Java 3D – Texture Mapping

Winter 2003

Texture Mapping Models

- Geometry Model

  Model the details of every 3D shape in our graph scene, but this requires a substantial modeling effort. The more shapes we have the more things to draw

- Image Model

  Create the illusion of geometry details by taking a picture of the "real image”, and then attaching the image onto a simple 3D geometry. The benefits of this approach is that realism is increased without having to draw a large amount of geometry objects

Appearance Object

- We recall that the Appearance object is a container for several visual attributes of a 3D shape:
  - Coloring Attributes
  - Transparency Attributes
  - Rendering Control
  - Point Attributes
  - Line Attributes
  - Polygon Attributes
  - Rendering Attributes
  - Texture Control
  - Texture
  - Texture Attributes
  - Text Coordinate Generation

Describing 3D Geometry for Texture Mapping

- NodeComponent
  - Super class for Geometry and Appearance classes
  - GeometryArray class and its subclasses consists of separate arrays of coordinates, normals, RGB and RGBA colors and texture coordinates
  - Appearance objects may specify color, texture parameters, culling, and shading

- GeometryArray Methods:
  - GeometryArray(int vertexCount, int vertexFormat)

Vertex format is a mask indicating what is present in each vertex:

COORDINATES, NORMALS, COLOR_3 or COLOR_4,

TEXTURE_COORDINATE_2 or TEXTURE_COORDINATE_3

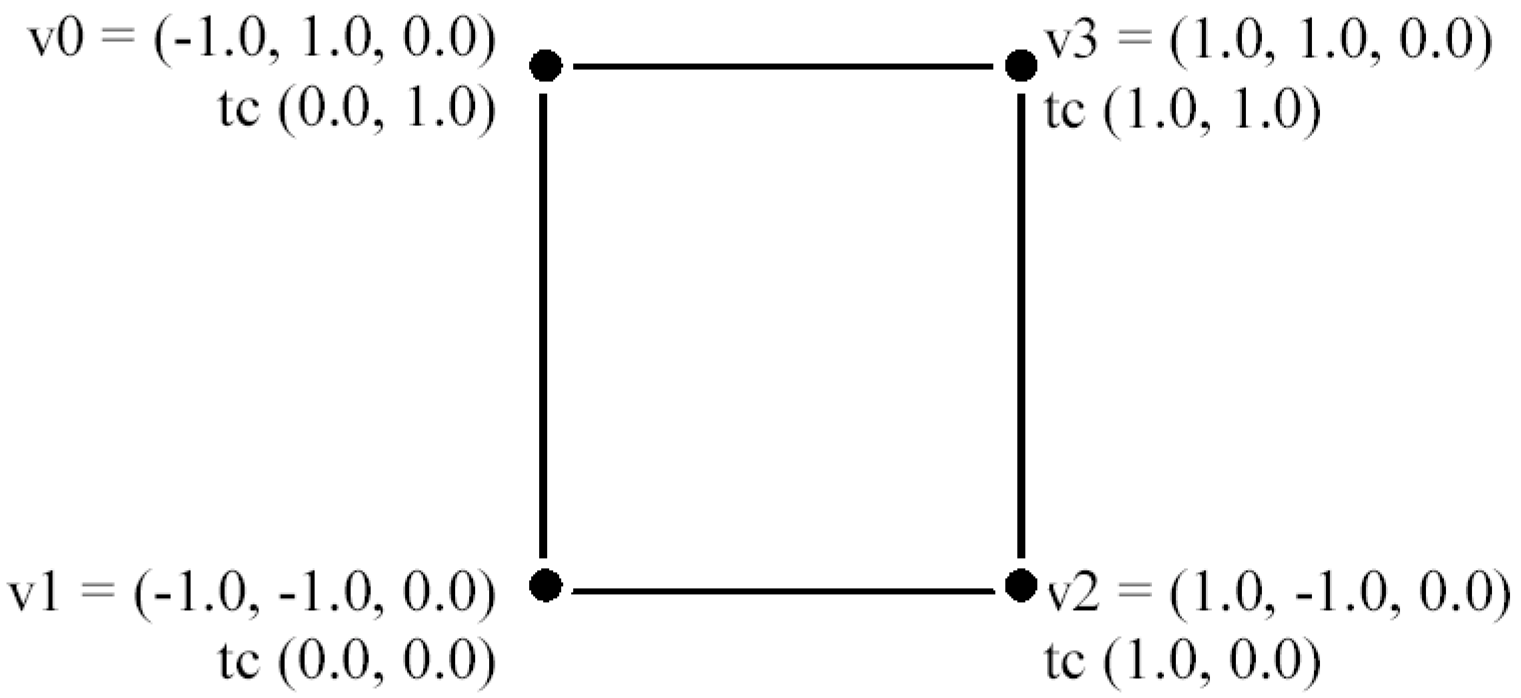
Describing 3D Geometry for Texture Mapping

