Chapter 6

MODEL REPRESENTATION
AND SIMPLIFICATION
Introduction

- 3D scenes in graphics are composed of various shapes and structures:
  - Geometric primitives (spheres)
  - Free – form surfaces mathematically defined (NURBS patches)
  - Arbitrary surfaces mathematically undefined (surface of a scanned object)
  - Volume objects, where the internal structure of the object is equally important to its boundary surface (human organ)
  - Fuzzy objects (smoke)

- *Models* are approximate representations of the actual objects, constructed to retain many of the properties of the object

- Models are amenable to the manipulation required by graphics algorithms
• *Polygonal models* are the most common representation for surfaces

• Information contained in models produced is growing constantly

• Mainstream graphics applications often require or benefit from less detailed models

• *Model simplification* reduces the amount of information present in a model, without significantly sacrificing the quality of the representation
Overview of Model Forms

• There are two main categories of models:
  - *Surface representation* (or *boundary representation* or *b-rep*)
    - Represents only the surface of an object
  - *Volume representation* (or *space subdivision*)
    - Represents the whole volume that a closed object occupies

• Surface representations are used more frequently because:
  - Many objects are not closed → volume representation is not applicable
  - Majority of objects are not transparent → space and processing power is saved by only representing their surface, which determines their appearance

• Volume representations are used:
  - When displaying semi-transparent objects
  - When displaying objects whose internal structure is of interest
  - As auxiliary structures in general graphics algorithms
Overview of Model Forms (2)

**Some models cannot be easily classified into these two categories:**

- *Constructive Solid Geometry (CSG) models* represent an object by combining geometric primitives
- Amorphous objects and phenomena may be modeled as point clouds or by aggregating simple surface or volume primitives

**Surface models are classified:**

- To those that have some mathematical description such as:
  - Geometric primitives
  - NURBS surfaces
  - Subdivision surfaces
  - General parametric surfaces
- And those that do not have such a mathematical description:
  - Consist of a set of points and a set of planar (usually) polygons constructed with these points as vertices → *polygonal models*
Overview of Model Forms (3)

• Comparing the two surface model forms:
  ■ Mathematical models:
    ◆ Are usually exact representations of the respective objects
    ◆ Allow computations on object (e.g. normal vector) to be performed exactly
    ◆ Are limited to specific kind of objects
    ◆ Cannot describe arbitrary shapes
  ■ Polygonal models:
    ◆ Are approximations of the original objects
    ◆ Albeit very precise ones if enough vertices are used
    ◆ Are the most general
    ◆ Even mathematical representations are usually rendered in a “discrete” form as polygonal models
• Polygon models may consist of polygons of any number of vertices. In practice:
  ■ Quadrilaterals
  ■ Triangles

• Quadrilateral models:
  ■ Are naturally generated when rasterizing parametric surfaces
  ■ Unfortunately, a quadrilateral in 3D is not necessary planar:
    ☾ restricts the shape and flexibility of the model
    ☾ Even if planarity is enforced, the computations are difficult

• Triangle models:
  ■ A triangle is always planar
  ■ Any polygon may be triangulated efficiently $\rightarrow$ a triangle model can be generated from any other polygonal model $\rightarrow$ triangle models (or triangle meshes) are almost always preferred for any application involving polygonal models
Overview of Model Forms (5)

- Polygon models are generalized to polyhedral models for volume representation
- Most basic polyhedral primitive is the tetrahedron $\rightarrow$ tetrahedral meshes are the most general and flexible representation of volume models
- Models consisting of parallelepipeds are abundant, mainly as the outcome of space subdivision processes that use rectangular grids
- Constituent parallelepipeds are called voxels (volume elements)
- Hierarchical volume representations (octrees, BSP trees) are also used

- We will focus on polygonal models
Properties of Polygonal Models

• A surface model is a 2-manifold if every point on the surface has a neighborhood homeomorphic to an open disk (circle interior)
  ■ Even though the surface exists in 3D space, it is topologically flat when examined closely in a small area around any given point

• On a manifold surface:
  ■ Every edge is shared by exactly 2 faces
  ■ Around each vertex exists a closed loop of faces

• A surface model is a manifold with boundary if every point on the surface has a neighborhood homeomorphic to a half disk

• On a manifold with a boundary:
  ■ Some edges (those on the boundary) belong to exactly one face
  ■ Around some vertices (those on the boundary) the loop of faces is open

• For the usual, 3D surfaces, a manifold surface without boundary is a closed surface
Properties of Polygonal Models (2)

(a) Part of manifold surface
(b) Boundary vertex of a manifold surface with boundary
(c) Non manifold edge
(d) Non manifold boundary vertex

- A surface model is a *simplicial complex* if its constituting polygons meet only along their edges, and the edges of the model intersect only at their endpoints
Properties of Polygonal Models (3)

- A surface model is a *simplicial complex* if its constituting polygons meet only along their edges, and the edges of the model intersect only at their endpoints.

