Computer Game Technologies

Java 3D Object Modelling - Appearance

Appearance in Java 3D

- To be visible an object must be coloured
- Requires specifying a colour for each vertex
- Non-vertex pixel colours determined by a *shading model*
- Many ways to specify vertex colours in Java 3D
  - Geometry node component
  - Appearance node component
  - Material node component
    - Final colours determined by scene lighting

Using Geometry Node Component

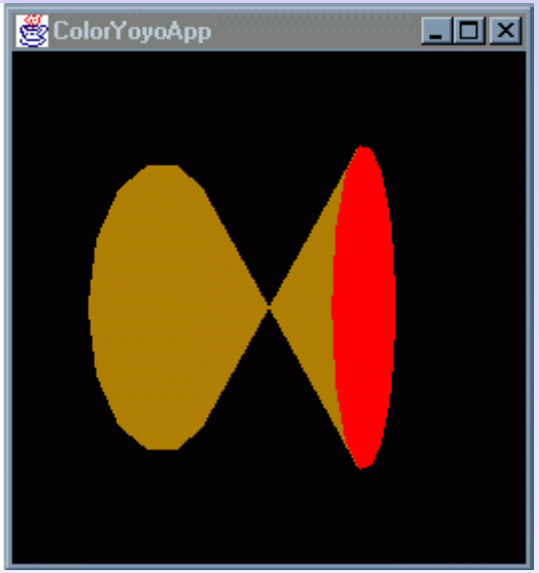
- Specify vertex colours: Yo-Yo example

```
1. private Geometry yoyoGeometry() {
2.
3. TriangleFanArray tfa;
4. int N = 17;
5. int totalN = 4*(N+1);
6. Point3f coords[] = new Point3f[totalN];
7. Color3f colors[] = new Color3f[totalN];
8. Color3f red = new Color3f(1.0f, 0.0f, 0.0f);
9. Color3f yellow = new Color3f(0.7f, 0.5f, 0.0f);
10. int stripCounts[] = {N+1, N+1, N+1, N+1};
11. float r = 0.6f;
12. float w = 0.4f;
13. int n;
14. double a;
15. float x, y;
16.
```

Yo-Yo Example (2)

```
17. // set the central points for four triangle fan strips
18. coords[0*(N+1)] = new Point3f(0.0f, 0.0f, w);
19. coords[1*(N+1)] = new Point3f(0.0f, 0.0f, 0.0f);
20. coords[2*(N+1)] = new Point3f(0.0f, 0.0f, 0.0f);
21. coords[3*(N+1)] = new Point3f(0.0f, 0.0f, -w);
22.
23. colors[0*(N+1)] = red;
24. colors[1*(N+1)] = yellow;
25. colors[2*(N+1)] = yellow;
26. colors[3*(N+1)] = red;
27.
28. for(a = 0,n = 0; n < N; a = 2.0*Math.PI/(N-1) * ++n){
29. x = (float) (r * Math.cos(a));
30. y = (float) (r * Math.sin(a));
31. coords[0*(N+1)+n+1] = new Point3f(x, y, w);
32. coords[1*(N+1)+N-n] = new Point3f(x, y, w);
33. coords[2*(N+1)+n+1] = new Point3f(x, y, -w);
34. coords[3*(N+1)+N-n] = new Point3f(x, y, -w);
35.
```

Yo-Yo Example (3)




```
36. colors[0*(N+1)+N-n] = red;
37. colors[1*(N+1)+n+1] = yellow;
38. colors[2*(N+1)+N-n] = yellow;
39. colors[3*(N+1)+n+1] = red;
40. }
41. tfa = new TriangleFanArray (totalN,
42. TriangleFanArray.COORDINATES|TriangleFanArray.COLOR_3,
43. stripCounts);
44.
45. tfa.setCoordinates(0, coords);
46. tfa.setColors(0,colors);
47.
48. return tfa;
49. } // end of method yoyoGeometry in class Yoyo
```

Appearance Bundle

- Appearance node component
  - Does not contain appearance information itself
  - References other *Appearance Attribute Node Components* that specify how the visual object will look
- Appearance bundle
  - Appearance node component referencing various appearance attribute node components

