Intro to 3D Graphics

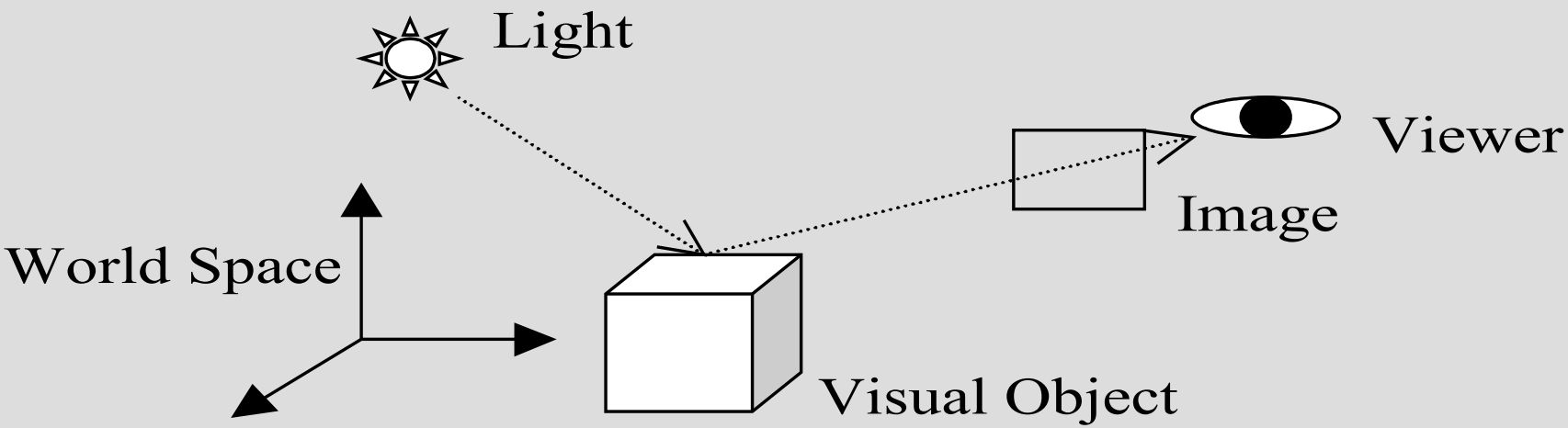
3D Rendering Process

- In 2D graphics:
  - Line between modeling and rendering often blurred
    - e.g., We've largely been doing both within the paintComponent method
  - Transformations in 2D graphics generally produce same result for object space transforms and device space transforms
  - In 2D, the scene often constructed on the fly

3D Rendering Process

- In 3D Graphics, everything is significantly more complex
  - Rendered image of 3D object different from original 3D version
  - Not feasible to go from rendered image to 3D model
    - Can do this easily for 2D
    - Computer Vision looks at this problem, but beyond scope of this course
- 3D Graphics Systems
  - Usually necessary to maintain a persistent “retained” model of virtual world
  - Modeling ans rendering engines separate from each other

3D Model and View

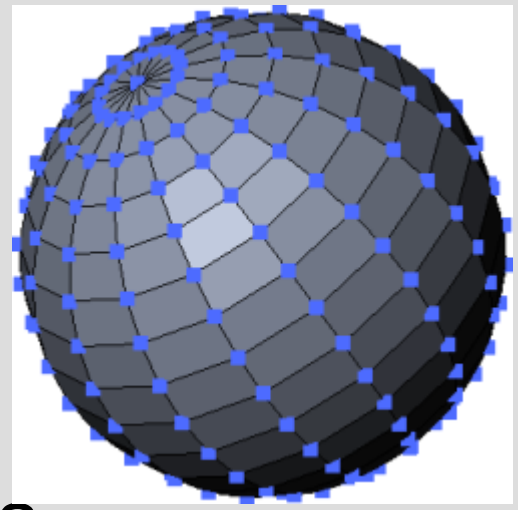


- Virtual world modeled separate from rendering engine
  - Models objects, lighting, textures, camera location
- Rendering engine projects visible portions onto 2D plane, rendering that image

Rendering Considerations

- Geometry of the graphics objects
- Location and position of the objects
- Geometric transformations applied to objects and views
- Material properties and texture of the objects
- Lights and their characteristics
- Type of projections in a view
- View position, field of view, and other properties
- Illumination and shading models
- Dynamic behaviors of various components
- Reactions to the user inputs

