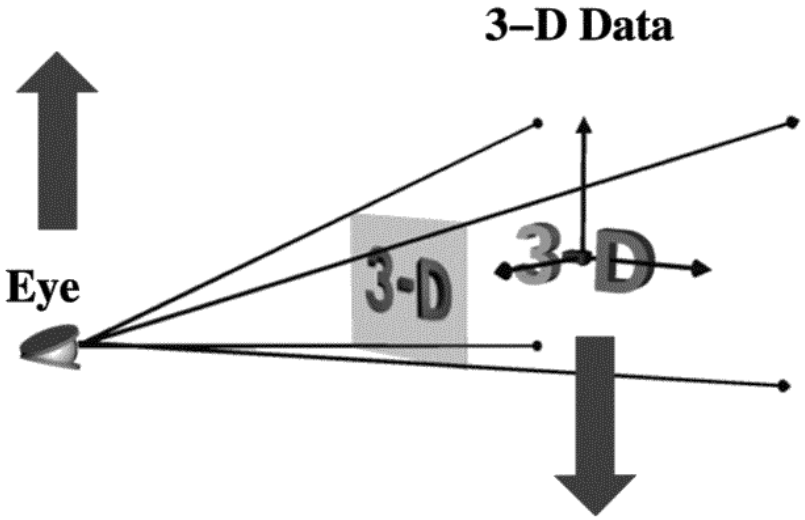
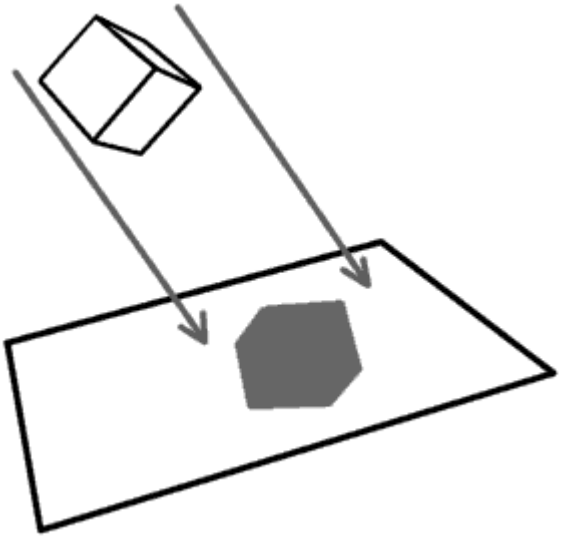
Introduction to Computer Graphics
(C S 6 0 2)
Lecture 16
3D Concepts

Welcome! You are about to embark on a journey into the wondrous world of three-dimensional computer graphics. Before we take the plunge into esoteric 3D jargon and mathematical principles (as we will in the next lectures), let's have a look at what the buzzword "3D" actually means.
We have heard the term "3D" applied to everything from games to the World Wide Web to Microsoft's new look for Windows XP. The term 3D is often confusing because games (and other applications) which claim to be 3D, are not really 3D. In a 3D medium, each of our eyes views the scene from slightly different angles. This is the way we perceive the real world. Obviously, the flat monitors most of us use when playing 3D games 3D applications can't do this. However, some Virtual Reality (VR) glasses have this capability by using a separate TV-like screen for each eye. These VR glasses may become common place some years from now, but today, they are not the norm. Thus, for present-day usage, we can define "3D to mean "something using a three-dimensional coordinate system."

A three-dimensional coordinate system is just a fancy term for a system that measures objects with width, height, and depth (just like the real world). Similarly, 2-dimensional coordinate systems measure objects with width and height --- ignoring depth properties (so unlike the real world).





Shadow of a 3D object on paper

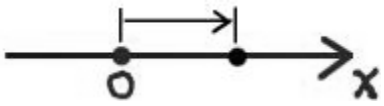
16.1 Coordinate Systems

Coordinate systems are the measured frames of reference within which geometry is defined, manipulated and viewed. In this system, you have a well-known point that serves as the origin (reference point), and three lines(axes) that pass through this point and are orthogonal to each other (at right angles – 90 degrees).

With the Cartesian coordinate system, you can define any point in space by saying how far along each of the three axes you need to travel in order to reach the point if you start at the origin.