- (a) Simplicial triangle mesh  
  (b) non simplicial triangle mesh
Properties of Polygonal Models (4)

- **Orientable surface**: surface that has 2 “sides”, like a sheet of paper
- Most of the surfaces are orientable
- On closed orientable surfaces the “external” and “internal” portions of the surface are distinguishable
- By convention, the normal vector of a closed orientable surface points towards “outside”
- The Moebius strip is a non–orientable surface:
Properties of Polygonal Models (5)

- Closed manifold models homeomorphic to a sphere satisfy Euler’s formula:

\[ V - E + F = 2 \]

where:

\[ \begin{align*}
V & : \text{# of vertices} \\
E & : \text{# of edges} \\
F & : \text{# of faces}
\end{align*} \]

of the model
Properties of Polygonal Models (6)

- For a closed triangular model the formula reveals:
  - That the number of triangles of the model is almost twice the number of its vertices
  - That the average number of triangles around each vertex is 6
- Euler’s formula has been generalized for arbitrary manifold models:
  \[ V - E + F = 2 - 2G \]
  where \( G \) is the genus of the model
- The genus of a model can be considered as the number of the penetrating holes of the model:
  - Torus has genus 1
  - Double torus has genus 2, and so on
Data Structures for Polygonal Models

• Several different data structures have been proposed for representation of polygon models; they differ:
  ■ In the type of polygon models that they are able to represent
  ■ In the amount and type of information that they capture directly about the model
  ■ In other information that can or cannot be derived indirectly from them about the model
• Useful information for several graphics operations is:
  ■ *Topological information*: whether the model is manifold, closed, has a boundary or holes
  ■ *Adjacency information*: neighboring faces of given edge and face, edges and faces around a given vertex, the boundary of an open model
  ■ *Attributes attached to the model*: normal vector, colors, material properties, texture coordinates
Data Structures for Polygonal Models (2)

- Most primitive data structures that were used:
  - *Explicit list of edges* contain for each edge/face of the model the coordinates of its vertices
  - *Explicit list of faces*

**EXAMPLE:**

- For the tetrahedron beside it holds:
  - List of edges:
    \[
    e_0 = (x_0, y_0, z_0), (x_1, y_1, z_1), e_3 = (x_1, y_1, z_1), (x_2, y_2, z_2),
    e_1 = (x_0, y_0, z_0), (x_2, y_2, z_2), e_4 = (x_1, y_1, z_1), (x_3, y_3, z_3),
    e_2 = (x_0, y_0, z_0), (x_3, y_3, z_3), e_5 = (x_2, y_2, z_2), (x_3, y_3, z_3)
    \]
  - List of faces:
    \[
    f_0 = (x_3, y_3, z_3), (x_2, y_2, z_2), (x_1, y_1, z_1),
    f_1 = (x_2, y_2, z_2), (x_3, y_3, z_3), (x_0, y_0, z_0),
    f_2 = (x_1, y_1, z_1), (x_0, y_0, z_0), (x_3, y_3, z_3),
    f_3 = (x_0, y_0, z_0), (x_1, y_1, z_1), (x_2, y_2, z_2)
    \]
Data Structures for Polygonal Models (3)

• List of edges:
  - Is not a b-rep
  - Does not specify the faces of the model
  - Faces must be inferred from the edge data → may lead to ambiguities

• List of faces:
  - The coordinates of each vertex are repeated for each edge or face containing it → wastes space
  - Provides no information on the adjacency of the faces and edges
  - Common vertices can only be detected by comparing coordinates → numerical accuracy problems may interfere → computing adjacency can be problematic
Several of the above drawbacks are addressed by the *indexed list of faces*:

- Contains a list of the vertices of the model and a list of its faces
- The vertices of its faces are given as references to the list of vertices

For instance, the previous tetrahedron is represented as:

\[
\begin{align*}
    v_0 &= (x_0, y_0, z_0), & f_0 &= (v_3, v_2, v_1), \\
    v_1 &= (x_1, y_1, z_1), & f_1 &= (v_2, v_3, v_0), \\
    v_2 &= (x_2, y_2, z_2), & f_2 &= (v_1, v_0, v_3), \\
    v_3 &= (x_3, y_3, z_3), & f_3 &= (v_0, v_1, v_2)
\end{align*}
\]

To represent orientable models, using indexed list of faces, it is customary to list the vertices of all faces either clockwise or counterclockwise \(\rightarrow\) easier to make computations on the model.
Indexed list of faces:
- Can represent any kind of polygon model
- Permits direct modifications to the positions of the vertices of the model
- Edges of the model are straightforward to discover but they are repeated for each polygon that uses them
- Processing is required in order to generate a valid list of unique edges
- Does not provide adjacency information although the data it contains is sufficient to compute it

Specifically for triangle models, neighboring triangles are handled more efficiently as triangle strips or triangle fans, in order to minimize data duplication
Specifically for triangle models, sets of neighboring triangles are handled more efficiently as *triangle strips* or *triangle fans*, in order to minimize data duplication.

**Triangle strip**

\[(v_0, v_1, v_2, v_3, v_4)\]

**Triangle fan**

\[(v_0, v_1, v_2, v_3, v_4)\]
Data Structures for Polygonal Models (7)

- Indexed list of faces can be combined with other indexed data for the attributes of the model bound to either vertices or faces:
  - i.e. color

- More advanced data structures can capture directly some adjacency information and allow for easy derivation of more adjacency relations

- These data structures are indexed, contain at least a list of vertices and deal with manifold models
Data Structures for Polygonal Models (8)

- **Winged-edge**, is one such data structure:
  - Central node of information is the edge
  - Each edge stores references:
    - To its 2 vertices
    - To its 2 adjacent faces
    - To its 4 neighboring edges
  - For each vertex a reference to one of its incident edges is stored
  - For each face a reference to one of its edges is stored
  - Possible to “navigate” in the topology of the model and compute adjacent queries efficiently
  - Winged-edge can be modified in order to represent some types of non-manifold models
• *Half-edge* data structure is similar to winged-edge representation, but uses oriented edges:
  - Each edge is “decomposed” into 2 half-edges
  - Each half-edge stores references:
    - To its start and end vertex
    - To its adjacent face
    - To its 2 neighboring half-edges along the adjacent face
    - To its opposite half-edge
  - Half-edge data structure is more efficient than winged-edge for several adjacency queries
Data Structures for Polygonal Models (10)

- **Quad-edge** data structure is similar to the above representations:
  - Its implementation is more sophisticated
  - Can be used to compute adjacency queries efficiently
  - Can represent simultaneously a manifold model and its *dual*
    - Dual of a model is constructed by rotating edges by 90°, replacing the vertices with faces and vice versa
      i.e. dual of a tetrahedron is a tetrahedron
    dual of a cube is an octahedron and vice versa
  - Useful in the context of *computational geometry*, the algorithmic study of geometric problems
Polygonal Model Simplification

- Polygonal models used in practice are produced automatically by:
  - Rasterization of mathematically defined surfaces
  - 3D scanning of real objects
  - Other similar procedures

- The steady increase of computer power and the advances of 3D scanning lead to models that capture finest details, with larger number of vertices and faces

- *Digital Michelangelo* project, for example, used high-tech scanners to scan and reconstruct some Michelangelo sculptures
  - The triangle meshes produced contain several hundred million triangles
  - They occupy several gigabytes of data storage
  - This amount of information is difficult to process
  - This amount of detail is only useful in specific applications
• Computer graphics applications can benefit from multiple resolutions (levels of detail (LODs)) of the model that can be used in different viewpoint conditions:
  - When screen projection of a model is small, only a small amount of detail is discernible
• It would also be beneficial to vary the detail in different parts of the model:
  - Coplanar triangles could be merged into fewer and larger ones
  - Areas of the surface closer to the viewer would require more detail than those further away
• LODs and selective detail, explained above, are suitable for interactive applications displaying large graphics scenes
Different LODs and sizes: (5000 vs. 1000 triangles)

For the above reasons, several model simplification techniques have been developed.

