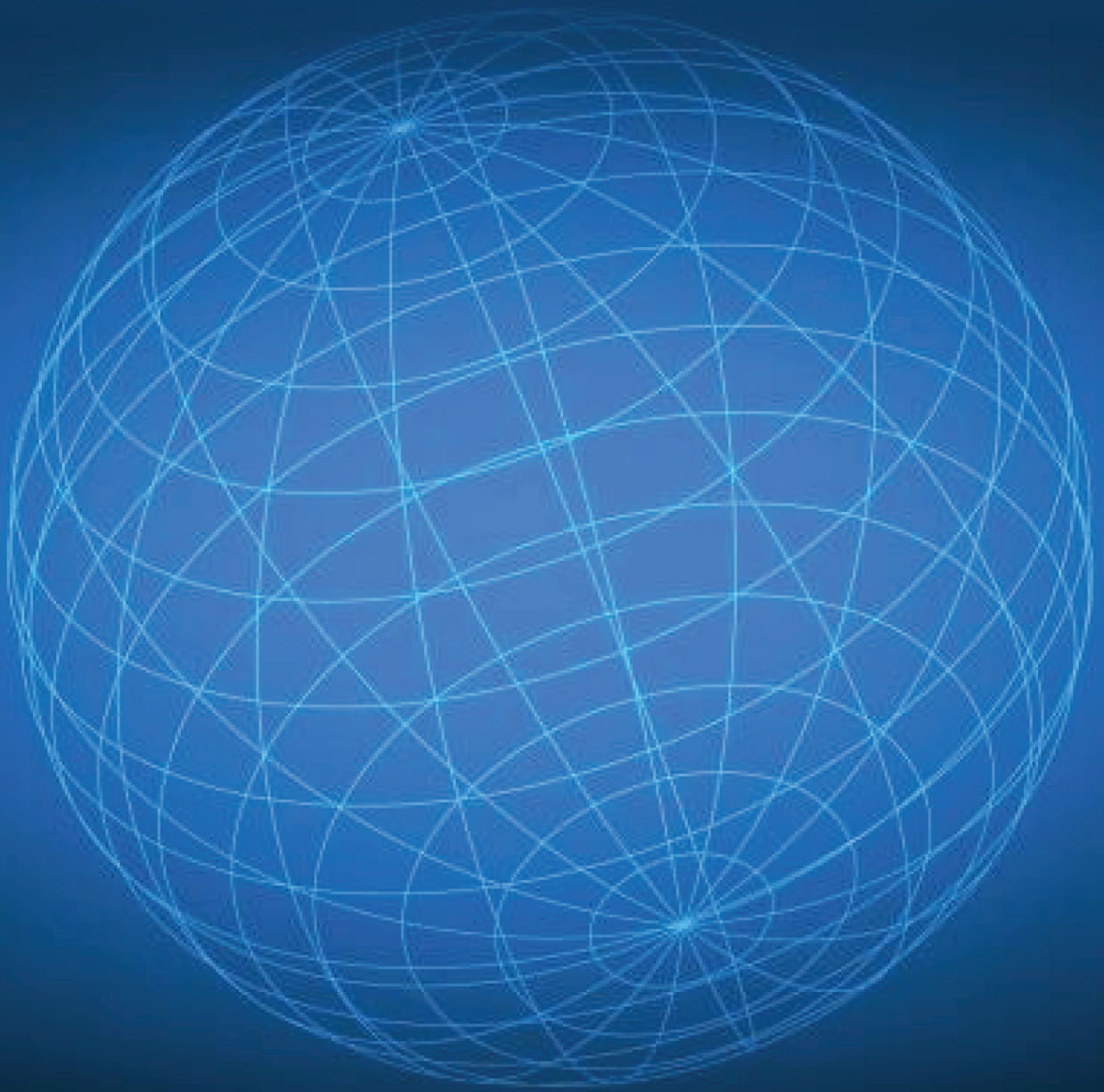
Advanced Computer Graphics

Horace Howard



ADVANCED COMPUTER GRAPHICS

ADVANCED COMPUTER GRAPHICS

Horace Howard

BIBLIOTEX
Digital Library

Advanced Computer Graphics
by Horace Howard

www.bibliotex.com

Ebook ISBN: 9781984664136

Published by:
Bibliotex
Canada
Website: www.bibliotex.com

Contents

1

Background of Computer Graphics

Today there are very few aspects of our lives not affected by computers. Practically every cash or monetary transaction that takes place daily involves a computer. In many cases, the same is true of computer graphics. Whether you see them on television, in newspapers, in weather reports or while at the doctor's surgery, computer images are all around you.

"A picture is worth a thousand words" is a well-known saying, and highlights the advantages and benefits of the visual presentation of our data. We are able to obtain a comprehensive overall view of our data and also study features and areas of particular interest.

A well-chosen graph is able to transform a complex table of numbers into meaningful results. Such graphs are used to illustrate papers, reports, and theses, as well as providing

the basis for presentation material in the form of slides and overhead transparencies. A range of tools and facilities are available to enable users to visualise their data, and this document provides a brief summary and overview. Computer graphics are used in many disciplines and subjects but for the purpose of this document, we will split the topic of computer graphics into the following fields:

CHARTING

One of the prime uses for graphical software at the University is to produce graphs and charts. Everyone has data of one kind or another, whether on paper, in the computer, or just in the mind. We often need to know the significance and properties of the data, or to be able to compare different parts of it against other data sets.

One of the simplest aspects of data display is the production of charts. This is where you would want to put your data into a graphical form to show relationships and comparisons between sets of values.

There may be a number of reasons why you would want to put your data into a chart:

- To illustrate differences between different sets of data,
- To show trends between two variables,
- To show patterns of behaviour in one variable.

There are basically two broad areas of graphs:

- Presentation charts and graphs of the kind used to illustrate a few principal points. We see these on news

and current affairs programmes on television. A bar chart or a pie chart is used to indicate results of data obtained so far and the general trends. They are often liberally decorated with bright colours to increase their visual appeal and attractiveness to the viewers and to hold their attention. They are used for visual impact and getting a simple point over clearly and effectively.

- Scientific charts and graphs are more concerned with ensuring that the detail in the data is represented accurately and faithfully. We may have some results obtained from experimental measurements and wish to display them. We may want to compare the results from the data measurements with the results we would expect according to a particular theoretical model. We may want to draw a curve through the data points (*i.e.* interpolate the data) and display this along with the original points.

The aims of the two are different, and so the facilities you will want from your charting package will also be different. Presentation charting has more to do with impressive presentation graphics where the aim is to put a salient point across to an audience. As a result the priority with this sort of charting is not always accuracy of representation. You want charts with strong colours, an impressive look and special effects. The effect of a presentation can be enhanced by using 3D graphs, adding pictures to the graph, or using

pictograms. These sorts of charts are rarely produced in isolation but as part of a general presentation. Therefore, some presentation packages also have their own charting module for this purpose. Word and PowerPoint use a module called Microsoft Graph and Excel's charting module has some very powerful presentation graphics features. Origin and Gsharp, both dedicated charting packages, also provide professional presentation charting facilities on the PC systems. Gsharp is also available on the UNIX systems.

In scientific charting you want to display data as accurately as possible in order to analyse it graphically or demonstrate clearly your comparisons and results. As this sort of charting is done mainly for analysis, it is rarely an isolated activity but is often done alongside detailed numerical analysis of your data. Two of the most powerful charting packages available are Origin on the PC network and Gsharp on the PC and UNIX systems. Also, many numerical analysis packages have their own charting modules integrated with the rest of the package.

It is clear that your choice of charting programme will depend very much on what purpose you want the chart to fulfil, and also what other programmes you are already using. On the whole, if you are already using a programme that has its own charting module, use that. The table below gives some rough guidelines on your choice of charting PC package, with the packages increasing in facilities and complexity going down the table.

Requirement	*Choice*
Simple bar, column, line or pie charts to integrate in a word processor	Microsoft Graph in Word, Charting Module in Excel
Charts for use in a presentation	Microsoft Graph in Word or PowerPoint, Charting Module in Excel, Origin
Raw data requiring good quality scientific charting	Origin, Gsharp
Data requiring simple mathematical or statistical analysis	Charting Module in Excel, Origin, Gsharp
Complicated statistical analysis and good quality scientific charts	Graphics module in SPSS

PRESENTATIONS

Presentation software is used to create material used in presentations, such as OHP transparencies and 35mm slides. The term is also commonly used when a presentation is given using the output from a computer screen. The use of presentation software is becoming of increasing importance as higher standards become expected in courses and presentations. This will often include making use of colour, graphics and the University logo.

Course materials produced using presentation packages can be delivered in a number of ways. The simplest way is to print the material on a laser printer and then use a photocopier to produce overhead projector (OHP) acetates (first making sure that the photocopier can accept acetates). You can also use the output services produced by Information Systems Services and University Media Services to produce colour output or output on 35mm slides.

Alternatively you can give a desktop presentation using OHP projection tablets or projection systems to deliver a presentation using the output from a computer system directly. The simplest presentation software is a word processor. Word processing packages such as Word, which can produce text in a variety of sizes, can be used to create OHP transparencies.

Specialist presentation packages, such as PowerPoint, provide a wider range of facilities than word processors and, in general, are easier to use for the production of presentation materials. PowerPoint is a presentation software programme that helps you quickly and easily create professional quality presentations. Presentations can be transferred onto paper, overheads or 35mm slides, or they can be shown on a video screen or computer monitor. PowerPoint's printing options include formats ranging from audience handouts to speaker's notes.

DRAWING, PAINTING AND DESIGN

Drawing and painting software is available on most platforms at the University. However, there are many differences between software intended primarily for drawing and that intended for painting. Drawing software will provide the user with a set of 'entities' used to construct the drawing (an entity is a drawing element such as a line, circle, or text string). Drawing entities can range from simple lines, points and curves in 2D to their equivalents in 3D and may include

3D surfaces. Advanced versions of drawing packages used for design are referred to as Computer Aided Design (CAD) systems. Painting software tends to work on a conceptually lower layer. Whilst it may provide some entities for constructing geometric shapes (these tend to be 2D geometric shapes), a painting package will also provide control over individual pixels in the image, *i.e.* it provides direct control over the bitmap. It is worth remembering that opening any image in a painting package causes it to become pixelated.

The following packages are available on the ISS NT Cluster Desktop:

- Paint Very basic painting programme. Can create simple pictures and edit bitmaps. Only possible to read in and save files in a BMP format.
- Picture Publisher Painting package used to edit and create pictures. Can read in and save files in a number of different formats.
- Paint Shop Pro Recommended as the main painting package on the desktop. Used to edit and create pictures. Can read in and save files in a number of different formats.
- CorelDraw Recommended as the main drawing package on the desktop. Useful for editing vector graphics.Can read in and save files in both vector and bitmap formats.

- Micrografx Designer Drawing package used for technical drawing.

The following drawing and painting software is available on the Suns:

- Island Paint Painting programme that provides tools for creating and editing images formed by monochrome and colour bitmaps. Several painting tools can be used to create geometric and freehand shapes. Scanned images and clip art can also be imported.
- Island Draw 2D drawing package.
- Island Paint General purpose CAD system in use in engineering, and allows 3D solid modelling as well as 2D/3D draughting. An extension, AEC, for architectural and construction applications, is also available.

COMPUTER AIDED DESIGN AND DRAWING

CAD systems provide drawing entities with powerful construction, editing and database techniques. CAD data can also be output and read in by other applications software for analysing the CAD model. For example, a CAD system could be used to generate a 3D model which could then be read into a finite element analysis package. A common requirement in engineering design is to produce a drawing which is a schematic layout of components, and which accurately reflects the relative sizes and relationships of these

parts. Engineering drawing and draughting is a specialist area with its own set of procedures and practices which have become de facto standards in the engineering industry. Manual methods are now being replaced by computer-assisted methods, and the software that is used to enable these drawings to be produced embodies the functions and capabilities that are required.

CAD applications are very powerful tools that can be used by a designer. The speed and ease with which a drawing can be prepared and modified using a computer have a tremendous advantage over hand-based drawing techniques. CAD-based drawings can be created very easily using the drawing primitives made available by the software (2D/3D lines, arcs, curves, 3D surfaces, text etc.). The drawing can be shared by a number of designers over a computer network who could all be specialists in particular design areas and located at different sites. CAD also allows drawings to be rapidly edited and modified, any number of times.

Drawings can also be linked into databases that could hold material specifications, material costs etc., thereby providing a comprehensive surveillance from design through to manufacturing. In engineering applications, CAD system specifications can be passed through to numerically controlled (NC) machines to manufacture parts directly.

For creating three-dimensional objects, most CAD systems will provide 3D primitives (such as boundary representations

of spheres, cubes, surfaces of revolution and surface patches). They may also provide a solid modelling facility through Constructive Solid Geometry (CSG). Using CSG, basic 3D solids (usually cubes, spheres, wedges, cones, cylinders and tori) more complex composite solids can be created using three basic operations: joining (union) solids, removing (subtraction) solids and finding the common volume (intersection) of solids. With solid modelling, mass properties of solids (*e.g.* moments of inertia, principal moments etc.) can be quickly calculated.

There is virtually no limit to the kind of drawings and models that can be prepared using a CAD system: if it can be created by hand, a CAD system will allow it to be drawn and modelled. Some of the applications where CAD is used are: architectural and interior design, almost all engineering disciplines (*e.g.* electronic, chemical, civil, mechanical, automotive and aerospace), presentation drawings, topographic maps, musical scores, technical illustration, company logos and line drawing for fine art.

Most CAD models can be enhanced for further understanding and presentation by the use of advanced rendering animation techniques (by adding material specifications, light sources and camera motion paths to the model) to produce realistic images and interactive motion through the model. AutoCad is the primary general purpose CAD system in use in engineering, and allows 3D solid

modelling as well as 2D/3D draughting. An extension, AEC, for architectural and construction applications, is also available.

SCIENTIFIC VISUALISATION

Scientific Visualisation is concerned with exploring data and information graphically - as a means of gaining insight into and understanding the data. By displaying multi-dimensional data in an easily-understandable form on a 2D screen, it enables insights into 3D and higher dimensional data and data sets that were not formerly possible. The difference between scientific visualisation and presentation graphics is that the latter is primarily concerned with the communication of information and results that are already understood. In scientific visualisation we are seeking to understand the data.

The recent upsurge of interest in scientific visualisation has been brought about principally by the provision of powerful and high-level tools coupled with the availability of powerful workstations, excellent colour graphics, and access to supercomputers if required. This symbiosis provides a powerful and flexible environment for visualising all kinds and quantities of data.

This was once regarded as the exclusive domain of expert system and application programmers who could write the large programmes required, incorporate the algorithms for the graphics, get rid of the bugs in the resulting programme

(a non-trivial and time-consuming task), and then process the data. Most of this now comes already available 'off the shelf' - all the users have to do is activate it and plug in their data sets.

Visualisation tools range from lower-level presentation packages, through turnkey graphics packages and libraries, to higher-level application builders. The former are used for simple and modest requirements on small to medium sized data sets and are often used on PCs. The second take larger and more complex data sets and have a variety of facilities for analysis and presentation of the data in two and three dimensions. The latter enable users to specify their requirements in terms of their application and 'build' a customised system out of pre-defined components supplied by the software. This can usually be done visually on the screen and then the data can be read in, processed and viewed. You can interact with it by changing parameters or altering values.

Presentation Packages

Many spreadsheet packages for the PC have the facilities for doing elementary 2D graphics, *i.e.* to take a table of X, Y data and show it in visual form on X, Y axes. This enables us to see the overall form of the data much more easily than looking at the table of numbers. It also enables us to identify any kinks or unusual features and even missing or incorrect data. These facilities are also available in PC graphics packages such as Origin - this is menu-driven and allows

users to read in data and select the options required without any programming knowledge.

Turnkey Graphics Packages and Libraries

Turnkey graphics packages include the Uniras interactive modules Unigraph, Unimap and Gsharp. Unigraph is used for scientific graphing and charting in two and three dimensions. Unimap is used for mapping, contouring and surface drawing. Gsharp is used for both. All these programmes contain advanced facilities for processing data and for the selection of curve and surface requirements. No programming knowledge or experience is required; the user interacts with the modules via menus on the screen.

Application Builders

These are large systems which contain a wide variety of pre-defined functions and facilities. Building an application consists of visually selecting the iconised functions on the screen, connecting them together by 'pipes' and then activating the network to read in the data and feed it through the interconnected modules. Many state-of-the-art functions for graphics, imaging, rendering, interfacing and displaying are contained in the system. Users can extend the functions available by writing their own modules and adding them to the system.

Examples of visualisation application builders are AVS/Express and IRIS Explorer. AVS/Express is an advanced

interactive visualisation environment for scientists and engineers. AVS/Express supports geometric, image and volume datasets. Modules can be dynamically added, connected and deleted. Modules have control panels for interactive control of input parameters in the form of on-screen sliders, file browsers, dials and buttons. AVS/Express has a wide range of data input, filter, mapper and renderer modules. Examples of mappers include isosurfaces of a 3D field, 2D slices of a 3D data volume and 3D meshes from 2D elevation datasets. Multiple visualisation techniques can be selected to suit the problem being studied. User-written programmes or subroutines in FORTRAN or C can be easily converted into AVS/Express modules which can then be integrated into networks using the network editor.

IRIS Explorer provides similar visualisation and analysis functionality. With IRIS Explorer, users view data and create applications by visually connecting software modules into flow chart configurations called module maps. Modules, the building blocks of IRIS Explorer, perform specific programme functions such as data reading, data analysis, image processing, geometric and volume rendering and many other tasks.

DESKTOP MAPPING AND GIS

Graphs which are maps, or have a cartographic component, are a special case of a 2D graph which requires some special techniques. Many people who are not

geographers require this form of graph. Mapping and GIS are two areas that benefit greatly from computer processing of images. It has been estimated that 85% of all the information used by private and public sector organisations contains some sort of geographic element such as street addresses, cities, states, postcodes or even telephone numbers with area codes. Any of these geographic components can be used to help visualise and summarise the data on a map display, enabling you to see patterns and relationships in the data quickly and easily.

MapInfo Professional is a comprehensive desktop mapping tool, available on the PC network, that enables you to create maps, create thematic maps, integrate tabular data onto maps, as well as perform complex geographic analysis such as redistricting and buffering, linking to your remote data, dragging and dropping map objects into your applications, and much more. A GIS (Geographical Information System) is a system for sorting, manipulating, analysing and displaying information with a significant spatial (map-related) content. ArcView and ArcInfo are the two packages available in this category. ArcView is a leading software package for GIS and mapping. It gives you the power to visualise, explore, query and analyse data geographically. ArcView also has three add-on packages - Spatial, Network and 3D Analyst - for more complicated queries. ArcView is available on the NT Cluster Desktop and on the Sun workstations.

ArcInfo is an advanced GIS that gives users of geographic data one of the best geoprocessing systems available at present. It integrates the modern principles of software engineering, database management and cartographic theory. Users are advised that this is a very comprehensive GIS package and requires familiarity with and understanding of GIS concepts. ArcInfo is available on the Sun workstations.

SUBROUTINE LIBRARIES FOR GRAPHICS

Uniras and OpenGL are subroutine libraries which are available at Leeds. The former is available on both the Sun and the Silicon Graphics workstations whilst the latter is only available on the Silicon Graphics workstations. Both libraries have at least FORTRAN and C bindings. This means that users have to embed their graphics requirements into their own application programmes and write their own programme code to do this.

In contrast, the interactive modules of Uniras (*e.g.* Gsharp or Unigraph) work entirely off data sets - you do not need to write a programme. If you have a pre-existing applications programme for which you require graphical output, it may be easier just to produce a data file from the execution of this programme and then read this data file into a software package.

It only becomes necessary to write your own programme (or extend your existing programme to include calls to graphics library routines) if you have to embed your graphics

requirements to make them an integral part of your application environment, or (in the case of Uniras) you need the more advanced library functions which are not available in the interactive modules.

Multimedia

There is joint provision for networked colour printing, graphics, slides and video by Information Systems Services and the Print & Copy Bureau.

On-line Services: Printers, Slide Makers and Scanners

A4 monochrome (black and white) and colour postscript printers are available on the network. Users can send electronic picture and text information for direct output on to paper or OHP foil. Additional printing facilities are provided by Media Services where users can also discuss converting draft electronic information into pre-designed images with design staff.

Computer-Based Video Production

Data can be displayed or animated in real-time on a high-powered workstation. However, the audience is clearly limited to those who can sit at the workstation. For research seminars, conference presentations, and grant proposals it is often more useful to be able to record the real-time image sequences on video tape and present them to the audience via a video player or video projector. To ensure such

presentations are effective, they have to be at a professional standard of presentation. All of us have become unconsciously accustomed to a high quality of presentation from watching programmes on television. Anything less than this immediately looks inferior and can often reflect on the content of what is being presented.

2

Photorealistic Rendering in Graphics

There was a time when the computer generated image wanted to be more like a photograph. We believe that by stumbling across Set Visions, a digital photography studio and their interpretation of cgi through their PIX 'brand', we might just have actually found the perfect photorealistic images!

SET VISIONS PHOTOREALISTIC RENDERING STUDIO

They successfully combine together the two disciplines of photography and cgi, whilst setting unrivalled new standards for 3D based imagery. PIX effectively produces visual solutions delivered to brief, within schedule and budget, whilst constantly showcasing a perfect blend of creative

solutions. For composition and lighting they benefit from photography skills that are critical especially if you are aiming for an exceptional degree of photorealism. They have professional stylists offering the delicate finishing differences, and a host of other internal resources to call on, which each adds that little 'something of realism' to a cg image.

Set Visions are a studio that constantly develops new techniques, to create a perfect ambience and spirit, ensuring a balance of reality to fulfil the most creative of briefs and produce images that simply look like photographs.

NON-PHOTOREALISTIC RENDERING

Non-Photorealistic Rendering (NPR) research will play a key role in the scientific understanding of visual art and illustration.

