Computer Game Technologies

Java 3D Object Modelling - Geometry

Modelling in Java 3D

Java 3D provides 3 ways to create physical objects:

1. Basic geometric primitives
   - ColorCube
   - Box, Cone, Cylinder, Sphere
2. Vertex specification via geometry classes
   - TriangleArray, QuadArray
3. Loading external models
   - Model created via specialist software eg. Blender
   - Loading software to convert geometry specification to Java 3D format

Shape3D Class

- Instance of Shape3D defines a visual object
  - Subclass of Leaf class
  - Forms a leaf node in a scene graph
  - Does not itself define the geometry or appearance of an object
- Geometry specified in a Geometry node component
- Appearance specified in an Appearance node component
- A Shape3D object refers to at most one Geometry and one Appearance node object

Example Scene Graph

- Shape3D node S references an Appearance and Geometry

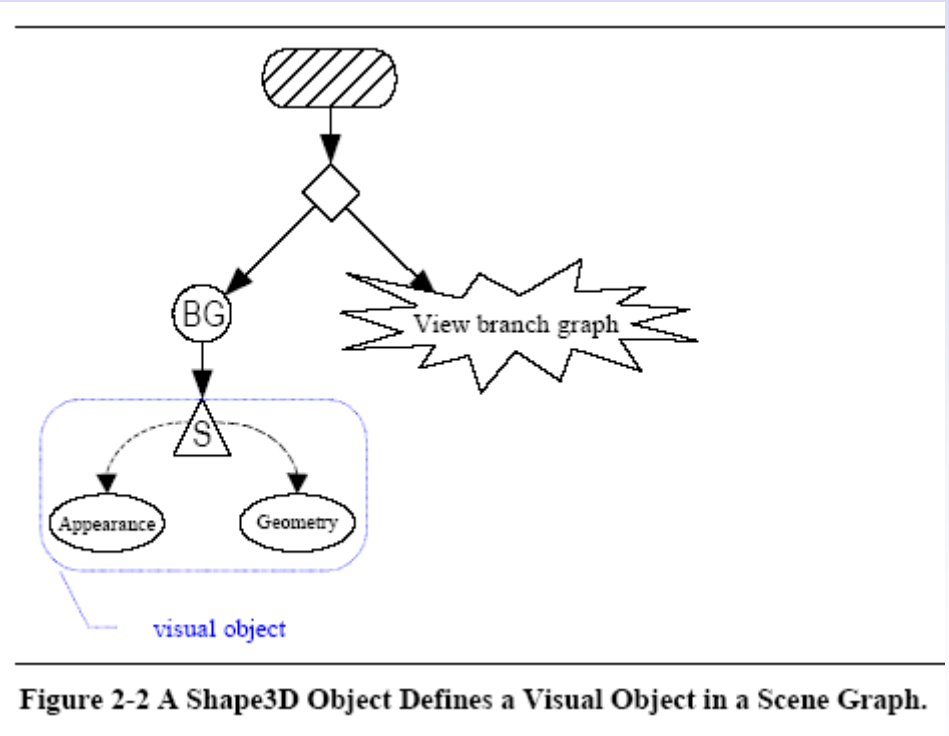


Figure 2-2 A Shape3D Object Defines a Visual Object in a Scene Graph.