They try to reduce number of faces of a polygonal model while retaining the appearance and structure of the original model.

Simplified models applications, usually employ several LODs of the original model and dynamically select the most suitable
Polygonal Model Simplification (4)

- Model simplification techniques vary greatly in many respects:
  - Can be applied to different kinds of models
  - Take different paths for simplification of the models
  - Have different priorities and applications

- Simplification algorithms:
  - Deal most easily with closed manifold meshes
  - Handle, in most cases, the boundary of non-closed models
  - Only few, are able to simplify non-manifold models
Polygonal Model Simplification (5)

- Simplification methods can be classified in two main classes:
  - Those that produce discrete levels of detail of the initial model
  - Those that produce continuous levels of detail of the initial model

- Discrete levels of detail:
  - A target number of faces is prescribed
  - New model with the required number of faces is generated
  - If another level of detail is requested, the algorithm is executed again

- Continuous level of details:
  - A continuous sequence of increasingly simplified models is produced using local simplifications of the model
  - By recording the simplification steps, any intermediate level of detail may be produced
Polyhedral Model Simplification (6)

- Continuous simplification algorithms are more interesting than discrete ones.
- They are flexible and easily reversible ➔ allow the application to move back and forth between levels of details.
- Support the selective refinement, enabling dynamic adjustment of detail in different parts of the model.
- Refine the mesh smoothly ➔ minimizes visual artifacts due to switching resolution of the model in interactive applications.
Polygonal Model Simplification (7)

- Important issue for all simplification methods: *how to assess the quality of simplified model with respect to the original one*

- Most algorithms are guided by certain criteria in order to determine:
  - Where the “best” to put the new vertex is
  - Which edge should be removed first in order to minimize discrepancy

- These criteria also provide a global estimate of the quality of the algorithms → simplification algorithms can be compared
Polygonal Model Simplification (8)

• Most widely used methods for this assessment is to measure some form of distance between the simplified and the original model:

  ■ *Hausdorff distance*: measures the maximum distance between any 2 points of 2 surfaces $M$ and $M'$

    \[ d_\infty(M, M') = \max(\max\{d(v, M')\}, \max\{d(v', M)\}), \]

    where $d(v, M) = \min\{|v - w|\}$ is the distance of a point $v$ from a surface $M$, defined as the distance of $v$ from the closest point $w$ of the surface

  ■ *Mean square distance* of 2 surfaces:

    \[ d_2(M, M') = \frac{1}{s} \int \limits_{v \in M} d(v, M') + \frac{1}{s'} \int \limits_{v' \in M'} d(v', M), \]

    where $s$ and $s'$ are the areas of $M$ and $M'$ respectively

  ■ These formulae must be discretized in order to be computed on polygonal models → accomplished by sampling a number of points on both surfaces and using them for the computations
Simplification using Iterative Edge Collapses

- **Edge collapse:**
  - Local operation on a triangle mesh
  - Removes an edge of the model and the 2 adjacent triangles by collapsing an edge to a single vertex

- Using edge collapses, it is easy to compute the distance between the simplified and the original mesh $\rightarrow$ only difference on the faces around the collapsed edge
- Variations of this method support non-manifold models
- We will concentrate on manifold models
Simplification using Iterative Edge Collapses (2)

• The algorithm is summarized as follows:
  1. For each edge of the model that can be collapsed, compute a collapse priority and sort the edges in a priority queue
  2. While more candidate edges exist in the queue and the simplification target (maximum error, number of faces of the mesh) is not reached:
     a) Remove from the queue the edge collapse with highest priority
     b) Collapse this edge (mesh only changes locally around the edge)
     c) Re-compute the priorities of all edges affected by the collapse

• Two factors that affect the result of this method are:
  ■ The measure used to assess each edge collapse and assign its priority
  ■ The position of the new vertex for each edge collapse

• Different techniques have been proposed for the above 2 elements of the method
Simplification using Iterative Edge Collapses (3)

- In some implementations, the position of the new vertex is fixed.
- In other implementations, the above 2 factors are interrelated:
  - The position of the new vertex is computed as a result of an optimization procedure, minimizing the approximation error.
  - The maximum error attained is used as the priority of the edge collapse.
Simplification using Iterative Edge Collapses (4)

