Computer science *(algorithms, data structures, software engineering, …)*
Mathematics *(geometry, numerical, …)*
Physics *(Optics, mechanics, …)*
Psychology *(Color, perception)*
maging *= representing 2D images*
Modeling *= representing 3D objects*
Rendering *= constructing 2D images from 3D models*
Animation *= simulating changes over time*
Object space: *Image represented as geometric features*
Image Space: *image presented on paper and Continuous*

Image Processing - analysis or reconstruction of objects from image data. Basically, this is the inverse of computer graphics in that it starts with the image and works from there. *Image processing* applies directly to the pixel grid and includes operations such as color correction, scaling, blurring, sharpening, etc.
Computer Generated Imagery (CGI) - Production of imagery using computers. Includes both computer graphics and image processing.

Representations in graphics

Vector and Raster Data

Computer images are classified into two general types: those defined as a *pixel map* and those defined as one or more *vector commands*. In the first case we refer to *raster graphics* and in the second case to *vector graphics.* Figure 2-1, shows two images of a cross, first defined as a bitmap, and then as a set of vector commands.

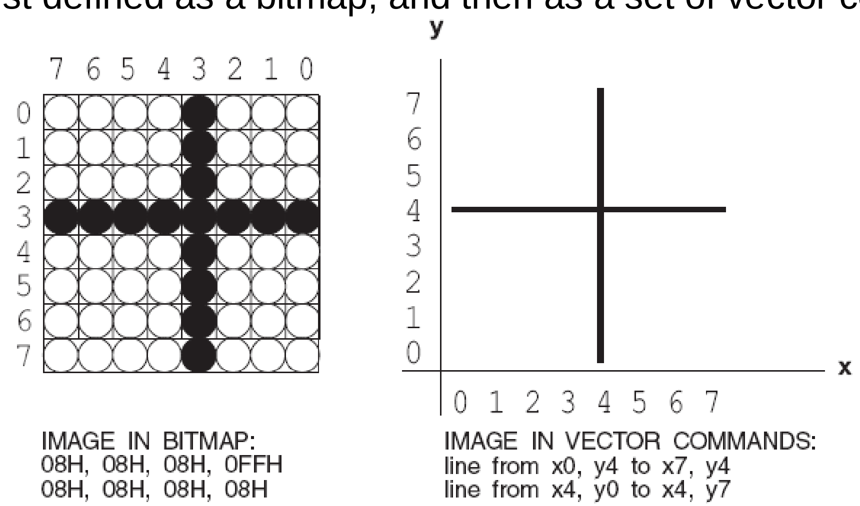


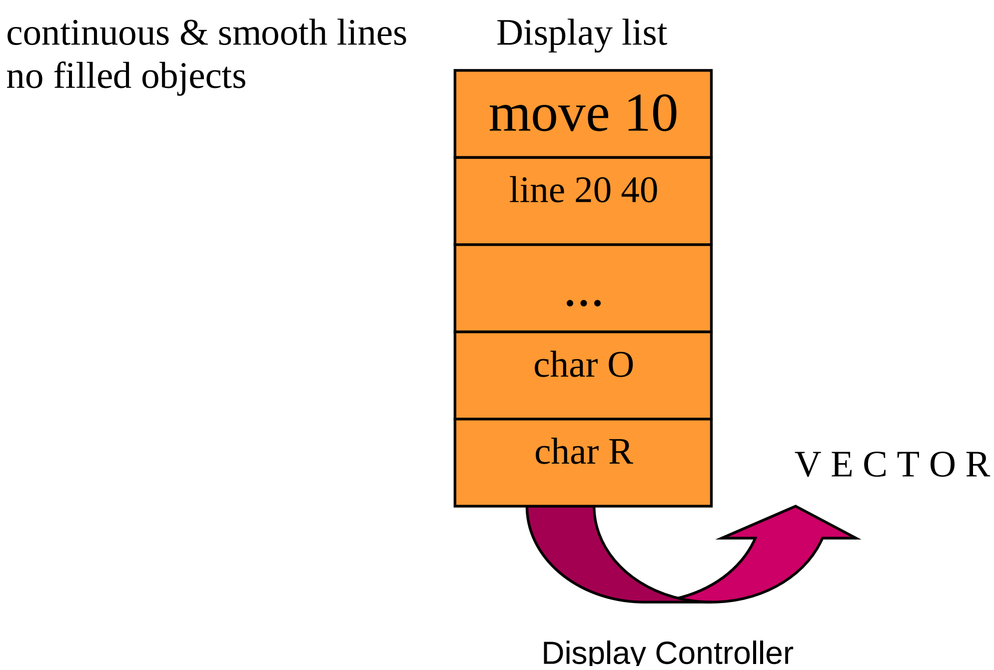
Figure 2-1 *Raster and Vector Representation of a Graphics Object*