Leaf Nodes and Node Components

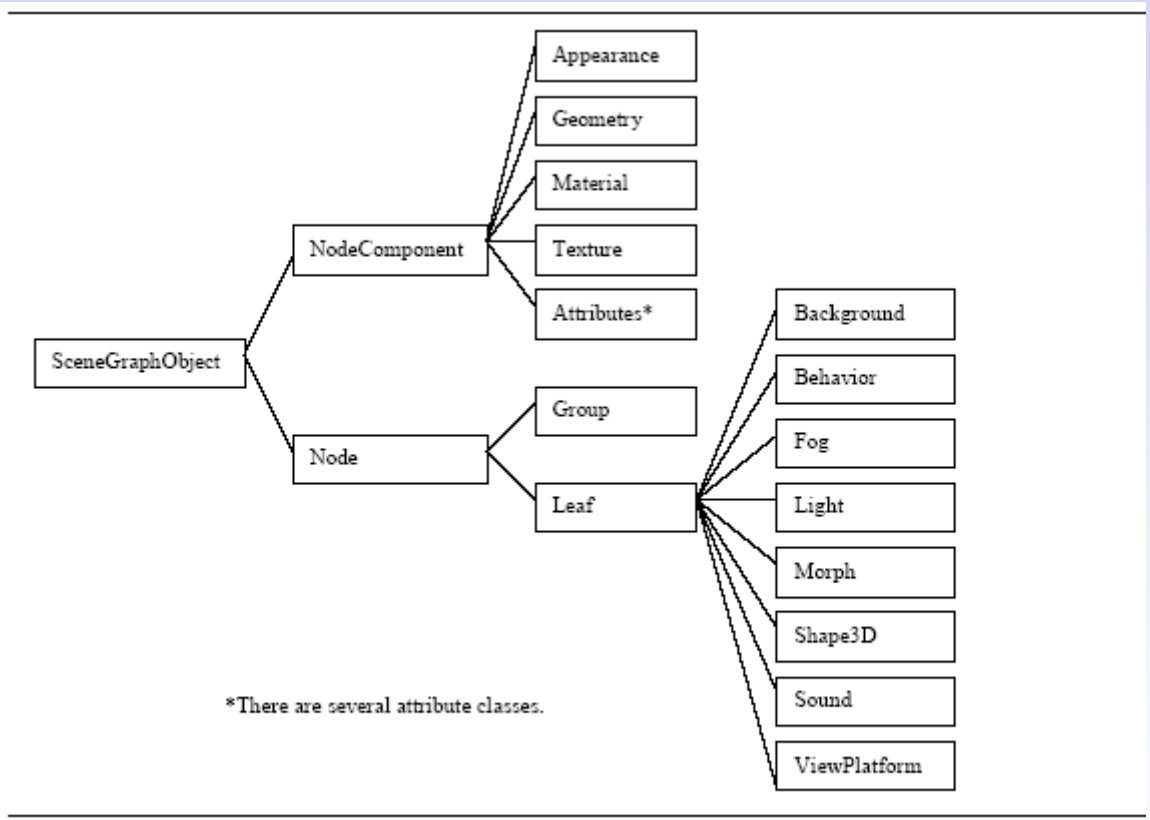


Figure 2-3 Partial Java 3D API Class Hierarchy Showing Subclasses of NodeComponent.

Defining a Visual Object Class

- One possibility for creating your own visual object class is to extend Shape3D

```
public class VisualObject extends Shape3D{

 private Geometry voGeometry;
 private Appearance voAppearance;

 // create Shape3D with geometry and appearance
 // the geometry is created in method createGeometry
 // the appearance is created in method createAppearance
 public VisualObject() {
   voGeometry = createGeometry();
   voAppearance = createAppearance();
   this.setGeometry(voGeometry);
   this.setAppearance(voAppearance);
 }
```

(Java 3D Tutorial Chapt 2)

Defining a Visual Object Class (2)

```
private Geometry createGeometry() {
// code to create default geometry of visual object
}

private Appearance createAppearance () {
// code to create default appearance of visual object
}

} // end of class VisualObject
```

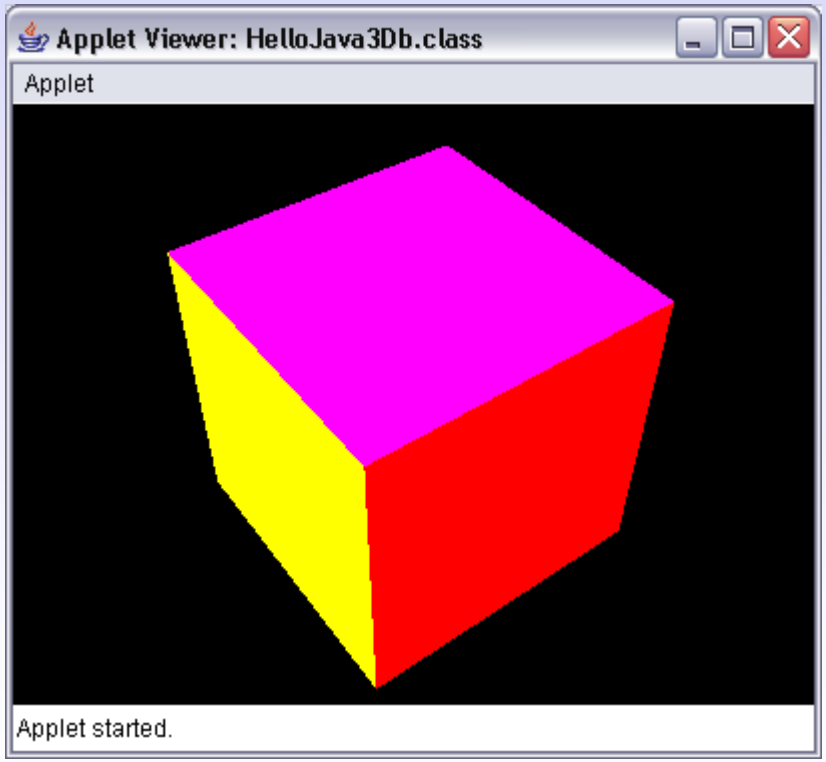
- Simply create object and add it to a group node

```
ojbRoot.addChild(new VisualObject());
```

