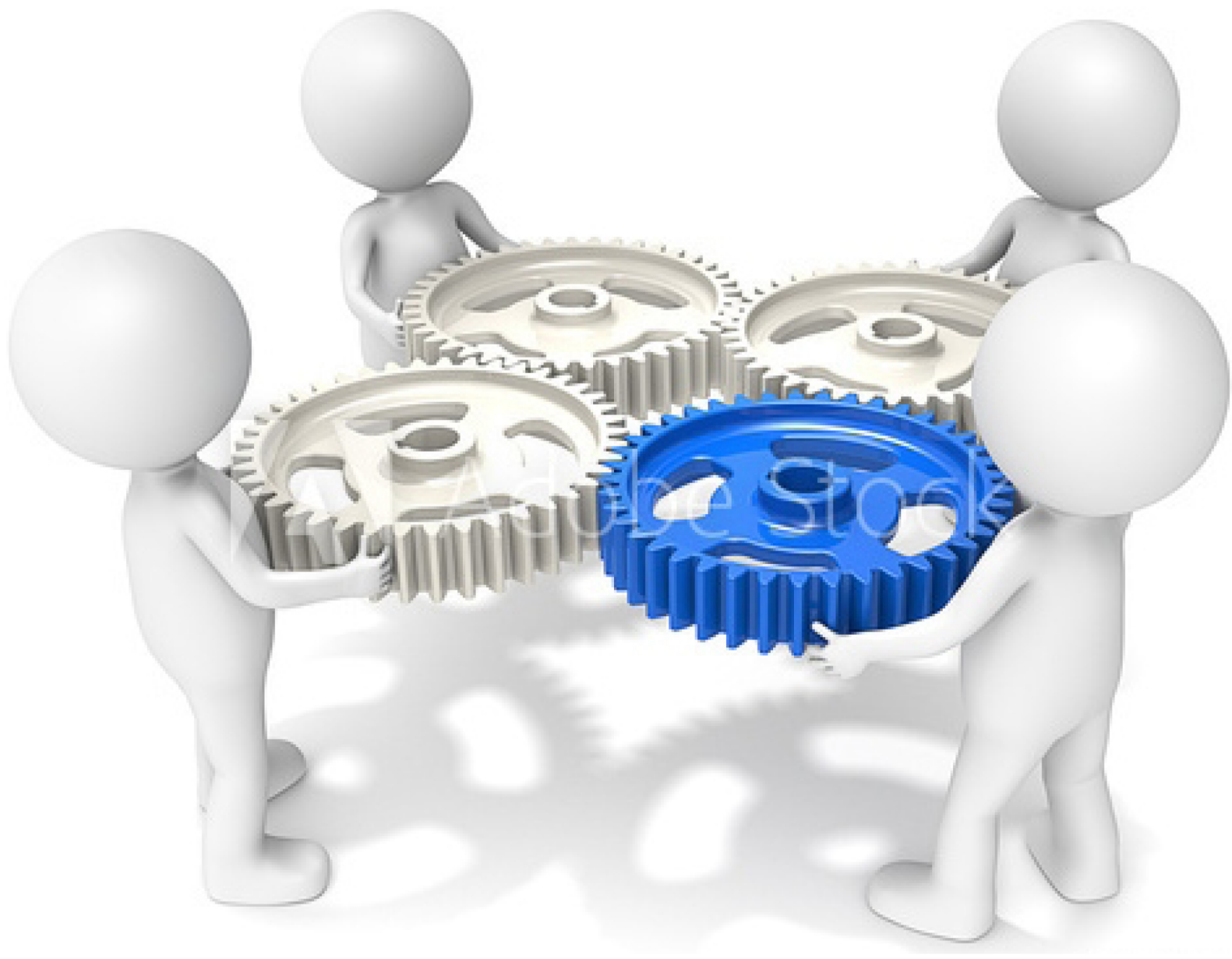


3-D Computer Graphics

Willis Meadows



3-D COMPUTER GRAPHICS

3-D COMPUTER GRAPHICS

Willis Meadows



3-D Computer Graphics
by Willis Meadows

Copyright© 2022 BIBLIOTEX

www.bibliotex.com

All rights reserved. No part of this book may be reproduced or used in any manner without the prior written permission of the copyright owner, except for the use brief quotations in a book review.

To request permissions, contact the publisher at info@bibliotex.com

Ebook ISBN: 9781984664228



Published by:

Bibliotex

Canada

Website: www.bibliotex.com

Contents

| | | |
|------------------|---|-----|
| Chapter 1 | An Introduction to 3D Computer Graphics | 1 |
| Chapter 2 | 3-D Geometric Primitives | 22 |
| Chapter 3 | Theory of Graphic Design | 45 |
| Chapter 4 | 3 Dimensional Transformation | 74 |
| Chapter 5 | Orthographic Projections | 92 |
| Chapter 6 | Design and Analysis Technologies | 117 |
| Chapter 7 | Computer Graphics Software | 136 |
| Chapter 8 | Sketchpad and Design Process | 185 |

1

An Introduction to 3D Computer Graphics

3D computer graphics (in contrast to 2D computer graphics) are graphics that utilize 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 (either inanimate or living). 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.

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.

3D computer graphics creation falls into three basic phases:

- *3D Modelling:* The process of forming a computer model of an object's shape
- *Layout and Animation:* The motion and placement of objects within a scene

- *3D Rendering:* The computer calculations that, based on light placement, surface types, and other qualities, generate the image

MODELLING

3D MODELLING

Modelling is the process of taking a shape and moulding it into a completed 3D mesh. The most typical means of creating a 3D model is to take a simple object, called a primitive, and extend or “grow” it into a shape that can be refined and detailed. Primitives can be anything from a single point (called a vertex), a two-dimensional line (an edge), a curve (a spline), to three dimensional objects (faces or polygons). Using the specific features of your chosen 3D software, each one of these primitives can be manipulated to produce an object. When you create a model in 3D, you’ll usually learn one method to create your model, and go back to it time and again when you need to create new models. There are three basic methods you can use to create a 3D model, and 3D artists should understand how to create a model using each technique.

- *Spline or Patch Modelling:* A spline is a curve in 3D space defined by at least two control points. The most common splines used in 3D art are bezier curves and NURBS (the software Maya has a strong NURBS modelling foundation.) Using splines to create a model is perhaps the oldest, most traditional form of 3D modelling available. A cage of splines is created to form a “skeleton” of the object you want

to create. The software can then create a patch of polygons to extend between two splines, forming a 3D skin around the shape. Spline modelling is not used very often these days for character creation, due to how long it takes to create good models. The models that are produced usually aren't useful for animation without a lot of modification. Spline modelling is used primarily for the creation of hard objects, like cars, buildings, and furniture. Splines are extremely useful when creating these objects, which may be a combination of angular and curved shapes. When creating a 3D scene that requires curved shapes, spline modelling should be your first choice.

- *Box Modelling:* Box modelling is possibly the most popular technique, and bears a lot of resemblance to traditional sculpting. In box modelling, one starts with a primitive (usually a cube) and begins adding detail by “slicing” the cube into pieces and extending faces of the cube to gradually create the form you're after. People use box modelling to create the basic shape of the model. Once practiced, the technique is very quick to get acceptable results. The downside is that the technique requires a lot of tweaking of the model along the way. Also, it is difficult to create a model that has a surface topology that lends well to animation. Box modelling is useful as a way to create organic models, like characters. Box modellers can also create hard objects like buildings, however precise curved shapes may be more difficult to create using this technique.

- *Poly Modelling/ edge Extrusion:* While it's not the easiest to get started with, poly modelling is perhaps the most effective and precise technique. In poly modelling, one creates a 3D mesh point-by-point, face-by-face. Often one will start out with a single quad (a 3D object consisting of 4 points) and extrude an edge of the quad, creating a second quad attached to the first. The 3D model is created gradually in this way. While poly modelling is not as fast as box modelling, it requires less tweaking of the mesh to get it "just right," and you can plan out the topology for animation ahead of time. Poly modellers use the technique to create either organic or hard objects, though poly modelling is best suited for organic models.

A Workflow that Works The workflow you choose to create a model will largely depend on how comfortable you are with a given technique, what object you're creating, and what your goals are for the final product. Someone who is creating an architectural scene, for example, may create basic models with cubes and other simple shapes to create an outline of the finished project. Meshes can then be refined or replaced with more detailed objects as you progress through the project. This is an organized, well-planned way to create a scene; it is a strategy used by professionals that makes scene creation straightforward. Beginners, on the other hand, tend to dive in headfirst and work on the most detailed objects first. This is a daunting way to work, and can quickly lead to frustration and overwhelm. Remember, sketch first, then refine. Likewise, when creating an organic model, beginners

tend to start with the most detailed areas first, and flesh out the remaining parts later, a haphazard way to create a character. This may be one reason why box modelling has grown to be so widely popular. A modeller can easily create the complete figure before refining the details, like eyes, lips, and ears. Perhaps the best strategy is to use a hybrid workflow when creating organic models. A well planned organic model is created using a combination of box modelling and poly modelling. The arms, legs, and torso can be sketched out with box modelling, while the fine details of the head, hands, and feet are poly modelled. This is a compromise professional modellers seek which prevents them from getting bogged down in details. It can make the difference between a completed character, and one that is never fleshed out beyond the head. Beginners would be wise to follow this advice.

Mesh Topology Another aspect of proper workflow is creating a model with an ideal 3D mesh topology. Topology optimization is usually associated with creating models used in animation. Models created without topology that flows in a smooth, circular pattern, may not animate correctly, which is why it is important to plan ahead when creating any 3D object that will be used for animation. The most frequently discussed topology is the proper creation or placement of edgeloops. An edgeloop is a ring of polygons placed in an area where the model may deform, as in the case of animation.

These rings of polygons are usually placed around areas where muscles might be, such as in the shoulder or elbow. Edgeloop placement is critical when creating faces. When edgeloops are ignored, models will exhibit “tearing” when animated, and the model will need to be reworked or scrapped

altogether in favour of a properly-planned model. Next Steps
The next step to creating great models is simply to practice and examine the work of artists you admire. Some of the best 3D modellers are also fantastic pencil-and-paper artists. It will be well worth your time to practice drawing, whether you're a character creator or a wanna-be architect.

Good modelling requires a lot of dedication. You'll need to thoroughly understand the software you're using, and the principles of good 3D model creation laid out above. Character artists will need to learn proportion and anatomy. The model describes the process of forming the shape of an object. The two most common sources of 3D models are those that an artist or engineer originates on the computer with some kind of 3D modelling tool, and models scanned into a computer from real-world objects. Models can also be produced procedurally or via physical simulation. Basically, a 3D model is formed from points called *vertices* (or *vertexes*) that define the shape and form *polygons*. A polygon is an area formed from at least three vertexes (a triangle). A four-point polygon is a *quad*, and a polygon of more than four points is an *n-gon*. The overall integrity of the model and its suitability to use in animation depend on the structure of the polygons.

LAYOUT AND ANIMATION

Before rendering into an image, objects must be placed (laid out) in a scene. This defines spatial relationships between objects, 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. These techniques

are often used in combination. As with modelling, physical simulation also specifies motion.

3D RENDERING

For those of us used to working in Photoshop and Illustrator it is important to realise that all that work is 2D, or two-dimensional. Photographs of real objects or painting them from scratch in Painter, they are still 2D. This is because we are either working with a pixel representation or flat objects, like lines, text, paths, *etc.* This is true even if we are attempting to simulate a 3D look. In 3D work, or three dimensions, we are producing a description of real objects with depth, scenes comprising many objects and the spatial relationships between them, along with the required lighting arrangements and viewing characteristics. The end result of 3D work is still usually 2D. This is either a still image or an animation, but it's still made up of pixels. In an ideal world our output would be three-dimensional too, as in a holographic projection or even a sculpture. This is a limitation of the output technologies that we have to work with at present, rather than an inherent characteristic of 3D work. Since, 3D printers exist (they are actually more like a numerically controlled milling machine in some ways), as do using LCD shutter glasses for direct 3D display, working completely in 3D is possible, just not the normal use.

Deep down, usually buried deep inside the software, our 3D work consists of rather mathematical descriptions of our scenes, such as place a sphere of radius k , with its centre at x, y, z point in space with a surface texture like stone. Thankfully, we rarely have to deal with the numerical level

unless we choose to. There are good reasons to dive down to the numerical level at times, such as exact placement. 3D software is largely click and drag operation these days for most common operations. It is important to remember that we are trying to represent things in the three-dimensional world that we are used to living in. Just as navigating around the real world can get you lost, so is it easy to become disoriented in 3D software.

KEEPING ORIENTED IN 3D

In 3D software the convention is to use a set of three coordinates, x, y and z. Co-ordinates can be absolute or relative. Absolute coordinates apply to the entire world that we are creating in the computer. Everything is specified relative to a universal origin, the centre of your digital universe, with coordinates of 0,0,0. Positive x values may lie to the right, negative ones to the left. Positive y values may be up and negative ones down from the origin. Positive z may be in front of and negative ones behind the origin. Absolute coordinates are used to position objects in our scene, to place cameras and lights, *etc.* Relative coordinates have their origin somewhere other than the world origin. For instance, in creating an object made up of many parts it may be more convenient to think in terms of positions relative to what you wish to consider the centre of the object.

How the software works can have an impact on how easy it is to keep oriented. Some Programmes, like Bryce, display only one window, so you only have one view of your objects/scene at a time. Other Programmes, like Vue d'Esprit or Lightwave, by default give you four views: a front, left and

top view plus the view through the main camera. This last solution is generally preferred but does tend to work best when you are using a large, high-resolution screen. This is why most of the consumer level Programmes use the one view approach, assuming home users have small screens, whilst professional software takes the four-view approach.

THE STAGES OF 3D WORK

The following are the main stages of creating a 3D work:

- Create objects;
- Place objects in relation to each other in scene;
- Place light sources;
- Place the camera or observer;
- Add textures to objects;
- Add atmospheric effects;
- Render to produce a final image or animation movie.

The exact order of this sequence is partly up to you and partly a function of the software that you are using. For instance, some software separates the creation of objects and their placing in the scene (as in Lightwave), others combine this into one step (as in Bryce). Likewise, sometimes the textures are placed on objects when you create them.

But they can also be added at the scene creation stage. Each person gradually finds their own order of working that suits their needs and the needs of the specific project. For projects involving many people there may be different order, or indeed some stages may be performed in parallel, than for projects where you are doing the whole thing. The order of steps can affect the performance of your software. The sequence given tends to produce the

least delays with most software, for reasons that will become clear as we progress through this series.

Creating objects and placing them in the scene is often called 'modelling'. This is because in creating an object and then a scene we are building a 'model' of it in the computer. Some software even separates the modelling function from the rest of the software by splitting the process into two Programmes. It is quite possible to do the modelling in on manufacturer's Programme and the rest of the process in another. I quite frequently use three different Programmes for this process, making use of the strengths of each, these being Poser and Bryce and Lightwave.

Light sources and a camera are necessary if you are to see anything of the wonderful model you have created. Light sources and cameras can be treated in much the same way as any other object. Light sources will have their own, special characteristics though, like the type of light source, whether it casts shadows, its colour, *etc.* The camera also has special characteristics, like its field of view, resolution of the resulting image(s), *etc.*

Rendering is the process of determining what the scene looks like from the camera position taking into account all the characteristics of the objects, light sources and their interaction. Rendering is usually a time consuming process for any scene of reasonable complexity. This can vary from a 'go get a cup of coffee' to 'lunch' up to a whole week, or more. This is one reason why high complexity rendering of still images or animations tends to require fast computers and lots of memory. One reason that the order with which you create your image(s) is important is that you will usually do

lots of little test renders along the way. Thus you want to leave the details which really slow the rendering down to as late in the sequence as possible.

WHY WOULD WE WANT TO USE 3D

We need to represent solid objects, whether in a still image for an ad or an animation to go in a movie. Since, real world objects are 3D, there will be times when a 3D representation is needed. Sure, we can paint or airbrush a 3D approximation but it will have a particular look, assuming that we have the skill level to create it. Working with 3D software creates a different look. This can vary from one with a very computer feel to a photorealistic one, depending on the software and what we do with it.

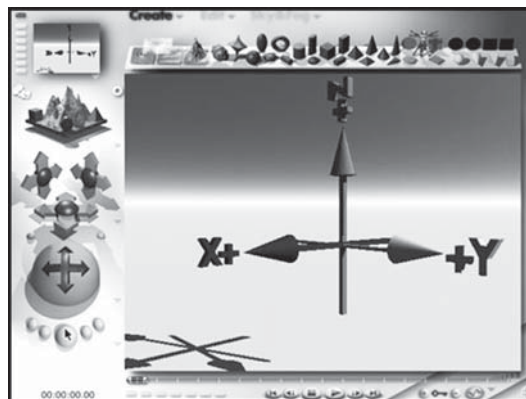
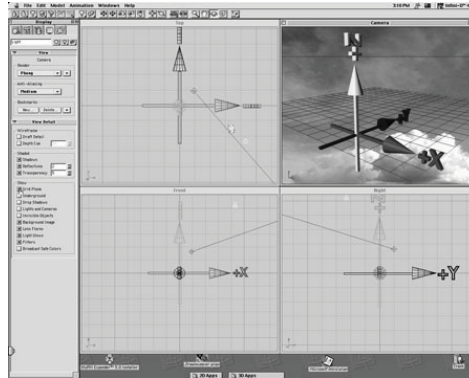
The major advantage of working with 3D software is that it is easy to produce changes. To change the viewpoint only requires that we move the camera and render. To change the lighting or reposition objects is equally easy. So having created a scene once, we can produce many different images from it. This is like photographing a real scene in everything from wide-angle to close-up, and from different positions.

3D software gives you flexibility. This very flexibility allows you to re-purpose images. You may do an illustration for a magazine ad and then the client comes back and wants an animation for a TV ad, or the web. Once you have built the models, you can re-use them repeatedly

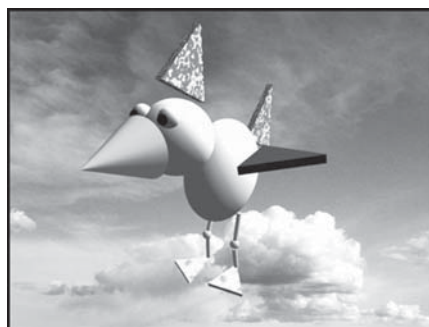
This screen grab of the old Metacreation's Infiniti-D 4.5 shows a four window, working environment. Three windows give front, top and side views whilst the fourth shows the camera view. This type of display, common to

3-D Computer Graphics

most of the higher-end 3D packages, works best on a high resolution, large screen.



The single view at atimedisplay, like this one from Bryce, works well on smaller displays. Usually keyboard shortcuts or button allow you to switch between views. Whilstnot as convenient as the four-window display it is quite workable. It seems natural once you get used to it



This simple cartoon bird was created out of basic object types and rendered in Infiniti-D 4.5. A background image was used.

WHAT SORTS OF OBJECTS

In most real scenes, the objects that we might want to incorporate will be complex. Unfortunately most 3D modellers and renderers don't support basic object types like 'tree', 'car', 'person' or 'house'. Such complex objects have to be created out of the actual object types that the renderer supports. The usual basic objects types are flat objects, like planes and polygons, and 3D objects like spheres, cylinders, cones, *etc.* Of course, you can also obtain libraries of already created objects. Some 3D Programmes come with lots of these, others few. There are web sites where people place free 'models' that you can download. There are also companies that specialise in creating 'models' that you can buy.

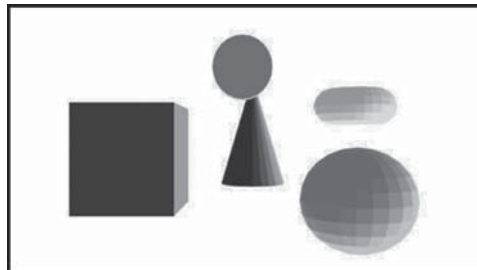
Polygons, for reasons that will become clearer later in this series, are the mainstay of most 3D modellers and renderers. A polygon is simply a shape made up of a number of straight lines, joined together to define a closed shape. The points that define the end of each line are called a vertex. Different Programmes allow variations on the basic polygon. Some Programmes require that polygons be totally flat, that all the vertices lie in a flat plane. Others allow curved polygons. Some require all polygons to have either three or four sides. Others allow you to construct polygons with greater numbers of sides. Many of these latter ones will actually subdivide the polygon into three or four sided ones before rendering, though this is usually hidden from the user. One major advantage

of three sided polygons, triangles, is that they have to be flat. Only four sided or higher polygons can have some vertices not in the same plane as the others. A variation on the polygon that you find in most 3D software is the infinite plane. As its name implies this plane is a flat surface that stretches off into infinity. Infinite planes are useful for things like water levels, cloud layers, *etc.*

Polygons are defined by the x , y , z coordinates of their vertices. It is not unusual to be required to define the vertices of a polygon in a particular order, such as clockwise or anticlockwise when looking at the front face of the polygon. Some software requires this to be able to calculate the surface normal. Surface normals are incredibly important in 3D work as they are used to work out how much light is hitting a surface, and thus it's colour. The surface normal points up from the surface of the polygon. Some software treats polygons as single sided, other software as double sided. 3D software that has single sided polygons will not display them if you are looking at their back surface. With such software if you want a bowl, for example, you have to define polygons forming both the inside and outside surfaces. Software that uses double sided polygons does not have this requirement, one layer of polygons can represent both the inside, and outside surfaces, though this is not natural, since, the bowl walls would have no thickness.

Basic 3D objects, like spheres, cylinders, boxes and cones are also incredibly useful. We can construct planets from spheres and tree trunks from cylinders, for instance. Since, these are the basic forms used in the construction of most man-made, and many natural, objects, they are indispensable.

Many Programmes, when you use one of these, create the basic object at a standard size. You can then usually modify the object by stretching it into the form you want. Other Programmes allow you to stretch out the shape when you insert it into the scene. This stretching process allows you to create oval footballs from a sphere, a rectangular building from a square cube and a long spear from a squat cylinder. Most software gives you the choice of doing this either by typing in numbers or by clicking and dragging. This stage of modifying the shape of your objects is usually much easier if you can easily switch between different views of the object, like front, side and top, either through having multiple views open at once or by switching views in the one window.



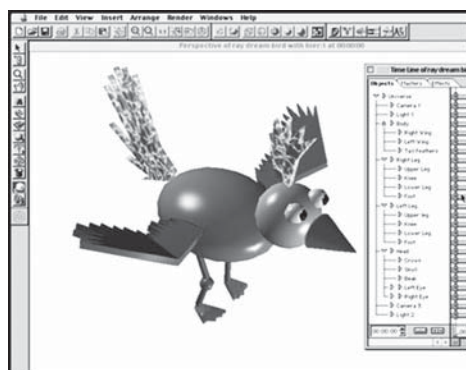
Boxes, spheres, cylinders, cones, polygons and text objects are the basic construction components available in most 3D software, as shown in this render done with Newtek's Inspire 3D. In some Programmes all these objects are constructed out of polygons, in others they are primitive objects that are rendered directly. If you examine the edges of the sphere and cone you can see that they are constructed out of polygons.

CREATING COMPOSITE OBJECTS

If all objects are treated as individual ones, you end up with a heap of them to try to manage. Since, most basic

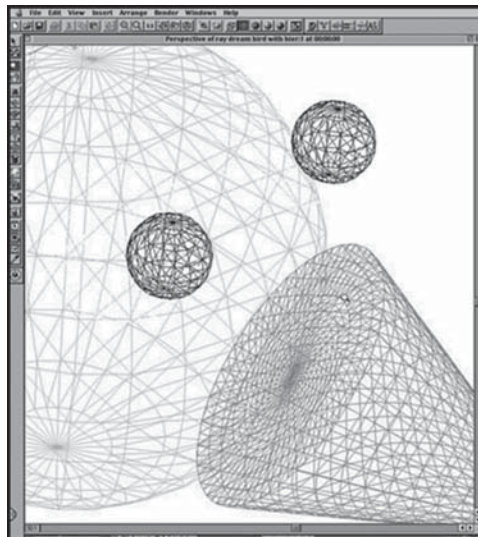
objects will actually be used to construct more complex objects it is useful to be able to group objects together that form parts of a whole. Thus we might create an object 'person' with parts 'head', 'body', 'arm12, 'arm22, 'leg12 and 'leg22. Then 'leg12 consists of 'upper', 'lower' and 'foot'. And so on. Building up complex objects out of hierarchies of other parts makes life a lot easier. If you want to move a whole object you can simply select the top level and move it, knowing that all the component parts will move too. Otherwise you would have to separately select every component and move them, and hope you didn't forget some small parts.

Object hierarchies are most flexible when you can give names to each component part. Such hierarchies are also essential to making character animation easier. Some Programmes allow you to readily display object hierarchies in a diagram form that shows the relationships between parts, similar to the folder hierarchy views that most operating systems allow. Software that doesn't do this is certainly harder to use for some things.



This screen grab, from Ray Dream Studio, shows a cartoon bird and it's hierarchical construction. Unfortunately too few Programmes provide this sort of display.

Another type of object related to the above is a polygon mesh. A mesh is a set of polygons which are joined together to represent a surface of some complexity. A good example of this is the polygon mesh that Bryce 3D uses to represent the shape of the landscape. The process of creating a polygon mesh usually does not require that the user manually position each vertex of each polygon in the mesh. Various other convenient methods are available. We'll examine these in later chapter.

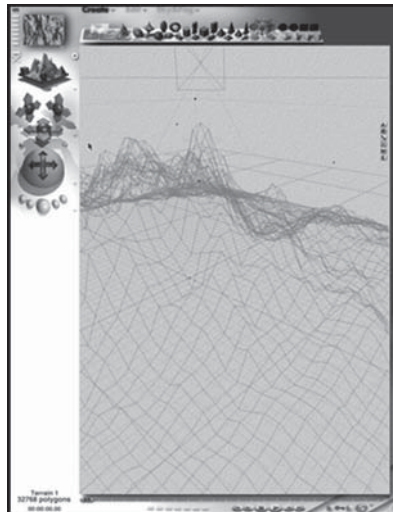


This close-up of part of a bird model in Ray Dream Studio shows how this Programme tessellates spheres into polygonal meshes.

THERE DIFFERENCES OF APPROACH

There are two choices the software developers have to make: what primitive objects are to be supported; and what rendering method is to be used? These two questions are interrelated. The rendering method determines what actual primitive objects the software works with to create images.

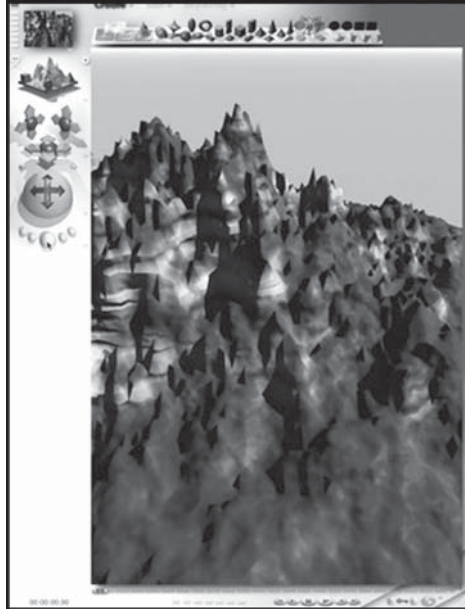
How we want the user interface to be will determine what primitive objects are available to the user. For a number of reasons that we will examine in the next part of the course, certain rendering techniques can only actually support polygons, whilst others can actually handle spheres, cylinders, *etc.* So a Programme that has to use polygons for rendering will convert a sphere into a polygonal approximation, in a process called tessellation, before actually rendering an image. This creates more primitive objects to render but allows the renderer to be highly optimised for the handling of polygons. A Programme which can directly support spheres, say, does not have to do this conversion and thus renders fewer objects in your scene but requires specialised Programme code for each object type it supports.



Another use for polygon meshes is to represent irregular objects, like this landscape in Bryce 3D.

These internal differences in approach are what make some 3D packages good for some types of work and others more suitable for others. Some will handle transparent objects superbly, other handle interior lighting well, for example.

3-D Computer Graphics



Some will make dealing with certain types of objects easy, whereas others make those objects hard but others easy. It is for these reasons that many people working with 3D software will use a number of packages for different parts of the process. Whilst this is certainly not necessary, it can be a useful approach. It's the same as people using Painter for some things and Photoshop for others, sometimes switching backwards and forwards between the two.

The designers of 3D software have to make a complex set of choices based on their priorities. These choices lead to the differences in single or double sided polygons, whether tessellation is done and what types of rendering options are available, to pick just three.

Some choices will speed up the execution of the Programme whilst others will slow it down. These tradeoffs account for the huge variety that we encounter in 3D Programmes.

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 modelling 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.

2

3-D Geometric Primitives

3D PRIMITIVES

Primitives are the building blocks of 3D—basic geometric forms that you can use as is or modify with transforms and Booleans. Although it's possible to create most of these objects by lathing or extruding 2D shapes, most software packages build them in for speed and convenience.

The most common 3D primitives are cubes, pyramids, cones, spheres, and tori. Like 2D shapes, these primitives can have a resolution level assigned to them so that you can make them look smoother by boosting the number of sides and steps used to define them.

Inappropriate use of unmodified primitives is probably one of the most common novice mistakes. By their very nature, primitives have a mathematically perfect appearance that screams, “I am a 3D object!” 2D shapes as the basis

for your 3D objects. Primitives are best suited as building blocks for more complex forms or for use in your scene's background, where any extra detail will be lost anyway. Also, primitives can be very useful as foreground objects when they're altered through the use of transforms and modifiers.

Some programs may offer an array of additional, more sophisticated primitives that may be better suited to foreground use because they offer beveled or rounded edges instead of that "chopped-off 3D look". A number of these extended primitives would be a real chore to create from scratch, so they can be a time-saver in that regard as well.

That about does it for building basic 3D objects. It's time to move on to the particulars of positioning these objects in 3D space. It's a big universe in there, so it's essential to know how to get around.

GEOMETRIC PRIMITIVE

The term geometric primitive, or prim, in computer graphics and CAD systems is used in various senses, with the common meaning of the simplest (i.e. 'atomic' or irreducible) geometric objects that the system can handle (draw, store). Sometimes the subroutines that draw the corresponding objects are called "geometric primitives" as well. The most "primitive" primitives are point and straight line segment, which were all that early vector graphics systems had. In constructive solid geometry, primitives are simple geometric shapes such as a cube, cylinder, sphere, cone, pyramid, torus.

Modern 2D computer graphics systems may operate with primitives which are lines (segments of straight lines, circles and more complicated curves), as well as shapes (boxes, arbitrary polygons, circles).

A common set of two-dimensional primitives includes lines, points, and polygons, although some people prefer to consider triangles primitives, because every polygon can be constructed from triangles. All other graphic elements are built up from these primitives. In three dimensions, triangles or polygons positioned in three-dimensional space can be used as primitives to model more complex 3D forms. In some cases, curves (such as Bézier curves, circles, etc.) may be considered primitives; in other cases, curves are complex forms created from many straight, primitive shapes.

COMMON PRIMITIVES

- points
- lines and line segments
- planes
- circles and ellipses
- triangles, 'quads' or arbitrary polygons
- spline curves

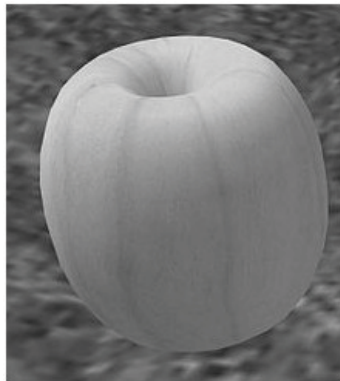


Fig. A 3D torus prim created in Second Life

In 3D applications, basic geometric shapes and forms are considered to be primitives rather than the above list. Such shapes and forms include:

- spheres
- cubes or boxes
- toroids
- cylinders
- pyramids
- triangle meshes or polygon meshes are a common unit from which scenes are composed, which are in turn composed of connected vertices.

