Web Application Development

Graph Models for Web-App

09/5/2019

Outline

- Java Applet
- Java-Chart
- VRML
- X3D
- Web360 & Virtual Reality
- Web-GL

Java Applet

- Applet restrictions
- Basic applet and HTML template
- The applet life-cycle33
- Customizing applets through HTML parameters
- Methods available for graphical operations
- Loading and drawing images
- Controlling image loading
- Java Plug-In and HTML converter
- Control Java from JavaScript

Applet Restrictions

- Do not read from the local (client) disk
- Do not write to the local (client) disk
- Do not open network connections other than to the server from which the applet was loaded
- Do not link to client-side C code or call programs installed on the browser machine
- Cannot discover private information about the user

Applet Template

```
import java.applet.Applet;
import java.awt.*;
public class AppletTemplate extends Applet {
  // Variable declarations.
  public void init() {
    // Variable initializations, image loading, etc.
  }
  public void paint(Graphics g) {
    // Drawing operations.
  }
}
```

- Browsers cache applets: in Netscape, use Shift-RELOAD to force loading of new applet. In IE, use Control-RELOAD
- Can use appletviewer for initial testing

Applet HTML Template

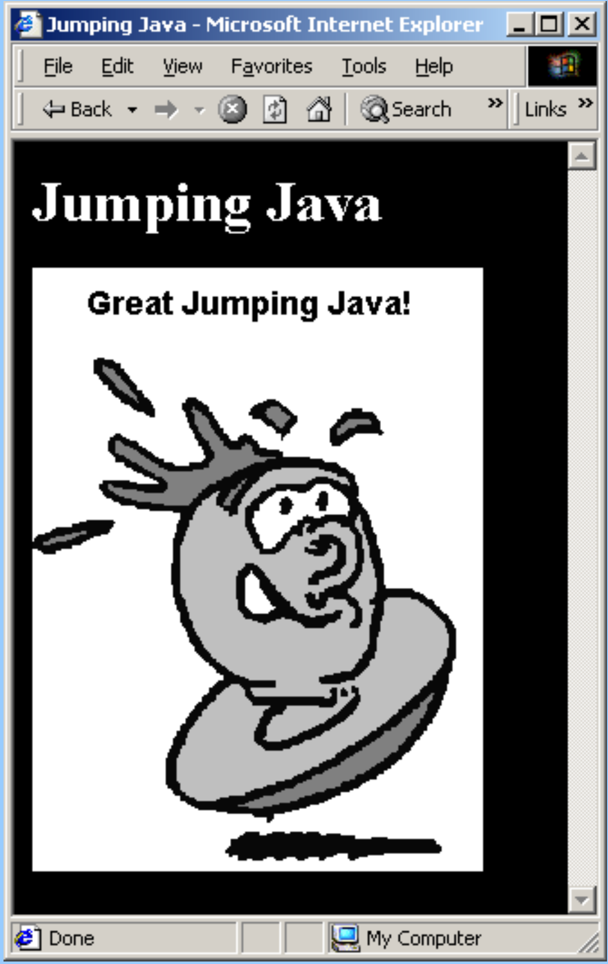
```
<html><head>
  <title>A Template for Loading Applets</title>
</head>
<body>
<h1>A Template for Loading Applets</h1>
<p>
<applet code="AppletTemplate.class" width=120
   height=60>
  <b>Error! You must use a Java-enabled
   browser.</b>
</applet>
</body>
</html>
```

Applet Example

```
import java.applet.Applet;
import java.awt.*;

/** An applet that draws an image. */

public class JavaJump extends Applet {
  private Image jumpingJava; // Instance var declarations here

  public void init() {        // Initializations here
    setBackground(Color.white);
    setFont(new Font("SansSerif", Font.BOLD, 18));
    jumpingJava = getImage(getDocumentBase(),
                           "images/Jumping-Java.gif");
    add(new Label("Great Jumping Java!"));
    System.out.println("Yow! I'm jiving with Java.");
  }

  public void paint(Graphics g) {  // Drawing here
    g.drawImage(jumpingJava, 0, 50, this);
  }
}
```