NPR can contribute to scientific understanding of two kinds of problems: how do artists create imagery, and how do observers respond to artistic imagery?

Some simple but effective ideas in NPR, for cartoon illustration, painting, and line drawing. The algorithms involved are based on simple mathematical or algorithmic ideas, and, given a set of high-level parameters, run completely automatically. While there is much to criticize in these results, I believe it is nonetheless amazing and intriguing that they can be achieved by simple and automatic algorithms.

Fig. A photograph Processed by the Algorithm and the Variant.

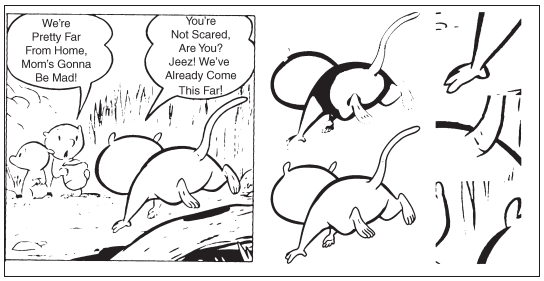


Fig. *Left*: Illustration from "Bone". *Middle*: A 3D model of the possum we created, and a rendering of the possum with our algorithm. *Right:* The line thicknesses in the drawing exhibit a number of properties that can be derived from the Isophote Distance formula. These are rules that can be checked without knowing the 3D geometry. (top) Because the wrist has higher curvature than the upper arm, it has thinner strokes. (middle) Occluded strokes are not tapered. (right) Distant objects have thinner strokes than nearby objects of the same type.

GLOBAL ILLUMINATION

Global illumination is a general name for a group of algorithms used in 3D computer graphics that are meant to add more realistic lighting to 3D scenes. Such algorithms take into account not only the light which comes directly from a light source (*direct illumination*), but also subsequent cases in which light rays from the same source are reflected

by other surfaces in the scene, whether reflective or not (*indirect illumination*).

Theoretically reflections, refractions, and shadows are all examples of global illumination, because when simulating them, one object affects the rendering of another object (as opposed to an object being affected only by a direct light). In practice, however, only the simulation of diffuse inter-reflection or caustics is called global illumination.

Images rendered using global illumination algorithms often appear more photorealistic than images rendered using only direct illumination algorithms. However, such images are computationally more expensive and consequently much slower to generate. One common approach is to compute the global illumination of a scene and store that information with the geometry, *i.e.*, radiosity.

That stored data can then be used to generate images from different viewpoints for generating walkthroughs of a scene without having to go through expensive lighting calculations repeatedly. Radiosity, ray tracing, beam tracing, cone tracing, path tracing, Metropolis light transport, ambient occlusion, photon mapping, and image based lighting are examples of algorithms used in global illumination, some of which may be used together to yield results that are not fast, but accurate.

These algorithms model diffuse inter-reflection which is a very important part of global illumination; however most of

these (excluding radiosity) also model specular reflection, which makes them more accurate algorithms to solve the lighting equation and provide a more realistically illuminated scene.

The algorithms used to calculate the distribution of light energy between surfaces of a scene are closely related to heat transfer simulations performed using finite-element methods in engineering design.

In real-time 3D graphics, the diffuse inter-reflection component of global illumination is sometimes approximated by an "ambient" term in the lighting equation, which is also called "ambient lighting" or "ambient colour" in 3D software packages. Though this method of approximation (also known as a "cheat" because it's not really a global illumination method) is easy to perform computationally, when used alone it does not provide an adequately realistic effect. Ambient lighting is known to "flatten" shadows in 3D scenes, making the overall visual effect more bland. However, used properly, ambient lighting can be an efficient way to make up for a lack of processing power.

REAL-TIME GLOBAL ILLUMINATION

Global illumination takes into account light reflecting off all surfaces in the world (indirect illumination) in addition to light coming directly from light sources (direct illumination). The images created using global illumination are more photorealistic and natural-looking than images lit with direct

light only. Special effects companies and animation studios have known this for a long time. They have been using techniques to approximate global illumination, like hand placing dozens of area lights, for years.

Recently, global illumination has started to replace the approximation techniques in films. For example, Shrek 2 was rendered with one bounce of indirect light, and newer films like Monster House are being rendered with full global illumination.

Global illumination looks great, but traditional techniques for calculating it are far too slow for real-time applications. Render times for global illuminated scenes are often measured in hours per frame. Render times for a frame in real-time applications are measured in milliseconds. This situation has lead many to pursue ways of pre-computing global illumination information and using the results in real-time rendering.

Unfortunately, such techniques do not work satisfactorily with deformable geometry in dynamic environments. Just as bad, they often force the user to separate objects being

rendered from the environment that provides the light, which is often impossible.

For simplicity and flexibility we use a new technique in the Fantasy Engine that allows us to compute global illumination on the fly. The system is simple to use since any surface in a scene can be a light source for, reflect light onto, or shadow any surface in the scene, including itself. It avoids the complexity and limitations of having to treat objects and environment separately. The technique does not involve spherical harmonics, ambient occlusion, image-based lighting (including reflection mapping), or pre-computed radiance transfer. Deformable surfaces in dynamic environments are automatically supported. They are not a problematic special case.

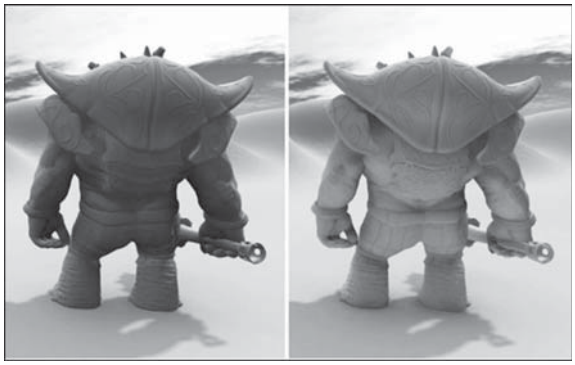


Fig. The Image on the Left was Rendered with Direct Illumination Only. The one on the Right was Rendered with Indirect Illumination too, which Makes Quite a Difference.

Our global illumination technique supports specular material properties for rendering shiny and glossy surfaces. While the specular results are not as accurate as those achieved with ray tracing, they look quite convincing and

allow for rendering all sorts of materials. An approximation to subsurface scattering is also supported, allowing for translucent materials too.

A very important feature of our global illumination method is that it supports displacement-mapped, subdivision surfaces. It allows us to render complex geometry that deforms correctly when animated. There is more information on displacement mapping of subdivision surfaces here.

These QuickTime video clips were captured from a real-time (> 30 fps at 1280x1024) Fantasy Engine demo. All the direct light in the first clip on the left comes from the glowing sky geometry. Sunlight is added for the second clip. The next clip shows 360° view of the model rendered with and without materials. In the final clip the glowing club supplies most of the light.

Performance

An NVIDIA® GeForce™ Go 7900 GTX calculates the global illumination solution (light bouncing to convergence) for the scene in these clips above in about 3.3 milliseconds (300 frames per second) per frame treating all surfaces as dynamic. Static geometry can be handled much faster.

PARTICIPATING MEDIA RENDERING

When rendering large scenes or scenes with smoke, dust etc. it is necessary to take into account the scattering of light as it passes through the media. This involves solving the radiative transport equation (an integro-differential equation), which is more complicated than the traditional rendering equation solved by global illumination algorithms. The photon map is quite good at simulating light scattering in participating media.

NEW TECHNIQUE IMPROVES RENDERING OF SMOKE, DUST AND PARTICIPATING MEDIA

Computer graphic artists often struggle to render smoke and dust in a way that makes a scene look realistic, but researchers at Disney Research, Zürich, Karlsruhe Technical Institute in Germany, and the University of Montreal in Canada have developed a new and efficient way to simulate how light is absorbed and scattered in such scenes.

LED LIGHTING

"Our technique could be used to simulate anything from vast cloudscapes, to everyday 'solid' objects such as a glass of orange juice, a piece of fruit or virtually any organic substance," said Dr. Wojciech Jarosz of Disney Research Zürich, who led the research team. The team's new virtual ray lights technique will be presented Aug. 7 in the "Light Rays" session at this year's international SIGGRAPH conference, which focuses on the latest and greatest advances in Computer Graphics and Interactive Techniques, at the Los Angeles Convention Center.

Beneath the surface, this new approach leverages another Disney Research technology — photon beams, also developed by a team lead by Jarosz — which challenged the traditional views on how to simulate light in scenes with smoke, dust or other "participating media." Normally, realistic rendering techniques simulate light using a set of particles (virtual "photons") that bounce off of walls and objects, depositing tiny bits of light-energy along their trip. It's this light-energy that's collected to form the final simulated image. But Disney researchers have found that it's much more efficient to use long and thin beams of light, instead of tiny photon particles, as a building block for generating images. This photon beams approach was presented at last year's SIGGRAPH conference in Vancouver and was also used to create magical wispy effects for Disney's Tangled. Since then Disney researchers

have been hard at work to make the technique even better. This latest work in the photon beams family looks at how photon beams contribute to so-called secondary lighting events in participating media: namely, when light enters a smoky or dusty room, the light particles actually hit and bounce off of the smoke or dust. This special game of ping-pong happens in reality at the speed of light, and the newly proposed technique investigates how entire beams of light are ping-ponged around the dust and smoke clouds in a room, or the pulp inside a glass of orange juice. With this new technique computer graphics experts can simulate more realistic participating media effects, which occur more commonly than one would expect in the real-world: participating media effects account for the way we observe clouds, the appearance of fruit juices and milk, the haze in smoggy cities, and even the subtle dimming of distant objects on an otherwise clear day. The amount to which these effects can contribute to a final rendered image varies from tiny shifts to major rifts, but even the most subtle of changes to the appearance of a virtual scene, when executed correctly, can help convince otherwise oblivious audience members that what they are seeing is "real."

The virtual ray lights technique was also designed to be flexible, and the researchers hope it will find its way into many different areas of the computer animation and special effects industry. Virtual ray lights were designed to be

progressive, meaning that they can very quickly generate a preview-quality result while converging to a final result as time goes by. On the one hand, this allows skilled technical artists at feature-film studios to quickly get feedback about their lighting setups and designs, as opposed to making changes and grabbing a coffee before being able to tell if their design changes are helpful or not.

This rapid-feedback property reduces iteration time, allowing artists to focus on the end-goal instead of wrestling with the lighting tool. Another benefit of the progressive nature of the new technique is that users can choose between quality and performance, which is ideal for game developers who don't care so much about being 100 percent realistic but instead want their scenes to "just look great." Finally, the virtual ray lights project is a shining example of Disney's commitment to bringing together the brightest experts from across industrial and academic research settings in order to push the limits of the state-of-the-art.

This project was undertaken by four researchers in three countries across two continents: Jan Novák, an intern at Disney Research in Zürich and full-time PhD student at Karlsruhe Institute of Technology, worked alongside Wojciech Jarosz, a research scientist and head of the Rendering Group at Disney Research Zürich, Derek Nowrouzezahrai, a Disney Research post-doc and now assistant professor at the

University of Montreal, and Carsten Dachsbacher, the head of the Computer Graphics Group at Karlsruhe Institute of Technology.

RAY TRACING

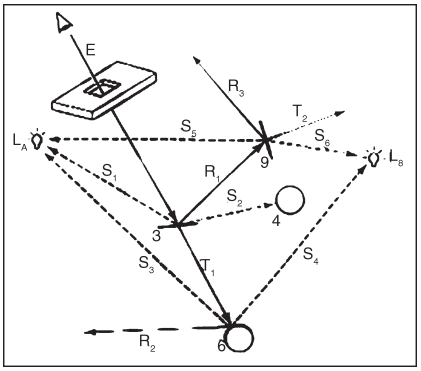
Ray Tracing is a global illumination based rendering method. It traces rays of light from the eye back through the image plane into the scene. Then the rays are tested against all objects in the scene to determine if they intersect any objects.

If the ray misses all objects, then that pixel is shaded the background colour. Ray tracing handles shadows, multiple specular reflections, and texture mapping in a very easy straight-forward manner.

Note that ray tracing, like scan-line graphics, is a point sampling algorithm. We sample a continuous image in world coordinates by shooting one or more rays through each pixel. Like all point sampling algorithms, this leads to the potential problem of aliasing, which is manifested in computer graphics by jagged edges or other nasty visual artifacts. In ray tracing, a ray of light is traced in a backwards direction. That is, we start from the eye or camera and trace the ray through a pixel in the image plane into the scene and determine what it hits. The pixel is then set to the colour values returned by the ray.

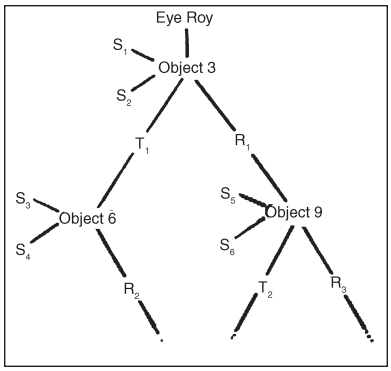
SIMPLE GLOBAL ILLUMINATION MODEL FOR RAY TRACING

A primary ray is shot through each pixel and tested for intersection against all objects in the scene. If there is an intersection with an object then several other rays are generated. Shadow rays are sent towards all light sources to determine if any objects occlude the intersection spot. The shadow rays are labelled S_i and are sent towards the two light sources L_A and L_B. If the surface is reflective then a reflected ray, R_i, is generated. If the surface is not opaque, then a transmitted ray, T_i, is generated. Each of the secondary rays is tested against all the objects in the scene.



The reflective and/or transmitted rays are continually generated until the ray leaves the scene without hitting any object or a preset recursion level has been reached. This then generates a ray tree, as shown below.

The appropriate local illumination model is applied at each level and the resultant intensity is passed up through the tree, until the primary ray is reached. Thus we can modify the local illumination model by (at each tree node)



I = Ilocal + Kr * R + Kt * T where R is the intensity of light from the reflected ray and T is the intensity of light from the transmitted ray. Kr and Kt are the reflection and transmission coefficients. For a very specular surface, such as plastic, we sometimes do not compute a local intensity, Ilocal, but only use the reflected/transmitted intensity values.

RAY OBJECT INTERSECTIONS

The general idea behind ray-object intersections is to put the mathematical equation for the ray into the equation for the object and determine if there is a real solution. If there is a real solution then there is an intersection (hit) and we must return the closest point of intersection and the normal (N) at the intersection point.

For a shadow ray we must return whether any ray-object intersection is closer than the ray-light intersection. For a ray tested against a boundary volume, we just return a simple hit or no hit. For texture mapping we need the intersection point relative to some reference frame for the surface.

We define a ray as:

R0 = [x0, y0, z0] - origin of ray

Rd = [xd, yd, zd] - direction of ray

then define a set of points on the ray:

R(t) = R0 + Rd * t with t > 0.0

If Rd is normalized, then t equals the distance of the ray from origin in World Coordinates, else it is just a multiple of Rd, so we want to normalize Rd. Note that many of the intersection computations require the solution of the quadratic equation.

PRACTICAL CONSIDERATIONS IN WRITING A RAY TRACER

Process: For each pixel a primary ray will be generated and then tested against all objects in the scene.

Create Model

The first step is to create the model of the image. One should not hardcode objects into the programme, but instead use an input file. Here is a sample Input file:

Generate Primary Rays and Test for Object-Ray Intersections

For each pixel we must generate a primary ray and test for intersection with all of the objects in the scene. If there is more than one ray-object intersection then we must choose the closest intersection (the smallest positive value of t). To ensure that there are no objects intersected in front of the

image plane (this is called near plane clipping), we keep the distance of the primary ray to the screen and test all intersections against this distance. If the t value is less than this distance, then we ignore the object.

A sample calculation of forming a ray and testing it for intersection with a sphere. If there is an intersection then we must compute the shadow rays and the reflection rays.

Shadow Ray

The shadow ray is a ray from the point of intersection to the light source. Its purpose is to determine if the intersection point is in the shadow of a particular light. There should be one shadow ray for each light source. The origin of the shadow ray is the intersection point and the direction vector is the normalized vector between the intersection point and the position of the light source. Note that this is the same as the light vector (L) that is used to compute the local illumination. Compute the Local Illumination at each point, carry it back to the next level of the ray tree so that the intensity I = Ilocal + Kr * R + Kt * T. Note that Kr can be taken as the same as Ks.

For each colour (R, G, B) I is in the range 0.0 <= I <= 1.0. This must be converted to an integer value of 0 <= I <= 255. The result is then written to the output file.

Output File

The output file will consist of three intensity values (Red, Green, and Blue) for each pixel. For a system with a 24-bit

framebuffer this file could be directly displayed. However, for a system with an 8-bit framebuffer, the 24-bit image must be converted to an 8 bit image, which can then be displayed. A suggested format for the output file is the Microsoft Windows 24-bit BMP image file format.

ACCELERATING RAY TRACING

Ray Tracing is so time-consuming because of the intersection calculations. Since each ray must be checked against all objects, for a naive raytracer (with no speedup techniques) the time is proportional to the number of rays X the number of objects in the scene. Each intersection requires from a few (5-7) to many (15-20) floating point (fp) operations. Thus for a scene with 100 objects and computed with a spatial resolution of 512 × 512, assuming 10 fp operations per object test there are about 250,000 × 100 × 10 = 250,000,000 fps. This is just for the primary rays (from the eye through the image plane) with no anti-aliasing. Clearly there are computational problems with this.

There are several approaches to speeding up computations:

- Use faster machines
- Use specialized hardware, especially parallel processors.
- Speed up computations by using more efficient algorithms
- Reduce the number of ray - object computations.

REDUCING RAY-OBJECT INTERSECTIONS

Adaptive Depth Control

This means that we stop generating reflected/transmitted rays when the computed intensity becomes less than a certain threshold. You must always set a certain maximum depth or else the programme would generate an infinite number of rays. But it is not always necessary to go to the maximum depth if the surfaces are not highly reflective. To test for this the ray tracer must compute and keep the product of the global and reflection coefficients as the rays are traced.

Example: let Kr = 0.5 for a set of surfaces. Then from the first surface the maximum contribution is 0.5, for the reflection from the second: 0.5 * 0.5 = 0.25, the third: 0.25 * 0.5 = 0.125, the fourth: 0.125 * 0.5 = 0.0625, the fifth: 0.0625 * 0.5 = 0.03125, etc. In addition we might implement a distance attenuation factor such as 1/D2, which would also decrease the intensity contribution.

For a transmitted ray we could do something similar but in that case the distance traveled through the object would cause even faster intensity decrease. As an example of this, Hall & Greenbergfound that even for a very reflective scene, using this with a maximum depth of 15 resulted in an average ray tree depth of 1.7.

Bounding Volumes

We enclose groups of objects in sets of hierarchical bounding volumes and first test for intersection with the

bounding volume, and then only if there is an intersection, against the objects enclosed by the volume.

Bounding volumes should be easy to test for intersection, for example a sphere or box (slab). The best bounding volume will be determined by the shape of the underlying object or objects. For example, if the objects are long and thin then a sphere will enclose mainly empty space and a box is much better. Boxes are also easier for hierarchical bounding volumes.

Note that using a herarchical system like this (assuming it is done carefully) changes the intersection computational time from a linear dependence on the number of objects to something between linear and a logorithmic dependence. This is because, for a perfect case, each interesction test would divide the possibilities by two, and we would have a binary tree type structure. Spatial subdivision methods, discussed below, try to achieve this.

Kay & Kajiya give a list of properties for hierarchical bounding volumes:

- Subtrees should contain objects that are near each other and the further down the tree the closer should be the objects.
- The volume of each node should be minimal.
- The sum of the volumes of all bounding volumes should be minimal.
- Greater attention should be placed on the nodes near the root since pruning a branch near the root will

remove more potential objects than one farther down the tree.

- The time spent constructing the hierarchy should be much less than the time saved by using it.

First-Hit Speedup

On adaptive depth control, by using that technique the average depth of the ray tree may be less than two. This means that a large percentage of the work is performed in finding the first intersection. Weghorst has suggested using a modified Z-buffer algorithm to determine the first hit. The scene would be pre-processed, with the resultant z-buffer storing pointers to the objects intersected. Then the ray tracing would proceed from that point. He showed that incorporating the above three techniques (adaptive depth control, hierarchical bounding volumes, and first-hit speedup) approximately halved the intersection computational time for complex scenes. Note that he use spheres as the bounding volumes. In general the computational improvement was inversely dependent on scene complexity.

Weghorst showed that incorporating the above three techniques (adaptive depth control, hierarchical bounding volumes, and first-hit speedup) approximately halved the intersection computational time for complex scenes. Note that he used spheres as the bounding volumes. In general the computational improvement was inversely dependent on scene complexity.

DETAILED DESCRIPTION OF RAY TRACING COMPUTER ALGORITHM AND ITS GENESIS

What Happens in Nature

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 absorb part of the light ray, resulting in a loss of intensity of the reflected and/or refracted light.

It might also reflect all or part of the light ray, in one or more directions. If the surface has any transparent or translucent properties, it refracts a portion of the light beam into itself in a different direction while absorbing some (or all) of the spectrum (and possibly altering the colour). 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

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 modelling techniques and easily rendered.

Ray Tracing Algorithm

Fig. Ray Tracing can Achieve a Very High Degree of Visual Realism.

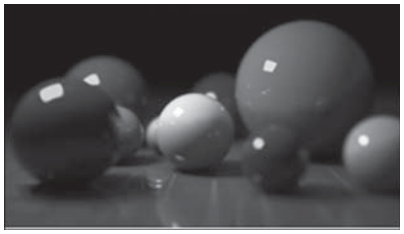


Fig. In Addition to the High Degree of Realism, Ray Tracing can Simulate the Effects of a Camera due to Depth of Field and Aperture Shape.