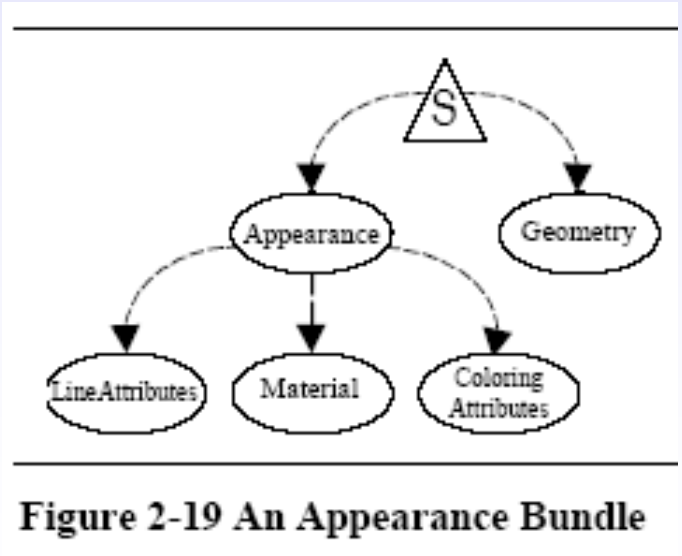


Figure 2-19 An Appearance Bundle

Appearance Bundle (2)

- Appearance attribute node components
  - PointAttributes
  - LineAttributes
  - PolygonAttributes
  - ColoringAttributes
  - TransparencyAttributes
  - RenderingAttributes

  - Material
  - TextureAttributes
  - Texture
  - TexCoordGeneration

Creating an Appearance Bundle

- Appearance with colour attributes

```
1. ColoringAttributes ca = new ColoringAttributes();
2. ca.setColor (1.0, 1.0, 0.0);
3. Appearance app = new Appearance();
4. app.setColoringAttributes(ca);
5. Shape3D s3d = new Shape3D();
6. s3d.setAppearance (app);
7. s3d.setGeometry (someGeomObject);
```

Code Fragment 2-9 Using Appearance and ColoringAttributes NodeComponent Objects

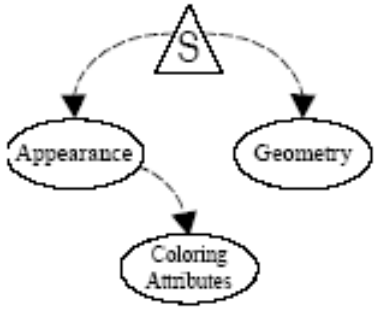


Figure 2-20 Appearance Bundle Created by Code Fragment 2-9.

Sharing Node Components

- Multiple Appearance objects can share attribute components
  - May improve rendering performance