Following are three types of the coordinate systems.

a) 1-D Coordinate Systems:

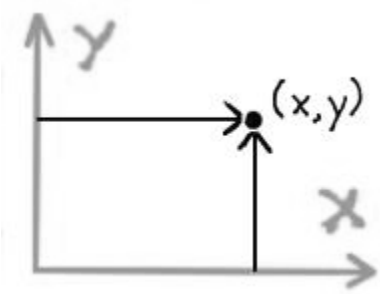


This system has the following characteristics:

- Direction and magnitude along a single axis, with reference to an origin
- Locations are defined by a single coordinate
- Can define points, segments, lines, rays

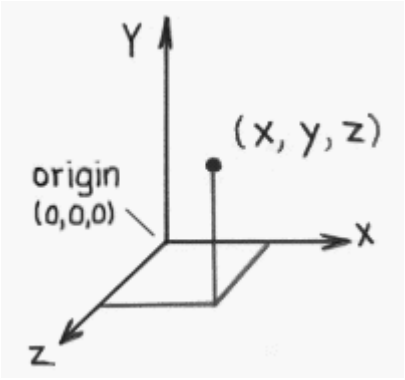
- Can have multiple origins (frames of reference) and transform coordinates among them

b) 2-D Coordinate Systems:



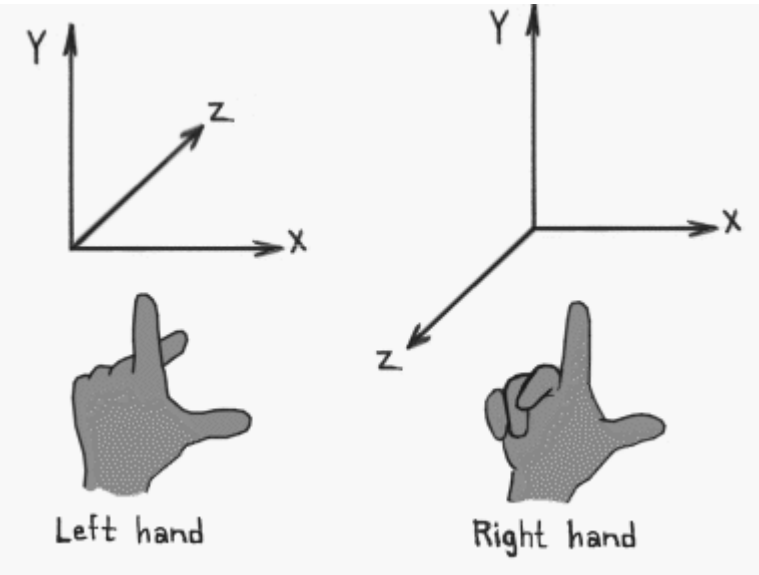
- Direction and magnitude along two axes, with reference to an origin
- Locations are defined by x, y coordinate pairs
- Can define points, segments, lines, rays, curves, polygons, (any planar geometry)
- Can have multiple origins (frames of reference and transform coordinates among them

c) 3-D Coordinate Systems:



- 3D Cartesian coordinate systems
- Direction and magnitude along three axes, with reference to an origin
- Locations are defined by x, y, z triples
- Can define cubes, cones, spheres, etc., (volumes in space) in addition to all one- and two-dimensional entities
- Can have multiple origins (frames of reference) and transform coordinates among them

16.2 Left-handed versus Right-handed



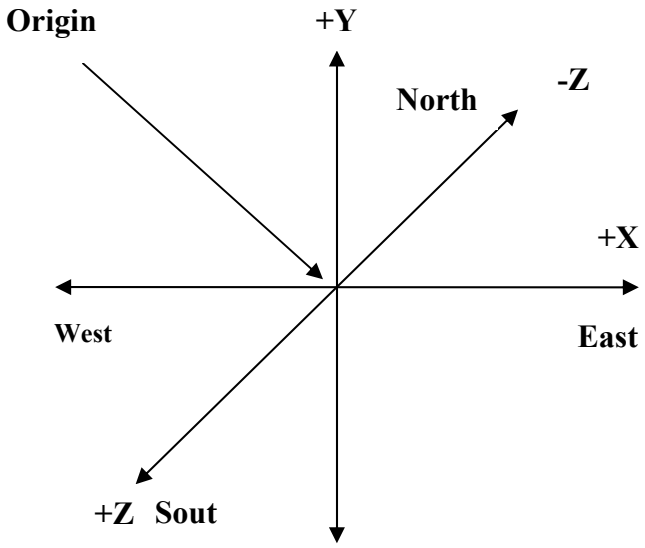
- Determines orientation of axes and direction of rotations
- Thumb = pos x, Index up = pos y, Middle out = pos z
- Most world and object axes tend to be right handed
- Left handed axes often are used for cameras

a) Right Handed Rule:

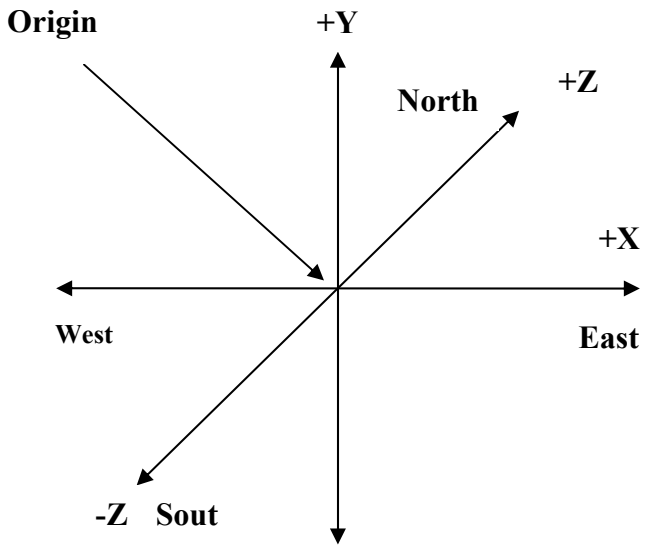
"Right Hand Rule" for rotations: grasp axis with right hand with thumb oriented in positive direction, fingers will then curl in direction of positive rotation for that axis.

Right handed Cartesian coordinate system describes the relationship of the X,Y, and Z in the following manner:

- X is positive to the right of the origin, and negative to the left.
- Y is positive above the origin, and negative below it.
- Z is *negative* beyond the origin, and *positive* behind it.



b) Left Handed Rule:



Left handed Cartesian coordinate system describes the relationship of the X, Y and Z in the following manner:

- X is positive to the right of the origin, and negative to the left.
- Y is positive above the origin, and negative below it.
- Z is positive beyond the origin, and negative behind it.

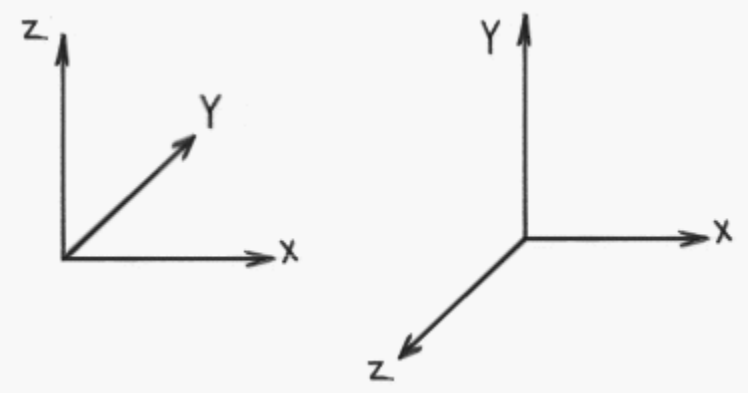
16.3 Defining 3D points in mathematical notations

3D points can be described using simple mathematical notations

P = (X, Y, Z)

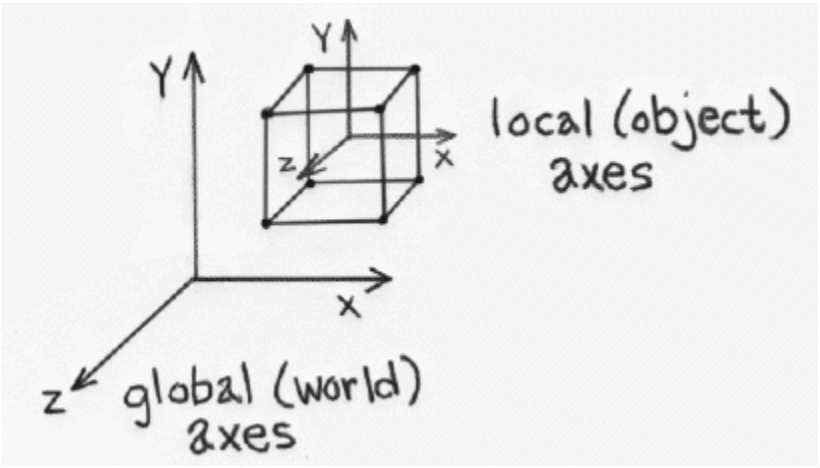
Thus the origin of the Coordinate system is located at point (0,0,0), while five units to the right of that position might be located at point (5,0,0).

Y-up versus Z-up:



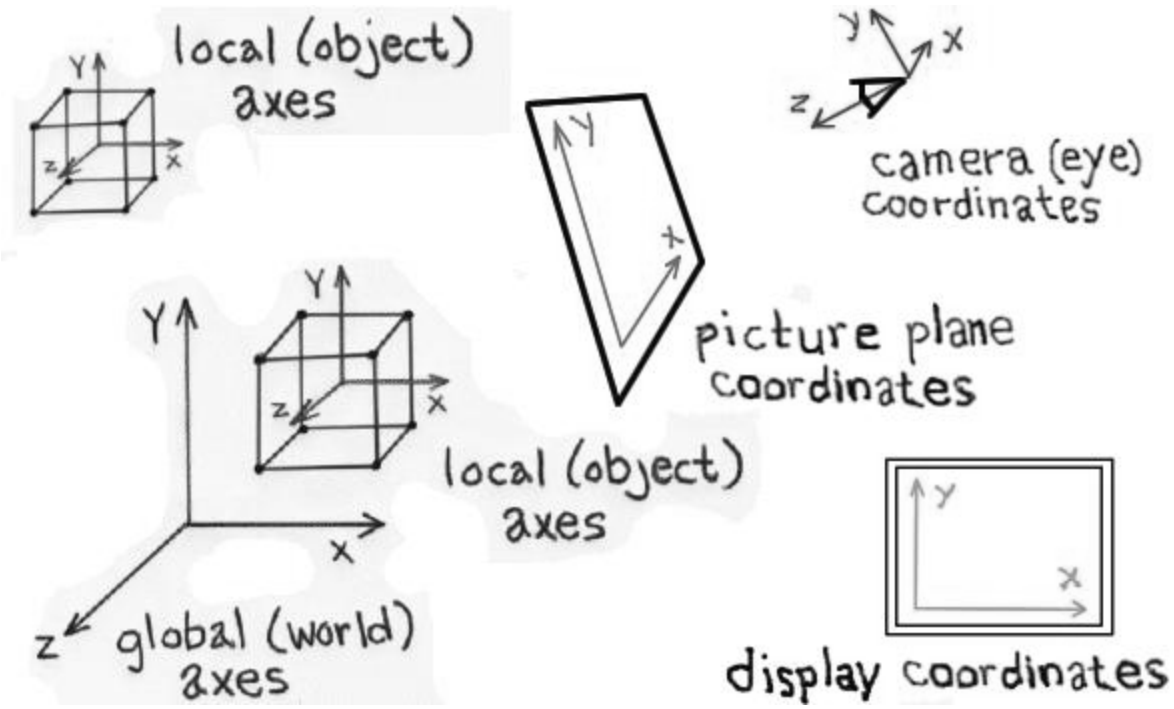
- z-up typically used by designers
- y-up typically used by animators
- orientation by profession supposedly derives from past work habits
- often handled differently when moving from application to application

16.4 Global and Local Coordinate Systems:



- Local coordinate systems can be defined with respect to global coordinate system
- Locations can be relative to any of these coordinate systems
- Locations can be translated or "transformed" from one coordinate system to another.

16.5 Multiple Frames of Reference in a 3-D Scene:



- In fact, there usually are multiple coordinate systems within any 3-D screen
- Application data will be transformed among the various coordinate systems, depending on what's to be accomplished during program execution
- Individual coordinate systems often are hierarchically linked within the scene

16.6 Defining points in C language structure

You can now define any point in the 3D by saying how far east, up, and north it is from your origin. The center of your computer screen ? it would be at a point such as "1.5 feet east, 4.0 feet up, 7.2 feet north." Obviously, you will want a data structure to represent these points. An example of such a structure is shown in this code snippet:

```
typedef struct _POINT3D
{
    float x;
    float y;
    float z;
}POINT3D;

POINT3D screenCenter = {1.5, 4.0, 7.2};
```

16.7 The Polar Coordinate System

Cartesian systems are not the only ones we can use. We could have also described the object position in this way: "starting at the origin, looking east, rotate 38 degrees

northward, 65 degrees upward, and travel 7.47 feet along this line. “As you can see, this is less intuitive in a real world setting. And if you try to work out the math, it is harder to manipulate (when we get to the sections that move points around). Because such polar coordinates are difficult to control, they are generally not used in 3D graphics.

16.8 Using Multiple Coordinate Systems

As we start working with 3D objects, you may find that it is more efficient to work with groups of points instead of individual single points. For example, if you want to model your computer, you may want to store it in a structure such as that shown in this code snippet:

```
typedef struct _CPU{

        POINT3D center;       // the center of the CPU, in World coordinates
        POINT3D coord[8];     // the 8 corners of the CPU box relative to the center point

}CPU;
```

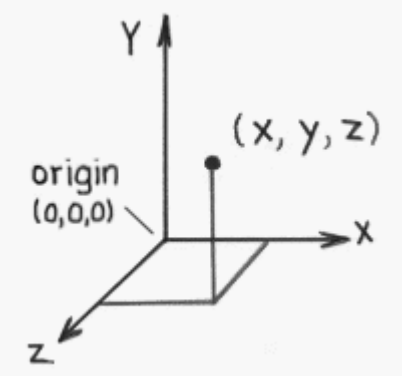
In next lectures we will learn how we can show 3D point on 2D computer screen.

16.9 Defining Geometry in 3-D

Here are some definitions of the technical names that will be used in 3D lectures.