- **Quadric error-metric** method:
  - minimizes the square distance of the new vertex from the faces around the collapsed edge
  - Let $\Delta$ be a triangular face of a model with plane equation:
    \[
    ax + by + cz + d = 0
    \]
  - Squared distance of a point $\mathbf{x} = [x, y, z]^T$ from the plane of $\Delta$ is:
    \[
    Q_\Delta(\mathbf{x}) = \frac{(ax + by + cz + d)^2}{a^2 + b^2 + c^2} = \frac{(\hat{n}^T \mathbf{x} + \hat{d})^2}{|\hat{n}|^2} = (\hat{n}^T \mathbf{x} + \hat{d})^2 = \mathbf{x}^T (\hat{n}n^T) \mathbf{x} + 2\hat{d} \hat{n}^T \mathbf{x} + \hat{d}^2,
    \]
    where $\hat{n} = \frac{\mathbf{n}}{|\mathbf{n}|}$ is the unit normal vector of $\Delta$ and $\hat{d} = \frac{d}{|\mathbf{n}|}$
  - It can also be represented by the quadratic form:
    \[
    Q_\Delta = (A, b, p) = (\hat{n}n^T, \hat{d} \hat{n}, \hat{d}^2),
    \]
    so that:
    \[
    Q_\Delta(\mathbf{x}) = \mathbf{x}^T A \mathbf{x} + 2b^T \mathbf{x} + p
    \]
Simplification using Iterative Edge Collapses (5)

- The sum of the squared distances of $x$ from 2 triangles $\Delta 1$ and $\Delta 2$ can be computed by summing coordinate-wise the quadratic forms:
  \[ Q_{\Delta 1} = (A_1, b_1, p_1) \quad \text{and} \quad Q_{\Delta 2} = (A_2, b_2, p_2) : \]
  \[ Q_{\Delta 1}(x) + Q_{\Delta 2}(x) = (Q_{\Delta 1} + Q_{\Delta 2})(x) = x^T (A_1 + A_2)x + 2(b_1 + b_2)^T x + (p1 + p2) \]

- The above is also a quadratic form
- This result generalizes naturally to any number of triangles
- The simplification algorithm assigns initially to each vertex $v$ of the mesh, the form that expresses the sum of squared distances of a point from the faces around the vertex:
  \[ Q_v = \sum_{\Delta \text{arround } x} w_\Delta Q_\Delta \]

  where $w_\Delta$ is a surface weight of the respective face $\rightarrow$ better scaling
Simplification using Iterative Edge Collapses (6)

- Then, when an edge $e(v_0, v_d)$ is collapsed, the total squared distance of the resulting vertex $v_s$ from all the faces around $v_0$ and $v_d$ is:
  
  $$Q(v_s) = Q_{v_0}(v_s) + Q_{v_d}(v_s) \quad \text{or} \quad Q = Q_{v_0} + Q_{v_d}$$

  which is the familiar form $Q = (A, b, p)$

- The position that minimizes $Q$ is the optimal for $v_s$

- Minimum of $Q$ is attained at $v_s = A^{-1}b$ and the minimum is:
  
  $$Q(v_s) = -b^T A^{-1}b + p = b^T v_s + p$$

- If $A$ is a singular matrix $\Rightarrow$ minimization is restricted along the edge $e(v_0, v_d)$

- If this fails, $v_s$ is selected between $v_0$ and $v_d$, depending on which vertex gives smaller value for $Q$
Simplification based on iterative edge collapses has all the properties of continuous level of detail methods:

- It is easily reversible by performing *vertex split* in reverse order to the corresponding edge collapses \( \rightarrow \) original positions must be kept with each edge collapse
- By retaining some more information on the neighboring vertices and faces of each collapsed edge, it is possible to perform selective refinement and coarsening of the mesh on region of interest
- Various error metrics and vertex-positioning strategies may be employed, so the method can be adapted to various interests and available resources
Simplification using Iterative Edge Collapses (8)

• The simplification of large models is a lengthy operation
  ■ If an optimization procedure is used, it is even more costly → typically performed offline
  ■ At run time, generated levels of detail can be exploited interactively in real time for selectively refining the model

• Simplification based on edge collapses is becoming a standard feature in several graphics packages (e.g. DirectX)