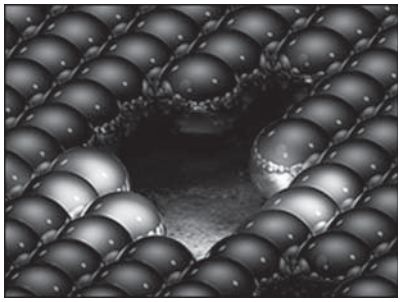


Fig. The Number of Reflections a "Ray" can Take and how it is Affected each Time it Encounters a Surface is all Controlled via Software Settings during Ray Tracing. Here, Each Ray was Allowed to Reflect up to 16 Times. Multiple "Reflections of Reflections" can thus be Seen.

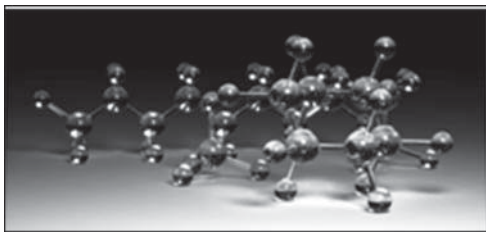


Fig. The Number of Refractions a "Ray" can take and how it is Affected each Time it Encounters a Surface is all Controlled via Software Settings during Ray Tracing.

The next important research breakthrough came from Turner Whitted in 1979. Previous algorithms cast rays from the eye into the scene until they hit an object, 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 spatial anti-aliasing and improve image quality where needed. Although it does handle interreflection and optical effects such as refraction accurately, traditional ray tracing is also not necessarily photorealistic. True photorealism occurs when the rendering equation 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.

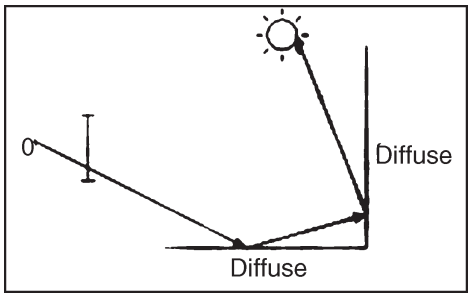


Fig. To the Right is an Image Showing a Simple Example of a Path of Rays Recursively Generated from the Camera (or Eye) to the Light Source using the above Algorithm. A Diffuse Surface Reflects Light in all Directions.

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 colour of the pixel now depends on the colours of the first and second diffuse surface and the colour 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 colour of the pixel is blue.

3

Hypermedia Development Tools

Hypermedia may be developed a number of ways. Any programming tool can be used to write programmes that link data from internal variables and nodes for external data files. Multimedia development software such as Adobe Flash, Adobe Director, Macromedia Authorware, and MatchWare Mediator may be used to create stand-alone hypermedia applications, with emphasis on entertainment content. Some database software such as Visual FoxPro and FileMaker Developer may be used to develop stand-alone hypermedia applications, with emphasis on educational and business content management.

Hypermedia applications may be developed on embedded devices for the mobile and the Digital signage industries using the Scalable Vector Graphics (SVG) specification from W3C

(World Wide Web Consortium). Software applications such as Ikivo Animator and Inkscape simplify the development of Hypermedia content based on SVG. Embedded devices such as iPhone natively support SVG specifications and may be used to create mobile and distributed Hypermedia applications.

Hyperlinks may also be added to data files using most business software via the limited scripting and hyperlinking features built in. Documentation software such as the Microsoft Office Suite and LibreOffice allow for hypertext links to other content within the same file, other external files, and URL links to files on external file servers. For more emphasis on graphics and page layout, hyperlinks may be added using most modern desktop publishing tools. This includes presentation programmes, such as Microsoft Powerpoint and LibreOffice Impress, add-ons to print layout programmes such as Quark Immedia, and tools to include hyperlinks in PDF documents such as Adobe InDesign for creating and Adobe Acrobat for editing. Hyper Publish is a tool specifically designed and optimized for hypermedia and hypertext management. Any HTML Editor may be used to build HTML files, accessible by any web browser. CD/DVD authoring tools such as DVD Studio Pro may be used to hyperlink the content of DVDs for DVD players or web links when the disc is played on a personal computer connected to the Internet.

A BRIEF HISTORY OF HYPERMEDIA

DEFINITION OF THE HYPERTEXT

"Information is linked and cross-referenced in many different ways and is widely available to end users".

"Hypertext means a database in which information (text) has been organised nonlinearly. The database consists of nodes and links between nodes".

Vannever Bush's article "As We May Think": There should be a tool that would enhance human memory and thinking and that allows people to retrieve information from a computer in many of the same ways in which retrieval is accomplished within human memory.

A Hypermedia Timeline

Ted Nelson: Xanadu is Ted Nelson's dream since early '60s: all the world literature in one publicly accessible global online system (analogy: you can today get a telephone link from anywhere to anywhere, so why not from any text to any other?). Every reference to a text will lead to royalties being paid automatically to the author. Includes the use of full versioning (claimed to be horrifyingly complex), "hot links" (called transclusions) and zippered texts (*e.g.* parallel texts like for translations or annotations.). A few of ideas in Xanadu are now implemented in WWW.

Dough Englebart: Can be called as The father of the Hypertext. He invented f.ex. mouse and touch screens. He is

also creator of one of the first hypermedia systems NLS/ Augment.

Definitions of Concepts

A link is defined by source and destination nodes, and by an anchor in the source node. The destination of a link can be a file (so-called string-to-lexia link) or a string in a file (string-to-string link). With a string-to-lexia link it is not possible to reference to a certain part of a file. This kind of link can make hypermedia easily navigable, especially if the destination nodes are "short" documents. String-to-string links would permit the destination to be a string in a file, but this kind of link requires more planning in the design process.

String-to-lexia links also support implicit linking. Implicit links are generated by the hypermedia software on runtime, for example referential links from a concept to the definition of the concept. To some extent hypermedia software should generate links from the fluctuated forms of concepts (a link from word "matrix" and "matrices"). In contrast to implicit links, explicit links are generated by the hypermedia author.

The nodes and links form a network structure in the database. Hypermedia is a database which contains pictures, digitised videos, sound and animations in addition to text.

DOCUMENT MARKUP LANGUAGES

Why we need a document markup? Bryan defines markup as: "Markup is the term used to describe codes added to

electronically prepared text to define the structure of the text or the format in which it is to appear." There can be two types of markups: specific markup and generalized markup. Specific markup describes the format of the document whereas generalized markup describes the structure of the document (headings, citations etc.).

For example, Rich Text Format (RTF) is a specific markup language and TeX, LaTeX, SGML and HTML are general markup languages.

Standard Generalized Markup Language (SGML)

SGML is an international stardard (ISO 8879) for document markup. An SGML document contains a document type definition (DTD) and a set of elements, that are defined in DTD. Each element has a name and it can be used as a tag in SGML document.

HyperText Markup Language (HTML)

HTML is a SGML based markup language for WWW documents. HTML is actually a DTD, a set of definitions of how to interpret HTML tags.

HyTime

HyTime is an international standard for hypermedia documents. It is based on SGML, but it can reference to a data in almost any format. Only the hypertext link information is required to be in SGML format.

TeX and LaTeX

TeX and LaTeX are also general markup languages in the sense that we only describe document structures with LaTeX macros. The definition of the macros can be later changed and the document could be formatted differently.

There is a HyperTeX that has limited hypertext capability by implementing/special keyword so that is supports for example URL's. DVI viewer is then used to display ps files containing URL's. DVI viewer could call WWW browser to follow the URL.

Rich Text Format (RTF)

The difference between SGML and RTF is that SGML describes the structure of a document, whereas RTF describes mainly the physical characteristics of the text (text face, size, etc). However, RTF includes also certain tags that describe document structure. The author can define a set of styles for the document (heading 1, heading 3, abstract, etc.) that are written into the beginning of the RTF file and have a special tag in the RTF markup. An RTF file contains all text formatting, pictures and formulas and it is a standard defined by Microsoft.

OpenMath

OpenMath consortium is an international group of researchers designing a protocol for exchanging mathematical information between applications. For example, a general

purpose computer algebra system could call a specific purpose application to execute an algorithm implemented only in this application. OpenMath tries to preserve semantic information in addition to the structural information of the formula. For example, TeX describes only the visual appearance of a formula, not the semantic structure of the formula. Similar visual representation of mathematical formulas has been planned to SGML. MathLink is communications protocol for exchanging Mathematical expressions and data between Mathematical and external applications. The difference between MathLink and OpenMath is that MathLink does not define the semantical information of a formula.

HYPERMEDIA MODELS

Hypertext Abstract Machine

Hypermedia is divided into: (1) User interface, (2) hypermedia application (client), (3) HAM Hypermedia "engine" (server) that retrieves link and node information from database and passes that to the hypermedia application, (4) database.

Dexter Hypertext Reference Model

Purpose of Dexter model is that it is standard hypertext terminology coupled with a formal model of the important abstractions commonly found in a wide range of hypertext systems. Dexter model is actually a formal specification of

generic hypermedia system written in Z (Proceedings of the Hypertext Workshop, NIST Special publication 1990).

HYPERMEDIA SYSTEMS

Intermedia

A well known hypermedia system is Intermedia developed at Brown Universities Institute for Research in Information and Scholarship (IRIS) between 1985 and 1990. Intermedia is a multiuser hypermedia framework where hypermedia functionality is handled at system level. Intermedia presents the user graphical file system browser and a set of applications that can handle text, graphics, timelines, animations and videodisc data. There is also a browser for link information, a set of linguistic tools and the ability to create and traverse links. Link information is isolated from the documents and are saved into separate database. The start and end position of the link are called anchors.

World Wide Web

World Wide Web (WWW) is a global hypermedia system on Internet. It can be described as wide-area hypermedia information retrieval initiative aiming to give universal access to a large universe of documents. It was originally developed in CERN for transforming research and ideas effectively throughout the organization. Through WWW it is possible to deliver hypertext, graphics, animation and sound between different computer environments. To use WWW the user

needs a browser, for example NCSA Mosaic and a set of viewers, that are used to display complex graphics, animation and sound. NCSA Mosaic is currently available on X-Windows, Windows and Macintosh.

NSCA Mosaic and Netscape

The browser itself can read hypertext documents that are marked with HyperText Markup Language (HTML). HTML is based on Standard Generalized Markup Language (SGML), and contains all formatting and link information as ASCII text. HTML documents can reside on different computers on Internet, and a document is referenced by URL (Universal Resource Locator). URL is of the form http://computer.org.country/doc.html where computer.org.country is the name of the computer and doc.html is the search path to the document. In order to create a node for WWW, a HTTP (HyperText Transfer Protocol) server application is needed. A link in WWW document is always expressed as URL. Links can be references to files in ftp-servers, Gophers, HTTP-servers or Usenet newsgroup.

Netscape is a popular WWW browser developed by Netscape Communications Corp. Netscape 1.1 supports some HTML 3.0 features (tables) and has interesting API, that makes it possible to develop. Documents that are formatted using RTF can be transfered to HTML by using a converter RTF to HTML. It generates HTML document from the original RTF document and a set of picture files if the RTF document contained

pictures. In the HTML document links are created to the graphics files. The graphics can be viewed on most environments if pictures are of type GIF.

Arena

Arena is an experimental WWW browser developed in CERN. It supports HTML 3.0 and thus is able to display mathematical formulas and tables.

MathBrowser

Recently, The Mathsoft company has announced MathBrowser, a WWW-browser that can display HTML and MathCAD documents. MathBrowser has a computational engine and interface similar than MathCAD, allowing the student to edit MathCAD documents through the Internet. MathBrowser is used to distribute a collection of Shaum's outline series in electronic form.

HyperCard, Toolbook and MetaCard

HyperCard is hypermedia authoring software for Macintosh computers. It is based on a card-metafora. Hypercard application is called a stack or a collection of stacks. Each stack consists of cards and only one card is visible in a stack. A card is displayed in fixed size window. Hypertext links can be programmed by creating buttons and writing a HyperTalk script for the button. MetaCard is similar application than HyperCard but it runs in Unix environments. MetaCard offers the ability to create and modify applications using interactive

tools and a simple scripting language. Intrestingly, HyperCard stacks can be imported to MetaCard. However there are some incoptabilities on the HyperTalk and MetaTalk, so advanced stacks dont run without modifications.

LinksWare

LinksWare is a commercial hypermedia authoring software for Macintosh that can create hypertext links between text files created with different word processors. LinksWare uses a set of translators to convert files to its own format (Claris XTND system). This can make the opening of a file very slow. LinksWare can open files that contain mathematical text, but files may be formatted differently than in original document, especially formulae do not appear to have proper line heights. In addition, it can not create links to other applications. However, it can create links to Apple script command files that can open an application and execute commands for that application.

HyperG

HyperG is the name of an hypermedia project currently under development at the IICM. Like other hypermedia undertakings, HyperG will offer facilities to access a diversity of databases with very heterogeneous information (from textual data, to vector graphics and digitized pictures, courseware and software, digitized speech and sound, synthesized music and speech, and digitized movie-clips).

Like other hypermedia-systems it will allow browsing, searching, hyperlinking, and annotation. Like no other big hypermedia system known today, it will also support automatic indexing and link-generation, a variety of automatic consistency-checks, a built-in messaging and computer conferencing system, a special editor allowing the incorporation of animation sequences, question/answer dialogues, and a number of unorthodox man-machine interfaces. Further, and maybe most important of all, it is built on the basis of already existing large databases: hundreds of CAI lessons, a large general-purpose encyclopaedia in hypermedia form, a number of smaller special-purpose lexica, a data-base of thousands of pictures, some pieces of digitized sound and movie-clips, and links to other databases in other networks. A number of smaller spin-off applications are surfacing which are mainly pursued by IMMIS and have led to research in the area of computerisation of various aspects of museums.

Interactive Calculus Courses

However, there are also whole courses implemented in computer form. A package called Calculus & Mathematical is a calculus course that uses Mathematical notebooks to present theory, examples and exercises. Students can fully access Mathematical by typing in commands in Mathematical language. Each notebook introduces the theory of the topic, which is followed by examples. Calculus & Mathematical

contains a special font for the presentation of mathematical formulas.

The Transitional Mathematics Project at Imperial College London has produced similar Mathematical notebooks to be used in calculus courses. At the end of each notebook there is a set of exercises to be solved with pen and paper and evaluated by the computer. Some feedback is provided for the student ("you got 2 out of 4 correct"). Mathematical formulas are created with a special formula editor and transferred to notebooks as pictures.

This kind of material certainly has its place in the classroom. It offers the capabilities of computer algebra systems to students right from the beginning of their mathematics studies. The disadvantage is that students must learn the material in sequential order; it is not possible to choose an individual order of progress. Apart from the possibility to open and close notebook cells, there are no hypertext features in Mathematical.Until recently only a few courses in mathematics have been implemented as hypertext. One example is "A Simple Introduction to Numerical Analysis". The material is published on a CD-ROM that contains the hypertext material, animations and graphical tools. Another interesting course is a hypertext version of "Introduction to Algorithms". This CD-ROM contains hypertext, several animations and QuickTime movies. These high-quality texts and animations add a new dimension to

learning mathematics. To see a sorting algorithm in action gives the user a mental reference or orientation basis to which to relate the theory and implementation of that algorithm.

However, there is a problem related to the way the material is put together. For the reader the material seems to be a collection of cards or stacks that do not form an entity. This problem is partly due to a limitation in HyperCard; HyperCard is not able to treat mathematical formulas as written text. That is why they must be transferred as pictures. HyperCard can present pictures and text only on cards, not on scrollable text fields. The card itself is not in a scrollable window and only one card can be visible at a time. As a result, when a new card is turned on, all the previous information disappears with the old card.

CAN/RIACA Interactive Book on Lie Algebra

In Amsterdam CAN/RIACA is developing a hypermedia system on Lie Algebra. They also participate in OpenMath development.

Microcosm (UK)

Microcosm is an attempt to include hypertext functionality into several applications like Microsoft Word and Exel. Microcosm has a dynamical linking facility, *i.e.* it tries to create possible links based on the selection made by the user. Applications are divided as Microcosm aware or simple viewers. Links are saved to separate link database in Microcosm server.

Mathematical Simulation Environments

Simulation environments are not hypermedia applications as such. However, they are an important part of a hypermedia based learning environment on mathematics and engineering sciences.

DESIGNING A HYPERMEDIA

Important questions in designing the hypermedia are:

- Converting linear text to hypertext
- Text format conversions
- Dividing the text into nodes
- Link structures, automatic generation of links
- Are nodes in a database or are they separate files on file system
- Client-server of standalone.

Text indexing is a well know problem area and results from there can be used to study automatic generation of links. In principle, a document can be analysed semantically (with the help of AI), statistically or lexically (by computing the occurrences of words). Problems in semantic analysis are that natural language is not easy to understand by the computer. In lexical analysis problems are for example the conflation of words and recognition of phrases (Esim. matriisi, matriisin, matriisilla mutta ei jälki, jälkeen). Solutions:

- Conflation algorithm
- Stemming algorithm
- Stopword-list.

GENERIC ADAPTIVE HYPERMEDIA SYSTEM

Several adaptive hypermedia applications have been introduced over the past few years. The overview article by Brusilovsky names most of these, and describes the adaptive techniques used in each application. In most cases the software used for maintaining the underlying user models and for generating the adaptive content and link structure is tied closely to the single application for which that software was developed. A notable exception is the Interbook system, which is a descendant from ELM-ART. While ELM-ART is only a Lisp course, which incidentally also uses Lisp for the software that maintains the user-model and generates the adaptive content, Interbook can be used to create courses on different topics. Still, although more general than ELM-ART, Interbook is still aimed specifically at educational applications. It uses a fixed frames structure to represent an (adaptive) table of contents, a set of known or required concepts, a content page, etc. The front-end of such a system (the user-interface which is realised using HTML) is tied closely to the back-end (the engine which maintains the user model and performs the adaptation).

This chapter presents the development at the Eindhoven University of Technology, which goes one step further: the software for adaptive hypermedia which was originally

developed for a course on "Hypermedia Structures and Systems" has been made very generic by concentrating all functionality in the back-end, and leaving presentation issues to the author. The adaptive system, still called AHA (for Adaptive Hypermedia Architecture), can be used with HTML presentations with or without frames. The engine which maintains the user model can be used to generate conditional text and to adapt the link structure through link removal, link hiding and link annotation. In six application areas for adaptive hypermedia are mentioned. AHA supports five of them: *on-line information systems*, *on-line help*, *educational hypermedia*, *institutional hypermedia* and *personalized views*. The one application area which is somewhat more difficult to realise using this software is *information retrieval hypermedia* (because that requires sorting of links, which AHA does not support).

The structure of a user model and how it is used to generate adaptivity. We show how adaptive content can be realised using standard HTML. The different ways to adapt the link structure through the use of link classes. We present the global architecture of the back-end and show how this engine interacts with the browser to ensure that presented pages always correspond to the correct user model.

USER MODELLING IN AHA

Adaptive systems try to anticipate the needs and desires of the user. Any knowledge a system has about the user is

based on that user's (prior) actions. The system may simply monitor what a user is doing or it may ask questions. Intelligent tutoring systems are typical: they build a user model based on what reading material is offered to the user, and validates that model by means of (mostly multiple-choice) tests. A simple way to represent a user model is by means of a set of pairs (c,v) where c is a *concept* and v is a *value* which indicates how much (or little) the user knows about that concept. However, a set of subject-value pairs need not represent knowledge about a concept. User preferences can be represented in the same way. Authors can offer variants of hyperdocuments by presetting some pairs. In the sequel we will always use the term *concept*, keeping in mind that it may mean *preference* as well.

The data types used for indicating knowledge (or preferences) are simple: Some systems allow many values, for instance a "percentage" (or integer values between 0 and 100), some have a few numeric or named values, like *no knowledge*, *read about* and *knows about*, and others have just Booleans (*true* and *false*). Some applications use only a few concepts (which they have knowledge tests for) while others use many concepts (maybe even one for every page).

Every representation system with a finite number of values can be simulated by a system with just Booleans. For systems with many values, like percentages, such a simulation would be impractical. However, we know of no system which actually

calculates percentages to the point that more than a few discrete values can be obtained, or to the point that every discrete value has a different influence on the adaptation. When a system supports three or four values per concept, each concept can be replaced by two. For instance, instead of concept *something* we could create concepts *read-about-something* and *knows-about-something*. In the AHA system we opted for Boolean values.

In AHA knowledge about a concept is generated either by reading a (single) page or by taking a test. This implies that concepts are fairly fine-grained: if the user must read five pages to achieve some desired state, each of the five pages must be associated with a different concept, and the five concepts together define the desired goal.

Apart from a set of known concepts, the AHA software also maintains a logfile for each user. For each time the user accesses a page there are two log entries: one for the start and one for the end of the period the user (supposedly) reads the page. (In this way the reading time for each page is logged.) For each test the score is stored as well. The log is part of the user model so it can be used to mark links to pages the user already read differently from links to unread pages. Most WWW-browsers also change link colours, but this behaviour would interfere with the adaptive linking. Also, users may be able to clear the browser's history and associated link colouring scheme and thereby disable the guidance the

adaptive hypertext software tries to provide through link colours. While concepts can be used to represent user preferences the AHA system implements colour preferences differently. Colours of links can be selected by the user. A Boolean representation would not be desirable, so explicit colour values are stored in the user model.

ADAPTIVE CONTENT

Depending on the user's knowledge state information on a given subject may need to be presented in different ways. Students who are first reading about hypertext for instance may be confused when they see the term "node" whereas the word "page", used in the same context, would be meaningful to them, and probably sufficiently accurate in an introductory text. In the course text for the course "2L670: Hypermedia Structures and Systems", the students must first visit a "readme" page with instructions on how to use the courseware and how to configure their WWW-browser. Therefore a short paragraph which tells students to go to the readme page is prominently displayed, along with a link to that page. After reading the instructions the textual content of the start-page of the course is changed automatically. The pointer to the readme page is removed from the top, and a small reminder at the bottom of the page is all that remains.

Such adaptive changes to a page, using what Brusilovsky calls *fragment variants*, are non-trivial to realise in World Wide Web, because HTML does not allow for text fragments

which are *conditionally* made visible or hidden. It would be possible to use *Dynamic HTML* for this purpose, but this would lead to HTML documents that are complicated (because of the JavaScript or VBscript code they need to contain) and thus difficult to author.