(Java 3D Tutorial Chapt 2)

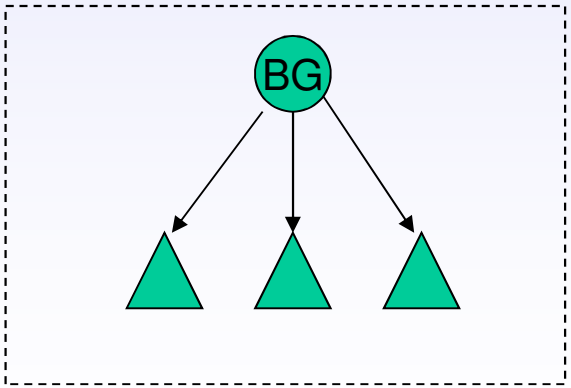
ColorCube Class

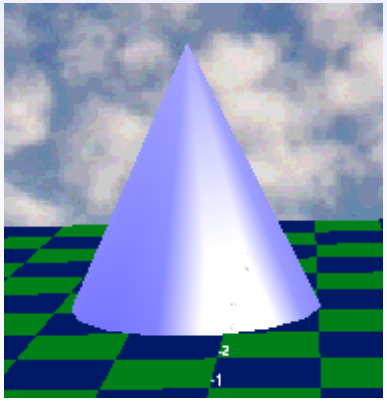
- ColorCube is an example of a class that extends Shape3D



More General Visual Object Classes

- Extending Shape3D only allows an object to be defined by a single Geometry
- What if we want to have a more complex object that is made up of subobjects?
- Define a class that contains a branch group that holds more than one Shape3D
  - Single visible object composed of more than one Shape3D





General Visual Object Class

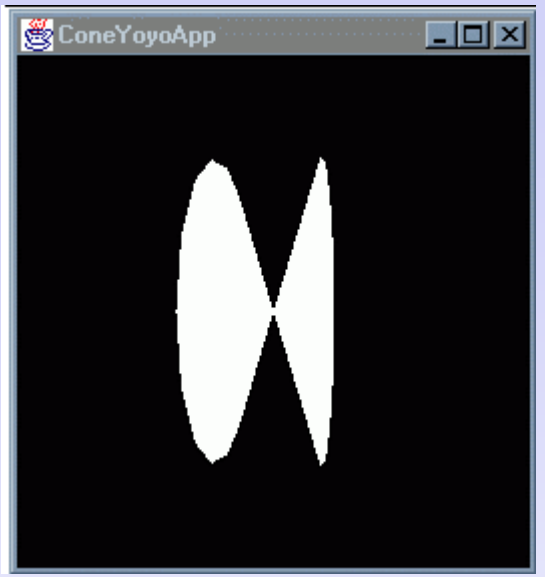
```
public class VisualObject {

 private BranchGroup voBG;

 public VisualObject() {
   voBG = new BranchGroup();
   // create subobjects
   ColorCube cube1 = new ColorCube();
   ColorCube cube2 = new ColorCube();
   // add subobjects to branch group
   voBG.addChild(cube1);
   voBG.addChild(cube2);
   // optimise scene graph
   voBG.compile();
 }

 public BranchGroup getBG() {
   return voBG;
 }
}
```

