stored in the computer for the purposes of performing calculations and rendering 2D images. Such images may be for later display or for real-time viewing.

Despite these differences, 3D computer graphics rely on many of the same algorithms as 2D computer vector graphics in the wire frame model and 2D computer raster graphics in the final rendered display. In computer graphics software, the distinction between 2D and

3D is occasionally blurred; 2D applications may use 3D techniques to achieve effects such as lighting, and primarily 3D may use 2D rendering techniques.

3D computer graphics are often referred to as 3D models. Apart from the rendered graphic, the model is contained within the graphical data file. However, there are differences. A 3D model is the mathematical representation of any three-dimensional object. A model is not technically a graphic until it is visually displayed. Due to 3D printing, 3D models are not confined to virtual space. A model can be displayed visually as a two-dimensional image through a process called *3D rendering,* or used in non-graphical computer simulations and calculations. There are some 3D computer graphics software for users to create 3D images.

Computer animation



An example of Computer animation produced using Motion capture

Computer animation is the art of creating moving images via the use of computers. It is a subfield of computer graphics and animation. Increasingly it is created by means of 3D computer graphics, though 2D computer graphics are still widely used for stylistic, low bandwidth, and faster real-time rendering needs. Sometimes the target of the animation is the computer itself, but sometimes the target is another medium, such as film. It is also referred to as CGI (Computer-generated imagery or computer-generated imaging), especially when used in films.

Virtual entities may contain and be controlled by assorted attributes, such as transform values (location, orientation, and scale) stored in an object's transformation matrix. Animation is the change of an attribute over time. Multiple methods of achieving animation exist; the rudimentary form is based on the creation and editing of keyframes, each storing a value at a given time, per attribute to be animated. The 2D/3D graphics software will interpolate between keyframes, creating an editable curve of a value mapped over time, resulting in animation. Other methods of animation include procedural

and expression-based techniques: the former consolidates related elements of animated entities into sets of attributes, useful for creating particle effects and crowd simulations; the latter allows an evaluated result returned from a user-defined logical expression, coupled with mathematics, to automate animation in a predictable way (convenient for controlling bone behavior beyond what a hierarchy offers in skeletal system set up).

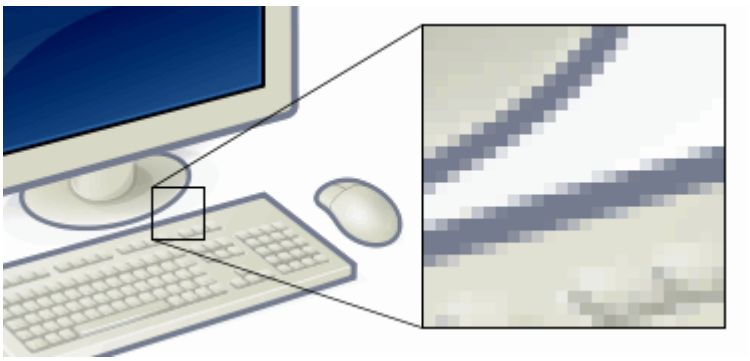
To create the illusion of movement, an image is displayed on the computer screen then quickly replaced by a new image that is similar to the previous image, but shifted slightly. This technique is identical to the illusion of movement in television and motion pictures.

Concepts and principles

Images are typically produced by optical devices;such as cameras, mirrors, lenses, telescopes, microscopes, etc. and natural objects and phenomena, such as the human eye or water surfaces.

A digital image is a representation of a two-dimensional image in binary format as a sequence of ones and zeros. Digital images include both vector images and raster images, but raster images are more commonly used.

Pixel



In the enlarged portion of the image individual pixels are rendered as squares and can be easily seen.

In digital imaging, a pixel (or picture element) is a single point in a raster image. Pixels are normally arranged in a regular 2-dimensional grid, and are often represented using dots or squares. Each pixel is a sample of an original image, where more samples typically provide a more accurate representation of the original. The intensity of each pixel is variable; in color systems, each pixel has typically three components such as red, green, and blue.

Graphics

Graphics are visual presentations on some surface, such as a wall, canvas, computer screen, paper, or stone to brand, inform, illustrate, or entertain. Examples are photographs, drawings, line art, graphs, diagrams, typography, numbers, symbols, geometric designs, maps, engineering drawings, or other images. Graphics often combine text, illustration, and color. Graphic design may consist of the deliberate selection, creation, or arrangement of typography alone, as in a brochure, flier, poster, web site, or book without any other element. Clarity or effective communication may be the objective, association with other cultural elements may be sought, or merely, the creation of a distinctive style.

Arambilet: *Dots on the I's*, D-ART 2009 Online Digital Art Gallery, exhibited at IV09 and CG09 computer Graphics conferences, at Pompeu Fabra University, Barcelona; Tianjin University, China; Permanent Exhibition at the London South Bank University

Rendering

Rendering is the process of generating an image from a model (or models in what collectively could be called a *scene* file), by means of computer programs. A scene file contains objects in a strictly defined language or data structure; it would contain geometry, viewpoint, texture, lighting, and shading information as a description of 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 rendering program is usually built into the computer graphics software, though others are available as plug-ins or entirely separate programs. The term "rendering" may be by analogy with an "artist's rendering" of a scene. Though the technical details of rendering methods vary, the general challenges to overcome in producing a 2D image from a 3D representation stored in a scene file are outlined as the graphics pipeline along a rendering device, such as a GPU. A GPU is a purpose-built device able to assist a CPU in performing complex rendering calculations. If a scene is to look relatively realistic and predictable under virtual lighting, the rendering software should solve the rendering equation. The rendering equation doesn't account for all lighting phenomena, but is a general lighting

model for computer-generated imagery. 'Rendering' is also used to describe the process of calculating effects in a video editing file to produce final video output.

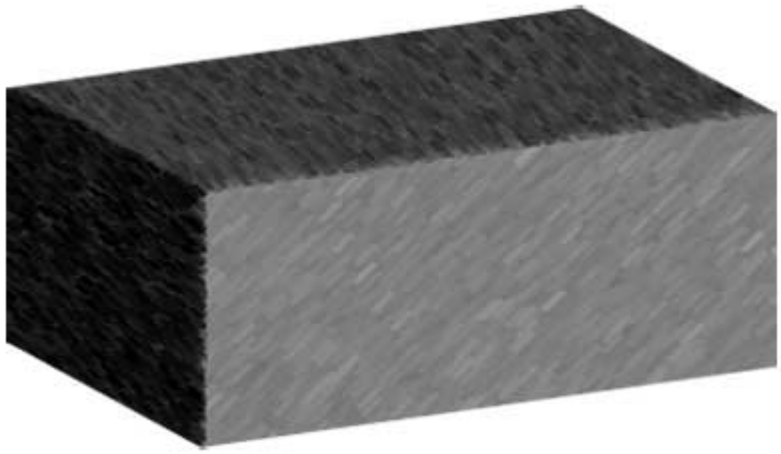
3D projection

3D projection is a method of mapping three dimensional points to a two dimensional plane. As most current methods for displaying graphical data are based on planar two dimensional media, the use of this type of projection is widespread, especially in computer graphics, engineering and drafting.

Ray tracing

Ray tracing is a technique for generating an image by tracing the path of light through pixels in an image plane. The technique is capable of producing a very high degree of photorealism; usually higher than that of typical scanline rendering methods, but at a greater computational cost.

Shading



Example of shading.

Shading refers to depicting depth in 3D models or illustrations by varying levels of darkness. It is a process used in drawing for depicting levels of darkness on paper by applying media more densely or with a darker shade for darker areas, and less densely or with a lighter shade for lighter areas. There are various techniques of shading including cross hatching where perpendicular lines of varying closeness are drawn in a grid pattern to shade an area. The closer the lines are together, the darker the area appears. Likewise, the farther apart the lines are, the lighter the area appears. The term has been recently generalized to mean that shaders are applied.

Texture mapping

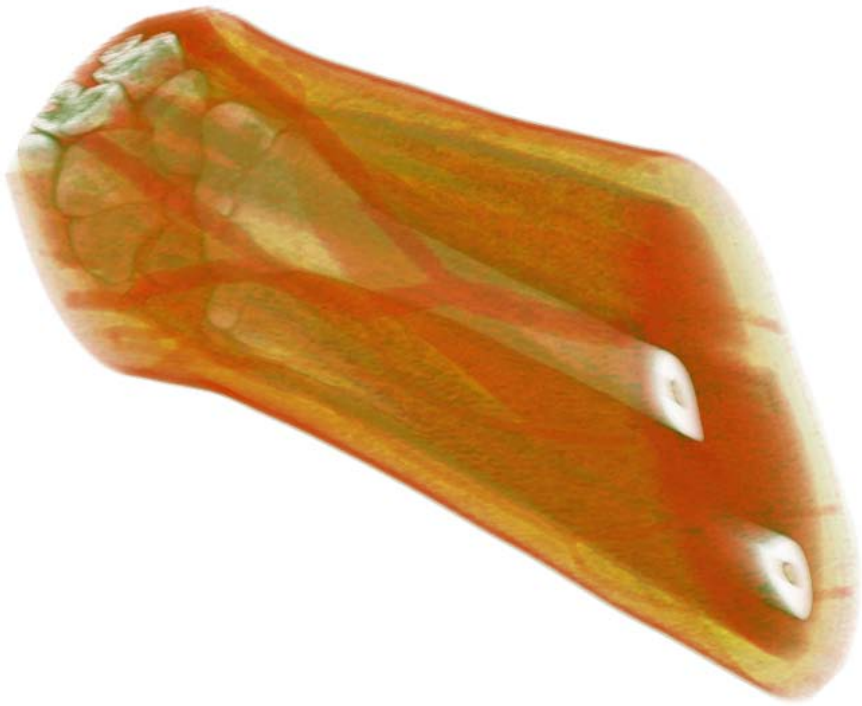
Texture mapping is a method for adding detail, surface texture, or colour to a computer-generated graphic or 3D model. Its application to 3D graphics was pioneered by Dr Edwin Catmull in 1974. A texture map is applied (mapped) to the surface of a shape, or polygon. This process is akin to applying patterned paper to a plain white box. Multitexturing is the use of more than one texture at a time on a polygon. Procedural textures (created from adjusting parameters of an underlying algorithm that produces an output texture), and bitmap textures (created in an image editing application) are, generally speaking, common

methods of implementing texture definition from a 3D animation program, while intended placement of textures onto a model's surface often requires a technique known as UV mapping.

Anti-aliasing

Rendering resolution-independent entities (such as 3D models) for viewing on a raster (pixel-based) device such as a LCD display or CRT television inevitably causes aliasing artifacts mostly along geometric edges and the boundaries of texture details; these artifacts are informally called "jaggies". Anti-aliasing methods rectify such problems, resulting in imagery more pleasing to the viewer, but can be somewhat computationally expensive. Various anti-aliasing algorithms (such as supersampling) are able to be employed, then customized for the most efficient rendering performance versus quality of the resultant imagery; a graphics artist should consider this trade-off if anti-aliasing methods are to be used. A pre-anti-aliased bitmap texture being displayed on a screen (or screen location) at a resolution different than the resolution of the texture itself (such as a textured model in the distance from the virtual camera) will exhibit aliasing artifacts, while any procedurally-defined texture will always show aliasing artifacts as they are resolution-independent; techniques such as mipmapping and texture filtering help to solve texture-related aliasing problems.

Volume rendering



Volume rendered CT scan of a forearm with different colour schemes for muscle, fat, bone, and blood.

Volume rendering is a technique used to display a 2D projection of a 3D discretely sampled data set. A typical 3D data set is a group of 2D slice images acquired by a CT or MRI scanner.

Usually these are acquired in a regular pattern (e.g., one slice every millimeter) and usually have a regular number of image pixels in a regular pattern. This is an example of a regular volumetric grid, with each volume element, or voxel represented by a single value that is obtained by sampling the immediate area surrounding the voxel.

3D modeling

3D modeling is the process of developing a mathematical, wireframe representation of any three-dimensional object, called a "3D model", via specialized software. Models may be created automatically or manually; the manual modeling process of preparing geometric data for 3D computer graphics is similar to plastic arts such as sculpting. 3D models may be created using multiple approaches: use of NURBS curves to generate accurate and smooth surface patches, polygonal mesh modeling (manipulation of faceted geometry), or polygonal mesh subdivision (advanced tessellation of polygons, resulting in smooth surfaces similar to NURBS models). A 3D model can be displayed as a two-dimensional image through a process called *3D rendering*, used in a computer simulation of physical phenomena, or animated directly for other purposes. The model can also be physically created using 3D Printing devices.

Pioneers in graphic design

Charles Csuri

: Charles Csuri is a pioneer in computer animation and digital fine art and created the first computer art in 1964. Csuri was recognized by *Smithsonian* as the father of digital art and computer animation, and as a pioneer of computer animation by the Museum of Modern Art (MoMA) and Association for Computing Machinery-SIGGRAPH.

Donald P. Greenberg

: Donald P. Greenberg is a leading innovator in computer graphics. Greenberg has authored hundreds of articles and served as a teacher and mentor to many prominent computer graphic artists, animators, and researchers such as Robert L. Cook, Marc Levoy, and Wayne Lytle. Many of his former students have won Academy Awards for technical achievements and several have won the SIGGRAPH Achievement Award. Greenberg was the founding director of the NSF Center for Computer Graphics and Scientific Visualization.

A. Michael Noll

: Noll was one of the first researchers to use a digital computer to create artistic patterns and to formalize the use of random processes in the creation of visual arts. He began creating digital computer art in 1962, making him one of the earliest digital computer artists. In 1965, Noll along with Frieder Nake and Georg Nees were the first to publicly exhibit their computer art. During April 1965, the Howard Wise Gallery exhibited Noll's computer art along with random-dot patterns by Bela Julesz.

A modern render of the Utah teapot, an iconic model in 3D computer graphics created by Martin Newell, 1975.
Other pioneers

- Jim Blinn
- Arambilet
- Benoît B. Mandelbrot
- Henri Gouraud
- Bui Tuong Phong
- Pierre Bézier
- Paul de Casteljau
- Daniel J. Sandin
- Alvy Ray Smith
- Ton Roosendaal
- Ivan Sutherland
- Steve Russell

The study of computer graphics

The study of computer graphics is a sub-field of computer science which studies methods for digitally synthesizing and manipulating visual content. Although the term often refers to three-dimensional computer graphics, it also encompasses two-dimensional graphics and image processing.

As an academic discipline, computer graphics studies the manipulation of visual and geometric information using computational techniques. It focuses on the *mathematical* and *computational* foundations of image generation and processing rather than purely aesthetic issues. Computer graphics is often differentiated from the field of visualization, although the two fields have many similarities.

Chapter 2

2D Computer Graphics and Pixel Art

2D computer graphics

2D computer graphics is the computer-based generation of digital images—mostly from two-dimensional models (such as 2D geometric models, text, and digital images) and by techniques specific to them. The word may stand for the branch of computer science that comprises such techniques, or for the models themselves.

Raster graphic sprites (left) and masks (right)

2D computer graphics are mainly used in applications that were originally developed upon traditional printing and drawing technologies, such as typography, cartography, technical drawing, advertising, etc. In those applications, the two-dimensional image is not just a representation of a real-world object, but an independent artifact with added semantic value; two-dimensional models are therefore preferred, because they give more direct control of the image than 3D computer graphics (whose approach is more akin to photography than to typography).

In many domains, such as desktop publishing, engineering, and business, a description of a document based on 2D computer graphics techniques can be much smaller than the corresponding digital image—often by a factor of 1/1000 or more. This representation is also more flexible since it can be rendered at different resolutions to suit different output devices. For these reasons, documents and illustrations are often stored or transmitted as 2D graphic files.

2D computer graphics started in the 1950s, based on vector graphics devices. These were largely supplanted by raster-based devices in the following decades. The PostScript language and the X Window System protocol were landmark developments in the field.

2D graphics techniques

2D graphics models may combine geometric models (also called vector graphics), digital images (also called raster graphics), text to be typeset (defined by content, font style and size, color, position, and orientation), mathematical functions and equations, and more. These components can be modified and manipulated by two-dimensional geometric transformations such as translation, rotation, scaling. In object-oriented graphics, the image is described indirectly by an object endowed with a self-rendering method—a procedure which assigns colors to the image pixels by an arbitrary algorithm. Complex models can be built by combining simpler objects, in the paradigms of object-oriented programming.

Direct painting

A convenient way to create a complex image is to start with a blank "canvas" raster map (an array of pixels, also known as a bitmap) filled with some uniform background color and then "draw", "paint" or "paste" simple patches of color onto it, in an appropriate order. In particular, the canvas may be the frame buffer for a computer display.

Some programs will set the pixel colors directly, but most will rely on some 2D graphics library and/or the machine's graphics card, which usually implement the following operations:

- paste a given image at a specified offset onto the canvas;
- write a string of characters with a specified font, at a given position and angle;
- paint a simple geometric shape, such as a triangle defined by three corners, or a circle with given center and radius;
- draw a line segment, arc, or simple curve with a virtual pen of given width.

Extended color models

Text, shapes and lines are rendered with a client-specified color. Many libraries and cards provide color gradients, which are handy for the generation of smoothly-varying backgrounds, shadow effects, etc. The pixel colors can also be taken from a texture, e.g. a digital image (thus emulating rub-on screentones and the fabled "checker paint" which used to be available only in cartoons).

Painting a pixel with a given color usually replaces its previous color. However, many systems support painting with transparent and translucent colors, which only modify the previous pixel values. The two colors may also be combined in fancier ways, e.g. by computing their bitwise exclusive or. This technique is known as inverting color or color inversion, and is often used in graphical user interfaces for highlighting, rubber-band

drawing, and other volatile painting—since re-painting the same shapes with the same color will restore the original pixel values.

Layers

The models used in 2D computer graphics usually do not provide for three-dimensional shapes, or three-dimensional optical phenomena such as lighting, shadows, reflection, refraction, etc.. However, they usually can model multiple *layers* (conceptually of ink, paper, or film; opaque, translucent, or transparent—stacked in a specific order. The ordering is usually defined by a single number (the layer's *depth*, or distance from the viewer).

Layered models are sometimes called *2½-D computer graphics*. They make it possible to mimic traditional drafting and printing techniques based on film and paper, such as cutting and pasting; and allow the user to edit any layer without affecting the others. For these reasons, they are used in most graphics editors. Layered models also allow better anti-aliasing of complex drawings and provide a sound model for certain techniques such as mitered joints and the even-odd rule.

Layered models are also used to allow the user to suppress unwanted information when viewing or printing a document, e.g. roads and/or railways from a map, certain process layers from an integrated circuit diagram, or hand annotations from a business letter.

In a layer-based model, the target image is produced by "painting" or "pasting" each layer, in order of decreasing depth, on the virtual canvas. Conceptually, each layer is first rendered on its own, yielding a digital image with the desired resolution which is then painted over the canvas, pixel by pixel. Fully transparent parts of a layer need not be rendered, of course. The rendering and painting may be done in parallel, i.e. each layer pixel may be painted on the canvas as soon as it is produced by the rendering procedure.

Layers that consist of complex geometric objects (such as text or polylines) may be broken down into simpler elements (characters or line segments, respectively), which are then painted as separate layers, in some order. However, this solution may create undesirable aliasing artifacts wherever two elements overlap the same pixel.

2D graphics hardware

Modern computer graphics card displays almost overwhelmingly use raster techniques, dividing the screen into a rectangular grid of pixels, due to the relatively low cost of raster-based video hardware as compared with vector graphic hardware. Most graphic hardware has internal support for blitting operations and sprite drawing. A co-processor dedicated to blitting is known as a *Blitter chip*.

Classic 2D graphics chips of the late 1970s and early 80s, used in the 8-bit video game consoles and home computers, include:

- Atari's ANTIC (actually a 2D GPU), TIA, CTIA, and GTIA
- Commodore/MOS Technology's VIC and VIC-II

2D graphics software

Many graphical user interfaces (GUIs), including Mac OS, Microsoft Windows, or the X Window System, are primarily based on 2D graphical concepts. Such software provides a visual environment for interacting with the computer, and commonly includes some form of window manager to aid the user in conceptually distinguishing between different applications. The user interface within individual software applications is typically 2D in nature as well, due in part to the fact that most common input devices, such as the mouse, are constrained to two dimensions of movement.

2D graphics are very important in the control peripherals such as printers, plotters, sheet cutting machines, etc.. They were also used in most early video and computer games; and are still used for card and board games such as solitaire, chess, mahjongg, etc..