AHA implements adaptive content in HTML by means of a preprocessor that filters content fragments by means of conditionals encoded in structured HTML comments:

```
<!– if definition and history –>
  This part appears if the two "concepts"
  definition and history are both known
  according to the user model.
<!– else –>
  If this is not the case then this
  alternative is presented instead.
<!– endif –>
```

Another example is the use of such conditionals is to combine viewgraphs with and without comments in a single source, like:

```
    <LI>This is a viewgraph item.
<!– if verbose –>
```

And this is some additional comment which is only shown when "verbose" is true.

```
<!– endif –>
```

Similar constructs can be used to choose between a text paragraph, an image, video, etc.

Because the comments have no "meaning" in HTML, there is no need to respect the proper nesting of conditionals and HTML tags. The following example would not be allowed if the comments would be meaningful HTML tags:

```
    <LI>This is an item in an unordered list.
<!– if interrupt-list –>
    </UL>
```

This conditional text interrupts the list.

```
  <UL>
<!— endif —>
  <LI>This is the next item in the list.
```

This flexibility does imply that the author must ensure that the proper begin and end tags for lists or other HTML constructs are included under all circumstances.

ADAPTIVE LINKING

Depending on the user model an adaptive hypertext system will guide the user towards some "desired" pages, and away from pages that contain information which is not relevant at that time, or for which the user does not have the necessary foreknowledge. Brusilovsky distinguishes five types of adaptive linking: *direct guidance*, *adaptive link sorting*, *adaptive link annotation*, *adaptive link hiding* and *map adaptation*. Most adaptive hypertext systems offer only one (or maybe two) of these features. Interbook for instance concentrates on *link annotation*, meaning that desired links are marked differently from undesired links, in this case by means of a coloured (big) dot. (It also offers direct guidance through a "teach me" feature.)

The course 2L670 offered only link hiding, but in AHA this restriction has been lifted. Calvi has subdivided link hiding into three subclasses:

- (Pure) *link hiding* means that links may be hidden by making the link anchor indistinguishable from the surrounding text (*i.e.* black in most cases). The link

remains functional, so users who know that the link is there can still use it.

- *Link removal* means that the link anchor is not only made to look like normal content, but the link is removed as well. This means that even a user who knows where the link should be cannot follow the link.
- *Link disabling* means that the link is not made invisible but its link functionality is removed. The user still sees the link, but that link doesn't work. (The original version of AHA, used for a version of the course 2L670 inadvertedly used link disabling. Users almost unanimously disliked this feature.)

The latest version of AHA directly supports *link annotation* and *link hiding*, but *link removal* can easily be simulated, and *direct guidance* can be implemented with a little more authoring effort.

Adaptive link annotation in AHA is realised by means of link classes which are turned into different colours by using *cascading style sheets*. Because of differences in author and user preferences the courseware can be configured to use any colour scheme (for desired, undesired, neutral and external links) the author desires, and the user may override these choices through a "setup" HTML page and a CGI-script. Links that may be desired or undesired at times are marked in the HTML source of the pages using a link class we call

"conditional". The AHA preprocessor translates this to the classes "good" or "bad" depending on the user model.

Adaptive link hiding in AHA is achieved by choosing the colour black for one of the link types, typically the "undesired" links. When the link text is black it cannot be distinguished from the surrounding text. Only by moving the mouse pointer over the link text may the user become aware of the presence of a link, because the browser normally alerts the user by displaying the name of the destination of a link in a message line. In AHA the author chooses between link annotation and link hiding by means of the colour scheme. The user can override this choice through a setup form.

Adaptive link removal in AHA is achieved by turning the HTML anchor tag into conditional text, like in the following example:

```
<!– if desired –>
<a href="...">
<!– endif –>
here is the link anchor text
<!– if desired –>
</a>
<!– endif –>
```

This trick, with the two "if" clauses is only needed if you wish to avoid writing the link anchor text twice. A simpler, but redundant form would be:

```
<!– if desired –>
<a href="...">here is the link anchor text</a>
<!– else –>
```

here is the link anchor text

```
<!– endif –>
```

Direct guidance can be achieved through conditional content in a similar (but more laborious) way.

Direct guidance means for instance that the adaptive hypertext system offers a button which leads to a suggested page to read next. The URL to which the link leads can be filled out using "if" comments, depending on logical combinations of concepts. The following example shows how a "next" button leads to a definition if that hasn't been read before, and otherwise leads to a theorem.

```
<!– if definition –>
<a href="theorem.html">
<!– else –>
<a href="definition.html">
<!– endif –>
next
</a>
```

Offering direct guidance is difficult because the author needs to be aware of what the possible best pages to go to are, depending on each user's knowledge state. Although this may seem to be an authoring problem and not a problem with the AHA software, direct guidance can be delivered automatically (to some extent) by a system by checking the dependencies between concepts. AHA currently does not offer this feature, but Interbook for instance does (through a "teach me" button).

ARCHITECTURE OF THE ADAPTIVE HYPERMEDIA SYSTEM AHA

The AHA system delivers HTML pages that consist of four parts:

- Each page starts with a generated header, which contains the definition of the style sheet with the link colouring scheme.
- The page body starts with a header, which the author creates once, and which is automatically included in every page.
- The page content is authored using a standard HTML editor. (It only consists of a part between the <body> and </body> tags.)
- Every page ends with a common footer, also created by the author and included automatically.

The pages, as they are delivered to the user, are assembled from these parts by means of a CGI-script (or optionally a Fast-CGI script).

Configuring the AHA software is done by setting a few variables in a supporting shell script. Variables to set include the course title, the directory on the Web server, the e-mail address of the author, the name of the Web server, etc. (All this is fairly straightforward.)

Each HTML page the author creates (optionally) starts with two comments:

- The first line indicates which Boolean expression on concepts needs to be true in order for a links to that page to be desired. Example:

  <!— requires readme and history but not (intro or definition) —>

- The second line indicates which concept(s) become "known" after reading this page:
  <!— generates history —>

From these lines in every page the system creates:

- A *dependency file*, containing pairs of page names and expressions. Each line indicates which expression must be true for links to that page to be desired.
- A *concept list*, containing all concepts. This list is used in a "setup" page: the user can change the user model by setting the value for each concept to true or false.

The AHA script recognizes three types of links:

- Links to an absolute URL (containing a ':') are called *external*. These links are coloured *red* by default. When an external link has been followed the colour changes to *gray* by default.
- Links to a relative URL, of class "unconditional" (or of no class) are considered to always be desired. The link class is changed to "good" and the link is initially coloured *blue* by default. If the link's destination is a page that was read before the link class becomes "neutral" and the colour changes *purple* by default.
- Links of class "conditional" are investigated by comparing the user model to the required concept-expression. If the expression evaluates to true the link class is changed to "good" or "neutral" (as in unconditional links) and the colour becomes *blue* or

purple by default. If the expression is false, the link class becomes "bad" and the corresponding colour is *black* by default.

If the link anchor consists of an image instead of text it is the colour of the image border that changes.

The user can change this colour scheme through a "setup" page. Also, a list of pages that have been read and a list of pages still to be read can be displayed. Multiple-choice tests are available in two versions: one that requires the user to answer correctly (and offers no explanatory feedback to wrong answers) and one that offers explanations of errors and does not require multiple tries. All these features are available through the same CGI-script that generates the pages. When a user "logs on" (by filling out a form) the system generates URLs to the pages that contain the name of the CGI-script that generates pages, the identity of the user, and the name of the requested page.

Although pages are individualized, they are currently not password protected. (The login form requires a password only to prevent users from accidentally logging on as another user. This may happen occasionally because student's use numbers as their user-id.)

The AHA software is freely available upon request. It is written entirely in Java (1.1) and should be easy to integrate into any Unix-based Web-server that supports CGI or Fast-CGI. (Porting to other platforms may be a bit tricky because

of the use of a few Unix utilities to generate the dependency file and concept list, and of a shell script to read environment variables, which is not possible in Java 1.1.)

Visitors are welcome to a few applications of this software: "http://wwwis.win.tue.nl/2L690/" is the address of the most recent version of the course on "Hypermedia Structures and Systems". "http://wwwis.win.tue.nl/2M350/" is the address of a course on "Graphical User-Interfaces" which uses the adaptivity mostly to set a user-preference for viewgraphs versus full text. "http://wwwis.win.tue.nl/IShype/" is the address of a small set of information pages (in Dutch) about writing a master's thesis in our Information Systems Group. It illustrates the use of AHA in an application which is not an on-line course.

A final, but important remark is that, like probably all adaptive hypermedia systems, AHA does not reduce the need for skilled authors. While adaptive hypermedia in general, and AHA in particular, offers the possibility to avoid certain usability problems caused by the user taking paths through a hyperdocument which the author did not foresee, it certainly does not eliminate usability problems automatically. Although we have not done this, it is possible to base an adventure game on AHA, using the adaptive features to make it *more difficult* to find your way by changing the contents of pages and the link structure in ways which the player does not expect.

CATEGORIZATION OF MULTIMEDIA

Multimedia may be broadly divided into linear and non-linear categories. Linear active content progresses often without any navigational control for the viewer such as a cinema presentation. Non--linear uses interactivity to control progress as with a video game or self-paced computer based training. Hypermedia is an example of non-linear content.

Multimedia presentations can be live or recorded. A recorded presentation may allow interactivity via a navigation system. A live multimedia presentation may allow interactivity via an interaction with the presenter or performer.

MAJOR CHARACTERISTICS OF MULTIMEDIA

Multimedia presentations may be viewed by person on stage, projected, transmitted, or played locally with a media player. A broadcast may be a live or recorded multimedia presentation. Broadcasts and recordings can be either analog or digital electronic media technology. Digital online multimedia may be downloaded or streamed. Streaming multimedia may be live or on-demand.

Multimedia games and simulations may be used in a physical environment with special effects, with multiple users in an online network, or locally with an offline computer, game system, or simulator.

The various formats of technological or digital multimedia may be intended to enhance the users' experience, for

example to make it easier and faster to convey information. Or in entertainment or art, to transcend everyday experience.

Enhanced levels of interactivity are made possible by combining multiple forms of media content. Online multimedia is increasingly becoming object-oriented and data-driven, enabling applications with collaborative end-user innovation and personalization on multiple forms of content over time. Examples of these range from multiple forms of content on Web sites like photo galleries with both images (pictures) and title (text) user-updated, to simulations whose co-efficients, events, illustrations, animations or videos are modifiable, allowing the multimedia "experience" to be altered without reprogramming. In addition to seeing and hearing, Haptic technology enables virtual objects to be felt. Emerging technology involving illusions of taste and smell may also enhance the multimedia experience.

VIRTUAL REALITY

Virtual reality is an artificial environment that is created with software and presented to the user in such a way that the user suspends belief and accepts it as a real environment. On a computer, virtual reality is primarily experienced through two of the five senses: sight and sound.

The simplest form of virtual reality is a 3D image that can be explored interactively at a personal computer, usually by manipulating keys or the mouse so that the content of the image moves in some direction or zooms in or out. More

sophisticated efforts involve such approaches as wraparound display screens, actual rooms augmented with wearable computers, and haptics devices that let you feel the display images. Virtual reality is often used to describe a wide variety of applications commonly associated with immersive, highly visual, 3D environments.

The development of CAD software, graphics hardware acceleration, head mounted displays, database gloves, and miniaturization have helped popularize the notion. In the book *The Metaphysics of Virtual Reality* by Michael R. Heim, seven different concepts of virtual reality are identified: simulation, interaction, artificiality, immersion, telepresence, full-body immersion, and network communication. People often identify VR with head mounted displays and data suits.

Virtual reality can be divided into:

- The simulation of a real environment for training and education.
- The development of an imagined environment for a game or interactive story.

Popular products for creating virtual reality effects on personal computers include Bryce, Extreme 3D, Ray Dream Studio, trueSpace, 3D Studio MAX, and Visual Reality. The Virtual Reality Modelling Language (VRML) allows the creator to specify images and the rules for their display and interaction using textual language statements.

Virtual reality can trace its roots to the 1860s, when 360-degree art through panoramic murals began to appear. An example of this would be Baldassare Peruzzi's piece titled, *Sala delle Prospettive*. In the 1920s, vehicle simulators were introduced. Morton Heilig wrote in the 1950s of an "Experience Theatre" that could encompass all the senses in an effective manner, thus drawing the viewer into the onscreen activity. He built a prototype of his vision dubbed the Sensorama in 1962, along with five short films to be displayed in it while engaging multiple senses (sight, sound, smell, and touch). Predating digital computing, the Sensorama was a mechanical device, which reportedly still functions today. Around this time, Douglas Englebart uses computer screens as both input and output devices. In 1966, Thomas A. Furness III introduces a visual flight stimulator for the Air Force. In 1968, Ivan Sutherland, with the help of his student Bob Sproull, created what is widely considered to be the first virtual reality and augmented reality (AR) head mounted display (HMD) system. It was primitive both in terms of user interface and realism, and the HMD to be worn by the user was so heavy it had to be suspended from the ceiling. The graphics comprising the virtual environment were simple wireframe model rooms. The formidable appearance of the device inspired its name, The Sword of Damocles. Also notable among the earlier hypermedia and virtual reality systems was the Aspen Movie Map, which was created at MIT in 1977.

The Programme was a crude virtual simulation of Aspen, Colorado in which users could wander the streets in one of three modes: summer, winter, and polygons. The first two were based on photographs—the researchers actually photographed every possible movement through the city's street grid in both seasons—and the third was a basic 3D model of the city. In the late 1980s, the term "virtual reality" was popularized by Jaron Lanier, one of the modern pioneers of the field. Lanier had founded the company VPL Research in 1985, which developed and built some of the seminal "goggles and gloves" systems of that decade. In 1991, Antonio Medina, a MIT graduate and NASA scientist, designed a virtual reality system to "drive" Mars rovers from Earth in apparent real time despite the substantial delay of Mars-Earth-Mars signals. The system, termed "Computer-Simulated Teleoperation" as published by Rand, is an extension of virtual reality.

IMPACT

There has been an increase in interest in the potential social impact of new technologies, such as virtual reality. In the new book (2011) *Infinite Reality: Avatars, Eternal Life, New Worlds, and the Dawn of the Virtual Revolution,* Blascovich and Bailenson review the literature on the psychology and sociology behind life in virtual reality. In addition, Mychilo S. Cline, in his book *Power, Madness, and Immortality: The Future of Virtual Reality*, argues that virtual

reality will lead to a number of important changes in human life and activity.

He argues that:

- Virtual reality will be integrated into daily life and activity, and will be used in various human ways. Another such speculation has been written up on how to reach ultimate happiness via virtual reality.
- Techniques will be developed to influence human behaviour, interpersonal communication, and cognition.
- As we spend more and more time in virtual space, there will be a gradual "migration to virtual space", resulting in important changes in economics, worldview, and culture.

Haptic Interfaces and Devices

Before the widespread use of computers in the work place, almost all human tasks involved the use of exquisite sensory-motor skills. By and large, computer interfaces have not taken great advantage of these fundamental human capabilities. With the exception of input devices such as the mouse, computer interaction relies on skills similar to those needed for using typewriters. Haptic interfaces may be viewed as an approach to address this limitation. It is thus possible to classify haptics in the area of computer-human interfaces. Unlike traditional interfaces that provide visual and auditory information, haptic interfaces generate mechanical signals

that stimulate human kinesthetic and touch channels. Haptic interfaces also provide humans with the means to act on their environment. We can therefore attempt to de ne haptic interfaces as being concerned with the association of gesture to touch and kinesthesia to provide for communication between the humans and machines.

The eld is inherently multidisciplinary and borrows from many areas, including robotics, experimental psychology, biology, computer science, systems and control, and others.

The Field of haptics is also growing rapidly. At present, the number of published papers with the word ª hapticº in them approaches a thousand a year, all disciplines included. Just 10 years back, there were only a few dozens.

The word haptics refers to the capability to sense a natural or synthetic mechanical environment through touch. Haptics also includes kinesthesia (or proprioception), the ability to perceive one's body position, movement and weight. It has become common to speak of the ª haptic channelº to collectively designate the sensory and motor components of haptics. This is because certain anatomical parts (in particular the hand) are unitary organs in which perceiving the world and acting upon it are activities that take place together. For example, grasping an unfamiliar object also involves exploring it actively with our hands.

Tactile and kinesthetic channels work together to provide humans with means to perceive and act on their environment.

The Function of Haptic Interfaces

The idea of using touch as a means of communication was popularized by Craig and Rollman (1999) and Sherrick (1985):

Our understanding of how simple patterns combine to yield the complexity needed to increase channel capacity for continuous information streams is still primitive.

It certainly still, is the case today. It is possible to discuss the function of a haptic interface by considering, on the one hand, an input device such as a computer mouse, and on the other hand a sheet of paper, viewed as a display device. Consider first, a blank sheet of paper: it contains little information (barring being a sheet of paper). The sheet is intended to support the information coded in the form of structure and discontinuities laid out on it by means of ink to change its respective properties. Next, consider a computer screen with graphics capabilities.

It can be programmed pixel by pixel to display information, also using structured discontinuities. Analogously, a computer mouse (or any other conventional input device) contains little mechanically-encoded information (just a fixed weight, shape, and rubbing properties). It is not programmable.

The step that was made to move from the sheet of paper to the graphics screen is analogous to the step made to move from a computer mouse to a haptic interface. Whereas the graphics screen can change its optical properties under

computer control, a haptic device can change its mechanical properties under computer control.

The ability to have programmable mechanical properties provides for a bidirectional exchange of energy, and therefore information, between the user and the outside world.

While the term haptic display is sometimes used, it is probably not the best suited because it emphasizes unidirectional information transfer like that of typical graphic displays (such as cathode ray tubes) and audio systems (like high fidelity music reproduction systems).

This fundamental difference can be understood by considering, in which a regular mouse is compared to a haptically enabled mouse with programmable mechanical properties. The arrows represent the direction of information flow. With a typical mouse, this is limited to a unidirectional input from the mouse to the computer. The user of a conventional mouse receives almost no information from its movements, although its friction and inertial properties may assist the user in performing skilful movements. The buttons on it are considerably richer: their mechanical detent and the small acoustical noise they produce inform the user that a discrete-state change has occurred. Nevertheless, the buttons are not programmable. The haptic mouse, on the other hand, can provide the user with programmable feedback based on the sense of touch, allowing a faster and more intuitive interaction with the machine. In general, haptic

interfaces attempt to make the information flow non--zero to the user, as in the example of moving from the blank sheet of paper to the graphics screen. This can be further explained from an information-theoretic view point: consider a channel in which x is the input and y is the output. In a lossless channel, the entropy of x given y, $H(x|y)$; is zero: the output uniquely species the input. In a useless channel $H(x|y) = H(x)$; the knowledge of the output says nothing about the input, x and y are independent. This is the case of ordinary input devices such as a mouse. They can be moved here and there, but the mechanical signals they produce are unrelated to the state of the machine; they are useless as a channel.

This distinction is also most apparent if one considers that visual, auditory, olfactory and vestibular signals can be recorded and replayed (people watch movies, listen to audio recordings, or have machine-controlled rides in vehicle simulators). On the other hand, recording and replaying kinesthetic and tactile sensations must involve user movement, except possibly for the display of vibro-tactile sensations.

All objects, natural or manufactured, fall into one of the two categories. They are inert or active, roughly speaking, inanimate or animate. Inert objects can only dissipate mechanical energy, while active ones may supply some energy. Thus, there can be two kinds of haptic devices, conventionally termed passive or active, but they all share the property of being programmable.

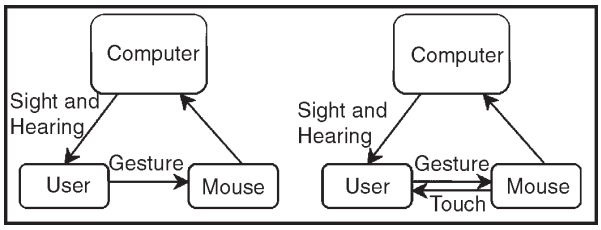


Figure : Distinguishing Feature of Haptic Interfaces is the Simultaneous Exchange of Information between the user and the Machine.

Passive devices are often designed to have programmable dissipation, as a function of position or time. To this category belong the devices having controllable brakes. Another category of passive devices consists of those that rely on non-holonomic constraints (constraints involving velocity). Yet another possibility is to modify the elastic behaviour of an element to become harder or softer. The programmability of passive devices comes from the possibility of modifying these constraints under computer control.

As for active devices, the energy exchange between a user and the machine is entirely a function of the feedback control which is applied. Then two categories arise: either the actuators act as a force source (a variable of effort), and position is measured, or the actuators act as a position source and then force is measured.

The former case is termed isotonic (force does not change with position) while the latter is called isometric (position does not change with force). Closing the loop around an isotonic device corresponds to specifying an impedance to produce a simulation, and the other case corresponds to an admittance.

It is often desired that active devices be used to reproduce synthetic environments such that these environments are passive, for example to simulate a surgical act. How well this is achieved is, in fact, a particular challenge. Conversely, the ability to create a temporally active simulation can be quite useful to increase the flow of information between the machine and the user. For example, simulating the behaviour of the steering wheel of a race car requires the interaction to be active.

Passive devices cannot create active simulations. Finally, it must be noticed that the possibility exists in unstable interactions with passive environments (a drum roll, for example) if the conditions are such that the user can supply the energy needed to sustain the oscillation.

To summarize, regardless of the approach to their design, bi-directionality is the single most distinguishing feature of haptic interfaces, when compared with other machine interfaces, and this observation explains in part why they create a strong sensation of immediacy. A haptic device must be designed to 'read and write' from the human hand (or foot, or other body parts).

This combined read and write property may explain why the first applications of this technology involved fast-paced interactivity. As it turns out, the read part has been extensively explored, and a great many types of devices already exist (knobs, keys, joysticks, pointing devices, *etc.*). The 'write' part is comparatively more difficult to achieve.