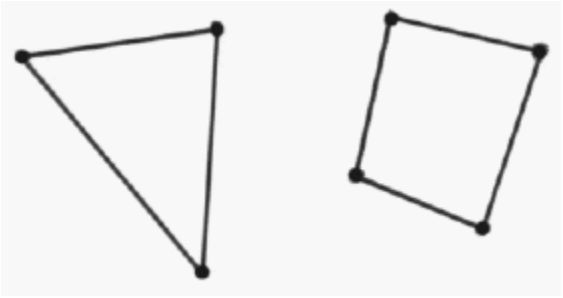
Modeling: is the process of describing an object or scene so that we can construct an image of it.

Points & Polygons:

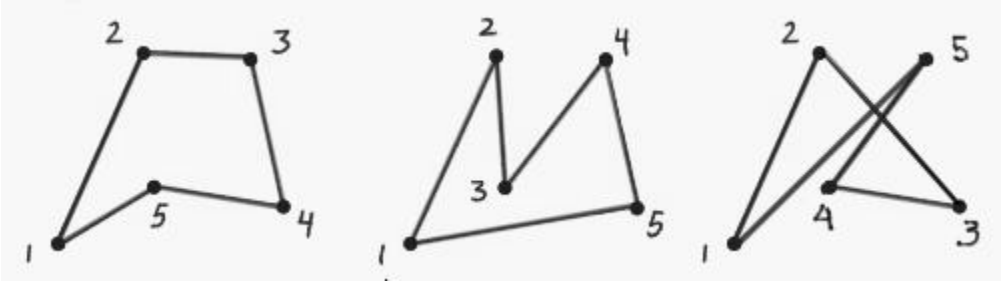


- Points: three-dimensional locations (or coordinate triples)

- Vectors: - have direction and magnitude; can also be thought of as displacement



- Polygons: - sequences of "correctly" co-planar points; or an initial point and a sequence of vectors

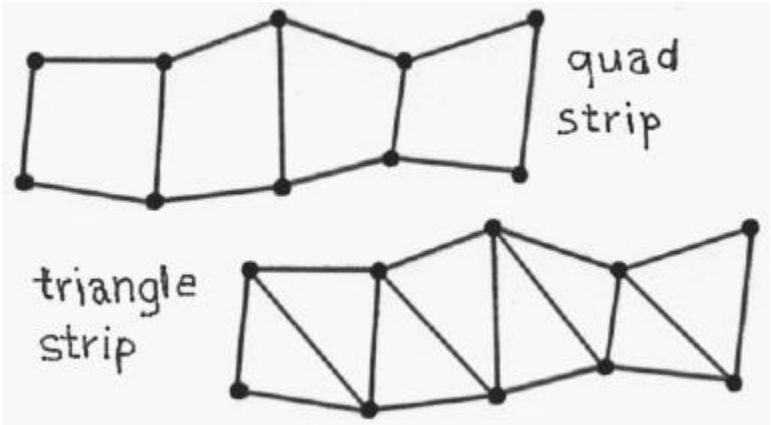


Primitives

Primitives are the fundamental geometric entities within a given data structure.

- We have already touched on point, vector and polygon primitives

- Regular Polygon Primitives - square, triangle, circle, n-polygon, etc.

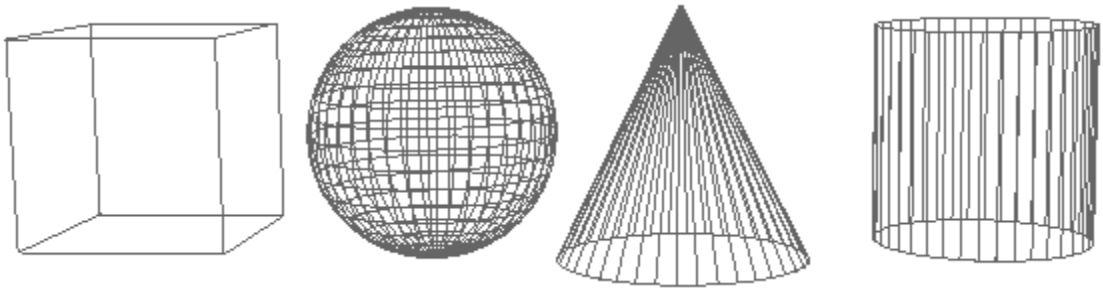


- Polygon strips or meshes
- Meshes provide a more economical description than multiple individual polygons

  For example, 100 individual triangles, each requiring 3 vertices, would require 100 x 3 or 300 vertex definitions to be stored in the 3-D database.

  By contrast, triangle strips require n + 2 vertex definitions for any n number or triangles in the strip. Hence, a 100 triangle strip requires only 102 unique vertex definitions.

- Meshes also provide continuity across surfaces which is important for shading calculations



- **3D primitives in a polygonal database**

3D shapes are represented by polygonal meshes that define or approximate geometric surfaces.

- With curved surfaces, the accuracy of the approximation is directly proportional to the number of polygons used in the representation.
- More polygons (when well used) yield a better approximation.
- But more polygons also exact greater computational overhead, thereby degrading interactive performance, increasing render times, etc.