2D graphics editors or *drawing programs* are application-level software for the creation of images, diagrams and illustrations by direct manipulation (through the mouse, graphics tablet, or similar device) of 2D computer graphics primitives. These editors generally provide geometric primitives as well as digital images; and some even support procedural models. The illustration is usually represented internally as a layered model, often with a hierarchical structure to make editing more convenient. These editors generally output graphics files where the layers and primitives are separately preserved in their original form. MacDraw, introduced in 1984 with the Macintosh line of computers, was an early example of this class; recent examples are the commercial products Adobe Illustrator and CorelDRAW, and the free editors such as xfig or Inkscape. There are also many 2D graphics editors specialized for certain types of drawings such as electrical, electronic and VLSI diagrams, topographic maps, computer fonts, etc.

Image editors are specialized for the manipulation of digital images, mainly by means of free-hand drawing/painting and signal processing operations. They typically use a direct-painting paradigm, where the user controls virtual pens, brushes, and other free-hand artistic instruments to apply paint to a virtual canvas. Some image editors support a multiple-layer model; however, in order to support signal-processing operations like blurring each layer is normally represented as a digital image. Therefore, any geometric primitives that are provided by the editor are immediately converted to pixels and painted onto the canvas. The name *raster graphics editor* is sometimes used to contrast this approach to that of general editors which also handle *vector graphics*. One of the first popular image editors was Apple's MacPaint, companion to MacDraw. Modern examples are the free GIMP editor, and the commercial products Photoshop and Paint Shop Pro. This class too includes many specialized editors — for medicine, remote sensing, digital photography, etc.

Developmental animation

With the resurgence of 2D animation and its booming popularity, free and proprietary software packages have become widely available for amateurs and professional animators. However, the principal issue with 2D animation is labor requirements. With advanced software like RETAS and Adobe After Effects, coloring and compositing can be easily done with significantly less time.

Additional software is being developed to aid and speed up the process of digital 2D animation, specifically in the area of automatic coloring and in-betweening.

Pixel art

Pixel art is a form of digital art, created through the use of raster graphics software, where images are edited on the pixel level. Graphics in most old (or relatively limited) computer and video games, graphing calculator games, and many mobile phone games are mostly pixel art.

History

The term *pixel art* was first published by Adele Goldberg and Robert Flegal of Xerox Palo Alto Research Center in 1982. The concept, however, goes back about 10 years before that, for example in Richard Shoup's SuperPaint system in 1972, also at Xerox PARC.

Some traditional art forms, such as counted-thread embroidery (including cross-stitch) and some kinds of mosaic and beadwork, are very similar to pixel art. These art forms construct pictures out of small colored units similar to the pixels of modern digital computing. A similar concept on a much bigger scale can be seen in the mass games.

Definition

Image filters (such as blurring or alpha-blending) or tools with automatic anti-aliasing are considered not valid tools for pixel art, as such tools calculate new pixel values automatically, contrasting with the precise manual arrangement of pixels associated with pixel art.

Techniques

Drawings usually start with what is called the line art, which is the basic line that defines the character, building or anything else the artist is intending to draw. Linearts are usually

traced over scanned drawings and are often shared among other pixel artists. Other techniques, some resembling painting, also exist.

The limited palette often implemented in pixel art usually promotes dithering to achieve different shades and colors, but due to the nature of this form of art this is done completely by hand. *Hand-made* anti-aliasing is also used.

Here are a few parts of the above image of "The Gunk" in detail, depicting a few of the techniques involved:

1. The basic form of dithering, using two colors in a 2×2 checkerboard pattern. Changing the density of each color will lead to different subtones.

2. Stylized dithering with 2×2 pixel squares randomly scattered can produce interesting textures. Small circles are also frequent.

3. Anti-aliasing can be done, by hand, to smooth curves and transitions. Some artists only do this internally, to keep crisp outlines that can go over any background. The PNG alpha channel can be used to create external anti-aliasing for any background.

Saving and compression

Pixel art is preferably stored in a file format utilizing lossless data compression, such as run-length encoding or an indexed color palette. GIF and PNG are two file formats commonly used for storing pixel art. The JPEG format is avoided because its lossy compression algorithm is designed for smooth continuous-tone images and introduces visible artifacts in the presence of dithering.

GIF file (318 bytes)

PNG file (254 bytes)

JPEG file (706 bytes)

Magnified JPEG to show artifacts

Categories

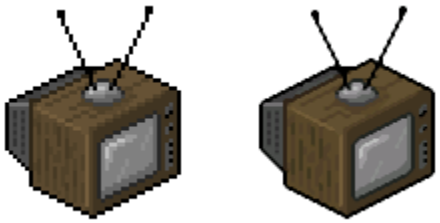
Isometric

Non-isometric

Pixel art is commonly divided in two subcategories: isometric and non-isometric. The isometric kind is drawn in a near-isometric dimetric projection. This is commonly seen in games to provide a three-dimensional view without using any real three-dimensional processing. Technically, an isometric angle would be of 30 degrees from the horizontal, but this is avoided since the pixels created by a line drawing algorithm would not follow a neat pattern. To fix this, lines with a 1:2 pixel ratio are picked, leading to an angle of about 26.6 degrees (arctan 0.5).

Non-isometric pixel art is any pixel art that does not fall in the isometric category, such as views from the top, side, front, bottom or perspective views. These are also called Planometric views.

Scaling



2x zoom interpolated using the 2xSaI algorithm

When pixel art is displayed at a higher resolution than the source image, it is often scaled using the nearest neighbor interpolation algorithm. This avoids blurring caused by other algorithms, such as bilinear and bicubic interpolation—which interpolate between adjacent pixels and work best on continuous tones, but not sharp edges or lines. Nearest-neighbor interpolation preserves these sharp edges, but it makes diagonal lines and curves look blocky, an effect called pixelation. Thus, hybrid algorithms have been devised to interpolate between continuous tones while preserving the sharpness of lines in the piece; such attempts include the 2xSaI and Super Eagle algorithms.

Uses

Pixel art was very often used in older computer and video console games. With the increasing use of 3D graphics in games, pixel art lost some of its use. Despite that, this is still a very active professional/amateur area, since mobile phones and other portable devices still have low resolution and then require a skillful use of space and memory. Sometimes pixel art is used for advertising too. One such company that uses pixel art to advertise is Bell. The group eboy specializes in pixel graphics for advertising and has been featured in magazines such as *Wired*, *Popular Science*, and *Fortune 500*.

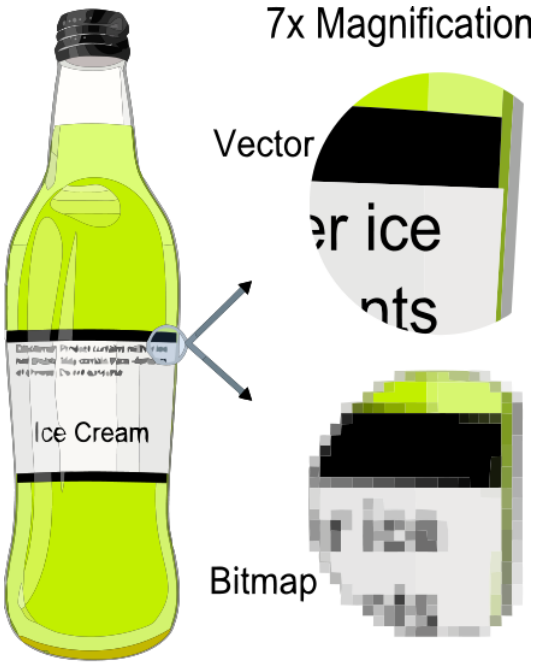
Icons for operating systems with limited graphics abilities are also pixel art. The limited number of colors and resolution presents a challenge when attempting to convey complicated concepts and ideas in an efficient way. On the Microsoft Windows desktop icons are raster images of various sizes, the smaller of which are not necessarily scaled from the larger ones and could be considered pixel art. On the GNOME and KDE desktops, icons are represented primarily by SVG images, but with hand-optimized, pixel art PNGs for smaller sizes such as 16x16 and 24x24. Another use of pixel art on modern desktop computers is favicons.

Modern pixel art has been seen as a reaction to the 3D graphics industry by amateur game/graphic hobbyists. Many retro enthusiasts often choose to mimic the style of the past. Some view the pixel art revival as restoring the golden age of second and third generation consoles, where it is argued graphics were more aesthetically pleasing. Pixel art still remains popular and has been used in the virtual worlds Citypixel and Habbo as well as among hand-held devices such as the Nintendo DS and Cellphones.

Chapter 3

Vector Graphics and 3D Computer Graphics

Vector graphics



Example showing effect of vector graphics versus raster graphics. The original vector-based illustration is at the left. The upper-right image illustrates magnification of 7x as a vector image. The lower-right image illustrates the same magnification as a bitmap image. Raster images are based on pixels and thus scale with loss of clarity, while vector-based images can be scaled indefinitely without degrading quality.

Vector graphics is the use of geometrical primitives such as points, lines, curves, and shapes or polygon(s), which are all based on mathematical equations, to represent images in computer graphics.

Vector graphics formats are complementary to raster graphics, which is the representation of images as an array of pixels, as it is typically used for the representation of photographic images. There are instances when working with vector tools and formats is the best practice, and instances when working with raster tools and formats is the best practice. There are times when both formats come together. An understanding of the advantages and limitations of each technology and the relationship between them is most likely to result in efficient and effective use of tools.

Overview

Computer displays are made up from grids of small rectangular cells called pixels. The picture is built up from these cells. The smaller and closer the cells are together, the better the quality of the image, but the bigger the file needed to store the data. If the number of pixels is kept constant, the size of each pixel will grow and the image becomes grainy (pixellated) when magnified, as the resolution of the eye enables it to pick out individual pixels.

Vector graphics files store the lines, shapes and colours that make up an image as mathematical formulae. A vector graphics program uses these mathematical formulae to construct the screen image, building the best quality image possible, given the screen resolution. The mathematical formulae determine where the dots that make up the image should be placed for the best results when displaying the image. Since these formulae can produce an image scalable to any size and detail, the quality of the image is limited only by the resolution of the display, and the file size of vector data generating the image stays the same. Printing the image to paper will usually give a sharper, higher resolution output than printing it to the screen but can use exactly the same vector data file.

Editing vector graphics

Vector graphic drawing software is used for creating and editing vector graphics. The image can be changed by editing screen objects which are then saved as modifications to the mathematical formulae. Mathematical operators in the software can be used to stretch, twist, and colour component objects in the picture or the whole picture, and these tools are presented to the user intuitively through the graphical user interface of the computer. It is possible to save the screen image produced as a bitmap/raster file or generate a bitmap of any resolution from the vector file for use on any device.

The size of the file generated will depend on the resolution required, but the size of the vector file generating the bitmap/raster file will always remain the same. Thus, it is easy to convert from a vector file to a range of bitmap/raster file formats but it is much more difficult to go in the opposite direction, especially if subsequent editing of the vector picture is required. It might be an advantage to save an image created from a vector source file as a bitmap/raster format, because different systems have different (and incompatible) vector formats, and some might not support vector graphics at all. However, once a file is converted from the vector format, it is likely to be bigger, and it loses the advantage of scalability without loss of resolution. It will also no longer be

possible to edit individual parts of the image as discrete objects. The file size of vector graphic depends on the number of graphic elements it contains.

Vector formats are not always appropriate in graphics work. For example, devices such as cameras and scanners produce raster graphics that are impractical to convert into vectors, and so for this type of work, the editor will operate on the pixels rather than on drawing objects defined by mathematical formulae. Comprehensive graphics tools will combine images from vector and raster sources, and may provide editing tools for both, since some parts of an image could come from a camera source, and others could have been drawn using vector tools.

Standards

The W3C standard for vector graphics is SVG. The standard is complex and has been relatively slow to be established at least in part owing to commercial interests. Many web browsers now have some support for rendering SVG data but full implementations of the standard are still comparatively rare.

An original reference photograph before vectorization

Detail can be added or removed from vector art. Vector illustrations can have their own colours, allowing artists to achieve desired results.

Note:- But there is no any content about list of file display command

Applications

One of the first uses of vector graphic displays was the US SAGE air defense system. Vector graphics systems were only retired from U.S. en route air traffic control in 1999, and are likely still in use in military and specialised systems. Vector graphics were also used on the TX-2 at the MIT Lincoln Laboratory by computer graphics pioneer Ivan Sutherland to run his program Sketchpad in 1963.

Subsequent vector graphics systems, most of which iterated through dynamically modifiable stored lists of drawing instructions, include Digital's GT40. There was a home gaming system that used vector graphics called Vectrex as well as various arcade games like *Asteroids* and *Space Wars*. Storage scope displays, such as the Tektronix 4014, could display vector images but not modify them without first erasing the display.

Modern vector graphics displays can sometimes be found at laser light shows, where two fast-moving X-Y mirrors are used to rapidly draw shapes and text on a screen.

The term "vector graphics" is mainly used today in the context of two-dimensional computer graphics. It is one of several modes an artist can use to create an image on a raster display. Other modes include text, multimedia, and 3D rendering. Virtually all modern 3D rendering is done using extensions of 2D vector graphics techniques. Plotters used in technical drawing still draw vectors directly to paper.

Information

For example, consider a circle of radius *r*. The main pieces of information a program needs in order to draw this circle are

1. an indication that what is to be drawn is a circle
2. the radius *r*
3. the location of the center point of the circle
4. stroke line style and colour (possibly transparent)
5. fill style and colour (possibly transparent)

Advantages to this style of drawing over raster graphics:

- This minimal amount of information translates to a much smaller file size compared to large raster images (the size of representation does not depend on the dimensions of the object), though a vector graphic with a small file size is often said to lack detail compared with a real world photo.
- Correspondingly, one can indefinitely zoom in on e.g. a circle arc, and it remains smooth. On the other hand, a polygon representing a curve will reveal being not really curved.
- On zooming in, lines and curves need not get wider proportionally. Often the width is either not increased or less than proportional. On the other hand, irregular curves represented by simple geometric shapes may be made proportionally wider when zooming in, to keep them looking smooth and not like these geometric shapes.
- The parameters of objects are stored and can be later modified. This means that moving, scaling, rotating, filling etc. doesn't degrade the quality of a drawing. Moreover, it is usual to specify the dimensions in device-independent units, which results in the best possible rasterization on raster devices.

- From a 3-D perspective, rendering shadows is also much more realistic with vector graphics, as shadows can be abstracted into the rays of light from which they are formed. This allows for photo realistic images and renderings.

Typical primitive objects

- Lines and polylines
- Polygons
- Circles and ellipses
- Bézier curves
- Bezigons
- Text (in computer font formats such as TrueType where each letter is created from Bézier curves)

This list is not complete. There are various types of curves (Catmull-Rom splines, NURBS etc.), which are useful in certain applications.

Often, a bitmap image is considered as a primitive object. From the conceptual view, it behaves as a rectangle.

Vector operations

Vector graphics editors typically allow rotation, movement, mirroring, stretching, skewing, affine transformations, changing of z-order and combination of primitives into more complex objects.

More sophisticated transformations include set operations on closed shapes (union, difference, intersection, etc.).

Vector graphics are ideal for simple or composite drawings that need to be device-independent, or do not need to achieve photo-realism. For example, the PostScript and PDF page description languages use a vector graphics model.

Printing

Vector art is key for printing. Since the art is made from a series of mathematical curves it will print very crisply even when resized. For instance, one can print a vector logo on a small sheet of copy paper, and then enlarge the same vector logo to billboard size and keep the same crisp quality. A low-resolution raster graphic would blur or pixelate excessively if it were enlarged from business card size to billboard size. (The precise resolution of a raster graphic necessary for high-quality results depends on the viewing distance; e.g., a billboard may still appear to be of high quality even at low resolution if the viewing distance is large enough.)

If we regard typographic characters as images, then the same considerations that we have made for graphics apply even to composition of written text for printing (typesetting).

Older character sets were stored as bitmaps, therefore to achieve maximum print quality they had to be used at a given resolution only; these font formats are said to be non-scalable. High quality typography is nowadays based on character drawings (fonts) which are typically stored as vector graphics, and as such are scalable to any size. Examples of these vector formats for characters are Postscript fonts and TrueType fonts.

Vector illustration

Vector illustration is a popular technique of many digital illustrators worldwide. Some of the greatest internationally acclaimed artists in the field are Catalina Estrada, Petra Stefankova, Nathan Jurevicius, J. Otto Seibold, Leo Blanchette and others.

3D modelling

In 3D computer graphics, vectorized surface representations are most common (bitmaps can be used for special purposes such as surface texturing, height-field data and bump mapping). At the low-end, simple meshes of polygons are used to represent geometric detail in applications where interactive frame rates or simplicity are important. At the high-end, where one is willing to trade-off higher rendering times for increased image quality and precision, smooth surface representations such as Bézier patches, NURBS or Subdivision surfaces are used. One can, however, achieve a smooth surface rendering from a polygonal mesh through the use of shading algorithms such as Phong and Gouraud.

Formats

One example of vector graphics format is SVG (Scalable Vector Graphics), an open standard created and developed by the World Wide Web Consortium to address the need (and attempts of several corporations) for a versatile, scriptable and all-purpose vector format for the web and otherwise. Another example is VML, a proposed standard that was adopted by Microsoft. Arguably, the AI format, the native format for Adobe Illustrator, is also standard in vector graphics.

The SWF Adobe's (formerly Macromedia's) file format is also a vector based container used to store animation. Web pages created in Flash can be enlarged to fit any monitor size while retaining the same graphic quality.

Adobe Fireworks, which is also (mostly) a vector-based application, saves its files in PNG format; vectors created in Adobe Fireworks can be transferred to Adobe Illustrator, and vice versa.

3D computer graphics

3D computer graphics (in contrast to 2D computer graphics) are graphics that use a three-dimensional representation of geometric data (often Cartesian) that is stored in the computer for the purposes of performing calculations and rendering 2D images. Such images may be stored for viewing later or displayed in real-time.

Despite these differences, 3D computer graphics rely on many of the same algorithms as 2D computer vector graphics in the wire-frame model and 2D computer raster graphics in the final rendered display. In computer graphics software, the distinction between 2D and 3D is occasionally blurred; 2D applications may use 3D techniques to achieve effects such as lighting, and 3D may use 2D rendering techniques.

3D computer graphics are often referred to as 3D models. Apart from the rendered graphic, the model is contained within the graphical data file. However, there are differences. A 3D model is the mathematical representation of any three-dimensional object. A model is not technically a graphic until it is displayed. Due to 3D printing, 3D models are not confined to virtual space. A model can be displayed visually as a two-dimensional image through a process called *3D rendering,* or used in non-graphical computer simulations and calculations.

History

William Fetter was credited with coining the term *computer graphics* in 1960 to describe his work at Boeing. One of the first displays of computer animation was *Futureworld* (1976), which included an animation of a human face and hand — produced by Ed Catmull and Fred Parke at the University of Utah.

Overview

The process of creating 3D computer graphics can be sequentially divided into three basic phases: 3D modeling which describes the process of forming the shape of an object, layout and animation which describes the *motion* and *placement* of objects within a scene, and 3D rendering which produces an *image* of an object.