More specifically, the function of the haptic interface is to recreate constitutive properties: relationships between variables of flow and effort. Haptic interfaces are concerned with the technical means needed to make use of the extensive and exquisite capabilities of human touch, including proprioception, motor control, *etc.*. To achieve this, they must be programmable devices capable of recreating mechanical phenomena of perceptual relevance and functional importance.

It is also important to recall that haptics, as a technological niche, inherits much from teleoperation, which can be considered as its mother discipline. In a sense, haptics is like teleoperation, but the remote slave system is purely computational, *i.e.*, virtual. The virtual aspect has been helped greatly by the tremendous progress in computing and telecommunications. Plainly speaking, one replaces the teleoperator slave by a computer, thereby creating the possibility of virtuality: the slave and the world are computational, and thereby can be imaginary, or not restricted by normal physical constraints (as a matter of fact, virtual reality simulations rarely are). Driven by this, haptics became an independent technological niche in the past decade.

There is another relationship to robotics. Haptic devices can be regarded as robots, however, as robots having a very special function or task, that of interacting with humans.

This occurs mostly through the hand, but also via other anatomical regions, often, but not always, limbs and extremities. Thus, many robotic problems are relevant to haptic interfaces and *vice-versa*.

Examples of Applications

Graphical user interfaces (GUIs) have demonstrated that interactive presentation of data does not have to imitate reality, not even remotely. Being suggestive is what matters the most. Pull-down menus and scrolling slider bars cannot be found anywhere, but on computer screens; real paper file folders are not infinitely recursive, and so on.

The same holds for haptic interfaces. For example, the interaction forces that we experience when moving objects occur when these objects contact one another (except with magnets and inertial effects). With haptics, we can perfectly suggest a relationship between two distinct objects by creating a mutual interaction force, even if they are visually presented as being disconnected. Alternatively, some applications demand a significant amount of fidelity with respect to the actual tasks being recreated.

In other words, haptic interfaces can be designed to provide for a literal reproduction of the phenomena that occur during actual manipulation. This is what is called quest for realism in computer graphics. The training of sensory-motor skills such as surgical ability is one example in which the need for realism exists.

It is useful to keep these distinctions in mind while surveying the applications of haptic devices. An interesting aspect of this technology is that some applications are presently part of the commercial activities, good many of them at the precompetitive stage.

For example, one of the earliest researched application of haptic interfaces was the layering of haptic cues on conventional graphical interfaces. Currently, this has reached the consumer arena.

In the following subsections, applications are surveyed in terms of activity areas. The research is now so intense that only a few references will be included.

Force-reflecting Input Devices for use with Graphical User Interfaces

As mentioned, one of the first researched applications of haptic interfaces was the enhancement of existing graphical user interfaces. Elements of these GUIs (windows, pushbuttons, pull-down menus, words of a text, drawings) can be rendered mechanically.

Human factor studies indicate improvements in routine computer interactions in speed, precision, and reduction of fatigue. More specifically, cases that benefit from the enhancement of designation tasks (point and click, dragging, snap-to and so on) include drawing packages, text editors, spreadsheets, hypertext navigation, and operating system interfaces.

In the latter case, haptic cues can further be used to represent topological relationships in terms of importance: strength, recency, or urgency. Haptic cues may also be used to provide for interactive annotations. For example, haptic tabs can be inserted for efficient retrieval in large documents and databases by specific users. They can also provide for efficient multi-author document editing.

Games

Modes of interaction and the sense of user immersion are greatly enhanced by applying force feedback to the player. Dexterity games available earlier in ® xed form can be made in® nitely programmable: placing, balancing, hitting and bouncing.

As well, many opportunities exist for educational games. It is possible to illustrate concepts in dynamics, kinematics, magnetism, waves, flows and many other physical phenomena, or in mathematics and anatomy. Other kinds of games include combinatorial mind games, puzzles, and guess games that include visual and mechanical constraints, as well as most situation games. In the latter case, force feedback is already at the commercial stage, to assist in driving, piloting, exploring, and so on.

Multimedia Publishing

Current multimedia and hypertext applications include text, sound, images, and video. For lack of appropriate devices

so far, haptics has been ignored as a medium of communication. One could envision mechanical documents. For example, a new form of document that would include movement which can be experienced visually (video), auditively (spatialization), and also haptically. This raises the question of authoring tools (such as Immersion Studioe) and their necessity for the design of haptic sensations. Material properties can also be conveyed. A frequently mentioned application of this capability is the creation of online catalogues with haptic feedback. These would however bene t greatly from the development of practical, distributed tactile displays, which are not yet available.

Scientific Discovery

Data display was in fact one of the earliest applications of haptics, with the molecule docking project. Other display applications include: multidimensional maps, data mining in geology (or in related, applied fields such as oil and gas prospecting), remote sensing, and the display of fields and flows.

An attractive property of haptics is the ability to convey the existence of small details, which typically clutter the graphical presentation of data, while minimizing the need to zoom in and out.

Projects exist to use haptics to enhance the human interface of imaging instruments such as scanning, tunnelling, and atomic force microscopes.

Arts and Creation

Musicians and visual artists are increasingly using computers. However, creators often prefer to use their hands as directly as possible (as in sketching). Haptic communication with computers opens completely new opportunities. In music, advances in real-time synthesis tools increase the demand for interactive controllers which are presently mostly confined to the existing MIDI Fixed interfaces. In the graphic arts and design, especially the creation of animation, much activity is under way.

Editing Sounds and Images

Haptics can provide for rapid access, and browsing through sound and video documents for editing, splicing, and mixing.

Vehicle Operation and Control Rooms

In stressful, and fast-paced environments, haptic communication can be used to alleviate visual load. Haptic controllers are already commercially available in cars (iDrivee equipped BMW 7 series and Rolls-Royce Phantom). With a single programmable rotary controller, users can navigate menus, scroll lists, control sliders, *etc.* by experiencing distinctive haptic sensations for each widget. In this fashion a single controller serves as the input for a multitude of functions, with the haptic feedback serving to make the interface more intuitive and natural to use. Similarly, applications are ending their way into control rooms (air traffic control, nuclear).

Engineering

In computer-aided design, designers can experience minute details with their hands, such as wanted or unwanted artefacts of a design which are cumbersome to display visually. Simulated structures can be manually tested, assessed and debugged.

Manufacturing

In manufacturing, many opportunities exist. For example, haptics can assist design for assembly, in terms of reducing the need for prototyping, and as well as for rapid prototyping. It is also possible to assess human maintainability of complex systems before they are built. Programming of complex manufacturing devices such as multi-axis, numerically-controlled machines or robots can be facilitated.

Telerobotics and Teleoperation

As commented earlier, teleoperation is the mother discipline. Haptic devices are used in supervisor control modes such as teleprogramming, predictive displays, *etc.* Teleoperation systems still have a need for highquality manual controllers.

Education and Training

Dangerous systems or systems with very limited availability (*e.g.*, surgery patients) can be simulated using haptics for training purposes. Surgical training, in particular, is the subject of intense research. Other opportunities include the training of sensorymotor skills in general.

Rehabilitation

Applications include the improvement of working conditions for visually impaired people, and better interfaces to alleviate motor system impairment.

Scientific Study of Touch

Last but not the least, the availability of haptic devices makes it possible to study the haptic channel in humans (and other species) in exciting and perhaps earlier impossible ways. Haptic devices allow the creation of special, computer-controlled stimuli which are used in studies that explore the sense of touch functions. This is analogous to the use of programmable sound cards and computer graphics in human hearing and vision studies. In turn, the knowledge gained of the haptic function contributes to the development of new haptic interfaces and applications.

PRINCIPLE OF OPERATION

Tactile Sensations and the Kinesthetic Sense

In general, tactile sensations include pressure, texture, puncture, thermal properties, softness, wetness, friction-induced phenomena such as slip, adhesion, and micro failures, as well as local features of objects such as shape, edges, embossings and recessed features. In addition, vibrotactile sensations refer to the perception of oscillating objects in contact with the skin. This is appreciated by attending to the sensations experienced while holding a sheet

of paper where the three main functions of touch are used. The grade and texture of the paper are perceived by gently rubbing it (identify material), and its border is found by exploring the edges (identify shape). Speaking loudly near it causes vibrotactile sensations to be experienced (rapid oscillations). This distinction appears to correspond to specific mechanoreceptors and neural codes.

Several kinds of receptors have been found to mediate tactile sensation in the skin or in the subcutaneous tissues; consequently, it is customary to designate the skin as the seat of this sense (A very large organ, indeed; it covers roughly 2m2; it weighs about 5m2kg, its innervation is up to hundreds of receptors per square centimetre). The biophysical attributes of the skin vary tremendously with the parts of the body it covers. The tactile system occupies a great part of the afferent pathways of the peripheral nervous system, as well as a significant part of the central nervous system.

Proprioceptive, or kinesthetic perception, refers to the awareness of one's body state, including position, velocity and forces supplied by the muscles through a variety of receptors located in the skin, joints, skeletal muscles, and tendons. Together, proprioception and tactile sensations are fundamental to manipulation and locomotion.

Human Perception and Haptic Interfaces

When we watch a high-resolution digital movie, we do not perceive a series of still pictures that are presented in

sequence, nor do we apprehend an array of coloured pixels. Instead, we perceive a visual scene that is strikingly close to everyday visual experiences. This is possible because the temporal sensitivity of the human visual system is not sufficient to detect the fast presentation of the movie frames nor it can resolve individual pixels. This is an example of how the architecture and limitations of a perceptual system can be exploited to build engineering systems that elicit realistic, complex perceptual experiences. Examples of these systems include graphics screens, TV, tape recorders, audio synthesizers, right simulators, and, not surprisingly, haptic interfaces. The sense of touch differs from the visual system in that it requires update rates significantly higher than those needed to display video (1 kHz or more is required to satisfy the signal representation theorem and to minimize interaction delay). The physical interface that enables user-machine interaction can also have a great deal of variability. It is in general very difficult to produce perfectly realistic haptic interaction. Fortunately, even while using an imperfect haptic device, a user quickly adapts to its interference, ignores its imperfections, and naturally associates the device's mechanical stimulation to everyday experiences such as perceiving surface texture and shape of the objects through touch. Also, when haptic interfaces are combined with graphic displays, the user readily associates adequate haptic stimulation to a graphically displayed object. It is not unusual

to perceive the haptic sensations as if they occurred at the graphic display itself. This happens even though what is seen and what is haptically felt may occur in completely different spatial locations (*i.e.*, the haptic interfacemay be on a table alongside the graphic display where the objects are viewed).

However, if the imperfections in the haptic device are too obtrusive, the sense of haptic realism breaks down. This is analogous to what happens if a movie projector slows down to one frame per second: the movie turns into a series of stills. The quality of the illusory haptic experience as with any other technological devices is a function of the interplay between the user's perceptual system and the intrinsic technical qualities of the interfaces, such as dynamic range, resolution, and appropriateness of the signals being generated.

Components

A complete haptic interface usually includes one or several electromechanical transducers (sensors and actuators) in contact with a user in order to apply mechanical signals to distinct areas of the body, and to measure other mechanical signals at the same distinct areas of the body. Whether these signals should refer to forces, displacements, or a combination of these and their time derivatives, is still the object of debate.

Another important part of a complete interface is the computational system driving the transducers. The function

of this computational system is to provide haptic rendering capabilities, which are analogous to the visual rendering functions of common graphic systems. Haptic rendering, however, stresses the bidirectional exchange of information between the interface and the user. The computational task in haptic rendering is to generate signals that are relevant to a particular application. Several approaches exist for creating such haptic feedback. For example, a model may be used to represent an environment, and its equations solved computationally to and forces as a function of displacements and their derivatives (or *vice-versa*). The model may be developed from First principles, or parameterized to represent only certain desired aspects.

The characteristics of the human haptic system allow in some cases the use of simplifixed physical models to render haptic objects that compete in realism with actually physical objects. Another possibility is the recording of ground data and replaying it as a function of state variables and/or time. The computational task can range from the light (translation of a GUI into a force Field) to the intractable (for example, objects described by continum mechanics). So many possibilities exist, that this should be the topic of a separate discussion. This computational task is usually mapped onto a data processing hierarchy consisting of several computing units and communication channels. The engineering problem is to map the computational task onto the computational

hierarchy so that no constraint is violated in terms of update rates and data transfer rates. For a recent survey of haptic rendering.

DEVICES: CONCEPTS AND EXAMPLES

We examine a cross-section of the existing devices selected to illustrate the diversity of design niches being explored and the vitality of the activity in this Field (a complete survey would be much too long). We also comment on prominent features of these designs. Specific technical requirements of devices are reviewed in Hayward and Astley. In this section, the description of entire families of haptic devices, completely passive devices, and foot contacting devices, and distributed tactile displays, unfortunately had to be omitted, despite significant activity in all these areas.

Programmable Keyboard

One of the most documented examples of a multiple force-feedback implementation is the Clavier ReÂ troactif Modulaire, a project headed by Cadoz, which consists of a piano-like Lorentz-actuated keyboard providing computer-driven force feedback for each of its 16 keys and directed at musical creation research.

Exoskeletons

The exoskeleton devices developed by Bergamasco and co-workers incorporate many observations regarding the human biomechanics. To achieve wearability, the system uses a

variety of techniques including motor remotizing, sophisticated cable routing, and friction reduction by feedback. Being worn, the device body interface is partly bracing and partly held. Many other devices have been designed by this laboratory.

Desktop Scale

A six degree of freedom device is the result of the pioneering work of Iwata, who advocated the design of small devices. It adopts a parallel platform design supported by three gear driven five bar linkages. The result is a compact and powerful table top design. The initial design is described by Iwata; several versions have been developed thereafter.

Grasping

Howe designed a double, two degree of freedom apparatus intended for two-finger grasping studies. It uses direct-driven, parallel linkages resulting in a very wide dynamic range. The user's fingers interact unilaterally with the device on the inner side of boxes, allowing precision grip.

Point Interaction

The Phantome has become a popular device in research laboratories. There are several variants, but generally a stylus is grasped, or a thimble braces the user's finger. There are three actuated degrees of freedom and three sensed orientations. A typical configuration has a work volume of 2.7 dm3. A key design aspect is a capstan drivewhich avoids

the use of gears and makes it possible to amplify the torque of small DC motors with a concomitant increase of damping and inertia. The initial design is described in Massie and Salisbury and is commercially available.

High Power Devices

Colgate and his group have created number of devices that were used for studies in control. One early design, described in Millman *et al.* , features high power and bandwidth for tool use simulation. This group also investigates a number of designs in the family of passive devices. Other high power devices were developed by Ellis *et al.* and by Hannaford's group.

Augmented Mice

An innovative system is described by Akamatsu *et al.*. It has the general shape and function of a computer mouse, but includes two haptic feedback features. One is an electromagnetic braking system which provides programmable dissipative friction forces, and the other is a transducer to provide vibro-tactile sensations.

Joystick

The force-feedback two degree of freedom joystick described by Adelstein and Rosen is one example of a device designed with specific performance figures in mind. Many other force-feedback joysticks were designed for various applications.

Separate Carrier

Luecke *et al.* describe a design concept whereby individual high-fidelity and direct-driven force feedback devices act on the fingers of the hand and are moved about by a large workspace stiff robotic carrier.

Horizontal Planar Workspace

The Pantograph has been made in many variants, which were characterized by simplicity and a uniform peak acceleration ratio contained in a 3 dB band. It has two actuated degrees of freedom in the horizontal plane, provided by a stiff parallel linkage driven without transmission. The finger rests on the interface, resulting in a unilateral interaction. One variant is operated by the thumb and its in the hand. Larger ones have a working area of 1.6 dm2. An industrial version, the PenCat/Proe, has a sensed 2.5 cm vertical movement, passively actuated by an elastic return.

VRML (VIRTUAL REALITY MODELLING LANGUAGE)

VRML (Virtual Reality Modelling Language) is a language for describing three-dimensional (3D) image sequences and possible user interactions to go with them. Using VRML, you can build a sequence of visual images into Web settings with which a user can interact by viewing, moving, rotating, and otherwise interacting with an apparently 3D scene. For example, you can view a room and use controls to move the

room as you would experience it if you were walking through it in real space.

To view a VRML file, you need a VRML viewer or browser, which can be a plug-in for a Web browser you already have. Among viewers you can download for the Windows platforms are blaxxun's CC Pro, Platinum's Cosmo Player, WebFX, WorldView, and Fountain. Whurlwind and Voyager are two viewers for the Mac.

Virtual Reality Modelling Language pronounced as vermal and its started to use in year 1995. Initially, this is also called by a name Virtual Really Markup language. This is a standard language which is used for interactive simulation with in the world wide web. It allows to represent 3-dimensional interactive vector graphics and a "virtual worlds" networked via internet and hyper linked with the world wide web. VRML is text file format Which able to design vertices and edges for 3D polygon with any of the specified colour. It is also used to perform UV mapping which create the 3D models of the 2D image. VRML files are commonly called with a name of "worlds" and have.wrl extension. VRML worlds contain the text format so that it easily compressed by using the gzip and compressed file transfer through internet more quickly.

VRML architecture includes the Input processor, Simulation processor, Rendering Processor and World Database. Input Processor is used to control the input information to the computer *e.g.*, Keyboard, mouse, 3D

position trackers and a voice recognition system. Simulation Processor is the heart of Virtual Reality system. It takes the user input along with a number of task that are programmed with it and determine the actions that will take place in the virtual environment. Rendering processor is used to create sensation that are output to the user. World Database is also known as World Description Files. It stores the objects that describe the actions of all those objects. VRML is successfully used in various fields like Entertainment, Medicine, Manufacturing, Education and training. In Entertainment, it is helpful in designing a more attractive and exciting virtual environment. In medicine, used to perform the practice test of surgery, surgery on remote patient and teach new skills in a safe, controlled environment. In manufacturing, used to make a low cost and highly efficient products. In education and training used to driving, flight, ship and tank simulators.

The VRML or Virtual Reality Modelling Language is a modelling language for interactive graphics specification. VRML was originally known as Virtual Reality Markup Language. As the name implies it was used just like other markup languages to be used among many platform. VRML is platform independent and can be used by downloading the required files. VRML is a graphics modelling and specification language for interactive animations with high originality. VRML is used to specify the graphics properties

using VRML file. VRML files are text description of various geometries. It allows web developers to design three dimensional space and objects in full range of effects and properties. VRML is used to provide special texture, lighting effects and animations. This enables the users to feel the virtual reality on the web.

VRML (Virtual Reality Modelling Language) is used to create a virtual world and present it to users via the Internet. It shows a realistic looking virtual world without using a special set of glasses or virtual reality helmet.

VRML uses vector graphics to present a 3D world to users using a 2D screen. As the user chooses to change their vantage point, the world view is recalculated and then presented. VRML (Virtual Reality Modelling Language) stores the vector information for the locations of objects, their edges and vertices. Information about the object such as its colour and texture are also stored in the VRML file, typically with a.wrl extension. When the user moves so that the object is deemed to come into their view, its shape and visual effects are generated.

A wide range of 3D file formats can be converted to VRML. For example, AutoCAD DXF and Autodesk 3D studio files can be converted to VRML (Virtual Reality Modelling Language) files. Converters also exist to turn IGES and Alias wire files into VRML. Users converting files into VRML (Virtual Reality Modelling Language) need to be careful to use binary

file conversion. Texture information is kept in an.rgb format when added to the VRML presentation.

VRML (Virtual Reality Modelling Language) has been used in gaming. VRML has been used extensively in academia such as the display of complex molecules, the demonstration of proposed structures for folding proteins and showing changes to materials over time. VRML has also been used to build interactive engineering models of transportation grids and structural designs.

The two most commonly VRML (Virtual Reality Modelling Language) viewers are Worldview by Intervista and WebSpace by SGI. Cortona3D Viewer is a browser plug-in for Internet Explorer.

FORMAT

VRML is a text file format where, *e.g.* vertices and edges for a 3D polygon can be specified along with the surface colour, UV mapped textures, shininess, transparency, and so on. URLs can be associated with graphical components so that a web browser might fetch a webpage or a new VRML file from the Internet when the user clicks on the specific graphical component. Animations, sounds, lighting, and other aspects of the virtual world can interact with the user or may be triggered by external events such as timers. A special Script Node allows the addition of Programme code (*e.g.* written in Java or JavaScript (ECMAScript)) to a VRML file.

VRML files are commonly called “worlds” and have the *.wrl extension (for example island.wrl). VRML files are in plain text and generally compresses well using gzip which is useful for transferring over the internet more quickly (some gzip compressed files use the *.wrz extension). Many 3D modelling Programmes can save objects and scenes in VRML format.

4

Graphical Animation

In computer graphics, the process of updating a graphical display so that it changes over time is called animation. Implementing animation typically involves displaying an initial version of the picture and then changing it slightly over time so that the individual changes appear continuous from one version of the picture to the next. This strategy is analogous to classical film animation in which cartoonists break up the motion of the scene into a series of separate frames. The difference in time between each frame is called a time step and is typically very short. Movies, for example, typically run at 30 frames a second, which makes the time step approximately 33 milliseconds. If you want to obtain smooth motion in Java, you need to use a time step around this scale or even faster.

A SIMPLE EXAMPLE OF ANIMATION

The easiest way to animate graphical programmes is to include a loop in your run method that updates the picture from one frame to the next and then pauses for the duration of the time step. An example of this style of animation appears, which moves a GLabel across the screen from right to left, just the way the headline displays in New York's Times Square do.