Example: a Yo-Yo




```
public class ConeYoyo{

 private BranchGroup yoyoBG;

 public ConeYoyo() {

   yoyoBG = new BranchGroup();

   Transform3D rotate = new Transform3D();
   Transform3D translate = new Transform3D();
   Appearance yoyoAppear = new Appearance();

   rotate.rotZ(Math.PI/2.0d);
   TransformGroup yoyoTGR1 = new TransformGroup(rotate);

   translate.set(new Vector3f(0.1f, 0.0f, 0.0f));
   TransformGroup yoyoTGT1 = new TransformGroup(translate);
```

Example: a Yo-Yo (2)

```
Cone cone1 = new Cone(0.6f, 0.2f);
cone1.setAppearance(yoyoAppear);

yoyoBG.addChild(yoyoTGT1);
yoyoTGT1.addChild(yoyoTGR1);
yoyoTGR1.addChild(cone1);

translate.set(new Vector3f(-0.1f, 0.0f, 0.0f));
TransformGroup yoyoTGT2 = new TransformGroup(translate);

rotate.rotZ(-Math.PI/2.0d);
TransformGroup yoyoTGR2 = new TransformGroup(rotate);

Cone cone2 = new Cone(0.6f, 0.2f);
cone2.setAppearance(yoyoAppear);

yoyoBG.addChild(yoyoTGT2);
yoyoTGT2.addChild(yoyoTGR2);
yoyoTGR2.addChild(cone2);
```

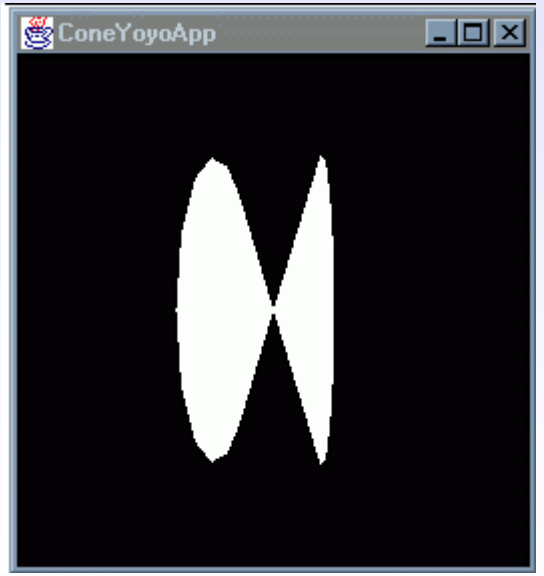
Example: a Yo-Yo (3)

```
    yoyoBG.compile();

    } // end of ConeYoyo constructor

  public BranchGroup getBG(){
    return yoyoBG;
  }

} // end of class ConeYoyo
```