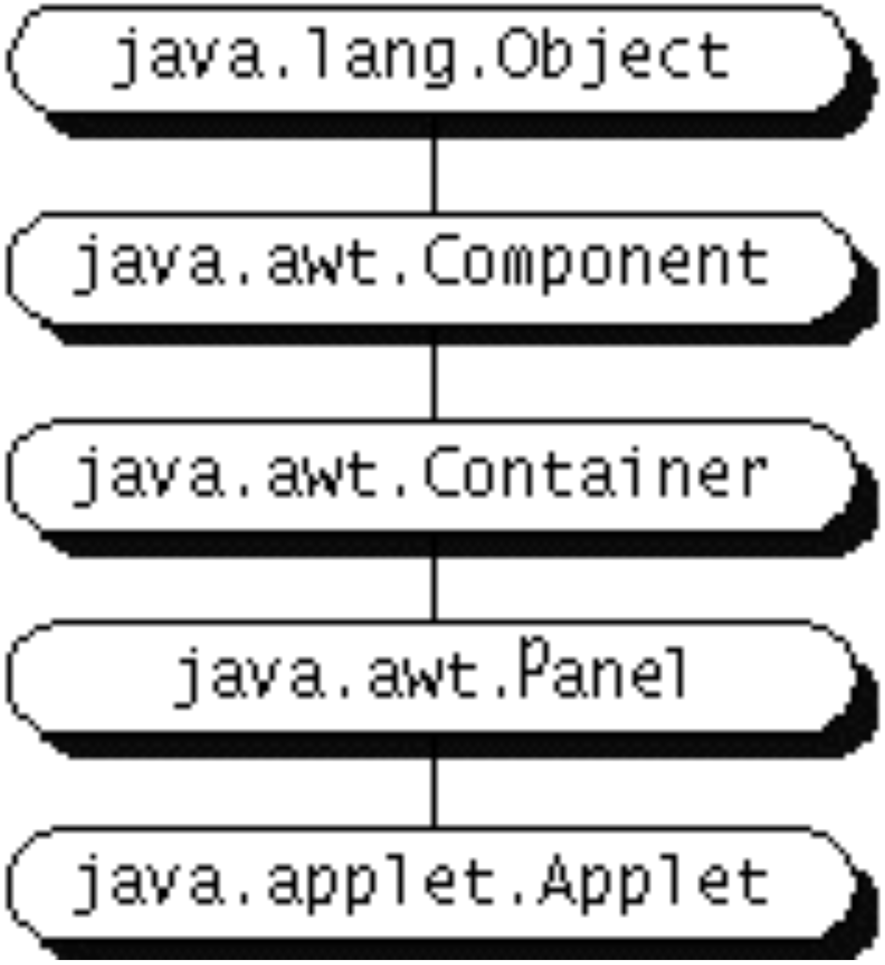
Applet Example, Result

```
<HTML>
<HEAD>
  <TITLE>Jumping Java</TITLE>
</HEAD>
<BODY BGCOLOR="BLACK" TEXT="WHITE">
<H1>Jumping Java</H1>
<P>
<APPLET CODE="JavaJump.class"
        WIDTH=250
        HEIGHT=335>
  <B>Sorry, this example requires
  Java.</B>
</APPLET>
</BODY>
</HTML>
```


Jumping Java - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Links
Jumping Java
Great Jumping Java!
Done
My Computer

Applet Inheritance Hierarchy

- **Applet class derives from the Abstract Window Toolkit (AWT) hierarchy.**



Applet Life Cycle

1. **public void init() - Called when applet is first initialized**
2. **public void start()**
   - Called immediately after init and again revisited after browser left page containing applet
   - Used to start animation threads
3. **public void paint(Graphics g)**
   - Called by the browser after init and start, and again whenever the browser redraws the screen, (typically when part of the screen has been obscured and then reexposed)
   - This method is where user-level drawing is placed
   - Inherited from java.awt.Container
4. **public void stop()**
   - Called when the browser leaves the page
   - Used to stop animation threads
5. **public void destroy()**
   - Called when applet is killed (rarely used)

AWT: UI Components

- **AWT supplies the following UI components:**
  - Buttons (java.awt.Button)
  - Checkboxes (java.awt.Checkbox)
  - Single-line text fields (java.awt.TextField)
  - Menus (java.awt.MenuItem)
  - Containers (java.awt.Panel)
  - Lists (java.awt.List)

Useful Applet Methods

- getCodeBase, getDocumentBase
  - The URL of the:

    Applet file - `getCodeBase`

    HTML file - `getDocumentBase`
- getParameter
  - Retrieves the value from the associated HTML `PARAM` element
- getSize
  - Returns the `Dimension` (width, height) of the applet
- getGraphics
  - Retrieves the current `Graphics` object for the applet
  - The `Graphics` object does not persist across `paint` invocations

Useful Applet Methods, (Continued)

- showDocument **(AppletContext method)**

```
getAppletContext().showDocument(...)
```

  - Asks the browser to retrieve and display a Web page
  - Can direct page to a named `FRAME` cell
- showStatus
  - Displays a string in the status line at the bottom of the browser
- getCursor, setCursor
  - Defines the `Cursor` for the mouse, for example, `CROSSHAIR_CURSOR`, `HAND_CURSOR`, `WAIT_CURSOR`

Useful Applet Methods, (Continued)

- getAudioClip
- play
  - Retrieves an audio file from a remote location and plays it
  - JDK 1.1 supports .au only. Java 2 also supports MIDI, .aiff and .wav
- getBackground, setBackground
  - Gets/sets the background color of the applet
  - `SystemColor` class provides access to desktop colors
- getForeground, setForeground
  - Gets/sets foreground color of applet (default color of drawing operations)

HTML APPLET Element

```
<applet code="..." width=xxx height=xxx ...>
...
</applet>
```

- **Required Attributes**
  - CODE
    - Designates the filename of the Java class file to load
    - Filename interpreted with respect to directory of current HTML page (default) unless `CODEBASE` is supplied
  - WIDTH and HEIGHT
    - Specifies area the applet will occupy
    - Values can be given in pixels or as a percentage of the browser window width

HTML APPLET Element, continued

- **Other Attributes**
  - ALIGN, HSPACE, and VSPACE
    - Controls position and border spacing just like `IMG` element (in pixels)
  - ARCHIVE
    - Designates JAR file (zip file with .jar extension) containing all classes and images used by applet
    - Save considerable time download multiple class files
  - NAME
    - Names the applet for interapplet and JavaScript communication
  - MAYSCRIPT (nonstandard)
    - Permits JavaScript to control the applet