The left-side image of Figure 2-1 shows the attribute of each pixel encoded in a bitmap. The simplest scheme consists of using a 0-bit to represent a white pixel and a 1-bit to represent a black pixel. Vector commands, on the other hand, refer to the geometrical elements in the image. The vector commands in Figure 2-1 define the image in terms of two intersecting straight lines. Each command contains the start and end points of the corresponding line in a Cartesian coordinate plane that represents the system's video display.

Vector Graphics

- *Image is represented by continuous geometric objects: points, lines, curves, etc.*
- *Graphics objects: geometry + color*
- *Geometric transformation possible without loss of information (zoom, rotate, …)*
- *Examples: PowerPoint, CorelDraw, AutoCAD...*

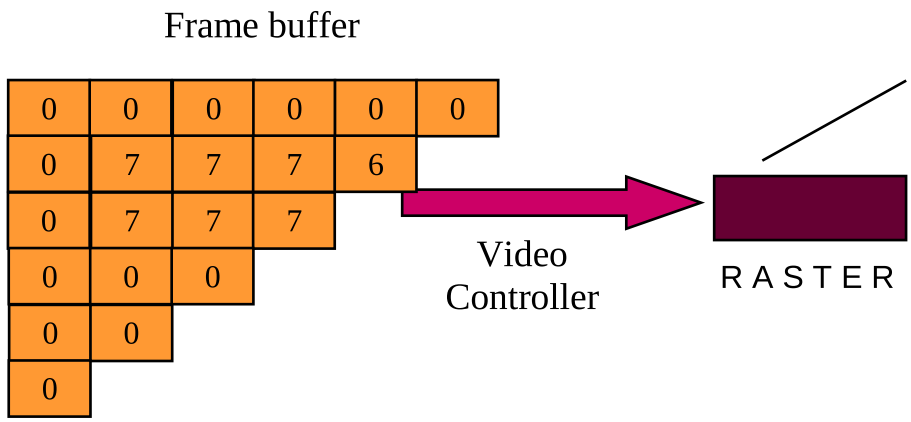
Vector Graphics Hardware



Raster Graphics

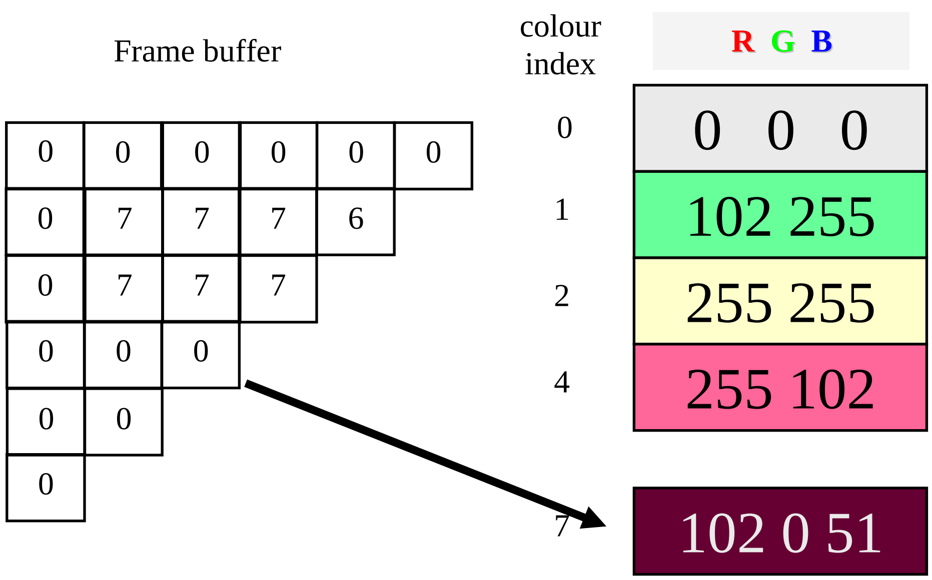
- *Image is represented as an rectangular grid of colored squares*
- Image processing techniques
- Geometric Transformation: loss of information
- Realistic images, textures, ...
- Examples: Paint, PhotoShop
- Raster graphics are also sometimes called *bitmapped graphics*

