# Polygonal modeling

In 3D computer graphics, **polygonal modeling** is an approach for modeling objects by representing or approximating their surfaces using polygon meshes. Polygonal modeling is well suited to scanline rendering and is therefore the method of choice for real-time computer graphics. Alternate methods of representing 3D objects include NURBS surfaces, subdivision surfaces, and equation-based representations used in ray tracers.

## Contents



## Geometric theory and polygons

The basic object used in mesh modeling is a vertex, a point in three-dimensional space. Two vertices connected by a straight line become an edge. Three vertices, connected to each other by three edges, define a triangle, which is the simplest polygon in Euclidean space. More complex polygons can be created out of multiple triangles, or as a single object with more than 3 vertices. Four sided polygons (generally referred to as quads)[1][2] and triangles are the most common shapes used in polygonal modeling. A group of polygons, connected to each other by shared vertices, is generally referred to as an **element**. Each of the polygons making up an element is called a **face**.

In Euclidean geometry, any three non-collinear points determine a plane. For this reason, triangles always inhabit a single plane. This is not necessarily true of more complex polygons, however. The flat nature of triangles makes it simple to determine their surface normal, a three-dimensional vector perpendicular to the triangle's surface. Surface normals are useful for determining light transport in ray tracing, and are a key component of the popular Phong shading model. Some rendering systems use vertex normals instead of face normals to create a better-looking lighting system at the cost of more processing. Note that every triangle has two face normals, which point to opposite directions from each other. In many systems only one of these normals is considered valid – the other side of the polygon is referred to as a **backface**, and can be made visible or invisible depending on the programmer’s desires.

Many modeling programs do not strictly enforce geometric theory; for example, it is possible for two vertices to have two distinct edges connecting them, occupying exactly the same spatial location. It is also possible for two vertices to exist at the same spatial coordinates, or two faces to exist at the same location. Situations such as these are usually not desired and many packages support an auto-cleanup function. If auto-cleanup is not present, however, they must be deleted manually.

A group of polygons which are connected by shared vertices is referred to as a **mesh**. In order for a mesh to appear attractive when rendered, it is desirable that it be **non-self-intersecting**, meaning that no edge passes through a polygon. Another way of looking at this is that the mesh cannot pierce itself. It is also desirable that the mesh not contain any errors such as doubled vertices, edges, or faces. For some purposes it is important that the mesh be a manifold – that is, that it does not contain holes or singularities (locations where two distinct sections of the mesh are connected by a single vertex).

# Construction of polygonal meshes

Although it is possible to construct a mesh by manually specifying vertices and faces, it is much more common to build meshes using a variety of tools. A wide variety of 3D graphics software packages are available for use in constructing polygon meshes.

One of the more popular methods of constructing meshes is box modeling, which uses two simple tools:

- The **subdivide** tool splits faces and edges into smaller pieces by adding new vertices. For example, a square would be subdivided by adding one vertex in the center and one on each edge, creating four smaller squares.
- The **extrude** tool is applied to a face or a group of faces. It creates a new face of the same size and shape which is connected to each of the existing edges by a face. Thus, performing the **extrude** operation on a square face would create a cube connected to the surface at the location of the face.

A second common modeling method is sometimes referred to as **inflation modeling** or **extrusion modeling**. In this method, the user creates a 2D shape which traces the outline of an object from a photograph or a drawing. The user then uses a second image of the subject from a different angle and extrudes the 2D shape into 3D, again following the shape's outline. This method is especially common for creating faces and heads. In general, the artist will model half of the head and then duplicate the vertices, invert their location relative to some plane, and connect the two pieces together. This ensures that the model will be symmetrical.

Another common method of creating a polygonal mesh is by connecting together various **primitives**, which are predefined polygonal meshes created by the modeling environment. Common primitives include:

- Cubes
- Pyramids
- Cylinders
- 2D primitives, such as squares, triangles, and disks
- Specialized or esoteric primitives, such as the Utah Teapot or Suzanne, Blender's monkey mascot.
- Spheres - Spheres are commonly represented in one of two ways:
  - **Icospheres** are icosahedrons which possess a sufficient number of triangles to resemble a sphere.
  - **UV spheres** are composed of quads, and resemble the grid seen on some globes - quads are larger near the "equator" of the sphere and smaller near the "poles," eventually terminating in a single vertex.

Finally, some specialized methods of constructing high or low detail meshes exist. Sketch based modeling is a user-friendly interface for constructing low-detail models quickly, while 3D scanners can be used to create high detail meshes based on existing real-world objects in almost automatic way. These devices are very expensive, and are generally only used by researchers and industry professionals but can generate high accuracy sub-millimetric digital representations.