Modeling

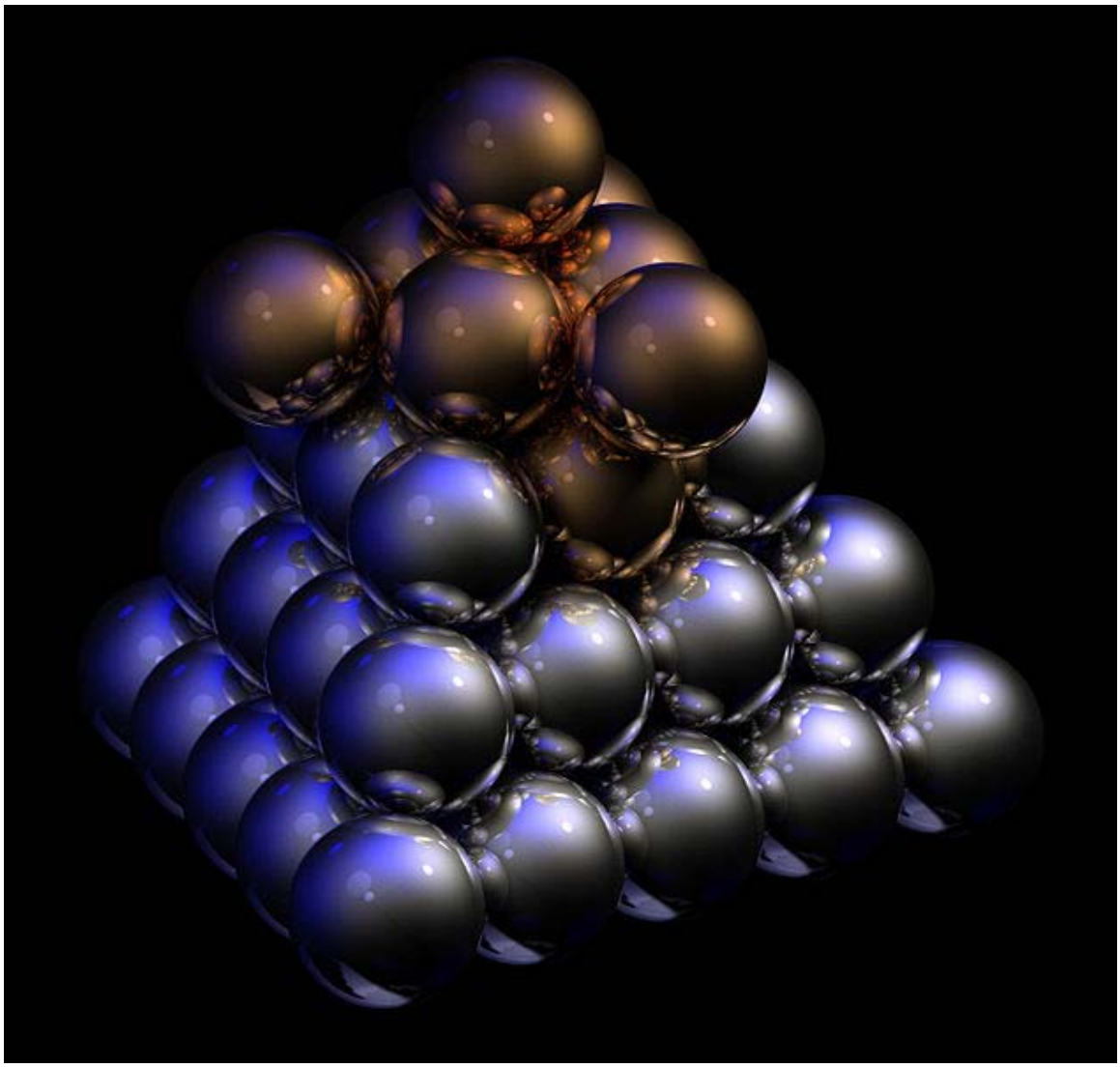
A 3D rendering with ray tracing and ambient occlusion using Blender and Yafray

The model describes the process of forming the shape of an object. The two most common sources of 3D models are those originated on the computer by an artist or engineer using some kind of 3D modeling tool, and those scanned into a computer from real-world objects. Models can also be produced procedurally or via physical simulation.

Layout and animation

Before objects are rendered, they must be placed (laid out) within a scene. This is what defines the spatial relationships between objects in a scene including location and size. Animation refers to the *temporal* description of an object, i.e., how it moves and deforms over time. Popular methods include keyframing, inverse kinematics, and motion capture, though many of these techniques are used in conjunction with each other. As with modeling, physical simulation is another way of specifying motion.

Rendering



During the 3D rendering step, the number of reflections "light rays" can take, as well as various other attributes, can be tailored to achieve a desired visual effect. *Image created with Cobalt*

Flat shading, 3DCG Dunkerque class battleship

Rendering converts a model into an image either by simulating light transport to get photorealistic images, or by applying some kind of style as in non-photorealistic rendering. The two basic operations in realistic rendering are transport (how much light gets from one place to another) and scattering (how surfaces interact with light). This step is usually performed using 3D computer graphics software or a 3D graphics API. The process of altering the scene into a suitable form for rendering also involves 3D projection which allows a three-dimensional image to be viewed in two dimensions.

Communities

There are a multitude of websites designed to help educate and support 3D graphic artists. Some are managed by software developers and content providers, but there are standalone sites as well. These communities allow for members to seek advice, post tutorials, provide product reviews or post examples of their own work.

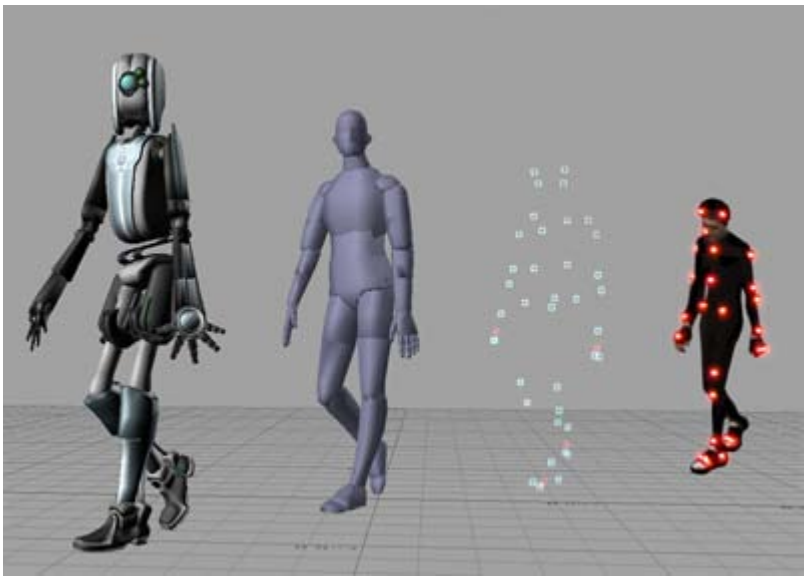
Distinction from photorealistic 2D graphics

Not all computer graphics that appear 3D are based on a wireframe model. 2D computer graphics with 3D photorealistic effects are often achieved without wireframe modeling

and are sometimes indistinguishable in the final form. Some graphic art software includes filters that can be applied to 2D vector graphics or 2D raster graphics on transparent layers. Visual artists may also copy or visualize 3D effects and manually render photorealistic effects without the use of filters.

Chapter 4

Computer Animation



An example of computer animation which is produced in the "motion capture" technique

Computer animation is the process used for generating animated images by using computer graphics. The more general term computer generated imagery encompasses both static scenes and dynamic images, while *computer animation* only refers to moving images produced by exploiting the persistence of vision to make a series of images look animated. Given that images last for about one twenty-fifth of a second on the retina fast image replacement creates the illusion of movement.

Modern computer animation usually uses 3D computer graphics, although 2D computer graphics are still used for stylistic, low bandwidth, and faster real-time renderings. Sometimes the target of the animation is the computer itself, but sometimes the target is another medium, such as film.

Computer animation is essentially a digital successor to the art of stop motion animation of 3D models and frame-by-frame animation of 2D illustrations. Computer generated animations are more controllable than other more physically based processes, such as constructing miniatures for effects shots or hiring extras for crowd scenes, and because it

allows the creation of images that would not be feasible using any other technology. It can also allow a single graphic artist to produce such content without the use of actors, expensive set pieces, or props.

To create the illusion of movement, an image is displayed on the computer screen and repeatedly replaced by a new image that is similar to the previous image, but advanced slightly in the time domain (usually at a rate of 24 or 30 frames/second). This technique is identical to how the illusion of movement is achieved with television and motion pictures.

For 3D animations, objects (models) are built on the computer monitor (modeled) and 3D figures are rigged with a virtual skeleton. For 2D figure animations, separate objects (illustrations) and separate transparent layers are used, with or without a virtual skeleton. Then the limbs, eyes, mouth, clothes, etc. of the figure are moved by the animator on key frames. The differences in appearance between key frames are automatically calculated by the computer in a process known as tweening or morphing. Finally, the animation is rendered.

For 3D animations, all frames must be rendered after modeling is complete. For 2D vector animations, the rendering process is the key frame illustration process, while tweened frames are rendered as needed. For pre-recorded presentations, the rendered frames are transferred to a different format or medium such as film or digital video. The frames may also be rendered in real time as they are presented to the end-user audience. Low bandwidth animations transmitted via the internet (e.g. 2D Flash, X3D) often use software on the end-users computer to render in real time as an alternative to streaming or pre-loaded high bandwidth animations.

A simple example

The screen is blanked to a background color, such as black. Then a goat is drawn on the right of the screen. Next the screen is blanked, but the goat is re-drawn or duplicated slightly to the left of its original position. This process is repeated, each time moving the goat a bit to the left. If this process is repeated fast enough the goat will appear to move smoothly to the left. This basic procedure is used for all moving pictures in films and television.

The moving goat is an example of shifting the location of an object. More complex transformations of object properties such as size, shape, lighting effects often require calculations and computer rendering instead of simple re-drawing or duplication.

Explanation

To trick the eye and brain into thinking they are seeing a smoothly moving object, the pictures should be drawn at around 12 frames per second (frame/s) or faster (a frame is one complete image). With rates above 70 frames/s no improvement in realism or smoothness is perceivable due to the way the eye and brain process images. At rates

below 12 frame/s most people can detect jerkiness associated with the drawing of new images which detracts from the illusion of realistic movement. Conventional hand-drawn cartoon animation often uses 15 frames/s in order to save on the number of drawings needed, but this is usually accepted because of the stylized nature of cartoons. Because it produces more realistic imagery computer animation demands higher frame rates to reinforce this realism.

The reason no jerkiness is seen at higher speeds is due to “persistence of vision.” From moment to moment, the eye and brain working together actually store whatever one looks at for a fraction of a second, and automatically "smooth out" minor jumps. Movie film seen in theaters in the United States runs at 24 frames per second, which is sufficient to create this illusion of continuous movement.

History

CGI was first used in movies in 1973's *Westworld,* a science-fiction film about a society in which robots live and work among humans, though the first use of 3D Wireframe imagery was in its sequel, *Futureworld* (1976), which featured a computer-generated hand and face created by then University of Utah graduate students Edwin Catmull and Fred Parke. The third movie to use this technology was *Star Wars* (1977) for the scenes with the wireframe Death Star plans and the targeting computers in the X-wings and the *Millennium Falcon*. *The Black Hole* (1979) used raster wire-frame model rendering to depict a black hole. The science fiction-horror film *Alien* of that same year also used a raster wire-frame model, in this case to render the image of navigation monitors in the sequence where a spaceship follows a beacon to a land on an unfamiliar planet.

In 1978, graduate students at the New York Institute of Technology Computer Graphics Lab began work on what would have been the first full-length CGI film, *The Works*, and a trailer for it was shown at SIGGRAPH 1982, but the film was never completed. *Star Trek II: The Wrath of Khan* premiered a short CGI sequence called The Genesis Wave in June 1982. The first two films to make heavy investments in Solid 3D CGI, *Tron* (1982) and *The Last Starfighter* (1984), were commercial failures, causing most directors to relegate CGI to images that were supposed to look like they were created by a computer.

Another Star Trek movie - 1986's *Star Trek IV: The Voyage Home* - features a little-known use of CGI that would prove to be a major stepping stone for the technology in years to come. The dream-like sequence where Kirk and his crew (in a commandeered Klingon starship) travel back in time features computer-generated images of the crew's faces "morphing" into one another. Although the images look like clay sculptures, they represent the antecedents of the photo-realistic morphing that would become popular in the early 90s, and the ILM team responsible would be sought out by James Cameron three years later (see below).

The first movie to feature photo-realistic CGI integrated seamlessly into scenes was *The Abyss* (1989). A five minute sequence featuring an animated water tentacle or "pseudopod" was created using groundbreaking new algorithms design by ILM, and

supervised by Denis Muren for James Cameron's underwater action movie. It featured reflection, refraction and a short "morphing" sequence, taking on the facial forms of actors Ed Harris and Mary Elizabeth Mastrantonio. Although Cameron's subsequent movie, *Terminator 2: Judgment Day*, would be the one that brought CGI to the attention of the masses, it was The Abyss that represented the true technological milestone, and the foundation upon which the modern CGI industry would be built.

1991 could be considered the breakout year for the new CGI technology. Two huge hits, *Terminator 2: Judgment Day* and *Beauty and the Beast*, both made heavy use of CGI. These successes marked Hollywood's transition from stop-motion animation and conventional optical effects to digital techniques. *Beauty and the Beast* became the first animated film to be nominated for Best Picture, and *Terminator 2: Judgment Day* won the Oscar for Best Visual Effects.

In 1993, another affirmation of CGI came from *Jurassic Park* where CGI dinosaurs were integrated into hydraulically-controlled life-sized puppets shot on-set. The palette of tools available from CGI was becoming larger. In 1994, CGI was used to create the special effects for *Forrest Gump*. The most noteworthy effects shots were those that featured the digital removal of actor Gary Sinise's legs. Other effects included a napalm strike, the fast-moving Ping-Pong balls, and the digital insertion of Tom Hanks into several scenes of historical footage.

In 1993, *Babylon 5* and *SeaQuest* became the first television series to use CGI as the primary method for their visual effects (rather than using hand-built models). It also marked the first TV use of virtual sets. That same year, *Insektors* became the first full-length completely computer animated TV series. Soon after, in 1994, the hit Canadian CGI show *ReBoot* aired.

In 1995, the first fully computer-generated feature film, Disney-Pixar's *Toy Story*, was a resounding commercial success. Additional digital animation studios such as Blue Sky Studios (20th Century Fox), DNA Productions (Paramount Pictures and Warner Bros.), Omation Studios (Paramount Pictures), Sony Pictures Animation (Columbia Pictures), Vanguard Animation (Walt Disney Pictures, Lions Gate Entertainment and 20th Century Fox), Big Idea Productions (Universal Pictures and FHE Pictures), Animal Logic (Warner Bros.) and Pacific Data Images (Dreamworks SKG) went into production, and existing animation companies, such as The Walt Disney Company, began to make a transition from traditional animation to CGI. Between 1995 and 2005 the average effects budget for a wide-release feature film skyrocketed from $5 million to $40 million. According to one studio executive, as of 2005, more than half of feature films have significant effects. However, CGI has made up for the expenditures by grossing over 20% more than their real-life counterparts.

In the early 2000s, computer-generated imagery became the dominant form of special effects. The technology progressed to the point that it became possible to include virtual stunt doubles. Camera tracking software was refined to allow increasingly complex visual effects developments that were previously impossible. Computer-generated extras also

became used extensively in crowd scenes with advanced flocking and crowd simulation software. Virtual sets, in which part or all of the background of a shot is digitally generated, also became commonplace. The timeline of CGI in film and television shows a detailed list of pioneering uses of computer-generated imagery in film and television.

CGI for films is usually rendered at about 1.4–6 megapixels. *Toy Story*, for example, was rendered at 1536 × 922 (1.42MP). The time to render one frame is typically around 2–3 hours, with ten times that for the most complex scenes. This time has not changed much in the last decade, as image quality has progressed at the same rate as improvements in hardware, since with faster machines, more and more complexity becomes feasible. Exponential increases in GPUs processing power, as well as massive increases in parallel CPU power, storage and memory speed and size have greatly increased CGI's potential.

In 2001, Square Pictures created the CGI film *Final Fantasy: The Spirits Within*, which made headlines for attempting to create photo-realistic human actors. The film was not a box-office success. Some commentators have suggested this may be partly because the lead CGI characters had facial features which fell into the uncanny valley. Square Pictures produced only two more films using a similar visual style *Final Flight of the Osiris*, a short film which served as a prologue to *The Matrix Reloaded* and *Final Fantasy VII: Advent Children*, based on their extremely popular video game series.

Developments in CGI technologies are reported each year at SIGGRAPH, an annual conference on computer graphics and interactive techniques, attended each year by tens of thousands of computer professionals. Developers of computer games and 3D video cards strive to achieve the same visual quality on personal computers in real-time as is possible for CGI films and animation. With the rapid advancement of real-time rendering quality, artists began to use game engines to render non-interactive movies. This art form is called *machinima*.

Methods of animating virtual characters

In most 3D computer animation systems, an animator creates a simplified representation of a character's anatomy, analogous to a skeleton or stick figure. The position of each segment of the skeletal model is defined by *animation variables*, or Avars. In human and animal characters, many parts of the skeletal model correspond to actual bones, but skeletal animation is also used to animate other things, such as facial features (though other methods for facial animation exist). The character "Woody" in *Toy Story*, for example, uses 700 Avars, including 100 Avars in the face. The computer does not usually render the skeletal model directly (it is invisible), but uses the skeletal model to compute the exact position and orientation of the character, which is eventually rendered into an image. Thus by changing the values of Avars over time, the animator creates motion by making the character move from frame to frame.

There are several methods for generating the Avar values to obtain realistic motion. Traditionally, animators manipulate the Avars directly. Rather than set Avars for every frame, they usually set Avars at strategic points (frames) in time and let the computer

interpolate or 'tween' between them, a process called keyframing. Keyframing puts control in the hands of the animator, and has roots in hand-drawn traditional animation.

In contrast, a newer method called motion capture makes use of live action. When computer animation is driven by motion capture, a real performer acts out the scene as if they were the character to be animated. His or her motion is recorded to a computer using video cameras and markers, and that performance is then applied to the animated character.

Each method has its advantages, and as of 2007, games and films are using either or both of these methods in productions. Keyframe animation can produce motions that would be difficult or impossible to act out, while motion capture can reproduce the subtleties of a particular actor. For example, in the 2006 film Pirates of the Caribbean: Dead Man's Chest, actor Bill Nighy provided the performance for the character Davy Jones. Even though Nighy himself doesn't appear in the film, the movie benefited from his performance by recording the nuances of his body language, posture, facial expressions, etc. Thus motion capture is appropriate in situations where believable, realistic behavior and action is required, but the types of characters required exceed what can be done through conventional costuming.

Creating characters and objects on a computer

3D **computer animation** combines 3D models of objects and programmed or hand "keyframed" movement. Models are constructed out of geometrical vertices, faces, and edges in a 3D coordinate system. Objects are sculpted much like real clay or plaster, working from general forms to specific details with various sculpting tools. A bone/joint animation system is set up to deform the CGI model (e.g., to make a humanoid model walk). In a process called rigging, the virtual marionette is given various controllers and handles for controlling movement. Animation data can be created using motion capture, or keyframing by a human animator, or a combination of the two.

3D models rigged for animation may contain thousands of control points - for example, the character "Woody" in Pixar's movie *Toy Story*, uses 700 specialized animation controllers. Rhythm and Hues Studios labored for two years to create Aslan in the movie *The Chronicles of Narnia: The Lion, the Witch and the Wardrobe* which had about 1851 controllers, 742 in just the face alone. In the 2004 film *The Day After Tomorrow*, designers had to design forces of extreme weather with the help of video references and accurate meteorological facts. For the 2005 remake of *King Kong*, actor Andy Serkis was used to help designers pinpoint the gorilla's prime location in the shots and used his expressions to model "human" characteristics onto the creature. Serkis had earlier provided the voice and performance for Gollum in J. R. R. Tolkien's *The Lord of the Rings* trilogy.

Computer animation development equipment

Professional animators of movies, television, and video sequences on computer games make photorealistic animation with high detail. This level of quality for movie animation would take tens to hundreds of years to create on a home computer. Many powerful workstation computers are used instead. Graphics workstation computers use two to four processors, and thus are a lot more powerful than a home computer, and are specialized for rendering. A large number of workstations (known as a render farm) are networked together to effectively act as a giant computer. The result is a computer-animated movie that can be completed in about one to five years (this process is not comprised solely of rendering, however). A workstation typically costs $2,000 to $16,000, with the more expensive stations being able to render much faster, due to the more technologically advanced hardware that they contain. Pixar's Renderman is rendering software which is widely used as the movie animation industry standard, in competition with Mental Ray. It can be bought at the official Pixar website for about $3,500. It will work on Linux, Mac OS X, and Microsoft Windows based graphics workstations along with an animation program such as Maya and Softimage XSI. Professionals also use digital movie cameras, motion capture or performance capture, bluescreens, film editing software, props, and other tools for movie animation.

The future

One open challenge in computer animation is a photorealistic animation of humans. Currently, most computer-animated movies show animal characters (*A Bug's Life*, *Finding Nemo*, *Ratatouille*, *Ice Age*, *Over the Hedge*), fantasy characters (*Monsters Inc.*, *Shrek*, *Teenage Mutant Ninja Turtles 4*, *Monsters vs. Aliens*), anthropomorphic machines (*Cars*, *WALL-E*, *Robots*) or cartoon-like humans (*The Incredibles*, *Despicable Me*, *Up*). The movie *Final Fantasy: The Spirits Within* is often cited as the first computer-generated movie to attempt to show realistic-looking humans. However, due to the enormous complexity of the human body, human motion, and human biomechanics, realistic simulation of humans remains largely an open problem. Another problem is the distasteful psychological response to viewing nearly perfect animation of humans, known as "the uncanny valley." It is one of the "holy grails" of computer animation. Eventually, the goal is to create software where the animator can generate a movie sequence showing a photorealistic human character, undergoing physically-plausible motion, together with clothes, photorealistic hair, a complicated natural background, and possibly interacting with other simulated human characters. This could be done in a way that the viewer is no longer able to tell if a particular movie sequence is computer-generated, or created using real actors in front of movie cameras. Complete human realism is not likely to happen very soon, but when it does it may have major repercussions for the film industry.