- GeometryArray Methods:
  - final int getVertexCount()
  - final int getVertexFormat()
  - final void setCoordinate(…)
  - final void setCoordinates(…)
  - final void setColor(…)
  - final void setColors(…)
  - final void setNormal(…)
  - final void setNormals(…)
  - final void setTextureCoordinates(…)

Texture Appearance Attributes

- Texture appearance attributes are divided among several node components:
  - Texture: Allows the selection of a texture image and controls basic mapping attributes
  - TextureAttributes: Controls advanced mapping attributes
  - TexCoordGeneration: Automatically generates texture coordinates unless user defined coordinates are provided

Specify Geometry and Texture Coordinates



Sample Code

```
QuadArray plane = new QuadArray(4, GeometryArray.COORDINATES
                             | GeometryArray.TEXTURE_COORDINATE_2);
Point3f p = new Point3f();
p.set(-1.0f, 1.0f, 0.0f);
plane.setCoordinate(0, p);
p.set(-1.0f, -1.0f, 0.0f);
plane.setCoordinate(1, p);
p.set( 1.0f, -1.0f, 0.0f);
plane.setCoordinate(2, p);
p.set( 1.0f, 1.0f, 0.0f);
plane.setCoordinate(3, p);

TexCoord2f q = new TexCoord2f();
q.set(0.0f, 1.0f);
plane.setTextureCoordinate(0, 0, q);
q.set(0.0f, 0.0f);
plane.setTextureCoordinate(0, 1, q);
q.set(1.0f, 0.0f);
plane.setTextureCoordinate(0, 2, q);
q.set(1.0f, 1.0f);
plane.setTextureCoordinate(0, 3, q);
```

Texture Objects

- Texture is the base class for two node components that select the image to use
  - Texture2D: a standard 2D image
  - Texture3D: a 3D volume of images

- Texture2D and Texture3D Methods:
  - Texture2D(): Default constructor
  - Texture3D(): Default constructor
  - void setImage(int level, ImageComponent2D image): Select mip-map level and which image to use
  - void setEnable(boolean onOff): Set texture mapping on or off

Texture Loader

- Getting a texture map requires:
  - A file to load from disk or network using a URL
  - A TextureLoader object to load the file
  - An ImageComponent to hold the loaded image in memory, which in turn uses a standard BufferedImage object
- ImageComponent:
  - Base class for two image containers
    - ImageComponent2D: Holds a 2D image
    - ImageComponent3D: Holds a 3D volume of images
  - Used for Background or Texture objects
  - Can utilize java.awt.Image.BufferedImage object

Texture Loader

- ImageComponent2D and 3D Methods:
  - ImageComponent2D(int format, BufferedImage image): Default 2D constructor
  - ImageComponent3D(int format, BufferedImage image): Default 3D constructor
  - final int getWidth(): Get image width
  - final int getHeight(): Get image height
  - final int getDepth(): Get image depth. Used for 3D images only
  - final int getFormat(): Get internal pixel format. Image component has support for several internal pixel formats
  - final void set(Image): Set the image buffer essentially copies the buffered image into the object

Adding a Texture Map

- Adding a texture map to a 3D shape can be done in 4 steps:
  - Load an image from local storage or the network using a Texture Loader object into a Component Image object
  - Create a Texture2D object using the Component Image loaded into memory
  - Create an Appearance object and place the texture map into it
  - Assemble a shape object by attaching the geometry and the appearance object into it