Setting Applet Parameters

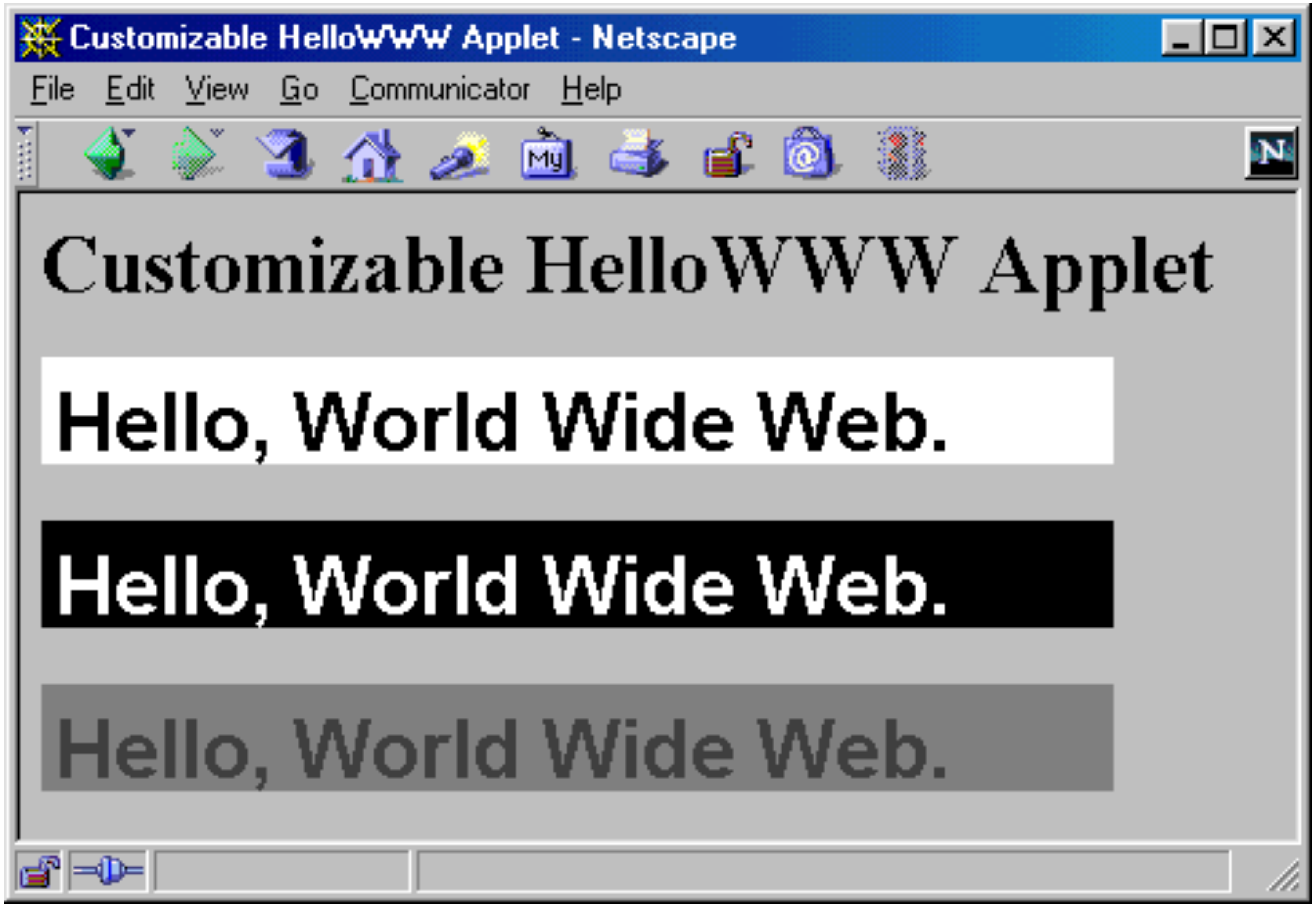
```
<h1>Customizable HelloWWW Applet</h1>

<applet code="HelloWWW2.class" width=400 height=40>
  <param name="BACKGROUND" value="LIGHT">
  <b>Error! You must use a Java-enabled browser.</b>
</applet>

<applet code="HelloWWW2.class" width=400 height=40>
  <param name="BACKGROUND" value="DARK">
  <b>Error! You must use a Java-enabled browser.</b>
</applet>

<applet code="HelloWWW2.class" width=400 height=40>
  <b>Error! You must use a Java-enabled browser.</b>
</applet>
```

Reading Applet Parameters

- Use getParameter(name) to retrieve the value of the PARAM element and the name argument is case sensitive

```
public void init() {
   Color background = Color.gray;
   Color foreground = Color.darkGray;
   String backgroundType = getParameter("BACKGROUND");
   if (backgroundType != null) {
     if (backgroundType.equalsIgnoreCase("LIGHT")) {
       background = Color.white;
       foreground = Color.black;
     } else if (backgroundType.equalsIgnoreCase("DARK")) {
       background = Color.black;
       foreground = Color.white;
     }
   } ...
  }
}
```

Reading Applet Parameters: Result


Customizable HelloWWW Applet - Netscape
File Edit View Go Communicator Help

Customizable HelloWWW Applet

Hello, World Wide Web.

Hello, World Wide Web.

Hello, World Wide Web.

Useful Graphics Methods

- **drawString(string, left, bottom)**
  - Draws a string in the current font and color with the *bottom left* corner of the string at the specified location
  - One of the few methods where the y coordinate refers to the bottom of shape, not the top. But y values are still with respect to the *top left* corner of the applet window
- **drawRect(left, top, width, height)**
  - Draws the outline of a rectangle (1-pixel border) in the current color
- **fillRect(left, top, width, height)**
  - Draws a solid rectangle in the current color
- **drawLine(x1, y1, x2, y2)**
  - Draws a 1-pixel-thick line from (x1, y1) to (x2, y2)

Useful Graphics Methods, continued

- drawOval, fillOval
  - Draws an outlined and solid oval, where the arguments describe a rectangle that bounds the oval
- drawPolygon, fillPolygon
  - Draws an outlined and solid polygon whose points are defined by arrays or a `Polygon` (a class that stores a series of points)
  - By default, polygon is closed; to make an open polygon use the drawPolyline method
- drawImage
  - Draws an image
  - Images can be in JPEG or GIF (including GIF89A) format

Graphics Color

- setColor, getColor
  - Specifies the foreground color prior to drawing operation
  - By default, the graphics object receives the foreground color of the window
  - AWT has 16 predefined colors (`Color.red`, `Color.blue`, etc.) or create your own color,
    `new Color(r, g, b)`
  - Changing the color of the `Graphics` object affects only the drawing that explicitly uses that `Graphics` object
    - To make permanent changes, call the *applet's* `setForeground` method.