For the moment it looks like three dimensional computer animation can be divided into two main directions; photorealistic and non-photorealistic rendering. Photorealistic computer animation can itself be divided into two subcategories; real photorealism (where performance capture is used in the creation of the virtual human characters) and stylized photorealism. Real photorealism is what Final Fantasy tried to achieve and will

in the future most likely have the ability to give us live action fantasy features as *The Dark Crystal* without having to use advanced puppetry and animatronics, while *Antz* is an example on stylistic photorealism (in the future stylized photorealism will be able to replace traditional stop motion animation as in *Corpse Bride*). None of these mentioned are perfected as of yet, but the progress continues.

The non-photorealistic/cartoonish direction is more like an extension of traditional animation, an attempt to make the animation look like a three dimensional version of a cartoon, still using and perfecting the main principles of animation articulated by the Nine Old Men, such as squash and stretch.

While a single frame from a photorealistic computer-animated feature will look like a photo if done right, a single frame vector from a cartoonish computer-animated feature will look like a painting (not to be confused with cel shading, which produces an even simpler look).

Detailed examples and pseudocode

In 2D computer animation, moving objects are often referred to as “sprites.” A sprite is an image that has a location associated with it. The location of the sprite is changed slightly, between each displayed frame, to make the sprite appear to move. The following pseudocode makes a sprite move from left to right:

```
var int x := 0, y := screenHeight / 2;
while x < screenWidth
drawBackground()
drawSpriteAtXY (x, y) // draw on top of the background
x := x + 5 // move to the right
```

Computer animation uses different techniques to produce animations. Most frequently, sophisticated mathematics is used to manipulate complex three dimensional polygons, apply “textures”, lighting and other effects to the polygons and finally rendering the complete image. A sophisticated graphical user interface may be used to create the animation and arrange its choreography. Another technique called constructive solid geometry defines objects by conducting boolean operations on regular shapes, and has the advantage that animations may be accurately produced at any resolution.

Let's step through the rendering of a simple image of a room with flat wood walls with a grey pyramid in the center of the room. The pyramid will have a spotlight shining on it. Each wall, the floor and the ceiling is a simple polygon, in this case, a rectangle. Each corner of the rectangles is defined by three values referred to as X, Y and Z. X is how far left and right the point is. Y is how far up and down the point is, and Z is far in and out of the screen the point is. The wall nearest us would be defined by four points: (in the order x, y, z). Below is a representation of how the wall is defined

```
(0, 10, 0)                          (10, 10, 0)

(0,0,0)                             (10, 0, 0)
```

The far wall would be:

```
(0, 10, 20)                    (10, 10, 20)

(0, 0, 20)                     (10, 0, 20)
```

The pyramid is made up of five polygons: the rectangular base, and four triangular sides. To draw this image the computer uses math to calculate how to project this image, defined by three dimensional data, onto a two dimensional computer screen.

First we must also define where our view point is, that is, from what vantage point will the scene be drawn. Our view point is inside the room a bit above the floor, directly in front of the pyramid. First the computer will calculate which polygons are visible. The near wall will not be displayed at all, as it is behind our view point. The far side of the pyramid will also not be drawn as it is hidden by the front of the pyramid.

Next each point is perspective projected onto the screen. The portions of the walls 'furthest' from the view point will appear to be shorter than the nearer areas due to perspective. To make the walls look like wood, a wood pattern, called a texture, will be drawn on them. To accomplish this, a technique called "texture mapping" is often used. A small drawing of wood that can be repeatedly drawn in a matching tiled pattern (like wallpaper) is stretched and drawn onto the walls' final shape. The pyramid is solid grey so its surfaces can just be rendered as grey. But we also have a spotlight. Where its light falls we lighten colors, where objects blocks the light we darken colors.

Next we render the complete scene on the computer screen. If the numbers describing the position of the pyramid were changed and this process repeated, the pyramid would appear to move.

Movies

CGI short films have been produced as independent animation since 1976, though the popularity of computer animation (especially in the field of special effects) skyrocketed during the modern era of U.S. animation. The first completely computer-generated television series was *ReBoot*, in 1994, and the first completely computer-generated animated movie was *Toy Story* (1995).

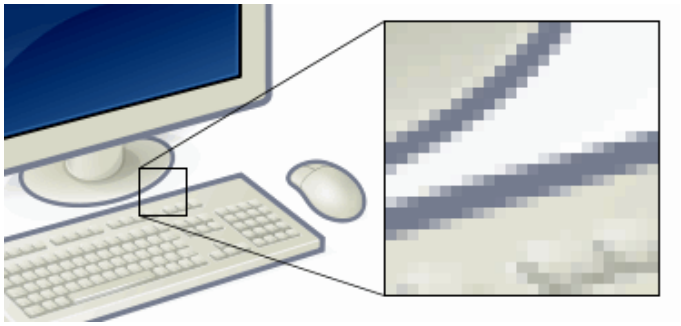
Amateur animation

The popularity of websites which allows members to upload their own movies for others to view has created a growing community of amateur computer animators. With utilities and programs often included free with modern operating systems, many users can make their own animated movies and shorts. Several free and open source animation software applications exist as well. A popular amateur approach to animation is via the animated GIF format, which can be uploaded and seen on the web easily.

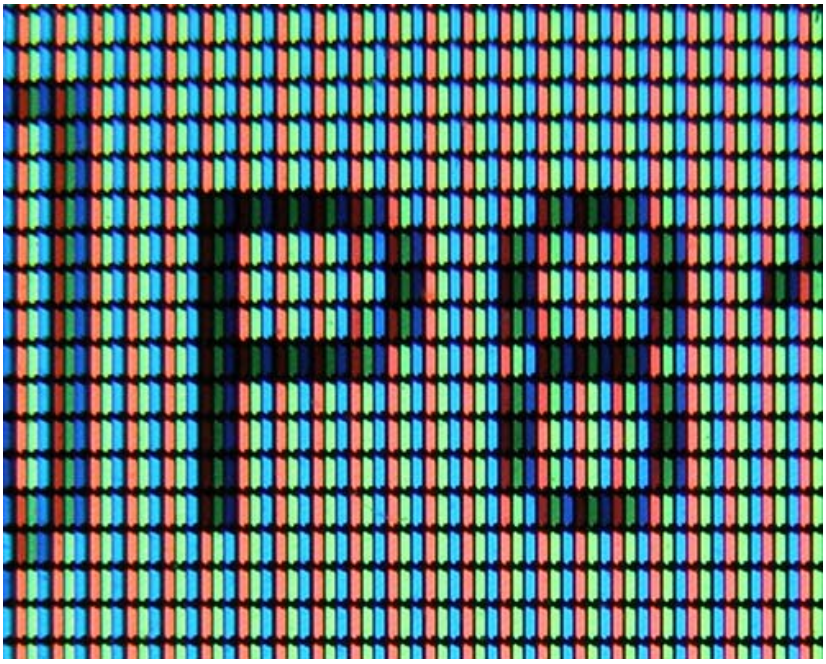
Chapter 5

Pixel and Rendering Equation

Pixel



This example shows an image with a portion greatly enlarged, in which the individual pixels are rendered as little squares and can easily be seen.



A photograph of sub-pixel display elements on a laptop's LCD screen

In digital imaging, a **pixel** (or **picture element**) is a single point in a raster image. The pixel is the smallest addressable screen element; it is the smallest unit of picture that can be controlled. Each pixel has its own address. The address of a pixel corresponds to its coordinates. Pixels are normally arranged in a two-dimensional grid, and are often represented using dots or squares. Each pixel is a sample of an original image; more samples typically provide more accurate representations of the original. The intensity of each pixel is variable. In color image systems, a color is typically represented by three or four component intensities such as red, green, and blue, or cyan, magenta, yellow, and black.

In some contexts (such as descriptions of camera sensors), the term *pixel* is used to refer to a single scalar element of a multi-component representation (more precisely called a *photosite* in the camera sensor context, although the neologism *sensel* is also sometimes used to describe the elements of a digital camera's sensor), while in others the term may refer to the entire set of such component intensities for a spatial position. In color systems that use chroma subsampling, the multi-component concept of a pixel can become difficult to apply, since the intensity measures for the different color components correspond to different spatial areas in a such a representation.

The word *pixel* is based on a contraction of *pix* ("pictures") and *el* (for "element"); similar formations with *el* for "element" include the words voxel and texel.

Etymology

The word "pixel" was first published in 1965 by Frederic C. Billingsley of JPL, to describe the picture elements of video images from space probes to the Moon and Mars. However, Billingsley did not coin the term himself. Instead, he got the word "pixel" from Keith E. McFarland, at the Link Division of General Precision in Palo Alto, who did not know where the word originated. McFarland said simply it was "in use at the time" (circa 1963).

The word is a combination of *picture* and *element*, via *pix*. The word *pix* appeared in *Variety* magazine headlines in 1932, as an abbreviation for the word *pictures*, in reference to movies. By 1938, "pix" was being used in reference to still pictures by photojournalists.

The concept of a "picture element" dates to the earliest days of television, for example as "Bildpunkt" (the German word for *pixel,* literally *picture point*) in the 1888 German patent of Paul Nipkow. According to various etymologies, the earliest publication of the term *picture element* itself was in *Wireless World* magazine in 1927, though it had been used earlier in various U.S. patents filed as early as 1911.

Some authors explain *pixel* as *picture cell,* as early as 1876. In video processing, *pel* is often used instead of *pixel*. For example, IBM used it in their Technical Reference for the original PC.

Words with similar etymologies

- Texel (texture element) and luxel (lux element) are words used to describe a pixel when it is used in specific context (texturing and light mapping respectively)
- A voxel is a *volume element*, the 3D analogue of a 2D pixel.
- Surfels (surface elements) have the same naming pattern as pixels, but share more similarities with shrunken triangles than expanded pixels.
- A resel is a *resolution element*, used to describe the discrete number of resolvable elements of a continuous signal (e.g. in optical imaging).

Technical

A pixel does not need to be rendered as a small square. This image shows alternative ways of reconstructing an image from a set of pixel values, using dots, lines, or smooth filtering.

A pixel is generally thought of as the smallest single component of a digital image. The definition is highly context-sensitive. For example, there can be "printed pixels" in a page, or pixels carried by electronic signals, or represented by digital values, or pixels on a display device, or pixels in a digital camera (photosensor elements). This list is not exhaustive, and depending on context, there are several terms that are synonymous in particular contexts, such as pel, sample, byte, bit, dot, spot, etc. The term "pixels" can be used in the abstract, or as a unit of measure, in particular when using pixels as a measure of resolution, such as: 2400 pixels per inch, 640 pixels per line, or spaced 10 pixels apart.

The measures dots per inch (dpi) and pixels per inch (ppi) are sometimes used interchangeably, but have distinct meanings, especially for printer devices, where dpi is a measure of the printer's density of dot (e.g. ink droplet) placement. For example, a high-quality photographic image may be printed with 600 ppi on a 1200 dpi inkjet printer. Even higher dpi numbers, such as the 4800 dpi quoted by printer manufacturers since 2002, do not mean much in terms of achievable resolution.

The more pixels used to represent an image, the closer the result can resemble the original. The number of pixels in an image is sometimes called the resolution, though

resolution has a more specific definition. Pixel counts can be expressed as a single number, as in a "three-megapixel" digital camera, which has a nominal three million pixels, or as a pair of numbers, as in a "640 by 480 display", which has 640 pixels from side to side and 480 from top to bottom (as in a VGA display), and therefore has a total number of 640 × 480 = 307,200 pixels or 0.3 megapixels.

The pixels, or color samples, that form a digitized image (such as a JPEG file used on a web page) may or may not be in one-to-one correspondence with screen pixels, depending on how a computer displays an image. In computing, an image composed of pixels is known as a *bitmapped image* or a *raster image*. The word *raster* originates from television scanning patterns, and has been widely used to describe similar halftone printing and storage techniques.

Sampling patterns

For convenience, pixels are normally arranged in a regular two-dimensional grid. By using this arrangement, many common operations can be implemented by uniformly applying the same operation to each pixel independently. Other arrangements of pixels are also possible, with some sampling patterns even changing the shape (or kernel) of each pixel across the image. For this reason, care must be taken when acquiring an image on one device and displaying it on another, or when converting image data from one pixel format to another.

For example:

- LCD screens typically use a staggered grid, where the red, green, and blue components are sampled at slightly different locations. Subpixel rendering is a technology which takes advantage of these differences to improve the rendering of text on LCD screens.
- Some digital cameras use a Bayer filter, resulting in a regular grid of pixels where the *color* of each pixel depends on its position on the grid.
- A clipmap uses a hierarchical sampling pattern, where the size of the support of each pixel depends on its location within the hierarchy.
- Warped grids are used when the underlying geometry is non-planar, such as images of the earth from space.
- The use of non-uniform grids is an active research area, attempting to bypass the traditional Nyquist limit.
- Pixels on computer monitors are normally "square" (this is, having equal horizontal and vertical sampling pitch); pixels in other systems are often "rectangular" (that is, having unequal horizontal and vertical sampling pitch – oblong in shape), as are digital video formats with diverse aspect ratios, such as the anamorphic widescreen formats of the Rec. 601 digital video standard.

Display resolution vs. native resolution in computer monitors

Computers can use pixels to display an image, often an abstract image that represents a GUI. The resolution of this image is called the display resolution and is determined by the video card of the computer. LCD computer *monitors* also use pixels to display an image, and have a native resolution. Each pixel is made up of triads, with the number of these triads determining the native resolution. On some CRT monitors, the beam sweep rate may be fixed, resulting in a fixed native resolution. Most CRT monitors do not have a fixed beam sweep rate, meaning they don't have a native resolution at all - instead they have a set of resolutions that are equally well supported.

To produce the sharpest images possible on an LCD, the user must ensure the display resolution of the computer matches the native resolution of the monitor. On a CRT with a fixed beam sweep rate, you are required to use the native resolution. On a CRT without this restriction, you may use any resolution supported by the monitor that matches the monitor's physical aspect ratio and it will look fine. If the aspect ratio of the physical display and the selected resolution are different, many different things can happen. On some LCDs, the monitor will stretch or squash the image to fill the entire display. This can result in the image appearing blurry or jagged.

On others, the aspect ratio will be maintained while expanding the image to fit the display, resulting in black bars on the top or sides of the image. This can also result in the image appearing blurry or jagged, depending on the native resolution of the display and the selected resolution on the computer. For example, let's take the relatively common case of a full-screen application written assuming a 4:3 aspect ratio running on a 16:10 aspect ratio widescreen display. If the selected resolution were 1600×1200 and you were running it on a 1920×1200 display that maintains aspect ratio while expanding the image to fit, the image would not look blurry, because each pixel in the 1600×1200 image maps to exactly 1 pixel on the 1920×1200 display. If the selected resolution were 1280×960, the display would have to try to stretch the 960 pixels to fill 1200 pixels, which would mean each of the selected resolution pixels needs to take up 1.25 pixels on the physical display. Since this can't be done, the monitor uses some scheme to figure out how to distribute the colors from those 960 pixels to fill the 1200 pixels of the physical display. This mapping results in a blurry or jagged appearance. However, if the selected resolution were 800×600, you would be OK again, because 600 pixels can be expanded to 1200 pixels by having each pixel from the selected resolution take up 2 pixels on the physical display.

On yet other LCD monitors, if the selected resolution is less than the native resolution of the monitor, the monitor will display things in the selected resolution, with a black border around the edges.

Bits per pixel

The number of distinct colors that can be represented by a pixel depends on the number of bits per pixel (bpp). A 1 bpp image uses 1-bit for each pixel, so each pixel can be

either on or off. Each additional bit doubles the number of colors available, so a 2 bpp image can have 4 colors, and a 3 bpp image can have 8 colors:

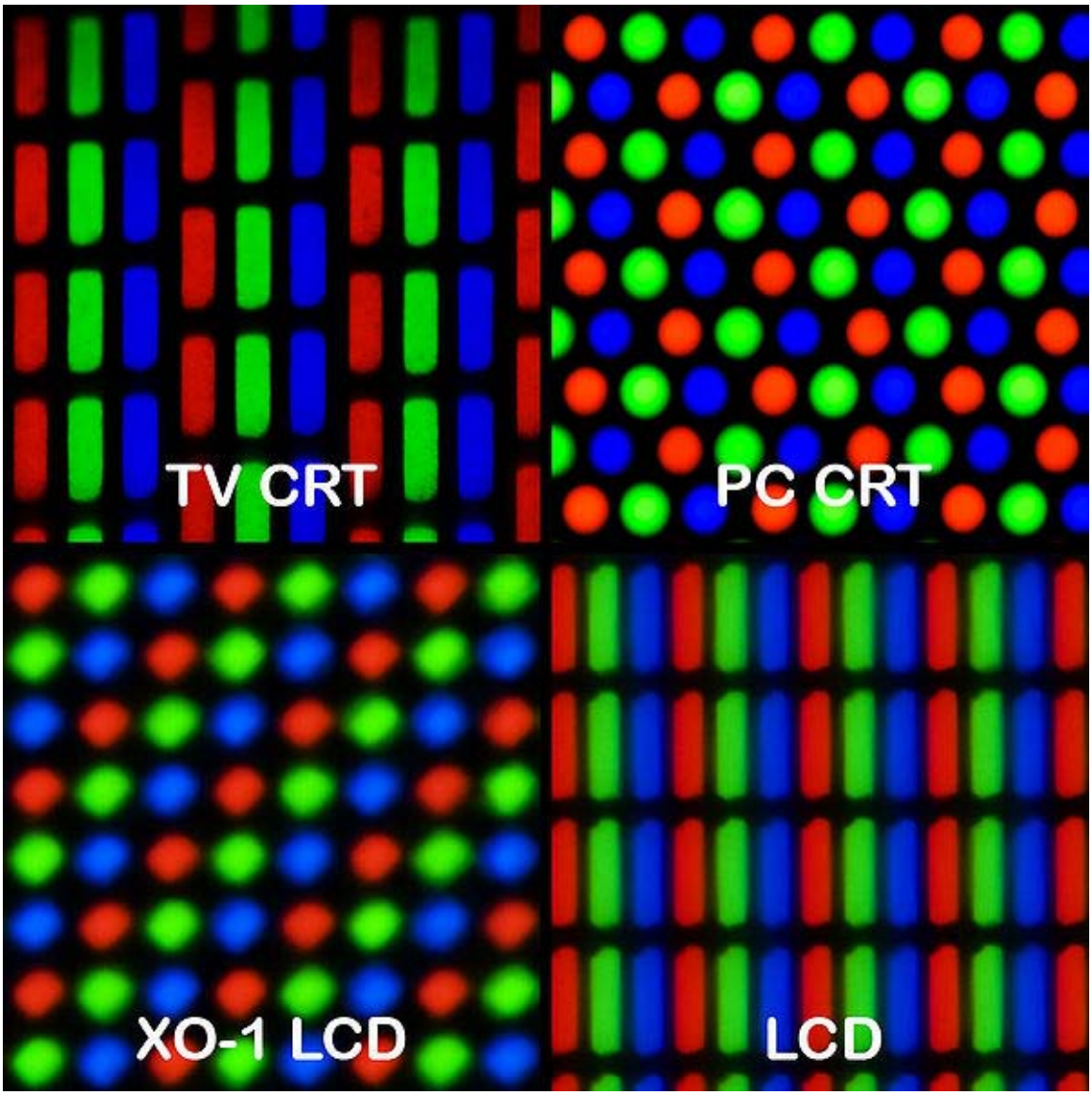
- 1 bpp, 2^1 = 2 colors (monochrome)
- 2 bpp, 2^2 = 4 colors
- 3 bpp, 2^3 = 8 colors

...