Code to Move Text Across the Screen

```
  /*
  * File: TimesSquare.java
  * ——————————
  * This programme displays the text of the string HEADLINE
on the
  * screen in an animated way that moves it across the
display
  * from left to right.
  */
  import acm.graphics.*;
  import acm.programme.*;

  public class TimesSquare extends GraphicsProgram {

  /** Runs the programme */
    public void run() {
        GLabel label = new GLabel(HEADLINE);
        label.setFont("Serif-72");
            add(label, getWidth(), (getHeight() +
label.getAscent()) / 2);
        while (label.getX() + label.getWidth() > 0) {
            label.move(-DELTA_X, 0);
            pause(PAUSE_TIME);
        }
    }

  /* The number of pixels to shift the label on each cycle
*/
    private static final int DELTA_X = 2;
```

```
  /* The number of milliseconds to pause on each cycle */
    private static final int PAUSE_TIME = 20;
  /* The string to use as the value of the label */
    private static final String HEADLINE =
        "When in the course of human events it becomes
necessary " +
      "for one people to dissolve the political bands which
" +
       "connected them with another...";
  }
```

The TimesSquare programme begins by creating a GLabel object and positioning it so that it is centred vertically in the window. Its starting point in the horizontal dimension, however, is just at the right edge of the canvas, which means that the entire label is outside the visible area of the canvas. The animation is accomplished by the following lines:

```
  while (label.getX() + label.getWidth() > 0) {
    label.move(-DELTA_X, 0);
    pause(PAUSE_TIME);
  }
```

This code loops until the label has moved entirely past the left edge of the display, shifting it DELTA_X pixels to the left on every time step. The call to pause(PAUSE_TIME) inside the loop causes the programme to suspend operation for PAUSE_TIME milliseconds. This call is necessary to achieve the effect of animation, because computers run so quickly that the label would instantly zip off the left side of the window if you didn't slow things down.

Bouncing a Ball

A slightly more sophisticated application of animation appears. This programme bounces a ball around the walls of the graphics window and forms the foundation for such

classic video games as Pong or Breakout. Because a static picture in this text would offer little insight into how such an animated programme works, it is useful to run this as an applet. If you are reading this tutorial on the JTF web site, you can bring up the applet in a separate window by clicking on the applet marker in the caption, but you can also run any of the applets from the demo site at http://jtf.acm.org/demos/index.html.

Programme to Bounce a Ball off the Boundaries of the Canvas

```
  /*
  * File: BouncingBall.java
  * ———————————
  * This file implements a simple bouncing ball using the
run method
  * to drive the animation.
  */

  import acm.graphics.*;
  import acm.programme.*;

  public class BouncingBall extends GraphicsProgram {
  /** Initialize the ball and its velocity components */
     public void init() {
      ball = new GBall(BALL_RADIUS);
      add(ball, getWidth() / 2, getHeight() / 2;
      dx = 2;
      dy = 1;
     }

  /** Run forever bouncing the ball */
      public void run() {
       waitForClick();
       while (true) {
          advanceOneTimeStep();
          pause(PAUSE_TIME);
       }
```

```
      }

  /* Check for bounces and advance the ball */
      private void advanceOneTimeStep() {
        double bx = ball.getX();
        double by = ball.getY();
       if (bx < BALL_RADIUS || bx > getWidth() - BALL_RADIUS)
dx = -dx;
       if (by < BALL_RADIUS || by > getHeight() - BALL_RADIUS)
dy = -dy;
        ball.move(dx, dy);
     }
  /* Private constants */
      private static final double BALL_RADIUS = 10;
      private static final int PAUSE_TIME = 20;

  /* Private instance variables */
      private GBall ball;   /* The ball object */
       private double dx;      /* Velocity delta in the x
direction */
       private double dy;     /* Velocity delta in the y
direction */
  }
```

The code uses the GBall class to create a ball whose reference point is at the centre. Doing so makes the geometric calculation simpler when checking whether a bounce occurs because all four edges can be treated symmetrically. The programme code is also divided between the init method, which creates the ball and adds it to the window, and the run method, which runs the animation. The code for the run method is

```
  public void run() {
      waitForClick();
      while (true) {
          advanceOneTimeStep();
          pause(PAUSE_TIME);
      }
  }
```

which is almost precisely the paradigmatic for an animation loop. The new statement is the call to the waitForClick

method, which is implemented by GraphicsProgram and suspends the programme until a mouse click occurs in the graphics canvas. This call means that the programme does not start up immediately, but instead waits for a mouse click before proceeding. The code that implements the underlying physics of the animation appears in the private method advanceOneTimeStep. This method checks to see whether the ball has reached one of the edges of the canvas, in which case it changes the sign of the appropriate component of the ball's velocity, which is stored in the variables dx and dy. It then moves the ball by those displacements to update its position on the display.

Simulating Randomness in Animations

As written, the bouncing ball programme from the preceding section is altogether too predictable. The ball begins with a constant velocity and then makes perfectly reflective bounces off the edges of the canvas, tracing the same trajectory each time. Many animated programmes will involve some kind of random behaviour, and students will quickly want to know how they can implement random processes in their own code.

Although it is certainly possible to use either the Math.random method or the Random class in java.util for this purpose, there are a couple of pedagogical advantages to using the RandomGenerator class in theacm.util package instead:

1. The name of the class emphasizes that a RandomGenerator object is a *generator* for random values and not a random value in itself. When students use the Random class, they are much more likely to create a newRandom instance for each value they wish to generate.
2. The RandomGenerator class offers several additional methods that are often much easier to use than those in the base class.

Useful Methods in the RandomGenerator Class

Factory Method

```
  static RandomGenerator getInstance()
    Returns a standard random generator.
   Methods inherited from the Random class in java.util

  int nextInt(int n)
     Returns a random integer chosen from the n values in
the range 0 to n - 1, inclusive.
  double nextDouble()
      Returns a random double d in the range
0 ¾ d < 1.
  void nextBoolean()
    Returns a random boolean that is true approximately 50%
of the time.
  void setSeed(long seed)
     Sets a "seed" to indicate a starting point for the
pseudorandom sequence.

   Additional methods defined by RandomGenerator

  int nextInt(int low, int high)
     Returns a random integer in the specified range
(inclusive).
  double nextDouble(double low, double high)
    Returns a random double in the specified range.
  boolean nextBoolean(double p)
```

```
    Returns a random boolean that is true with probability
p (0 = never, 1 = always).
  Colour nextColor()
    Returns a random opaque colour.
```

The conventional pattern for using the RandomGenerator class is to declare and initialize an instance variable to hold the generator using the line

```
private RandomGenerator rgen = RandomInteger.getInstance();
```

Once this declaration is made, every method in this class can then generate new random values by invoking the appropriate method on the rgen variable. For example, you could use this strategy in the BouncingBallprogram to initialize each velocity component of the ball to a random value between –3 and 3:

```
dx = rgen.nextDouble(-3, 3);
dy = rgen.nextDouble(-3, 3);
```

The RandomShapes programme makes more extensive use of the facilities of the RandomGenerator class. The programme generates ten shapes and positions them on the canvas using randomness in each of the following ways:

- The shapes are randomly chosen to be rectangles, ovals, or stars. The stars are represented internally using the GStar class define.
- The shapes are given a random size that ranges between MIN_SIZE and MAX_SIZE in each dimension.
- The shapes are positioned randomly on the canvas subject to the condition that the entire shape must fit inside the boundaries.
- The shape is filled in a random colour.

Most of RandomShapes programme is reasonably straightforward, but there are nonetheless a few aspects of the code that are easier to understand with some additional explanation:

- The code for the run method includes a while loop that allows the user to generate a new set of shapes by clicking the mouse. The waitForClick method was introduced earlier in the chapter in the discussion of the bouncing ball programmes and simply waits for a mouse click.
- The calculation of the random coordinate positions seems slightly more complex that necessary. At first glance, it would seem as if one could ensure that the entire figure was inside the canvas by writing

```
double x = rgen.nextDouble(0, getWidth() - gobj.getWidth());
double y = rgen.nextDouble(0, getHeight() - gobj.getHeight());
```

- While that code would be sufficient for the GRect and GOval objects that have their reference point in the upper left corner, it doesn't work for figures like GStar for which the reference point is inside the figure. ThegetBounds method returns the actual bounding box of the figure, which means that gobj.getBounds().getX() returns the actual *x* coordinate of the left edge of the figure. You can make sure that the figure fits on the screen by adjusting the coordinates to compensate for the shift in origin.

The RandomGenerator class from the java.util class has applications in a wide variety of contexts beyond graphical animation.

In our experience it is far and away the most widely used class in the java.utilpackage.

Programme to Generate Random Shapes

```
  /*
   * File: RandomShapes.java
   * ———————————
    * This file creates ten boxes, ovals, and stars at
random locations
   * on the screen, pausing for a suitable interval between
each one.
   */

  import acm.graphics.*;
  import acm.programme.*;
  import acm.util.*;

  public class RandomShapes extends GraphicsProgram {

  /** Runs the programme */
      public void run() {
          while (true) {
              for (int i = 0; i < NOBJECTS; i++) {
                  addOneRandomShape();
                  pause(PAUSE_TIME);
              }
              waitForClick();
              removeAll();
          }
      }
  /* Adds a random shape to the canvas */
      private void addOneRandomShape() {
          GObject gobj = createRandomShape();
          gobj.setColor(rgen.nextColor());
              if (gobj instanceof GFillable) ((GFillable)
gobj).setFilled(true);
              double x = rgen.nextDouble(0, getWidth() -
gobj.getWidth())
```

```
                - gobj.getBounds().getX();
        double  y  =  rgen.nextDouble(0,  getHeight()  -
gobj.getHeight())
                - gobj.getBounds().getY();
          add(gobj, x, y);
      }

  /* Generates a random shape whose reference point is the
origin */
      private GObject createRandomShape() {
         double width = rgen.nextDouble(MIN_SIZE, MAX_SIZE);
         double height = rgen.nextDouble(MIN_SIZE, MAX_SIZE);
          switch (rgen.nextInt(3)) {
           case 0: return new GRect(width, height);
           case 1: return new GOval(width, height);
             case 2: return new GStar(width);
           default: throw new ErrorException("Illegal shape
index");
          }
      }

  /* Private constants */
      private static final int NOBJECTS = 10;
      private static final int PAUSE_TIME = 1000;
      private static final double MIN_SIZE = 25;
      private static final double MAX_SIZE = 150;
  /* Private instance variables */
              private    RandomGenerator    rgen    =
RandomInteger.getInstance();
  }
```

CYCLIC ANIMATION

THE DELIGHT OF VISUAL RHYTHMS

Human beings find aural and visual rhythms immensely satisfying.

In fact we are pattern seekers and take great pleasure in notions of 'return' and the familiar. Animators have taken advantage of this human disposition.

Cycles - Cutting Down the Work

There are lots of tricks that animators constantly use to cut down the amount of work to be done. Cycles of repeating action are just one of these ways - and whenever animators find an opportunity to include a cycle in a sequence, you can bet they will seize upon it.

Some of the very early cartoons were almost entirely based on cyclic actions, especially when it was discovered that animation could echo the rhythmic patterns found in music. Walt Disney's 1928 'Steamboat Willie' was the first sound cartoon to amaze audiences of the day with its close synchronism between image and sound. This relationship was exploited to the hilt, (giving rise to the term 'Mickey Mousing' - a sound track which follows exactly what the image is doing) as was the use of cyclic animation which took its cues from the repeated phases and beats of the musical score.

Types of Cyclic Action

Cycles can be cyclic in nature, that is, the artwork is used in order 1,2,3,4 followed by exact repeats of that order again 1,2,3,4 etc. This type of cycle is useful for representing things like a wheel spinning. Cycles may also oscillate. That is the artwork is used in order 1,2,3,4 but then the artwork is used in reverse order 4,3,2,1 to return to the start position again etc. - like the motion of a clock pendulum. Or indeed cycles can be random, 1,4,3,1,2,4,3,1,2, etc. - to mimic a flag fluttering wildly in a stiff breeze. Using the technique of cycle

animation, it is possible for the animator to reuse such a sequence of drawings over and over again to build up screen time without any additional effort. Some cycles may consist of only two drawings, while others may be involve several tens of complex actions.

ANIMATION AND VISUAL EFFECTS

The Animation & Visual Effects Programme offers students an understanding of the animation film-making process; from the story idea to the final film. Students will gain knowledge of the animation sector and build a strong portfolio which enables them to step out into the world confidently.

The programme has been designed to meet the needs of the three stages of Design Pedagogy which are Design for Business, Design for Humanity and Design as Experimentation. Each sector of this programme feeds in to high-end skills, that helps students to work with the above stages effectively.

2D Animation Programme

The courses offered here begin with learning various concepts such as line of action, perspective, balance, weight & composition followed by different styles of drawings, gesture drawing, anatomy drawings, environment & props and from character design to Storyboarding & Animatics. This programme allows students to gain deeper understanding of the basic and advanced levels of animation

and also takes them through the traditional methods followed, by learning software skills such as Flash, Adobe and After Effects.

3D Animation Programme

Courses begin with sculpture making and taking the student through an orientation programme to benefit students from traditional artistic backgrounds as also for those who are new to the 3D medium.

Courses such as 3D Modelling, Texturing, and Rigging (Character Setup), Animation, Lighting & Rendering, help students to explore a wide range of possibilities to apply their knowledge in animation film making or for design purposes.

Compositing & Visual Effects Programme

This programme helps the students to develop an eye for detail when it comes to 3D Film-making, Television Commercials or producing Vfx for films. Courses covered include Motion Graphics, Basic & Advanced levels of compositing, Post production techniques and Match Moving, which help students to produce stunning visual effects according to their requirements.

Internship

Internship practice provides students with real life challenges to meet the needs of the industry and professional standards; this experience allows them to test and practice

their acquired creative & technical talents to the extreme. Internship offers students an opportunity to work with some leading professional Animation & VFX studios and Professional artists.

5

Initial Development of Computer Graphics Design

The advance in computer graphics was to come from Ivan Sutherland. In 1961 Sutherland created another computer drawing programme 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 programme. 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 programme 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 programme, 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 programme 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

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

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 Vector graphics consists in encoding information about shapes and colors that comprise the image, which can allow for more flexibility in rendering. 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

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 behaviour 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 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.

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 programmes. 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 programme to be processed and output to a digital image or raster graphics image file. The rendering programme is usually built into the computer

graphics software, though others are available as plug-ins or entirely separate programmes. 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

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 programme, 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 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.

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

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.

GRAPHIC DESIGN

Graphic design is a creative process – most often involving a client and a designer and usually completed in conjunction with producers of form (i.e., printers, programmers, signmakers, etc.) – undertaken in order to convey a specific message (or messages) to a targeted audience. The term "graphic design" can also refer to a number of artistic and professional disciplines that focus on visual communication and presentation. The field as a whole is also often referred to as *Visual Communication* or *Communication Design*.

Various methods are used to create and combine words, symbols, and images to create a visual representation of ideas and messages. A graphic designer may use typography, visual arts and page layout techniques to produce the final result. Graphic design often refers to both the process (designing) by which the communication is created and the products (designs) which are generated.

Common uses of graphic design include identity (logos and branding), web sites, publications (magazines, newspapers, and books), advertisements and product packaging. For example, a product package might include a logo or other artwork, organized text and pure design elements such as shapes and color which unify the piece. Composition is one of the most important features of graphic design, especially when using pre-existing materials or diverse elements.

HISTORY

While Graphic Design as a discipline has a relatively recent history, with the name 'graphic design" first coined by William Addison Dwiggins in 1922, graphic design-like activities span the history of humankind: from the caves of Lascaux, to Rome's Trajan's Column to the illuminated manuscripts of the Middle Ages, to the dazzling neons of Ginza. In both this lengthy history and in the relatively recent explosion of visual communication in the 20th and 21st centuries, there is sometimes a blurring distinction and over-lapping of advertising art, graphic design and fine art. After all, they share many of the same elements, theories, principles, practices and languages, and sometimes the same benefactor or client. In advertising art the ultimate objective is the sale of goods and services. In graphic design, "the essence is to give order to information, form to ideas, expression and feeling to artifacts that document human experience."

The Advent of Printing

During the Tang Dynasty (618–907) between the 4th and 7th century AD, wood blocks were cut to print on textiles and later to reproduce Buddhist texts. A Buddhist scripture printed in 868 is the earliest known printed book. Beginning in the 11th century, longer scrolls and books were produced using movable type printing making books widely available during the Song dynasty (960–1279). Sometime around 1450, Johann Gutenberg's printing press made books widely available in Europe. The book design of Aldus Manutius developed the book structure which would become the foundation of western publication design. This era of graphic design is called Humanist or Old Style.

Emergence of the Design Industry

In late 19th century Europe, especially in the United Kingdom, the movement began to separate graphic design from fine art. In 1849, Henry Cole became one of the major forces in design education in Great Britain, informing the government of the importance of design in his *Journal of Design and Manufactures*. He organized the Great Exhibition as a celebration of modern industrial technology and Victorian design. From 1891 to 1896, William Morris' Kelmscott Press published books that are some of the most significant of the graphic design products of the Arts and Crafts movement, and made a very lucrative business of creating books of great stylistic refinement and selling them

to the wealthy for a premium. Morris proved that a market existed for works of graphic design in their own right and helped pioneer the separation of design from production and from fine art. The work of the Kelmscott Press is characterized by its obsession with historical styles. This historicism was, however, important as it amounted to the first significant reaction to the stale state of nineteenth-century graphic design. Morris' work, along with the rest of the Private Press movement, directly influenced Art Nouveau and is indirectly responsible for developments in early twentieth century graphic design in general.

Twentieth Century Design

The name "Graphic Design" first appeared in print in the 1922 essay "New Kind of Printing Calls for New Design" by William Addison Dwiggins, an American book designer in the early 20th century. Raffe's *Graphic Design*, published in 1927, is considered to be the first book to use "Graphic Design" in its title. The signage in the London Underground is a classic design example of the modern era and used a font designed by Edward Johnston in 1916. In the 1920s, Soviet constructivism applied 'intellectual production' in different spheres of production. The movement saw individualistic art as useless in revolutionary Russia and thus moved towards creating objects for utilitarian purposes. They designed buildings, theater sets, posters, fabrics, clothing, furniture, logos, menus, etc.

Jan Tschichold codified the principles of modern typography in his 1928 book, *New Typography*. He later repudiated the philosophy he espoused in this book as being fascistic, but it remained very influential. Tschichold, Bauhaus typographers such as Herbert Bayer and Laszlo Moholy-Nagy, and El Lissitzky have greatly influenced graphic design as we know it today. They pioneered production techniques and stylistic devices used throughout the twentieth century. The following years saw graphic design in the modern style gain widespread acceptance and application. A booming post-World War II American economy established a greater need for graphic design, mainly advertising and packaging. The emigration of the German Bauhaus school of design to Chicago in 1937 brought a "mass-produced" minimalism to America; sparking a wild fire of "modern" architecture and design. Notable names in mid-century modern design include Adrian Frutiger, designer of the typefaces Univers and Frutiger; Paul Rand, who, from the late 1930s until his death in 1996, took the principles of the Bauhaus and applied them to popular advertising and logo design, helping to create a uniquely American approach to European minimalism while becoming one of the principal pioneers of the subset of graphic design known as corporate identity; and Josef Müller-Brockmann, who designed posters in a severe yet accessible manner typical of the 1950s and 1970s era.

The growth of the graphic design industry has grown in parallel with the rise of consumerism. This has raised some concerns and criticisms, notably from within the graphic design community with the First Things First manifesto. First launched by Ken Garland in 1964, it was re-published as the First Things First 2000 manifesto in 1999 in the magazine *Emigre* 51 stating "We propose a reversal of priorities in favour of more useful, lasting and democratic forms of communication - a mindshift away from product marketing and toward the exploration and production of a new kind of meaning. The scope of debate is shrinking; it must expand. Consumerism is running uncontested; it must be challenged by other perspectives expressed, in part, through the visual languages and resources of design." Both editions attracted signatures from respected design practitioners and thinkers, for example; Rudy VanderLans, Erik Spiekermann, Ellen Lupton and Rick Poynor. The 2000 manifesto was also notably published in Adbusters, known for its strong critiques of visual culture.

APPLICATIONS

From road signs to technical schematics, from interoffice memorandums to reference manuals, graphic design enhances transfer of knowledge. Readability is enhanced by improving the visual presentation of text. Design can also aid in selling a product or idea through effective visual communication. It is applied to products and elements of company identity like logos, colors, packaging, and text.

Together these are defined as branding (see also advertising). Branding has increasingly become important in the range of services offered by many graphic designers, alongside corporate identity. Whilst the terms are often used interchangeably, branding is more strictly related to the identifying mark or trade name for a product or service, whereas corporate identity can have a broader meaning relating to the structure and ethos of a company, as well as to the company's external image. Graphic designers will often form part of a team working on corporate identity and branding projects. Other members of that team can include marketing professionals, communications consultants and commercial writers.

Textbooks are designed to present subjects such as geography, science, and math. These publications have layouts which illustrate theories and diagrams. A common example of graphics in use to educate is diagrams of human anatomy. Graphic design is also applied to layout and formatting of educational material to make the information more accessible and more readily understandable. Graphic design is applied in the entertainment industry in decoration, scenery, and visual story telling. Other examples of design for entertainment purposes include novels, comic books, DVD covers, opening credits and closing credits in film, and programmes and props on stage. This could also include artwork used for t-shirts and other items screenprinted for

sale. From scientific journals to news reporting, the presentation of opinion and facts is often improved with graphics and thoughtful compositions of visual information - known as information design. Newspapers, magazines, blogs, television and film documentaries may use graphic design to inform and entertain. With the advent of the web, information designers with experience in interactive tools such as Adobe Flash are increasingly being used to illustrate the background to news stories.

SKILLS

A graphic design project may involve the stylization and presentation of existing text and either preexisting imagery or images developed by the graphic designer. For example, a newspaper story begins with the journalists and photojournalists and then becomes the graphic designer's job to organize the page into a reasonable layout and determine if any other graphic elements should be required. In a magazine article or advertisement, often the graphic designer or art director will commission photographers or illustrators to create original pieces just to be incorporated into the design layout.

Or the designer may utilize stock imagery or photography. Contemporary design practice has been extended to the modern computer, for example in the use of WYSIWYG user interfaces, often referred to as interactive design, or multimedia design.

Visual Arts

Before any graphic elements may be applied to a design, the graphic elements must be originated by means of visual art skills. These graphics are often (but not always) developed by a graphic designer. Visual arts include works which are primarily visual in nature using anything from traditional media, to photography or computer generated art. Graphic design principles may be applied to each graphic art element individually as well as to the final composition.