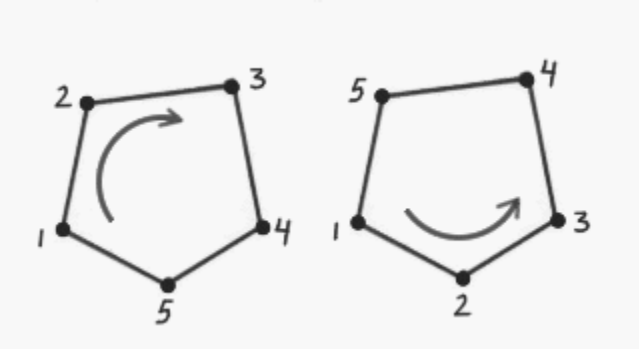
Rendering - The process of computing a two dimensional image using a combination of a three-dimensional database, scene characteristics, and viewing transformations. Various algorithms can be employed for rendering, depending on the needs of the application.

Tessellation - The subdivision of an entity or surface into one or more non-overlapping primitives. Typically, renderers decompose surfaces into triangles as part of the rendering process.

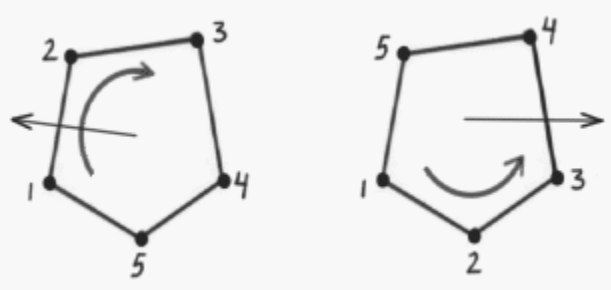
Sampling - The process of selecting a representative but finite number of values along a continuous function sufficient to render a reasonable approximation of the function for the task at hand.

Level of Detail (LOD) - To improve rendering efficiency when dynamically viewing a scene, more or less detailed versions of a model may be swapped in and out of the scene database depending on the importance (usually determined by image size) of the object in the current view.

Polygons and rendering

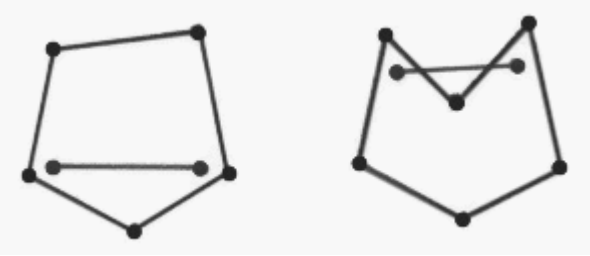


- Clockwise versus counterclockwise

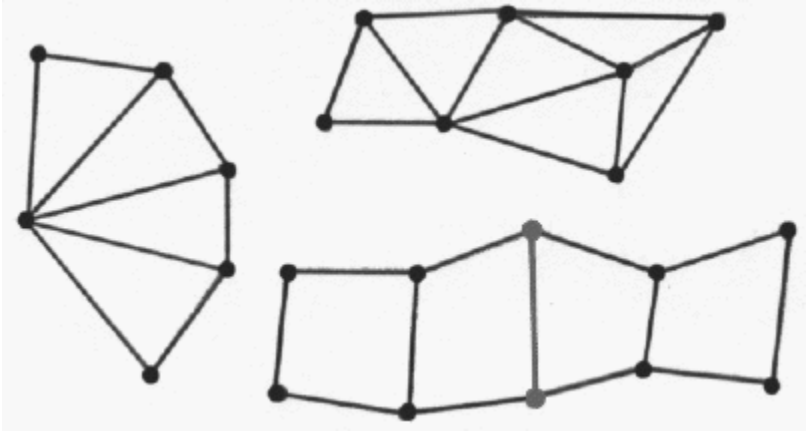


Surface normal - a vector that is perpendicular to a surface and "outward" facing

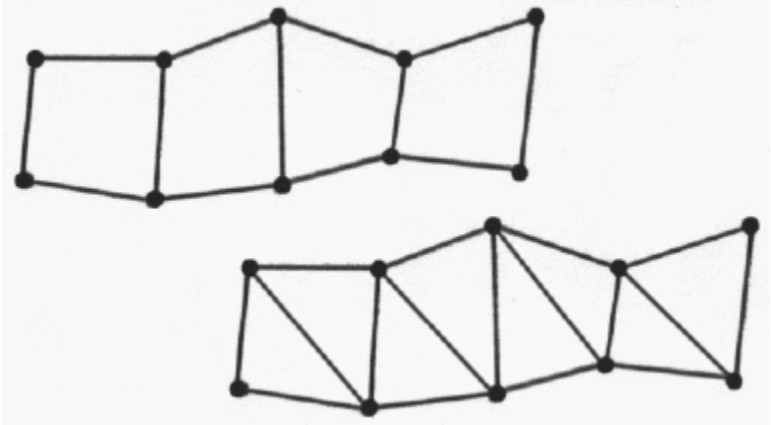
- Surface normals are used to determine visibility and in the calculation of shading values (among other things)



- Convex versus concave
    - A shape is convex if any two points within the shape can be connected with a straight line that never goes out of the shape. If not, the shape is concave.
    - Concave polygons can cause problems during rendering (e.g. tears, etc., in apparent surface).



