

# CSC 551: Web Programming

Fall 2001

## Java Applets

- applets & HTML
  - default methods (init, paint, ...)
  - APPLET & OBJECT tags, applet parameters & dimensions
- graphical applets
  - Graphics object: drawString, drawLine, drawOval, drawRect, ...
  - double buffering
- GUI applets
  - GUI elements, layout, event handling

## Applets

recall: an applet is a special form of Java program

- compiled into Java byte code, then downloaded as part of a Web page
- executed by the JVM embedded within the Web browser

important point: Java applets & applications look different!

- can define dual-purpose programs, but tricky

as with JavaScript, security is central

- when a Java applet is downloaded, the bytecode verifier of the JVM verifies to see if it contains bytecodes that open, read, write to local disk
- a Java applet can open a new window but they have Java logo to prevent them from being disguised as system window (for stealing password)
- a Java applet is not allowed to connect back to other servers except the host
  
- this secure execution environment is called *sand box model*

## First Java applet

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

/**
 * This class displays "Hello world!" on the applet window.
 */
public class HelloWorld extends Applet
{
    public void paint(Graphics g)
    {
        g.drawString("Hello world!", 10, 10);
    }
}
```

all applets inherit from the Applet class (in java.applet)

default methods include:

- `init()`: called when page is loaded to create/initialize variables  
*by default, does nothing*
- `paint(Graphics g)`: called to draw (after init) or redraw (after being obscured)  
*here, the paint method is overridden to display text on the applet window*

## Embedding an applet in HTML

to include an applet in a Web page, use either

- **APPLET** tag (deprecated)  
CODE specifies applet name, HEIGHT and WIDTH specify window size  
text between the APPLET tags is displayed if unable to execute (e.g., Java not enabled)
- **OBJECT** tag  
preferred for HTML 4, but not universally supported

```
<html>
<!-- Dave Reed   HelloWorld1.html   10/20/01 -->

<head>
<title>Hello World Page</title>
</head>

<body>

<p>
<applet code="HelloWorld1.class" height=100 width=100>
  You must use a Java-enabled browser to view this applet.
</applet>

</body>
</html>
```

[view page in browser](#)

## Hello again

```
import java.awt.*;
import java.applet.*;
import java.util.Random;

/**
 * This class displays lots of "Hello world!"s on the applet window.
 */
public class HelloWorld2 extends Applet
{
    private static final int NUM_WORDS=100;
    private static final Color[] colors =
        {Color.black,Color.red,Color.blue,Color.green,Color.yellow};
    private static Random randy;

    private int randomInRange(int low, int high)
    {
        return (Math.abs(randy.nextInt()) % (high-low+1)) + low;
    }

    public void init()
    {
        randy = new Random();
    }

    public void paint(Graphics g)
    {
        for (int i = 0; i < NUM_WORDS; i++) {
            int x = randomInRange(1, 200);
            int y = randomInRange(1, 200);
            g.setColor(colors[randomInRange(0, colors.length-1)]);
            g.drawString("Hello world!", x, y);
        }
    }
}
```

can change drawing color  
using setColor method

can override init method  
to allocate & initialize  
(similar to a constructor)

## HTML & applets

```
<html>
<!-- Dave Reed   HelloWorld2.html   10/20/01 -->

<head>
<title>Hello World Page</title>
</head>

<body>

<p>
<div align="center">
<table border=1>
<tr><td>

<applet code="HelloWorld2.class" height=200 width=200>
  You must use a Java-enabled browser to view this applet.
</applet>

</td></tr>
</table>
</div>

</body>
</html>
```

an applet can be  
embedded within  
HTML elements just  
like any other element

useful for formatting  
and layout

[view page in browser](#)

## Parameters & applet dimensions

```
import java.awt.*;
import java.applet.*;
import java.util.Random;

public class HelloWorld3 extends Applet
{
    private static final int NUM_WORDS=100;
    private static final Color[] colors =
        {Color.black,Color.red,Color.blue,Color.green,Color.yellow};
    private static Random randy;

    private int randomInRange(int low, int high)
    {
        return (Math.abs(randy.nextInt()) % (high-low+1)) + low;
    }

    public void init()
    {
        randy = new Random();
    }

    public void paint(Graphics g)
    {
        Dimension dim = getSize();
        int numReps = Integer.parseInt(getParameter("reps"));
        for (int i = 0; i < numReps; i++) {
            int x = randomInRange(1, dim.width-65);
            int y = randomInRange(10, dim.height);
            g.setColor(colors[randomInRange(0,colors.length-1)]);
            g.drawString("Hello world!", x, y);
        }
    }
}
```

getParameter  
accesses the values  
of the parameters

must be specified in  
HTML document  
using PARAM tag

getSize returns a  
Dimension object  
that specifies the  
width and height of  
the applet

## Parameters in HTML