Texture2D Example

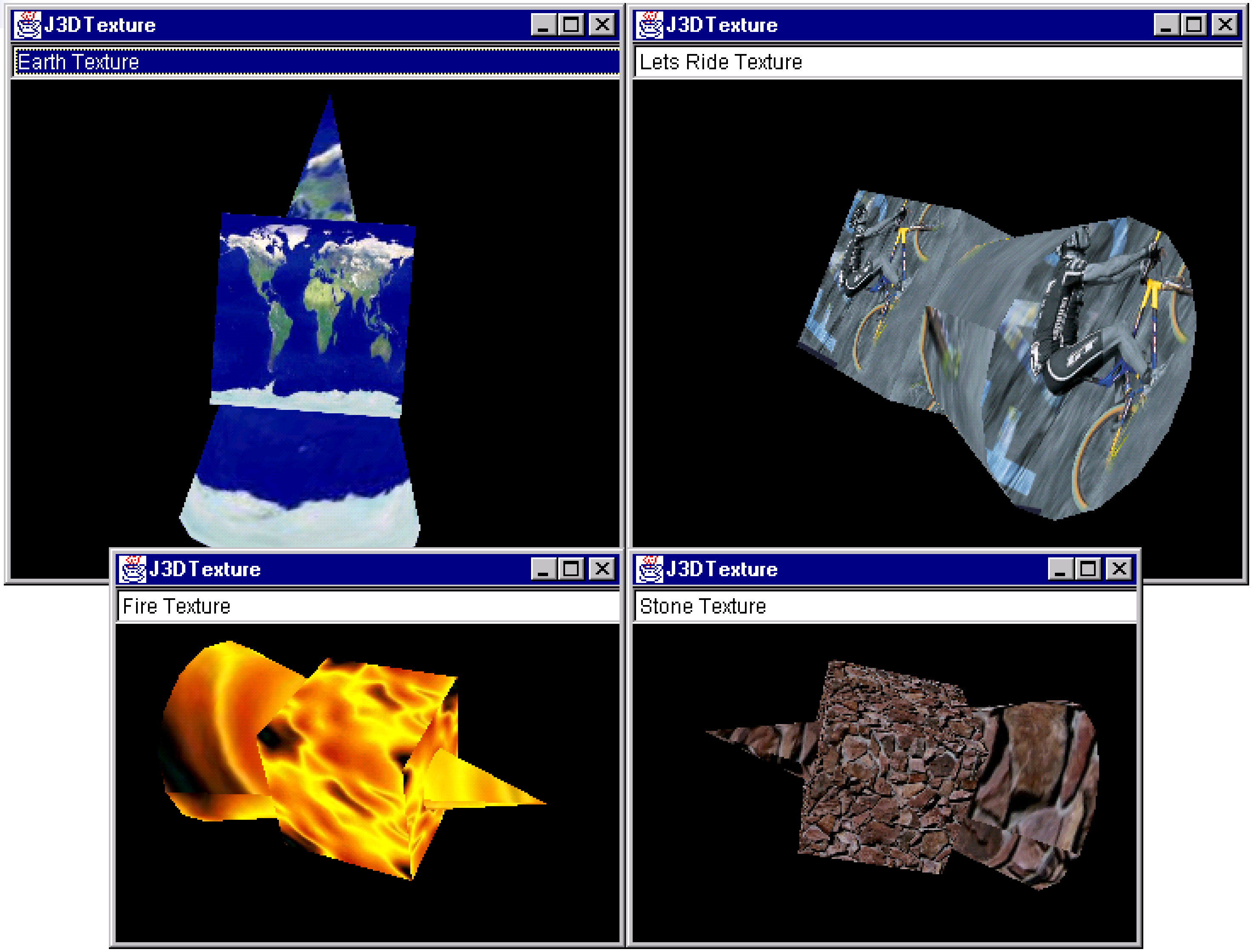
```
Void createTexture() {
    // load a texture image from disk
    TextureLoader myLoader = new TextureLoader(”Earth.jpg”);
    ImageComponent2D myImage = myLoader.getImage();

    // create a Texture2D using the image loaded
    Texture2D myTexture = new Texture2D();
    myTexture.setImage(0, myImage);

    // create an Appearance object and place the texture map into it
    Appearance myAppearance = new Appearance();
    myAppearance.setTexture(myTexure);

    // assemble the shape object by attaching the geometry and appearance object into
       it
    Shape3D myShape = new Shape3D(myGeometry, myAppearance);
}
```

Appendix: J3DTexture Example


J3DTexture
Earth Texture
J3DTexture
Lets Ride Texture
J3DTexture
Fire Texture
J3DTexture
Stone Texture