Yo-Yo Scene Graph

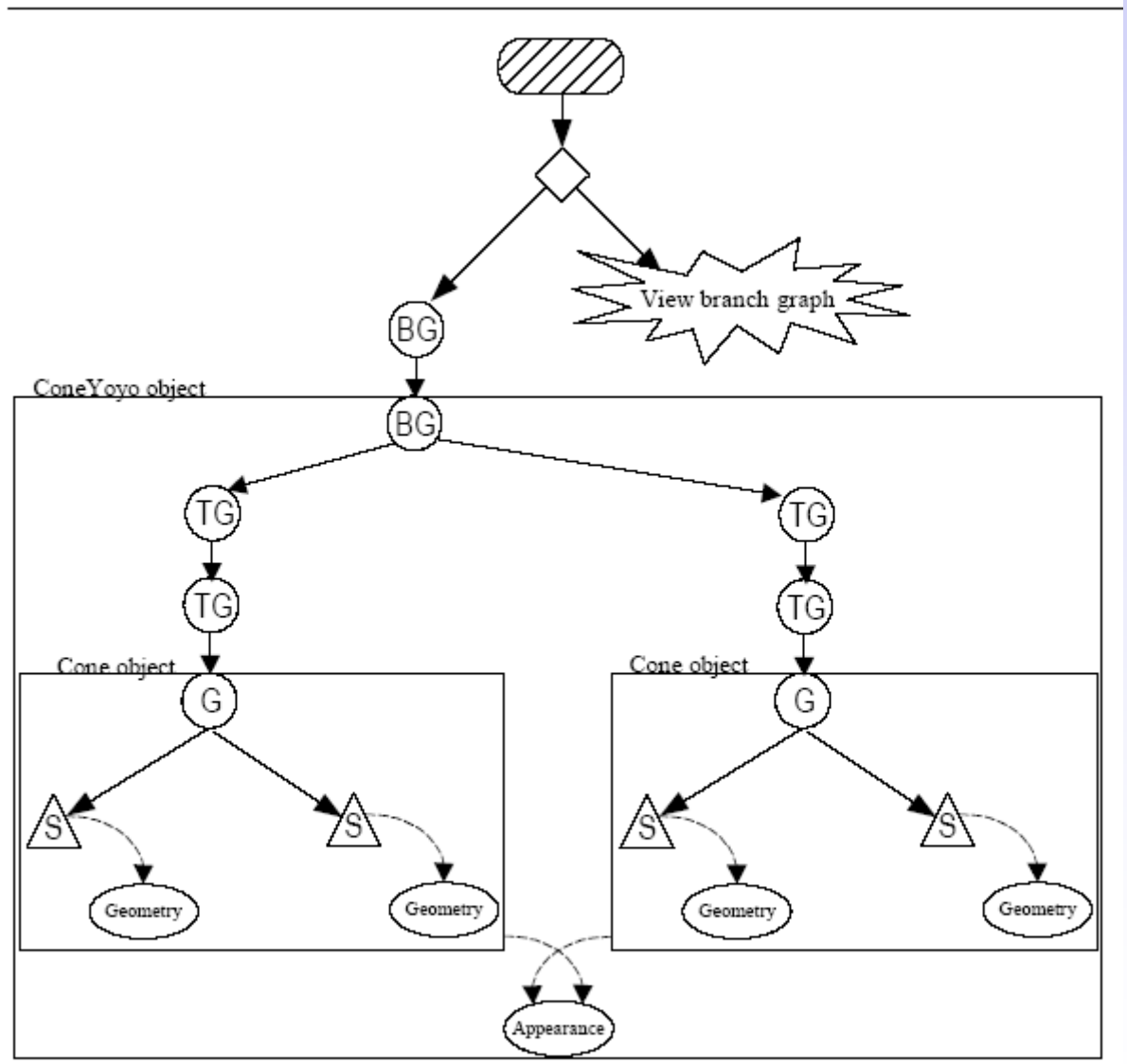
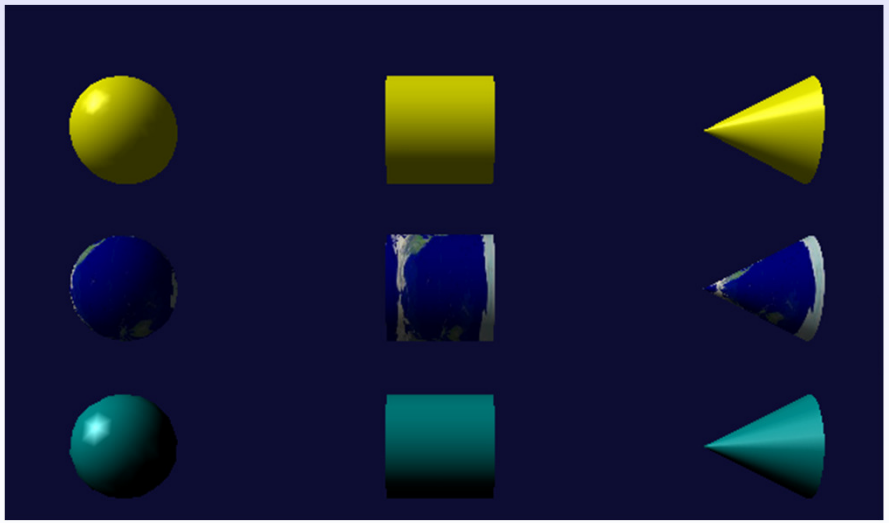


Figure 2-6 Scene Graph for ConeYoyoApp[7]