Raster Graphics Hardware



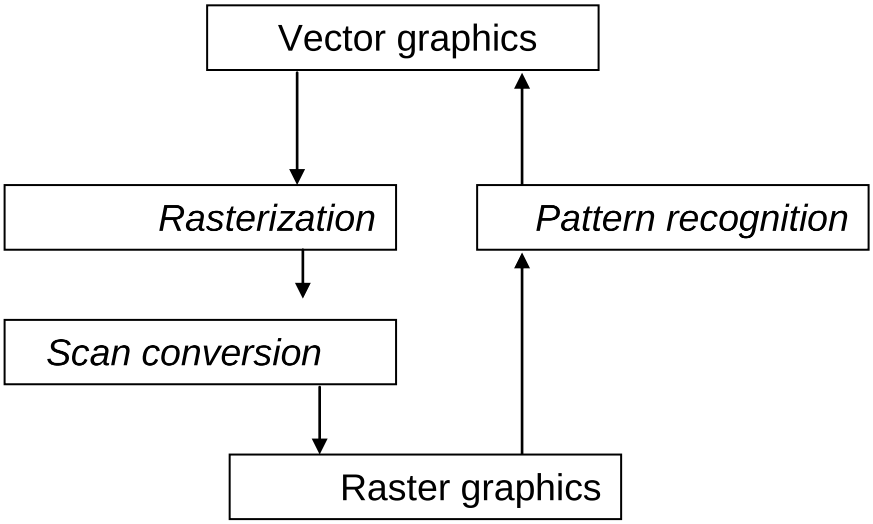
filled objects

Colour Lookup Table



CLUT:
pixel = code

True color:
pixel = R,G,B



Photoreal Rendering

Photoreal rendering ***refers to rendering a 3D scene in a realistic way***, Modern photoreal rendering algorithms are essentially a physically based simulation of light propagation and scattering throughout a 3D environment. Most modern photoreal rendering algorithms are based on the classic ***ray tracing algorithm***, that traces the path of individual light rays starting from the eye and working backwards to the light sources.

Computer Vision

A central goal in computer vision is to take a set of 2D images (usually from a video or set of photos) and infer from that a 3D description of what is being viewed, This is a very different process than rendering, and is more of a form of artificial intelligence.

Animation

An animation is just a sequence of individual images, basically, the subject of computer animation focuses on how things change over time. Usually, this refers to motion, but can also refer to other properties changing over time.

Modeling

Modeling refers to the ***techniques*** involved with creating, scanning, editing, and manipulating 3D geometric data, Modeling is often done by a human user with an interactive editing program, More complex objects, such as trees, can be constructed with automatic *procedural modeling* algorithms, 3D models are also often acquired from real world objects using laser scanning or computer vision techniques, Modeling also includes the use of curved surfaces and other higher order primitives, which are often converted into triangles using various *tessellation* algorithms, Another important area of modeling includes mesh reconstruction for surface simplification.

3D Models

A basic 3D model might consist of a simple array of triangles, Each triangle stores 3 vertices, Each vertex contains an xyz position, and possibly some other information (color,normal…)

Framebuffer

The *framebuffer* refers to the memory dedicated to storing the image, It would generally be a 2D array of pixels, where each pixel stores a color , Color is typically stored as a 24 bit RGB value. This offers 8 bits (256 levels) for red, green, and blue, for a total of 16,777,216 different colors

Traditional Graphics Pipeline

In the *traditional graphics pipeline*, each primitive is processed through the following steps:

- Transformation
- Lighting
- Clipping
- Scan conversion
- Pixel processing

Lighting

Lighting operations are applied to each vertex to compute its color

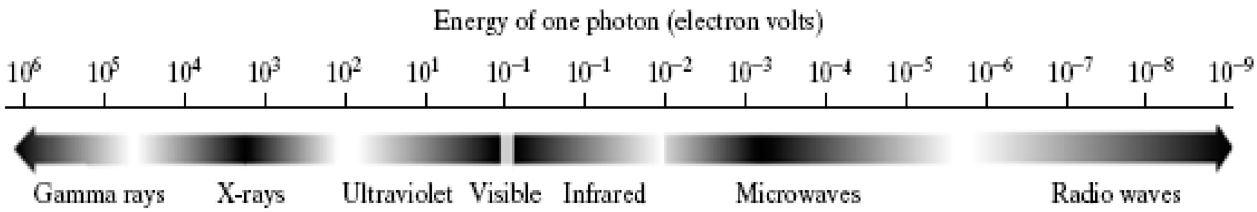
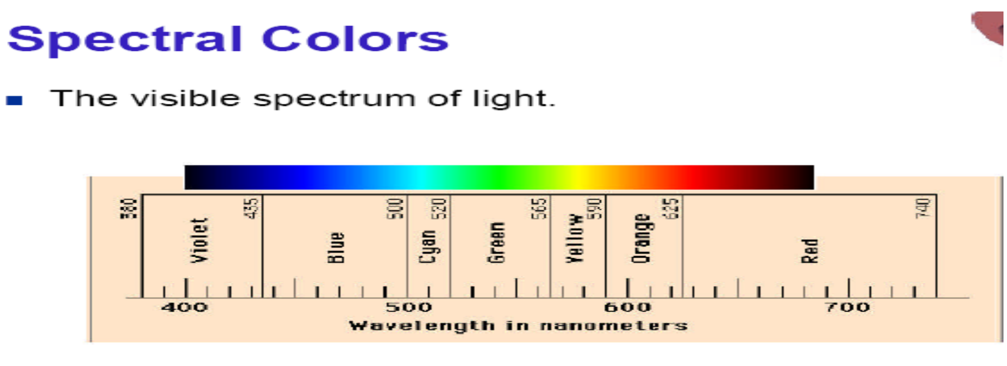


FIGURE 1.5 The electromagnetic spectrum arranged according to energy per photon.



In more advanced rendering, lighting operations are computed per pixel, rather than per vertex, A variety of light types can be defined such as point lights, directional lights, spot lights, etc., More advanced lighting operations can account for shadows, reflections, and a wide variety of optical effects

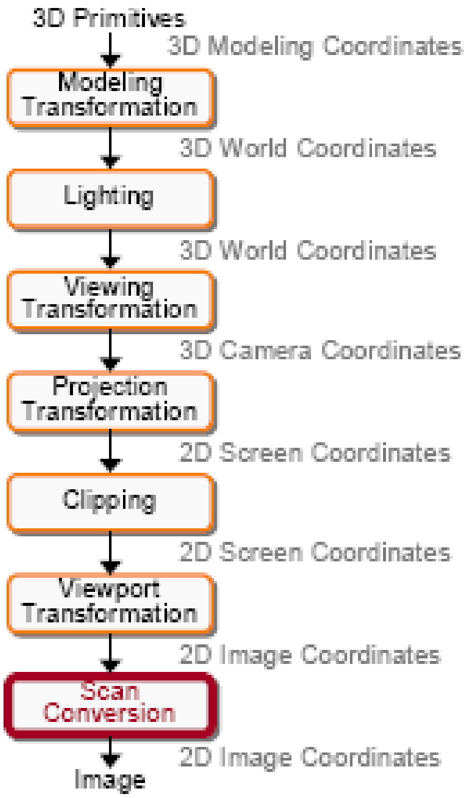
Scan Conversion

The *scan conversion* (or *rasterization*) process takes 2D triangles as input and outputs the exact pixels covered by the triangle

Scan conversion of geometric primitives : First transform the defining vertices to integer screen coordinates, then approximate the geometric object by assigning color values to pixels.