Graphics Font

- **setFont, getFont**
  - Specifies the font to be used for drawing text
  - Determine the size of a character through `FontMetrics` (in Java 2 use `LineMetrics`)
  - Setting the font for the `Graphics` object does not persist to subsequent invocations of `paint`
  - Set the font of the window (I.e., call the *applet's* `setFont` method) for permanent changes to the `Graphics` object
  - In JDK 1.1, only 5 fonts are available: `Serif` (aka `TimesRoman`), `SansSerif` (aka `Helvetica`), `Monospaced` (aka `Courier`), `Dialog`, and `DialogInput`

Graphics Behavior

- **Browser calls `repaint` method to request redrawing of applet**
  - Called when applet first drawn or applet is hidden by another window and then re-exposed

$$\begin{tikzcd}
\texttt{repaint()} \arrow[d, dashed, "\text{``sets } \textit{flag}\text{''}"] \\
\texttt{update(Graphics g)} \arrow[d, "\text{Clears screen, Calls } \texttt{paint}"] \\
\texttt{paint(Graphics g)}
\end{tikzcd}$$

Drawing Images

- **Register the Image (from `init`)**

```
Image image = getImage(getCodeBase(), "file");
Image image = getImage (url);
```

  - Loading is done in a separate thread
  - If URL is absolute, then `try/catch` block is required

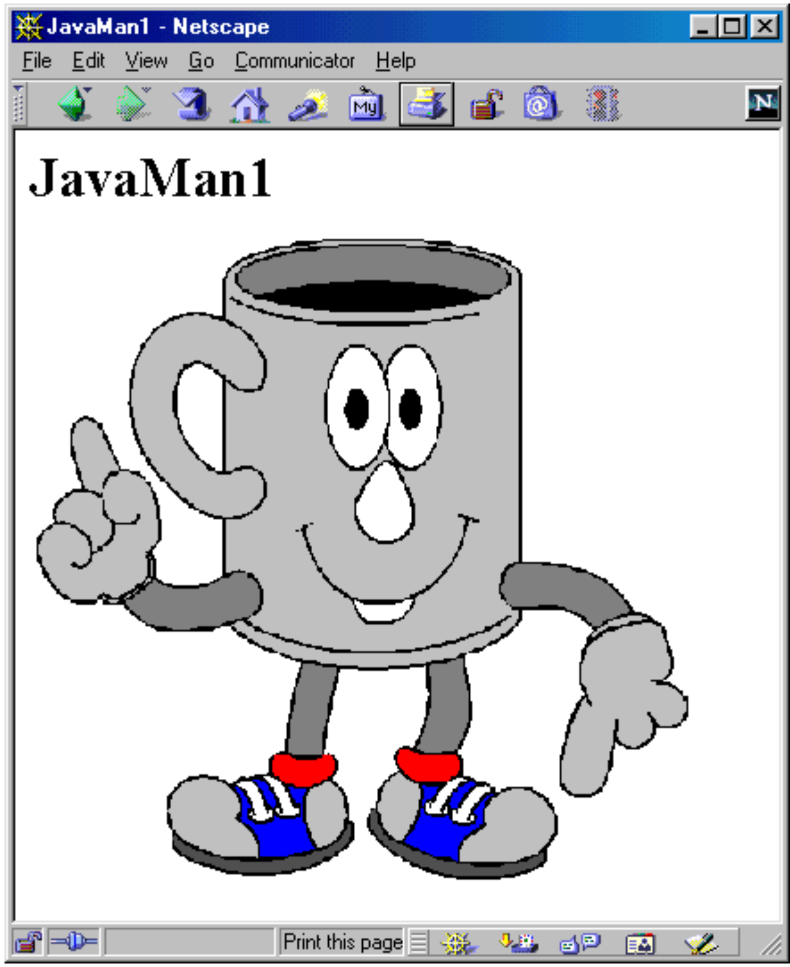
- **Draw the image (from `paint`)**

```
g.drawImage(image, x, y, window);
g.drawImage(image, x, y, w, h, window);
```

  - May draw partial image or nothing at all
  - Use the applet (`this`) for the `window` argument

Loading Applet Image from Relative URL

```
import java.applet.Applet;
import java.awt.*;
/** An applet that loads an image
        from a relative URL. */
public class JavaMan1 extends Applet {
  private Image javaMan;
  public void init() {
    javaMan = getImage(getCodeBase(),
                       "images/Java-Man.gif");
  }
  public void paint(Graphics g) {
    g.drawImage(javaMan, 0, 0, this);
  }
}
```