- Polygon meshes and shared vertices



- Polygons consisting of non-co-planar vertices can cause problems when rendering (e.g. visible tearing of the surface, etc.)
- With quad meshes, for example, vertices within polygons can be inadvertently transformed into non-co-planer positions during modeling or animation transformations.
- With triangle meshes, all polygons are triangles and therefore all vertices within any given polygon will be coplanar.

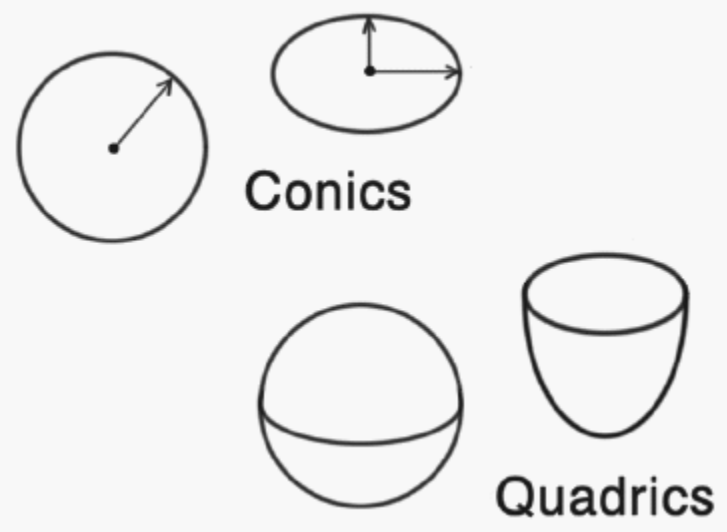
With polygonal databases:

- Explicit, low-level descriptions of geometry tend to be employed
- Object database files can become very large relative to more economical, higher order descriptions.
- Organic forms or free-form surfaces can be difficult to model.

16.10 Surface models

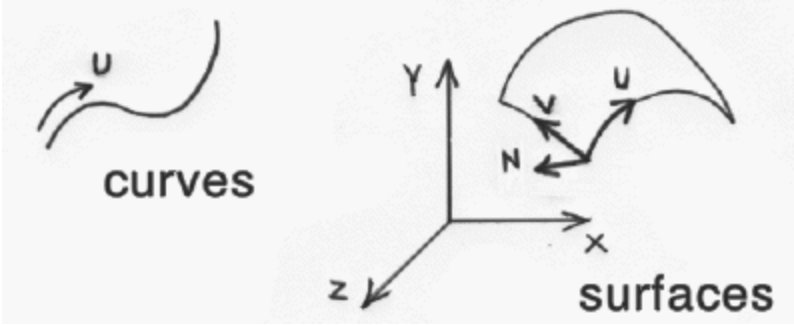
Here is brief over view of surface models:

- Surfaces can be constructed from mathematical descriptions
- Resolution independent - surfaces can be tessellated at rendering with an appropriate level of approximation for current display devices and/or viewing parameters
- Tessellation can be adaptive to the local degree of curvature of a surface.