These are considered to be primitives in 3D modelling because they are the building blocks for many other shapes and forms. A 3D package may also include a list of extended primitives which are more complex shapes that come with the package. For example, a teapot is listed as a primitive in 3D Studio Max.

IN 3D MODELLING

In CAD software or 3D modelling, the interface may present the user with the ability to create primitives which may be further modified by edits. For example in the practice of box modelling the user will start with a cuboid, then use extrusion and other operations to create the model. In this use the primitive is just a convenient starting point, rather than the fundamental unit of modelling.

IN GRAPHICS HARDWARE

Various graphics accelerators exist with hardware acceleration for rendering specific primitives such as lines

or triangles, frequently with texture mapping and shaders. Modern 3D accelerators typically accept sequences of triangles as triangle strips.

3-D OBJECT REPRESENTATION

Representation schemes for solid objects are divided into two categories as follows: 1. Boundary Representation (B-reps). It describes a three dimensional object as a set of surfaces that separate the object interior from the environment. Examples are polygon facets and spline patches.

SPACE PARTITIONING REPRESENTATION

It describes the interior properties, by partitioning the spatial region containing an object into a set of small, nonoverlapping, contiguous solids(usually cubes). Eg: Octree Representation.

POLYGON SURFACES

Polygon surfaces are boundary representations for a 3D graphics object is a set of polygons that enclose the object interior.

POLYGON TABLES

The polygon surface is specified with a set of vertex coordinates and associated attribute parameters.

For each polygon input, the data are placed into tables that are to be used in the subsequent processing.

Polygon data tables can be organized into two groups: Geometric tables and attribute tables.

Geometric Tables Contain vertex coordinates and parameters to identify the spatial orientation of the polygon surfaces.

Attribute tables Contain attribute information for an object such as parameters specifying the degree of transparency of the object and its surface reflectivity and texture characteristics. A convenient organization for storing geometric data is to create three lists:

1. The Vertex Table: Coordinate values for each vertex in the object are stored in this table.
2. The Edge Table: It contains pointers back into the vertex table to identify the vertices for each polygon edge.
3. The Polygon Table: It contains pointers back into the edge table to identify the edges for each polygon.

Listing the geometric data in three tables provides a convenient reference to the individual components (vertices, edges and polygons) of each object.

The object can be displayed efficiently by using data from the edge table to draw the component lines.

Extra information can be added to the data tables for faster information extraction. For instance, edge table can be expanded to include forward points into the polygon table so that common edges between polygons can be identified more rapidly.

E1 : V1, V2, S1

E2 : V2, V3, S1

E3 : V3, V1, S1, S2

E4 : V3, V4, S2

E5 : V4, V5, S2

E6 : V5, V1, S2

is useful for the rendering procedure that must vary surface shading smoothly across the edges from one polygon to the next. Similarly, the vertex table can be expanded so that vertices are cross-referenced to corresponding edges.

Additional geometric information that is stored in the data tables includes the slope for each edge and the coordinate extends for each polygon. As vertices are input, we can calculate edge slopes and we can scan the coordinate values to identify the minimum and maximum x, y and z values for individual polygons.

The more information included in the data tables will be easier to check for errors. Some of the tests that could be performed by a graphics package are:

1. That every vertex is listed as an endpoint for at least two edges.
2. That every edge is part of at least one polygon.
3. That every polygon is closed.
4. That each polygon has at least one shared edge.
5. That if the edge table contains pointers to polygons, every edge referenced by a polygon pointer has a reciprocal pointer back to the polygon.

Plane Equations:

To produce a display of a 3D object, we must process the input data representation for the object through several procedures such as,

- Transformation of the modeling and world coordinate descriptions to viewing coordinates.
- Then to device coordinates:
- Identification of visible surfaces
- The application of surface-rendering procedures.

For these processes, we need information about the spatial orientation of the individual surface components of the object. This information is obtained from the vertex coordinate value and the equations that describe the polygon planes.

The equation for a plane surface is $Ax + By + Cz + D = 0$

Where (x, y, z) is any point on the plane, and the coefficients A, B, C and D are constants describing the spatial properties of the plane.

We can obtain the values of A, B, C and D by solving a set of three plane equations using the coordinate values for three non collinear points in the plane.

For that, we can select three successive polygon vertices (x_1, y_1, z_1) , (x_2, y_2, z_2) and (x_3, y_3, z_3) and solve the following set of simultaneous linear plane equations for the ratios A/D , B/D and C/D .

$$(A/D)x_k + (B/D)y_k + (C/D)z_k = -1, \quad k=1,2,3$$

The solution for this set of equations can be obtained in determinant form, using Cramer's rule as

Expanding the determinants, we can write the calculations for the plane coefficients in the form:

$$A = y_1 (z_2 - z_3) + y_2 (z_3 - z_1) + y_3 (z_1 - z_2)$$

$$B = z_1 (x_2 - x_3) + z_2 (x_3 - x_1) + z_3 (x_1 - x_2)$$

$$C = x_1 (y_2 - y_3) + x_2 (y_3 - y_1) + x_3 (y_1 - y_2)$$

$$D = -x_1 (y_2 z_3 - y_3 z_2) - x_2 (y_3 z_1 - y_1 z_3) - x_3 (y_1 z_2 - y_2 z_1)$$

As vertex values and other information are entered into the polygon data structure, values for A, B, C and D are computed for each polygon and stored with the other polygon data.

Plane equations are used also to identify the position of spatial points relative to the plane surfaces of an object. For any point (x, y, z) not on a plane with parameters A,B,C,D, we have

$$Ax + By + Cz + D = 0$$

We can identify the point as either inside or outside the plane surface according to the sign (negative or positive) of $Ax + By + Cz + D$:

If $Ax + By + Cz + D < 0$, the point (x, y, z) is inside the surface. If $Ax + By + Cz + D > 0$, the point (x, y, z) is outside the surface.

These inequality tests are valid in a right handed Cartesian system, provided the plane parameters A,B,C and D were calculated using vertices selected in a counter clockwise order when viewing the surface in an outside-to-inside direction.

Polygon Meshes

A single plane surface can be specified with a function such as fillArea. But when object surfaces are to be tiled, it is more convenient to specify the surface facets with a mesh function. One type of polygon mesh is the triangle

strip. A triangle strip formed with 11 triangles connecting 13 vertices. Another similar function in the quadrilateral mesh, which generates a mesh of $(n-1)$ by $(m-1)$ quadrilaterals, given the coordinates for an n by m array of vertices.

Curved Lines and Surfaces Displays of three dimensional curved lines and surface can be generated from an input set of mathematical functions defining the objects or from a set of user specified data points. When functions are specified, a package can project the defining equations for a curve to the display plane and plot pixel positions along the path of the projected function. For surfaces, a functional description is decorated to produce a polygon-mesh approximation to the surface.

Spline Representations A Spline is a flexible strip used to produce a smooth curve through a designated set of points. Several small weights are distributed along the length of the strip to hold it in position on the drafting table as the curve is drawn.

The Spline curve refers to any sections curve formed with polynomial sections satisfying specified continuity conditions at the boundary of the pieces.

A Spline surface can be described with two sets of orthogonal spline curves. Splines are used in graphics applications to design curve and surface shapes, to digitize drawings for computer storage, and to specify animation paths for the objects or the camera in the scene. CAD applications for splines include the design of automobiles bodies, aircraft and spacecraft surfaces, and ship hulls.

Interpolation and Approximation Splines Spline curve can be specified by a set of coordinate positions called control points which indicates the general shape of the curve. These control points are fitted with piecewise continuous parametric polynomial functions in one of the two ways. 1. When polynomial sections are fitted so that the curve passes through each control point the resulting curve is said to interpolate the set of control points.

A set of six control points interpolated with piecewise continuous polynomial sections

When the polynomials are fitted to the general control point path without necessarily passing through any control points, the resulting curve is said to approximate the set of control points.

A set of six control points approximated with piecewise continuous polynomial sections.

Interpolation curves are used to digitize drawings or to specify animation paths. Approximation curves are used as design tools to structure object surfaces. A spline curve is designed, modified and manipulated with operations on the control points. The curve can be translated, rotated or scaled with transformation applied to the control points. The convex polygon boundary that encloses a set of control points is called the convex hull. The shape of the convex hull is to imagine a rubber band stretched around the position of the control points so that each control point is either on the perimeter of the hull or inside it. Convex hull shapes (dashed lines) for two sets of control points.

PARAMETRIC CONTINUITY CONDITIONS

For a smooth transition from one section of a piecewise parametric curve to the next various continuity conditions are needed at the connection points.

If each section of a spline is described with a set of parametric coordinate functions or the form $x = x(u)$, $y = y(u)$, $z = z(u)$, $u_1 \leq u \leq u_2$

We set parametric continuity by matching the parametric derivatives of adjoining curve sections at their common boundary.

Zero order parametric continuity referred to as C0 continuity, means that the curves meet. (i.e) the values of x, y , and z evaluated at u_2 for the first curve section are equal. Respectively, to the value of x, y , and z evaluated at u_1 for the next curve section. First order parametric continuity referred to as C1 continuity means that the first parametric derivatives of the coordinate functions in equation (a) for two successive curve sections are equal at their joining point.

Second order parametric continuity, or C2 continuity means that both the first and second parametric derivatives of the two curve sections are equal at their intersection.

Higher order parametric continuity conditions are defined similarly.

GEOMETRIC CONTINUITY CONDITIONS

To specify conditions for geometric continuity is an alternate method for joining two successive curve sections.

The parametric derivatives of the two sections should be

proportional to each other at their common boundary instead of equal to each other.

Zero order Geometric continuity referred as G0 continuity means that the two curves sections must have the same coordinate position at the boundary point.

First order Geometric Continuity referred as G1 continuity means that the parametric first derivatives are proportional at the interaction of two successive sections.

Second order Geometric continuity referred as G2 continuity means that both the first and second parametric derivatives of the two curve sections are proportional at their boundary. Here the curvatures of two sections will match at the joining position.

Spline specifications There are three methods to specify a spline representation:

1. We can state the set of boundary conditions that are imposed on the spline; (or)
2. We can state the matrix that characterizes the spline; (or)
3. We can state the set of blending functions that determine how specified geometric constraints on the curve are combined to calculate positions along the curve path.

To illustrate these three equivalent specifications, suppose we have the following parametric cubic polynomial representation for the x coordinate along the path of a spline section.

$$x(u) = au^3 + bu^2 + cu + d \quad 0 \leq u \leq 1$$

Boundary conditions for this curve might be set on the endpoint coordinates $x(0)$ and $x(1)$ and on the parametric first derivatives at the endpoints $x'(0)$ and $x'(1)$. These boundary conditions are sufficient to determine the values of the four coordinates ax , bx , cx and dx . From the boundary conditions we can obtain the matrix that characterizes this spline curve by first rewriting as the matrix product

where U is the row matrix of power of parameter u and C is the coefficient column matrix. Using we can write the boundary conditions in matrix form and solve for the coefficient matrix C .

$$C = Mspline \cdot Mgeom$$

Where $Mgeom$ in a four element column matrix containing the geometric constraint values on the spline and $Mspline$ in the 4×4 matrix that transforms the geometric constraint values to the polynomial coefficients and provides a characterization for the spline curve.

Matrix $Mgeom$ contains control point coordinate values and other geometric constraints. We can substitute the matrix representation for C into equation to obtain.

$$x(u) = U \cdot Mspline \cdot Mgeom$$

The matrix $Mspline$, characterizing a spline representation, called the basis matrix is useful for transforming from one spline representation to another.

Finally we can expand equation to obtain a polynomial representation for coordinate x in terms of the geometric constraint parameters.

$x(u) = \sum g_k \cdot BF_k(u)$ where g_k are the constraint parameters, such as the control point coordinates and slope

of the curve at the control points and $BF_k(u)$ are the polynomial blending functions.

3D ANIMATION SOFTWARE IN THE 1990S

There were many developments, mergers and deals in the 3D software industry in the '90s and later.

- Wavefront followed the success of *Personal Visualiser* with the release of *Dynamation* in 1992, a powerful tool for interactively creating and modifying realistic, natural images of dynamic events. In 1993, Wavefront acquired Thomson Digital Images (TDI), with their innovative product *Explore*, a tool suite that included *3Design* for modelling, *Anim* for animation, and *Interactive Photorealistic Renderer*(IPR) for rendering. In 1995, Wavefront was bought by Silicon Graphics, and merged with Alias.
- Alias Research continued the success of *PowerAnimator* with movies like *Terminator 2: Judgment Day*, *Batman Returns* and *Jurassic Park*, and in 1993 started the development of a new entertainment software, which was later to be named *Maya*. Alias found customers in animated film, TV series, visual effects, and video games, and included many prominent studios, such as Industrial Light & Magic, Pixar, Sony Pictures Imageworks, Walt Disney, and Warner Brothers. Other Alias products were developed for applications in architecture and engineering. In 1995, SGI purchased both Alias Research and Wavefront in a 3-way deal, and the merged company Alias Wavefront was launched.

- Alias Wavefront's new mission was to focus on developing the world's most advanced tools for the creation of digital content. *PowerAnimator* continued to be used for visual effects and movies (such as *Toy Story* and *Batman Forever*), and also for video games. Further development of the *Maya* software went ahead, adding new features such as motion capture, facial animation, motion blur, and "time warp" technology. CAD industrial design products like *AliasStudio* and *Alias Designer* became standardized on Alias | Wavefront software. In 1998, Alias | Wavefront launched *Maya* as its new 3D flagship product, and this soon became the industry's most important animation tool. *Maya* was the merger of three packages—Wavefront's *Advanced Visualizer*, Alias's *Power Animator*, and TDI's *Explore*. In 2003 the company was renamed simply "Alias". In 2004, SGI sold the business to a private investment firm, and it was later renamed to Alias Systems Corporation. In 2006, the company was bought by Autodesk.
- Softimage developed further features for *Creative Environment*, including the *Actor Module* (1991) and *Eddie* (1992), including tools such as inverse kinematics, enveloping, metaclay, flock animation, and many others. Softimage customers include many prominent production companies, and Softimage has been used to create animation for hundreds of major feature films and games. In 1994, Microsoft acquired Softimage, and renamed the package *Softimage 3D*, releasing a Windows NT port two years later. In 1998,

after helping to port the products to Windows and financing the development of *Softimage* and *Softimage|DS*, Microsoft sold the Softimage unit to Avid Technology, who was looking to expand its visual effect capabilities. Then, in 2008, Autodesk acquired the brand and the animation assets of Softimage from Avid, thereby ending Softimage Co. as a distinct entity. The video-related assets of Softimage, including *Softimage|DS* (now *Avid|DS*) continue to be owned by Avid.

- Autodesk Inc's PC DOS-based *3D Studio* was eventually superseded in 1996 when The Yost Group developed 3D Studio Max for Windows NT. Priced much lower than most competitors, *3D Studio Max* was quickly seen as an affordable solution for many professionals. Of all animation software, *3D Studio Max* serves the widest range of users. It is used in film and broadcast, game development, corporate and industrial design, education, medical, and web design. In 2006, Autodesk acquired Alias, bringing the *StudioTools* and *Maya* software products under the Autodesk banner, with *3D Studio Max* rebranded as *Autodesk 3ds Max*, and *Maya* as *Autodesk Maya*. Now one of the largest software companies in the world, Autodesk serves more than 4 million customers in over 150 countries.
- Side Effects Software's *PRISMS* was used extensively to create visual effects for broadcast and feature films into the '90s, with projects like *Twister*, *Independence Day*, and *Titanic*. In 1996, Side Effects Software introduced *Houdini*, a next-generation 3D package

that proved to be more sophisticated and artist-friendly than its predecessor. *Houdini* is used around the world to develop cutting edge 3D animation in the film, broadcast and gaming industries, and Side Effects Software has consistently proved itself to be an industry innovator.

CGI IN THE 2000S

2000 BREAKTHROUGH CAPTURE OF THE REFLECTANCE FIELD OVER THE HUMAN FACE

In 2000, a team led by Paul Debevec managed to adequately capture (and simulate) the reflectance field over the human face using the simples of light stages. which was the last missing piece of the puzzle to make digital look-alikes of known actors.

MOTION CAPTURE, PHOTOREALISM, AND UNCANNY VALLEY

The first mainstream cinema film fully made with motion capture was the 2001 Japanese-American *Final Fantasy: The Spirits Within* directed by Hironobu Sakaguchi, which was also the first to use photorealistic CGI characters. The film was not a box-office success. Some commentators have suggested this may be partly because the lead CGI characters had facial features which fell into the “uncanny valley”. In 2002, Peter Jackson’s *The Lord of the Rings: The Two Towers* was the first feature film to use a real-time motion capture system, which allowed the actions of actor Andy Serkis to be fed direct into the 3D CGI model of Gollum as it was

being performed. Motion capture is seen by many as replacing the skills of the animator, and lacking the animator's ability to create exaggerated movements that are impossible to perform live. The end credits of Pixar's film *Ratatouille* (2007) carry a stamp certifying it as "100% Pure Animation — No Motion Capture!" However, proponents point out that the technique usually includes a good deal of adjustment work by animators as well. Nevertheless, in 2010, the US Film Academy (AMPAS) announced that motion-capture films will no longer be considered eligible for "Best Animated Feature Film" Oscars, stating "Motion capture by itself is not an animation technique."

VIRTUAL CINEMATOGRAPHY

The early 2000s saw the advent of fully virtual cinematography with its audience debut considered to be in the 2003 movies *Matrix Reloaded* and *Matrix Revolutions* with its digital look-alikes so convincing that it is often impossible to know if some image is a human imaged with a camera or a digital look-alike shot with a simulation of a camera. The scenes built and imaged within virtual cinematography are the "*Burly brawl*" and the end showdown between Neo and Agent Smith. With conventional cinematographic methods the burly brawl would have been prohibitively time consuming to make with years of compositing required for a scene of few minutes. Also a human actor could not have been used for the end showdown in *Matrix Revolutions*: Agent Smith's cheekbone gets punched in by Neo leaving the digital look-alike naturally unhurt.

3D ANIMATION SOFTWARE IN THE 2000S

- Blender (software) is a free open source virtual cinematography package, used by professionals and enthusiasts alike.
- Poser is another DIY 3D graphics program especially aimed at user-friendly animation of soft objects
- Pointstream Software is a professional optical flow program that uses a pixel as its basic primitive form usually tracked over a multi-camera setup from the esteemed Arius3D, makers of the XYZRGB scanner, used in the production process of the Matrix sequels

CGI IN THE 2010S

In SIGGRAPH 2013 Activision and USC presented a real-time digital face look-alike of “Ira” utilizing the USC light stage X by Ghosh et al. for both reflectance field and motion capture. The end result, both precomputed and real-time rendered with the state-of-the-artGraphics processing unit: *Digital Ira*, looks fairly realistic. Techniques previously confined to high-end virtual cinematography systems are rapidly moving into the video games and leisure applications.

FURTHER DEVELOPMENTS

New developments in computer animation technologies are reported each year in the USA at SIGGRAPH, the largest annual conference on computer graphics and interactive techniques, and also at Eurographics, and at other conferences around the world.

SOFTWARE

3D computer graphics software produces computer-generated imagery (CGI) through 3D modeling and 3D rendering or produces 3D models for analytic, scientific and industrial purposes.

MODELING

3D modeling software is a class of 3D computer graphics software used to produce 3D models. Individual programs of this class are called modeling applications or modelers.

3D modelers allow users to create and alter models via their 3D mesh. Users can add, subtract, stretch and otherwise change the mesh to their desire. Models can be viewed from a variety of angles, usually simultaneously. Models can be rotated and the view can be zoomed in and out.

3D modelers can export their models to files, which can then be imported into other applications as long as the metadata are compatible. Many modelers allow importers and exporters to be plugged-in, so they can read and write data in the native formats of other applications.

Most 3D modelers contain a number of related features, such as ray tracers and other rendering alternatives and texture mapping facilities. Some also contain features that support or allow animation of models. Some may be able to generate full-motion video of a series of rendered scenes (i.e. animation).

COMPUTER-AIDED DESIGN (CAD)

Computer aided design software may employ the same fundamental 3D modeling techniques that 3D modeling

software use but their goal differs. They are used in computer-aided engineering, computer-aided manufacturing, Finite element analysis, product lifecycle management, 3D printing and computer-aided architectural design.

COMPLEMENTARY TOOLS

After producing video, studios then edit or composite the video using programs such as Adobe Premiere Pro or Final Cut Pro at the mid-level, or Autodesk Combustion, Digital Fusion, Shake at the high-end. Match moving software is commonly used to match live video with computer-generated video, keeping the two in sync as the camera moves.

Use of real-time computer graphics engines to create a cinematic production is called machinima.

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.

DIFFERENCES WITH OTHER TYPES OF COMPUTER GRAPHICS

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.

PSEUDO-3D AND *TRUE 3D*

Some video games use restricted projections of three-dimensional environments, such as isometric graphics or virtual cameras with fixed angles, either as a way to improve performance of the game engine, or for stylistic and gameplay concerns. Such games are said to use pseudo-3D graphics. By contrast, games using 3D computer graphics without such restrictions are said to use *true 3D*.

3

Theory of Graphic Design

Graphic design has often looked to architecture as an intellectual model. We long to infuse our work with the same kind of dense theoretical knowledge and the same kind of broad ranging, legendary critiques. But we're not architects. We're graphic designers. Our role is less defined. We cross between print and web, 2-D and 3-D. Our work is easier to produce and more ephemeral. This fluidity, coupled with a discipline-wide pragmatic streak, makes it difficult to establish a defined body of graphic design theory.

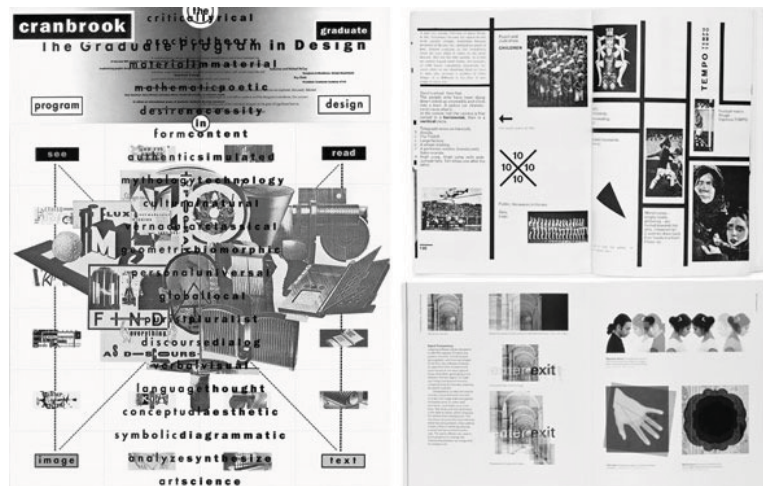
OR DOES IT?

Graphic designers have written about the ideas behind their work since the inception of the profession. Consider F. T. Marinetti, László Moholy-Nagy, Herbert Bayer, Josef Müller-Brockman, Karl Gerstner, Katherine McCoy, Jan van Toorn and, more recently, Jessica Helfand, Dmitri Siegel

and Kenya Hara. This body of work is small compared to architecture and fine arts, but it is passionate and smart.

Texts about graphic design fall under different categories of “theory.” Some analyze the process of making. Think Bauhaus experiments, methodologies that fall under the umbrella of International Typographic Style, and contemporary explorations labeled “design research.” Some texts examine the ideas behind the visual work. Authors “read” designs or design texts and put them into a wider historical/cultural context. And some apply outside theoretical discourses to the field of graphic design—deconstruction, semiotics, gender studies. Many seminal texts, of course, blur such categorizations.

Through my research I work to emphasize the value of our own theoretical base and inspire others to read and write more. Working on a recent book project got me thinking about a range of issues that face the profession today. Theory can help us address them.



(Clockwise from left): Katherine McCoy’s “See Read” poster for Cranbrook Graduate Design, 1989, a photographic collage

of recent graduate student work overlaid by a list of possibly opposing design values and a diagram of communication theories—a model for how deconstruction and structuralist/poststructuralist literary theories might be applied to graphic design’s visual and verbal processes; a spread from László Moholy-Nagy’s *Malerei, Photographie, Film* (Painting, Photography, Film), 1925; and a spread from *Graphic Design: The New Basics* (New York: Princeton Architectural Press, 2008), written and designed by Ellen Lupton and Jennifer Cole Phillips, in which Lupton explores emerging universals within the practice of graphic design, including newly relevant concepts like transparency and layering.

DESIGN INCREASINGLY LIVES IN THE ACTIONS OF ITS USERS

Think Flickr, Facebook, Etsy, Lulu, Threadless and the multitude of blogs. Users approach software and the web with the expectation of filling in their own content and shaping their own visual identities—often with guidance from prepackaged forms.

Dmitri Siegel calls this phenomenon “the templated mind.” Designers are grappling with their own place in this DIY phenomenon.

Creativity is no longer the sole territory of a separate “creative class.” Designers can lead this new participatory culture by developing frameworks that enable others to create; doing so, however, means allowing our once-specialized skills to become more widespread and accessible. That transfer of knowledge is threatening to some, liberating to others.

TECHNOLOGY ALTERS OUR AESTHETICS EVEN AS WE STRUGGLE AGAINST IT

Designers everywhere strive to create unique visual voices despite the prevalence of stock photography and the monolithic hold of Adobe Creative Suite. Simultaneously, as noted by design and media critic Lev Manovich, specific techniques, artistic languages, and vocabularies previously isolated within individual professions are being imported and exported across software applications and professions.

This new common language of hybridity and “remixability,” through which most visual artists now work, is unlike anything seen before. Technology has irreversibly changed our sense of aesthetics, giving us both more power and less.

WE SHOULD ENCOURAGE COLLABORATION AND COMMUNAL EXPERIENCE

What’s the good of multi-touch technology if we don’t want to sit down together? Collaboration and community fuel world-changing design solutions. Despite our connections online, many people are experiencing a growing sense of personal isolation. How can we, as designers, combat that isolation with projects that foster community? Media activist Kalle Lasn has warned designers: “We have lost our plot. Our story line. We have lost our soul.” Producing work that fosters real connections may be one way of getting that soul back.

WE ALL WRITE MORE TODAY THAN WE DID 15 YEARS AGO

Blogs, emails, Twitter—we communicate with many more people through text than through speech. If grammar imparts

order and structure to our thoughts, then this increase in writing brings value to our society and our discipline. Design authorship, an issue debated by influential figures like Michael Rock, Ellen Lupton and Jessica Helfand over the course of the last decade, foregrounded the active relationship between text and image and between a discipline and its discourse. The expansion of written communication makes possible thoughtful contributions to the larger discourse of design by a wider slice of the graphic design population.

THE CENTRAL METAPHOR OF OUR CURRENT SOCIETY IS THE NETWORK

Even if we don't all understand the computer codes that run the back end of our digital age, we can comprehend the networked structure of our day and design to meet it. Avant-garde artists at the beginning of the last century, including F. T. Marinetti, László Moholy-Nagy and Aleksandr Rodchenko, were adept at activating their own networks: newspapers, magazines, lectures and written correspondence. Recently, I heard lectures by Emily Pilloton of Project H and Cameron Sinclair of Architecture for Humanity, two young designers who are creating opportunities, locally and around the world, for designers to improve basic human living conditions. The connectivity of the web is critical to their success. Efficient networks for spreading change and prosperity are already in place. We just have to grasp them. Designers in the early 20th century rose to the challenges of their societies. We too can take on the complexities of our time, the rising millennium. Delving into our theoretical base equips us to address critical material problems in the world and our discipline.

COLOR THEORY

Color theory tutorials cover the aesthetics as well as the visual impact of color combinations in web design. Learn to use colors based on their emotional message. Smart use of appropriate colors can make a site not only look professional, but also be easier to navigate and apprehend. Not to mention, color scheme of a site defines the first impression and has a great impact on a leave/stay factor.

BLOW UP YOUR WEBSITE CONVERSIONS USING COLOR

Human beings make decisions based on their senses all the time, especially when it comes to online purchases. Of those senses, sight is the biggest decision maker for most. For that reason, the colors you choose for your website design can have the most impact on turning your website visitors into customers.

COLOR PSYCHOLOGY

In order to gain an understanding of the colors you should use for your website, you need an understanding of how color affects human behavior.

Though some are skeptic of the actual effects of colors on human behavior, study after study has shown that colors play a big part in daily decision making.

A study by Satyendra Singh (Department of Administrative Studies, University of Winnipeg, Winnipeg, Canada) published in the Journal of Management History revealed that it only takes 90 seconds for a customer to make a decision regarding a product, and, on average,

75 percent of the time, that decision is based on the color of the product alone. With this information in mind, it's clear that colors have a significant influence on customer's purchasing decisions. A large part of understanding color psychology is knowing what colors mean. Listed below are some of the most common color.

BLACK

Authority and Power. Popular in the fashion realm because it gives the illusion of thin.

WHITE

Innocence, purity, and sterility. Light, neutral, and provides excellent contrast with darker colors.

RED

Love, passion, rage. Can induce a faster heart rate and breathing, and is a popular color for the food industry.

BLUE

Peace, trust. Puts people at ease and makes them more trusting of a company.

GREEN

Nature, tranquility. Catches attention when used in the right shades and is perfect for environmentally friendly messages.

YELLOW

Optimism, overstimulation. It catches attention, but can be unpleasantly overwhelming when used excessively.

ORANGE

Confidence, impulsiveness. Gives the feeling of haste or impulse, which is perfect for retail websites.

PURPLE

Luxury, wealth, sophistication. Connotes femininity and romance, though it can appear artificial if paired with the wrong colors.

BROWN

Solid, reliable, traditional. Creates an atmosphere of trust and genuineness. Knowing the meaning of each of these colors and using them correctly in your website design can be the most important step you take to gain customers and retain their loyalty.

USING COLORS APPROPRIATELY

Color helps catch a website visitor's attention, which is integral for driving conversions. In general, colors that stand out against a complementary background are more likely to be remembered than those that blend in. So if the purpose of your webpage is to entice users to try your product, the button proclaiming "Try it for free," should stand out against the other colors on your webpage.

Overall, remember that converting isn't always about what looks good. It's about keeping with the overall theme of your website while making important parts stand out. Keeping with the theme doesn't always mean that the colors will match. It often means that the background colors will be conducive to the purpose of your products, and the CTAs

will be surrounded in bright, contrasting colors meant to stand out.

Just be cautious with your color use. Too many colors on one web page will only overwhelm the user, hurting your conversions. If you have too many colors on one page, no particular part will stand out, and the call to action won't be clear.

COLORS FOR GENDERS

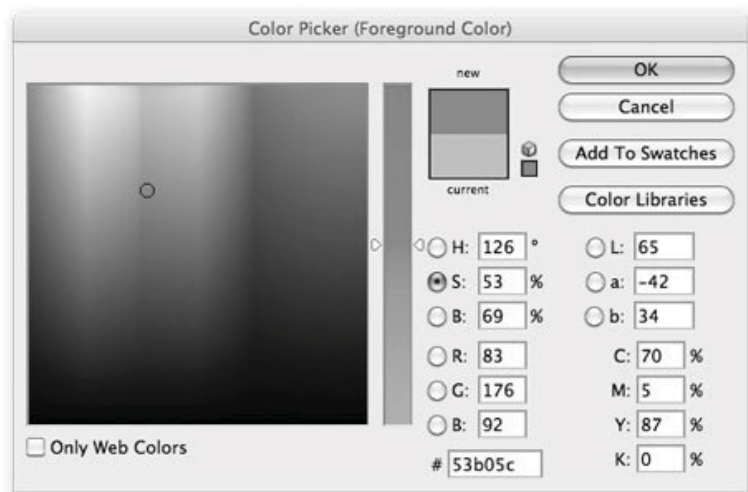
Different colors usually mean different things for each gender, so if your website is gender specific, this tip is for you. Women tend to gravitate towards blue, purple, and green and stray from gray, orange and brown. Men generally prefer blue, green, and black, but avoid purple, orange, and brown. With this information in mind, it's probably safe to assume that if your site is gender neutral, blues, greens, blacks, and whites will go over well with your customers.