Loading Applet Image from Absolute URL

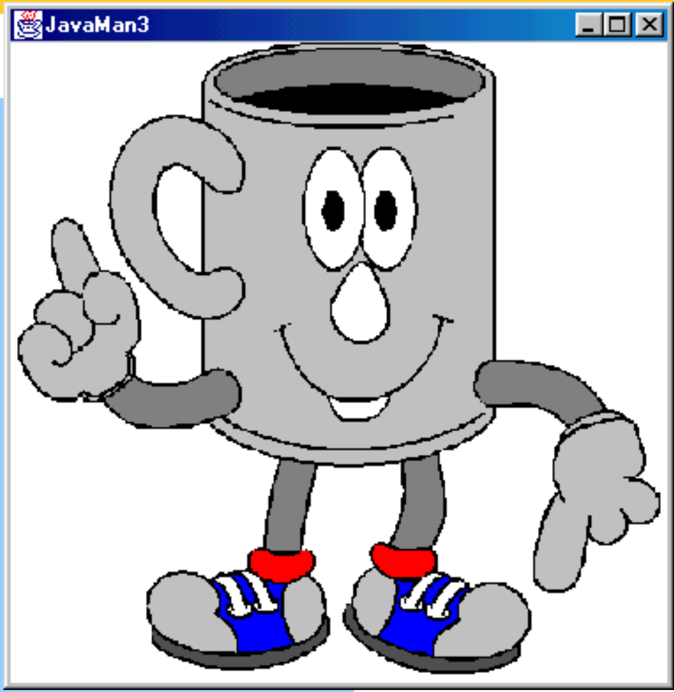
```
import java.applet.Applet;
import java.awt.*;
import java.net.*;
...
  private Image javaMan;
  public void init() {
    try {
      URL imageFile =
        new URL("http://www.corewebprogramming.com" +
                "/images/Java-Man.gif");
      javaMan = getImage(imageFile);
    } catch(MalformedURLException mue) {
      showStatus("Bogus image URL.");
      System.out.println("Bogus URL");
    }
  }
```

Loading Images in Applications

```
import java.awt.*;
import javax.swing.*;

class JavaMan3 extends JPanel {
  private Image javaMan;

  public JavaMan3() {
    String imageFile = System.getProperty("user.dir") +
                       "/images/Java-Man.gif";
    javaMan = getToolkit().getImage(imageFile);
    setBackground(Color.white);
  }

  public void paintComponent(Graphics g) {
    super.paintComponent(g);
    g.drawImage(javaMan, 0, 0, this);
  }
  ...
```

Loading Images in Applications (Continued)

```
  ...
  public void paintComponent(Graphics g) {
    super.paintComponent(g);
    g.drawImage(javaMan, 0, 0, this);
  }
  public static void main(String[] args) {
    JPanel panel = new JavaMan3();
    WindowUtilities.setNativeLookAndFeel();
    WindowUtilities.openInJFrame(panel, 380, 390);
  }
}
```



- See Swing chapter for WindowUtilities

Application: Accessing Java from JavaScript

- **Calling Java Methods Directly**
  - JavaScript can access Java variables and methods simply by using the fully qualified name. For instance:

```
java.lang.System.out.println("Hello Console");
```

  - Limitations:
    - Can't perform operations forbidden to applets
    - No try/catch, so can't call methods that throw exceptions
    - Cannot write methods or create subclasses

Controlling Applets from JavaScript, Example

- MoldSimulation.html, cont.

```
<BODY BGCOLOR="#C0C0C0">
<H1>Mold Propagation Simulation</H1>
<APPLET CODE="RandomCircles.class" WIDTH=100 HEIGHT=75></APPLET>
<P>
<APPLET CODE="RandomCircles.class" WIDTH=300 HEIGHT=75>
</APPLET>
<P>
<APPLET CODE="RandomCircles.class" WIDTH=500 HEIGHT=75>
</APPLET>
<FORM>
<INPUT TYPE="BUTTON" VALUE="Start Simulations“
   onClick="startCircles()">
<INPUT TYPE="BUTTON" VALUE="Stop Simulations" onClick="stopCircles()">
</FORM>
</BODY>
</HTML>
```

Controlling Applets from JavaScript, Example

- **MoldSimulation.html**

```
<HTML><HEAD>
  <TITLE>Mold Propagation Simulation</TITLE>
<SCRIPT TYPE="text/javascript"><!--
function startCircles() {
  for(var i=0; i<document.applets.length; i++) {
    document.applets[i].startCircles();    }
}
function stopCircles() {
  for(var i=0; i<document.applets.length; i++) {
    document.applets[i].stopCircles();    }
}
// --></SCRIPT>
</HEAD>
```

Controlling Applets from JavaScript, Example

- RandomCircles.java

```
public class RandomCircles extends Applet implements Runnable {
  private boolean drawCircles = false;
  public void startCircles() {
    Thread t = new Thread(this);
    t.start();
  }
  public void run() {
    Color[] colors = { Color.lightGray, Color.gray, Color.darkGray,
   Color.black };
    int colorIndex = 0;
    int x, y;
    int width = getSize().width;
    int height = getSize().height;
    Graphics g = getGraphics();
    drawCircles = true;
```