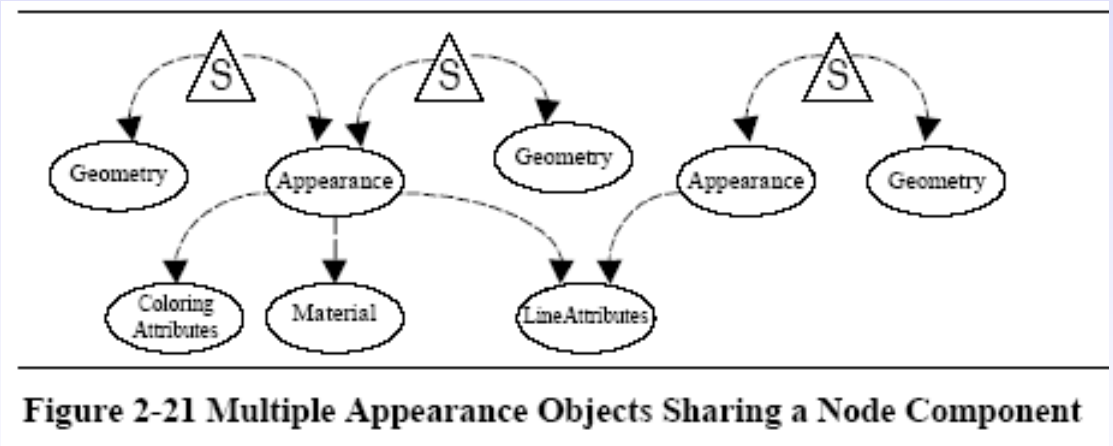
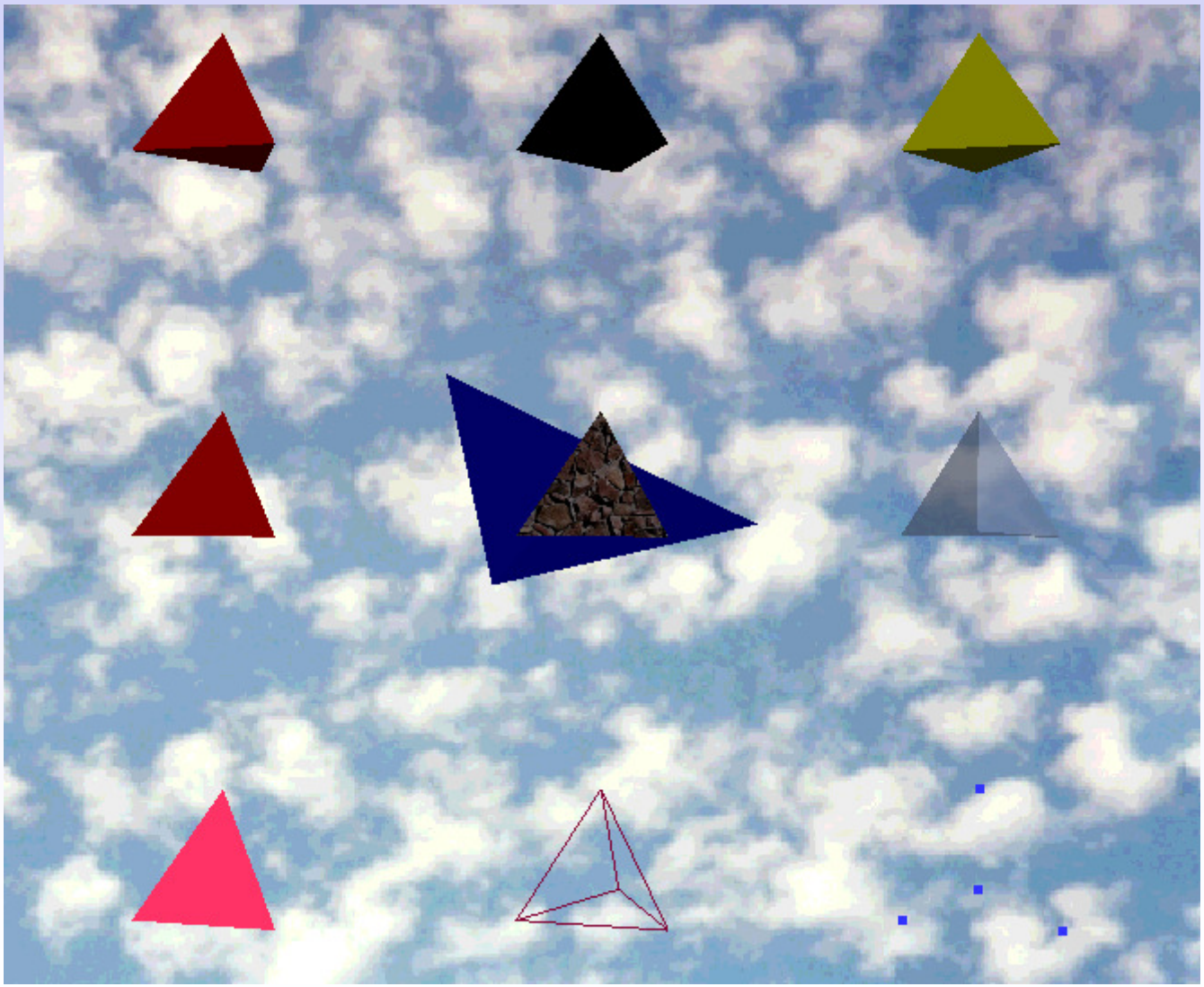


Figure 2-21 Multiple Appearance Objects Sharing a Node Component

Attribute Classes

- Point attributes
  - Vertex rendered as a single pixel by default
  - Can increase size of square "point"
  - Specify antialiasing to make points look more rounded
- Line attributes
  - Solid, one pixel wide, not antialiased by default
  - Can specify style, width and antialiasing
  - E.g. dash, dot, dash-dot
- Polygon attributes
  - Drawn filled with back-face-culling by default
  - Can specify wireframe rendering
  - Can specify front-face or no culling

Examples



Face Culling Revisited

- Front face of a polygon is determined by vertex ordering
  - Counter clockwise
  - (right-hand rule)