Typography

Typography is the art, craft and techniques of type design, modifying type glyphs, and arranging type. Type glyphs (characters) are created and modified using a variety of illustration techniques. The arrangement of type is the selection of typefaces, point size, line length, leading (line spacing) and letter spacing. Typography is performed by typesetters, compositors, typographers, graphic artists, art directors, and clerical workers. Until the Digital Age, typography was a specialized occupation. Digitization opened up typography to new generations of visual designers and lay users.

Page Layout

The page layout aspect of graphic design deals with the arrangement of elements (content) on a page, such as image placement, and text layout and style. Beginning from early

illuminated pages in hand-copied books of the Middle Ages and proceeding down to intricate modern magazine and catalogue layouts, structured page design has long been a consideration in printed material. With print media, elements usually consist of type (text), images (pictures), and occasionally place-holder graphics for elements that are not printed with ink such as die/laser cutting, foil stamping or blind embossing.

Interface Design

Since the advent of the World Wide Web and computer software development, many graphic designers have become involved in interface design. This has included web design and software design, when end user interactivity is a design consideration of the layout or interface. Combining visual communication skills with the interactive communication skills of user interaction and online branding, graphic designers often work with software developers and web developers to create both the look and feel of a web site or software application and enhance the interactive experience of the user or web site visitor. An important aspect of interface design is icon design.

Printmaking

Printmaking is the process of making artworks by printing on paper and other materials or surfaces. Except in the case of monotyping, the process is capable of producing multiples

of the same piece, which is called a print. Each piece is not a copy but an original since it is not a reproduction of another work of art and is technically known as an impression. Painting or drawing, on the other hand, create a unique original piece of artwork. Prints are created from a single original surface, known technically as a matrix. Common types of matrices include: plates of metal, usually copper or zinc for engraving or etching; stone, used for lithography; blocks of wood for woodcuts, linoleum for linocuts and fabric plates for screen-printing. But there are many other kinds, discussed below. Works printed from a single plate create an edition, in modern times usually each signed and numbered to form a limited edition. Prints may also be published in book form, as artist's books. A single print could be the product of one or multiple techniques.

TOOLS

The mind may be the most important graphic design tool. Aside from technology, graphic design requires judgment and creativity. Critical, observational, quantitative and analytic thinking are required for design layouts and rendering. If the executor is merely following a solution (e.g. sketch, script or instructions) provided by another designer (such as an art director), then the executor is not usually considered the designer. The method of presentation (e.g. arrangement, style, medium) may be equally important to the design. The layout is produced using external traditional or digital image editing

tools. The appropriate development and presentation tools can substantially change how an audience perceives a project. In the mid 1980s, the arrival of desktop publishing and graphic art software applications introduced a generation of designers to computer image manipulation and creation that had previously been manually executed. Computer graphic design enabled designers to instantly see the effects of layout or typographic changes, and to simulate the effects of traditional media without requiring a lot of space. However, traditional tools such as pencils or markers are useful even when computers are used for finalization; a designer or art director may hand sketch numerous concepts as part of the creative process.

Some of these sketches may even be shown to a client for early stage approval, before the designer develops the idea further using a computer and graphic design software tools. Computers are considered an indispensable tool in the graphic design industry. Computers and software applications are generally seen by creative professionals as more effective production tools than traditional methods. However, some designers continue to use manual and traditional tools for production, such as Milton Glaser. New ideas can come by way of experimenting with tools and methods. Some designers explore ideas using pencil and paper. Others use many different mark-making tools and resources from computers to sculpture as a means of

inspiring creativity. One of the key features of graphic design is that it makes a tool out of appropriate image selection in order to possibly convey meaning.

Computers and the Creative Process

There is some debate whether computers enhance the creative process of graphic design. Rapid production from the computer allows many designers to explore multiple ideas quickly with more detail than what could be achieved by traditional hand-rendering or paste-up on paper, moving the designer through the creative process more quickly. However, being faced with limitless choices does not help isolate the best design solution and can lead to endless iterations with no clear design outcome. A graphic designer may use sketches to explore multiple or complex ideas quickly without the distractions and complications of software. Hand-rendered comps are often used to get approval for an idea execution before a design invests time to produce finished visuals on a computer or in paste-up. The same thumbnail sketches or rough drafts on paper may be used to rapidly refine and produce the idea on the computer in a hybrid process. This hybrid process is especially useful in logo design where a software learning curve may detract from a creative thought process. The traditional-design/computer-production hybrid process may be used for freeing one's creativity in page layout or image development as well. In the early days of computer publishing, many 'traditional' graphic designers relied on

computer-savvy production artists to produce their ideas from sketches, without needing to learn the computer skills themselves. However, this practice has been increasingly less common since the advent of desktop publishing over 30 years ago. The use of computers and graphics software is now taught in most graphic design courses.

OCCUPATIONS

Graphic design career paths cover all ends of the creative spectrum and often overlap. The main job responsibility of a Graphic Designer is the arrangement of visual elements in some type of media. The main job titles include graphic designer, art director, creative director, and the entry level production artist. Depending on the industry served, the responsibilities may have different titles such as "DTP Associate" or "Graphic Artist", but despite changes in title, graphic design principles remain consistent. The responsibilities may come from or lead to specialized skills such as illustration, photography or interactive design. Today's graduating graphic design students are normally exposed to all of these areas of graphic design and urged to become familiar with all of them as well in order to be competitive. Graphic designers can work in a variety of environments. Whilst many will work within companies devoted specifically to the industry, such as design consultancies or branding agencies, others may work within publishing, marketing or other communications companies.

Increasingly, especially since the introduction of personal computers to the industry, many graphic designers have found themselves working within non-design oriented organizations, as in-house designers. Graphic designers may also work as free-lance designers, working on their own terms, prices, ideas, etc.

A graphic designer reports to the art director, creative director or senior media creative. As a designer becomes more senior, they may spend less time designing media and more time leading and directing other designers on broader creative activities, such as brand development and corporate identity development. As graphic designers become more senior, they are often expected to interact more directly with clients.

6

Advanced in Information Graphics

Information graphics or infographics are graphic visual representations of information, data or knowledge. These graphics present complex information quickly and clearly, such as in signs, maps, journalism, technical writing, and education. With an information graphic, computer scientists, mathematicians, and statisticians develop and communicate concepts using a single symbol to process information.

OVERVIEW

Today information graphics surround us in the media, in published works both pedestrian and scientific, in road signs and manuals. They illustrate information that would be unwieldy in text form, and act as a visual shorthand for everyday concepts such as stop and go. In newspapers, infographics are commonly used to show the weather, as

well as maps and site plans for newsworthy events, and graphs for statistical data. Some books are almost entirely made up of information graphics, such as David Macaulay's The Way Things Work. Although they are used heavily in children's books, they are also common in scientific literature, where they illustrate physical systems, especially ones that cannot be photographed (such as cutaway diagrams, astronomical diagrams, and images of microscopic or sub-microscopic systems).

Modern maps, especially route maps for transit systems, use infographic techniques to integrate a variety of information, such as the conceptual layout of the transit network, transfer points, and local landmarks. Traffic signs and other public signs rely heavily on information graphics, such as stylized human figures (the ubiquitous stick figure), icons and emblems to represent concepts such as yield, caution, and the direction of traffic. Public places such as transit terminals usually have some sort of integrated "signage system" with standardized icons and stylized maps. Technical manuals make extensive use of diagrams and also common icons to highlight warnings, dangers, and standards certifications.

HISTORY

Early Experiments

In prehistory, early humans created the first information graphics: cave paintings and later maps. Map-making began

several millennia before writing, and the map at Çatalhöyük dates from around 7500 BCE. Later icons were used to keep records of cattle and stock. The Indians of Mesoamerica used imagery to depict the journeys of past generations. Illegible on their own, they served as a supportive element to memory and storytelling.

In 1626 Christopher Scheiner published the *Rosa Ursina sive Sol* which used a variety of graphics to reveal his astronomical research on the sun. He used a series of images to explain the rotation of the sun over time (by tracking sunspots). In 1786, William Playfair published the first data graphs in his book *The Commercial and Political Atlas*. The book is filled with statistical graphs, bar charts, line graphs and histograms, that represent the economy of 18th century England. In 1801 Playfair introduced the first area chart and pie chart in *Statistical Breviary*.

In 1857, English nurse Florence Nightingale used information graphics persuading Queen Victoria to improve conditions in military hospitals, principally the Coxcomb chart, a combination of stacked bar and pie charts, depicting the number and causes of deaths during each month of the Crimean War. 1861 saw the release of a seminal information graphic on the subject of Napoleon's disastrous march on Moscow. The creator, Charles Joseph Minard, captured four different changing variables that contributed to the failure, in a single two-dimensional image: the army's direction as

they traveled, the location the troops passed through, the size of the army as troops died from hunger and wounds, and the freezing temperatures they experienced. James Joseph Sylvester introduced the term "graph" in 1878 and published a set of diagrams showing the relationship between chemical bonds and mathematical properties. These were also the first mathematic graphs.

The Development of a Visual Language in the 20th Century

In 1936 Otto Neurath introduced a system of pictographs intended to function as an international visual or picture language. Isotype included a set of stylized human figures which were the basis for the ubiquitous modern stick figures. In 1942 Isidore Isou published the Lettrist manifesto. In 1958 Stephen Toulmin proposed a graphical argument model that became influential in argumentation theory and its applications. The 1972 Munich Olympics were the venue for Otl Aicher to introduce a new set of pictograms that proved to be extremely popular, and influenced the ubiquitous modern stick figures used in public signs. Also in 1972 the Pioneer Plaque was launched into space with the Pioneer 10 probe. Inscribed into the plaque was an information graphic intended as a kind of interstellar message in a bottle, designed by Carl Sagan and Frank Drake. The message is unique in that it is intended to be understood by extraterrestrial beings who would share no common language with humans. It

depicts a picture of a man and a woman standing in front of a simplified silhouette of the probe in order to give a sense of scale.

It also contains a map locating the sun relative to a number of pulsars, and a simplified depiction of the solar system, with the probe's path from earth into outer space shown with an arrow. 2005–Present day. The information graphic trend starts to become popular amongst the larger social media aggregation sites Digg Reddit. The data contained in modern info graphics tends to be research centric and attributed to multiple sources. Information graphics of note are the Infographic Resume of Michael and the more modern UK Government Spending live infographic With the popularity of the information graphics continuing to grow, see google search trends, many internet marketing companies use this to generate viral content that web users will share freely.

INFORMATION GRAPHICS SUBJECTS

Visual Devices

Information graphics are visual devices intended to communicate complex information quickly and clearly. The devices include, according to Doug Newsom (2004), charts, diagrams, graphs, tables, maps and lists. Among the most common devices are horizontal bar charts, vertical column charts, and round or oval pie charts, that can summarize a lot of statistical information. Diagrams can be used to show

how a system works, and may be an organizational chart that shows lines of authority, or a systems flowchart that shows sequential movement. Illustrated graphics use images to related data. The snapshots features used every day by *USA Today* are good examples of this technique. Tables are commonly used and may contain lots of numbers. Modern interactive maps and bulleted numbers are also infographic devices.

Elements of Information Graphics

The basic material of an information graphic is the data, information, or knowledge that the graphic presents. In the case of data, the creator may make use of automated tools such as graphing software to represent the data in the form of lines, boxes, arrows, and various symbols and pictograms.

The information graphic might also feature a key which defines the visual elements in plain English. A scale and labels are also common. The elements of an info graphic do not have to be an exact or realistic representation of the data, but can be a simplified version.

Interpreting Information Graphics

Many information graphics are specialised forms of depiction that represent their content in sophisticated and often abstract ways. In order to interpret the meaning of these graphics appropriately, the viewer requires a suitable

level of graphicacy. In many cases, the required graphicacy involves comprehension skills that are learned rather than innate.

At a fundamental level, the skills of decoding individual graphic signs and symbols must be acquired before sense can be made of an information graphic as a whole. However, knowledge of the conventions for distributing and arranging these individual components is also necessary for the building of understanding.

Interpreting with a Common Visual Language

In contrast to the above, many other forms of infographics take advantage of innate visual language that is largely universal. The disciplined use of the color red, for emphasis, on an otherwise muted design, demands attention in a primal way even children understand. Many maps, interfaces, dials and gauges on instruments and machinery use icons that are easy to grasp and speed understanding for safe operation. The use of a rabbit and a turtle icon to represent fast and slow, respectively, is one such successful use by the John Deere company on the throttle of their tractors.

MODERN PRACTITIONERS

A statistician and sculptor, Edward Tufte has written a series of highly regarded books on the subject of information graphics. Tufte also delivers lectures and workshops on a regular basis. He describes the process of incorporating many

dimensions of information into a two-dimensional image as 'escaping flatland' (alluding to the 2-dimensional world of the Victorian novella *Flatland*). The work done by Peter Sullivan for The Sunday Times in the 1970s, 80s and 90s, was one of the key factors in encouraging newspapers to use more graphics. Sullivan is also one of the few authors who have written about information graphics in newspapers. Likewise the staff artists at USA Today, the colorful United States newspaper that debuted in 1982, firmly established the philosophy of using graphics to make information easier to comprehend. The paper received criticism for oversimplifying news and sometimes creating infographics that emphasized entertainment over respect for content and data, sometimes referred to as chartjunk. While some critics deride the graphic qualities of this work, its role in establishing infographics as a practice cannot be ignored.

Nigel Holmes is an established commercial creator of what he calls "explanation graphics". His works deal not only with the visual display of information but also of knowledge – how to do things. He created graphics for *Time* magazine for 16 years, and is the author of several books on the subject. Close and strongly related to the field of information graphics, is information design. Actually, making infographics is a certain discipline within the information design world. Author and founder of the TED, Richard Saul Wurman, is considered the originator of the phrase, "information architect", and

many of his books, such as *Information Anxiety*, helped propel the phrase, "information design", from a concept to an actual job category. While the art form of infographics has its roots in print, by the year 2000, the use of Adobe Flash-based animations on the web has allowed to make mapping solutions and other products famous and addictive by using many key best practices of infographics. Likewise, their use in television is relatively recent, for in 2002, two Norwegian musicians of Röyksopp issued a music video for their song Remind Me that was completely made from animated infographics. In 2004, a television commercial for the French energy company Areva used similar animated infographics and both of these videos and their high visibility have helped the corporate world recognize the value in using this form of visual language to describe complex information efficiently.

INFORMATION VISUALIZATION

Information visualization is the interdisciplinary study of "the visual representation of large-scale collections of non-numerical information, such as files and lines of code in software systems, library and bibliographic databases, networks of relations on the internet, and so forth".

OVERVIEW

The field of information visualization has emerged "from research in human-computer interaction, computer science, graphics, visual design, psychology, and business methods.

It is increasingly applied as a critical component in scientific research, digital libraries, data mining, financial data analysis, market studies, manufacturing production control, and drug discovery".

Information visualization presumes that "visual representations and interaction techniques take advantage of the human eye's broad bandwidth pathway into the mind to allow users to see, explore, and understand large amounts of information at once. Information visualization focused on the creation of approaches for conveying abstract information in intuitive ways."

HISTORY

The modern study of visualization started with computer graphics, which "has from its beginning been used to study scientific problems. However, in its early days the lack of graphics power often limited its usefulness. The recent emphasis on visualization started in 1987 with the special issue of Computer Graphics on Visualization in Scientific Computing.

Since then there have been several conferences and workshops, co-sponsored by the IEEE Computer Society and ACM SIGGRAPH". They have been devoted to the general topics of data visualisation, information visualization and scientific visualisation, and more specific areas such as volume visualization.

SPECIFIC METHODS AND TECHNIQUES

- Cladogram (phylogeny)
- Dendrogram (classification)
- Information visualization reference model
- Graph drawing
- Heatmap
- HyperbolicTree
- Multidimensional scaling
- Problem Solving Environment
- Treemapping
- Southbeach Notation

APPLICATIONS

Information visualization insights are being applied in areas such as:

- scientific research,
- digital libraries,
- data mining,
- financial data analysis, market studies,
- manufacturing production control,
- and crime mapping.

And also:

- Command Post of the Future
- Informedia Digital Library
- Information graphics
- Starlight Information Visualization System

EXPERTS

Stuart K. Card

Stuart K. Card is an American researcher. He is a Senior Research Fellow at Xerox PARC and one of the pioneers of applying human factors in human–computer interaction. The 1983 book *The Psychology of Human-Computer Interaction*, which he co-wrote with Thomas P. Moran and Allen Newell, became a very influential book in the field, partly for introducing the Goals, Operators, Methods, and Selection rules (GOMS) framework. His currently research is in the field of developing a supporting science of human–information interaction and visual-semantic prototypes to aid sensemaking.

George W. Furnas

George Furnas is a professor and Associate Dean for Academic Strategy at the School of Information of the University of Michigan. Furnas has also worked with Bell Labs where he earned the moniker "Fisheye Furnas" while working with fisheye visualizations. He is a pioneer of Latent semantic analysis, Professor Furnas is also considered a pioneer in the concept of Mosaic of Responsive Adaptive Systems (MoRAS).

James D. Hollan

James D. Hollan directs the Distributed Cognition and Human-Computer Interaction Laboratory at University of

California, San Diego. His research explores the cognitive consequences of computationally-based media. The goal is to understand the cognitive and computational characteristics of dynamic interactive representations as the basis for effective system design. His current work focuses on cognitive ethnography, computer-mediated communication, distributed cognition, human-computer interaction, information visualization, multiscale software, and tools for analysis of video data.

More Related Scientists

- Scott Meyers
- George G. Robertson
- Pierre Rosenstiehl
- Ben Shneiderman
- John Stasko

ORGANIZATION

- International Symposium on Graph Drawing
- Panopticon Software
- University of Maryland Human-Computer Interaction Lab
- Vvi

SCIENTIFIC VSUALIZATION

Scientific visualization (also spelled scientific visualisation) is an interdisciplinary branch of science according to Friendly

(2008) "primarily 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".

HISTORY

One of the earliest examples of three-dimensional scientific visualisation was Maxwell's thermodynamic surface, sculpted in clay in 1874 by James Clerk Maxwell. This prefigured modern scientific visualization techniques which use computer graphics. Notable early two-dimensional examples include the flow map of Napoleon's March on Moscow produced by Charles Joseph Minard in 1869; the "coxcombs" used by Florence Nightingale in 1857 as part of a campaign to improve sanitary conditions in the British army; and the dot map used by John Snow in 1855 to visualise the Broad Street cholera outbreak.

SCIENTIFIC VISUALIZATION TOPICS

Computer Animation

Computer animation is the art, technique and science of creating moving images via the use of computers. 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.

Computer Simulation

Computer simulation is a computer programme, or network of computers, that attempts to simulate an abstract model of a particular system. Computer simulations have become a useful part of mathematical modelling of many natural systems in physics, and computational physics, chemistry and biology; human systems in economics, psychology, and social science; and in the process of engineering and new technology, to gain insight into the operation of those systems, or to observe their behaviour. The simultaneous visualization and simulation of a system is called visulation. Computer simulations vary from computer programmes that run a few minutes, to network-based groups of computers running for hours, to ongoing simulations that run for months. The scale of events being simulated by computer simulations has far exceeded anything possible (or perhaps even imaginable) using the traditional paper-and-pencil mathematical modeling: over 10 years ago, a desert-battle simulation, of one force invading another, involved the modeling of 66,239 tanks, trucks and other vehicles on simulated terrain around Kuwait, using multiple supercomputers in the DoD High Performance Computer Modernization Programme.

Information Visualization

Information visualization is the study of "the visual representation of large-scale collections of non-numerical information, such as files and lines of code in software systems, library and bibliographic databases, networks of relations on the internet, and so forth". Information visualization focused on the creation of approaches for conveying abstract information in intuitive ways. Visual representations and interaction techniques take advantage of the human eye's broad bandwidth pathway into the mind to allow users to see, explore, and understand large amounts of information at once.

Interface Technology and Perception

Interface technology and perception shows how new interfaces and a better understanding of underlying perceptual issues create new opportunities for the scientific visualization community.

Surface Rendering

Rendering is the process of generating an image from a model, by means of computer programmes. The model is a description of three dimensional objects in a strictly defined language or data structure. It would contain geometry, viewpoint, texture, lighting, and shading information. The image is a digital image or raster graphics image. The term may be by analogy with an "artist's rendering" of a scene.

'Rendering' is also used to describe the process of calculating effects in a video editing file to produce final video output. Important rendering techniques are:

Scanline Rendering and Rasterisation

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.

Ray Casting

Ray casting 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.

Radiosity

Radiosity, also known as Global Illumination, 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.

Ray Tracing

Ray tracing is an extension of the same technique developed in scanline rendering and ray casting. Like those, it handles complicated objects well, and the objects may be described mathematically. Unlike scanline and casting, ray tracing is almost always a Monte Carlo technique, that is one based on averaging a number of randomly generated samples from a model.

Volume Rendering

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.

Volume Visualization

According to Rosenblum (1994) "volume visualization examines a set of techniques that allows viewing an object

without mathematical representing the other surface. Initially used in medical imaging, volume visualization has become an essential technique for many sciences, portraying phenomena become an essential technique such as clouds, water flows, and molecular and biological structure. Many volume visualization algorithms are computationally expensive and demand large data storage. Advances in hardware and software are generalizing volume visualization as well as real time performances".

SCIENTIFIC VISUALIZATION APPLICATIONS

This section will give a series of examples how scientific visualization can be applied today.

Star formation: The featured plot is a Volume plot of the logarithm of gas/dust density in an Enzo star and galaxy simulation. Regions of high density are white while less dense regions are more blue and also more transparent.

Gravity waves: Researchers used the Globus Toolkit to harness the power of multiple supercomputers to simulate the gravitational effects of black-hole collisions.

Massive Star Supernovae Explosions: In the image three Dimensional Radiation Hydrodynamics Calculations of Massive Star Supernovae Explosions The DJEHUTY stellar evolution code was used to calculate the explosion of SN 1987A model in three dimensions.