CTA PRIMARY ACTIONS

When it comes to your calls to action, primary colors tend to go over best. Bright reds, yellows, and blues feel very familiar to customers and help to cultivate a relationship of trust between you and your company. They're also very effective at catching attention, and when paired against a neutral backdrop, it creates an extremely compelling call to action that most customers won't be able to resist. Overall, when it comes to enticing website visitors to become customers, never underestimate the power of color. It's one of the biggest and most important secrets for efficient web design.

HOW TO USE COLOR TO ENHANCE YOUR DESIGNS

People are physically, psychologically, and socially influenced by color. Color has been found to have connections to health and it can help set the mood through which your designs are seen. Color communicates meaning and so we need to be conscious of what meaning we're conveying when we choose to use one color over another. It's not enough for a designer to use a color simply because he or she likes that color.



Color is a tool in the designer's toolbox much the same as a grid or *whitespace* and it's important to understand how to use that tool.

Last week we talked about the *color theory* and how we could represent color and choose different color schemes. This week we'll take a look at the meaning colors communicate, how we can better control our designs through our color choices and finally how to go about choosing a color scheme that reinforces the message your design aims to communicate.

COLOR MEANING

The first and perhaps most important thing to understand about the meaning of color is that there is no substantive evidence that support a universal system of color meaning. It's not that colors themselves have specific meaning, but rather that we have culturally assigned meanings to them.

While some color symbolism exists globally (red as the color of a stop sign, yellow for caution), color symbolism tends to be more common within a given culture than across different cultures (white is used for weddings in Western cultures and for funerals in Eastern cultures).

Even within a single culture individual differences will exist. You and we will not necessarily be affected in the same way by seeing the same color.

The above means that it's important to understand who your target audience is and how your audience attaches meaning to color. Again it's not that a color has a specific meaning on its own. It's that we've culturally assigned meaning to colors. Keep that in mind as you read some of the specifics about the colors mentioned below.



Warm Colors: For the sake of simplicity let's define warm colors as red, orange and yellow. These are the colors of fire. They radiate warmth. Warm colors are more often associated with passion, energy, impulsiveness, happiness, coziness, and comfort. They draw attention and have the advantage of being inviting and harmonious.



Cool Colors: Again for the sake of simplicity let's define cool colors as green, blue, and violet. These are the colors of water. Cool colors are more often associated with calm, trust, and professionalism. They are also associated with sadness and melancholy. They have the advantage of being professional and harmonious, but can also turn people off by the coolness they radiate.

Note: The demarcation point between warm and cool colors falls somewhere between yellow/green and violet red. Green and purple don't fall neatly into either warm or cool camps. They tend to take on the properties of one or the other based on the surrounding context.



Red: is the color of fire and blood. It's emotionally intense. Red is associated with energy, war, danger, strength, power, determination, action, confidence, courage, vitality, passion, desire, and love. It can enhance metabolism, increase respiration, and raise blood pressure. Red has a high visibility and advances to the foreground. It is often used for buttons in order to get people to take impulsive action.

Yellow: is the color of the sun. Bright yellow attracts attention, though it can also be distracting when overused. Yellow is associated with joy, happiness, wisdom, and intellectual energy. It stimulates mental activity and

generates muscle energy. Yellow produces a warming effect, arouses cheerfulness and is often used to evoke pleasant feelings. Shades of yellow can become dingy lessening the pleasing effect.



Blue: is the color of the sky and the sea. It has the opposite effect of red and slows metabolism, breathing, and heart rate. It's seen as a masculine color. Blue is associated with trust, loyalty, wisdom, intelligence, expertise, confidence, stability and depth. It creates a calming effect, suppresses appetite and has been considered to be beneficial to both body and mind. Blue is often used for corporate sites given the previously mentioned associations.



Orange: combines the energy or red with the happiness of yellow. It's not as aggressive as red and calls to mind healthy food (citrus).. Orange is associated with joy, sunshine, the tropics, enthusiasm, happiness, fascination, creativity, determination, attraction, success, encouragement, stimulation, and strength. It can increase appetite and evokes thoughts of fall and harvest.

3-D Computer Graphics



Green: is the color of nature. It symbolizes growth, hope, freshness, and fertility. In countries with green money such as the U.S. it evokes thoughts and feelings of financial wealth. Green is associated with healing, stability, endurance, harmony, safety, life, and well being. It can sometimes signify a lack of experience and is often used to indicate the safety of drugs and medical products in advertising.



Purple: combines the stability of blue and the energy of red. It conveys wealth and extravagance and is seen as the color of royalty. It symbolizes power, nobility, luxury, and ambition. Purple is associated with wisdom, dignity, independence, creativity, mystery, and magic. Light purple is seen as feminine and purple is a popular color with children. Purple occurs less frequently in nature and some may consider it artificial. In Catholic cultures it is representative of death and in some Islamic nations it is associated with prostitution.

White: is associated with light, goodness, innocence, purity, virginity. It usually has positive connotations and is seen as clean and safe.

Black: is associated with power, elegance, formality, death, evil, and mystery. It denotes strength and authority,

is seen as formal and elegant, and brings forth feelings of fear and the unknown.



Gray: is the color of sorrow, detachment, and isolation. It connotes responsibility and conservative practicality. It's a neutral color and creates a non-invasive feeling. It's associated with security, maturity, and dependability. It can be used to reduce the intense energy of another color and to emphasize a willingness to comply. Some people who prefer gray may be seen as the lone wolf type or narrow-minded.



Brown: is the color of the earth and tends to blend into the background. It's associated with material things, order, and convention. It's connection to the earth gives it stability. Brown can convey a solid and wholesome feeling.

The following articles offer additional information about the possible meaning some will associate with a given color.

The Visual Effect of Color in Your Designs

Using colors that don't work well together, using too many colors, or even not enough could drive people away before they have a chance to absorb your content. Ideally you should plan and choose a color scheme from the start and you should be consistent in how you use color in your

design. Color is used to attract attention, *group related elements*, convey meaning, and generally enhance the aesthetics of your site. It can be used to organize your elements and create a *visual hierarchy in your design*.

A small dose of color that *contrasts* with your main color will draw attention. It will give emphasis. Repeating colors on elements like page headings gives an immediate visual cue that those headings are related.

Warmer colors advance into the foreground while cooler colors recede into the background. Your choice in warm and cool colors can affect the *figure/ground relationship* of your elements. Since cooler colors recede you may decide to use them for background elements and because warmed colors advance they make a good choice for elements in the foreground.



By mixing warm and cool colors you can create depth in your design. Consider Van Gogh's *Cornfield with Cypress* above. Color is not the only way Van Gogh gives depth to the painting, but notice how the colors add to the depth.

The mountains, sky and clouds use cooler colors, while the cornfield uses warmer colors.

You might choose a warmer color for the type on top on an image to ensure it's seen. Naturally it depends on the colors in the image as it will be more important to make sure the text color contrasts.

Darker colors tend to be seen first and carry more *visual weight*. A larger area of a lighter color is necessary to balance the visual weight.

Highly saturated colors (pure hues) are perceived as more dynamic. They attract attention. Too many saturated colors can compete and cause eye fatigue. Desaturated colors lend themselves to performance and efficiency. They might be a better choice to help people complete a specific task. Desaturated/Bright colors are perceived as friendly and professional. Desaturated/Dark colors are seen as serious and professional.

Be careful about using too many colors. You want to limit colors in the same way you limit fonts. You need enough to be able to offer contrast, but not too much to lack similarity. 5 colors is generally a good maximum, though you can use more. The more colors you use the harder it will be to use them effectively.

CLASSICAL GRAPHIC DESIGN THEORY

Here is a short introduction to graphic design theory, explaining the different aspects of design which are considered when composing a piece of fine art or producing a graphic layout in commercial art. I call it classic theory because it forms the basis for many decisions in design.

ELEMENTS OF DESIGN

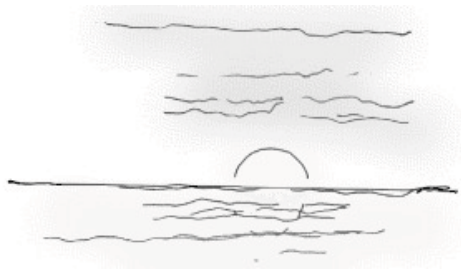
LINE DESIGN

A line is a form with width and length, but no depth. Artists use lines to create edges, the outlines of objects. A line is created by the movement of the artist's pen.

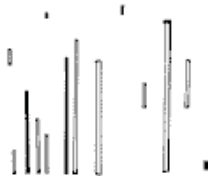
LINE DIRECTION

The direction of a line can convey mood.

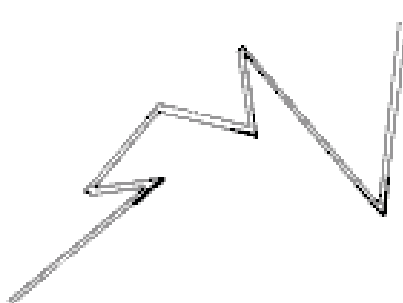
Horizontal lines are calm and quiet,



vertical lines suggest more of a potential for movement,



while diagonal lines strongly suggest movement and give more of a feeling of vitality to a picture.



CONTOUR AND GESTURE



Lines used to follow the edges of forms are called contour drawings



Drawings which seem to depict more movement than actual outline are called gesture drawings.

LINE AS VALUE

Lines or crosshatching can also be used to create areas of grey inside a drawing. These areas of darker shading inside a figure, called areas of value, can give a more three-dimensional feeling to an object.

SHAPE DESIGN

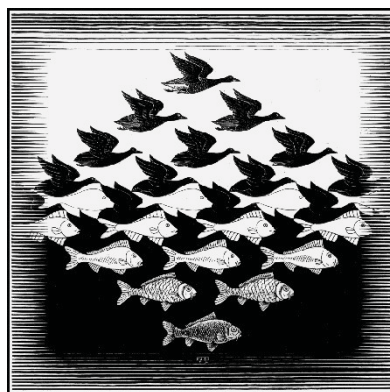
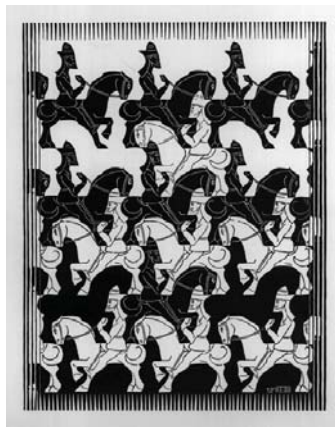
A shape is an enclosed object. Shapes can be created by line, or by color and value changes which define their edges.

Volume and Mass: Shape is considered to be a two-dimensional element, while three-dimensional elements have

volume or mass. Therefore, a painting has shapes, while a sculpture has volume and mass.

POSITIVE/NEGATIVE SHAPES

In a picture, the shapes that the artist has placed are considered the positive shapes. The spaces around the shapes are the negative spaces. It is just as important to consider the negative space in a picture as the positive shapes. Sometimes artists create pieces that have no distinction between positive and negative spaces. M. C. Escher was a master at creating drawings where there was no distinction between positive and negative space. Here are two examples of Escher's work which show the interplay between positive and negative space:



TEXTURE DESIGN

Texture is the surface quality of an object. We experience texture when we touch objects and feel their roughness, smoothness or patterns. Texture is the artist's way of mapping these tactile impressions on to the two-dimensional picture. Texture is created by varying the pattern of light and dark areas on an object.

Value : Value refers to the relative lightness or darkness of a certain area. Value can be used for emphasis. Variations in value are used to create a focal point for the design of a picture. A light figure on a dark background will be immediately recognized as the center of attention, similarly for a dark figure on a mostly white background. Gradations of value are also used to create the illusion of depth. Areas of light and dark can give a three-dimensional impression, such as when shading areas of a person's face.

DRAWING BY MARGUERITE SMITH, SASKATOON

COLOR

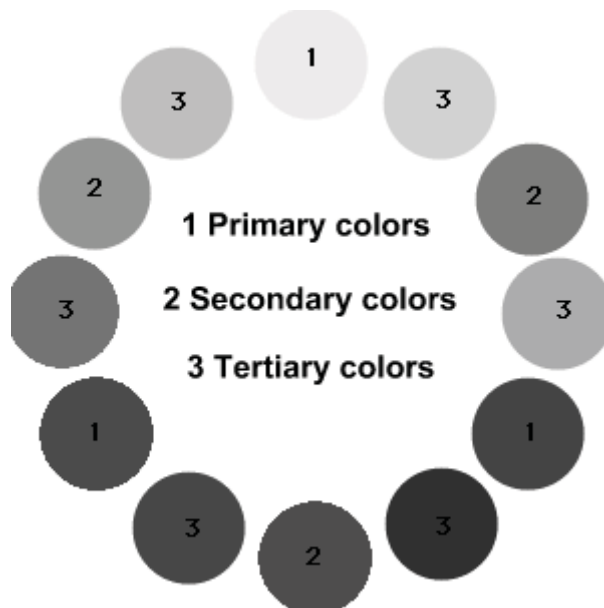
Color occurs when light in different wavelengths strikes our eyes. Objects have no color of their own, only the ability to reflect a certain wavelength of light back to our eyes. As you know, color can vary in differing circumstances. For example, grass can appear gray in the morning or evening or bright green at noon. Colors appear different depending on whether you view them under incandescent, florescent or natural sunlight. Colors also change according to their

surroundings. You can see this by looking at the color squares below - the reddish outline box is the same color in all the examples.

PROPERTIES OF COLOR

HUE

Hue refers to the color itself. Each different hue is a different reflected wavelength of light. White light broken in a prism has seven hues: red, orange, yellow, green, blue, indigo and violet. White light occurs when all the wavelengths are reflected back to your eye, and black light occurs when no light is reflected to your eye. This is the physics of light.



When it comes to using color in art, things get quite messy. Looking at the color wheel above, when using color pigments, the three primary colors used are yellow, blue and red. These three colors are blended together to produce other colors, called secondary colors, such as green, orange

and purple. Mix enough colors together, and you get black. Pretty strange, eh?

COMPUTER COLORS



Computer colors are produced by combining the three colors of red, green and blue together. Believe it or not, you can get yellow by combining these colors.

PRINTER COLORS



Things get even dicier on computers when you go to print out these colors. Printing uses the CYMK convention which takes cyan (light blue), yellow, magenta (pinky red) and black inks and tries to recreate the color that your computer created with red, green and blue light.

COLOR VALUE

Color value refers to the lightness or darkness of the hue. Adding white to a hue produces a high-value color, often called a tint. Adding black to a hue produces a low-value color, often called a shade.

INTENSITY

Intensity, also called chroma or saturation, refers to the brightness of a color. A color is at full intensity when not mixed with black or white - a pure hue. You can change the intensity of a color, making it duller or more neutral by adding gray to the color. You can also change the intensity of a color by adding its complement (this is the color found directly opposite on the traditional color wheel). When changing colors this way, the color produced is called a tone.

When you mix complementary colors together, you produce a dull tone. However, when you put complementary colors side by side, you increase their intensity. This effect is called simultaneous contrast - each color simultaneously intensifies the visual brightness of the other color.

Below are some examples of how this works, using a program called Metacreations painter. As you can see, you choose a hue from the outer ring. Inside the triangle, you can vary the saturation of the hue (amount of color), the tint or the shade.

OPTICAL COLOR MIXING

When small dots of color are placed adjacent to each other, your eye will combine the colors into a blended color. This is the principle used when printing color in magazines. Dots of cyan, magenta, yellow and black are distributed in a pattern on the paper, and depending on the quantity of a certain dot, you will see a specific color on the page. Paul Signac used a technique called pointillism that involved creating art using the combination of dots to form images.

COLOR AND SPACE

Certain colors have an advancing or receding quality, based on how our eye has to adjust to see them. Warm colors such as red, orange or yellow seem to come forward while cool colors such as blue and green seem to recede slightly. In the atmosphere, distant objects appear bluish and the further away an object appears, the less colorful and distinct it becomes. Artists use this to give an illusion of depth, by using more neutral and grayish colors in the background.

COLOR SCHEMES

MONOCHROMATIC

This color scheme involves the use of only one hue. The hue can vary in value, and black or white may be added to create various shades or tints.

ANALOGOUS

This color scheme involves the use of colors that are located adjacent on the color wheel. The hues may vary in value. The color scheme for this site is analogous, with the colors varying only slightly from each other.

COMPLEMENTARY

This color scheme involves the use of colors that are located opposite on the color wheel such as red and green, yellow and purple, or orange and blue. Complementary colors produce a very exciting, dynamic pattern.

TRIADIC

This color scheme involves the use of colors that are equally spaced on the color wheel. The primary colors of yellow, red and green could be used together in a color scheme to produce a lively result.

Check out Color Picker web software. This application will allow you to choose a color and then display its complementary or triadic match. Hint: read the instructions first, then click on the link which says “Open Color Picker 2”. Color Picker 2.

COLOR DISCORD

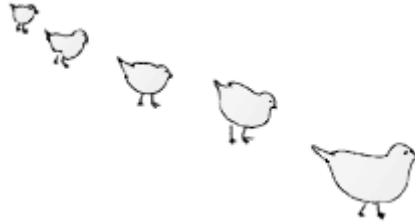
While monochromatic, analogous, complementary or triadic color schemes are considered to be harmonious, there are some color schemes considered dissonant. Discordant colors are visually disturbing - we say they clash. Colors that are widely separated on the color wheel (but not complementary or triadic) are considered to be discordant. Discordant colors can be eye-catching and are often used for attention-getting devices in advertising.

ILLUSION OF SPACE AND DEPTH

We live in a three-dimensional world of depth. When we look around us, some things seem closer, some further away. The artist can also show the illusion of depth by using the following means:

- Size & Vertical Location
- Overlapping
- Detail (Aerial or Atmospheric Perspective)
- Linear Perspective.

SIZE & VERTICAL LOCATION



Since objects in our environment look smaller when they are farther away, the easiest way to show depth is to vary the size of objects, with closer objects being larger and more distant objects being smaller. As well, we perceive objects that are higher on the page and smaller as being further away than objects which are in the forefront of a picture.

OVERLAPPING



When objects are partially obscured by other objects in front of them, we perceive them as further back than the covering objects.

We do not see them as incomplete forms, just further back.

DETAIL (AERIAL OR ATMOSPHERIC PERSPECTIVE)

Atmospheric perspective uses color and value contrasts to show depth. Objects which are further away generally

have less distinct contrast - they may fade into the background or become indistinct dark areas. The foreground objects will be clear with sharper contrast. Here is a link to Leonardo da Vinci's use of aerial perspective: [Investigating aerial perspective](#).



LINEAR PERSPECTIVE (CONVERGING LINES)



Linear perspective is based on the idea that all lines will converge on a common point on the horizon called the vanishing point. You have observed linear perspective when you notice that the lines on the highway appear to meet at a point in the distance.

Artists use linear perspective to create a focal point for a picture. Any walls, ceilings, floors or other objects with lines will appear to come together at the horizon line.

These lines converging lead our eyes towards that point. Often, the most important object or person in the picture will be located at that point. You can see in the drawing above how all the lines in the drawing seem to lead your eye toward the church in the center back of the drawing.

Other types of perspective, such as two-point or multipoint perspective are also used. Two-point perspective, which occurs when you display a building from a corner view, as opposed to a front view, is often used by architects to show a more three-dimensional view of a building.

4

3 Dimensional Transformation

BASIC TRANSFORMATION

Animation are produced by moving the 'camera' or the objects in a scene along animation paths. Changes in orientation, size and shape are accomplished with geometric transformations that alter the coordinate descriptions of the objects. The basic geometric transformations are translation, rotation, and scaling. Other transformations that are often applied to objects include reflection and shear.

USE OF TRANSFORMATIONS IN CAD

In mathematics, "Transformation" is the elementary term used for a variety of operation such as rotation, translation, scaling, reflection, shearing *etc.* CAD is used throughout the engineering process from conceptual design and layout, through detailed engineering and analysis of components to

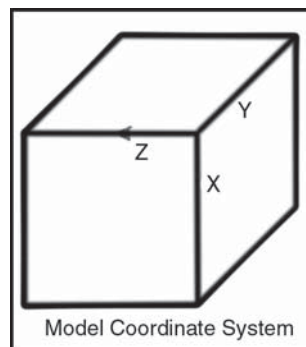
definition of manufacturing methods. Every aspect of modelling in CAD is dependent on the transformation to view model from different directions we need to perform rotation operation. To move an object to a different location translation operation is done. Similarly Scaling operation is done to resize the object.

COORDINATE SYSTEMS

In CAD three types of coordinate systems are needed in order to input, store and display model geometry and graphics. These are the Model Coordinate System (MCS), the World Coordinate System (WCS) and the Screen Coordinate System (SCS).

MODEL COORDINATE SYSTEM

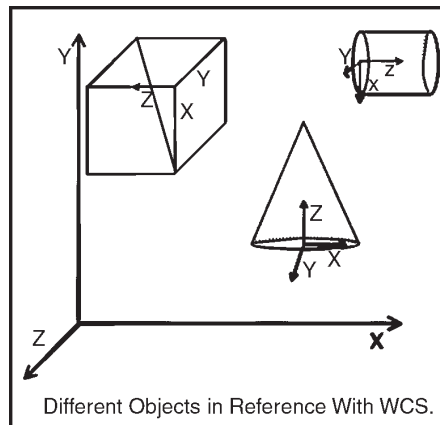
The MCS is defined as the reference space of the model with respect to which all the model geometrical data is stored. The origin of MCS can be arbitrary chosen by the user.



WORLD COORDINATE SYSTEM

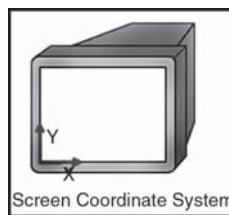
Every object have its own MCS relative to which its geometrical data is stored. Incase of multiple objects in the same working space then there is need of a World Coordinate System which relates each MCS to each other

with respect to the orientation of the WCS. It can be seen by the picture shown below.



SCREEN COORDINATE SYSTEM

In contrast to the MCS and WCS the Screen Coordinate System is defined as a two dimensional device-dependent coordinate system whose origin is usually located at the lower left corner of the graphics display as shown in the picture below. A transformation operation from MCS coordinates to SCS coordinates is performed by the software before displaying the model views and graphics.



VIEWING TRANSFORMATIONS

The objects are modelled in WCS, before these object descriptions can be projected to the view plane, they must be transferred to viewing coordinate system. The view plane or the projection plane, is set up perpendicular to the viewing

z_v axis. The World coordinate positions in the scene are transformed to viewing coordinates, then viewing coordinates are projected onto the view plane.

The transformation sequence to align WCS with Viewing Coordinate System is.

- Translate the view reference point to the origin of the world coordinate system.
- Apply rotations to align x_v , y_v , and z_v with the world x_w , y_w and z_w axes, respectively.

HIDDEN LINES AND SURFACES

In displaying objects, there frequently occurs overlapping, meaning that the objects which are closer are visible, and those behind them are non-visible, because they appear as hidden behind them. There have been designed many, rather complex algorithms for solving this issue. Their goal is to find out objects whose parts are visible only from the observer place. Sometimes it is necessary to draw also edges of overlapped objects, most often by dash lines. The visibility algorithms always depend on the fact of how the objects are represented in an area. The optimal variant is to have objects described using their limits, best by surfaces. Most of these algorithms too are prepared for this sort of representation of objects.

VISIBILITY ALGORITHM CATEGORIZATION

Sutherland, Sproull and Schumacker termed the visibility algorithms according to their approaches and principles used. They are either image area or object area types. According to, the visibility algorithms can be split depending on the form of output data.

- Line algorithm (*Hidden line eliminator*) On the output we receive a set of abscissas representing visible edges. The advantage of this solution is the fact that we have a set of vectors, and we are not limited due to issues of resolution. In this case, we can scale the scene without any loss in quality.
- Raster algorithm (*Hidden surface eliminator*) On the output, we receive a set of pixels representing the specific points of the scene in the fixed given resolution, without any option to scale the scene. Here it is, however, possible to draw also surfaces with shading and lighting.

PRE-PROCESSING OF DATA

Because it is time demanding to calculate all algorithms, at the beginning it is necessary to analyse the object in the scene. This is called data pre-processing. First, all objects outside the visual angle, are excluded and all polygons of which the scalar vector product (of the normal vector of the polygon and the direction of view) is positive, and the given polygon is then reversed

RENDERING AND ILLUMINATION

One of the most important aspects of computer graphics is simulating the illumination in a scene. Computer-generated images are two-dimensional arrays of computed pixel values, with each pixel coordinate having three numbers indicating the amount of red, green, and blue light coming from the corresponding direction in the scene. Figuring out what these numbers should be for a scene is not trivial, because each pixel's colour is based both on how the object

at that pixel reflects light and the light that illuminates it. Furthermore, the illumination comes not only directly from the light sources in the scene, but also indirectly from all of the surrounding surfaces in the form of “bounce” light. The complexities of the behaviour of light — one reason the world around us appears rich and interesting — make generating “photoreal” images both conceptually and computationally complex. As a simple example, suppose we stand at the back of a square white room in which the left wall is painted red, the right wall is painted blue, and the light comes from the ceiling. If we take a digital picture of this room and examine its pixel values, we will indeed find that the red wall is red and the blue wall is blue. But when we look closely at the white wall in front of us, we will notice that it isn’t perfectly white. Towards the right it becomes bluish, and towards the left it becomes pink. The reason for this is indirect illumination: towards the right, blue light from the blue wall adds to the illumination on the back wall, and towards the left, red light does the same.

Indirect illumination is responsible for more than the somewhat subtle effect of white surfaces picking up the colours of nearby objects — it is often responsible for most, sometimes all, of the illumination on an object or in a scene. If I sit in a white room illuminated by a small skylight in the morning, the indirect light from the patch of sunlight on the wall lights the rest of the room, not the direct light from the sun itself. If light did not bounce between surfaces, the room would be nearly dark!

In early computer graphics, interreflections of light between surfaces in a scene were poorly modelled. Light falling on

each surface was computed solely as a function of the light coming directly from light sources, with perhaps a roughly determined amount of “ambient” light added irrespective of the actual colours of light in the scene. The groundbreaking publication showing that indirect illumination could be modelled and computed accurately was presented at SIGGRAPH 84, when Goral *et al.* of Cornell University described how they had simulated the appearance of the red, white, and blue room example using a technique known as radiosity.

Inspired by physics techniques for simulating heat transfer, the Cornell researchers first divided each wall of the box into a 7×7 grid of patches; for each patch, they determined the degree of its visibility to every other patch, noting that patches reflect less light if they are farther apart or facing away from each other. The final light colour of each patch could then be written as its inherent surface colour times the sum of the light coming from every other patch in the scene. Despite the fact that the illumination arriving at each patch depends on the illumination arriving (and thus leaving) every other patch, the radiosity equation could be solved in a straightforward way as a linear system of equations.

The result that Goral *et al.* obtained correctly modelled that the white wall would subtly pick up the red and blue of the neighbouring surfaces. Soon after this experiment, when Cornell researchers constructed such a box with real wood and paint, they found that photographs of the box matched their simulations so closely that people could not tell the difference under controlled conditions. The first “photoreal” image had been rendered!

One limitation of this work was that the time required to solve the linear system increased with the cube of the number of patches in the scene, making the technique difficult to use for complex models (especially in 1984). Another limitation was that all of the surfaces in the radiosity model were assumed to be painted in matte colours, with no shine or gloss.

A subsequent watershed work in the field was presented at SIGGRAPH 86, by Jim Kajiya from Caltech, who published the “The Rendering Equation,” which generalized the ideas of light transport to any kind of geometry and any sort of surface reflectance. The titular equation of Kajiya’s paper stated in general terms that the light leaving a surface in each direction is a function of the light arriving from all directions upon the surface, convolved by a function that describes how the surface reflects light. The latter function, called the bidirectional reflectance distribution function (BRDF), is constant for diffuse surfaces but varies according to the incoming and outgoing directions of light for surfaces with shine and gloss.

Kajiya described a process for rendering images according to this equation using a randomized numerical technique known as path tracing. Like the earlier fundamental technique of ray tracing, path tracing generates images by tracing rays from a camera to surfaces in the scene, then tracing rays out from these surfaces to determine the incidental illumination on the surfaces. In path tracing, rays are traced not only in the direction of light sources, but also randomly in all directions to account for indirect light from the rest of the scene. This simple scene shows realistic light

interactions among both diffuse and glossy surfaces, as well as other complex effects, such as light refracting through translucent objects. Although still computationally intensive, Kajiya's randomized process for estimating solutions to the rendering equation made the problem tractable both conceptually and computationally.

BRINGING REALITY INTO THE COMPUTER

Using the breakthroughs in rendering techniques developed in the mid-1980s, it was no simple endeavour to produce synthetic images with the full richness and realism of images in the real world. Photographs appear "real" because shapes in the real world are typically distinctive and detailed, and surfaces in the real world reflect light in interesting ways, with different characteristics that vary across surfaces. And also, very importantly, light in the real world is interesting because typically there are different colours and intensities of light coming from every direction, which dramatically and subtly shape the appearance of the forms in a scene. Computer-generated scenes, when constructed from simple shapes, textured with ideal plastic and metallic reflectance properties, and illuminated by simple point and area light sources, lack "realism" no matter how accurate or computationally intensive the lighting simulation. As a result, creating photoreal images was still a matter of skilled artistry rather than advanced technology. Digital artists had to adjust the appearance of scenes manually.

Realistic geometry in computer-generated scenes was considerably advanced in the mid-1980s when 3D digitizing techniques became available for scanning the shapes of real-

world objects into computer-graphics Programmes. The Cyberware 3D scanner, an important part of this evolution, transforms objects and human faces into 3D polygon models in a matter of seconds by moving a stripe of laser light across them. An early use of this scanner in a motion picture was in Star Trek IV for an abstract time-travel sequence showing a collage of 3D models of the main characters' heads. 3D digitization techniques were also used to capture artists' models of extinct creatures to build the impressive digital dinosaurs for Jurassic Park.

DIGITIZING AND RENDERING WITH REAL-WORLD ILLUMINATION

Realism in computer graphics advanced again with techniques that can capture illumination from the real world and use it to create lighting in computer-generated scenes. If we consider a particular place in a scene, the light at that place can be described as the set of all colours and intensities of light coming towards it from every direction. As it turns out, there is a relatively straightforward way to capture this function for a real-world location by taking an image of a mirrored sphere, which reflects light coming from the whole environment towards the camera. Other techniques for capturing omnidirectional images include fisheye lenses, tiled panoramas, and scanning panoramic cameras.

The first and simplest form of lighting from images taken from a mirrored sphere is known as environment mapping. In this technique, the image is directly warped and applied to the surface of the synthetic object. The technique using images of a real scene was used independently by Gene Miller

and Mike Chou and Williams. Soon after, the technique was used to simulate reflections on the silvery, computer-generated spaceship in the 1986 film *Flight of the Navigator* and, most famously, on the metallic T1000 “terminator” character in the 1991 film *Terminator 2*. In all of these examples, the technique not only produced realistic reflections on the computer-graphics object, but also made the object appear to have truly been in the background environment. This was an important advance for realism in visual effects. Computer-graphics objects now appeared to be illuminated by the light of the environment they were in.