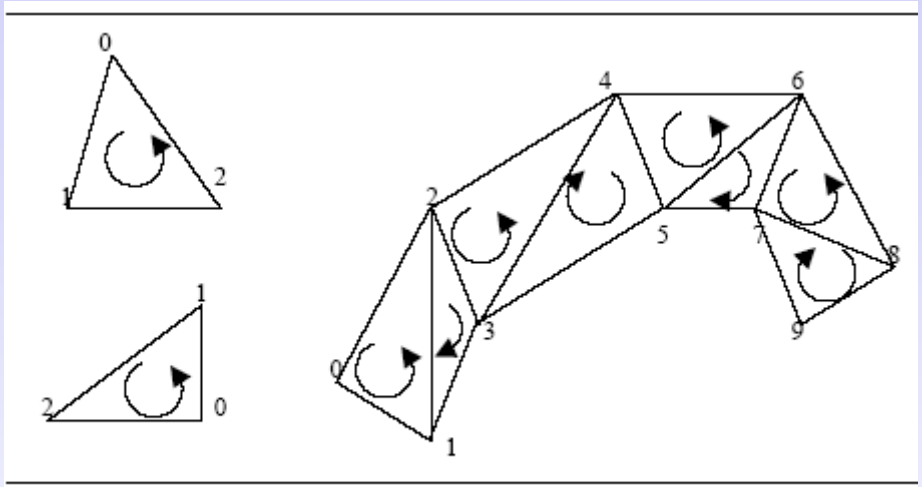
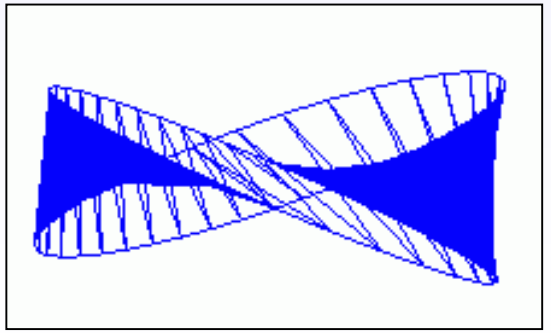


Figure 2-24 Determining the Front Face of Polygons and Strips

- Back-face culling is default setting
- No-face or front-face culling may sometimes be more appropriate
  - E.g checkerboard floor
  - Twisted strip



Attribute Classes (2)

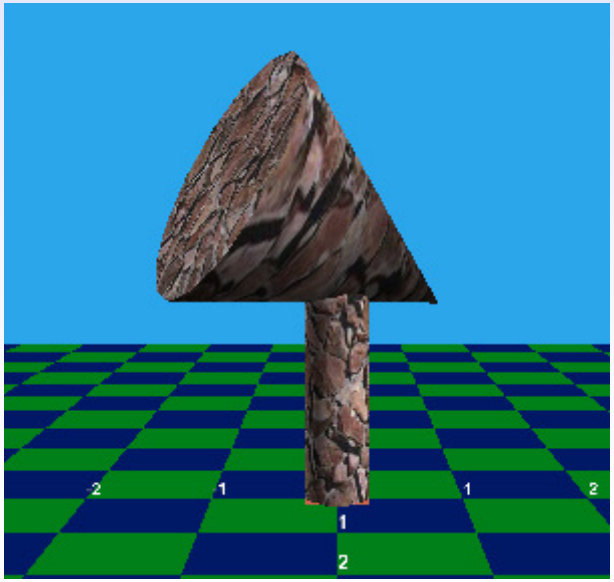
- Coloring attributes
  - Specify vertex colours
  - Specify how pixels between vertices are coloured using interpolation of vertex colours
    - Flat or Gouraud shading
  - Colours specified in Geometry node component will override an Appearance coloring attribute
  - Appearance coloring attribute also ignored if scene is lit
    - Colours determined by Material plus lights
    - Lighting to be discussed later

Attribute Classes (3)

- Transparency attributes
  - Specifies transparency of the visible object via an alpha level
- Rendering attributes
  - Controls per pixel rendering
  - Depth test: determines whether depth buffer is used for hidden surface removal
    - Z-buffering
  - Alpha test: determines whether alpha is used to give transparency

Other Attribute Classes

- Material
  - Specifies an object's intrinsic colour and how it is affected by lights e.g. shininess
- Texture attributes
  - Possible to use a 2D image to colour an object
  - Texture mapping



Materials and textures to be looked at in more detail later

The End