Molecular rendering: VisIt's general plotting capabilities were used to create the molecular rendering shown in the

featured visualization. The original data was taken from the Protein Data Bank and turned into a VTK file before rendering.

In Geography and Ecology

Terrain rendering: VisIt can read several file formats common in the field of Geographic Information Systems (GIS), allowing one to plot raster data such as terrain data in visualizations. The featured image shows a plot of a DEM dataset containing mountainous areas near Dunsmuir, CA. Elevation lines are added to the plot to help delineate changes in elevation.

Tornado Simulation: This image was created from data generated by a tornado simulation calculated on NCSA's IBM p690 computing cluster. High-definition television animations of the storm produced at NCSA were included in an episode of the PBS television series NOVA called "Hunt for the Supertwister." The tornado is shown by spheres that are colored according to pressure; orange and blue tubes represent the rising and falling airflow around the tornado.

Climate visualization: This visualization depicts the carbon dioxide from various sources that are advected individually as tracers in the atmosphere model. Carbon dioxide from the ocean is shown as plumes during February 1900.

Atmospheric Anomaly in Times Square In the image the results from the SAMRAI simulation framework of an atmospheric anomaly in and around Times Square are visualized.

In the Formal Sciences

Computer mapping of topographical surfaces: Through computer mapping of topographical surfaces, mathematicians can test theories of how materials will change when stressed. The imaging is part of the work on the NSF-funded Electronic Visualization Laboratory at the University of Illinois at Chicago

Curve plots: VisIt can plot curves from data read from files and it can be used to extract and plot curve data from higher dimensional datasets using lineout operators or queries. The curves in the featured image correspond to elevation data along lines drawn on DEM data and were created with the feature lineout capability. Lineout allows you to interactively draw a line, which specifies a path for data extraction. The resulting data was then plotted as curves.

Image annotations: The featured plot shows Leaf Area Index (LAI), a measure of global vegetative matter, from a NetCDF dataset. The primary plot is the large plot at the bottom, which shows the LAI for the whole world. The plots on top are actually annotations that contain images generated earlier. Image annotations can be used to include material that enhances a visualization such as auxiliary plots, images of experimental data, project logos, etc.

Scatter plot: VisIt's Scatter plot allows to visualize multivariate data of up to four dimensions. The Scatter plot takes multiple scalar variables and uses them for different

axes in phase space. The different variables are combined to form coordinates in the phase space and they are displayed using glyphs and colored using another scalar variable.

In the Applied Sciences

Porsche 911 model (NASTRAN model): The featured plot contains a Mesh plot of a Porsche 911 model imported from a NASTRAN bulk data file. VisIt can read a limited subset of NASTRAN bulk data files, generally enough to import model geometry for visualization.

YF-17 aircraft Plot: The featured image displays plots of a CGNS dataset representing a YF-17 jet aircraft. The dataset consists of an unstructured grid with solution. The image was created by using a pseudocolor plot of the dataset's Mach variable, a Mesh plot of the grid, and Vector plot of a slice through the Velocity field.

City rendering: An ESRI shapefile containing a polygonal description of the building footprints was read in and then the polygons were resampled onto a rectilinear grid, which was extruded into the featured cityscape.

Inbound traffic measured: This image is a visualization study of inbound traffic measured in billions of bytes on the NSFNET T1 backbone for the month of September 1991. The traffic volume range is depicted from purple (zero bytes) to white (100 billion bytes). It represents data collected by Merit Network, Inc.

SCIENTIFIC VISUALIZATION EXPERTS

Bruce H. McCormick

Bruce H. McCormick (1930 - 2007) was an American computer scientist, who studied Physics from MIT, Cambridge University and Harvard University in the 1950s. In the 1960s he initiated and directed the ILLIAC III Image Processing Computer project and developed the first imaging of blood flow and macular degeneracy in the human retina. In the 1980s he organized and chaired the first Brain Mapping Machine Design Workshop in 1985. Two years later in 1987, he developed and promoted the concept of "scientific visualization" at the National Science Foundation Advisory Panel on Graphics, Image Processing, and Workstations.

Thomas A. DeFanti

Thomas A. DeFanti (born 1948) is an American computer graphics researcher and Director of the Electronic Visualization Laboratory (EVL), who studied mathematics and computer information science, with a PhD in Computer Graphics Research received in 1973. He joined the faculty of the University of Illinois at Chicago, and next he amassed a number of credits. He cofounded the Electronic Visualization Laboratory (EVL), used the EVL hardware and software for the computer animation produced for the Star Wars movie. DeFanti contributed greatly to the growth of the SIGGRAPH organization and conference.

Maxine D. Brown

Maxine D. Brown is an American computer scientist, and associate director of the Electronic Visualization Laboratory (EVL). She also studied mathematics and later computer science in the 1970s.

She has a long history of service to the computer graphics and supercomputing communities, and has contributed to many facets of SIGGRAPH.

Clifford A. Pickover

Clifford A. Pickover is an American author, editor, and columnist in the fields of science, mathematics, and science fiction, primary interested in finding new ways to expand creativity by melding art, science, mathematics, and other seemingly disparate areas of human endeavor. In the 1990s he has edited several books, like "Frontiers of Scientific Visualization" (1994) and "Visualizing Biological Information" (1995).

Lawrence Jay Rosenblum

Lawrence J. Rosenblum (born 1949) is an American mathematician, and Programme Director for Graphics and Visualization at the National Science Foundation. Rosenblum's research interests include mobile augmented reality (AR), scientific and uncertainty visualization, VR displays, and applications of VR/AR systems. His research group has produced advances in mobile augmented reality

(AR), scientific and uncertainty visualization, VR displays, applications of VR/AR systems, and understanding human performance in graphics systems.

7

Computer Graphics Science

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 the study of three-dimensional computer graphics, it also encompasses two-dimensional graphics and image processing.

OVERVIEW

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. Connected studies include:

- Scientific visualization
- Information visualization
- Computer vision
- Image processing
- Computational Geometry
- Computational Topology
- Applied mathematics

Applications of computer graphics include:

- Special effects
- Visual effects
- Video games
- Digital art

HISTORY

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. There are several international conferences and journals where the most significant results in computer graphics are published. Among them are the SIGGRAPH and Eurographics conferences and the Association for Computing Machinery (ACM) Transactions on Graphics journal. The joint Eurographics and ACM SIGGRAPH symposium series features the major venues for the more specialized sub-fields: Symposium on Geometry Processing,Symposium on Rendering, and Symposium on Computer Animation. As in the rest of computer science,

conference publications in computer graphics are generally more significant than journal publications (and subsequently have lower acceptance rates).

SUBFIELDS IN COMPUTER GRAPHICS

A broad classification of major subfields in computer graphics might be:

1. Geometry: studies ways to represent and process surfaces
2. Animation: studies with ways to represent and manipulate motion
3. Rendering: studies algorithms to reproduce light transport
4. Imaging: studies image acquisition or image editing

Geometry

The subfield of geometry studies the representation of three-dimensional objects in a discrete digital setting. Because the appearance of an object depends largely on its exterior, boundary representations are most commonly used. Two dimensional surfaces are a good representation for most objects, though they may be non-manifold. Since surfaces are not finite, discrete digital approximations are used. Polygonal meshes (and to a lesser extent subdivision surfaces) are by far the most common representation, although point-based representations have become more popular recently (see for instance the Symposium on Point-Based Graphics).

These representations are *Lagrangian*, meaning the spatial locations of the samples are independent. Recently, *Eulerian* surface descriptions (i.e., where spatial samples are fixed) such as level sets have been developed into a useful representation for deforming surfaces which undergo many topological changes (with fluids being the most notable example).

Geometry Subfields

- Implicit surface modeling - an older subfield which examines the use of algebraic surfaces, constructive solid geometry, etc., for surface representation.
- Digital geometry processing - surface reconstruction, simplification, fairing, mesh repair, parameterization, remeshing, mesh generation, surface compression, and surface editing all fall under this heading.
- Discrete differential geometry - a nascent field which defines geometric quantities for the discrete surfaces used in computer graphics.
- Point-based graphics - a recent field which focuses on points as the fundamental representation of surfaces.
- Subdivision surfaces
- Out-of-core mesh processing - another recent field which focuses on mesh datasets that do not fit in main memory.

Animation

The subfield of animation studies descriptions for surfaces (and other phenomena) that move or deform over time. Historically, most work in this field has focused on parametric and data-driven models, but recently physical simulation has become more popular as computers have become more powerful computationally.

Subfields

- Performance capture
- Character animation
- Physical simulation (e.g. cloth modeling, animation of fluid dynamics, etc.)

Rendering

Rendering generates images from a model. Rendering may simulate light transport to create realistic images or it may create images that have a particular artistic style in non-photorealistic rendering. The two basic operations in realistic rendering are transport (how much light passes from one place to another) and scattering (how surfaces interact with light). See Rendering (computer graphics) for more information.

Transport

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

Scattering

Models of *scattering* and *shading* are used to describe the appearance of a surface. In graphics these problems are often studied within the context of rendering since they can substantially affect the design of rendering algorithms. Shading can be broken down into two orthogonal issues, which are often studied independently:

1. scattering - how light interacts with the surface *at a given point*
2. shading - how material properties vary across the surface

The former problem refers to scattering, i.e., 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. The latter issue 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 programme called a shader. (Note that there is some confusion since the word “shader” is sometimes used for programmes that describe local *geometric* variation.)

Other Subfields

- physically-based rendering - concerned with generating images according to the laws of geometric optics

- real time rendering - focuses on rendering for interactive applications, typically using specialized hardware like GPUs
- non-photorealistic rendering
- relighting - recent area concerned with quickly re-rendering scenes

NOTABLE RESEARCHERS IN COMPUTER GRAPHICS

- Jim Blinn
- Jack E. Bresenham
- Loren Carpenter
- Edwin Catmull
- Robert L. Cook
- Paul Debevec
- Ron Fedkiw
- James D. Foley
- David Forsyth
- Henry Fuchs
- Pat Hanrahan
- Takeo Kanade
- Jim Kajiya
- Kenneth Knowlton
- Marc Levoy
- James O'Brien
- Ken Perlin
- Przemyslaw Prusinkiewicz

- William Reeves
- James Sethian
- Ivan Sutherland
- Greg Turk
- Andries van Dam
- Henrik Wann Jensen
- Lance Williams

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. 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 programmes 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. (See also Gouraud shading). 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 1980s, 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 programmes* 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.

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

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

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.

APPENDIX: 3D GRAPHIC SOFTWARES

ART OF ILLUSION

Art of Illusion is a software package used for 3D modeling, texturing, ray tracing, and otherwise rendering computer generated imagery stills or animations (movies).

The goal of Art of Illusion is to provide powerful 3D modeling tools with a user interface that improves on those found in other 3D software packages. Though its interface is simple, Art of Illusion contains many features found in high-end commercial graphics software. Some of its features, like the use of online repositories and a built-in downloading tool for installing extensions, are not found in similar proprietary software. The primary creator and maintainer of the software is Peter Eastman. Art of Illusion is written in the Java programming language. Distributed under the GNU General Public License, it is free software.

FEATURES

- Interface
    - o Object list, scene layout windows, and animation score

- Tooltips and explanatory icons
- Bundled documentation and help interface
- Built-in "live help" chat client for connecting to the freenode Art of Illusion support channel
- Extensions available as scripts or plugins with automated installation and update from online repository (requires internet connection)
- Grid view and realtime display modes including Wireframe, Smooth, Textured.

• Modeling
  - Primitives: cube, sphere, cone, tube, curve, triangle mesh, spline mesh
  - Boolean modeling operations
  - Lathe, sweep, extrude (straight or along a curve)
  - Subdivision surfaces (smoothing) with edge creasing available for triangle meshes.
  - Mesh editor featuring adjustable mesh tension, bevel, taper, various selection methods (including select edge loop/strip), etc.
  - Object array tool (multidimensional arrays or arrays along a curve)
  - Isosurface modeling (via procedural nodes or direct numeric input)

• Animation
  - Distortion tracks for effects like bend, twist, and shatter
  - Skeletal animation with weighting, constraints and inverse kinematics
  - Poses and Gestures
  - Keyframe editor with interpolating curves

  - Path animation
  - Animation through scripted objects, e.g. particles using Particle Jet script
- Textures
  - Types: Uniform, image mapped, procedural 2D and 3D
  - Mapping options: Projection, spherical, cylindrical, UV
  - Per-vertex, per-face and per-face-per-vertex texture assignment (depending on textured object)
  - Layered textures
  - Graphical language for procedural texture/ material design
  - Procedural textures can be based on parameters like view angle (for fresnel-like effects), etc.
  - Animateable textures via texture parameters and use of Time module
  - Environment background can be mapped with any photo (HDR or RGB), as well as procedural textures
- Materials (Represent the internal properties of an object)
  - Procedural and uniform materials
  - Adjustable index of refraction, scattering, eccentricity
  - Animatable materials via Time module
- Rendering
  - Multithreading
  - Global illumination rendering, along with caustics and subsurface scattering

  - Global illumination methods: Monte carlo, photon mapping with final gathering, direct photon mapping, and ambient occlusion
  - Light types: Point lights, spotlights (both with editable radii), and directional lights
  - Soft shadows
  - Light scattering for materials: Both Single Scattering and Photon Scattering (BSSRDF). User can specify either or both at rendertime.
  - Raytraced depth of field
  - Gloss/translucency (blurred reflections and blurry transparency)
  - Motion blur
  - HDRI scene illumination
  - Save renders as HDRI images
- Post Processing
  - Camera filters (Exposure Correction, Glow, Outline, Tint, Blur, etc.)
  - Noise reduction for Global Illumination
- Scripting
  - Scripting language (Beanshell) allowing development of new tools/commands
  - Dynamic Scripts ("smart objects" - see flyswatter tutorial)
- File Handling
  - 3D Import: .obj natively; .dem, .dxf, .geo, .lwo, .pov, .inc, .3ds via plugins
  - 2D Import via scripts: .ai, .svg
  - 3D Export: .pov, .obj, VRML, all native; .stl via plugin

  - 2D Export/Save: .jpg, .bmp, .png, .tif, .hdr, all native; .svg via plugin
- Extended Functionality Through Scripts and Plugins
  - Direct-to-disk rendering using presets like "Letter," "A4," with DPI and Bleed settings (Advanced Rendering plugin)
  - Wireframe/polygon renders with antialiasing and SVG export (Vector Renderer plugin)
  - Preview renders in sidebar (Preview Plugin)
  - Subdivision (Ngon) modeler with tools for both open- and closed-mesh modeling and support for Catmull-Clark subdivision surfaces (Polymesh Editor plugin)
  - ABF+ UV unwrapping available through Polymesh Editor plugin
  - Procedural tree and plant creator (Tree and Plant Designer plugin)
  - Other major features provided through script or plugin extensions such as: 3D text, mesh thickening, platonic solids, sculpt, fractal/image-based heightfields, hair, grunge, and more

DEVELOPMENT

- Art of Illusion is a free and open source application and contributions to the source code, scripts and plugins base, and other aspects of the software are welcome.
- Art of Illusion's development cycle usually includes two builds called "Early Access" (EA) releases followed

by a Beta, or "bug fix" version. However, with a recent announcement, the bug-fixing portion of development has been shifted to make sure that more bugs are fixed in existing features.

- Developers are welcome to submit scripts and plugins via email to maintainers of the Scripts and Plugins Repository for inclusion in the online repository. This ensures that users will be able to download scripts and plugins easily from within Art of Illusion itself.
- Developers with code to contribute to AoI itself should submit it to Peter Eastman for approval and inclusion. More information
- Non-code contributions can be made as well. Details (Note that the Smartgroups link on that page is obsoleted by the Scripts and Plugins Repository)

IRTC COMPETITION

The Internet Ray Tracing Competition is an online computer graphics competition. Several Art of Illusion users have submitted entries to the competition.

- Nate Ryan's Fungus Dance image won 1st place in the September - October 2003 "Decay" contest. link
- Julian MacDonald's "Gladiators" animation won Honorable Mention: Artistic Merit in the July-October 2004 contest. link
- "Bathtime Fun", also by Julian MacDonald, won 2nd place in the September-October 2004 contest. link

- Kevin Lynn's "Is Juggling a Sport?" animation won 1st place in the July-October 2005 contest. link
  - Kevin is continuing an Art of Illusion tradition by offering a box of chocolates to the next winner of an online graphics competition using Art of Illusion.

BLENDER (SOFTWARE)

Blender is a free, open source 3D graphics application that can be used for modeling, UV unwrapping, texturing, rigging, water and smoke simulations, skinning, animating, rendering, particle and other simulations, non-linear editing, compositing, and creating interactive 3D applications, including video games, animated film, or visual effects. Blender's features include advanced simulation tools such as rigid, realistic body, fluid, cloth and softbody dynamics, modifier-based modeling tools, powerful character animation tools, a node-based material and compositing system and Python for embedded scripting. Released as free software under the GNU General Public License, Blender is available for a number of operating systems, including GNU/Linux, Mac OS X, FreeBSD, OpenBSD and Microsoft Windows.

HISTORY

Blender was developed as an in-house application by the Dutch animation studio NeoGeo and Not a Number Technologies (NaN). It was primarily authored by Ton

Roosendaal, who had previously written a ray tracer called Traces for Amiga in 1989. The name "Blender" was inspired by a song by Yello, from the album *Baby*. Roosendaal founded NaN in June 1998 to further develop and distribute the program. The programme was initially distributed as shareware until NaN went bankrupt in 2002.

The creditors agreed to release Blender under the terms of the GNU General Public License, for a one-time payment of 100,000 (US$100,670 at the time). On July 18, 2002, a Blender funding campaign was started by Roosendaal in order to collect donations and on September 7, 2002 it was announced that enough funds had been collected and that the Blender source code would be released. Today, Blender is free, open-source software and is, apart from the two half-time employees and the two full-time employees of the Blender Institute, developed by the community.

The Blender Foundation initially reserved the right to use dual licensing, so that, in addition to GNU GPL, Blender would have been available also under the "Blender License", which did not require disclosing source code but required payments to the Blender Foundation. However, this option was never exercised and was suspended indefinitely in 2005. Currently, Blender is solely available under GNU GPL.

SUZANNE

In January/February 2002 it was quite clear that NaN could not survive and would close the doors in March.

Nevertheless, they found the energy for doing at least one more release: 2.25. As a sort-of easter egg, a last personal tag, the artists and developers decided to add a chimpanzee primitive. It was created by Willem-Paul van Overbruggen (SLiD3), who also named it Suzanne, after the orangutan in the Kevin Smith film *Jay and Silent Bob Strike Back*.

Suzanne is Blender's alternative to more common "test models" such as the Utah Teapot and the Stanford Bunny. A low-polygon model with only 500 faces, Suzanne is often used as a quick and easy way to test material, animation, rigs, texture, and lighting setups, and is also frequently used in joke images. The largest Blender contest gives out an award called the Suzanne Awards.

FEATURES

Blender has a relatively small installation size and runs on several popular computing platforms, including Linux, Mac OS X, and Microsoft Windows, along with FreeBSD, IRIX, NetBSD, OpenBSD and Solaris. Unofficial ports are also available for AmigaOS 4, BeOS, MorphOS , Pocket PC and SkyOS. Though it is often distributed without documentation or extensive example scenes, the software contains features that are characteristic of high-end modeling software. Among its capabilities are:

- Support for a variety of geometric primitives, including polygon meshes, fast subdivision surface modeling,

Bezier curves, NURBS surfaces, metaballs, digital sculpting, and outline fonts.

- Versatile internal rendering capabilities and integration with YafaRay, a free software ray tracer.
- Keyframed animation tools including inverse kinematics, armature (skeletal), hook, curve and lattice-based deformations, shape keys (morphing), non-linear animation, constraints, vertex weighting, soft body dynamics including mesh collision detection, LBM fluid dynamics, Bullet rigid body dynamics, particle-based hair, and a particle system with collision detection.
- Modifiers to apply non-destructive effects.
- Python scripting for tool creation and prototyping, game logic, importing and/or exporting from other formats, task automation and custom tools.
- Basic non-linear video/audio editing and compositing capabilities.
- Game Blender, a sub-project, offers interactivity features such as collision detection, dynamics engine, and programmable logic. It also allows the creation of stand-alone, real-time applications ranging from architectural visualization to video game construction.
- A fully-integrated node-based compositor within the rendering pipeline.

USER INTERFACE

Blender has had a reputation as being difficult to learn for users accustomed to other 3D graphics software. Nearly every function has a direct keyboard shortcut and there can be several different shortcuts per key. Since Blender became free software, there has been effort to add comprehensive contextual menus as well as make the tool usage more logical and streamlined. There have also been efforts to visually enhance the user interface, with the introduction of color themes, transparent floating widgets, a new and improved object tree overview, and other small improvements (such as a color picker widget). Blender's user interface incorporates the following concepts:

EDITING MODES

The two primary modes of work are *Object Mode* and *Edit Mode*, which are toggled with the Tab key. Object mode is used to manipulate individual objects as a unit, while Edit mode is used to manipulate the actual object data. For example, Object Mode can be used to move, scale, and rotate entire polygon meshes, and Edit Mode can be used to manipulate the individual vertices of a single mesh. There are also several other modes, such as Vertex Paint, Weight Paint, and Sculpt Mode. The 2.45 release also had the UV Mapping Mode, but it was merged with the Edit Mode in 2.46 Release Candidate 1.

HOTKEY UTILIZATION

Most of the commands are accessible via hotkeys. Until the 2.x and especially the 2.3x versions, this was in fact the only way to give commands, and this was largely responsible for creating Blender's reputation as a difficult-to-learn program. The new versions have more comprehensive GUI menus.

NUMERIC INPUT

Numeric buttons can be "dragged" to change their value directly without the need to aim at a particular widget, thus saving screen real estate and time. Both sliders and number buttons can be constrained to various step sizes with modifiers like the Ctrl and Shift keys. Python expressions can also be typed directly into number entry fields, allowing mathematical expressions to be used to specify values.