Environment mapping produced convincing results for shiny objects, but innovations were necessary to extend the technique to more common computer-graphics models, such as creatures, digital humans, and cityscapes. One limitation of environment mapping is that it cannot reproduce the effects of object surfaces shadowing themselves or of light reflecting between surfaces. The reason for this limitation is that the lighting environment is applied directly to the object surface according to its surface orientation, regardless of the degree of visibility of each surface in the environment. For surface points on the convex hull of an object, correct answers can be obtained. However, for more typical points on an object, appearance depends both on which directions of the environment they are visible to and light received from other points on the object.

A second limitation of the traditional environment mapping process is that a single digital or digitized photograph of an environment rarely captures the full range of light visible in a scene. In a typical scene, directly visible light sources are

usually hundreds or thousands of times brighter than indirect illumination from the rest of the scene, and both types of illumination must be captured to represent the lighting accurately. This wide dynamic range typically exceeds the dynamic range of both digital and film cameras, which are designed to capture a range of brightness values of just a few hundred to one. As a result, light sources typically become “clipped” at the saturation point of the image sensor, leaving no record of their true colour or intensity. This is not a major problem for shiny metal surfaces, because shiny reflections would become clipped anyway in the final rendered images. However, when lighting more typical surfaces — surfaces that blur the incidental light before reflecting it back towards the camera — the effect of incorrectly capturing the intensity of direct light sources in a scene can be significant.

We developed a technique to capture the full dynamic range of light in a scene, up to and including direct light sources. Photographs are taken using a series of varying exposure settings on the camera; brightly exposed images record indirect light from the surfaces in the scene, and dimly exposed images record the direct illumination from the light sources without clipping. Using techniques to derive the response curve of the imaging system (*i.e.* how recorded pixel values correspond to levels of scene brightness), we assemble this series of limited-dynamic-range images into a single high-dynamic image representing the full range of illumination for every point in the scene. Using IEEE floating-point numbers for the pixel values of these high-dynamic-range images (called HDR images or HDRIs), ranges exceeding even one to a million can be captured and stored.

The following year we presented an approach to illuminating synthetic objects with measurements of real-world illumination known as image-based lighting (IBL), which addresses the remaining limitations of environment mapping. The first step in IBL is to map the image onto the inside of a surface, such as an infinite sphere, surrounding the object, rather than mapping the image directly onto the surface of the object.

We then use a global illumination system to simulate this image of incidental illumination actually lighting the surface of the object. In this way, the global illumination algorithm traces rays from each object point out into the scene to determine what is lighting it.

Some of the rays have a free path away from the object and thus strike the environmental lighting surface. In this way, the illumination from each visible part of the environment can be accounted for. Other rays strike other parts of the object, blocking the light it would have received from the environment in that direction.

If the system computes additional ray bounces, the colour of the object at the occluding surface point is computed in a similar way; otherwise, the algorithm approximates the light arriving from this direction as zero. The algorithm sums up all of the light arriving directly and indirectly from the environment at each surface point and uses this sum as the point's illumination. The elegance of this approach is that it produces all of the effects of the real object's appearance illuminated by the light of the environment, including self-shadowing, and it can be applied to any material, from metal to plastic to glass.

The techniques of HDRI and IBL, and the techniques and systems derived from them, are now widely used in the visual-effects industry and have provided visual-effects artists with new lighting and compositing tools that give digital actors, airplanes, cars, and creatures the appearance of actually being present during filming, rather than added later via computer graphics. Examples of elements illuminated in this way include the transforming mutants in *X-Men* and *X-Men 2*, virtual cars and stunt actors in *The Matrix Reloaded*, and whole cityscapes in *The Time Machine*. In latest computer animation, we extended the techniques to capture the full range of light of an outdoor illumination environment — from the pre-dawn sky to a direct view of the sun — to illuminate a virtual 3D model of the Parthenon on the Athenian Acropolis.

APPLYING IMAGE-BASED LIGHTING TO ACTOR

In a laboratory's most recent work, we have examined the problem of illuminating real objects and people, rather than computer-graphics models, with light captured from real-world environments. To accomplish this we use a series of light stages to measure directly how an object transforms incidental environmental illumination into reflected radiance, a data set we call the reflectance field of an object.

The first version of the light stage consisted of a spotlight attached to a two-bar rotation mechanism that rotated the light in a spherical spiral about a person's face in approximately one minute. At the same time, one or more digital video cameras recorded the object's appearance under every form of directional illumination.

From this set of data, we could render the object under any form of complex illumination by computing linear combinations of the colour channels of the acquired images. The illumination could be chosen to be measurements of illumination in the real world or the illumination present in a virtual environment, allowing the image of a real person to be photorealistically composited into a scene with the correct illumination.

An advantage of this photometric approach for capturing and rendering objects is that the object need not have well defined surfaces or easy-to-model reflectance properties. The object can have arbitrary self-shadowing, interreflection, translucency, and fine geometric detail. This is helpful for modelling and rendering human faces, which exhibit all of these properties, as do many objects we encounter in our everyday lives.

Recently, a group constructed two additional light stages. Light Stage 2 uses a rotating semicircular arm of strobe lights to illuminate the face from a large number of directions in about eight seconds, much more quickly than Light Stage 1. For this short a period of time, an actor can hold a steady facial expression for the entire capture session. By blending the geometry and reflectance of faces with different facial expressions, they have been able to create novel animated performances that can be realistically rendered from new points of view and under arbitrary illumination (Hawkins *et al.*, in press). Mark Sagar and his colleagues at Sony Pictures Imageworks used related techniques to create the digital stunt actors of Tobey Maguire and Alfred Molina from light stage data sets for the film Spider-Man 2.

For Light Stage 3, we built a complete sphere of 156 light sources that can illuminate an actor from all directions simultaneously. Each light consists of a collection of red, green, and blue LEDs interfaced to a computer so that any light can be set to any colour and intensity. The light stage can be used to reproduce the illumination from a captured lighting environment by using the light stage as a 156-pixel display device for the spherical image of incidental illumination. A person standing inside the sphere then becomes illuminated by a close approximation of the light that was originally captured. When composited over a background image of the environment, it appears nearly as if the person were there. This technique may improve on how green screens and virtual sets are used today. Actors in a studio can be filmed lit as if they were somewhere else, giving visual-effects artists much more control over the realism of the lighting process.

In our latest tests, we use a high-frame camera to capture how an actor appears under several rapidly cycling basis lighting conditions throughout the course of a performance. In this way, we can simulate the actor's appearance under a wide variety of different illumination conditions after filming, providing directors and cinematographers with never-before-available control of the actor's lighting during postproduction.

A REMAINING FRONTIER: DIGITIZING REFLECTANCE PROPERTIES

Significant challenges remain in the capture and simulation of physically accurate illumination in computer graphics. Although techniques for capturing object geometry

and lighting are maturing, techniques for capturing object reflectance properties — the way the surfaces of a real-world object respond to light — are still weak. In a recent project, a laboratory presented a relatively simple technique for digitizing surfaces with varying colour and shininess components.

They found that by moving a neon tube light source across a relatively flat object and recording the light's reflections using a video camera they could independently estimate the diffuse colour and the specular properties of every point on the object. For example, we digitized a 15th-century illuminated manuscript with coloured inks and embossed gold lettering). Using the derived maps for diffuse and specular reflection, we were able to render computer-graphics versions of the manuscript under any sort of lighting environment with realistic glints and reflections from different object surfaces.

A central complexity in digitizing reflectance properties for more general objects is that the way each point on an object's surface responds to light is a complex function of the direction of incidental light and the viewing direction — the surface's four-dimensional BRDF. In fact, the behaviour of many materials and objects is even more complicated than this, in that incidental light on other parts of the object may scatter within the object material, an effect known as subsurface scattering. Because this effect is a significant component of the appearance of human skin, it has been the subject of interest in the visual-effects industry. New techniques for simulating subsurface scattering effects on computer-generated models have led to more realistic renderings of

computer-generated actors and creatures, such as the Gollum character in Lord of the Rings.

Obtaining models of how real people and objects scatter light in their full generality is a subject of ongoing research. In a recent study, Goesele *et al.* (in press) used a computer-controlled laser to shine a narrow beam onto every point of a translucent alabaster sculpture and recorded images of the resulting light scattering using a specially chosen high-dynamic-range camera. By making the simplifying assumption that any point on the object would respond equally to any incidental and radiant light direction, the dimensionality of the problem was reduced from eight to four dimensions yielding a full characterization of the object's interaction with light under these assumptions. As research in this area continues, we hope to develop the capability of digitizing anything — no matter what it is made of or how it reflects light — so it can become an easily manipulated, photoreal computer model. For this, we will need new acquisition and analysis techniques and continued increases in computing power and memory capacity.

5

Orthographic Projections

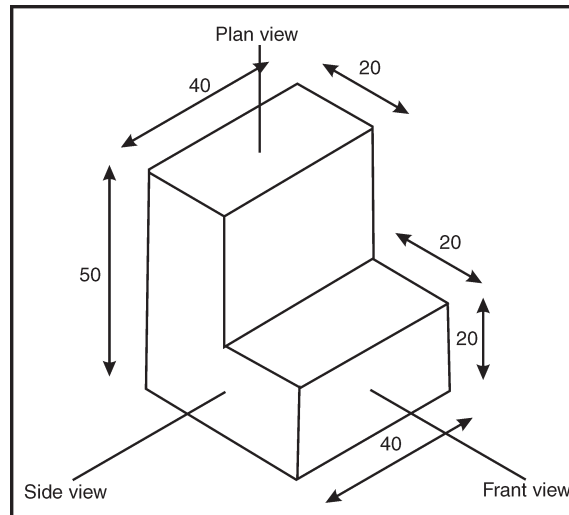
FIRST ANGLE PROJECTION

Orthographic projections are a way of representing 3D objects in 2D. It is useful in accurately representing the various sides of a 3D object. When you draw in orthographic, the front, side and plan views are drawn separately.

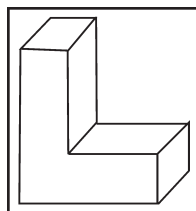
First angle projection is the standard orthographic projection used in Europe and Asia. First angle orthographic projection will always show the left view to the right of the front view as if the shape has been rolled to the right. The plan view is shown below as if it were rolled down.

When you draw an Orthographic, you must always use construction lines. This saves you from measuring the same measurements twice when you draw the side and plan views. The symbol for First Angle Projection is shown. It shown a cone shape drawn in first angle.

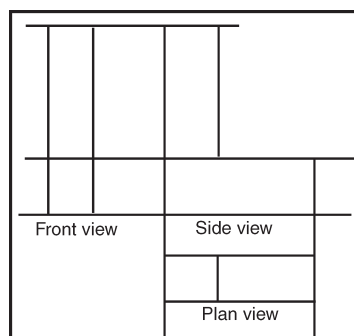
3-D Computer Graphics



Now copy the isometric view of the 'L' shape, then draw the first angle orthographic projection. Then, turn over the page and choose three shapes to draw in first angle. It is helpful to label the front, side and plan views first.



Opposite is a simple L-shape, drawn in three dimensions. Below is the same shape drawn in orthographic projection.

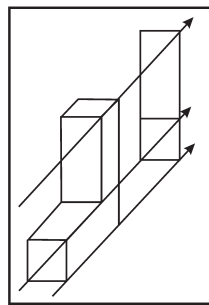


This orthographic projection appears to have three separate drawings but they are the same L-shape. The first draw-

ing is the front view (drawn looking straight at the front of the L-shape), the second is a drawing of the L-shape seen from the side and last of all a drawing from above known as a plan view. These lines are faint guidelines and they are drawn to help keep each view in line, level and the same size.

This is an example of first angle orthographic project. There is another type called third angle which is used commonly. The front, side and plan views are in different positions

THE FRONT VIEW



It is seen that standing directly in front of the L-shape, would only see the front edges, not the sides.

THE SIDE VIEW

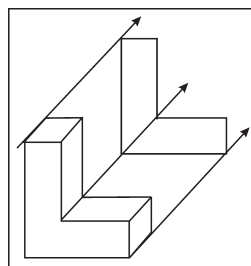
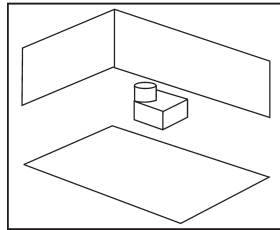


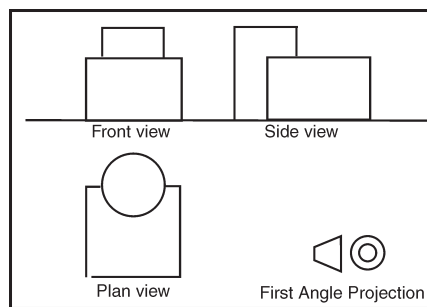
Fig. First Angle Projection

Now imagine standing directly at the side of the L-shape, the drawing opposite shows exactly what you would see.

Another example of first angle orthographic projection is shown below. Follow the lines as the front, side and plan view are constructed.



The final arrangement of the views are shown in the drawing below. Notice how the symbol for first angle orthographic projection has been added and the paper has a title block and borderline.



THIRD ANGLE PROJECTION

The 3rd angle projection is the most favoured of the two projections and is commonly seen on technical and architectural drawings. Tabs can display the 3rd angle projection with the right hand view. Students can also display projections rendered naturally, or solid white with lines, or as line drawings with the hidden lines included!

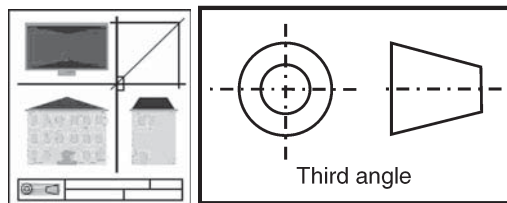
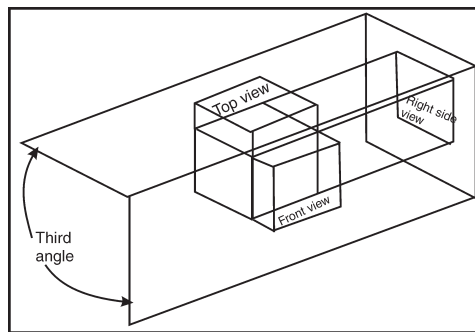


Figure below shows a third-angle projection of a cube. As you can see, you get a front view on the vertical plane, a top view on the horizontal plane, and a right side view on the profile plane.



Again you assume that the vertical plane is already in the plane of your drawing paper. To get the other two views into the same plane, you rotate them both clockwise. Figure below shows a third-angle projection of an object brought into a single plane. The top view is above the front view; the right side of the object, as shown in the front view, is towards the right side view; and the top, as shown in the front view, is towards the top view:

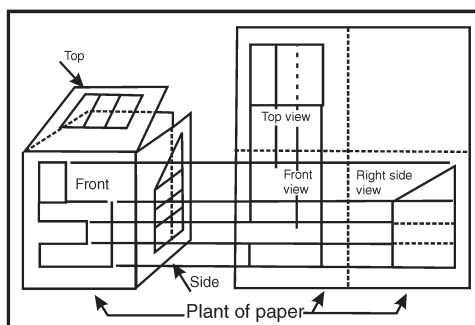
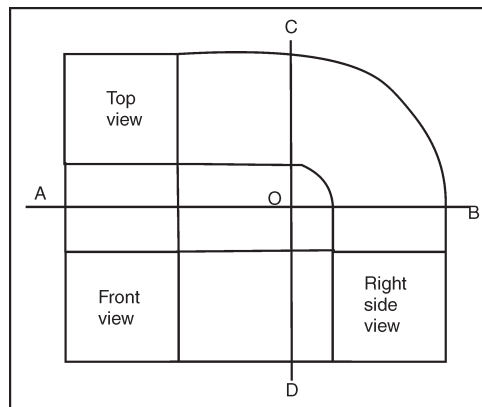
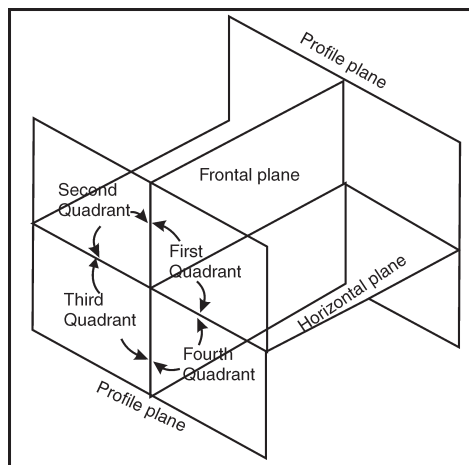


Figure below shows the basic principles of the method by which you would actually make the projection shown in figure. Draw a horizontal line AB and a vertical line CD, intersecting at O. AB represents the joint between the horizontal and the vertical plane; CD represents the joint between these two and

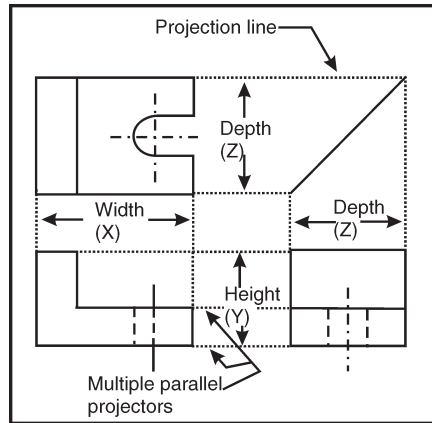
the profile plane. Any one of the three views could be drawn first, and the other two projected from it. Assume that the front view is drawn first on the basis of given dimensions of the front face. Draw the front view, and project it upward with vertical projection lines to draw the top view. Project the top view to CD with horizontal projection lines. With O as a center, use a compass to extend these projection lines to AB. Draw the right side view by extending the projection lines from AB vertically downward and by projecting the right side of the front view horizontally to the right



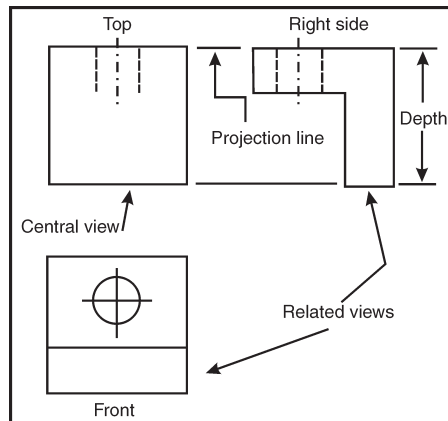
PLANES OF PROJECTION



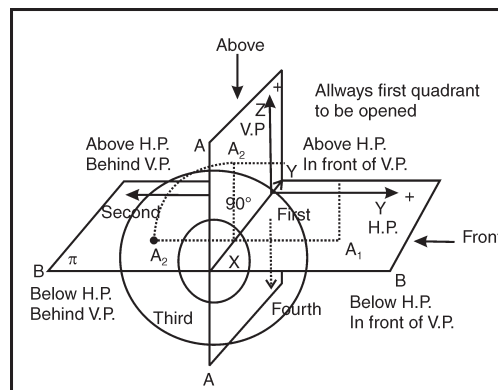
STANDARD VIEWS



ALTERNATIVE VIEWS

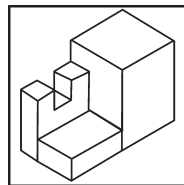
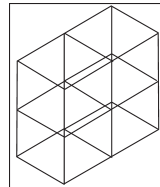


FOUR QUADRANTS

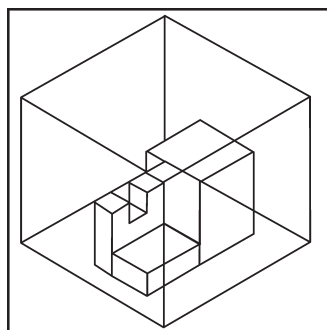
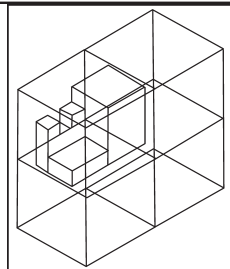
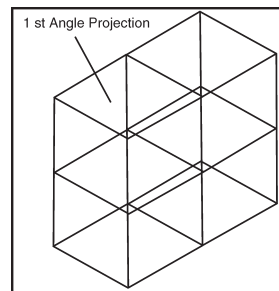


FIRST ANGLE EXAMPLES

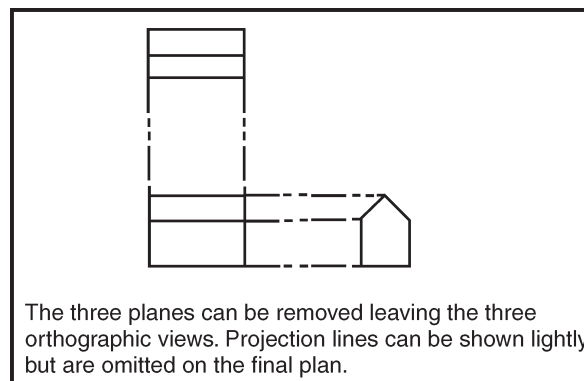
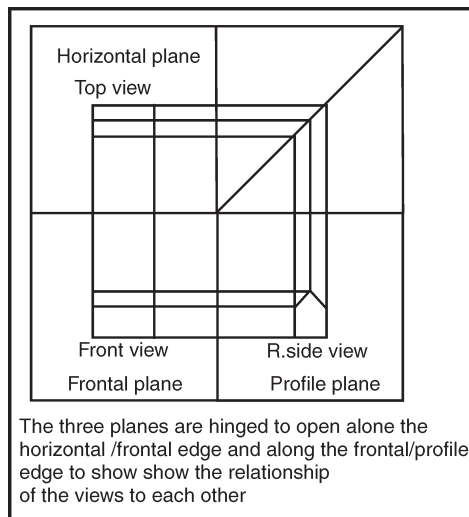
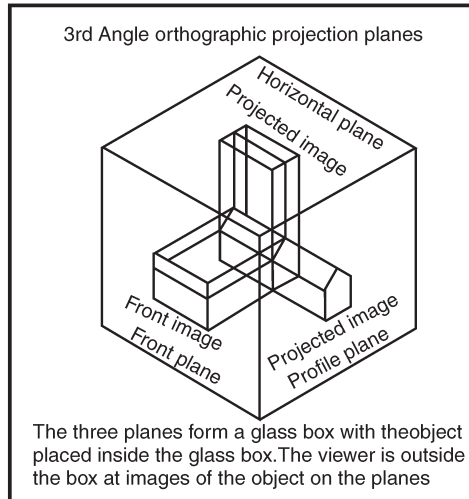
Example 1:



The 1st Angle Projection the section of the box we use is always the top left corner, which seems to make sense.



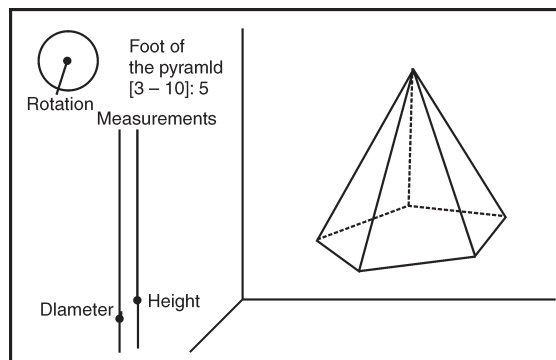
THIRD-ANGLE PROJECTION EXAMPLES



PROJECTION OF SOLIDS

The solid figures are drawn in two different forms of visibility. When first presented to the youngest pupils it is better to work with those of the first type (full solids). Later, when we want pupils to imagine even the back part of the solid, we use the other form, in which a dashed line marks the invisible edges.

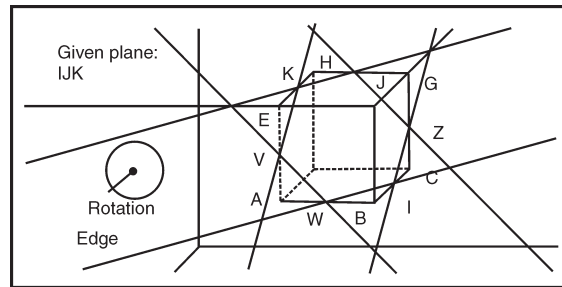
Interactive objects can also be used to construct sections of solids, making a decision about the mutual position of lines, which can be added into the interactive polyhedron, etc.



When working on a task, one can complete the task with the figure as presented. If the orientation of the figure is not suitable for the task, we can turn the solid into an optimal position.

Taking measurements of diagonals and angles is clearer if we use colour to highlight the triangles with which we are working. By rotating the object, pupils can better image the situation in space. When making a decision about the mutual position of lines in a figure, pupils can add the required lines into interactive polyhedron. By rotating the object, pupil can better image the situation in space. We can use the same

method to convince a pupil that he or she is wrong to consider the illusive point of intersection of skew lines to be a real point.

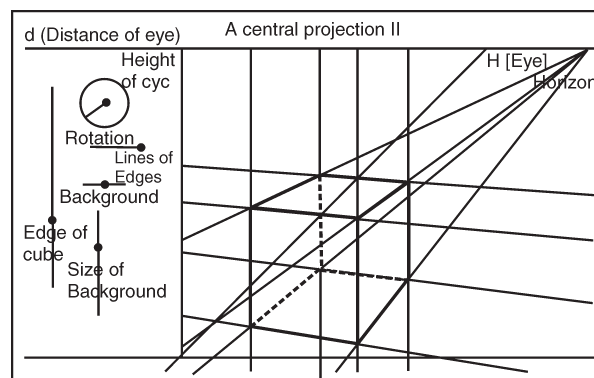


The generation of a cone or cylinder by rotating a triangle or rectangle can be demonstrated to the class as a whole.

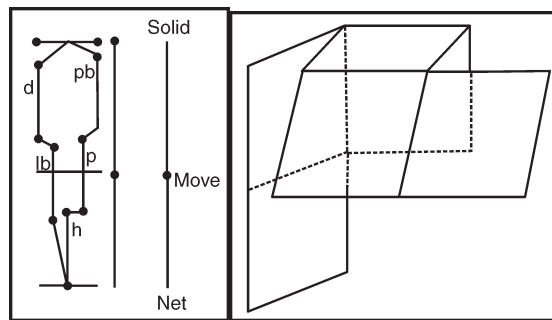
PROJECTIONS

The second group of teaching files are used to illustrate vertical, oblique and central projections. We can study the features of a particular projection by applying the projection to a cube.

Parameters of each projection are variable. In this way we can compare oblique projections, which differ by the angle between the x, y axes and by the ratio of units on those axes. Later we can experimentally verify whether a given projection preserves parallelism, perpendicularity or length.



We recommend showing the illustration of central projections because of their similarity to our visual perception of the world. Pupils who use only parallel projection in maths lessons would certainly be interested in it because of an attractive link between mathematics and art.



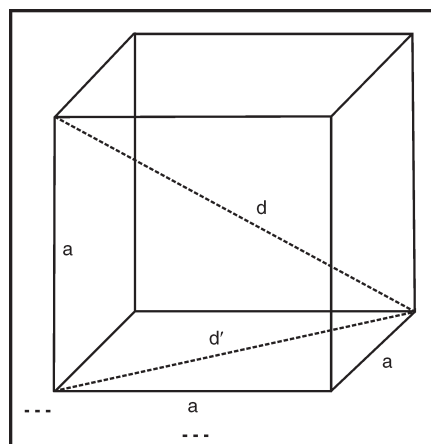
TYPES OF SOLIDS

CUBE

The cube is a mathematical body formed by six equal squares.

Other names are cubus or hexahedron.

In the 1980's Rubik's cube was so popular that it was simply called "the cube".



There are 12 equal edges.

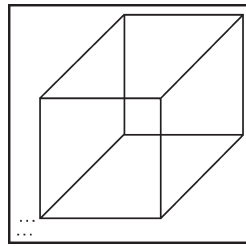
Three edges meet in a corner and stand perpendicularly on each other in pairs.

The length of an edge is a . There are

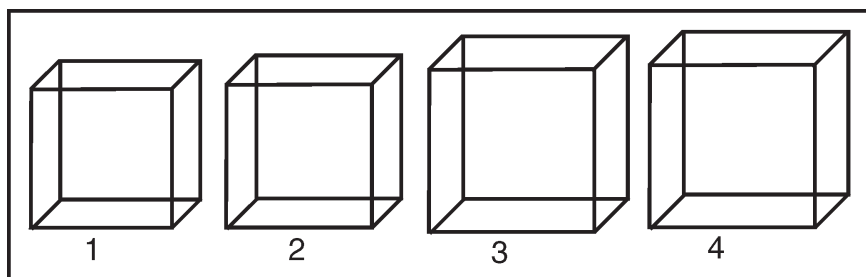
- The 12 square diagonals of the length $d'=\sqrt{2}*a$.
- The four space diagonals of the length $d=\sqrt{3}*a$.
- The volume $V=a^3$, the surface $O=6*a^2$.

The circumscribed sphere has the radius $R=a*\sqrt{3}/2$, the inscribed sphere the radius $r=a/2$.

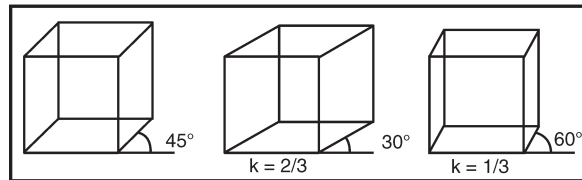
CUBES IN PERSPECTIVE



If students shall draw the picture of a cube in perspective and you tell them before that all edges of the cube are equal some of them draw the picture on the left. There is the question, how much you must shorten to get a nice view. If you have the choice of the following four drawings most people would choose picture 3 as the best.

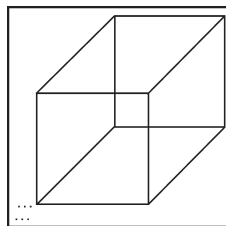


The sloping line is about half as large as the true length of an edge. Thus you get the ratio $k=1/2$. The ratio depends on the angles of the sloping lines. The three following statements produce good pictures:



The mathematical base is the “sloping parallel projection”, where all ratios and angles are possible. You choose simple angles and simple ratios. Every real 2d-tripod OABC can be produced by parallel projection of a Cartesian 3d-tripod O'A'B'C'.

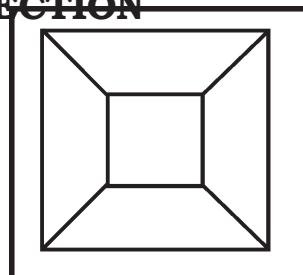
CUBES IN PERSPECTIVE



Actually the drawing with equal edges on the left was not used as a picture of a cube. Nevertheless you may take it.

The advantage is that the length in direction of the lines are real. The “isometric perspective” (30° instead of 45°) is not so distorted and preferred.

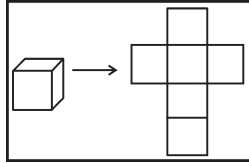
CENTRAL PROJECTION



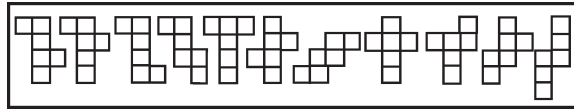
Think of the edge model of a cube.

If you project it with light which comes from a point you can get the picture on the left.

NETS OF A CUBE

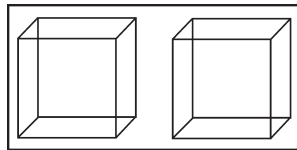


Think of a paper cube and cut it along the edges. You get nets of a cube. There are 11 nets.

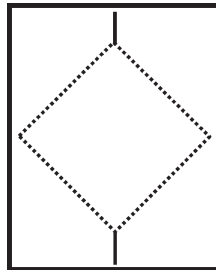


STEREOGRAM

You can see the cube three-dimensionally in the following picture.



SHADOW PICTURES OF A ROTATING CUBE



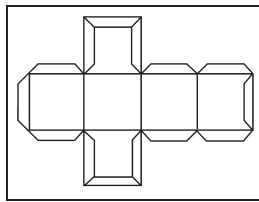
A nice exhibit is a rotating edge model of a cube. It is produced by parallel rays.