# Operations

There is a very large number of operations which may be performed on polygonal meshes. Some of these roughly correspond to real-world manipulations of 3D objects, while others do not. Polygonal mesh operations include:

- Creations - Create new geometry from some other mathematical object
  - *Loft* - generate a mesh by creating a shape along two or more profile curves
  - Extrude - creates a surface by sweeping a profile curve or polygon surface along a straight or linear line
  - Revolve - generate a mesh by revolving (rotating) a shape around an axis
  - Marching cubes - algorithm to construct a mesh from an implicit function
- Binary Creations - Create a new mesh from a binary operation of two other meshes
  - Add - boolean addition of two or more meshes
  - Subtract - boolean subtraction of two or more meshes
  - Intersect - boolean intersection
  - Union - boolean union of two or more meshes
  - Attach - attach one mesh to another (removing the interior surfaces)
  - Chamfer - create a beveled surface which smoothly connects two surfaces
- Deformations - Move only the vertices of a mesh
  - Deform - systematically move vertices (according to certain functions or rules)
  - Weighted Deform - move vertices based on localized weights per vertex
  - Morph - move vertices smoothly between a source and target mesh
  - Bend - move vertices to "bend" the object
  - Twist - move vertices to "twist" the object
- Manipulations - Modify the geometry of the mesh, but not necessarily topology
  - Displace - introduce additional geometry based on a "displacement map" from the surface
  - Simplify - systematically remove and average vertices
  - Subdivide - smooth a course mesh by subdividing the mesh (Catmull-Clark, etc.)
  - Convex Hull - generate another mesh which minimally encloses a given mesh (think shrink-wrap)
  - Cut - create a hole in a mesh surface
  - Stitch - close a hole in a mesh surface
- Measurements - Compute some value of the mesh
  - Volume - compute the 3D volume of a mesh (discrete volumetric integral)
  - Surface Area - compute the surface area of a mesh (discrete surface integral)
  - Collision Detection - determine if two complex meshes in motion have collided
  - Fitting - construct a parametric surface (NURBS, bicubic spline) by fitting it to a given mesh
  - Point-Surface Distance - compute distance from a point to the mesh
  - Line-Surface Distance - compute distance from a line to the mesh
  - Line-Surface Intersection - compute intersection of line and the mesh
  - Cross Section - compute the curves created by a cross-section of a plane through a mesh
  - Centroid - compute the centroid, geometric center, of the mesh
  - Center-of-Mass - compute the center of mass, balance point, of the mesh

- Circumcenter - compute the center of a circle or sphere enclosing an element of the mesh
- Incenter - compute the center of a circle or sphere enclosed by an element of the mesh

## Extensions

Once a polygonal mesh has been constructed, further steps must be taken before it is useful for games, animation, etc. The model must be texture mapped to add colors and texture to the surface and it must be given a skeleton for animation. Meshes can also be assigned weights and centers of gravity for use in physical simulation.

To display a model on a computer screen outside of the modeling environment, it is necessary to store that model in one of the file formats listed below, and then use or write a program capable of loading from that format. The two main methods of displaying 3D polygon models are OpenGL and Direct3D. Both of these methods can be used with or without a 3D accelerated graphics card.

## Advantages and disadvantages

There are many disadvantages to representing an object using polygons. Polygons are incapable of accurately representing curved surfaces, so a large number of them must be used to approximate curves in a visually appealing manner. The use of complex models has a cost in lowered speed. In scanline conversion, each polygon must be converted and displayed, regardless of size, and there are frequently a large number of models on the screen at any given time. Often, programmers must use multiple models at varying levels of detail to represent the same object in order to cut down on the number of polygons being rendered.

The main advantage of polygons is that they are faster than other representations. While a modern graphics card can show a highly detailed scene at a frame rate of 60 frames per second or higher, surface modelers, the main way of displaying non-polygonal models, are incapable of achieving an interactive frame rate (10 frame/s or higher) with a similar amount of detail. With sprites, another alternative to polygons, every required pose must be created individually, while a single polygonal model can perform any movement if the appropriate motion data is applied, and can be viewed from any angle.[3]

## File formats

A variety of formats are available for storing 3D polygon data. The most popular are:

- .3ds, .max, which is associated with 3D Studio Max
- .blend, which is associated with Blender
- .c4d associated with Cinema 4D
- .dae (COLLADA)
- .dxf, .dwg, .dwf, associated with AutoCAD
- .fbx (Autodesk former. Kaydara Filmbox)
- .jt originally developed by Siemens PLM Software; now an ISO standard.
- .lwo, which is associated with Lightwave
- .lxo, which is associated with MODO
- .mb and .ma, which are associated with Maya
- .md2, .md3, associated with the Quake series of games
- .mdl used with Valve's Source Engine
- .nif (NetImmerse/gamebryo)

- .obj (Wavefront's "The Advanced Visualizer")
- .ply used to store data from 3D scanners
- .rwx (Renderware)
- .stl used in rapid prototyping
- .u3d (Universal 3D)
- .wrl (VRML 2.0)

## See also

- Finite element method
- Mesh generation
- Polygon (computer graphics)
- Polygon mesh
- Vector graphics
- Geometry processing
- 3D modeling

## References


1. "Primitive - OpenGL Wiki" (https://www.opengl.org/wiki/Primitive#Quads). *www.opengl.org*.
2. "Using a Basic Effect with Texturing" (http://msdn.microsoft.com/en-us/library/bb464051.aspx). *msdn.microsoft.com*.
3. Rybicki, Joe (December 1996). "The Making of NBA Live 97". *Electronic Gaming Monthly*. No. 89. Ziff Davis. p. 301.

1. OpenGL SuperBible (3rd ed.), by Richard S Wright and Benjamin Lipchak ISBN 0-672-32601-9
2. OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 1.4, Fourth Edition by OpenGL Architecture Review Board ISBN 0-321-17348-1
3. OpenGL(R) Reference Manual : The Official Reference Document to OpenGL, Version 1.4 (4th Edition) by OpenGL Architecture Review Board ISBN 0-321-17383-X
4. Blender documentation: https://web.archive.org/web/20051212074804/http://blender.org/cms/Documentation.628.0.html
5. Maya documentation: packaged with Alias Maya, http://www.alias.com/eng/index.shtml