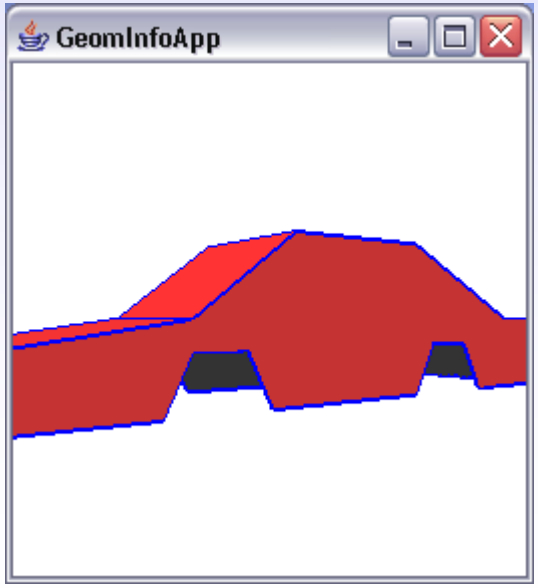
Primitive Visual Objects

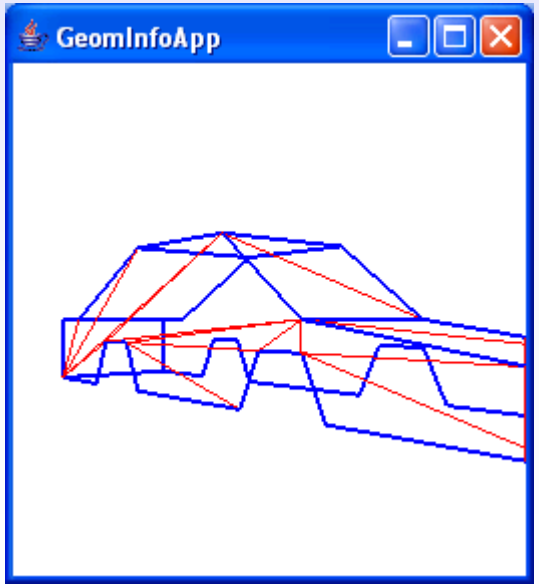
- Inbuilt basic geometric shapes
  - Box, Cone, Cylinder, Sphere
- Extend the Primitive class
- Contain more than one Shape3D



Constructing Geometry

- Possible to construct arbitrary shapes in Java 3D
  - BUT tedious and not worthwhile except for simple objects that are not boxes etc
- Specify arrays of 3D points
  - Vertices of polygons





Mathematical Classes

- A number of mathematical classes are provided to make geometry creation possible
- Specify vertex-related data
  - Point* (for 3D coordinates)
  - Color* (for colors)
  - Vector* (for surface normals)
  - TexCoord* (for texture coordinates)

Geometry Classes

- Use subclasses of GeometryArray class
- Specify sets of vertex coordinates
  - Plus appearance-related data, such a vertex colour (see later)

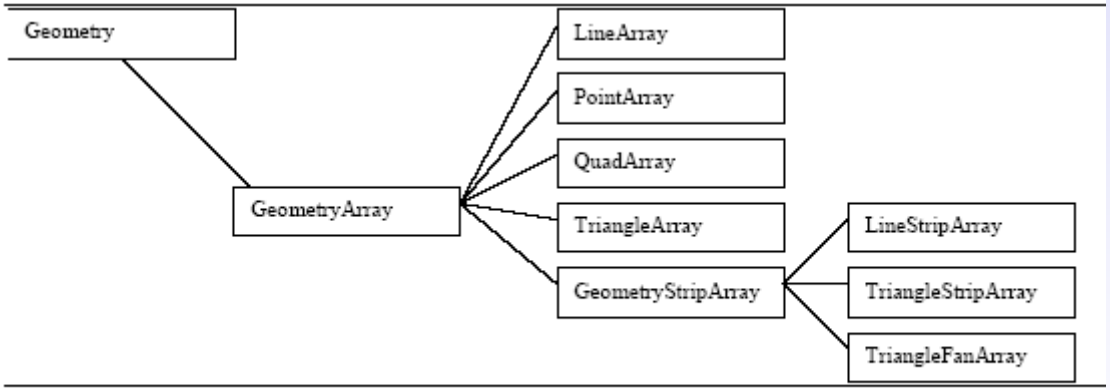


Figure 2-12 Non-Indexed GeometryArray Subclasses

Geometry Classes (2)