You see the figures on the left one after the other on a screen (square on the top, hexagon with diagonal and a rectangle with a middle line).

You don't believe that these figures come from a cube and must look twice.

The cube is positioned in the way, that two edges in opposite lie on top of each other. An axis (red) is interrupted in the middle.

MODEL WITH SQUARES



This is the well known way to build a cube.

Draw a net and give the edges stripes for gluing.

Flex the stripes and glue them on the pointed fields with the same colour. The cube will be closed with a lid on the right.

MODEL WITH EDGES

There are different methods of building cubes with rods.

Remove the top of a match and connect the sticks with two-component gluing.

Take a toothpick or chopstick and connect them with balls of modelling clay.

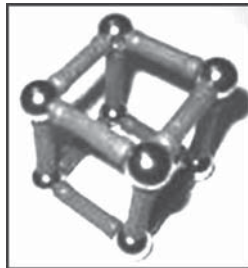
Cut equal wire pieces with pincers and connect them by soldering, so that they form a cube. It is clever to fix the three pieces of a corner and let them touch before soldering. You can manage if you use “die helfende Hand” (the helping hand) for two ends and the real hand for the third end. Cut drinking straws in equal pieces and connect them with tripods of florist’s wire or paper clips.

More exact: There are building kits, which use 12 straws and 8 tripods of plastic.

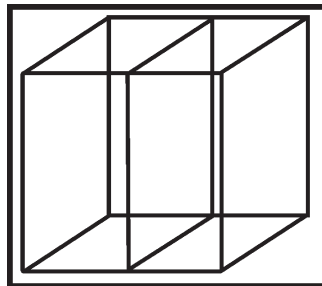
You can make tripods (below) yourself. You bend them twice in six lines. There are bends in the tops, the two ends of a wire meet in the middle.



You can find a clever fashionable toy: Bar magnets form the edges, steel balls the corners.



The centre of the cube is a symmetry centre.

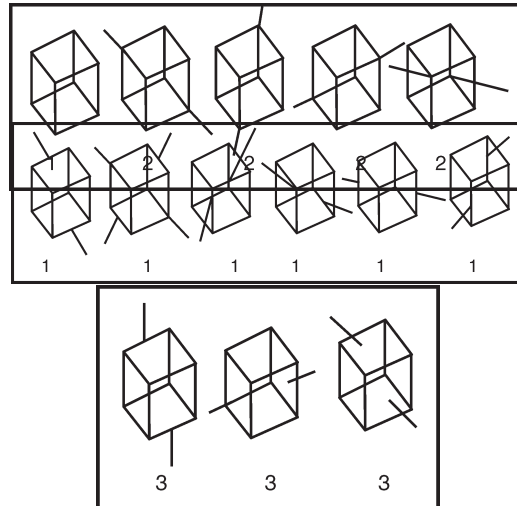


The cube has nine symmetry planes.

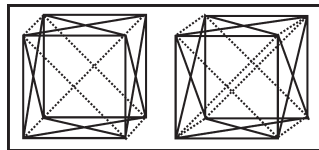
Three planes lie parallel to the side squares and go through the centre (picture).

Six planes go through opposite edges and two body diagonals. They divide the cube into prisms. You can find 13 rotation axes. If you turn around one of these axes, the cube goes back to itself.

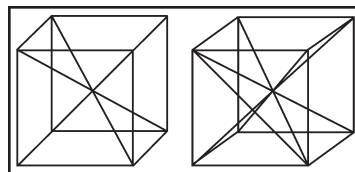
The following picture illustrates these facts. The numbers under the cubes indicate the number of turns.



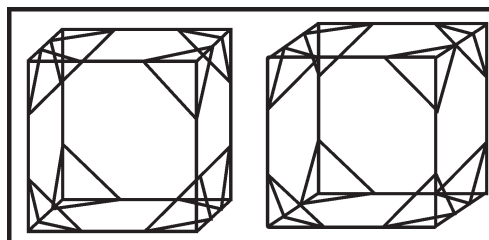
ALL DIAGONALS OF THE SURFACE SQUARES



ALL SPATIAL DIAGONALS

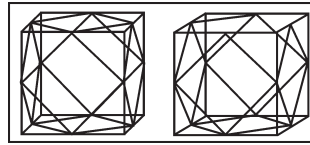


CUBE WITH CUT CORNERS



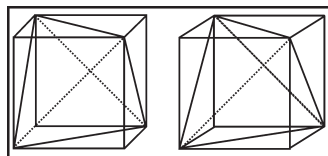
Cut the corners of a cube. Divide the edges in three equal pieces to do this. You get a body formed by 6 octagons and 8 equilateral triangles.

CUBEOCTAHEDRON



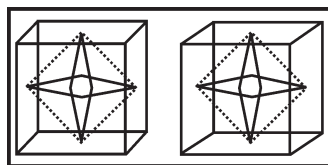
Cut the corners of a cube. Take half the edges. You get a body formed by 6 squares and 8 equilateral triangles..

TETRAHEDRON IN THE CUBE



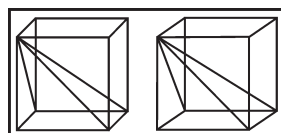
Draw some square diagonals and you will get a tetrahedron.

OCTAHEDRON IN THE CUBE

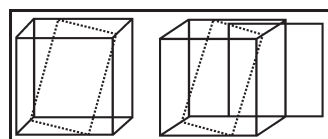


Join the centres of the squares by lines. You get an octahedron. If you join the centres of the triangles of a octahedron, a cube develops again. Cube and octahedron are dual to each other.

THREE PYRAMIDS OF EQUAL VOLUMES

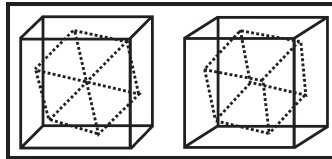


THE LARGEST SQUARE INSIDE A CUBE



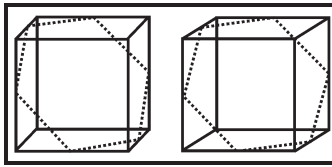
The red square is the largest square which fits a cube.
The corners of the square divide the edges in 1:3.

CUBE IN THE CUBE



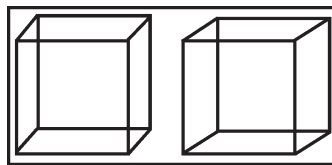
The red cube is the smallest cube which touches all sides of the black cube.

HEXAGON INSIDE A CUBE

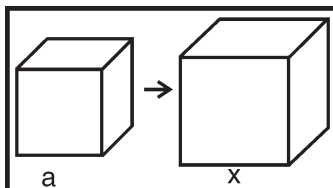


Join centres of some edges. You get an intersection through the cube. The intersection line is a hexagon of the edge length $\frac{\sqrt{2}}{2}a$, if a is the edge of the cube.

A SPATIAL EQUILATERAL HEXAGONS





















The ancient Greeks could get rid of the plague after an answer of the oracle of Delos, if they doubled the volume of its cube altar. (This is one version of the legend.)

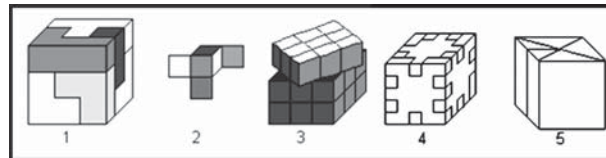


The problem of the cube duplication goes to the equation $2a^3=x^3$ and the solution $x=a*2^{(1/3)}$.

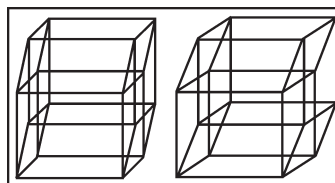
This was no solution, because the distance x had to be found from the length a only with circle and ruler. Circles and straight lines lead to linear and quadratic equations, which $x^3 = a^3$ does not belong to. You can form polycubes, if you add cubes touching in one or several squares as shown.

| | | | | | | | | | |
|-----------|---|---|---|---|---|---|---|--|---|
| Name: | Drilling | Tetrawürfel | Pentawürfel | Hexawürfel | Heptawürfel | Oktowürfel | Würfel 9 | Würfel 10 | Würfel 11 |
| Beispiel: |  |  |  |  |  |  |  |  |  |
| Anzahl: | 2 | 8 | 29 | 166 | 1023 | 6922 | 48311 | 346543 | 2522522 |
| Name: | | Tetromino | Pentomino | Hexomino | Heptomino | Oktomino | 9-omino | 10-omino | 11-omino |
| Beispiel: |  |  |  |  |  |  |  |  |  |
| Anzahl: | 2 | 5 | 12 | 35 | 108 | 369 | 1285 | 4655 | 17073 |

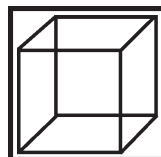
There are many puzzles based on polycubes. Here is my “hit list”:



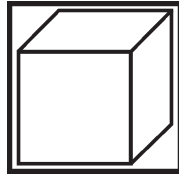
HYPERCUBE (4DIMENSIONAL CUBE)



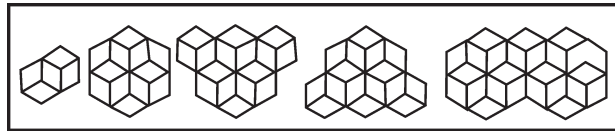
If you like, every representation of a cube in the plane is a optical illusions. You think you see a 3D cube, though the drawing plane is 2D. Well known illusions with cubes are “tilt figures”. I restrict on them.



The Necker cube (on the left) shows a cube in two perspectives (on the right).

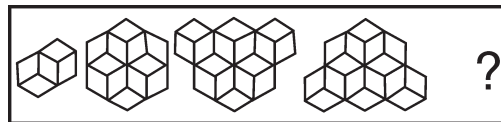


You switch over from one sight to the other. A square is sometimes in front or in the back. You can only see one view at one moment.

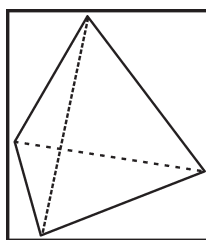


From below or above three or five cubes. Five or three cubes? How many cubes?

The five pictures above are ambiguous.



TETRAHEDRA



The tetrahedron has 4 faces, 4 vertices, and 6 edges. Each face is an equilateral triangle. Three faces meet at each vertex. Begin with a tetrahedron of edge length s . Its faces are equilateral triangles. The length of their sides is s , and the measure of their interior angles is $\pi/3$.

First, find the area of each triangular face. Multiply that by the number of faces to get the total surface area, A .

3-D Computer Graphics

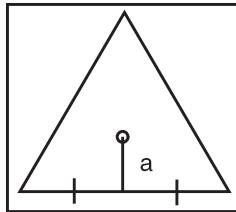
$$\text{Triangle area} = \frac{3}{4 \tan \frac{\pi}{3}} s^2 = \frac{\sqrt{3}}{4} s^2$$

$$A = 4 \left(\frac{\sqrt{3}}{4} \right) s^2 = \sqrt{3} s^2$$

The dihedral angle formula can be applied here because three faces meet at each vertex. All of the faces are equilateral triangles, so let $\alpha = \beta = \gamma = \pi/3$.

$$\cos \theta = \frac{\cos \gamma - \cos \alpha \cos \beta}{\sin \alpha \sin \beta} = \frac{\cos \frac{\pi}{3} - \cos^2 \frac{\pi}{3}}{\sin^2 \frac{\pi}{3}} = \frac{\frac{1}{2} - \left(\frac{1}{2}\right)^2}{\left(\frac{\sqrt{3}}{2}\right)^2} = \frac{1}{3}$$

$$\theta = \cos^{-1} \left(\frac{1}{3} \right)$$



Find the apothem of a face, and use it in the calculations for the inradius and circumradius.

$$a = \frac{s}{2 \tan \frac{\pi}{3}} = \frac{\sqrt{3}}{6} s$$

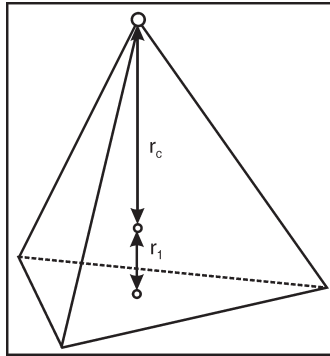
$$r_i = a \sqrt{\frac{1 - \cos \theta}{1 + \cos \theta}} = \left(\frac{\sqrt{3}}{6} s \right) \sqrt{\frac{1 - \frac{1}{3}}{1 + \frac{1}{3}}} = \frac{\sqrt{6}}{12} s$$

$$r_c = \sqrt{r_i^2 + a^2 + \left(\frac{s}{2} \right)^2} = \sqrt{\frac{s^2}{24} + \frac{s^2}{12} + \frac{s^2}{4}} = \frac{\sqrt{6}}{4} s$$

Now, the volume formula.

$$V = \frac{A_f}{3} = \frac{(\sqrt{3}s^2)\left(\frac{\sqrt{6}}{12}s\right)}{3} = \frac{\sqrt{2}}{12}s^3$$

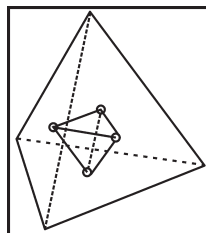
The tetrahedron is also a pyramid, and its height is the sum of the inradius and the circumradius. Use that fact and apply the pyramid volume formula. Redundant calculations like this are a good way of checking the results.



$$\begin{aligned} V &= \frac{1}{3}(\text{base area})(\text{height}) \\ &= \frac{1}{3}\left(\frac{\sqrt{3}}{4}s^2\right)\left(\frac{\sqrt{6}}{12}s + \frac{\sqrt{6}}{4}s\right) \\ &= \frac{\sqrt{2}}{12}s^3 \end{aligned}$$

OTHER PROPERTIES

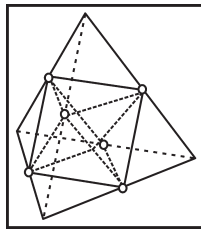
The tetrahedron is its own dual, meaning that if the centers of the adjacent faces are connected with line segments, the resulting figure is another tetrahedron. The smaller tetrahedron shown here has one-ninth the



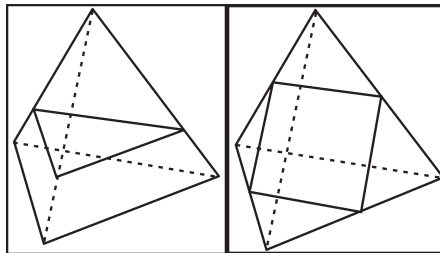
volume of the larger one, but it is not possible to assemble nine tetrahedrons into one.

The tetrahedron has 24 symmetries. Tetrahedra do not pack space, but it is possible to pack space by combining tetrahedra and octahedra. The tetrahedron has no parallel faces, no parallel edges, and no diametrically opposite vertices. All of these properties are unique among the Platonic solids.

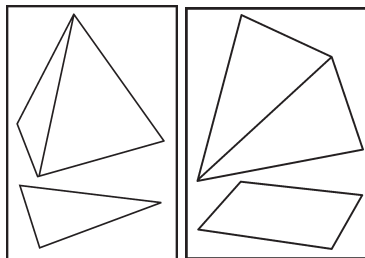
When the midpoints of the adjacent edges of a tetrahedron are connected, an octahedron is formed.



A cross-section of a tetrahedron can be an equilateral triangle or a square.



A planar projection of a tetrahedron can be an equilateral triangle or a square.



6

Design and Analysis Technologies

The Design and Analysis Technologies critical technology area includes technologies or processes that are pervasive within the aerospace and defence sector. The technology elements within the Design Technologies critical technology area are depicted in the figure below and described in subsequent paragraphs:

MULTIDISCIPLINARY DESIGN AND OPTIMIZATION

Multidisciplinary design and optimization is as the name implies, the process of combining a full set of computational design tools to create an optimum design. The process is necessarily iterative in nature and all of the disciplines normally utilized in an aircraft design are computationally intensive.

An MDO approach for an aircraft could include aerodynamics, structures, and systems Computer Aided Engineering (CAE) tools. Initial design assumptions would be input to each CAE toolset and the constraints and parameters to be optimized defined. Each CAE suite would then compute design parameters that would be utilized by the other CAE tools as a subset of their required inputs.

The ultimate design would theoretically be structurally sounder, lighter and more cost effective to fabricate. The design timeframe would be also very much shortened. The challenges to this process are in the exchange of data between the CAE applications and the tuning of the entire process to achieve convergence on the final solution set in an efficient manner.

STRUCTURAL ANALYSIS

The optimization of analytical design tools is a process that will lead to shortened design time frames, lighter and more efficient designs, with reduced production and life cycle costs of the final design. The many analytical tools now available have been typically developed for specific applications and are often not readily applicable outside of their original design target arena. An example lies in the structural analysis field where tools developed for metallics will be much different from those developed for composite materials where material properties may vary according to axis. The ability to rapidly define an optimized aircraft structure having light weight, and improved fatigue and damage tolerance capabilities, is a critical technology to maintain competitive leadership in the development and

supply of future new aircraft. This will be achieved by the extensive use of computerized methods for structural analysis and design optimization, and the analysis of failure and fracture mechanics. The methods must be integrated with the in-house design and manufacturing data bases, the 3-D CAD/CAM systems, and also be easy to use. Suppliers and partners will have access to the resulting design information via Technical Data Interchange (TDI). This will ensure consistency with an up-to-date knowledge of the requirements for loads, interfaces and the space envelopes available for their products. The immediate dissemination to suppliers of information on design changes will help diminish subsequent redesign activity and the time and cost penalties incurred for rework.

The preliminary structural design will often use detailed Finite Element Methods (FEM) for analysis, coupled with constrained optimization, and the process must be highly automated for rapid creation of FEM meshing for models. In order to achieve shortened design cycle time, the loads and dynamics stiffness requirements must become available much sooner than at present. This will require early development of MDO models for overall aerodynamic and structural optimization that will define the static and dynamic loads for flight and ground operations. Trade-off studies must rapidly search for the best designs and arrive at realistic structural sizes, providing space envelopes and accurate weights to minimize subsequent redesign.

STRUCTURAL DESIGN, ANALYSIS AND OPTIMIZATION

Shortened design cycle times are necessary for achieving market advantage in the aerospace and defence sector. Improvements in the structural analysis, design and

optimization of gas turbine engines is necessary to achieve these goals while also meeting the overall objectives of increased durability and efficiency at lower costs.

A Multi-disciplinary Design Optimization (MDO) approach that combines finite element analysis and aerodynamic design techniques is employed. MDO is necessary to rapidly determine the structure of the engine and identify critical areas requiring further or more detailed analysis.

Many of the structural and aerodynamic codes developed by companies are proprietary in nature and the integration and refinement of these codes is an on-going challenge.

COMPUTATIONAL FLUID DYNAMICS

COMPUTATIONAL DEVELOPMENT AND VALIDATION

Computational Fluid Dynamics (CFD) has had the greatest effect on both aircraft and engine design of any single design tool over the past twenty-five years. Computational power and cost have enabled widespread application and development of CFD techniques. Computational fluid dynamics is basically the use of computers to numerically model flows of interest. Nodes in the flowpath are identified and equations of motion solved at these locations to identify flow parameters. In essence a grid or mesh is defined over the surface of the object that extends outwards into the flowfield containing the object. Flow equations are then calculated at each node in the grid, and iteratively recalculated until all results for each node are within an acceptable variance. The equations used are either Euler

based which do not include viscous effects (boundary layers) directly, or Navier-Stokes equations which include viscous effects and which produce more accurate but computationally more demanding solutions. Such methods can be used for external flows about an aircraft or for internal flows in a gas turbine including combustion. The Euler based analyses are typically less computationally demanding but are less precise for modeling separated flows on wings and bodies, or for internal reversed flows. It should be noted that Navier first developed his equations in 1823 and that Stokes refined them in 1845. The development of solutions to these equations was not feasible until the latter part of this century. Today much R&D effort on NS methods is expended on improving modeling of the turbulent flow terms for specific problems. Numerous forms of Euler and Navier-Stokes solutions have been developed to address particular design problems. Solutions to these equations are dependent on experimentation for both coefficients and for validation.

Mesh selection and node placement is critical to the solution of the flowfield. The automated generation of meshes is now in wide spread use and can often be linked to Computer Aided Engineering and Design tools. The form of the equation used, the density of the mesh or grid and convergence requirements determine computational demands. Complete aircraft solutions require huge computer resources and much R&D is aimed at improving the speed of the solution.

COMPUTATIONAL FLUID DYNAMICS - GAS TURBINES

CFD is perhaps the single most critical technology for gas turbine engines. Gas turbine CFD needs have typically posed

the greatest challenges to engine designers, and computational power and code developers. While CFD is of utmost importance to the engine designer it is a very specific disciplinary design requirement and competence is held by a very small number of engine design firms worldwide.

Computation techniques for gas turbine engines also tend to be very module specific — compressor, transition duct, combustor, turbine and exhaust duct/military afterburner are examples. Computational techniques are often also specific to engine size class and thus Canada, focusing on small gas turbines, has a specific set of technology requirements.

Advanced 3D CFD codes have been used to generate the following design improvements:

- In the compressor to develop advanced swept airfoils capable of high compression ratios that in turn yield higher efficiency at less weight and with a smaller parts count (significant life cycle cost factor);
- In the combustor for higher intensity (smaller volumes with much higher energy density) combustors that approach stoichiometric conditions to yield higher efficiency with lower weight; and
- In the turbine to produce higher stage loading with reduced turbine cooling air requirements that again reduces weight and cost while reducing fuel burn.

COMBUSTION SYSTEMS COMPUTATION

The combustor of a gas turbine engine is that part of the engine that receives the compressed air from the compressor. Energy is added to the airflow in the combustor in the form

of chemical energy derived from fuel. The combustor discharge air is expanded across a turbine or turbines where energy is extracted to drive the compressor and gearbox of a turboshaft/turboprop engine, or to provide jet thrust via a turbofan and core nozzle in a thrust engine.

Small gas turbines, of the size that have typically been designed and built in Canada pose significant design challenges because of their size. Pratt and Whitney Canada combustors are the highest intensity combustors in the world, where intensity can be thought of as the amount of energy converted per unit volume within the combustor. The design objectives for gas turbine engines, including small ones, are to increase both overall pressure ratios and cycle temperatures, which lead to increased efficiency and smaller size and weight, while simultaneously producing reduced noise and noxious emissions levels.

Combustor technology development challenges for Canadian engine manufacturers include.

Computational fluid dynamics: CFD analyses are complicated by the reverse flow designs typically selected to maintain short combustors within small volumes. Cooling flow and chemical additions to the CFD design further complicate the process as the temperatures of gases at the core of the flows are well above the melting temperatures of the combustor materials. Pressure losses and cooling flow requirements must be minimized to improve performance.

Materials: Increasing compressor ratios result in increased compressor discharge temperatures and decreased cooling capability. These increased temperatures also push for higher fuel to air ratios and higher temperatures within

the combustor. Stoichiometric ratio is that ratio when all oxygen is consumed in the combustion process leaving less air for cooling. Materials challenges in this environment are the most demanding. Fuel injection and mixing: CFD and injector specific techniques are required.

Emissions: While not legislated and not contributing significantly in absolute terms, there is a drive for lower emissions that drives designs often in the opposite direction to those factors identified above.

AERODYNAMICS AND FLIGHT MECHANICS

Aerodynamics is the study of forces on wing bodies and controls due to air pressure and viscous (drag) effects. Flight mechanics is the study of the resulting motion of objects through the air and includes the stability and control Behaviour. The laws of motion and aerodynamics are combined to ensure that an aircraft flies in the intended manner. Much of the aerodynamics and flight mechanics work that is pursued for the purposes of aircraft designed and built in Canada will pertain to such issues as the design of improved wings, the integration of various components onto an aircraft or issues such as flight in adverse conditions where the handling qualities of an aircraft will be adversely influenced by the build up of ice on the surface of the wing. Advanced technology development in this field will be directed towards supersonic transports and eventually hypersonic flight. There are considerable differences between fixed wing and rotary wing aircraft aerodynamics and flight mechanics and both areas are of considerable interest to the Canadian aerospace and defence industry.

Technologies relevant to Aerodynamics and Flight Mechanics are described below:

ADVANCED AERODYNAMICS AND HANDLING

Included here are technologies that will enable the Canadian Aerospace industry to contribute to the design of advanced concept aircraft technologies or components or be the lead design integrator.

These enabling technologies should be pursued dependent on their links to, and pre-positioning for potential application to specific aircraft platforms or types as follows:

- *Future Transport Aircraft:* Future transport aircraft will have to demonstrate increased speed and load carrying capabilities over greatly extended ranges. Specific targets have been set by the U.S. for next generation transport aircraft although no new advanced concept transport aircraft are currently well advanced. Wing loading factors will double over that of existing aircraft with the development of materials new to the transport aircraft envelope. For shorter-range aircraft, a key enabling technology will be that of high efficiency turboprop engines with cruise speeds above the M.72 range. Propulsion technology and propulsion integration issues, aircraft design optimization, CFD, and materials technology development and insertion will be key to the success of the future transport aircraft.
- *Hypersonic Aircraft:* Hypersonic aircraft are in exploratory or advanced development model stage at this time and will be used initially for low cost space

launch and delivery platforms and subsequently for commercial transport. Propulsion technologies are significant to hypersonic vehicle feasibility and are now the limiting factor. Variable cycle engines, advanced materials, endothermic fuels and fuel control technologies are key aeropropulsion technology elements where significant R&D remains unsatisfied. Numerous controls and materials research topics require further investment as well, although less uncertainty remains in these areas due to advances made through the shuttle Programmes.

- *Advanced Rotorcraft*: Future rotorcraft will demonstrate increased cruise speeds of 200 kts or greater with tiltrotor speeds approaching 450 kts. These cruise speeds will be possible at significantly reduced vibration levels and with greatly increased range/fuel economy. Many of the design concepts for attaining these performance improvements are already in development, however much work remains undone.
- *Advanced Rotorcraft Flight Mechanics*: For both conventional helicopter and tiltrotor blades, the wings and propulsion system operate in a very complex aeromechanical environment. Aerodynamics, structures, vibration and acoustics parameters are inseparable and typically drive the design of the entire air vehicle. In trimmed forward flight the advancing blade tip will be moving at near sonic velocities whilst the retreating blade is often in near stall conditions.

ADVANCED DESIGN AND DEVELOPMENT

General aviation aircraft pose specific design challenges in all aspects of their design and fabrication. Increasing

availability of low cost and high performance avionics, advanced composite designs and powerplant integration all offer opportunities for general aviation aircraft designers and builders.

Many of the technologies being furthered for use in military unmanned aerial vehicles will be of pertinence to general aviation aircraft.

Low cost gas turbine technologies and composite structures development and certification issues will likely be the technologies of greatest interest.

The development of technologies for military purposes will underwrite some of the costs of introduction of those design concepts into general aviation use.

EXPERIMENTAL ASSESSMENT AND PERFORMANCE

Analytical design and analysis techniques are a prerequisite to reductions in design cycle time, design and production costs, and improved safety and environmental impact. The development of these analytical or numerical design techniques will remain heavily dependent on experimental validation of design codes and performance targets for another 10-15 years. Whereas in the past, experimental resources such as wind tunnels were used primarily for design development and refinement, in the future they may increasingly be used for the validation of computational design tools.

Notwithstanding the foregoing, there will continue to be a requirement for national facilities including wind tunnels, engine test facilities, flight test resources, and specialized resources including icing tunnels and rig test facilities for

some time to come. *Experimental design and performance validation technology investment will be required in the following areas to support the aerospace industry in Canada:*

- *Data Capture and Analysis Automation:* Automated methods for intelligent data capture and analysis will be required to reduce large facility run times and meet the challenges of design tool validation. This will require investment both in sensors and in computational tools;
- *Experimental Code Development:* Increased data capture rates and fidelity will be required and will necessitate the development of specific codes for experimental design and performance validation. Facilities and infrastructure will have to be maintained or enhanced to achieve these goals; and
- *Infrastructure Support:* The maintenance of critical national facilities will have to be supported in concert with other government departments and industry. The objective will not necessarily be to create new facilities but rather to improve the functionality of existing resources to meet the needs of new technology developments.

AEROPROPULSION PERFORMANCE ASSESSMENT

Test cells utilized for Canadian aero-engine Programmes, and also those developed for sale, have typically been sea-level static facilities offering little or no altitude, forward flight velocity or temperature pressure simulation. Some limited flying test bed capability exists in Canada for the testing of

engines. That being said, the National Research Council has participated in numerous international projects in the process ensuring that a world leading test cell capability exists both for engine qualification testing, performance testing and for the development of performance assessment techniques.

Engine test cells take a number of forms. Sea level test facilities are used for Engine Qualification Testing that involves the monitoring of a relatively small number of parameters over long periods where in-service usage is evaluated in a time compressed manner. Qualification testing also involves the ingestion of ice or water to ensure that unacceptable engine degradation does not occur in those instances. The NRC Institute for Aerospace Research has developed world recognized icing testing competencies and icing test facilities that are used by Canadian and off-shore engine manufacturers for qualification testing.

Altitude test cells are used to qualify engines over a full flight envelope as opposed to the endurance type testing previously described.

The National Research Council in collaboration with Pratt and Whitney Canada have developed and operated one small altitude test cell at NRC for some time. An initiative that began in 2000 will see the development and commissioning of a somewhat larger and more capable altitude facility, again as a collaborative effort between NRC and P&WC.

Test cells can also be used for the analysis of problems or validation of problem resolution. In these cases the test cells often require enhanced instrumentation suites and a

much more careful design to ensure that performance parameters are correctly measured. World interest in advanced test cell technologies has been directed at those required to support hypersonic vehicles for military uses or for space launch vehicles.

This type of test cell is very resource intensive and highly specialized and will likely be of little interest or utility to any but a limited number of Canadian firms. The Short Take Off and Vertical Landing (STOVL) version of the F35 Joint Strike Fighter has recently posed new challenges in the world of aeropropulsion testing. For this testing, in-flow preparation, exhaust treatment, fan drive systems, and 6 axis thrust measurement in the vertical axis will all pose significant new challenges to the performance assessment community.

ADVANCED CONCEPTS OF DESIGN

ANALYSIS AND DESIGN INTEGRATION

Advanced aerodynamics profile development in Canada will be primarily directed at wing design for subsonic aircraft carrying less than 120 passengers. The objective of work done on advanced aerodynamic profiles will be to increase efficiency and cruise speeds through reduced drag while improving structural and control characteristics. Wing profile, control surface effectiveness, airframe and engine interface effects with the wing and wing tip designs are areas of research and development interest. Also, developments improving wing-flap high lift performance are important areas for minimizing wing size required and hence costs.

Laminar flow control is a term that deserves discussion. Airflow over wings begins as a laminar or ordered flowfield and will transition to a higher drag producing turbulent flow based on flow characteristics such as speed and wing influences including wing shape, surface roughness. It has been estimated that if laminar flow could be maintained on the wings of a large aircraft, fuel savings of up to 25% could be achieved.

Wing and flight characteristics of small aircraft are such that laminar flow can be relatively easily maintained over much of the flight envelope. A variety of methods can be used to increase laminar flow regions on aircraft of larger size and having higher Reynolds numbers and sweep angles.

Computational fluid dynamics will be the most important technology relevant to the development of advanced aerodynamic profiles. A number of areas require R&D activity and support for aircraft design particular to Canadian aerospace interests. Large-scale CFD code refinement and validation is one area requiring work to improve accuracy and reduce computational times for MDO by more rapid design convergence. These CFD codes will also require validation in Laboratories and in wind tunnels.