- 8 bpp, 2^8 = 256 colors
- 16 bpp, 2^{16} = 65,536 colors ("Highcolor")
- 24 bpp, $2^{24} \approx$ 16.8 million colors ("Truecolor")

For color depths of 15 or more bits per pixel, the depth is normally the sum of the bits allocated to each of the red, green, and blue components. Highcolor, usually meaning 16 bpp, normally has five bits for red and blue, and six bits for green, as the human eye is more sensitive to errors in green than in the other two primary colors. For applications involving transparency, the 16 bits may be divided into five bits each of red, green, and blue, with one bit left for transparency. A 24-bit depth allows 8 bits per component. On some systems, 32-bit depth is available: this means that each 24-bit pixel has an extra 8 bits to describe its opacity (for purposes of combining with another image).

Subpixels



Geometry of color elements of various CRT and LCD displays; phosphor dots in a color CRT display (top row) bear no relation to pixels or subpixels.

Many display and image-acquisition systems are, for various reasons, not capable of displaying or sensing the different color channels at the same site. Therefore, the pixel grid is divided into single-color regions that contribute to the displayed or sensed color when viewed at a distance. In some displays, such as LCD, LED, and plasma displays, these single-color regions are separately addressable elements, which have come to be known as *subpixels*. For example, LCDs typically divide each pixel horizontally into three subpixels. When the square pixel is divided into three subpixels, each subpixel is necessarily rectangular. In the display industry terminology, subpixels are often referred to as *pixels*, as they are the basic addressable elements in a viewpoint of hardware, and they call *pixel circuits* rather than *subpixel circuits*.

Most digital camera image sensors also use single-color sensor regions, for example using the Bayer filter pattern, and in the camera industry these are known as *pixels* just like in the display industry, not *subpixels*.

For systems with subpixels, two different approaches can be taken:

- The subpixels can be ignored, with full-color pixels being treated as the smallest addressable imaging element; or
- The subpixels can be included in rendering calculations, which requires more analysis and processing time, but can produce apparently superior images in some cases.

This latter approach, referred to as subpixel rendering, uses knowledge of pixel geometry to manipulate the three colored subpixels separately, producing a slight increase in the apparent resolution of color displays. While CRT displays also use red-green-blue masked phosphor areas, dictated by a mesh grid called the shadow mask, it would require a difficult calibration step to be aligned with the displayed pixel raster, and so CRTs do not currently use subpixel rendering.

The concept of Subpixel is related to Sample (graphics).

Megapixel

A megapixel (MP or Mpx) is one million pixels, and is a term used not only for the number of pixels in an image, but also to express the number of image sensor elements of digital cameras or the number of display elements of digital displays. For example, a camera with an array of 2048×1536 sensor elements is commonly said to have "3.1 megapixels" (2048 × 1536 = 3,145,728). The megapixel count is often used as a figure of merit, though it is only one of the figures that determines camera quality.

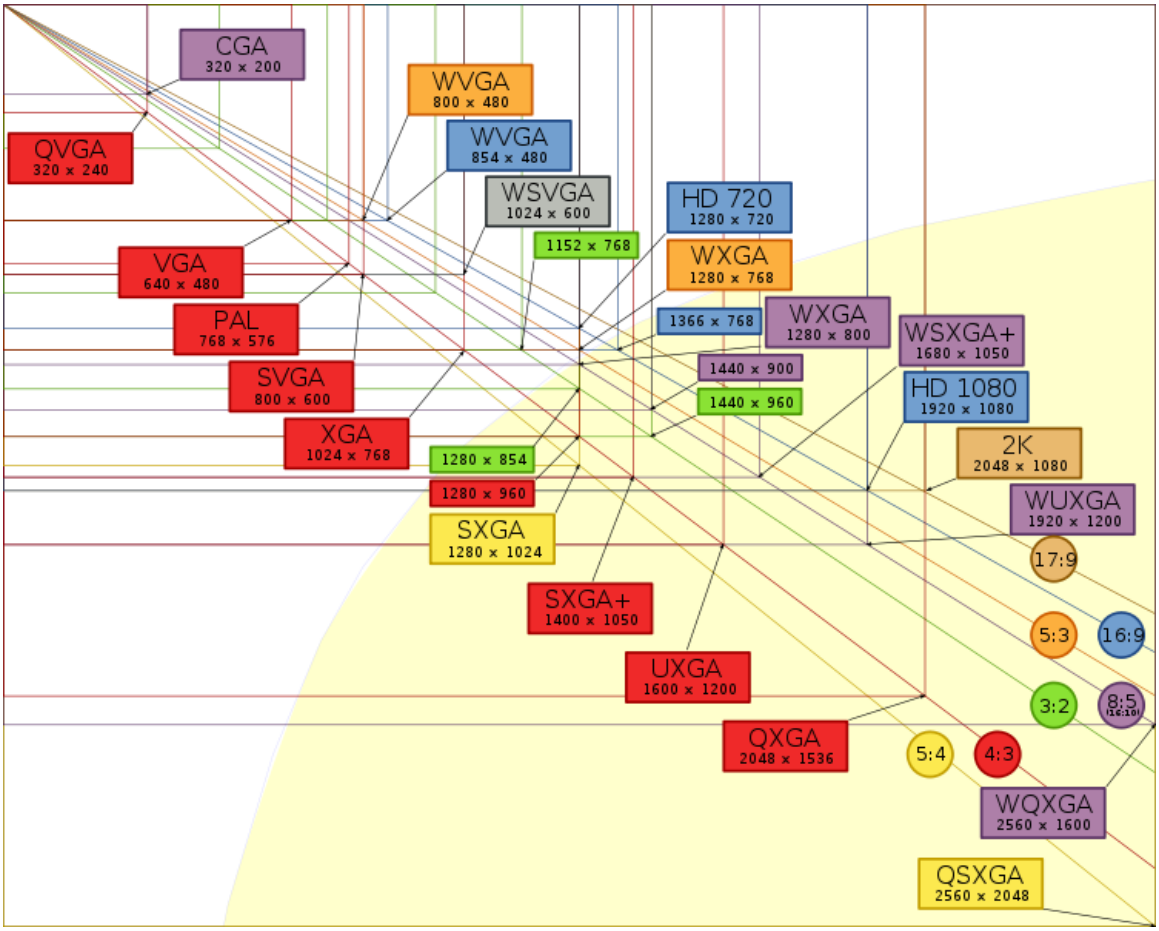
Digital cameras use photosensitive electronics, either charge-coupled device (CCD) or complementary metal–oxide–semiconductor (CMOS) image sensors, consisting of a large number of single sensor elements, each of which records a measured intensity level. In most digital cameras, the sensor array is covered with a patterned color filter mosaic having red, green, and blue regions in the Bayer filter arrangement, so that each sensor element can record the intensity of a single primary color of light. The camera interpolates the color information of neighboring sensor elements, through a process called demosaicing, to create the final image. These sensor elements are often called "pixels", even though they only record 1 channel (only red, or green, or blue) of the final color image. Thus, two of the three color channels for each sensor must be interpolated and a so-called *N-megapixel* camera that produces an N-megapixel image provides only one-third of the information that an image of the same size could get from a scanner. Thus, certain color contrasts may look fuzzier than others, depending on the allocation of the primary colors (green has twice as many elements as red or blue in the Bayer arrangement).

Standard display resolutions

The display resolution of a digital television or display device is the number of distinct pixels in each dimension that can be displayed. It can be an ambiguous term especially as the displayed resolution is controlled by all different factors in cathode ray tube (CRT) and flat panel or projection displays using fixed picture-element (pixel) arrays.

One use of the term “display resolution” applies to fixed-pixel-array displays such as plasma display panels (PDPs), liquid crystal displays (LCDs), Digital Light Processing (DLP) projectors, or similar technologies, and is simply the physical number of columns and rows of pixels creating the display (e.g., 1920×1200). A consequence of having a fixed grid display is that, for multi-format video inputs, all displays need a "scaling engine" (a digital video processor that includes a memory array) to match the incoming picture format to the display.

Note that the use of the word *resolution* here is misleading. The term “display resolution” is usually used to mean *pixel dimensions* (e.g., 1920×1200), which does not tell anything about the resolution of the display on which the image is actually formed. In digital measurement, the display resolution would be given in pixels per inch. In analog measurement, if the screen is 10 inches high, then the horizontal resolution is measured across a square 10 inches wide. This is typically stated as “xxx lines horizontal resolution, per picture height.” Example: Analog NTSC and PAL TVs can typically display 480 (for NTSC) lines horizontal resolution, per picture height which is equivalent to 640 total lines from left-edge to right-edge.



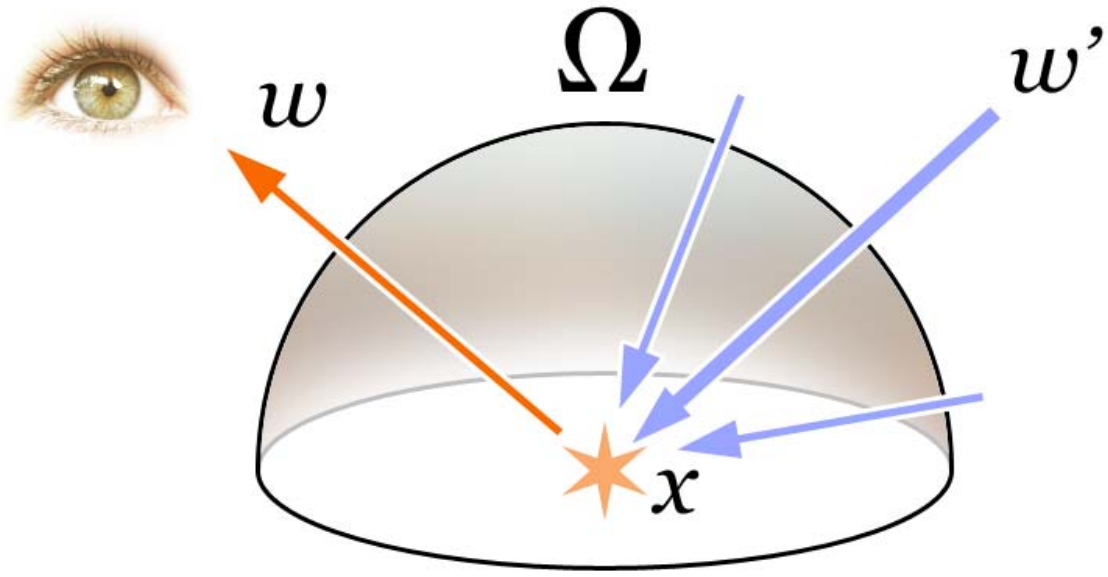
Display standards comparison

Selected standard display resolutions include:

Name	Megapixels	Width×Height
CGA	0.06	320×200
EGA	0.22	640×350
VGA	0.31	640×480
SVGA	0.48	800×600
XGA	0.79	1024×768
SXGA	1.31	1280×1024
UXGA	1.92	1600×1200
WUXGA	2.30	1920×1200
QXGA	3.15	2048×1536
QSXGA	5.24	2560×2048
QUXGA	7.68	3200×2400
WQUXGA	9.22	3840×2400

HXGA	12.58	4096×3072
HSXGA	20.97	5120×4096
HUXGA	30.72	6400×4800
WHUXGA	36.86	7680×4800

Rendering equation



The rendering equation describes the total amount of light emitted from a point **x** along a particular viewing direction, given a function for incoming light and a BRDF.

In computer graphics, the **rendering equation** is an integral equation in which the equilibrium radiance leaving a point is given as the sum of emitted plus reflected radiance under a geometric optics approximation. It was simultaneously introduced into computer graphics by David Immel et al. and James Kajiya in 1986. The various realistic rendering techniques in computer graphics attempt to solve this equation.

The physical basis for the rendering equation is the law of conservation of energy. Assuming that L denotes radiance, we have that at each 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 itself is the sum of the incoming light (L_i) from all directions, multiplied by the surface reflection and cosine of the incident angle.

The rendering equation may be written in the form

$$L_o(\mathbf{x}, \omega, \lambda, t) = L_e(\mathbf{x}, \omega, \lambda, t) + \int_\Omega f_r(\mathbf{x}, \omega', \omega, \lambda, t) L_i(\mathbf{x}, \omega', \lambda, t)(-\omega' \cdot \mathbf{n}) d\omega'$$

where

- $\lambda$is a particular wavelength of light
- $t$is time
- $L_o(\mathbf{x}, \omega, \lambda, t)$is the total amount of light of wavelength λdirected outward along direction ω at time t, from a particular position $\mathbf{x}$
- $L_e(\mathbf{x}, \omega, \lambda, t)$is emitted light
- $\int_\Omega \cdots d\omega'$ is an integral over a hemisphere of inward directions
- $f_r(\mathbf{x}, \omega', \omega, \lambda, t)$is the bidirectional reflectance distribution function, the proportion of light reflected from ω' to ω at position $\mathbf{x}$, time t, and at wavelength λ
- $L_i(\mathbf{x}, \omega', \lambda, t)$is light of wavelength λcoming inward toward $\mathbf{x}$from direction ω' at time t
- $-\omega' \cdot \mathbf{n}$is the attenuation of inward light due to incident angle

Two noteworthy features are: its linearity—it is composed only of multiplications and additions, and its spatial homogeneity—it is the same in all positions and orientations. These mean a wide range of factorings and rearrangements of the equation are possible.

Note this equation's spectral and time dependence—L_o may be sampled at or integrated over sections of the visible spectrum to obtain, for example, a trichromatic color sample. A pixel value for a single frame in an animation may be obtained by fixing t; motion blur can be produced by integrating $L_o$over t.

Although the equation is very general, it does not capture every aspect of light reflection. Some missing aspects include the following:

- phosphorescence, which occurs when light is absorbed at one moment in time and emitted at a different time,
- fluorescence, where the absorbed and emitted light have different wavelengths,
- interference, where the wave properties of light are exhibited, and
- subsurface scattering, where the spatial locations for incoming and departing light are different. Surfaces rendered without accounting for subsurface scattering may appear unnaturally opaque.

Solving the rendering equation for any given scene is the primary challenge in realistic rendering. One approach to solving the equation is based on finite element methods, leading to the radiosity algorithm. Another approach using Monte Carlo methods has led to many different algorithms including path tracing, photon mapping, and Metropolis light transport, among others.

For scenes that are either not composed of simple surfaces in a vacuum or for which the travel time for light is an important factor, researchers have generalized the rendering equation to produce a *volume rendering equation* suitable for volume rendering and a *transient rendering equation* for use with data from a time-of-flight camera.

Chapter 6

3D Projection and Ray Tracing (Graphics)

3D projection

3D projection is any method of mapping three-dimensional points to a two-dimensional plane. As most current methods for displaying graphical data are based on planar two-dimensional media, the use of this type of projection is widespread, especially in computer graphics, engineering and drafting.

Orthographic projection

When the human eye looks at a scene, objects in the distance appear smaller than objects close by. Orthographic projection ignores this affect to allow the creation of to-scale drawings for construction and engineering.

Orthographic projections are a small set of transforms often used to show profile, detail or precise measurements of a three dimensional object. Common names for orthographic projections include plane, cross-section, bird's-eye, and elevation.

If the normal of the viewing plane (the camera direction) is parallel to one of the 3D axes, the mathematical transformation is as follows; To project the 3D point a_x, a_y, a_z onto the 2D point b_x, b_y using an orthographic projection parallel to the y axis (profile view), the following equations can be used:

$$b_x = s_x a_x + c_x$$
$$b_y = s_z a_z + c_z$$

where the vector **s** is an arbitrary scale factor, and **c** is an arbitrary offset. These constants are optional, and can be used to properly align the viewport. Using matrix multiplication, the equations become:

$$\begin{bmatrix} b_x \\ b_y \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & 0 & s_z \end{bmatrix} \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} + \begin{bmatrix} c_x \\ c_z \end{bmatrix}.$$

While orthographically projected images represent the three dimensional nature of the object projected, they do not represent the object as it would be recorded photographically or perceived by a viewer observing it directly. In particular, parallel lengths at all points in an orthographically projected image are of the same scale regardless of whether they are far away or near to the virtual viewer. As a result, lengths near to the viewer are not foreshortened as they would be in a perspective projection.

Perspective projection

When the human eye looks at a scene, objects in the distance appear smaller than objects close by - this is known as perspective. While orthographic projection ignores this affect to allow accurate measurements, perspective definition shows distant objects as smaller to provide additional realism.

The perspective projection requires greater definition. A conceptual aid to understanding the mechanics of this projection involves treating the 2D projection as being viewed through a camera viewfinder. The camera's position, orientation, and field of view control the behavior of the projection transformation. The following variables are defined to describe this transformation:

- $\mathbf{a}_{x,y,z}$- the point in 3D space that is to be projected.
- $\mathbf{c}_{x,y,z}$- the location of the camera.
- $\theta_{x,y,z}$- The rotation of the camera. When $\mathbf{c}_{x,y,z} = \langle 0, 0, 0 \rangle$, and $\theta_{x,y,z} = \langle 0, 0, 0 \rangle$, the 3D vector <1,2,0> is projected to the 2D vector <1,2>.
- $\mathbf{e}_{x,y,z}$- the viewer's position relative to the display surface.

Which results in:

- $\mathbf{b}_{x,y}$- the 2D projection of $\mathbf{a}$.

First, we define a point $\mathbf{d}_{x,y,z}$ as a translation of point $\mathbf{a}$ into a coordinate system defined by $\mathbf{c}$. This is achieved by subtracting $\mathbf{c}$ from $\mathbf{a}$ and then applying a vector rotation matrix using $-\theta$ to the result. This transformation is often called a **camera transform**, and can be expressed as follows, expressing the rotation in terms of rotations about the *x, y,* and *z* axes (these calculations assume that the axes are ordered as a left-handed system of axes):

$$\begin{bmatrix} \mathbf{d}_x \\ \mathbf{d}_y \\ \mathbf{d}_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x \\ 0 & \sin\theta_x & \cos\theta_x \end{bmatrix} \begin{bmatrix} \cos\theta_y & 0 & \sin\theta_y \\ 0 & 1 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y \end{bmatrix} \begin{bmatrix} \cos\theta_z & -\sin\theta_z & 0 \\ \sin\theta_z & \cos\theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix} \left(\begin{bmatrix} \mathbf{a}_x \\ \mathbf{a}_y \\ \mathbf{a}_z \end{bmatrix} - \begin{bmatrix} \mathbf{c}_x \\ \mathbf{c}_y \\ \mathbf{c}_z \end{bmatrix} \right)$$

This representation correspond to rotating by three Euler angles (more properly, Tait–Bryan angles), using the *xyz* convention, which can be interpreted either as "rotate about the *extrinsic* axes (axes of the *scene*) in the order *z, y, x* (reading right-to-left)" or "rotate about the *intrinsic* axes (axes of the *camera*) in the order *x, y, z*) (reading left-to-right)". Note that if the camera is not rotated ($\theta_{x,y,z} = \langle 0, 0, 0 \rangle$), then the matrices drop out (as identities), and this reduces to simply a shift: $\mathbf{d} = \mathbf{a} - \mathbf{c}$.