Controlling Applets from JavaScript, Example

- RandomCircles.java

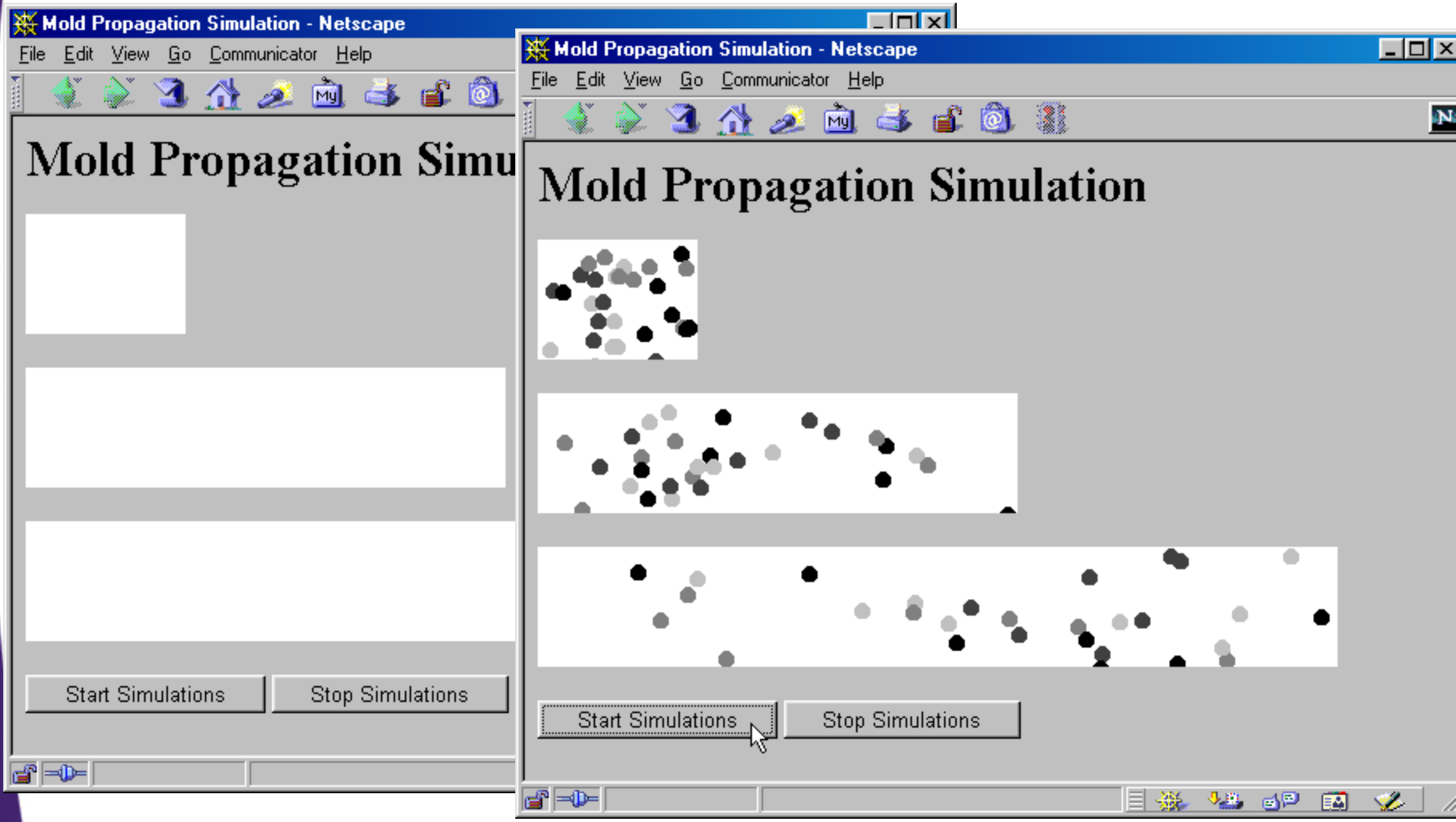
```
public class RandomCircles extends Applet implements Runnable {
  private boolean drawCircles = false;
  public void startCircles() {
    Thread t = new Thread(this);
    t.start();
  }
  public void run() {
    Color[] colors = { Color.lightGray, Color.gray, Color.darkGray,
   Color.black };
    int colorIndex = 0;
    int x, y;
    int width = getSize().width;
    int height = getSize().height;
    Graphics g = getGraphics();
    drawCircles = true;
    ...
```

Controlling Applets from JavaScript, Example

- RandomCircles.java, cont.

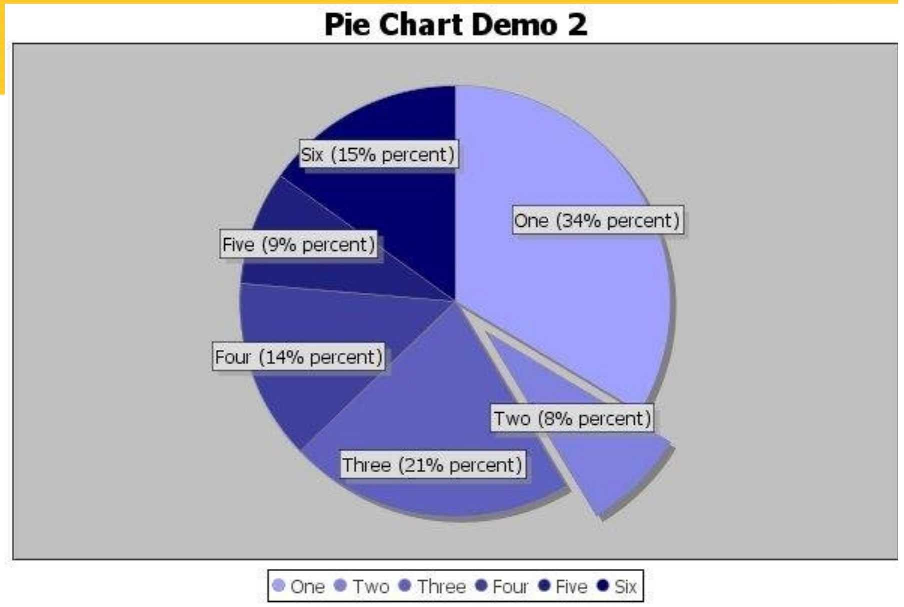
```
    while(drawCircles) {
      x = (int)Math.round(width * Math.random());
      y = (int)Math.round(height * Math.random());
      g.setColor(colors[colorIndex]);
      colorIndex = (colorIndex + 1) % colors.length;
      g.fillOval(x, y, 10, 10);
      pause(0.1);
    }
  }
  public void stopCircles() {   drawCircles = false;   }
  private void pause(double seconds) {
    try {   Thread.sleep((int)(Math.round(seconds * 1000.0)));
    } catch(InterruptedException ie) {}
  }
}
```