ALL-ELECTRIC AIRCRAFT CONCEPT DEVELOPMENT

The all-electric aircraft will utilize electronic actuators to replace equivalent hydraulic system components. The intent is to save weight and increase reliability. For example, electrical generators would provide power to electric actuators for flight control surface movement rather than

equivalent hydraulic powered components. Electric power cables are lighter and less prone to damage or service induced degradation such as fitting vibration that results in leakage in hydraulic systems. Alternate power supply redundancy is an additional advantage of this concept. Challenges associated with this type of technology insertion would be related to electromagnetic interference (EMI), and rapid load fluctuations imposed on the power generation engines.

FLY-BY-LIGHT CONCEPT DEVELOPMENT

Fly-by-Light (FBL) technology involves the replacement of electronic data transmission, mechanical control linkages, and electronic sensors with optical components and subsystems. Benefits include lower initial acquisition and life cycle costs, reduced weight, and increased aircraft performance and reliability.

Fibre-optic cables are essentially immune to electromagnetic interference and therefore not affected by fields generated by other lines or electrical devices in close proximity, nor are they affected by lightning strikes. For flight controls, hydraulic or electric actuators are still employed but receive their command inputs via fibre-optic cables. Weight reductions are significant as the fibre-optic cables need only be protected from physical damage, whereas electric cables must be insulated and shielded increasing weight significantly. Also with a FBL connection multiple routes can be readily provided that are well separated to provide control redundancy. There are a number of enabling technologies that must be developed in order to enable photonics technology

insertion. Fibre-optic connectors for in-line and end connections must be developed that are durable and insensitive to in-service maintenance activities. Fibre-optic sensors development will also be necessary to allow the achievement of the full range of benefits that can be obtained in fly-by-light aircraft. This technology is usually associated with smart structures concepts such as smart skins where fibre-optic cabling can be readily embedded in a composite lay-up to achieve dispersed damage, stress, temperature or vibration sensing capability.

DETECTION MANAGEMENT AND CONTROL SYSTEMS

Regional airliners and helicopters operating in lower level airspace are increasingly exposed to hazardous icing conditions. This has increased the need for technologies for proactive and reactive ice detection and protection. Reactive technologies are those related to the detection of runback icing and attempt to monitor real-time or infer likely aerodynamic performance degradation.

Proactive systems forecast the potential for icing conditions and provide on-board avoidance advisory information. Reactive systems provide reasonable protection of the aircraft within the regulated flight envelope but are essentially go/no-go decision aids. Aircraft on Search and Rescue Missions and most civil transport aircraft often do not have the option of avoiding hazardous icing conditions and should have pro-active pilot advisors and ice removal systems.

Reactive ice detection devices include: embedded sensors that are mounted on the wing surface in a critical location

and monitor ice build-up; and aerodynamic performance sensors that typically monitor pressure within the boundary layer of the wing to determine lift performance degradation. Proactive systems require the remote measurement of Liquid Water Content (LWC), Outside Air Temperature (OAT) and Mean Volume Diameter (MVD) of the liquid water. Knowledge of these three parameters is required to predict hazardous icing conditions. Additional R&D work on MVD measurement is required.

Ice control and removal systems may use heated air from the engines or electrical heat elements to remove ice from airfoil surfaces. Coatings that are termed "iceophobic" may also be applied to minimize ice build-up. CFD tools are needed to Analyse ice-buildup characteristics, assess aerodynamic degradation, and improve ice removal air supply performance. This technology area is of particular interest because of the types of aircraft produced in Canada and because of climatic conditions.

DESIGN TECHNIQUES

A previously stated objective for noise reduction is in the order of 6 EPNdB (Effective Perceived Noise in dB). This objective can be achieved through the utilization of larger by-pass ratio fans, innovative design concepts for turbo fans and sound conscious designs in the combustor and exhaust nozzles/liners. Generally speaking, noise improvements and fuel efficiency must be improved to meet future regulatory requirements without sacrifice of overall engine efficiency. Of special interest will be advanced ducted propulsors (ADF) that offer both noise attenuation and increased efficiency

potential. This technology area will be heavily dependent on computational design techniques and multidisciplinary design optimization.

The reduction in aircraft emissions is also a regulated requirement. While small aircraft engines contribute an insignificant amount of pollution they are still the targets of increased environmental scrutiny. Regulatory requirements are targeted at Nitrous Oxides (NO_x), Carbon Monoxide (CO) and visible particulate emissions. CFD analysis techniques specific to combustion processes will be the major tool used to lower aeropropulsion emissions.

7

Computer Graphics Software

In computer graphics, graphics software or image editing software is a programme or collection of programmes that enable a person to manipulate visual images on a computer. Computer graphics can be classified into two distinct categories: raster graphics and vector graphics. Before learning about computer software that manipulates or displays these graphics types, you should be familiar with both. Many graphics programmes focus exclusively on either vector or raster graphics, but there are a few that combine them in interesting and sometimes unexpected ways. It is simple to convert from vector graphics to raster graphics, but going the other way is harder. Some software attempts to do this. Most graphics programmes have the ability to import and export one or more graphics file formats. The use of a swatch is a palette of active colours that are selected and rearranged by the preference of the user. A

swatch may be used in a programme or be part of the universal palette on an operating system, it is used to change the colour of a project, that may be text, image or video editing.

Several graphics programmes support animation, or digital video. Vector graphics animation can be described as a series of mathematical transformations that are applied in sequence to one or more shapes in a scene. Raster graphics animation works in a similar fashion to film-based animation, where a series of still images produces the illusion of continuous movement.

HISTORY

SuperPaint (1973) was one of the earliest graphics software applications. Fauve Matisse (later Macromedia xRes) was a pioneering programme of the early 1990s, notably introducing layers in customer software. Currently Adobe Photoshop is one of the most used and best-known graphics programmes, having displaced more custom hardware solutions in the early 1990s, but was initially subject to various litigation. GIMP is a popular open source alternative to Adobe Photoshop. Other applications include:

- JPhotoBrush Pro, Java-based multi-platform, freeware.
- Corel Paint Shop Pro

INTEGRATED SOFTWARE

Integrated software is software for personal computers that combines the most commonly used functions of many

productivity software programmes into one application. The integrated software genre has been largely overshadowed by fully functional office suites, most notably Microsoft Office, but at one time was considered the “killer application” type responsible for the rise and dominance of the IBM PC in the desktop business computing world. In the early days of the PC before GUIs became common, user interfaces were text-only and were operated mostly by function key and modifier key sequences. Every programme used a different set of keystrokes, making it difficult for a user to master more than one or two programmes. Programmes were loaded from floppy disk, making it very slow and inconvenient to switch between programmes and difficult or impossible to exchange data between them (to transfer the results from a spreadsheet to a word processor document for example). In response to these limitations, vendors created multifunction “integrated” packages, eliminating the need to switch between programmes and presenting the user with a more consistent interface.

The potential for greater ease-of-use made integrated software attractive to home markets as well as business, and packages such as the original AppleWorks for the Apple II and Jane for the Commodore 128 were developed in the 1980s to run on most popular home computers of the day. Commodore even produced the Plus/4 computer with a simple integrated suite built into ROM. Context MBA was an early example of the genre, and featured spreadsheet, database, chart-making, word processing and terminal emulation functions. However, because it was written in Pascal for portability, it ran slowly on the relatively

underpowered systems of the day. Lotus 1-2-3, which followed it, had fewer functions but was written in assembler, providing it with a speed advantage that allowed it to become the predominant business application for personal computers. The integrated software market of today is exemplified by entry-level programmes such as Microsoft Works which are often bundled with personal computers as “starter” productivity suites.

SYSTEMS ARCHITECTURE

A system architecture or systems architecture is the conceptual model that defines the structure, behaviour, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structure of the system which comprises system components, the externally visible properties of those components, the relationships (e.g. the behaviour) between them, and provides a plan from which products can be procured, and systems developed, that will work together to implement the overall system. The language for architecture description is called the architecture description language (ADL).

OVERVIEW

There is no universally agreed definition of which aspects constitute a system architecture, and various organizations define it in different ways, including:

- The fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution.

- The composite of the design architectures for products and their life cycle processes.
- A representation of a system in which there is a mapping of functionality onto hardware and software components, a mapping of the software architecture onto the hardware architecture, and human interaction with these components.
- An allocated arrangement of physical elements which provides the design solution for a consumer product or life-cycle process intended to satisfy the requirements of the functional architecture and the requirements baseline.
- An architecture is the most important, pervasive, top-level, strategic inventions, decisions, and their associated rationales about the overall structure (i.e., essential elements and their relationships) and associated characteristics and behaviour.
- A description of the design and contents of a computer system. If documented, it may include information such as a detailed inventory of current hardware, software and networking capabilities; a description of long-range plans and priorities for future purchases, and a plan for upgrading and/or replacing dated equipment and software.
- A formal description of a system, or a detailed plan of the system at component level to guide its implementation.
- The structure of components, their interrelationships, and the principles and guidelines governing their design and evolution over time.

A system architecture can best be thought of as a set of representations of an existing (or To Be Created) system. It is used to convey the informational content of the elements comprising a system, the relationships among those elements, and the rules governing those relationships. The architectural components and set of relationships between these components that an architecture describes may consist of hardware, software, documentation, facilities, manual procedures, or roles played by organizations or people. A system architecture is primarily concerned with the internal interfaces among the system's components or subsystems, and the interface between the system and its external environment, especially the user. (In the specific case of computer systems, this latter, special interface, is known as the computer human interface, AKA human computer interface, or CHI; formerly called the man-machine interface.) A system architecture can be contrasted with system architecture engineering, which is the method and discipline for effectively implementing the architecture of a system:

- It is a *method* because a sequence of steps is prescribed to produce or change the architecture of a system within a set of constraints.
- It is a *discipline* because a body of knowledge is used to inform practitioners as to the most effective way to architect the system within a set of constraints.

HISTORY

It is important to keep in mind that the modern systems architecture did not appear out of nowhere. Systems architecture depends heavily on practices and techniques

which were developed over thousands of years in many other fields most importantly being, perhaps, civil architecture. Prior to the advent of digital computers, the electronics and other engineering disciplines used the term system as it is still commonly used today. However, with the arrival of digital computers and the development of software engineering as a separate discipline, it was often necessary to distinguish among engineered hardware artifacts, software artifacts, and the combined artifacts. A programmable hardware artifact, or computing machine, that lacks its software programme is impotent; even as a software artifact, or programme, is equally impotent unless it can be used to alter the sequential states of a suitable (hardware) machine. However, a hardware machine and its software programme can be designed to perform an almost illimitable number of abstract and physical tasks. Within the computer and software engineering disciplines (and, often, other engineering disciplines, such as communications), then, the term system came to be defined as containing all of the elements necessary (which generally includes both hardware and software) to perform a useful function.

Consequently, within these engineering disciplines, a system generally refers to a programmable hardware machine and its included programme. And a systems engineer is defined as one concerned with the complete device, both hardware and software and, more particularly, all of the interfaces of the device, including that between hardware and software, and especially between the complete device and its user (the CHI). The hardware engineer deals (more

or less) exclusively with the hardware device; the software engineer deals (more or less) exclusively with the software programme; and the systems engineer is responsible for seeing that the software programme is capable of properly running within the hardware device, and that the system composed of the two entities is capable of properly interacting with its external environment, especially the user, and performing its intended function. By analogy, then, a systems architecture makes use of elements of both software and hardware and is used to enable design of such a composite system. A good architecture may be viewed as a 'partitioning scheme,' or algorithm, which partitions all of the system's present and foreseeable requirements into a workable set of cleanly bounded subsystems with nothing left over. That is, it is a partitioning scheme which is exclusive, inclusive, and exhaustive.

A major purpose of the partitioning is to arrange the elements in the sub systems so that there is a minimum of communications needed among them. In both software and hardware, a good sub system tends to be seen to be a meaningful "object". Moreover, a good architecture provides for an easy mapping to the user's requirements and the validation tests of the user's requirements. Ideally, a mapping also exists from every least element to every requirement and test. A *robust architecture* is said to be one that exhibits an optimal degree of fault-tolerance, backward compatibility, forward compatibility, extensibility, reliability, maintainability, availability, serviceability, usability, and such other quality attributes as necessary and/or desirable.

TYPES OF SYSTEMS ARCHITECTURES

Several types of systems architectures (underlain by the same fundamental principles) have been identified as follows:

- Collaborative Systems (such as the Internet, intelligent transportation systems, and joint air defense systems)
- Manufacturing Systems
- Social Systems
- Software and Information Technology Systems
- Strategic Systems Architecture

SYSTEMS ARCHITECT

In systems engineering, the systems architect is the high-level designer of a system to be implemented. The systems architect establishes the basic structure of the system, defining the essential core design features and elements that provide the framework for all that follows, and are the hardest to change later. The systems architect provides the engineering view of the users' vision for what the system needs to be and do, and the paths along which it must be able to evolve, and strives to maintain the integrity of that vision as it evolves during detailed design and implementation.

OVERVIEW

In systems engineering, the systems architect is responsible for:

- Interfacing with the user(s) and sponsor(s) and all other stakeholders in order to determine their (evolving) needs.

- Generating the highest level of system requirements, based on the user's needs and other constraints such as cost and schedule.
- Ensuring that this set of high level requirements is consistent, complete, correct, and operationally defined.
- Performing cost-benefit analyses to determine whether requirements are best met by manual, software, or hardware functions; making maximum use of commercial off-the-shelf or already developed components.
- Developing partitioning algorithms (and other processes) to allocate all present and foreseeable requirements into discrete partitions such that a minimum of communications is needed among partitions, and between the user and the system.
- Partitioning large systems into (successive layers of) subsystems and components each of which can be handled by a single engineer or team of engineers or subordinate architect.
- Interfacing with the design and implementation engineers, or subordinate architects, so that any problems arising during design or implementation can be resolved in accordance with the fundamental architectural concepts, and user needs and constraints.
- Ensuring that a maximally robust architecture is developed.
- Generating a set of acceptance test requirements, together with the designers, test engineers, and the

user, which determine that all of the high level requirements have been met, especially for the computer-human-interface.

- Generating products such as sketches, models, an early user guide, and prototypes to keep the user and the engineers constantly up to date and in agreement on the system to be provided as it is evolving.
- Ensuring that all architectural products and products with architectural input are maintained in the most current state and never allowed to become obsolete.

MAIN TOPICS OF SYSTEMS ARCHITECT

Large systems architecture was developed as a way to handle systems too large for one person to conceive of, let alone design. Systems of this size are rapidly becoming the norm, so architectural approaches and architects are increasingly needed to solve the problems of large systems.

USERS AND SPONSORS

Engineers as a group do not have a reputation for understanding and responding to human needs comfortably or for developing humanly functional and aesthetically pleasing products. Architects *are* expected to understand human needs and develop humanly functional and aesthetically pleasing products. A good architect is a translator between the user/sponsor and the engineers—and even among just engineers of different specialities. A good architect is also the principal keeper of the user's vision of the end product—and of the process of deriving requirements from and implementing that vision.

Determining what the users/sponsors actually need, rather than what they say they want, is not engineering. An architect does not follow an exact procedure. S/he communicates with users/sponsors in a highly interactive way— together they extract the true *requirements* necessary for the engineered system. The architect must remain constantly in communication with the end users. Therefore, the architect must be intimately familiar with the user's environment and problem. (The engineer need only be very knowledgeable of the potential engineering solution space.)

HIGH LEVEL REQUIREMENTS

The user/sponsor should view the architect as the user's representative and provide *all input through the architect*. Direct interaction with project engineers is generally discouraged as the chance of mutual misunderstanding is very high. The user requirements' specification should be a joint product of the user and architect: the user brings his needs and wish list, the architect brings knowledge of what is likely to prove doable within cost and time constraints. When the user needs are translated into a set of high level requirements is also the best time to write the first version of the acceptance test, which should, thereafter, be religiously kept up to date with the requirements. That way, the user will be absolutely clear about what s/he is getting. It is also a safeguard against untestable requirements, misunderstandings, and requirements creep. The development of the first level of engineering requirements is not a purely analytical exercise and should also involve both the architect and engineer. If any compromises are to

be made— to meet constraints like cost, schedule, power, or space, the architect must ensure that the final product and overall look and feel do not stray very far from the user's intent. The engineer should focus on developing a design that optimizes the constraints but ensures a workable and reliable product.

The architect is primarily concerned with the comfort and usability of the product; the engineer is primarily concerned with the producibility and utility of the product. The provision of needed services to the user is the true function of an engineered system. However, as systems become ever larger and more complex, and as their emphases move away from simple hardware and software components, the narrow application of traditional systems development principles is found to be insufficient— the application of the more general principles of systems, hardware, and software architecture to the design of (sub)systems is seen to be needed. An architecture is also a simplified model of the finished end product— its primary function is to define the parts and their relationships to each other so that the whole can be seen to be a consistent, complete, and correct representation of what the user had in mind— especially for the computer-human-interface. It is also used to ensure that the parts fit together and relate in the desired way.

It is necessary to distinguish between the architecture of the user's world and the engineered systems architecture. The former represents and addresses problems and solutions in the *user's* world. It is principally captured in the computer-human-interfaces (CHI) of the engineered system. The engineered system represents the *engineering* solutions—

how the *engineer* proposes to develop and/or select and combine the components of the technical infrastructure to support the CHI. In the absence of an experienced architect, there is an unfortunate tendency to confuse the two architectures. But— the engineer thinks in terms of hardware and software and the technical solution space, whereas the user may be thinking in terms of solving a problem of getting people from point A to point B in a reasonable amount of time and with a reasonable expenditure of energy, or of getting needed information to customers and staff. A systems architect is expected to combine knowledge of both the architecture of the user's world and of (all potentially useful) engineering systems architectures. The former is a joint activity with the user; the latter is a joint activity with the engineers. The product is a set of high level requirements reflecting the user's requirements which can be used by the engineers to develop systems design requirements. Because requirements evolve over the course of a project, especially a long one, an architect is needed until the system is accepted by the user: the architect is the best insurance that all changes and interpretations made during the course of development do not compromise the user's viewpoint.

COST/BENEFIT ANALYSES

Most engineers are specialists. They know the applications of one field of engineering science intimately, apply their knowledge to practical situations— that is, solve real world problems, evaluate the cost/benefits of various solutions within their specialty, and ensure the correct operation of whatever they design. Architects are generalists. They are

not expected to be experts in any one technology but are expected to be knowledgeable of many technologies and able to judge their applicability to specific situations. They also apply their knowledge to practical situations, but evaluate the cost/benefits of various solutions using different technologies, for example, hardware versus software versus manual, and assure that the system as a whole performs according to the user's expectations. Many commercial-off-the-shelf or already developed hardware and software components may be selected independently according to constraints such as cost, response, throughput, etc. In some cases, the architect can already assemble the end system unaided. Or, s/he may still need the help of a hardware or software engineer to select components and to design and build any special purpose function. The architects (or engineers) may also enlist the aid of specialists— in safety, security, communications, special purpose hardware, graphics, human factors, test and evaluation, quality control, RMA, interface management, etc. An effective systems architectural team must have immediate access to specialists in critical specialties.,

PARTITIONING AND LAYERING

An architect planning a building works on the overall design, making sure it will be pleasing and useful to its inhabitants. While a single architect by himself may be enough to build a single-family house, many engineers may be needed, in addition, to solve the detailed problems that arise when a novel high-rise building is designed. If the job is large and complex enough, parts of the architecture may

be designed as independent components. That is, if we are building a housing complex, we may have one architect for the complex, and one for each type of building, as part of an *architectural team*. Large automation systems also require an architect and much engineering talent. If the engineered system is large and complex enough, the systems architect may defer to a hardware architect and a software architect for parts of the job, although they all may be members of a joint architectural team. The architect should sub-allocate the system requirements to major components or subsystems that are within the scope of a single hardware or software engineer, or engineering manager and team. *But the architect must never be viewed as an engineering supervisor.* (If the item is sufficiently large and/or complex, the chief architect will sub-allocate portions to more specialized architects.) Ideally, each such component/subsystem is a sufficiently stand-alone object that it can be tested as a complete component, separate from the whole, using only a simple testbed to supply simulated inputs and record outputs. That is, it is not necessary to know how an air traffic control system works in order to design and build a data management subsystem for it.

It is only necessary to know the constraints under which the subsystem will be expected to operate. A good architect ensures that the system, however complex, is built upon relatively simple and “clean” concepts for each (sub)system or layer and is easily understandable by everyone, especially the user, without special training. The architect will use a minimum of heuristics to ensure that each partition is well defined and clean of kludges, work-arounds, short-cuts, or

confusing detail and exceptions. As user needs evolve, (once the system is fielded and in use), it is a lot easier subsequently to evolve a simple concept than one laden with exceptions, special cases, and lots of “fine print.” *Layering* the architecture is important for keeping the architecture sufficiently simple at each *layer* so that it remains comprehensible to a single mind. As layers are ascended, whole systems at *lower layers* become simple *components* at the *higher layers*, and may disappear altogether at the *highest layers*.

ACCEPTANCE TEST

The acceptance test is a principal responsibility of the systems architect. It is the chief means by which the architect will prove to the user that the system is as originally planned and that all subordinate architects and engineers have met their objectives.

COMMUNICATIONS WITH USERS AND ENGINEERS

A building architect uses sketches, models, and drawings. An automation systems (or software or hardware) architect should use sketches, models, and prototypes to discuss different solutions and results with users, engineers, and other architects. An early, draft version of the user’s manual is invaluable, especially in conjunction with a prototype. A set of (engineering) *requirements* as a sole, or even principal, means of communicating with the users is explicitly to be avoided. Nevertheless, it is important that a workable, well written *set of requirements*, or specification, be created which

is understandable to the customer (so that they can properly sign off on it). But it must use precise and unambiguous language so that designers and other implementers are left in no doubt as to meanings or intentions. In particular, all requirements must be testable, and the initial draft of the test plan should be developed contemporaneously with the requirements. All stakeholders should sign off on the acceptance test descriptions, or equivalent, as the sole determinant of the satisfaction of the requirements, at the outset of the programme.

SYSTEMS ENGINEERING

Systems engineering is an interdisciplinary field of engineering that focuses on how complex engineering projects should be designed and managed over the life cycle of the project. Issues such as logistics, the coordination of different teams, and automatic control of machinery become more difficult when dealing with large, complex projects. Systems engineering deals with work-processes and tools to handle such projects, and it overlaps with both technical and human-centered disciplines such as control engineering, industrial engineering, organizational studies, and project management.

HISTORY

The term *systems engineering* can be traced back to Bell Telephone Laboratories in the 1940s. The need to identify and manipulate the properties of a system as a whole, which in complex engineering projects may greatly differ from the sum of the parts' properties, motivated the

Department of Defense, NASA, and other industries to apply the discipline. When it was no longer possible to rely on design evolution to improve upon a system and the existing tools were not sufficient to meet growing demands, new methods began to be developed that addressed the complexity directly. The evolution of systems engineering, which continues to this day, comprises the development and identification of new methods and modeling techniques. These methods aid in better comprehension of engineering systems as they grow more complex. Popular tools that are often used in the systems engineering context were developed during these times, including USL, UML, QFD, and IDEF0. In 1990, a professional society for systems engineering, the *National Council on Systems Engineering* (NCOSE), was founded by representatives from a number of U.S. corporations and organizations. NCOSE was created to address the need for improvements in systems engineering practices and education.

As a result of growing involvement from systems engineers outside of the U.S., the name of the organization was changed to the International Council on Systems Engineering (INCOSE) in 1995. Schools in several countries offer graduate programmes in systems engineering, and continuing education options are also available for practicing engineers.

CONCEPT

Systems engineering signifies both an approach and, more recently, a discipline in engineering. The aim of education in systems engineering is to simply formalize the approach and in doing so, identify new methods and research

opportunities similar to the way it occurs in other fields of engineering. As an approach, systems engineering is holistic and interdisciplinary in flavour.

ORIGINS AND TRADITIONAL SCOPE

The traditional scope of engineering embraces the design, development, production and operation of physical systems, and systems engineering, as originally conceived, falls within this scope. “Systems engineering”, in this sense of the term, refers to the distinctive set of concepts, methodologies, organizational structures (and so on) that have been developed to meet the challenges of engineering functional physical systems of unprecedented complexity. The Apollo programme is a leading example of a systems engineering project.

The use of the term “ system engineer “ has evolved over time to embrace a wider, more holistic concept of “systems” and of engineering processes. This evolution of the definition has been a subject of ongoing controversy [9], and the term continues to be applied to both the narrower and broader scope.

HOLISTIC VIEW

Systems engineering focuses on analyzing and eliciting customer needs and required functionality early in the development cycle, documenting requirements, then proceeding with design synthesis and system validation while considering the complete problem, the system lifecycle. Oliver *et al.* claim that the systems engineering process can be decomposed into

- a *Systems Engineering Technical Process*, and
- a *Systems Engineering Management Process*.

Within Oliver's model, the goal of the Management Process is to organize the technical effort in the lifecycle, while the Technical Process includes *assessing available information, defining effectiveness measures, to create a behaviour model, create a structure model, perform trade-off analysis, and create sequential build & test plan*. Depending on their application, although there are several models that are used in the industry, all of them aim to identify the relation between the various stages mentioned above and incorporate feedback. Examples of such models include the Waterfall model and the VEE model.

INTERDISCIPLINARY FIELD

System development often requires contribution from diverse technical disciplines. By providing a systems (holistic) view of the development effort, systems engineering helps mold all the technical contributors into a unified team effort, forming a structured development process that proceeds from concept to production to operation and, in some cases, to termination and disposal. This perspective is often replicated in educational programmes in that systems engineering courses are taught by faculty from other engineering departments which, in effect, helps create an interdisciplinary environment.

MANAGING COMPLEXITY

The need for systems engineering arose with the increase in complexity of systems and projects, in turn exponentially

increasing the possibility of component friction, and therefore the reliability of the design. When speaking in this context, complexity incorporates not only engineering systems, but also the logical human organization of data. At the same time, a system can become more complex due to an increase in size as well as with an increase in the amount of data, variables, or the number of fields that are involved in the design. The International Space Station is an example of such a system. The development of smarter control algorithms, microprocessor design, and analysis of environmental systems also come within the purview of systems engineering. Systems engineering encourages the use of tools and methods to better comprehend and manage complexity in systems. Some examples of these tools can be seen here:

- *System model, Modeling, and Simulation,*
- *System architecture,*
- *Optimization,*
- *System dynamics,*
- *Systems analysis,*
- *Statistical analysis,*
- *Reliability analysis, and*
- *Decision making*

Taking an interdisciplinary approach to engineering systems is inherently complex since the behaviour of and interaction among system components is not always immediately well defined or understood. Defining and characterizing such systems and subsystems and the interactions among them is one of the goals of systems

engineering. In doing so, the gap that exists between informal requirements from users, operators, marketing organizations, and technical specifications is successfully bridged.

SCOPE

One way to understand the motivation behind systems engineering is to see it as a method, or practice, to identify and improve common rules that exist within a wide variety of systems. Keeping this in mind, the principles of systems engineering — holism, emergent behaviour, boundary, et al. — can be applied to any system, complex or otherwise, provided systems thinking is employed at all levels. Besides defense and aerospace, many information and technology based companies, software development firms, and industries in the field of electronics & communications require systems engineers as part of their team. An analysis by the INCOSE Systems Engineering center of excellence (SECOE) indicates that optimal effort spent on systems engineering is about 15-20% of the total project effort. At the same time, studies have shown that systems engineering essentially leads to reduction in costs among other benefits. However, no quantitative survey at a larger scale encompassing a wide variety of industries has been conducted until recently. Such studies are underway to determine the effectiveness and quantify the benefits of systems engineering. Systems engineering encourages the use of modeling and simulation to validate assumptions or theories on systems and the interactions within them.

Use of methods that allow early detection of possible failures, in safety engineering, are integrated into the design

process. At the same time, decisions made at the beginning of a project whose consequences are not clearly understood can have enormous implications later in the life of a system, and it is the task of the modern systems engineer to explore these issues and make critical decisions. There is no method which guarantees that decisions made today will still be valid when a system goes into service years or decades after it is first conceived but there are techniques to support the process of systems engineering. Examples include the use of soft systems methodology, Jay Wright Forrester's System dynamics method and the Unified Modeling Language (UML), each of which are currently being explored, evaluated and developed to support the engineering decision making process.

EDUCATION

Education in systems engineering is often seen as an extension to the regular engineering courses, reflecting the industry attitude that engineering students need a foundational background in one of the traditional engineering disciplines (e.g. automotive engineering, mechanical engineering, industrial engineering, computer engineering, electrical engineering) plus practical, real-world experience in order to be effective as systems engineers. Undergraduate university programmes in systems engineering are rare. INCOSE maintains a continuously updated Directory of Systems Engineering Academic Programmes worldwide. As of 2006, there are about 75 institutions in United States that offer 130 undergraduate and graduate programmes in systems engineering. Education in systems engineering can be taken as *SE-centric* or *Domain-centric*.

- *SE-centric* programmes treat systems engineering as a separate discipline and all the courses are taught focusing on systems engineering practice and techniques.
- *Domain-centric* programmes offer systems engineering as an option that can be exercised with another major field in engineering.

Both these patterns cater to educate the systems engineer who is able to oversee interdisciplinary projects with the depth required of a core-engineer.

SYSTEMS ENGINEERING TOPICS

Systems engineering tools are strategies, procedures, and techniques that aid in performing systems engineering on a project or product. The purpose of these tools vary from database management, graphical browsing, simulation, and reasoning, to document production, neutral import/export and more.

SYSTEM

There are many definitions of what a system is in the field of systems engineering. Below are a few authoritative definitions:

- ANSI/EIA-632-1999: “An aggregation of end products and enabling products to achieve a given purpose.”
- IEEE Std 1220-1998: “A set or arrangement of elements and processes that are related and whose behaviour satisfies customer/operational needs and provides for life cycle sustainment of the products.”

- ISO/IEC 15288:2008: “A combination of interacting elements organized to achieve one or more stated purposes.”
- NASA Systems Engineering Handbook: “(1) The combination of elements that function together to produce the capability to meet a need. The elements include all hardware, software, equipment, facilities, personnel, processes, and procedures needed for this purpose. (2) The end product (which performs operational functions) and enabling products (which provide life-cycle support services to the operational end products) that make up a system.”
- INCOSE Systems Engineering Handbook: “homogeneous entity that exhibits predefined behaviour in the real world and is composed of heterogeneous parts that do not individually exhibit that behaviour and an integrated configuration of components and/or subsystems.”
- INCOSE: “A system is a construct or collection of different elements that together produce results not obtainable by the elements alone. The elements, or parts, can include people, hardware, software, facilities, policies, and documents; that is, all things required to produce systems-level results. The results include system level qualities, properties, characteristics, functions, behaviour and performance. The value added by the system as a whole, beyond that contributed independently by the parts, is primarily created by the relationship among the parts; that is, how they are interconnected.”

THE SYSTEMS ENGINEERING PROCESS

Depending on their application, tools are used for various stages of the systems engineering process:

USING MODELS

Models play important and diverse roles in systems engineering. A model can be defined in several ways, including:

- An abstraction of reality designed to answer specific questions about the real world
- An imitation, analogue, or representation of a real world process or structure; or
- A conceptual, mathematical, or physical tool to assist a decision maker.

Together, these definitions are broad enough to encompass physical engineering models used in the verification of a system design, as well as schematic models like a functional flow block diagram and mathematical (i.e., quantitative) models used in the trade study process. This section focuses on the last. The main reason for using mathematical models and diagrams in trade studies is to provide estimates of system effectiveness, performance or technical attributes, and cost from a set of known or estimable quantities. Typically, a collection of separate models is needed to provide all of these outcome variables. The heart of any mathematical model is a set of meaningful quantitative relationships among its inputs and outputs. These relationships can be as simple as adding up constituent quantities to obtain a total, or as complex as a set of differential equations describing the trajectory of

a spacecraft in a gravitational field. Ideally, the relationships express causality, not just correlation.