Alternatively, without using matrices, (note that the signs of angles are inconsistent with matrix form):

$$\begin{aligned}
d_x &= \cos\theta_y \cdot (\sin\theta_z \cdot (a_y - c_y) + \cos\theta_z \cdot (a_x - c_x)) - \sin\theta_y \cdot (a_z - c_z) \\
d_y &= \sin\theta_x \cdot (\cos\theta_y \cdot (a_z - c_z) + \sin\theta_y \cdot (\sin\theta_z \cdot (a_y - c_y) + \cos\theta_z \cdot (a_x - c_x))) + \cos\theta_x \cdot (\cos\theta_z \cdot (a_y - c_y) - \sin\theta_z \cdot (a_x - c_x)) \\
d_z &= \cos\theta_x \cdot (\cos\theta_y \cdot (a_z - c_z) + \sin\theta_y \cdot (\sin\theta_z \cdot (a_y - c_y) + \cos\theta_z \cdot (a_x - c_x))) - \sin\theta_x \cdot (\cos\theta_z \cdot (a_y - c_y) - \sin\theta_z \cdot (a_x - c_x))
\end{aligned}$$

This transformed point can then be projected onto the 2D plane using the formula (here, *x/y* is used as the projection plane, literature also may use *x/z*):

$$\begin{aligned}
\mathbf{b}_x &= (\mathbf{d}_x - \mathbf{e}_x)(\mathbf{e}_z / \mathbf{d}_z) \\
\mathbf{b}_y &= (\mathbf{d}_y - \mathbf{e}_y)(\mathbf{e}_z / \mathbf{d}_z)
\end{aligned}.$$

Or, in matrix form using homogeneous coordinates, the system

$$\begin{bmatrix} \mathbf{f}_x \\ \mathbf{f}_y \\ \mathbf{f}_z \\ \mathbf{f}_w \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -\mathbf{e}_x \\ 0 & 1 & 0 & -\mathbf{e}_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/\mathbf{e}_z & 0 \end{bmatrix} \begin{bmatrix} \mathbf{d}_x \\ \mathbf{d}_y \\ \mathbf{d}_z \\ 1 \end{bmatrix}$$

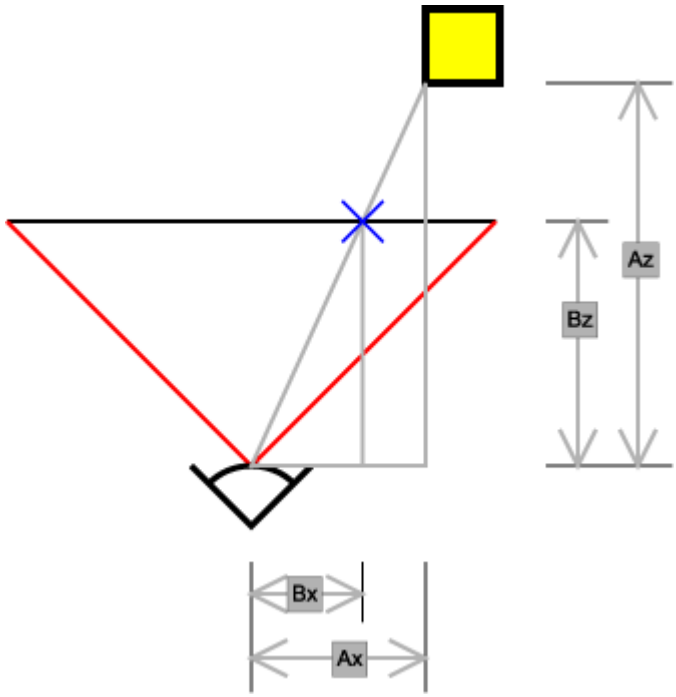
in conjunction with an argument using similar triangles, leads to division by the homogeneous coordinate, giving

$$\begin{aligned}
\mathbf{b}_x &= \mathbf{f}_x / \mathbf{f}_w \\
\mathbf{b}_y &= \mathbf{f}_y / \mathbf{f}_w
\end{aligned}.$$

The distance of the viewer from the display surface, $\mathbf{e}_z$, directly relates to the field of view, where $\alpha = 2 \cdot \tan^{-1}(1/\mathbf{e}_z)$is the viewed angle. (Note: This assumes that you map the points (-1,-1) and (1,1) to the corners of your viewing surface)

Subsequent clipping and scaling operations may be necessary to map the 2D plane onto any particular display media.

Diagram



To determine which screen x coordinate corresponds to a point at Ax,Az multiply the point coordinates by:

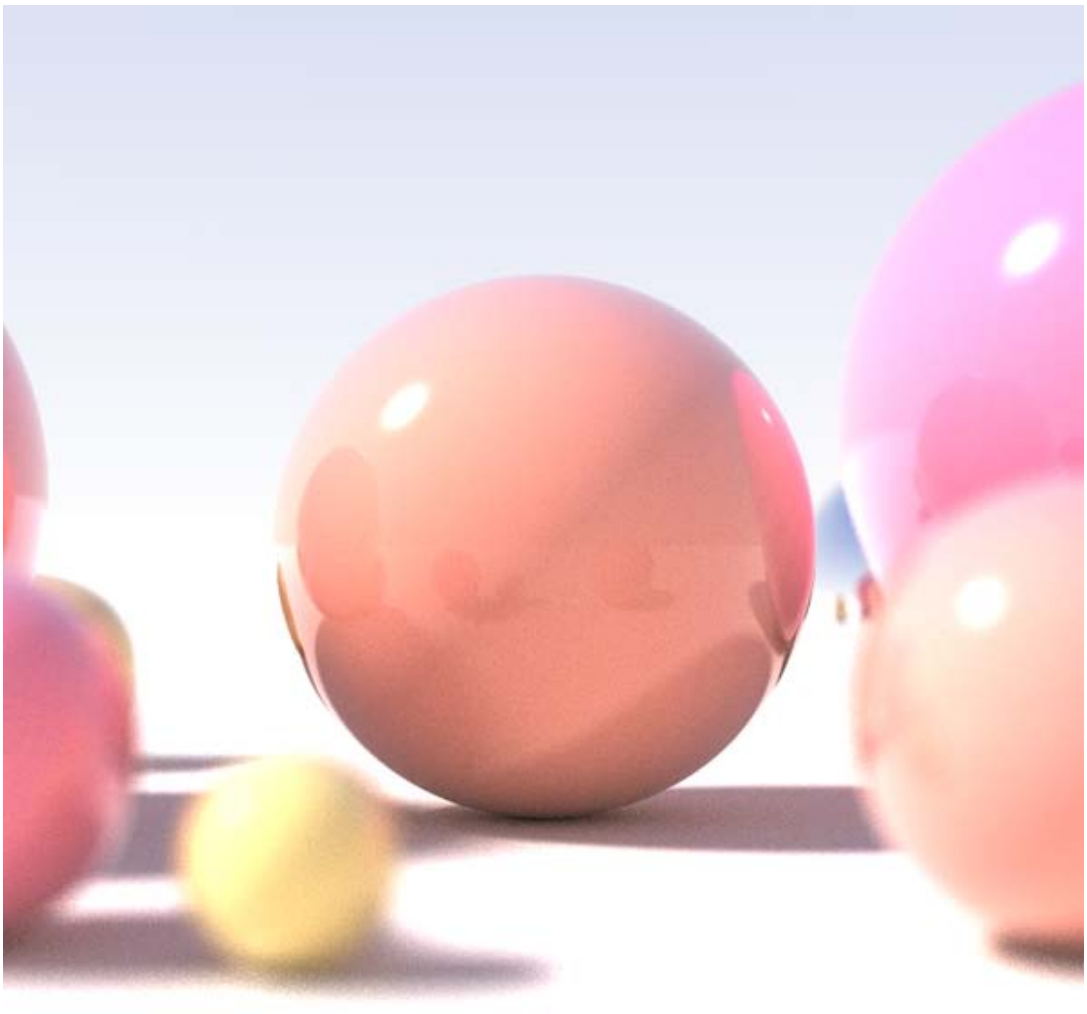
$$\text{screen x coordinate}(Bx) = \text{model x coordinate}(Ax) \times \frac{\text{distance from eye to screen}(Bz)}{\text{distance from eye to point}(Az)}$$

the same works for the screen y coordinate:

$$\text{screen y coordinate}(By) = \text{model y coordinate}(Ay) \times \frac{\text{distance from eye to screen}(Bz)}{\text{distance from eye to point}(Az)}$$

(where Ax and Ay are coordinates occupied by the object before the perspective transform)

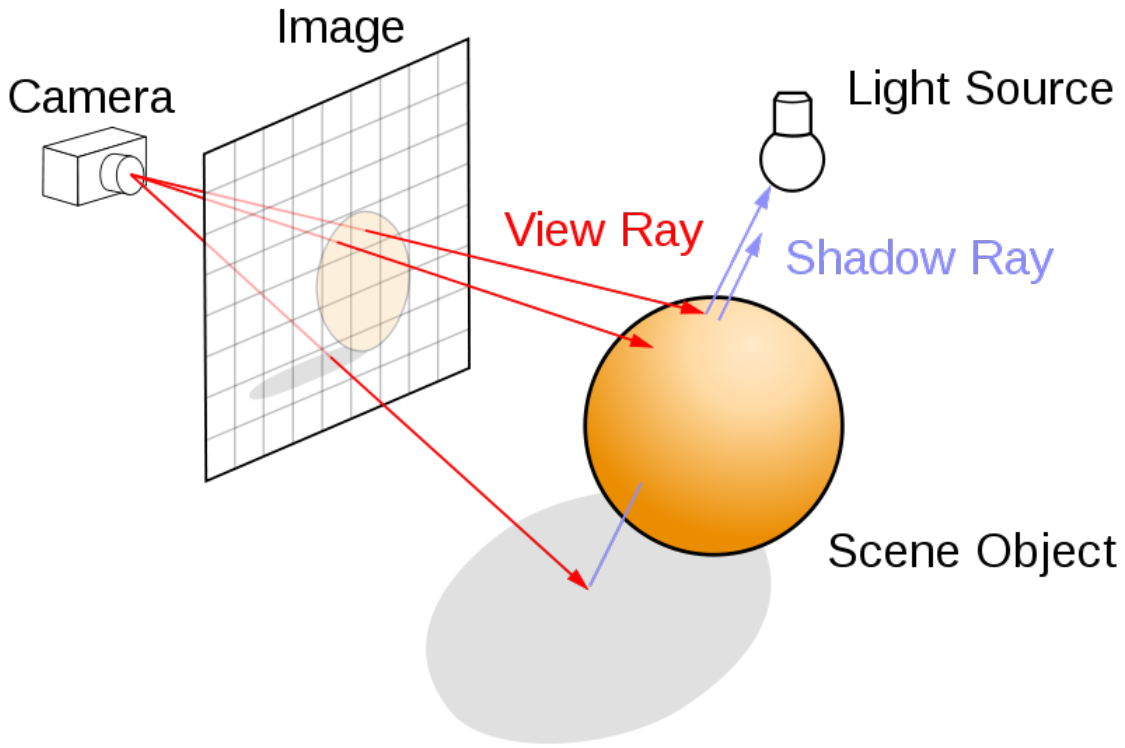
Ray tracing (graphics)



This recursive ray tracing of a sphere demonstrates the effects of shallow depth of field, area light sources, diffuse interreflection, ambient occlusion and fresnel reflection.

In computer graphics, **ray tracing** is a technique for generating an image by tracing the path of light through pixels in an image plane and simulating the effects of its encounters with virtual objects. The technique is capable of producing a very high degree of visual realism, usually higher than that of typical scanline rendering methods, but at a greater computational cost. This makes ray tracing best suited for applications where the image can be rendered slowly ahead of time, such as in still images and film and television special effects, and more poorly suited for real-time applications like computer games where speed is critical. Ray tracing is capable of simulating a wide variety of optical effects, such as reflection and refraction, scattering, and chromatic aberration.

Algorithm overview



The ray tracing algorithm builds an image by extending rays into a scene

Optical ray tracing describes a method for producing visual images constructed in 3D computer graphics environments, with more photorealism than either ray casting or scanline rendering techniques. It 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.

Scenes in raytracing are described mathematically by a programmer or by a visual artist (typically using intermediary tools). Scenes may also incorporate data from images and models captured by means such as digital photography.

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.

It may at first seem counterintuitive or "backwards" to send 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. A computer simulation that starts by casting rays from the light source is called Photon mapping, and it takes much longer than a comparable ray trace.

Therefore, the shortcut taken in raytracing is to presuppose that a given ray intersects the view frame. 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. The light intensity of this pixel is computed using a number of algorithms, which may include the classic rendering algorithm and may also incorporate techniques such as radiosity.

Detailed description of ray tracing computer algorithm and its genesis

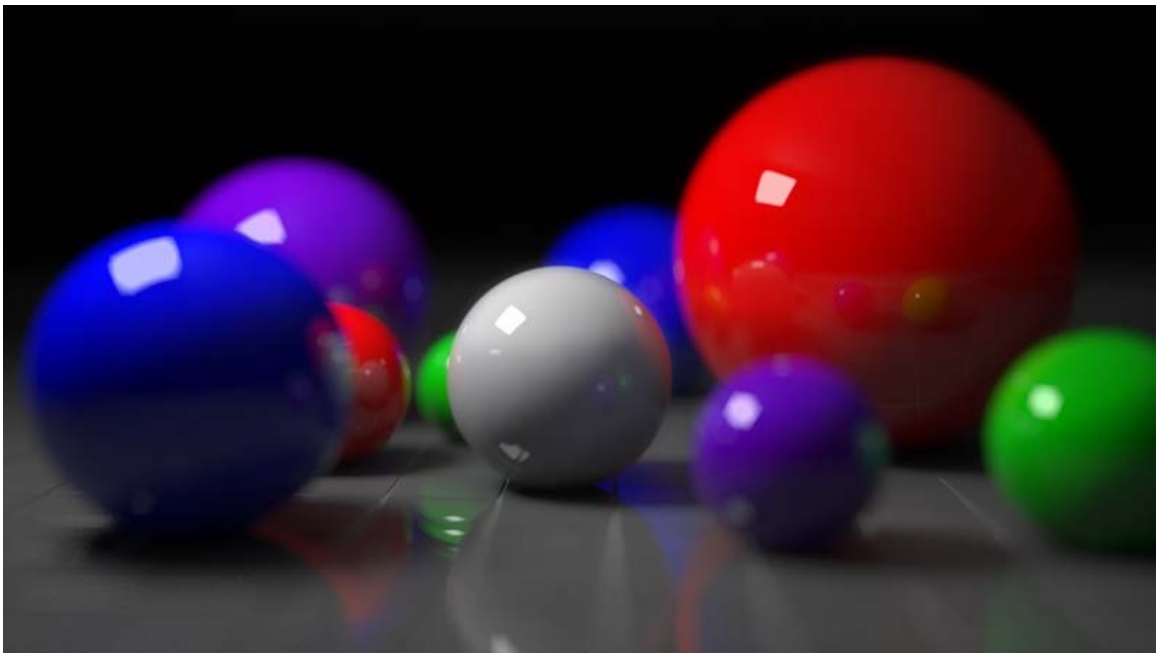
What happens in nature

Ray tracing can achieve a very high degree of visual realism

In nature, a light source emits a ray of light which travels, eventually, to a surface that interrupts its progress. One can think of this "ray" as a stream of photons traveling along the same path. In a perfect vacuum this ray will be a straight line (ignoring relativistic effects). In reality, any combination of four things might happen with this light ray: absorption, reflection, refraction and fluorescence. A surface may reflect all or part of the

light ray, in one or more directions. It might also absorb part of the light ray, resulting in a loss of intensity of the reflected and/or refracted light. 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 colour in a random direction, though this is rare enough that it can be discounted from most rendering applications. Between absorption, reflection, refraction and fluorescence, all of the incoming light must be accounted for, and no more. A surface cannot, for instance, reflect 66% of an incoming light ray, and refract 50%, since the two would add up to be 116%. From here, 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.

Ray casting algorithm

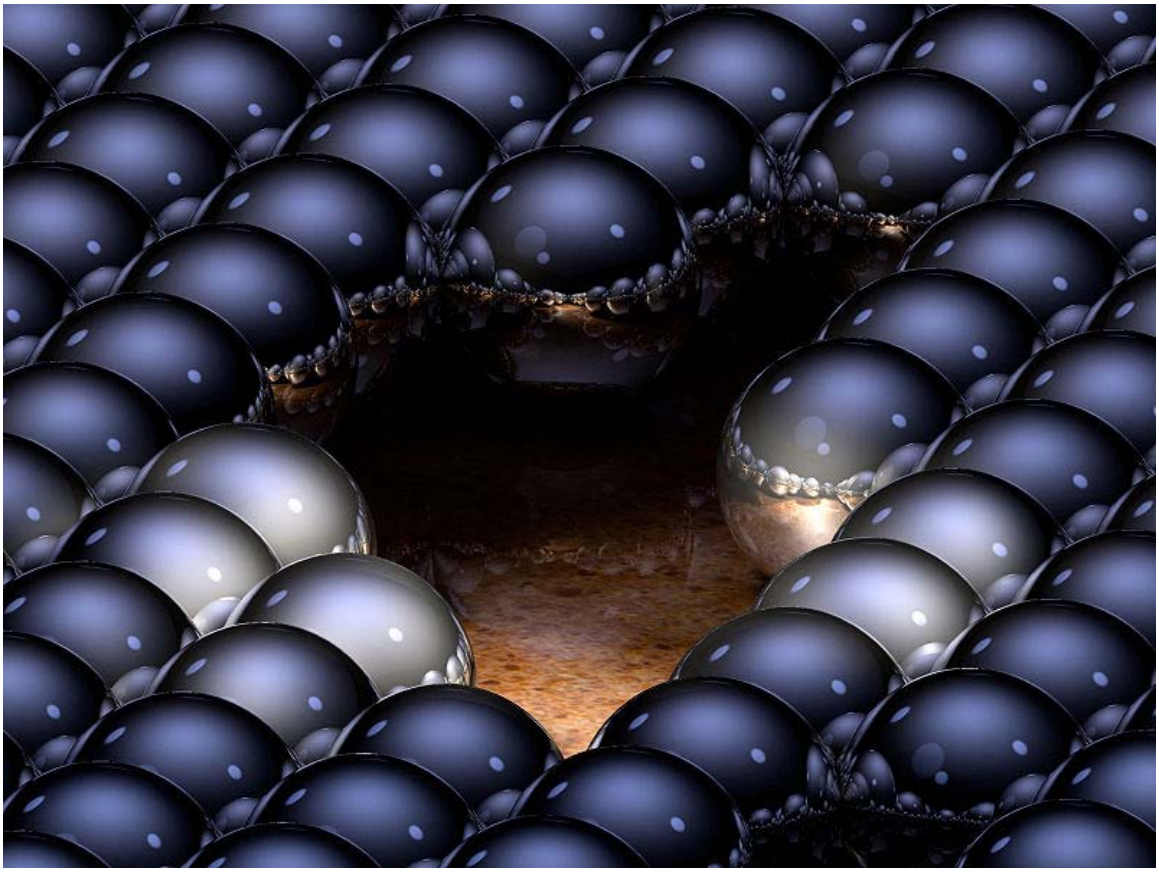


In addition to the high degree of realism, ray tracing can simulate the effects of a camera due to depth of field and aperture shape (in this case a hexagon).

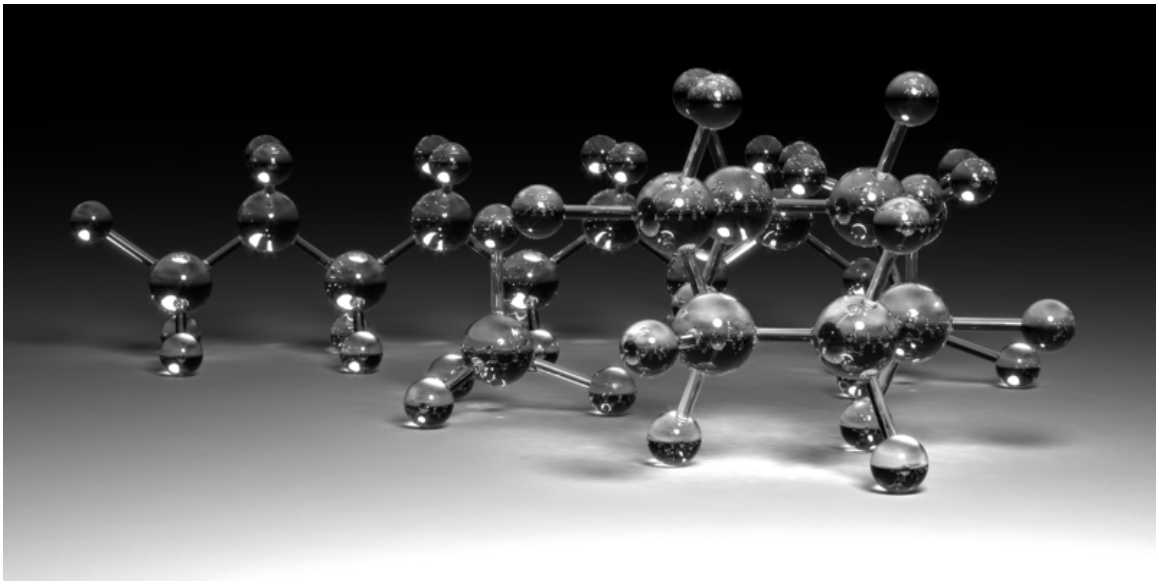
The first ray casting (versus ray tracing) algorithm used for rendering was presented by Arthur Appel in 1968. The idea behind ray casting is to shoot rays from the eye, one per pixel, and find the closest object blocking the path of that ray – think of an image as a screen-door, with each square in the screen being a pixel. This is then the object the eye normally sees through that pixel. Using the material properties and the effect of the lights in the scene, this algorithm can determine the shading of this object. The simplifying assumption is made that if a surface faces a light, the light will reach that surface and not be blocked or in shadow. The shading of the surface is computed using traditional 3D computer graphics shading models. One important advantage ray casting offered over

older scanline algorithms is its ability to easily deal with non-planar surfaces and solids, such as cones and spheres. If a mathematical surface can be intersected by a ray, it can be rendered using ray casting. Elaborate objects can be created by using solid modeling techniques and easily rendered.