Example: A line is defined by two end points – two vertices. A polygon is defined by n vertices. Usually the rasterization of geometric primitives are done by using the process of scan conversion. The primitive is drawn by going through the *scan lines* of the display and turning on the necessary pixels.

Render an image of a geometric primitive by setting pixel colors



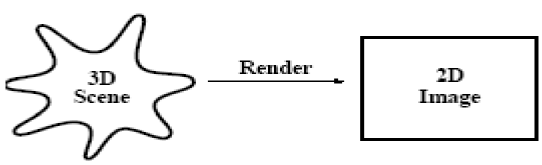
OpenGL

OpenGL is an interactive rendering standard designed for ease of use and portability across a wide range of systems, It is generally limited to things that can be done at interactive rates and does not include advanced rendering features such as ray tracing , It is based on the traditional graphics pipeline where triangles are processed one at a time

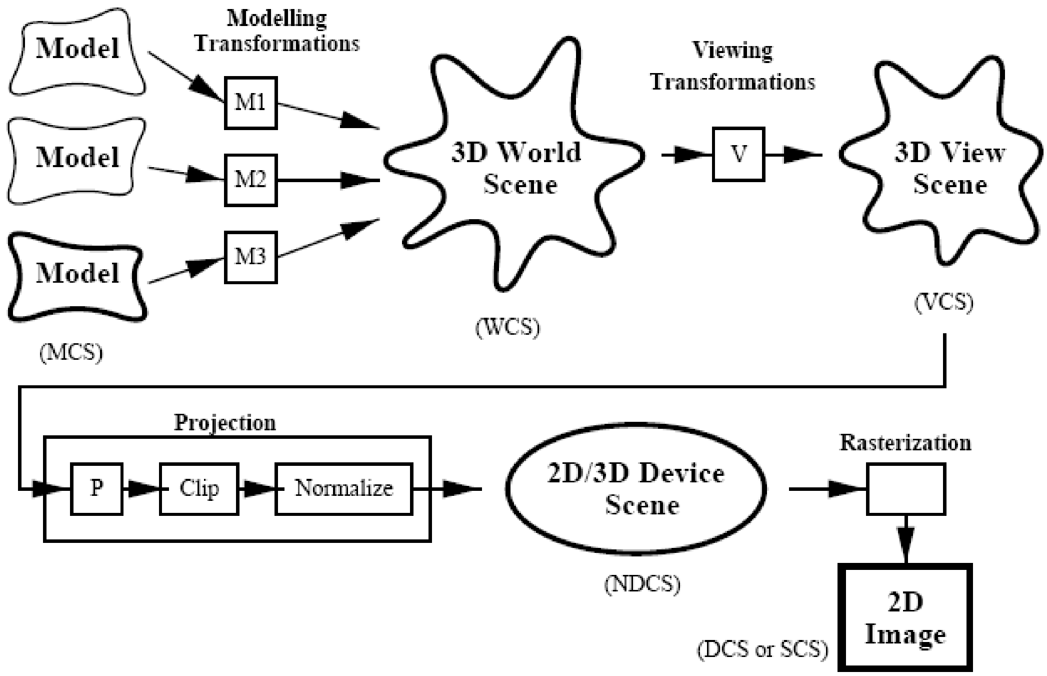
Pipeline

The Graphics Pipeline

_ Rendering is the conversion of a scene into an image:



- Scenes are composed of models in three-dimensional space. Models are composed of primitives supported by the rendering system. ***Models entered by hand or created by a program***. The image is drawn on monitor, printed on laser printer, or written to a raster in memory or a file. Requires us to consider device independence.
- Classically, "model" to "scene" to "image" conversion broken into finer steps, called the **graphics pipeline**. The basic forward projection pipeline looks like:



- Each stage refines the scene, converting primitives in modelling space to primitives in device space, where they are converted to pixels (rasterized).
- A number of coordinate systems are used:

MCS: Modeling Coordinate System.
WCS: World Coordinate System.
VCS: Viewer Coordinate System.
NDCS: Normalized Device Coordinate System.
DCS or SCS: Device Coordinate System or the Screen Coordinate System.
Keeping these straight is the key to understanding a rendering system.

- Transformation between two coordinate systems represented with matrix.
- Derived information may be added (lighting and shading) and primitives may be Removed (hidden surface removal) or modified (clipping).