TOOLS FOR GRAPHIC REPRESENTATIONS

Initially, when the primary purpose of a systems engineer is to comprehend a complex problem, graphic representations of a system are used to communicate a system's functional and data requirements. Common graphical representations include:

- Functional Flow Block Diagram (FFBD)
- VisSim
- Data Flow Diagram (DFD)
- N2 (N-Squared) Chart
- IDEF0 Diagram
- UML Use case diagram
- UML Sequence diagram
- USL Function Maps and Type Maps.
- Enterprise Architecture frameworks, like TOGAF, MODAF, Zachman Frameworks etc.

A graphical representation relates the various subsystems or parts of a system through functions, data, or interfaces. Any or each of the above methods are used in an industry based on its requirements. For instance, the N2 chart may be used where interfaces between systems is important. Part of the design phase is to create structural and behavioural models of the system. Once the requirements are understood, it is now the responsibility of a systems engineer to refine them, and to determine, along with other engineers, the best technology for a job. At this point starting

with a trade study, systems engineering encourages the use of weighted choices to determine the best option. A decision matrix, or Pugh method, is one way (QFD is another) to make this choice while considering all criteria that are important. The trade study in turn informs the design which again affects the graphic representations of the system (without changing the requirements). In an SE process, this stage represents the iterative step that is carried out until a feasible solution is found. A decision matrix is often populated using techniques such as statistical analysis, reliability analysis, system dynamics (feedback control), and optimization methods. At times a systems engineer must assess the existence of feasible solutions, and rarely will customer inputs arrive at only one. Some customer requirements will produce no feasible solution. Constraints must be traded to find one or more feasible solutions.

The customers' wants become the most valuable input to such a trade and cannot be assumed. Those wants/ desires may only be discovered by the customer once the customer finds that he has overconstrained the problem. Most commonly, many feasible solutions can be found, and a sufficient set of constraints must be defined to produce an optimal solution. This situation is at times advantageous because one can present an opportunity to improve the design towards one or many ends, such as cost or schedule. Various modeling methods can be used to solve the problem including constraints and a cost function. Systems Modeling Language (SysML), a modeling language used for systems engineering applications, supports the specification, analysis, design, verification and validation of a broad range of complex

systems. Universal Systems Language (USL) is a systems oriented object modeling language with executable (computer independent) semantics for defining complex systems, including software.

RELATED FIELDS AND SUB-FIELDS

Many related fields may be considered tightly coupled to systems engineering. These areas have contributed to the development of systems engineering as a distinct entity.

COGNITIVE SYSTEMS ENGINEERING

Cognitive systems engineering (CSE) is a specific approach to the description and analysis of human-machine systems or sociotechnical systems. The three main themes of CSE are how humans cope with complexity, how work is accomplished by the use of artefacts, and how human-machine systems and socio-technical systems can be described as joint cognitive systems. CSE has since its beginning become a recognised scientific discipline, sometimes also referred to as Cognitive Engineering. The concept of a Joint Cognitive System (JCS) has in particular become widely used as a way of understanding how complex socio-technical systems can be described with varying degrees of resolution. The more than 20 years of experience with CSE has been described extensively.

CONFIGURATION MANAGEMENT

Like systems engineering, Configuration Management as practiced in the defence and aerospace industry is a broad systems-level practice. The field parallels the taskings of

systems engineering; where systems engineering deals with requirements development, allocation to development items and verification, Configuration Management deals with requirements capture, traceability to the development item, and audit of development item to ensure that it has achieved the desired functionality that systems engineering and/or Test and Verification Engineering have proven out through objective testing.

CONTROL ENGINEERING

Control engineering and its design and implementation of control systems, used extensively in nearly every industry, is a large sub-field of systems engineering. The cruise control on an automobile and the guidance system for a ballistic missile are two examples. Control systems theory is an active field of applied mathematics involving the investigation of solution spaces and the development of new methods for the analysis of the control process.

INDUSTRIAL ENGINEERING

Industrial engineering is a branch of engineering that concerns the development, improvement, implementation and evaluation of integrated systems of people, money, knowledge, information, equipment, energy, material and process. Industrial engineering draws upon the principles and methods of engineering analysis and synthesis, as well as mathematical, physical and social sciences together with the principles and methods of engineering analysis and design to specify, predict and evaluate the results to be obtained from such systems.

INTERFACE DESIGN

Interface design and its specification are concerned with assuring that the pieces of a system connect and inter-operate with other parts of the system and with external systems as necessary. Interface design also includes assuring that system interfaces be able to accept new features, including mechanical, electrical and logical interfaces, including reserved wires, plug-space, command codes and bits in communication protocols. This is known as extensibility. Human-Computer Interaction (HCI) or Human-Machine Interface (HMI) is another aspect of interface design, and is a critical aspect of modern systems engineering. Systems engineering principles are applied in the design of network protocols for local-area networks and wide-area networks.

MECHATRONIC ENGINEERING

Mechatronic engineering, like Systems engineering, is a multidisciplinary field of engineering that uses dynamical systems modeling to express tangible constructs. In that regards it is almost indistinguishable from Systems Engineering, but what sets it apart is the focus on smaller details rather than larger generalizations and relationships. As such, both fields are distinguished by the scope of their projects rather than the methodology of their practice.

OPERATIONS RESEARCH

Operations research supports systems engineering. The tools of operations research are used in systems analysis, decision making, and trade studies. Several schools teach

SE courses within the operations research or industrial engineering department, highlighting the role systems engineering plays in complex projects. Operations research, briefly, is concerned with the optimization of a process under multiple constraints.

PERFORMANCE ENGINEERING

Performance engineering is the discipline of ensuring a system will meet the customer's expectations for performance throughout its life. Performance is usually defined as the speed with which a certain operation is executed or the capability of executing a number of such operations in a unit of time. Performance may be degraded when an operations queue to be executed is throttled when the capacity of the system is limited. For example, the performance of a packet-switched network would be characterised by the end-to-end packet transit delay or the number of packets switched within an hour. The design of high-performance systems makes use of analytical or simulation modeling, whereas the delivery of high-performance implementation involves thorough performance testing. Performance engineering relies heavily on statistics, queueing theory and probability theory for its tools and processes.

PROGRAMME MANAGEMENT AND PROJECT MANAGEMENT

Programme management (or programme management) has many similarities with systems engineering, but has broader-based origins than the engineering ones of systems

engineering. Project management is also closely related to both programme management and systems engineering.

PROPOSAL ENGINEERING

Proposal engineering is the application of scientific and mathematical principles to design, construct, and operate a cost-effective proposal development system. Basically, proposal engineering uses the “systems engineering process” to create a cost effective proposal and increase the odds of a successful proposal.

RELIABILITY ENGINEERING

Reliability engineering is the discipline of ensuring a system will meet the customer’s expectations for reliability throughout its life; i.e. it will not fail more frequently than expected. Reliability engineering applies to all aspects of the system. It is closely associated with maintainability, availability and logistics engineering. Reliability engineering is always a critical component of safety engineering, as in failure modes and effects analysis (FMEA) and hazard fault tree analysis, and of security engineering. Reliability engineering relies heavily on statistics, probability theory and reliability theory for its tools and processes.

SAFETY ENGINEERING

The techniques of safety engineering may be applied by non-specialist engineers in designing complex systems to minimize the probability of safety-critical failures. The “System Safety Engineering” function helps to identify “safety hazards” in emerging designs, and may assist with

techniques to “mitigate” the effects of (potentially) hazardous conditions that cannot be designed out of systems.

SECURITY ENGINEERING

Security engineering can be viewed as an interdisciplinary field that integrates the community of practice for control systems design, reliability, safety and systems engineering. It may involve such sub-specialties as authentication of system users, system targets and others: people, objects and processes.

SOFTWARE ENGINEERING

From its beginnings, software engineering has helped shape modern systems engineering practice. The techniques used in the handling of complexes of large software-intensive systems has had a major effect on the shaping and reshaping of the tools, methods and processes of SE.

SYSTEMS DESIGN

Systems design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. One could see it as the application of systems theory to product development. There is some overlap with the disciplines of systems analysis, systems architecture and systems engineering. If the broader topic of product development “blends the perspective of marketing, design, and manufacturing into a single approach to product development, then design is the act of taking the marketing information and creating the design of the product to be manufactured. Systems design is therefore the process

of defining and developing systems to satisfy specified requirements of the user. Until the 1990s systems design had a crucial and respected role in the data processing industry. In the 1990s standardization of hardware and software resulted in the ability to build modular systems. The increasing importance of software running on generic platforms has enhanced the discipline of software engineering. Object-oriented analysis and design methods are becoming the most widely used methods for computer systems design. The UML has become the standard language in object-oriented analysis and design. It is widely used for modeling software systems and is increasingly used for high designing non-software systems and organizations.

LOGICAL DESIGN

The logical design of a system pertains to an abstract representation of the data flows, inputs and outputs of the system. This is often conducted via modelling, using an over-abstract (and sometimes graphical) model of the actual system. In the context of systems design, modelling can undertake the following forms, including

PHYSICAL DESIGN

The physical design relates to the actual input and output processes of the system. This is laid down in terms of how data is input into a system, how it is verified/authenticated, how it is processed, and how it is displayed as output. Physical design, in this context, does not refer to the tangible physical design of an information system. To use an analogy, a personal computer's physical design

involves input via a keyboard, processing within the CPU, and output via a monitor, printer, etc. It would not concern the actual layout of the tangible hardware, which for a PC would be a monitor, CPU, motherboard, hard drive, modems, video/graphics cards, USB slots, etc.

ALTERNATIVE DESIGN METHODOLOGIES

RAPID APPLICATION DEVELOPMENT (RAD)

Rapid application development (RAD) is a methodology in which a systems designer produces prototypes for an end-user. The end-user reviews the prototype, and offers feedback on its suitability. This process is repeated until the end-user is satisfied with the final system.

JOINT APPLICATION DESIGN (JAD)

Joint application design (JAD) is a methodology which evolved from RAD, in which a systems designer consults with a group consisting of the following parties:

- Executive sponsor
- Systems Designer
- Managers of the system

JAD involves a number of stages, in which the group collectively develops an agreed pattern for the design and implementation of the system.

TOPICS OF SYSTEMS DESIGN

- Requirements analysis - analyzes the needs of the end users or customers

- Benchmarking — is an effort to evaluate how current systems are used
- Systems architecture - creates a blueprint for the design with the necessary specifications for the hardware, software, people and data resources. In many cases, multiple architectures are evaluated before one is selected.
- Design — designers will produce one or more ‘models’ of what they see a system eventually looking like, with ideas from the analysis section either used or discarded. A document will be produced with a description of the system, but nothing is specific — they might say ‘touchscreen’ or ‘GUI operating system’, but not mention any specific brands;
- Computer programming and debugging in the software world, or detailed design in the consumer, enterprise or commercial world - specifies the final system components.
- System testing - evaluates the system’s actual functionality in relation to expected or intended functionality, including all integration aspects.

SYSTEM SOFTWARE

System software is computer software designed to operate the computer hardware and to provide a platform for running application software. The most basic types of system software are:

- The computer BIOS and device firmware, which provide basic functionality to operate and control the hardware connected to or built into the computer.

- The operating system (prominent examples being Microsoft Windows, Mac OS X and Linux), which allows the parts of a computer to work together by performing tasks like transferring data between memory and disks or rendering output onto a display device. It also provides a platform to run high-level system software and application software.
- Utility software, which helps to analyze, configure, optimize and maintain the computer.

In some publications, the term *system software* is also used to designate software development tools (like a compiler, linker or debugger). Computer purchasers seldom buy a computer primarily because of its system software (But purchasers of devices like mobile phones because of there system software, as is the case with the iPhone, as the system software of such devices is difficult for the end-user to modify). Rather, system software serves as a useful (even necessary) level of infrastructure code, generally built-in or pre-installed. In contrast to system software, software that allows users to do things like create text documents, play games, listen to music, or surf the web is called application software.

TYPES OF SYSTEM SOFTWARE PROGRAMMES

System software helps use the operating system and computer system. It includes diagnostic tools, compilers, servers, windowing systems, utilities, language translator, data communication programmes, database systems and more. The purpose of system software is to insulate the applications programmer as much as possible from the

complexity and specific details of the particular computer being used, especially memory and other hardware features, and such accessory devices as communications, printers, readers, displays, keyboards, etc. Specific kinds of system software include:

- Loaders
- Linkers
- Utility software
- Desktop environment / Graphical user interface
- Shells
- BIOS
- Hypervisors
- Boot loaders
- Database Management Systems(SQL, NoSQL)

If system software is stored on non-volatile memory such as integrated circuits, it is usually termed firmware.

ENTERPRIZE SOFTWARE

Enterprize software, also known as enterprize application software (EAS), is software used in organizations, such as in a business or government, as opposed to software chosen by individuals (for example, retail software). Enterprize software is an integral part of a (Computer Based) Information System. Services provided by enterprize software are typically business-oriented tools such as online shopping and online payment processing, interactive product catalogue, automated billing systems, security, content management, IT service management, customer relationship management, resource planning, business intelligence, HR management, manufacturing, application integration, and forms automation.

DEFINITIONS

While there is no single, widely accepted list of enterprise software characteristics, this section is intended to summarize definitions from multiple sources. Enterprise software describes a collection of computer programmes with common business applications, tools for modeling how the entire organization works, and development tools for building applications unique to the organization. The software is intended to solve an enterprise-wide problem (rather than a departmental problem) and often written using an Enterprise Software Architecture. Enterprise level software aims to improve the enterprise's productivity and efficiency by providing business logic support functionality. Capterra broadly defines enterprise software in the following manner:

- Targets any type of organization — corporations, partnerships, sole proprietorships, nonprofits, government agencies — but does not directly target consumers.
- Targets any industry.
- Targets both large and small organizations — from Fortune 500 to “mom and pop” businesses.
- Includes function-specific (Accounting, HR, Supply Chain, etc.) and industry-specific (Manufacturing, Retail, Healthcare, etc.) solutions.

Due to the cost of building or buying what is often non-free proprietary software, only large enterprises attempt to implement such enterprise software that models the entire business enterprise and is the core IT system of governing the enterprise and the core of communication within the

enterprise. As business enterprises have similar departments and systems in common, enterprise software is often available as a suite of programmes that have attached enterprise development tools to customize the programmes to the specific enterprise. Generally, these tools are complex enterprise programming tools that require specialist capabilities. Thus, one often sees job listings for a programmer who is required to have specific knowledge of a particular set of enterprise tools, such as “must be an SAP developer”. Characteristics of enterprise software are performance, scalability, and robustness. Enterprise software typically has interfaces to other enterprise software (for example LDAP to directory services) and is centrally managed (a single admin page for example).

ENTERPRISE APPLICATION SOFTWARE

Enterprise application software is application software that performs business functions such as order processing, procurement, production scheduling, customer information management, and accounting. It is typically hosted on servers and provides simultaneous services to a large number of users, typically over a computer network. This is in contrast to a single-user application that is executed on a user’s personal computer and serves only one user at a time.

TYPES

- Enterprise software can be designed and implemented by an information technology (IT) group within a company.
- It may also be purchased from an independent enterprise software developer, that often installs and

maintains the software for their customers. Installation, customization, and maintenance can also be outsourced to an IT consulting company.

- Another model is based on a concept called on-demand software, or Software as a Service (SaaS). The on-demand model of enterprise software is made possible through the widespread distribution of broadband access to the Internet. Software as a Service vendors maintain enterprise software on servers within their own company data center and then provide access to the software to their enterprise customers via the Internet.

Enterprise software is often categorized by the business function that it automates - such as accounting software or sales force automation software. Similarly for industries - for example, there are enterprise systems devised for the health care industry, or for manufacturing enterprises.

DEVELOPERS

Major organizations in the enterprise software field include SAP, IBM, BMC Software, HP Software Division, Redwood Software, UC4 Software, JBoss (Red Hat), Microsoft, Adobe Systems, Oracle Corporation, Inquest Technologies, Computer Associates, and ASG Software Solutions but there are thousands of competing vendors.

CRITICISM

The word *enterprise* can have various connotations. Sometimes the term is used merely as a synonym for *organization*, whether it be very large (e.g., a corporation

with thousands of employees), very small (a sole proprietorship), or an intermediate size. Often the term is used only to refer to very large organizations, although it has become a corporate-speak buzzword and may be heard in other uses. Some enterprize software vendors using the latter definition develop highly complex products that are often overkill for smaller organizations, and the application of these can be a very frustrating task. Thus, sometimes “enterprize” might be used sarcastically to mean overly complex software. The adjective “enterprizey” is sometimes used to make this sarcasm explicit. In this usage, the term “enterprizey” is intended to go beyond the concern of “overkill for smaller organizations” to imply the software is overly complex even for large organizations and simpler solutions are available.

APPLICATION SOFTWARE

Application software, also known as an application or an “app”, is computer software designed to help the user to perform singular or multiple related specific tasks. Examples include enterprize software, accounting software, office suites, graphics software and media players. Many application programmes deal principally with documents. Application software is contrasted with system software and middleware, which manage and integrate a computer’s capabilities, but typically do not directly apply them in the performance of tasks that benefit the user. A simple, if imperfect, analogy in the world of hardware would be the relationship of an electric light bulb (an application) to an electric power generation plant (a system). The power station

merely generates electricity, not itself of any real use until harnessed to an application like the electric light that performs a service that benefits the user. Application software applies the power of a particular computing platform or system software to a particular purpose. Some apps such as Microsoft Office are available in versions for several different platforms; others have narrower requirements.

TERMINOLOGY

In information technology, an application is a computer programme designed to help people perform an activity. An application thus differs from an operating system (which runs a computer), a utility (which performs maintenance or general-purpose chores), and a programming language (with which computer programmes are created). Depending on the activity for which it was designed, an application can manipulate text, numbers, graphics, or a combination of these elements. Some application packages offer considerable computing power by focusing on a single task, such as word processing; others, called integrated software, offer somewhat less power but include several applications. User-written software tailors systems to meet the user's specific needs. User-written software include spreadsheet templates, word processor macros, scientific simulations, graphics and animation scripts. Even email filters are a kind of user software. Users create this software themselves and often overlook how important it is.

The delineation between system software such as operating systems and application software is not exact, however, and is occasionally the object of controversy. For

example, one of the key questions in the United States v. Microsoft antitrust trial was whether Microsoft's Internet Explorer web browser was part of its Windows operating system or a separable piece of application software. As another example, the GNU/Linux naming controversy is, in part, due to disagreement about the relationship between the Linux kernel and the operating systems built over this kernel. In some types of embedded systems, the application software and the operating system software may be indistinguishable to the user, as in the case of software used to control a VCR, DVD player or microwave oven. The above definitions may exclude some applications that may exist on some computers in large organizations. For an alternative definition of an app: see *Application Portfolio Management*.

APPLICATION SOFTWARE CLASSIFICATION

Application software falls into two general categories; horizontal applications and vertical applications. Horizontal Application are the most popular and its widely spread in departments or companies. Vertical Applications are designed for a particular type of business or for specific division in a company. There are many types of application software:

- An *application suite* consists of multiple applications bundled together. They usually have related functions, features and user interfaces, and may be able to interact with each other, e.g. open each other's files. Business applications often come in suites, e.g. Microsoft Office, OpenOffice.org and iWork, which bundle together a word processor, a

spreadsheet, etc.; but suites exist for other purposes, e.g. graphics or music.

- *Enterprize software* addresses the needs of organization processes and data flow, often in a large distributed environment. (Examples include financial systems, customer relationship management (CRM) systems and supply-chain management software). Note that Departmental Software is a sub-type of Enterprize Software with a focus on smaller organizations or groups within a large organization. (Examples include Travel Expense Management and IT Helpdesk)
- *Enterprize infrastructure software* provides common capabilities needed to support enterprize software systems. (Examples include databases, email servers, and systems for managing networks and security.)
- *Information worker software* addresses the needs of individuals to create and manage information, often for individual projects within a department, in contrast to enterprize management. Examples include time management, resource management, documentation tools, analytical, and collaborative. Word processors, spreadsheets, email and blog clients, personal information system, and individual media editors may aid in multiple information worker tasks.
- *Content access software* is software used primarily to access content without editing, but may include software that allows for content editing. Such software addresses the needs of individuals and groups to consume digital entertainment and published digital

content. (Examples include Media Players, Web Browsers, Help browsers and Games)

- *Educational software* is related to content access software, but has the content and/or features adapted for use in by educators or students. For example, it may deliver evaluations (tests), track progress through material, or include collaborative capabilities.
- *Simulation software* are computer software for simulation of physical or abstract systems for either research, training or entertainment purposes.
- *Media development software* addresses the needs of individuals who generate print and electronic media for others to consume, most often in a commercial or educational setting. This includes Graphic Art software, Desktop Publishing software, Multimedia Development software, HTML editors, Digital Animation editors, Digital Audio and Video composition, and many others.
- *Mobile applications* run on hand-held devices such as mobile phones, personal digital assistants and enterprize digital assistants : see mobile application development.
- *Product engineering software* is used in developing hardware and software products. This includes computer aided design (CAD), computer aided engineering (CAE), computer language editing and compiling tools, Integrated Development Environments, and Application Programmer Interfaces.

- A command-driven interface is one in which you type in commands to make the computer do something. You have to know the commands and what they do and they have to be typed correctly. DOS and Unix are examples of command-driven interfaces.
- A graphical user interface (GUI) is one in which you select command choices from various menus, buttons and icons using a mouse. It is a user-friendly interface. The Windows and Mac OS are both graphical user interfaces.

8

Sketchpad and Design Process

Construction of a drawing with Sketchpad is *itself* a model of the design process. The locations of the points and lines of the drawing model the variables of a design, and the geometric constraints applied to the points and lines of the drawing model the design constraints which limit the values of design variables. The ability of Sketchpad to satisfy the geometric constraints applied to the parts of a drawing models the ability of a good designer to satisfy all the design conditions imposed by the limitations of his materials, cost, *etc.* In fact, since, designers in many fields produce nothing themselves but a drawing of a part, design conditions may well be thought of as applying to the drawing of a part rather than to the part itself. When such design conditions are added to Sketchpad's vocabulary of constraints, the computer will be able to assist a user not only in arriving at a nice looking drawing, but also in arriving at a sound design.

PRESENT USEFULNESS

As more and more applications have been made, it has become clear that the properties of Sketchpad drawings make them most useful in four broad areas:

For storing and updating drawings: Each time a drawing is made, a description of that drawing is stored in the computer in a form that is readily transferred to magnetic tape. A library of drawings will thus develop, parts of which may be used in other drawings at only a fraction of the investment of time that was put into the original drawing.

For gaining scientific or engineering understanding of operations that can be described graphically: A drawing in the Sketchpad system may contain explicit statements about the relations between its parts so that as one part is changed the implications of this change become evident throughout the drawing. For instance, Sketchpad makes it easy to study mechanical linkages, observing the path of some parts when others are moved.

As a topological input device for circuit simulators, etc.: Since, the storage structure of Sketchpad reflects the topology of any circuit or diagram, it can serve as an input for many network or circuit simulating Programmes. The additional effort required to draw a circuit completely from scratch with the Sketchpad system may well be recompensed if the properties of the circuit are obtainable through simulation of the circuit drawn.

For highly repetitive drawings: The ability of the computer to reproduce any drawn symbol anywhere at the press of a button, and to recursively include subpictures within subpictures makes it easy to produce drawings which are composed of huge numbers of parts all similar in shape.

RING STRUCTURE

The basic n -component element structure described by Ross has been somewhat expanded in the implementation of Sketchpad so that all references made to a particular n -component element or block are collected together by a string of pointers which originates within that block. For example, not only may the end points of a line segment be found by following pointers in the line block (n -component element), but also all the line segments which terminate on a particular point may be found by following a string of pointers which starts within the point block. This string of pointers closes on itself; the last pointer points back to the first, hence the name “ring.” The ring points both ways to make it easy to find both the next and the previous member of the ring in case, as when deleting, some change must be made to them.

BASIC OPERATIONS

The basic ring structure operations are:

- Inserting a new member into a ring at some specified location on it, usually first or last.
- Removing a member from a ring.
- Putting all the members of one ring, in order, into another at some specified location in it, usually first or last.
- Performing some auxiliary operation on each member of a ring in either forward or reverse order.

These basic ring structure operations are implemented by short sections of Programme defined as MACRO instructions in the compiler language. By suitable treatment of zero and one member rings, the basic Programmes operate without

making special cases. Subroutines are used for setting up new n -component elements in free spaces in the storage structure. As parts of the drawing are deleted, the registers which were used to represent them become free. New components are set up at the end of the storage area, lengthening it, while free blocks are allowed to accumulate. Garbage collection periodically compacts the storage structure by removal of the free blocks.

GENERIC STRUCTURE, HIERARCHIES

The main part of Sketchpad can perform basic operations on any drawing part, calling for help from routines specific to particular types of parts when that is necessary. For example, the main Programme can show any part on the display system by calling the appropriate display subroutine. The big power of the clear-cut separation of the general and the specific is that it is easy to change the details of specific parts of the Programme to get quite different results without any need to change the general parts.

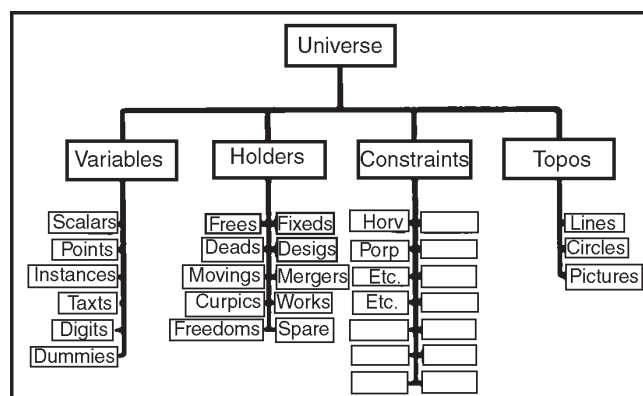


Figure : Generic Structure. The n -component Elements for each Point or line, etc. are Collected under the Generic Blocks "Lines," "Points," etc. Shown.

In the data storage structure the separation of general and specific is accomplished by collecting all things of one type together in a ring under a generic heading. The generic heading contains all the information which makes this type of thing different from all other types of things. Thus the data storage structure itself contains all the specific information. The generic blocks are further gathered together under super-generic or generic-generic blocks.

EXPANDING SKETCHPAD

Addition of new types of things to the Sketchpad system's vocabulary of picture parts requires only the construction of a new generic block (about 20 registers) and the writing of appropriate subroutines for the new type. The subroutines might be easy to write, as they usually are for new constraints, or difficult to write, as for adding ellipse capability, but at least a finite, well-defined task faces one to add a new ability to the system. Without a generic structure it would be almost impossible to add the instructions required to handle a new type of element.

LIGHT PEN

In Sketchpad the light pen is time shared between the functions of coordinate input for positioning picture parts on the drawing and demonstrative input for pointing to existing picture parts to make changes. Although almost any kind of coordinate input device could be used instead of the light pen for positioning, the demonstrative input uses the light pen optics as a sort of analog computer to remove from consideration all but a very few picture parts which happen to fall within its field of view, saving considerable Programme

time. Drawing systems using storage display devices of the Memotron type may not be practical because of the loss of this analog computation feature.

PEN TRACKING

To initially establish pen tracking, the Sketchpad user must inform the computer of an initial pen location. This has come to be known as “inking-up” and is done by “touching” any existing line or spot on the display, whereupon the tracking cross appears. If no picture has yet been drawn, the letters INK are always displayed for this purpose. Sketchpad uses loss of tracking as a “termination signal” to stop drawing. The user signals that he is finished drawing by flicking the pen too fast for the tracking Programme to follow.

DEMONSTRATIVE USE OF PEN

During the 90% of the time that the light pen and display system are free from the tracking chore, spots are very rapidly displayed to exhibit the drawing being built, and thus the lines and circles of the drawing appear. The light pen is sensitive to these spots and reports any which fall within its field of view.

The one-half inch diameter field of view of the light pen, although well suited to tracking, is relatively large for pointing. Therefore, the Sketchpad system will reject any seen part which is further from the centre of the light pen than some small minimum distance; about 1/8 inch was found to be suitable. For every kind of picture part some method must be provided for computing its distance from the light pen centre or indicating that this computation cannot be made.

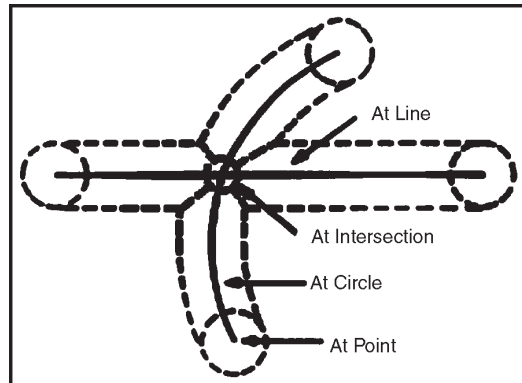


Figure : Areas in Which Pen must lie to “aim at” Existing Drawing Parts (solid lines).

After eliminating all parts seen by the pen which lie outside the smaller effective field of view, the Sketchpad system considers objects topologically related to the ones actually seen. End points of lines and attachment points of instances (subpictures) are especially important. One can thus aim at the end point of a line even though only the line is displayed. The various regions within which the pen must lie to be considered aimed at a line segment, a circle arc, their end points, or their intersection.

PSEUDO PEN LOCATION

When the light pen is aimed at a picture part, the exact location of the light pen is ignored in favour of a “pseudo pen location” exactly on the part aimed at. If no object is aimed at, the pseudo pen location is taken to be the actual pen location.

The pseudo pen location is displayed as a bright dot which is used as the “point of the pencil” in all drawing operations. As the light pen is moved into the areas outline the dot will lock onto the existing parts of the drawing, and any moving picture parts will jump to their new locations as the pseudo

pen location moves to lie on the appropriate picture part. With just the basic drawing creation and manipulation functions of “draw,” “move,” and “delete,” and the power of the pseudo pen location and demonstrative language Programmes, it is possible to make fairly extensive drawings. Most of the constructions normally provided by straight edge and compass are available in highly accurate form. Most important, however, the pseudo pen location and demonstrative language give the means for entering the topological properties of a drawing into the machine.

DISPLAY GENERATION

The display system, or “scope,” on the TX-2 is a ten bit per axis electrostatic deflection system able to display spots at a maximum rate of about 100,000 per second. The coordinates of the spots which are to be seen on the display are stored in a large table so that computation and display may proceed independently. If, instead of displaying each spot successively, the display Programme displays them in a random order or with interlace, the flicker of the display is reduced greatly.

MARKING OF DISPLAY FILE

Of the 36 bits available to store each display spot in the display file, 20 give the coordinates of that spot for the display system, and the remaining 16 give the address of the n -component element which is responsible for adding that spot to the display. Thus, all the spots in a line are tagged with the ring structure address of that line, and all the spots in an instance (sub-picture) are tagged as belonging to that

instance. The tags are used to identify the particular part of the drawing being aimed at by the light pen. If a part of the drawing is being moved by the light pen, its display spots will be recomputed as quickly as possible to show it in successive positions. The display spots for such moving parts are stored at the end of the display file so that the display of the many non--moving parts need not be disturbed. Moving parts are made invisible to the light pen.

MAGNIFICATION OF PICTURES

The shaft position encoder knobs are used to tell the Programme to change the display scale factor or the portion of the page displayed. The range of magnification of 2000 available makes it possible to work, in effect, on a 7-inch square portion of a drawing about $\frac{1}{4}$ mile on a side. For a magnified picture, Sketchpad computes which portion(s) of a curve will appear on the display and generates display spots for those portions only. The “edge detection” problem is the problem of finding suitable end points for the portion of a curve which appears on the display. In concept the edge detection problem is trivial. In terms of Programme time for lines and circles the problem is a small fraction of the total computational load of the system, but in terms of Programme logical complexity the edge detection problem is a difficult one. For example, the computation of the intersection of a circle with any of the edges of the scope is easy, but computation of the intersection of a circle with all four edges may result in as many as eight intersections, some pairs of which may be identical, the scope corners. Now which of these intersections are actually to be used as starts of circle arcs?