Controlling Applets from JavaScript, Results

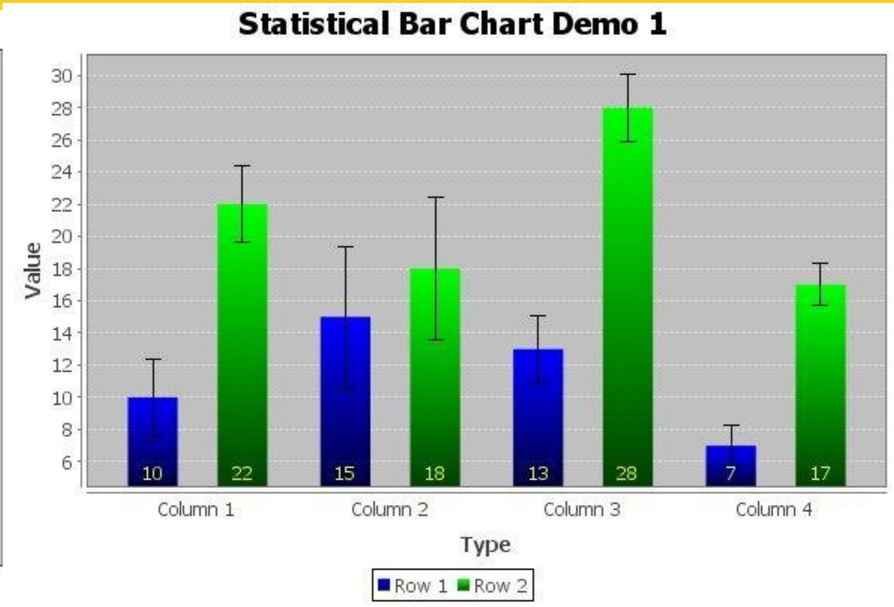

Mold Propagation Simulation - Netscape
File Edit View Go Communicator Help
Mold Propagation Simu
Start Simulations
Stop Simulations
Mold Propagation Simulation - Netscape
File Edit View Go Communicator Help
Mold Propagation Simulation
Start Simulations
Stop Simulations

JFreeChart Overview

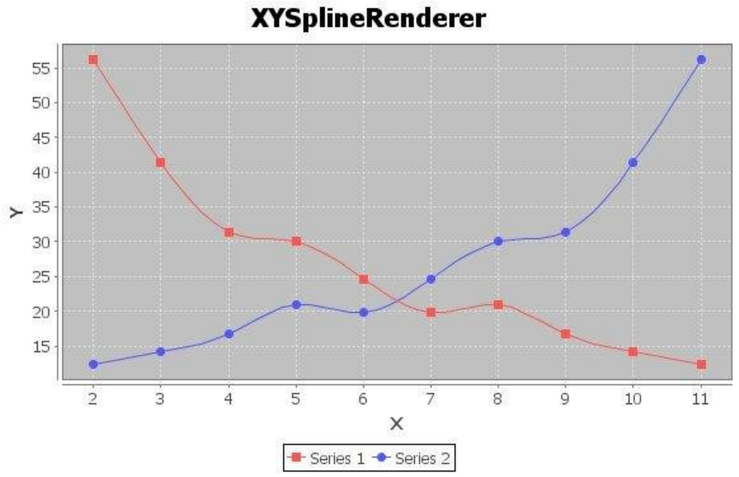
- Library for creating graphs and other charts in Java applications
- 4.9/5 Stars on SourceForge download page
- Easy to work with
- Highly extensible



Pie Chart Demo 2
Six (15% percent)
One (34% percent)
Five (9% percent)
Four (14% percent)
Two (8% percent)
Three (21% percent)
● One ● Two ● Three ● Four ● Five ● Six



Statistical Bar Chart Demo 1
Value
Type
Column 1
Column 2
Column 3
Column 4
10
22
15
18
13
28
7
17
■ Row 1 ■ Row 2



XYSplineRenderer
Y
X
■ Series 1 ● Series 2

VRML: History and Its Development Process

Learning Objectives

- Understand the history of VRML
- Understand the purpose and functions of VRML
- Learn how to view a VRML file in a file and have a general conception of how the file is defined
- Understand where VRML may go in the future.

History

1994	Labyrinth	Prototype 3D interface for the Web developed by Mark Pesce and Tony Parisi
1994	VRML 1	Developed based on Open Inventor format - described static 3D scenes
1996	VRML 2	Silicon Graphics' Moving Worlds proposal for a VRML revision is adopted
1997	VRML97	Recognized as an international standard by ISO and IEC (ISO/IEC 14772)
2000	VRML200x X3D	Proposed revision to VRML97 (ISO/IEC 14772:200x) Under development .... see the Web3D Consortium for details

Purpose

- 3D interactive objects and worlds
- Web based
- Universal interchange format for 3D graphics and multimedia