Ray tracing algorithm



The number of reflections a “ray” can take and how it is affected each time it encounters a surface is all controlled via software settings during ray tracing. Here, each ray was allowed to reflect up to 16 times. Multiple “reflections of reflections” can thus be seen. *Created with Cobalt*



The number of refractions a “ray” can take and how it is affected each time it encounters a surface is all controlled via software settings during ray tracing. Here, each ray was allowed to refract and reflect up to 9 times. Fresnel reflections were used. Also note the caustics. *Created with Vray*

The next important research breakthrough came from Turner Whitted in 1979. Previous algorithms cast rays from the eye into the scene, but the rays were traced no further. Whitted continued the process. When a ray hits a surface, it could generate up to three new types of rays: reflection, refraction, and shadow. A reflected ray continues on in the mirror-reflection direction from a shiny surface. It is then intersected with objects in the scene; 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. To further avoid tracing all rays in a scene, a shadow ray is used to test if a surface is visible to a light. A ray hits a surface at some point. If the surface at this point faces a light, a ray (to the computer, a line segment) is traced between this intersection point and the light. If any opaque object is found in between the surface and the light, the surface is in shadow and so the light does not contribute to its shade. This new layer of ray calculation added more realism to ray traced images.

Advantages over other rendering methods

Ray tracing's popularity stems from its basis in a realistic simulation of lighting over other rendering methods (such as scanline rendering or ray casting). Effects such as reflections and shadows, which are difficult to simulate using other algorithms, are a natural result of the ray tracing algorithm. Relatively simple to implement yet yielding impressive visual results, ray tracing often represents a first foray into graphics programming. The computational independence of each ray makes ray tracing amenable to parallelization.

Disadvantages

A serious disadvantage of ray tracing is performance. 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 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 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, then, must be evaluated as an approximation to the equation, and in the case of ray tracing, it is not necessarily the most realistic. Other methods, including photon mapping, are based upon ray tracing for certain parts of the algorithm, yet give far better results.

Reversed direction of traversal of scene by the rays

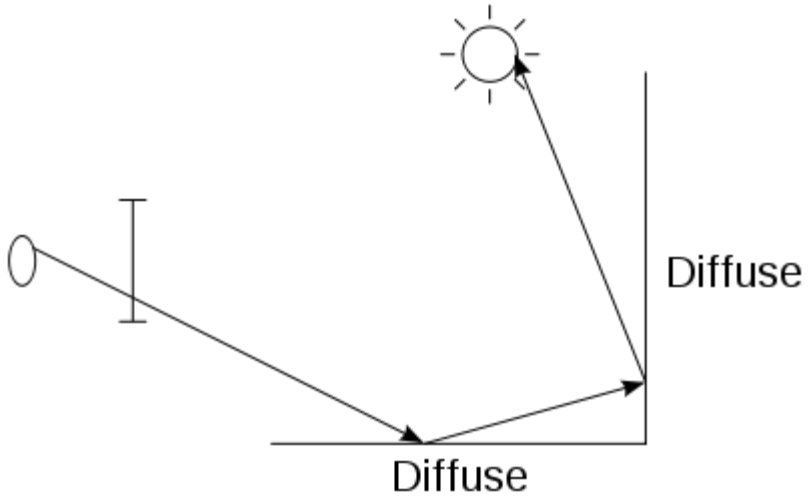
The process of shooting rays from the eye to the light source to render an image is sometimes called *backwards ray tracing*, since it is the opposite direction photons actually travel. However, there is confusion with this terminology. Early ray tracing was always done from the eye, and early researchers such as James Arvo used the term *backwards ray tracing* to mean shooting rays from the lights and gathering the results. Therefore it is clearer to distinguish *eye-based* versus *light-based* ray tracing.

While the direct illumination is generally best sampled using eye-based ray tracing, certain indirect effects can benefit from rays generated from the lights. Caustics are bright patterns caused by the focusing of light off a wide reflective region onto a narrow area of (near-)diffuse surface. An algorithm that casts rays directly from lights onto reflective objects, tracing their paths to the eye, will better sample this phenomenon. This integration of eye-based and light-based rays is often expressed as bidirectional path tracing, in which paths are traced from both the eye and lights, and the paths subsequently joined by a connecting ray after some length.

Photon mapping is another method that uses both light-based and eye-based ray tracing; in an initial pass, energetic photons are traced along rays from the light source so as to compute an estimate of radiant flux as a function of 3-dimensional space (the eponymous photon map itself). In a subsequent pass, rays are traced from the eye into the scene to determine the visible surfaces, and the photon map is used to estimate the illumination at the visible surface points. The advantage of photon mapping versus bidirectional path tracing is the ability to achieve significant reuse of photons, reducing computation, at the cost of statistical bias.

An additional problem occurs when light must pass through a very narrow aperture to illuminate the scene (consider a darkened room, with a door slightly ajar leading to a

brightly-lit room), or a scene in which most points do not have direct line-of-sight to any light source (such as with ceiling-directed light fixtures or torchieres). In such cases, only a very small subset of paths will transport energy; Metropolis light transport is a method which begins with a random search of the path space, and when energetic paths are found, reuses this information by exploring the nearby space of rays.



To the right is an image showing a simple example of a path of rays recursively generated from the camera (or eye) to the light source using the above algorithm. A diffuse surface reflects light in all directions.

First, a ray is created at an eyepoint and traced through a pixel and into the scene, where it hits a diffuse surface. From that surface the algorithm recursively generates a reflection ray, which is traced through the scene, where it hits another diffuse surface. Finally, another reflection ray is generated and traced through the scene, where it hits the light source and is absorbed. The color of the pixel now depends on the colors of the first and second diffuse surface and the color of the light emitted from the light source. For example if the light source emitted white light and the two diffuse surfaces were blue, then the resulting color of the pixel is blue.

In real time

The first implementation of a "real-time" ray-tracer was credited at the 2005 SIGGRAPH computer graphics conference as the REMRT/RT tools developed in 1986 by Mike Muuss for the BRL-CAD solid modeling system. Initially published in 1987 at USENIX, the BRL-CAD ray-tracer is the first known implementation of a parallel network distributed ray-tracing system that achieved several frames per second in rendering performance. This performance was attained by means of the highly-optimized yet platform independent LIBRT ray-tracing engine in BRL-CAD and by using solid implicit CSG geometry on several shared memory parallel machines over a commodity network. BRL-CAD's ray-tracer, including REMRT/RT tools, continue to be available and developed today as Open source software.

Since then, there have been considerable efforts and research towards implementing ray tracing in real time speeds for a variety of purposes on stand-alone desktop configurations. These purposes include interactive 3D graphics applications such as demoscene productions, computer and video games, and image rendering. Some real-time software 3D engines based on ray tracing have been developed by hobbyist demo programmers since the late 1990s.

The OpenRT project includes a highly-optimized software core for ray tracing along with an OpenGL-like API in order to offer an alternative to the current rasterisation based approach for interactive 3D graphics. Ray tracing hardware, such as the experimental Ray Processing Unit developed at the Saarland University, has been designed to accelerate some of the computationally intensive operations of ray tracing. On March 16, 2007, the University of Saarland revealed an implementation of a high-performance ray tracing engine that allowed computer games to be rendered via ray tracing without intensive resource usage.

On June 12, 2008 Intel demonstrated a special version of Enemy Territory: Quake Wars, titled Quake Wars: Ray Traced, using ray tracing for rendering, running in basic HD (720p) resolution. ETQW operated at 14-29 frames per second. The demonstration ran on a 16-core (4 socket, 4 core) Tigerton system running at 2.93 GHz.

At SIGGRAPH 2009, Nvidia announced OptiX, an API for real-time ray tracing on Nvidia GPUs. The API exposes seven programmable entry points within the ray tracing pipeline, allowing for custom cameras, ray-primitive intersections, shaders, shadowing, etc.

Example

As a demonstration of the principles involved in raytracing, let us consider how one would find the intersection between a ray and a sphere. In vector notation, the equation of a sphere with center $\mathbf{c}$ and radius $\mathbf{r}$ is

$$\|\mathbf{x} - \mathbf{c}\|^2 = r^2.$$

Any point on a ray starting from point $\mathbf{s}$ with direction $\mathbf{d}$ (here $\mathbf{d}$ is a unit vector) can be written as

$$\mathbf{x} = \mathbf{s} + t\mathbf{d},$$

where *t* is its distance between $\mathbf{x}$ and $\mathbf{s}$. In our problem, we know $\mathbf{c}$, $\mathbf{r}$, $\mathbf{s}$ (e.g. the position of a light source) and $\mathbf{d}$, and we need to find *t*. Therefore, we substitute for $\mathbf{x}$:

$$\|\mathbf{s} + t\mathbf{d} - \mathbf{c}\|^2 = r^2.$$

Let $\mathbf{v} \stackrel{\text{def}}{=} \mathbf{s} - \mathbf{c}$ for simplicity; then

$$\|\mathbf{v} + t\mathbf{d}\|^2 = r^2$$
$$\mathbf{v}^2 + t^2\mathbf{d}^2 + 2\mathbf{v} \cdot t\mathbf{d} = r^2$$
$$(\mathbf{d}^2)t^2 + (2\mathbf{v} \cdot \mathbf{d})t + (\mathbf{v}^2 - r^2) = 0.$$

Knowing that d is a unit vector allows us this minor simplification:

$$t^2 + (2\mathbf{v} \cdot \mathbf{d})t + (\mathbf{v}^2 - r^2) = 0.$$

This quadratic equation has solutions

$$t = \frac{-(2\mathbf{v} \cdot \mathbf{d}) \pm \sqrt{(2\mathbf{v} \cdot \mathbf{d})^2 - 4(\mathbf{v}^2 - r^2)}}{2} = -(\mathbf{v} \cdot \mathbf{d}) \pm \sqrt{(\mathbf{v} \cdot \mathbf{d})^2 - (\mathbf{v}^2 - r^2)}.$$

The two values of *t* found by solving this equation are the two ones such that $\mathbf{s} + t\mathbf{d}$ are the points where the ray intersects the sphere.

If one (or both) of them are negative, then the intersections do not lie on the ray but in the opposite half-line (i.e. the one starting from $\mathbf{s}$ with opposite direction).

If the quantity under the square root (the discriminant) is negative, then the ray does not intersect the sphere.

Let us suppose now that there is at least a positive solution, and let *t* be the minimal one. In addition, let us suppose that the sphere is the nearest object on our scene intersecting our ray, and that it is made of a reflective material. We need to find in which direction the light ray is reflected. The laws of reflection state that the angle of reflection is equal and opposite to the angle of incidence between the incident ray and the normal to the sphere.

The normal to the sphere is simply

$$\mathbf{n} = \frac{\mathbf{y} - \mathbf{c}}{\|\mathbf{y} - \mathbf{c}\|},$$

where $\mathbf{y} = \mathbf{s} + t\mathbf{d}$ is the intersection point found before. The reflection direction can be found by a reflection of $\mathbf{d}$ with respect to $\mathbf{n}$, that is

$$\mathbf{r} = \mathbf{d} - 2(\mathbf{n} \cdot \mathbf{d})\mathbf{n}.$$

Thus the reflected ray has equation

$$\mathbf{x} = \mathbf{y} + u\mathbf{r}.$$

Now we only need to compute the intersection of the latter ray with our field of view, to get the pixel which our reflected light ray will hit. Lastly, this pixel is set to an

appropriate color, taking into account how the color of the original light source and the one of the sphere are combined by the reflection.

This is merely the math behind the line–sphere intersection and the subsequent determination of the colour of the pixel being calculated. There is, of course, far more to the general process of raytracing, but this demonstrates an example of the algorithms used.

Chapter 7

Rendering (Computer Graphics)

An image created by using POV-Ray 3.6.

Rendering is the process of generating an image from a model (or models in what collectively could be called a *scene* file), by means of computer programs. A scene file contains objects in a strictly defined language or data structure; it would contain geometry, viewpoint, texture, lighting, and shading information as a description of 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" may be by analogy with an "artist's rendering" of a scene. Though the technical details of rendering methods vary, the general challenges to overcome in

producing a 2D image from a 3D representation stored in a scene file are outlined as the graphics pipeline along a rendering device, such as a GPU. A GPU is a purpose-built device able to assist a CPU in performing complex rendering calculations. If a scene is to look relatively realistic and predictable under virtual lighting, the rendering software should solve the rendering equation. The rendering equation doesn't account for all lighting phenomena, but is a general lighting model for computer-generated imagery. 'Rendering' is also used to describe the process of calculating effects in a video editing file to produce final video output.

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

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

In the case of 3D graphics, rendering may be done slowly, as in pre-rendering, or in real time. Pre-rendering is a computationally intensive process that is typically used for movie creation, while real-time rendering is often done for 3D video games which rely on the use of graphics cards with 3D hardware accelerators.

Usage

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

Image rendered with computer aided design

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.

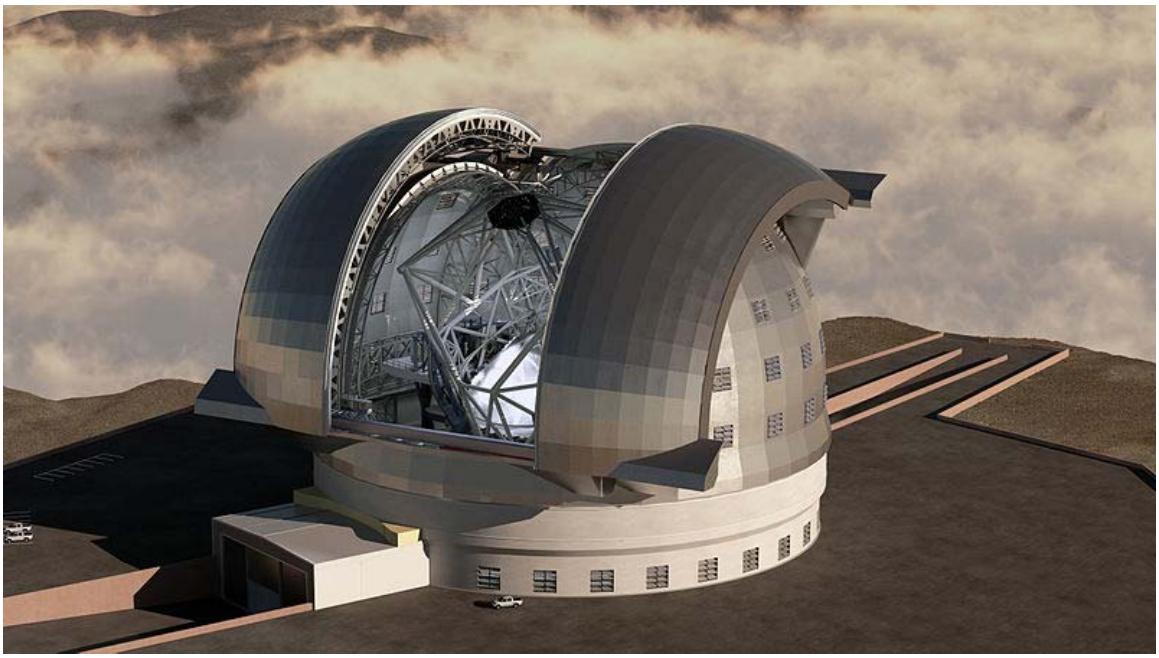
Tracing every particle of light in a scene is nearly always completely impractical and would take a stupendous amount of time. Even tracing a portion large enough to produce an image takes an inordinate amount of time if the sampling is not intelligently restricted.

Therefore, four loose families of more-efficient light transport modelling 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; and 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 slower. The fourth type of light transport techique, 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 rasterisation



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 3D rendering, 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 loops 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 aren't 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.

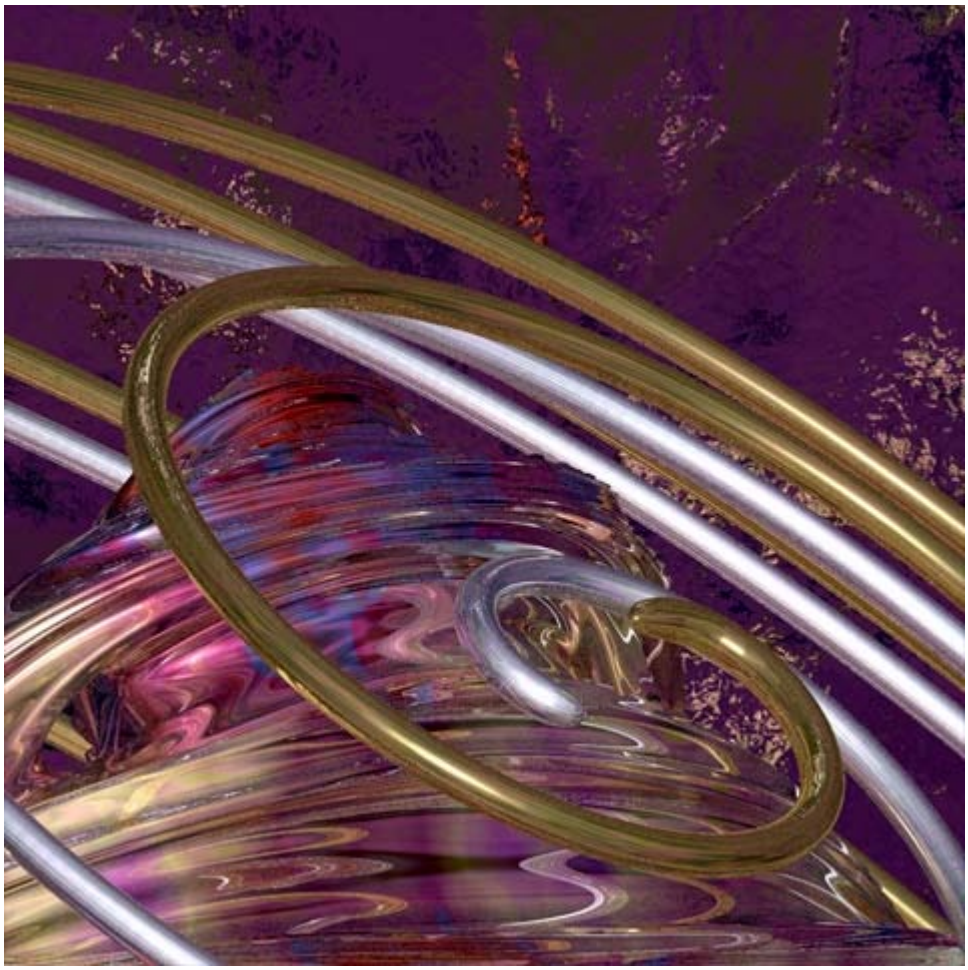
Ray casting

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 colour 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.

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.

Raycasting is primarily used for realtime simulations, such as those used in 3D computer games and cartoon animations, where detail is not important, or where it is more efficient to manually fake the details in order to obtain better performance in the computational stage. This is usually the case when a large number of frames need to be animated. The resulting surfaces have a characteristic 'flat' appearance when no additional tricks are used, as if objects in the scene were all painted with matte finish.

Ray tracing



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.

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.

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.

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.

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 the 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 algorithim, 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 graphic 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, 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

Optimizations used by an artist when a scene is being developed

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.

Common optimizations for real time rendering

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.

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

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:

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

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

Geometric optics

Rendering is practically exclusively concerned with the particle aspect of light physics — known as geometric 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 an almost infinite 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 the RenderMan shading language designed at Pixar. (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 IPR and hardware rendering/shading.

Chapter 8

Reflection (Computer Graphics)

Ray traced model demonstrating specular reflection

Reflection in computer graphics is used to emulate reflective objects like mirrors and shiny surfaces.

Reflection is accomplished in a ray trace renderer by following a ray from the eye to the mirror and then calculating where it bounces from, and continuing the process until no surface is found, or a non-reflective surface is found. Reflection on a shiny surface like wood or tile can add to the photorealistic effects of a 3D rendering.

- **Polished** - A Polished Reflection is an undisturbed reflection, like a mirror or chrome.

- **Blurry** - A Blurry Reflection means that tiny random bumps on the surface of the material cause the reflection to be blurry.
- **Metallic** - A reflection is Metallic if the highlights and reflections retain the color of the reflective object.
- **Glossy** - This term can be misused. Sometimes it is a setting which is the opposite of Blurry. (When "Glossiness" has a low value, the reflection is blurry.) However, some people use the term "Glossy Reflection" as a synonym for "Blurred Reflection." Glossy used in this context means that the reflection is actually blurred.