Geometric Descriptions of 3D Objects



- Building blocks for 3D objects:
  - Points, lines, surfaces, and solids
  - More advances: spline curves, spline surfaces
- Complex objects usually approximated by meshes of polygons
  - Similar to how in 2D we approximate curves with series of line segments
- 3D graphics systems (e.g., Java 3D)
  - High-level geometries for 3D text, geometric primitives (e.g., spheres, cones, boxes)

Transformations

- Geometric transforms used to:
  - Place objects in virtual world
  - Changes size, shape, position, etc of the objects
- 3D Affine Transformations
  - Commonly used to transform virtual world space
- Projective transforms
  - For 3D viewing
  - “Projecting” onto 2D plane

Properties of a Graphics Object

- More than just geometry effects rendering
- Graphics objects can have:
  - Colors
  - Textures
  - Material properties
- Lighting, illumination, shading
  - Light sources in model
  - Policies to specify how lighting, illumination, and shading effect colors and light intensities during rendering

3D viewing

- Projective transformations map 3D scene onto 2D plane
- Needs to deal with hidden objects or portions of objects

Dynamic vs Static

- 3D rendering not limited to static scene
- Virtual world can change over time
-
- Interaction
  - Altering scene based on user feedback
  - e.g., games, virtual reality simulators, etc
-
- Animation
  - Changes internal to virtual world
  - e.g., model includes modeling behavior of objects (e.g., computer controlled character in a game)

Java 3D Overview

- Automatic rendering
- Modeling with a scene graph
- Object oriented

A Java 3D Hello program

Java 3D API

- API for 3D graphics
- Sits on top of either OpenGL or DirectX
- The Java 3D rendering engine
  - Handles the rendering of the scene automatically
- Programming with Java 3D involves:
  - Specifying the scene
  - Modeling the objects
  - Modeling properties such as light sources, textures, materials, etc
  - Specifying projection rules, etc
- Java 3D rendering engine automatically handles the rendering of the image

Java 3D

- Java 3D is object oriented
  - This is unlike most lower level graphics APIs such as OpenGL
- Java 3D uses something called a scene graph
  - We will look at the basics of graph theory
  - Organization of all objects necessary to render a scene
  - Describes the entire virtual world
  - Defines geometries, appearances, transforms, lights, views, etc
- Java 3D rendering engine continuously traverses this graph for rendering

Java 3D packages

- javax.media.j3d
  - Main package of Java 3D
- javax.vecmath
  - Classes for vectors, matrices, and other 3D related math objects
- Other useful, though not strictly Java 3D, packages
  - com.sun.j3d.utils.universe
  - com.sun.j3d.utils.geometry

Java 3D

- Textbook examples involving Java 3D use AWT components rather than Swing
- Why?
  - Canvas3D used from Java 3D rendering is “heavyweight”
  - If you place a heavyweight component in Swings JFrame, unexpected consequences can happen
    - e.g., menus when opened from the menubar can appear underneath the object in the JFrame
- Have your classes extend Applet rather than Japplet
  - See Hello3D.java example
  - Add the Applet to a MainFrame object

Java 3D rendering

- Canvas3D
  - Canvas3D is subclass of AWT's Canvas
  - Canvas3D is where the scene is rendered
- SimpleUniverse
  - Basic framework for Java3D rendering
  - Subclass of VirtualUniverse
  - We will also be using VirtualUniverse
  - Handles all of the rendering
- BranchGroup
  - The scene graph is an object of this class
  - It gets attached to the SimpleUniverse (or Virtual Universe)

A Few of Java 3Ds Features

- Example illustrates some of what is capable in Java 3D
  - Can define 3D fonts with Font3D
  - Can have 3D text objects
  - Can derive shapes from 3D text
  - Transform3D is used to define 3D transformations
  - Light sources can be placed within the scene
  - BoundingSpheres can be used to limit the region where the light influences

Java 3D

- Installed in the labs
- Java 3D includes
  - 4 jar files (Java libraries)
    - j3dcore.jar, j3dutils.jar, j3daudio.jar, vecmath.jar
  - Native code
    - For windows, it is 3 dynamic link libraries
    - Similar type of thing for other platforms
- Java3D attempts to utilize hardware acceleration if available
- If you discover compatibility problems
  - See list on page 141 of things to try