- Examples

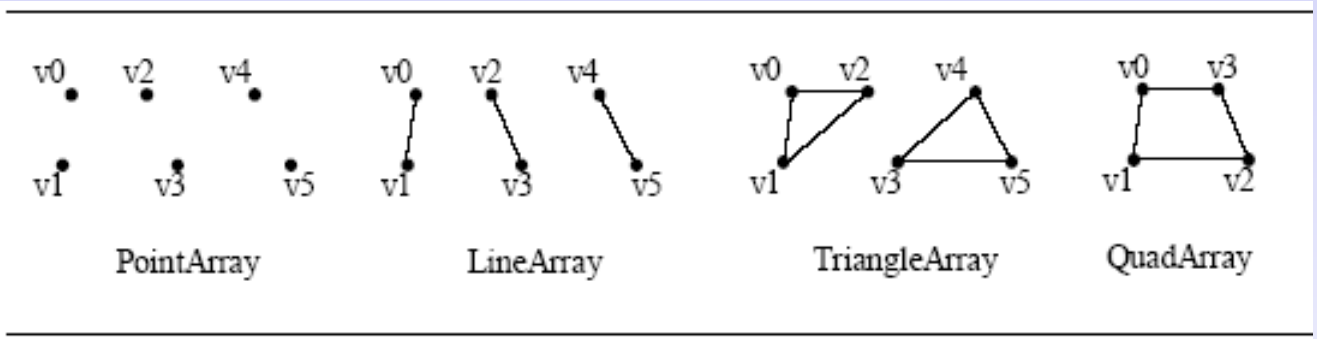


Figure 2-13 GeometryArray Subclasses

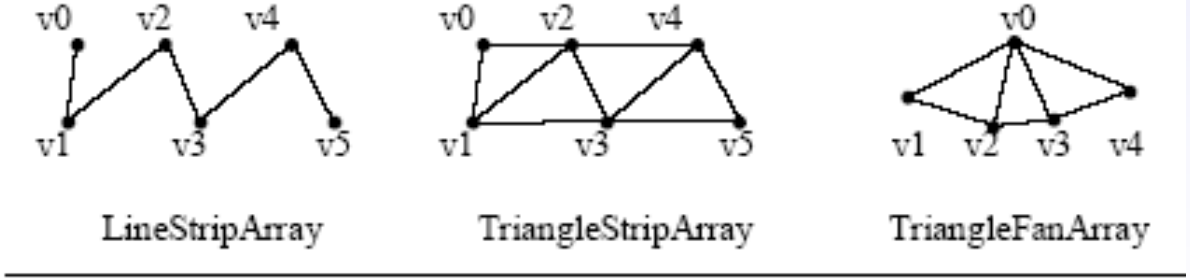
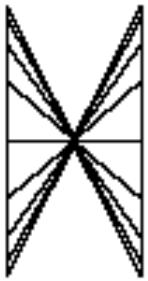


Figure 2-14 GeometryStripArray Subclasses

Example: Yo-Yo

- Yo-yo created from TriangleFanArray



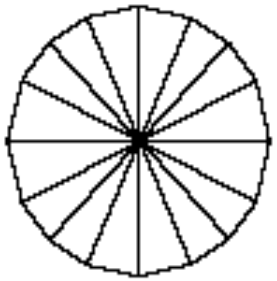


Figure 2-15 Three Views of the Yo-yo

Yo-Yo Code

```
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. int stripCounts[] = {N+1, N+1, N+1, N+1};
8. float r = 0.6f;
9. float w = 0.4f;
10. int n;
11. double a;
12. float x, y;
13.
14. // set the central points for four triangle fan strips
15. coords[0*(N+1)] = new Point3f(0.0f, 0.0f, w);
16. coords[1*(N+1)] = new Point3f(0.0f, 0.0f, 0.0f);
17. coords[2*(N+1)] = new Point3f(0.0f, 0.0f, 0.0f);
18. coords[3*(N+1)] = new Point3f(0.0f, 0.0f, -w);
19.
```