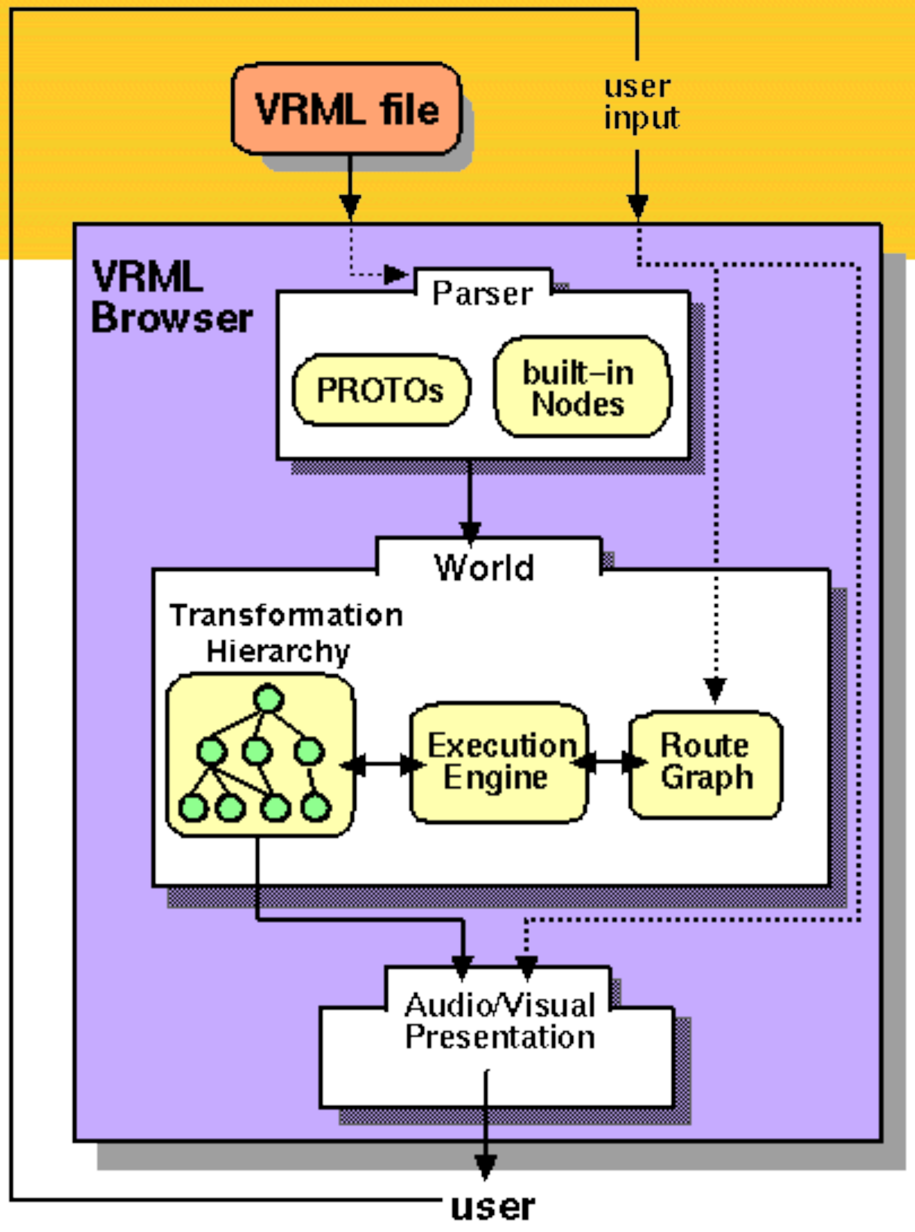
Design Criteria

- Authorability
- Composability
- Extensibility
- Performance
- Scalability

What it Does

- Implicitly establishes a world coordinate space for all objects defined in the file, as well as all objects included by the file;
- Explicitly defines and composes a set of 3D and multimedia objects;
- Can specify hyperlinks to other files and applications
- Can define object behaviors.

General Operational Model



Objects

- Box
- Cone
- Cylinder
- ElevationGrid
- Extrusion
- IndexedFaceSet
- IndexedLineSet
- PointSet
- Sphere
- Text

Learning Objectives

- Understand the formal grammar definition for VRML found here: http://www.vrml.org/technicalinfo/specifications/vrml97/part1/grammar.html
- Understand how grammar rules work
- Understand how a VRML file is parsed and understood by a software program.

Structure of a VRML File

- Header
- Scene graph
- Prototypes
- Event routing

VRML Header

- #VRML V2.0 <encoding type> [optional comment] <line terminator>

- Encoding type is usually “utf8”, but other encoding schemes are possible.

VRML Scene Graph

- Describes the scene graphs
- Defines the connections among events
- Identifies positions in the scene graph for reference by events

Prototypes

- PROTO Cube [ ] { Box { } }
  - Declares a new type, Cube, that has one shape node inside, Box.
  - You can also declare user-defined variables within the prototype.

Example

```
#VRML V2.0 utf8
Transform {
  children [
    NavigationInfo { headlight FALSE } # We'll add our own light

    DirectionalLight {       # First child
        direction 0 0 -1     # Light illuminating the scene
    }

    Transform {              # Second child - a red sphere
      translation 3 0 1
      children [
        Shape {
          geometry Sphere { radius 2.3 }
          appearance Appearance {
            material Material { diffuseColor 1 0 0 }   # Red
          }
        }
      ]
    }
Transform {              # Third child - a blue box
      translation -2.4 .2 1
      rotation     0 1 1  .9
      children [
        Shape {
          geometry Box {}
          appearance Appearance {
            material Material { diffuseColor 0 0 1 }  # Blue
         }
        }
      ]
    }

  ] # end of children for world
}
```

Some Points of Grammar

- Files begin with
  - #VRML V2.0 utf8 [optional comment] <line terminator>
- # begin comments
- control characters, space double or single quotes, sharp, comma, period, brackets, backslash or braces are not allowed in names
- First character can not be a digit, plus or minus

X3D