Anti-aliasing

Aliasing is the jagged effect seen on lines and polygon boundaries after rasterization *and it* is the process of getting rid of the jaggedness of rasterized geometric primitives.

<u>Algorithms</u>

A number of basic algorithms are needed:

- *Transformation*: Convert representations of models/primitives from one coordinate system to another.
- Clipping/Hidden Surface Removal: Remove primitives and parts of primitives that are not visible on the display.
- Rasterization: Convert a projected screen-space primitive to a set of pixels.
- Picking: Select a 3D object by clicking an input device over a pixel location.
- Shading and Illumination: Simulate the interaction of light with a scene.
- Animation: Simulate movement by rendering a sequence of frames.

Modeling Geometrical Objects

Much of 3D graphics programming relates to representing, storing, manipulating, and rendering vector-coded geometrical objects. In this sense, *the problem of representation precedes all others*. Many representational forms are in use; most are related to a particular rendering algorithms associated with a graphics or development package. In addition, representational forms determine data structures, processing cost, final appearance, and editing ease. The following are the most frequently used:

1. *Polygonal representations* are based on reducing the object to a set of polygonal surfaces. This approach is the most popular one due to its simplicity and ease of rendering.
2. Objects can also be represented as *bicubic parameteric patch nets*. A patch net is a set of curvilinear polygons that approximate the object being modeled. Although more difficult to implement than polygonal representations, objects represented by parameteric patches are more fluid; this explains their popularity for developing CAD applications.
3. *Constructive solid geometry* (CSG) modeling is based on representing complex object by means of simpler, more elementary ones, such as cylinders, boxes, and spheres. This representation finds use in manufacturing-related applications.
4. *Space subdivision techniques* consider the whole object space and define each point accordingly. The best known application of space subdivision technique is *ray tracing*. With ray tracing processing is considerably simplified by avoiding brute force operations on the entire object space.

Modeling with Polygons

A simple polygon is a 2D figure formed by more than two connected and non-intersecting line segments. The connection points for the line segments are called the vertices of the polygon and the line segments are called the sides. The fundamental requirements that the line segments be connected and non-intersecting eliminates from the polygon category certain geometrical figures, as shown in Figure 2-8.

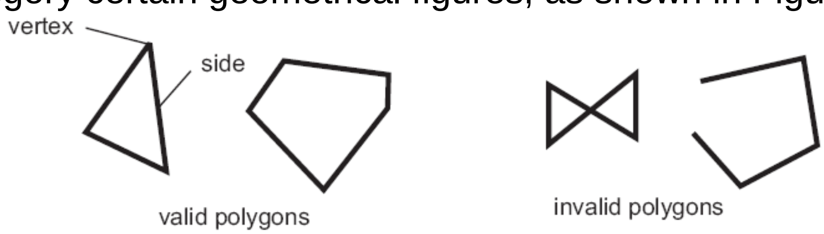


Figure 2-8 *Valid and Invalid Polygons*

Polygons are named according to their number of sides or vertices. A triangle, which is the simplest possible polygon, has three vertices. A quadrilateral has four, a pentagon has five, and so on. Figure 2-9 shows several regular polygons.

Figure 2-9 *Regular Polygons*

Polygons can be *convex* or *concave*. In a convex polygon the extension of any of its sides does not cut across the interior of the figure. We can also describe a convex polygon as one in which the extensions of the lines that form the sides never meet another side. Figure 2-10 shows a convex and a concave polygon.

Figure 2-10 *Concave and Convex Polygons*

The Triangle

Of all the polygons, the one most used in 3D graphics is the triangle. Not only is it the simplest of the polygons, but all the points in the surface of a triangular polygon must lie on the same plane. In other polygons this may or may not be the case. In other words, the figure defined by three vertices must always be a plane, but four or more vertices can describe a figure with more than one plane. When all the points on the figure are located on the same surface, the figure is said to be coplanar. Figure 2-11 shows coplanar and non-coplanar polygons.

Figure 2-11 *Coplanar and Non-Coplanar Polygons*

The coplanar property of triangular polygons simplifies rendering. In addition, triangles are always convex figures. For this reason 3D software such as Microsoft's Direct3D, rely heavily on triangular polygons.