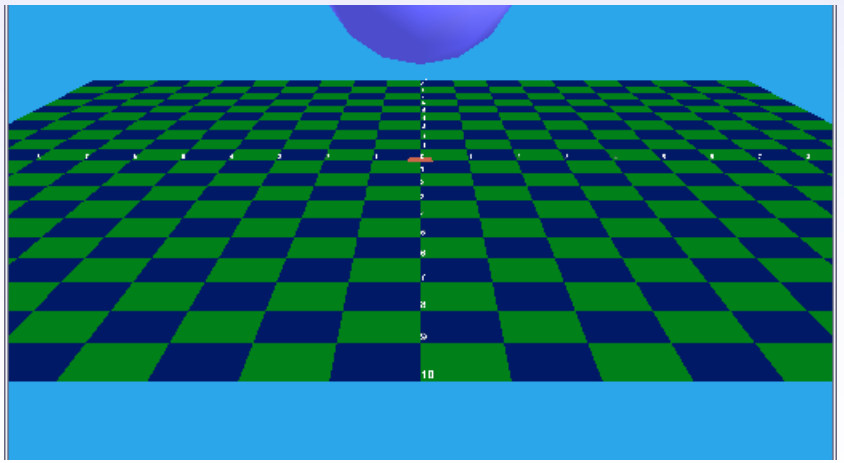
Yo-Yo Code (2)

```
20. for (a = 0,n = 0; n < N; a = 2.0*Math.PI/(N-1) * ++n){
21.   x = (float) (r * Math.cos(a));
22.   y = (float) (r * Math.sin(a));
23.
24.   coords[0*(N+1)+N-n] = new Point3f(x, y, w);
25.   coords[1*(N+1)+n+1] = new Point3f(x, y, w);
26.   coords[2*(N+1)+N-n] = new Point3f(x, y, -w);
27.   coords[3*(N+1)+n+1] = new Point3f(x, y, -w);
28. }
29.
30. tfa = new TriangleFanArray (totalN,
31.   TriangleFanArray.COORDINATES,
32.   stripCounts);
33.
34. tfa.setCoordinates(0, coords);
35.
36. return tfa;
37.} // end of method yoyoGeometry in class Yoyo
```

Checkered Floor Tiles

```
public class ColouredTiles extends Shape3D
{
  private QuadArray plane;


  public ColouredTiles(ArrayList coords, Color3f col)
  {
    plane = new QuadArray(coords.size(),
       GeometryArray.COORDINATES | GeometryArray.COLOR_3 );
    createGeometry(coords, col);
    createAppearance();
  }
```



Checkered Floor (2)

```
  private void createGeometry(ArrayList coords, Color3f col)
  {
    int numPoints = coords.size();

    Point3f[] points = new Point3f[numPoints];
    coords.toArray( points );
    plane.setCoordinates(0, points);

    Color3f cols[] = new Color3f[numPoints];
    for(int i=0; i < numPoints; i++)
      cols[i] = col;
    plane.setColors(0, cols);

    setGeometry(plane);
  }  // end of createGeometry()

::::
} // end of ColouredTiles class
```

Checkered Floor (3)

```
public class CheckerFloor
{
  private final static int FLOOR_LEN = 20;  // should be even

  // colours for floor, etc
  private Color3f blue = new Color3f(0.0f, 0.1f, 0.4f);
  private Color3f green = new Color3f(0.0f, 0.5f, 0.1f);
  private Color3f medRed = new Color3f(0.8f, 0.4f, 0.3f);
  private Color3f white = new Color3f(1.0f, 1.0f, 1.0f);

  private BranchGroup floorBG;

  public CheckerFloor()
  // create tiles, add origin marker, then the axes labels
  {
    ArrayList blueCoords = new ArrayList();
    ArrayList greenCoords = new ArrayList();
    floorBG = new BranchGroup();
```

Checkered Floor (4)

```
  boolean isBlue;
  for(int z=-FLOOR_LEN/2; z <= (FLOOR_LEN/2)-1; z++) {
    isBlue = (z%2 == 0)? true : false;    // set colour
    for(int x=-FLOOR_LEN/2; x <= (FLOOR_LEN/2)-1; x++) {
      if (isBlue)
        createCoords(x, z, blueCoords);
      else
        createCoords(x, z, greenCoords);
      isBlue = !isBlue;
    }
  }
  floorBG.addChild( new ColouredTiles(blueCoords, blue) );
  floorBG.addChild( new ColouredTiles(greenCoords, green) );

  addOriginMarker();
  labelAxes();
}  // end of CheckerFloor()
```

Checkered Floor (5)

```
  private void createCoords(int x, int z, ArrayList coords)
  // Coords for a single blue or green square,
  // its left hand corner at (x,0,z)
  {
    // points created in counter-clockwise order
    Point3f p1 = new Point3f(x, 0.0f, z+1.0f);
    Point3f p2 = new Point3f(x+1.0f, 0.0f, z+1.0f);
    Point3f p3 = new Point3f(x+1.0f, 0.0f, z);
    Point3f p4 = new Point3f(x, 0.0f, z);
    coords.add(p1); coords.add(p2);
    coords.add(p3); coords.add(p4);
  }  // end of createCoords()
// end of createGeometry()

::::
} // end of CheckerFloor class
```

The End