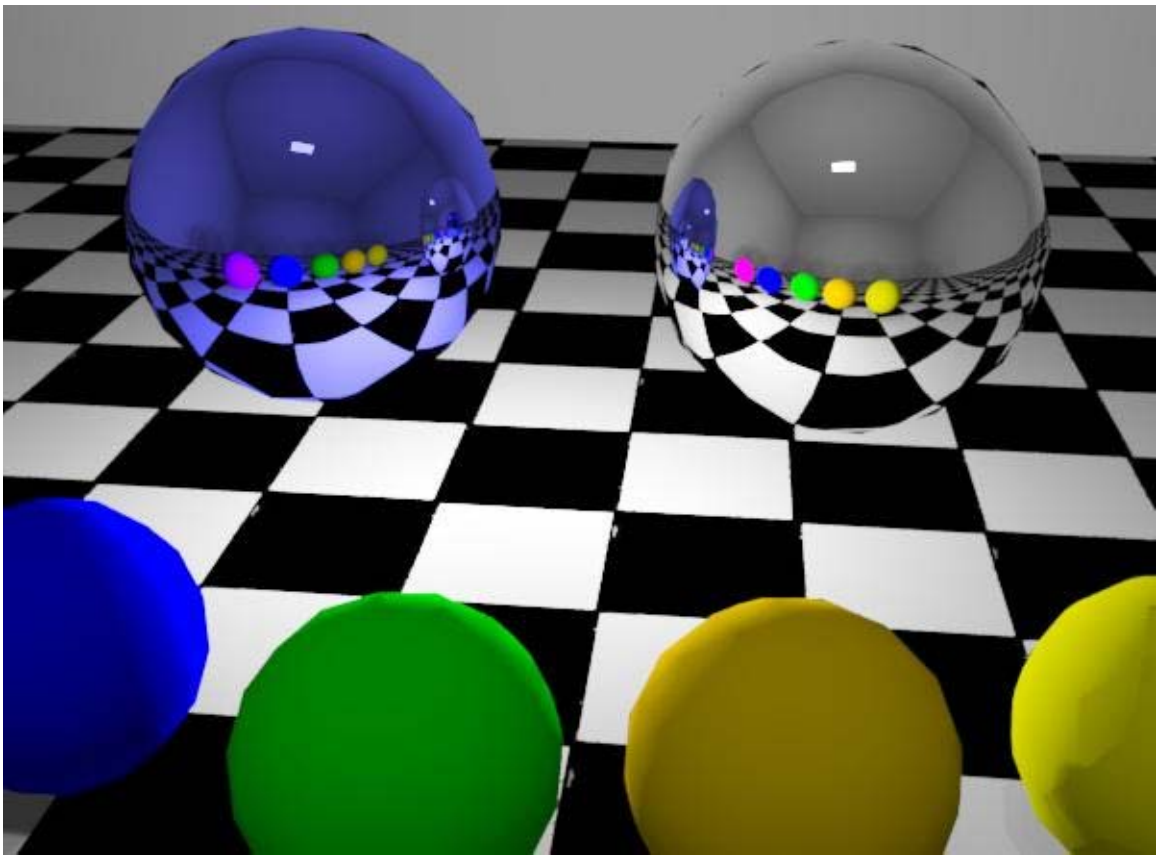
Examples

Polished or Mirror reflection

Mirror on wall rendered with 100% reflection

Mirrors are usually almost 100% reflective.

Metallic Reflection

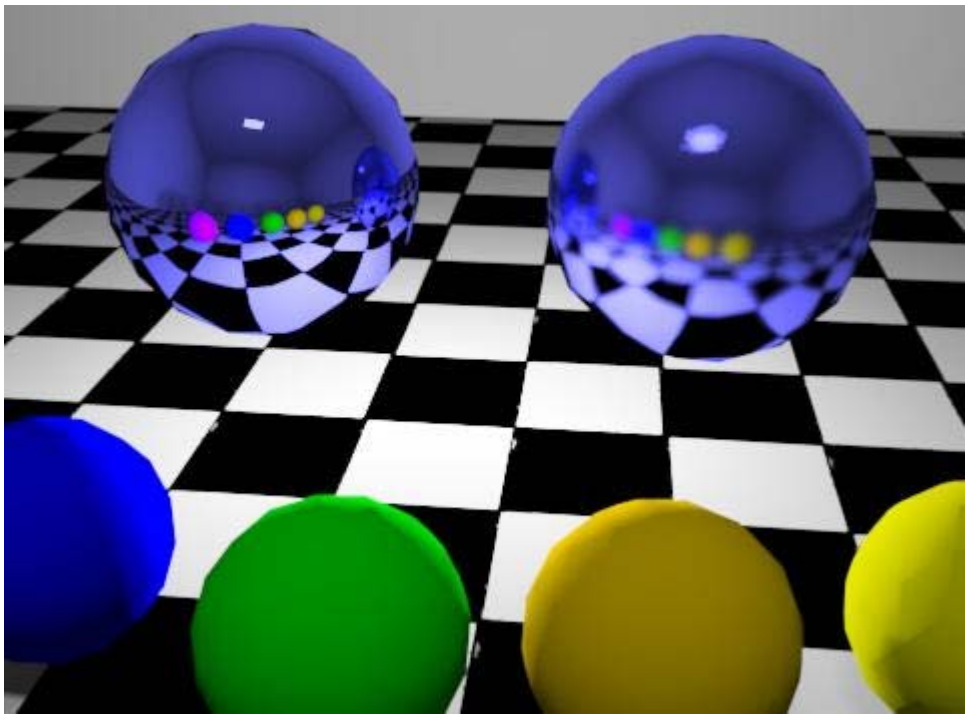


The large sphere on the left is blue with its reflection marked as metallic. The large sphere on the right is the same color but does not have the metallic property selected.

Normal, (non metallic), objects reflect light and colors in the original color of the object being reflected.

Metallic objects reflect lights and colors altered by the color of the metallic object itself.

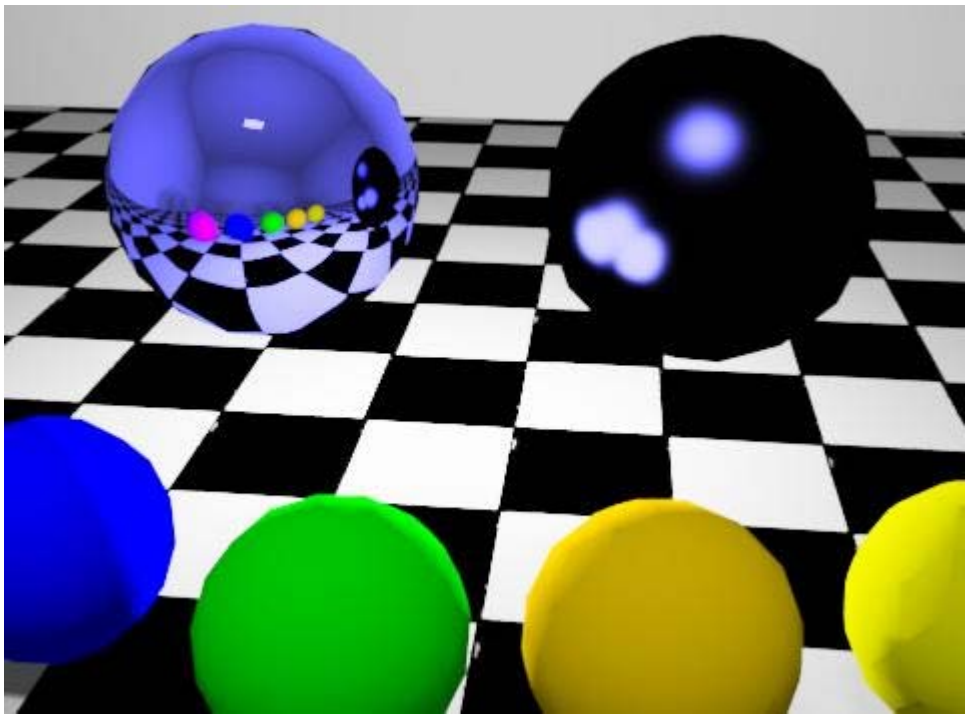
Blurry Reflection



The large sphere on the left has sharpness set to 100%. The sphere on the right has sharpness set to 50% which creates a blurry reflection.

Many materials are imperfect reflectors, where the reflections are blurred to various degrees due to surface roughness that scatters the rays of the reflections.

Glossy Reflection



The sphere on the left has normal, metallic reflection. The sphere on the right has the same parameters, except that the reflection is marked as "glossy".

Fully glossy reflection, shows highlights from light sources, but does not show a clear reflection from objects.

Chapter 9

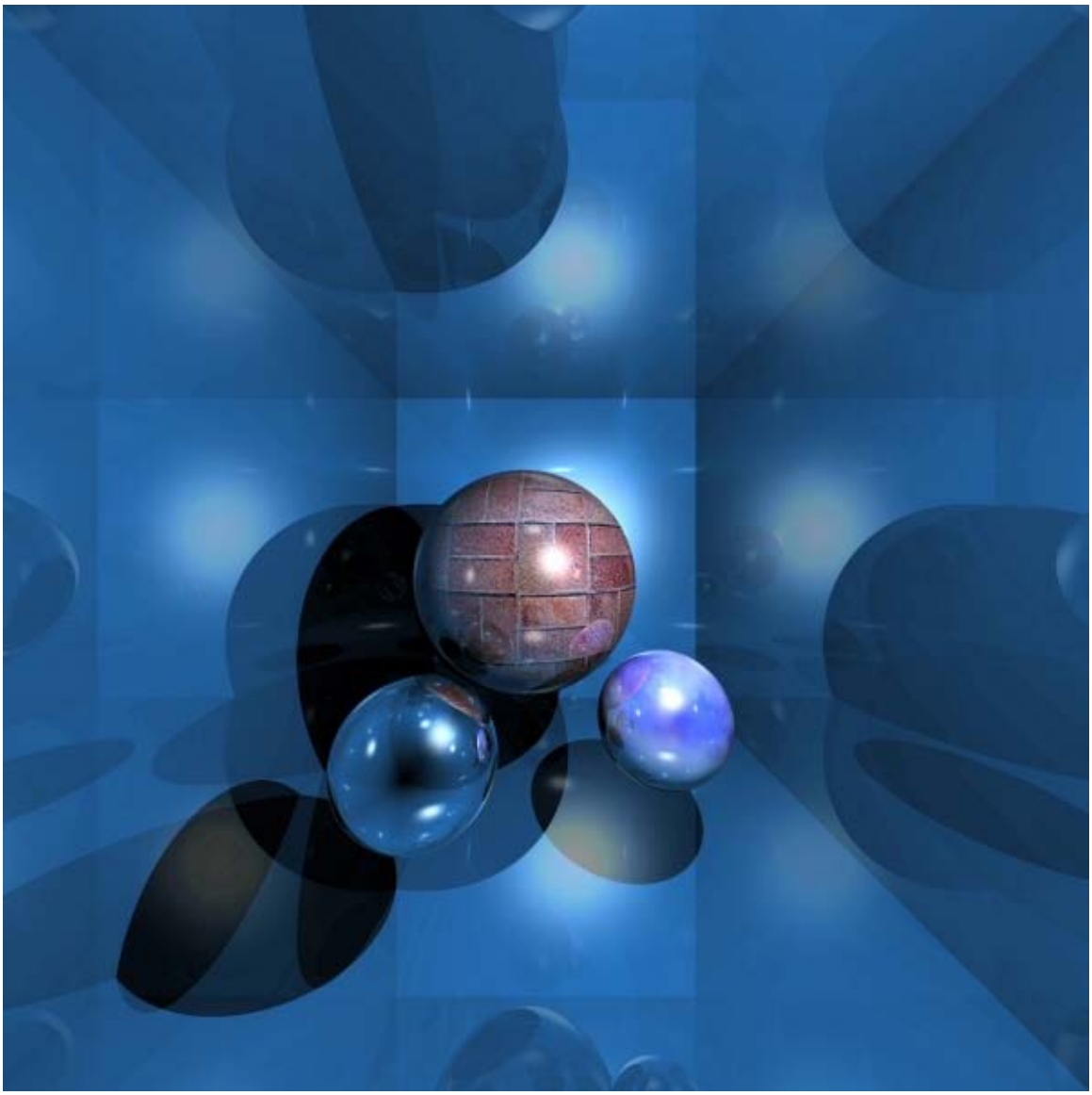
3D Rendering

3D rendering is the 3D computer graphics process of automatically converting 3D wire frame models into 2D images with 3D photorealistic effects on a computer.

Rendering methods

Rendering is the final process of creating the actual 2D image or animation from the prepared scene. This can be compared to taking a photo or filming the scene after the setup is finished in real life. Several different, and often specialized, rendering methods have been developed. These range from the distinctly non-realistic wireframe rendering through polygon-based rendering, to more advanced techniques such as: scanline rendering, ray tracing, or radiosity. Rendering may take from fractions of a second to days for a single image/frame. In general, different methods are better suited for either photo-realistic rendering, or real-time rendering.

Real-time



An example of a ray-traced image that typically takes seconds or minutes to render

Rendering for interactive media, such as games and simulations, is calculated and displayed in real time, at rates of approximately 20 to 120 frames per second. In real-time rendering, the goal is to show as much information as possible as the eye can process in a 30th of a second (or one frame, in the case of 30 frame-per-second animation). The goal here is primarily speed and not photo-realism. In fact, here exploitations are made in the way the eye 'perceives' the world, and as a result the final image presented is not necessarily that of the real-world, but one close enough for the human eye to tolerate. Rendering software may simulate such visual effects as lens flares, depth of field or motion blur. These are attempts to simulate visual phenomena resulting from the optical characteristics of cameras and of the human eye. These effects can lend an element of realism to a scene, even if the effect is merely a simulated artifact of a camera. This is the

basic method employed in games, interactive worlds and VRML. The rapid increase in computer processing power has allowed a progressively higher degree of realism even for real-time rendering, including techniques such as HDR rendering. Real-time rendering is often polygonal and aided by the computer's GPU.

Non real-time

Computer-generated image created by Gilles Tran

Animations for non-interactive media, such as feature films and video, are rendered much more slowly. Non-real time rendering enables the leveraging of limited processing power in order to obtain higher image quality. Rendering times for individual frames may vary from a few seconds to several days for complex scenes. Rendered frames are stored on a hard disk then can be transferred to other media such as motion picture film or optical disk. These frames are then displayed sequentially at high frame rates, typically 24, 25, or 30 frames per second, to achieve the illusion of movement.

When the goal is photo-realism, techniques such as ray tracing or radiosity are employed. This is the basic method employed in digital media and artistic works. Techniques have been developed for the purpose of simulating other naturally-occurring effects, such as the interaction of light with various forms of matter. Examples of such techniques include particle systems (which can simulate rain, smoke, or fire), volumetric sampling (to simulate fog, dust and other spatial atmospheric effects), caustics (to simulate light

focusing by uneven light-refracting surfaces, such as the light ripples seen on the bottom of a swimming pool), and subsurface scattering (to simulate light reflecting inside the volumes of solid objects such as human skin).

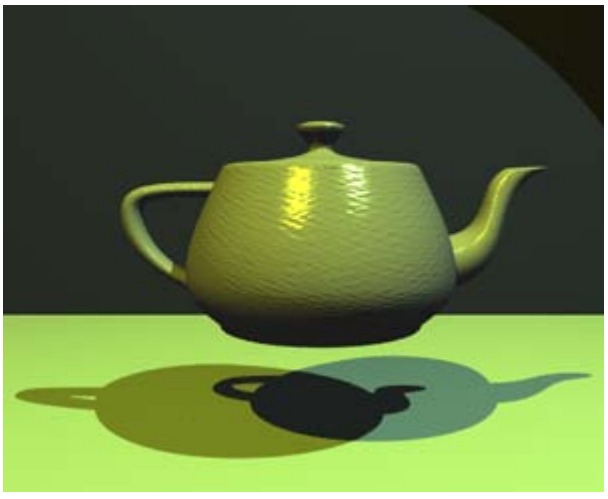
The rendering process is computationally expensive, given the complex variety of physical processes being simulated. Computer processing power has increased rapidly over the years, allowing for a progressively higher degree of realistic rendering. Film studios that produce computer-generated animations typically make use of a render farm to generate images in a timely manner. However, falling hardware costs mean that it is entirely possible to create small amounts of 3D animation on a home computer system. The output of the renderer is often used as only one small part of a completed motion-picture scene. Many layers of material may be rendered separately and integrated into the final shot using compositing software.

Reflection and shading models

Models of *reflection/scattering* and *shading* are used to describe the appearance of a surface. Although these issues may seem like problems all on their own, they are studied almost exclusively within the context of rendering. Modern 3D computer graphics rely heavily on a simplified reflection model called *Phong reflection model* (not to be confused with Phong shading). In refraction of light, an important concept is the refractive index. In most 3D programming implementations, the term for this value is "index of refraction," usually abbreviated "IOR." Shading can be broken down into two orthogonal issues, which are often studied independently:

- **Reflection/Scattering** - How light interacts with the surface *at a given point*
- **Shading** - How material properties vary across the surface

Reflection



The Utah teapot

Reflection or scattering is the relationship between incoming and outgoing illumination at a given point. Descriptions of scattering are usually given in terms of a bidirectional scattering distribution function or BSDF. Popular reflection rendering techniques in 3D computer graphics include:

- Flat shading: A technique that shades each polygon of an object based on the polygon's "normal" and the position and intensity of a light source.
- Gouraud shading: Invented by H. Gouraud in 1971, a fast and resource-conscious vertex shading technique used to simulate smoothly shaded surfaces.
- Texture mapping: A technique for simulating a large amount of surface detail by mapping images (textures) onto polygons.
- Phong shading: Invented by Bui Tuong Phong, used to simulate specular highlights and smooth shaded surfaces.
- Bump mapping: Invented by Jim Blinn, a normal-perturbation technique used to simulate wrinkled surfaces.
- Cel shading: A technique used to imitate the look of hand-drawn animation.

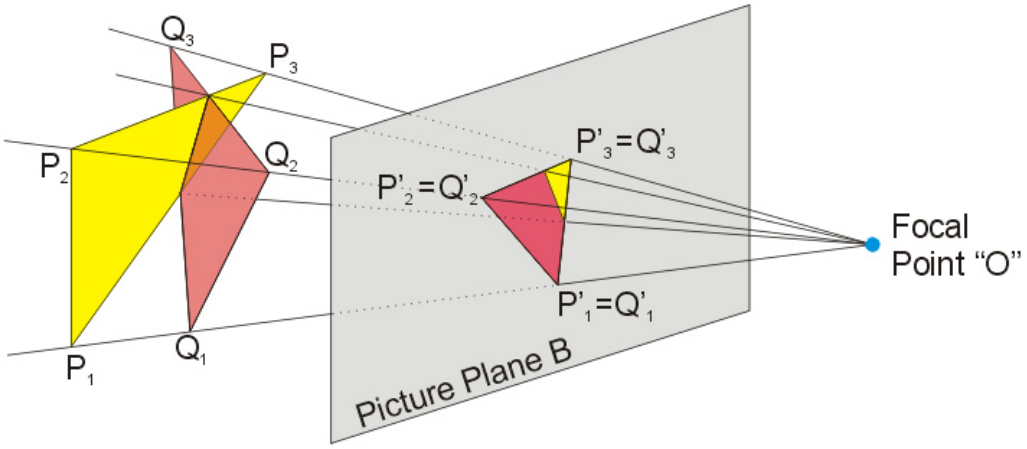
Shading

Shading addresses how different types of scattering are distributed across the surface (i.e., which scattering function applies where). Descriptions of this kind are typically expressed with a program called a shader. (Note that there is some confusion since the word "shader" is sometimes used for programs that describe local *geometric* variation.) A simple example of shading is texture mapping, which uses an image to specify the diffuse color at each point on a surface, giving it more apparent detail.

Transport

Transport describes how illumination in a scene gets from one place to another. Visibility is a major component of light transport.

Projection



Perspective Projection

The shaded three-dimensional objects must be flattened so that the display device - namely a monitor - can display it in only two dimensions, this process is called 3D projection. This is done using projection and, for most applications, perspective projection. The basic idea behind perspective projection is that objects that are further away are made smaller in relation to those that are closer to the eye. Programs produce perspective by multiplying a dilation constant raised to the power of the negative of the distance from the observer. A dilation constant of one means that there is no perspective. High dilation constants can cause a "fish-eye" effect in which image distortion begins to occur. Orthographic projection is used mainly in CAD or CAM applications where scientific modeling requires precise measurements and preservation of the third dimension.