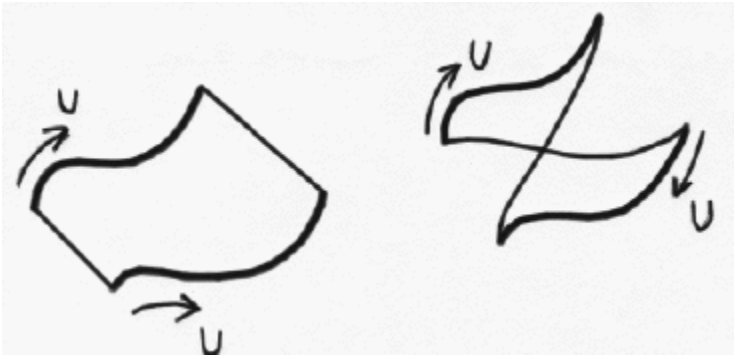


- Primitives

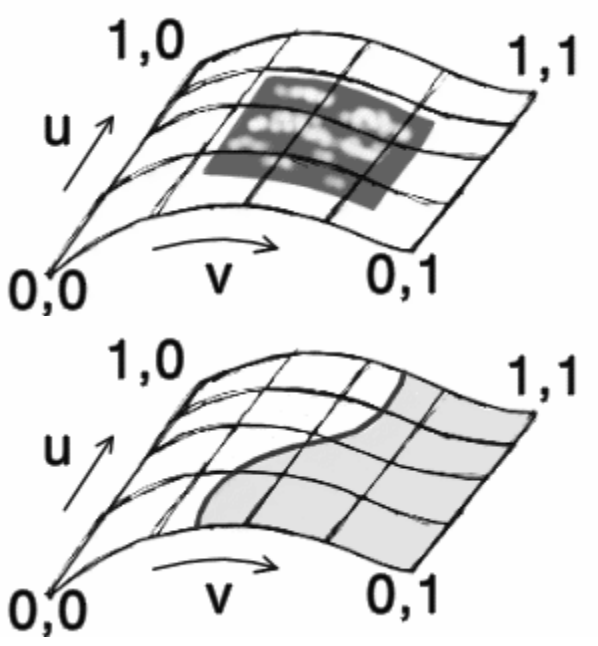
- Free-form surfaces can be built from curves
- Construction history, while also used in polygonal modeling, can be particularly useful with curve and surface modeling techniques.



- Parameterization



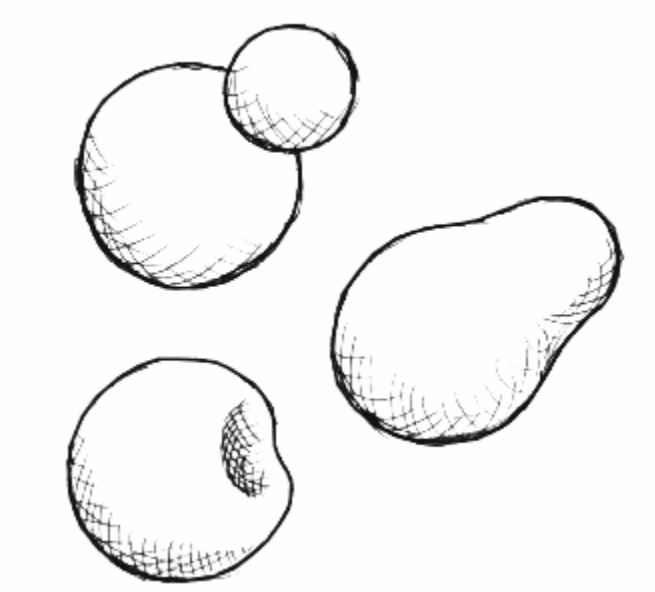
- Curve direction and surface construction



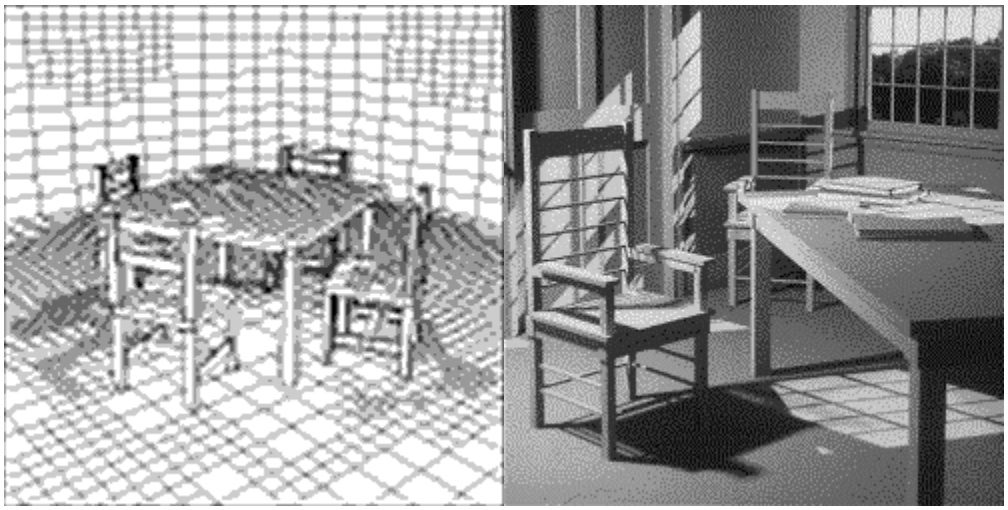
- Surface parameterization (u, v, w) are used
    - For placing texture maps, etc.
    - For locating trimming curves, etc.

Metaballs (blobby surfaces)

- Potential functions (usually radially symmetric Gaussian functions) are used to define surfaces surrounding points



Lighting Effects



Texture Mapping:

The texture mapping is of the following types that we will be studying in our coming lectures on 3D:

1. Perfect Mapping:
2. Affine Mapping
3. Area Subdivision
4. Scan-line Subdivision
5. Parabolic Mapping
6. Hyperbolic Mapping
7. Constant-Z Mapping