Java 3D – Building Shape

Winter 2003
Defining vertices

- A vertex describes a polygon corner and contains:
  - 3D coordinate
  - Color
  - Texture coordinate
  - Lighting normal vector (must be unit length)

- The 3D coordinate in a vertex is required, the other parameters are optional
Building geometry

- All geometry types are derived from class Geometry
- There are 14 different geometry array types grouped into:
  - Simple geometry: PointArray, LineArray, TriangleArray, and QuadArray
  - Strip geometry: LineStripArray, TriangleStripArray, and TriangleFanArray
  - Indexed simple geometry: IndexedPointArray, IndexedLineArray, IndexedTriangleArray, and IndexedQuadArray
  - Indexed stripped geometry: IndexedLineStripArray, IndexedTriangleStripArray, and IndexedTriangleFanArray
Building 3D Primitives

- Building a PointArray
  - A PointArray builds points one point at a time at each vertex
  - Point size may be controlled by shape appearance attributes

- Building a LineArray
  - A LineArray builds lines one line at a time between each pair of vertices
  - Line width and style may be controlled by shape appearance attributes

- Building a TriangleArray
  - A TriangleArray builds triangles one triangle at a time between each triple of vertices
  - Rendering may be controlled by shape appearance attributes
PointArray Example

- Create a list of 3D coordinates for the vertices
  
  ```java
  Point3f[] myCoordinates = {
      new Point3f(0.0f, 0.0f, 0.0f),
      ...  
  }
  ```

- Create a PointArray and set the vertex coordinates
  
  ```java
  PointArray myPoints = new PointArray(myCoordinates.length, GeometryArray.COORDINATES);

  myPoints.setCoordinates(0, myCoordinates);
  ```

- Assemble the shape
  
  ```java
  Shape3D myShape = new Shape3D(myPoints, myAppearance);
  ```
LineArray Example

- Create a list of 3D coordinates for the vertices
  
  ```java
  Point3f[] myCoordinates =
  {
      new Point3f(0.0f, 0.0f, 0.0f),
      ...  
  }
  ```

- Create a LineArray and set the vertex coordinates
  
  ```java
  LineArray myLines = new LineArray(myCoordinates.length,
      GeometryArray.COORDINATES
  );
  myLines.setCoordinates(0, myCoordinates);
  ```

- Assemble the shape
  
  ```java
  Shape3D myShape = new Shape3D(myLines, myAppearance);
  ```
**TriangleArray Example**

- Create lists of 3D coordinates and normals for the vertices
  ```java
  Point3f[] myCoordinates = {
    new Point3f(0.0f, 0.0f, 0.0f), . . .
  }
  Vector3f[] myNormals = {
    new Vector3f( 0.0f, 1.0f, 0.0f ), . . .
  }
  ```

- Create a TriangleArray and set the vertex coordinates and normals
  ```java
  TriangleArray myTriangle = new TriangleArray(myCoordinates.length,
  GeometryArray.COORDINATES|GeometryArray.NORMALS);
  myTriangle.setCoordinates(0, myCoordinates);
  myTriangle.setNormals(0, myNormals);
  ```

- Assemble the shape
  ```java
  Shape3D myShape = new Shape3D(myTriangle, myAppearance);
  ```
Building Simple and Strip Geometry

- Simple geometry use
  - Vertices in pairs, triples, and quadruples to build lines, triangles, and quadrilaterals one at a time

- Strip geometry use
  - Multiple vertices in a chain to build multiple lines and triangles
  - We must provide a coordinate list, lighting normal, color, and optionally texture coordinate lists
  - We must provide a strip length list
  - Each list entry gives the number of consecutive vertices to chain together
Building indexed geometry

Indexed geometry use

- Indices are used along with the lists of coordinates, lighting normals, color and texture coordinates
- Indices select which coordinates to use from each list
- Indices are also used for lighting normals, colors, and texture coordinates
- For surfaces, the same vertices are reused for adjacent lines and triangles, providing an efficient use of vertex information
- Simple and strip geometry require redundant coordinates, lighting normals, colors, and texture coordinates
- No redundant coordinates in indexed geometry
Summary

- A 3D shape is described by:
  - Geometry that describes form and structure
  - Appearance that describe coloration, transparency, and shading

- Java 3D has multiple geometry types that all use vertices with:
  - Coordinates: 3D xyz locations
  - Normals: 3D direction vectors
  - Colors: RGB colors mix
  - Texture coordinates: 2D texture image locations

- Simple geometry build points, lines, triangles, and quadrilaterals automatically using vertices in sets of 1, 2, 3, or 4

- Strip geometry build lines and triangles using vertices in user-defined chains

- Indexed geometry build points, lines, triangles, and quadrilaterals using coordinates, lighting normals, color, and texture coordinates selectable by indices
Appendix: J3DCube Example

Building 3D Shapes