Polygonal Approximations

Solid objects with curved surfaces can be approximately represented by combining several polygonal faces. For example, a circle can be approximated by means of a polygon. The more vertices in the polygon, the better the approximation. Figure 2-12 shows the polygonal approximation of a circle. The first polygon has 8 vertices, while the second one has 16.

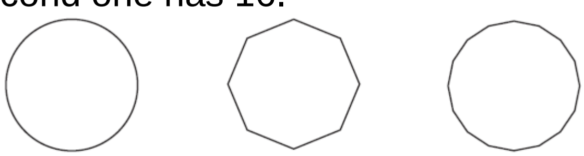


Figure 2-12 *Polygonal Approximation of a Circle*

A solid object, such as a cylinder, can be approximately represented by means of several polygonal surfaces. Here again, the greater the number of polygons, the more accurate the approximation. Figure 2-13 shows the polygonal approximation of a cylinder.

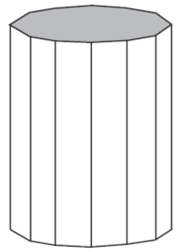


Figure 2-13 *Polygonal Approximation of a Cylinder*

Edges

When objects are represented by polygonal approximations, often two polygons share a common side. This connection between vertex locations that define a boundary is called an *edge*. Edge representations of polygons simplify the database by avoiding redundancy.

This is particularly useful if an object shares a large number of edges. Figure 2-14 shows a figure represented by two adjacent triangular polygons that share a common edge

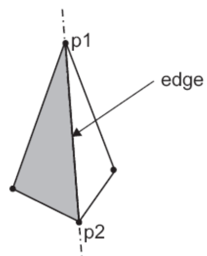


Figure 2-14 *Polygon Edge*

In an edge representation the gray triangle in Figure 2-14 is defined in terms of its three vertices, labeled *p*1, *p*2, and *p*3. The white triangle is defined in terms of its edge and point *p*4. Thus, points *p*2 and *p*3 appear but once in the database. Edge-based image databases provide a list of edges rather than of vertex locations. Figure 2-15 shows an object consisting of rectangular polygons.

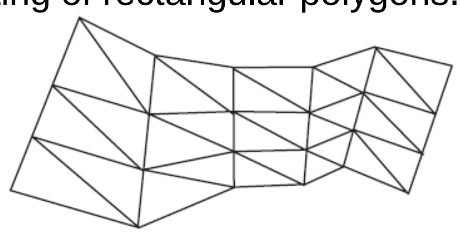


Figure 2-15 *Edge Representation of Polygons*

In Figure 2-15 each vertical panel consists of 6 triangles, for a total of 30 triangles. If each triangle were defined by its three vertices, the image database would require 90 vertices. Alternatively, the image could be defined in terms of sides and edges. There are 16 external sides which are not shared, and 32 internal sides, which are edges. Therefore, the edge-based representation could be done by defining 48 edges. The rendering system keeps track of which edges have already been drawn, avoiding duplication, processing overheads, and facilitating transparency.

Meshes

In 3D graphics an object can be represented as a polygon mesh. Each polygon in the mesh constitutes a *facet*. Facets are used to approximate curved surfaces; the more facets the better the approximation. Polygon-based modeling is straightforward and polygon meshes are quite suitable for using shading algorithms. In the simplest form a polygon mesh is encoded by means of the *x*, *y*, and *z* coordinates of each vertex. Alternatively, polygons can be represented by their edges, as previously described. In either case, each polygon is an independent entity that can be rendered as a unit. Figure 2-16 shows the polygon mesh representation of a teacup and the rendered image.

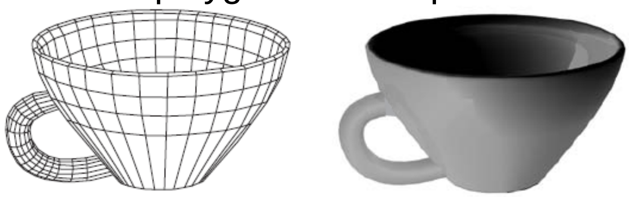


Figure 2-16 *Polygon Mesh Representation and Rendering of a Teacup*