LINE AND CIRCLE GENERATION

All of Sketchpad's displays are generated from straight line segments, circle arcs, and single points.

The generation of the lines and circles is accomplished by means of the difference equations:

$$x_i = x_{i-1} + \Delta x \quad y_i = y_{i-1} + \Delta y \quad (1)$$

for lines, and

$$\begin{aligned} x_i &= x_{i-2} + \frac{2}{R}(y_{i-1} - y_c) \\ y_i &= y_{i-2} - \frac{2}{R}(x_{i-1} - x_c) \end{aligned} \quad (2)$$

for circles, where subscripts i indicate successive display spots, subscript c indicates the circle centre, and R is the radius of the circle in Scope Units. In implementing these difference equations in the Programme, the fullest possible use is made of the coordinate arithmetic capability of the TX-2 so that both the x and y equation computations are performed in parallel on 18 bit subwords. Even so, about $\frac{3}{4}$ of the total Sketchpad computation time is spent in line and circle generation. A vector and circle generating display would materially reduce the computational load of Sketchpad.

For computers which do only one addition at a time, the difference equations:

$$\begin{aligned} x_i &= x_{i-1} + \frac{2}{R}(y_{i-1} - y_c) \\ y_i &= y_{i-1} - \frac{2}{R}(x_i - x_c) \end{aligned} \quad (3)$$

should be used to generate circles. Equations (3) approximate a circle well enough and are known to close exactly both in theory and when implemented, because the x and y equations are dissimilar.

DIGITS AND TEXT

Text, to put legends on a drawing, is displayed by means of special tables which indicate the locations of line and circle segments to make up the letters and numbers. Each piece of text appears as a single line of not more than 36 equally spaced characters which can be changed by typing. Digits to display the value of an indicated scalar at any position and in any size and rotation are formed from the same type face as text. It is possible to display up to five decimal digits with sign; binary to decimal conversion is provided, and leading zeros are suppressed.

Subpictures, whose use was seen in the introductory example above, are each represented in storage as a single n -component element. A subpicture is said to be an “instance” of its “master picture.” To display an instance, all of the lines, text, *etc.* of its master picture must be shown in miniature on the display. The instance display Programme makes use of the line, circle, number, and text display Programmes and *itself* to expand the internal structure of the instance.

DISPLAY OF ABSTRACTIONS

The usual picture for human consumption displays only lines, circles, text, digits, and instances. However, certain very useful abstractions which give the drawing the properties desired by the user are represented in the ring structure storage. For example, the fact that the start and end points of a circle arc should be equidistant from the circle’s centre point is represented in storage by a “constraint” block. To make it possible for a user to manipulate these abstractions, each abstraction must be able to be seen on the display if

desired. Not only does displaying abstractions make it possible for the human user to know that they exist, but also makes it possible for him to aim at them with the light pen and, for example, erase them. To avoid confusion, the display for particular types of objects may be turned on or off selectively by toggle switches. Thus, for example, one can turn on display of constraints as well as or instead of the lines and circles which are normally seen.

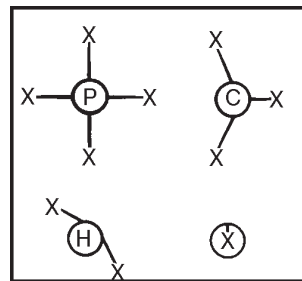


Figure : Display of Constraints.

If their selection toggle switch is on, constraints are displayed. The central circle and code letter are located at the average location of the variables constrained. The four arms of a constraint extend from the top, right side, bottom, and left side of the circle to the first, second, third, and fourth variables constrained, respectively. If fewer than four variables are constrained, excess arms are omitted. The constraints are shown applied to “dummy variables” each of which shows as an X.

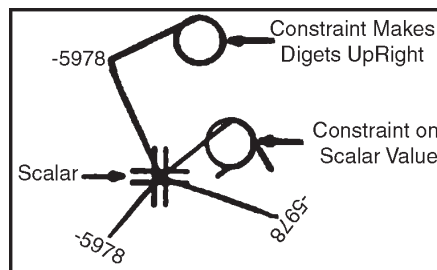


Figure : Three Sets of Digits Displaying the Same Scalar Value.

Another abstraction that can be displayed if desired is the value of a set of digits. For example, three sets of digits all displaying the same scalar value, -5978. The digits themselves may be moved, rotated, or changed in size, without changing the value displayed. If we wish to change the value, we point at its abstract display. The three sets of digits, all display the same value, as indicated by the lines connecting them to the #; changing this value would make all three sets of digits change. Constraints may be applied independently to either the position of the digits or their value as indicated by the two constraints in the figure.

RECURSIVE FUNCTIONS

In the process of making the Sketchpad system operate, a few very general functions were developed which make no reference at all to the specific types of entities on which they operate. These general functions give the Sketchpad system the ability to operate on a wide range of problems. The motivation for making the functions as general as possible came from the desire to get as much result as possible from the programming effort involved. For example, the general function for expanding instances makes it possible for Sketchpad to handle any fixed geometry subpicture. The power obtained from the small set of generalized functions in Sketchpad is one of the most important results of the research.

In order of historical development, the recursive functions in use in the Sketchpad system are:

- Expansion of instances, making it possible to have subpictures within subpictures to as many levels as desired.

- Recursive deletion, whereby removal of certain picture parts will remove other picture parts in order to maintain consistency in the ring structure.
- Recursive merging, whereby combination of two similar picture parts forces combination of similarly related other picture parts, making possible application of complex definitions to an object picture.

RECURSIVE DELETING

If a thing upon which other things depend is deleted, the dependent things must be deleted also. For example, if a point is to be deleted, all lines which terminate on the point must also be deleted. Otherwise, since, the n -component elements for lines contain no positional information, where would these lines end? Similarly, deletion of a variable requires deletion of all constraints on that variable; a constraint must have variables to act on.

RECURSIVE MERGING

If two things of the same type which are independent are merged, a single thing of that type results, and all things which depended on either of the merged things depend on the result of the merger.* For example, if two points are merged, all lines which previously terminated on either point now terminate on the single resulting point. In Sketchpad, if a thing is being moved with the light pen and the termination flick of the pen is given while aiming at another thing of the same type, the two things will merge. Thus, if one moves a point to another point and terminates, the points will merge, connecting all lines which formerly terminated on either. This makes it possible to draw closed polygons.

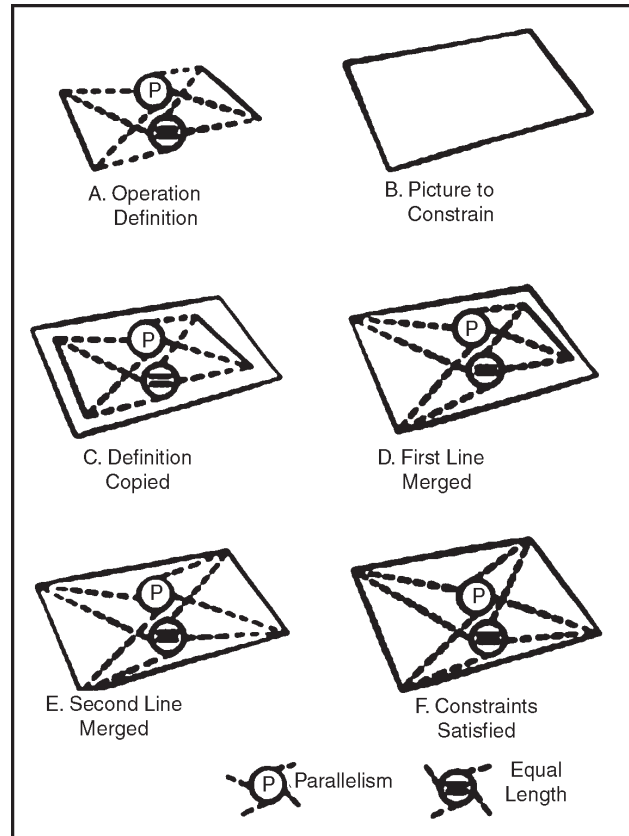


Figure : Applying a Two-constraint Definition to turn a Quadrilateral into a Parallelogram.

If two things of the same type which do depend on other things are merged, the things depended on by one will be forced to merge, respectively, with the things depended on by the other. The result of merging two dependent things depends, respectively, on the results* of the mergers it forces. For example, if two lines are merged, the resultant line must refer to only two end points, the results of merging the pairs of end points of the original lines. All lines which terminated on any of the four original end points now terminate on the appropriate one of the remaining pair. More important and useful, all constraints which applied to any of the four original end points now apply to the appropriate one of the remaining*

pair. This makes it possible to speak of line segments as being parallel even though (because line segments contain no numerical information to be constrained) the parallelism constraint must apply to their end points and not to the line segments themselves. If we wish to make two lines both parallel and equal in length, the steps outlined. More obscure relationships between dependent things may be easily defined and applied. For example, constraint complexes can be defined to make line segments be collinear, to make a line be tangent to a circle, or to make the values represented by two sets of digits be equal. The “result” of a merger is a single thing of the same type as the merged things.

RECURSIVE DISPLAY OF INSTANCES

The block of registers which represents an instance is remarkably small considering that it may generate a display of any complexity. For the purposes of display, the instance block makes reference to its master picture. The instance will appear on the display as a figure geometrically similar to its master picture at a location, size, and rotation indicated by the four numbers which constitute the “value” of the instance. The value of an instance is considered numerically as a four dimensional vector. The components of this vector are the coordinates of the centre of the instance and its actual size as it appears on the drawing times the sine and cosine of the rotation angle involved. In displaying an instance of a picture, reference is made to the master picture to find out what picture parts are to be shown. The master picture referred to may contain instances, however, requiring further reference, and so on until a picture is found which contains

no instances. At each stage in the recursion, any picture parts displayed must be relocated so that they will appear at the correct position, size and rotation on the display. Thus, at each stage of the recursion, some transformation is applied to all picture parts before displaying them. If an instance is encountered, the transformation represented by its value must be adjoined to the existing transformation for display of parts within it. When the expansion of an instance within an instance is finished, the transformation must be restored for continuation at the higher level.

ATTACHERS AND INSTANCES

Many symbols must be integrated into the rest of the drawing by attaching lines to the symbols at appropriate points, or by attaching the symbols directly to each other. For example, circuit symbols must be wired up, geometric patterns made by fitting shapes together, or mechanisms composed of links tied together appropriately. An instance may have any number of attachment points, and a point may serve as attacher for any number of instances. The light pen has the same affinity for the attachers of an instance that it has for the end point of a line.

An “instance-point” constraint, shown with code T, is used to relate an instance to each of its attachment points. An instance-point constraint is satisfied only when the point bears the same relationship to the instance that a master point in the master picture for that instance bears to the master picture coordinate system.

Any point may be an attacher of an instance, but the point must be designated as an attacher in the master drawing of

the instance. For example, when one first draws a resistor, the ends of the resistor must be designated as attachers if wiring is to be attached to instances of it. At each level of building complex pictures, the attachers must be designated anew. Thus of the three attachers of a transistor it is possible to select one or two to be the attachers of a flip-flop.

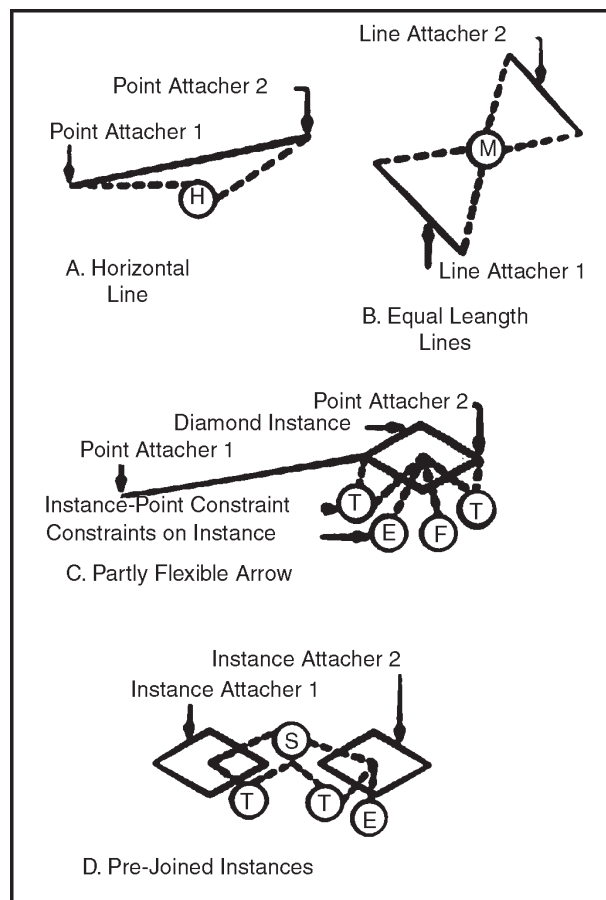


Figure : Definition Pictures to be Copied.

BUILDING A DRAWING, THE COPY FUNCTION

At the start of the Sketchpad effort certain *ad hoc* drawing functions were programmed as the atomic operations of the

system. Each such operation, controlled by a push button, creates in the ring structure a specific set of new drawing parts. For example, the “draw” button creates a line segment and two new end points (unless the light pen happens to be aimed at a point in which case only one new point need be created). Similarly, there are atomic operations for drawing circles, applying a horizontal or vertical constraint to the end points of a line aimed at, and for adding a “point-on-line” constraint whenever a point is moved onto a line and left there. The atomic operations make it possible to create in the ring structure new picture components and relate them topologically. The atomic operations are, of course, limited to creating points, lines, circles, and two or three types of constraints. Since, implementation of the copy function it has become possible to create in the ring structure any predefined combination of picture parts and constraints at the press of a button. The recursive merging function makes it possible to relate the copied set of picture parts to any existing parts. For example, if a line segment and its two end points are copied into the object picture, the action of the “draw” button may be exactly duplicated in every respect. Along with the copied line, however, one might copy as well a constraint, Code H, to make the line horizontal, or two constraints to make the line both horizontal and three inches long, or any other variation one cares to put into the ring structure to be copied.

When one draws a definition picture to be copied, certain portions of it to be used in relating it to other object picture parts are designated as “attachers.” Anything at all may be designated: for example, points, lines, circles, text, even constraints!.

The rules used for combining points when the “draw” button is pressed are generalized so that:

For copying a picture, the last-designated attacher is left moving with the light pen. The next-to-last-designated attacher is recursively merged with whatever object the pen is aimed at when the copying occurs, if that object is of like type. Previously designated attachers are recursively merged with previously designated object picture parts, if of like type, until either the supply of designated attachers or the supply of designated object picture parts is exhausted. The last-designated attacher may be recursively merged with any other object of like type when the termination flick is given. Normally only two designated attachers are used because it is hard to keep track of additional ones.

If the definition picture consists of two line segments, their four end points, and a constraint, Code M, on the points which makes the lines equal in length, with the two lines designated as attachers, copying enables the user to make any two lines equal in length. If the pen is aimed at a line when “copy” is pushed, the first of the two copied lines merges with it (taking its position and never actually being seen). The other copied line is left moving with the light pen and will merge with whatever other line the pen is aimed at when termination occurs. Since, merging is recursive, the copied equal-length constraint, Code M, will apply to the end points of the desired pair of object picture lines.

COPYING INSTANCES

The internal structure of an instance is entirely fixed. The internal structure of a copy, however, is entirely variable. An instance always retains its identity as a single part of the

drawing; one can only delete an entire instance. Once a definition picture is copied, however, the copy loses all identity as a unit; individual parts of it may be deleted at will. One might expect that there was intermediate ground between the fixed-internal-structure instance and the loose-internal-structure copy. One might wish to produce a collection of picture parts, some of which were fixed internally and some of which were not. *The entire range of variation between the instance and the copy can be constructed by copying instances.*

For example, the arrow can be copied into an object picture to result in a fixed-internal-structure diamond arrowhead with a flexible tail. As the definition is set up, drawing diamond-arrowheaded lines is just like drawing ordinary lines. One aims the light pen where the tail is to end, presses “copy,” and moves off with an arrowhead following the pen. The diamond arrowhead in this case will not rotate (constraint Code E), and will not change size (constraint Code F).

Copying pre-joined instances can produce vast numbers of joined instances very easily. For example, the definition, when repetitively copied, will result in a row of joined, equal size (constraint Code S) diamonds. In this case the instances themselves are attachers. Although each press of the “copy” button copies two new instances into the object picture, one of these is merged with the last instance in the growing row. In the final row, therefore, each instance carries all constraints which are applied to either of the instances in the definition. This is why only one of the instances carries the erect constraint, Code E.

CONSTRAINT SATISFACTION

The major feature which distinguishes a Sketchpad drawing from a paper and pencil drawing is the user's ability to specify to Sketchpad mathematical conditions on already drawn parts of his drawing which will be automatically satisfied by the computer to make the drawing take the exact shape desired. The process of fixing up a drawing to meet new conditions applied to it after it is already partially complete is very much like the process a designer goes through in turning a basic idea into a finished design. As new requirements on the various parts of the design are thought of, small changes are made to the size or other properties of parts to meet the new conditions. By making Sketchpad able to find new values for variables which satisfy the conditions imposed, it is hoped that designers can be relieved of the need of much mathematical detail. The effort expended in making the definition of constraint types as general as possible was aimed at making design constraints as well as geometric constraints equally easy to add to the system.

DEFINITION OF A CONSTRAINT TYPE

Each constraint type is entered into the system as a generic block indicating the various properties of that particular constraint type. The generic block tells how many variables are constrained, which of these variables may be changed in order to satisfy the constraint, how many degrees of freedom are removed from the constrained variables, and a code letter for human reference to this constraint type.

The definition of what a constraint type does is a subroutine which will compute, for the existing values of the variables

of a particular constraint of that type, the error introduced into the system by that particular constraint. For example, the defining subroutine for making points have the same x coordinate (to make a line between them vertical) computes the difference in their x coordinates. What could be simpler? The computed error is a scalar which the constraint satisfaction routine will attempt to reduce to zero by manipulation of the constrained variables. The computation of the error may be non-linear or time dependent, or it may involve parameters not a part of the drawing such as the setting of toggle switches, *etc.*

When the one pass method of satisfying constraints to be described later on fails, the Sketchpad system falls back on the reliable but slow method of relaxation to reduce the errors indicated by various computation subroutines to smaller and smaller values. For simple constructions such as the hexagon illustrated, the relaxation procedure is sufficiently fast to be useful. However, for complex systems of variables, especially directly connected instances, relaxation is unacceptably slow. Fortunately it is for just such directly connected instances that the one pass method shows the most striking success.

ONE PASS METHOD

Sketchpad can often find an order in which the variables of a drawing may be re-evaluated to completely satisfy all the conditions on them in just one pass. For the cases in which the one pass method works, it is far better than relaxation: it gives correct answers at once; relaxation may not give a correct solution in any finite time. Sketchpad can find an order in which to re-evaluate the variables of a drawing

for most of the common geometric constructions. Ordering is also found easily for the mechanical linkages. Ordering cannot be found for the bridge truss problem.

The way in which the one pass method works is simple in principle and was easy to implement as soon as the nuances of the ring structure manipulations were understood. To visualize the one pass method, consider the variables of the drawing as places and the constraints relating variables as passages through which one might pass from one variable to another. Variables are adjacent to each other in the maze formed by the constraints if there is a single constraint which constrains them both. Variables are totally unrelated if there is no path through the constraints by which to pass from one to the other.

Suppose that some variable can be found which has so few constraints applying to it that it can be re-evaluated to completely satisfy all of them. Such a variable we shall call a “free” variable. As soon as a variable is recognized as free, the constraints which apply to it are removed from further consideration, because the free variable can be used to satisfy them. Removing these constraints, however, may make adjacent variables free. Recognition of these new variables as free removes further constraints from consideration and may make other adjacent variables free, and so on throughout the maze of constraints. The manner in which freedom spreads is much like the method used in Moore’s algorithm to find the shortest path through a maze. Having found that a collection of variables is free, Sketchpad will re-evaluate them in reverse order, saving the first-found free variable until last. In re-evaluating any particular variable, Sketchpad

uses only those constraints which were present when that variable was found to be free.

EXAMPLES AND CONCLUSIONS

The library tape and thus serve to illustrate not only how the Sketchpad system can be used, but also how it actually has been used so far. We conclude from these examples that Sketchpad drawings can bring invaluable understanding to a user.

For drawings where motion of the drawing, or analysis of a drawn problem is of value to the user, Sketchpad excels. For highly repetitive drawings or drawings where accuracy is required, Sketchpad is sufficiently faster than conventional techniques to be worthwhile. For drawings which merely communicate with shops, it is probably better to use conventional paper and pencil.

PATTERNS

The instance facility enables one to draw any symbol and duplicate its appearance anywhere on an object drawing at the push of a button. This facility made the hexagonal pattern we saw. It took about one half hour to generate 900 hexagons, including the time taken to figure out how to do it. Plotting them takes about 25 minutes. The drafting department estimated it would take two days to produce a similar pattern.

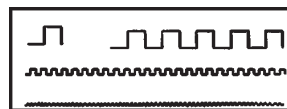


Figure : Zig-Zag for Delay Line.

The instance facility also made it easy to produce long lengths of the zig-zag pattern. A single “zig” was duplicated

in multiples of five and three, *etc.* Five hundred zigs were generated in a single row. Four such rows were plotted one-half inch apart to be used for producing a printed circuit delay line. Total time taken was about 45 minutes for constructing the figure and about 15 minutes to plot it.

A somewhat less repetitive pattern to be used for encoding the time in a digital clock. Each cross in the figure marks the position of a hole. The holes are placed so that a binary coded decimal (BCD) number will indicate the time. Total time for placing crosses was 20 minutes, most of which was spent trying to interpret a pencil sketch of their positions.

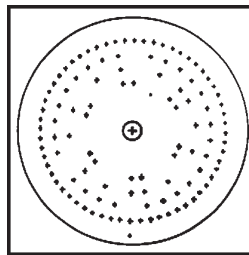


Figure : Binary Coded Decimal Encoder for Clock. Encoder was Plotted Exactly 12 Inches in Diameter for Direct use as a Layout.

LINKAGES

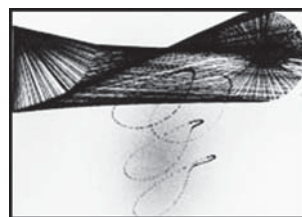


Figure : Three Bar Linkage. The Paths of Four Points on the Central Link are Traced. This is a 15 Second time Exposure of a Moving Sketchpad Drawing.

By far the most interesting application of Sketchpad so far has been drawing and moving linkages. The ability to draw and then move linkages opens up a new field of graphical

manipulation that has never before been available. It is remarkable how even a simple linkage can generate complex motions. For example, only three moving parts. In this linkage a central “ link is suspended between two links of different lengths. As the shorter link rotates, the longer one oscillates as can be seen in the multiple exposure. The motion of four points on the upright part of the “ may be seen.

To make the three bar linkage, an instance shaped like the “ was drawn and given 6 attachers, two at its joints with the other links and four at the places whose paths were to be observed. Connecting the “ shaped subpicture onto a linkage composed of three lines with fixed length created the picture shown. The driving link was rotated by turning a knob below the scope. Total time to construct the linkage was less than 5 minutes, but over an hour was spent playing with it.

A linkage that would be difficult to build physically. This linkage is based on the complete quadrilateral. The three circled points and the two lines which extend out of the top of the picture to the right and left are fixed. Two moving lines are drawn from the lower circled points to the intersections of the long fixed lines with the driving lever. The intersection of these two moving lines (one must be extended) has a box around it. It can be shown theoretically that this linkage produces a conic section which passes through the place labeled “point on curve” and is tangent to the two lines marked “tangent.” A time exposure of the moving point in many positions. At first, this linkage was drawn and working in 15 minutes. Since, then we have rebuilt it time and again until now we can produce it from scratch in about 3 minutes.

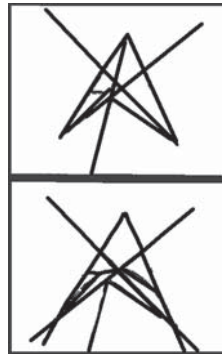


Figure : Conic Drawing Linkage. As the “Driving Lever” is Moved, the Point shown with a box Around it (in A) traces a Conic Section. This Conic can be Seen in the Time Exposure (B).

DIMENSION LINES

To make it possible to have an absolute scale in drawings, a constraint is provided which forces the value displayed by a set of digits to indicate the distance between two points on the drawing. This distance-indicating constraint is used to make the number in a dimension line correspond to its length. Putting in a dimension line is as easy as drawing any other line. One points to where one end is to be left, copies the definition of the dimension line by pressing the “copy” button, and then moves the light pen to where the other end of the dimension line is to be. The first dimension line took about 15 minutes to construct, but that need never be repeated since, it is a part of the library.

BRIDGES

One of the largest untapped fields for application of Sketchpad is as an input Programme for other computation Programmes. The ability to place lines and circles graphically, when coupled with the ability to get accurately computed results pictorially displayed, should bring about a revolution

in computer application. By using Sketchpad's relaxation procedure we were to demonstrate analysis of the force distribution in the members of a pin connected truss.

A bridge is first drawn with enough constraints to make it geometrically accurate. These constraints are then deleted and each member is made to behave like a bridge beam. A bridge beam is constrained to maintain constant length, but any change in length is indicated by an associated number. Under the assumption that each bridge beam has a cross-sectional area proportional to its length, the numbers represent the forces in the beams. The basic bridge beam definition (consisting of two constraints and a number) may be copied and applied to any desired line in a bridge picture by pointing to the line and pressing the "copy" button.

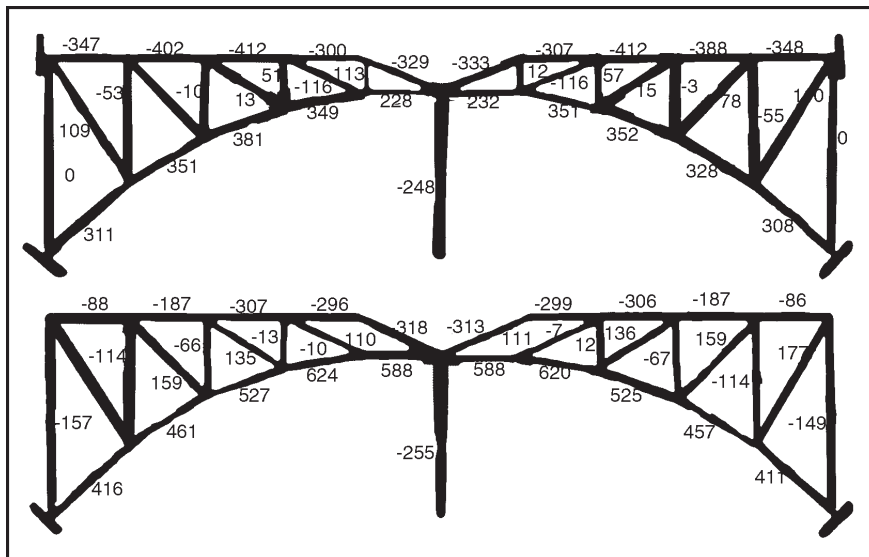


Figure : Cantilever and Arch Bridges. The Numbers Indicate the Forces in the Various Members as Computed by Sketchpad. Central load is not Exactly Vertical.

Having drawn a basic bridge shape, one can experiment with various loading conditions and supports effect of making

minor modifications is. For example, an arch bridge supported both as a three-hinged arch (two supports) and as a cantilever (four supports). For nearly identical loading conditions the distribution of forces is markedly different in these two cases.

ARTISTIC DRAWINGS

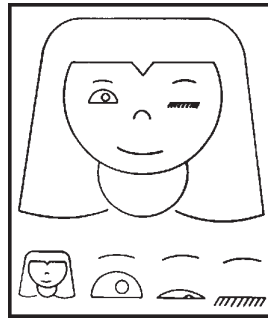


Figure : Winking girl, “Nefertite,” and her Component Parts.

Sketchpad need not be applied exclusively to engineering drawings. For example, the girl “Nefertite” can be made to wink by changing which of the three types of eyes is placed in position on her otherwise eyeless face. In the same way that linkages can be made to move, a stick figure could be made to pedal a bicycle or Nefertite’s hair could be made to swing. The ability to make moving drawings suggests that Sketchpad might be used for making animated cartoons.

ELECTRICAL CIRCUIT DIAGRAMS

Unfortunately, electrical circuits require a great many symbols which have not yet been drawn properly with Sketchpad and therefore are not in the library. After some time is spent working on the basic electrical symbols it may be easier to draw circuits. So far, however, circuit drawing has proven difficult.

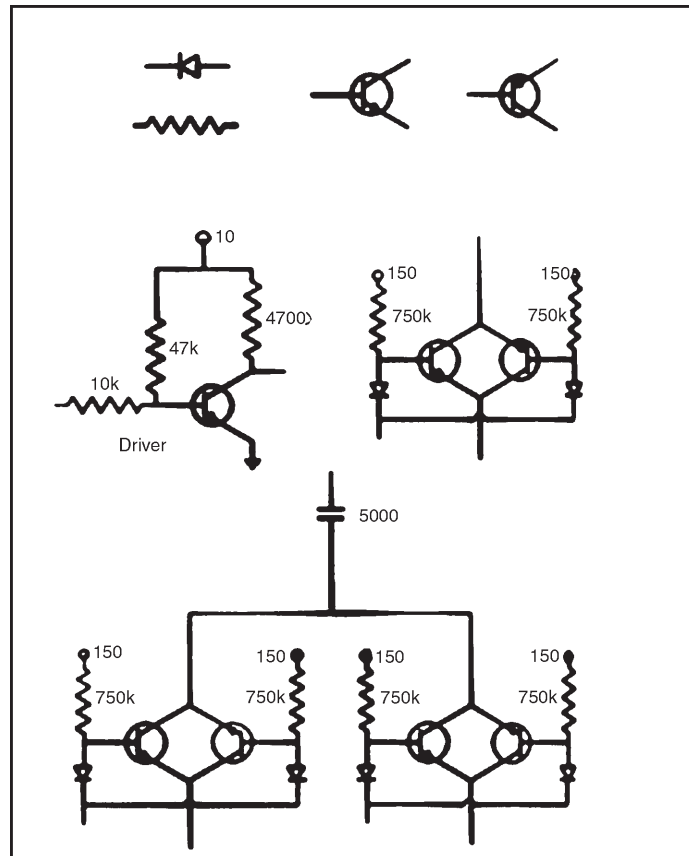


Figure : Circuit Diagram. These are parts of the Large Circuit Mentioned in the Text.

The circuits are parts of an analog switching scheme. You can see in the figure that the more complicated circuits are made up of simpler symbols and circuits. It is very difficult, however, to plan far enough ahead to know what composites of circuit symbols will be useful as subpictures of the final circuit. The simple circuits were compounded into a big circuit involving about 40 transistors. Including much trial and error, the time taken by a new user (for the big circuit not shown) was ten hours. At the end of that time the circuit was still not complete in every detail and he decided it would be better to draw it by hand after all.

CONCLUSIONS

The circuit experience points out the most important fact about Sketchpad drawings. It is only worthwhile to make drawings on the computer if you get something more out of the drawing than just a drawing. In the repetitive patterns we saw in the first examples, precision and ease of constructing great numbers of parts were valuable. In the linkage examples, we were able to gain an understanding of the behaviour of a linkage as well as its appearance. In the bridge examples we got design answers which were worth far more than the computer time put into them. If we had a circuit simulation Programme connected to Sketchpad so that we would have known whether the circuit we drew worked, it would have been worth our while to use the computer to draw it. We are as yet a long way from being able to produce routine drawings economically with the computer.