```
<html>
<!-- Dave Reed   HelloWorld3.html   10/20/01 -->

<head>
<title>Hello World Page</title>
</head>

<body>

<p>
<div align="center">
<table border=1>
<tr><td>

<applet code="HelloWorld3.class" height=300 width=400>
  <param name="reps" value=100>
  You must use a Java-enabled browser to view this applet.
</applet>

</td></tr>
</table>
</div>

</body>
</html>
```

each parameter must  
have its own PARAM  
tag inside the  
APPLET element

specifies parameter  
name and value

[view page in browser](#)

## Applet graphics

when paint is called, it is given the default Graphics object

- in addition to text, can also draw figures on a Graphics object

commonly used methods of the Graphics class:

```
void drawString(String msg, int x, int y)
void setColor(Color color)

void drawLine(int x1, int y1, int x2, int y2)

void drawRect(int x, int y, int width, int height)
void fillRect(int x, int y, int width, int height)

void drawOval(int x, int y, int width, int height)
void fillOval(int x, int y, int width, int height)
```

```
public class Montel extends Applet
{
    private static Random randy;
    private int NUM_POINTS;
    private int SIZE;

    private int RandomInRange(int low, int high) { CODE OMITTED }
    private double distance(int x1, int y1, int x2, int y2) { CODE OMITTED }

    public void init()
    {
        randy = new Random();
        NUM_POINTS = Integer.parseInt(getParameter("points"));
        Dimension dim = getSize();
        SIZE = Math.min(dim.width, dim.height);
    }

    public void paint(Graphics g)
    {
        g.setColor(Color.red);
        g.fillOval(0, 0, SIZE, SIZE);
        for (int i = 0; i < NUM_POINTS; i++) {
            int x = RandomInRange(0, SIZE);
            int y = RandomInRange(0, SIZE);
            if (distance(x, y, SIZE/2, SIZE/2) < SIZE/2) {
                g.setColor(Color.white);
            }
            else {
                g.setColor(Color.black);
            }
            g.drawLine(x, y, x, y);
        }
    }
}
```

## Graphical applet

init method creates  
random number generator  
& gets parameters

paint method draws a  
circle and a bunch of  
random points

```
<applet code="Montel.class" height=300 width=300>
<param name="points" value=20000>
You must use a Java-enabled browser...
</applet>
```

[view page in browser](#)

## Double buffering

note: paint is called every time the page is brought to the front

- in current version of Monte, this means new dots are drawn each time the page is obscured and then brought back to the front
  - wastes time redrawing
  - dots are different each time the applet is redrawn

the double buffering approach works by keeping an off-screen image

- in the init method (which is called when the page loads):  
*draw the figures on a separate, off-screen Graphics object*
- in the paint method (which is called whenever the page is brought forward):  
*simply display the off-screen image on the screen*

```
public class Monte2 extends Applet
{
    ...
    private Image offScreenImage;
    private Graphics offScreenGraphics;
    ...
    public void init()
    {
        randy = new Random();
        NUM_POINTS = Integer.parseInt(getParameter("points"));
        Dimension dim = getSize();
        SIZE = Math.min(dim.width, dim.height);

        offScreenImage = createImage(SIZE, SIZE);
        offScreenGraphics = offScreenImage.getGraphics();

        offScreenGraphics.setColor(Color.red);
        offScreenGraphics.fillOval(0, 0, SIZE, SIZE);
        for (int i = 0; i < NUM_POINTS; i++) {
            int x = RandomInRange(0, SIZE);
            int y = RandomInRange(0, SIZE);
            if (distance(x, y, SIZE/2, SIZE/2) < SIZE/2) {
                offScreenGraphics.setColor(Color.white);
            }
            else {
                offScreenGraphics.setColor(Color.black);
            }
            offScreenGraphics.drawLine(x, y, x, y);
        }
    }
    public void paint(Graphics g)
    {
        g.drawImage(offScreenImage, 0, 0, null);
    }
}
```

## Buffered applet

init method is called  
when page is loaded

does drawing to a separate,  
off-screen Graphics object

paint is called after init  
and whenever the applet is  
revisited

*Note: don't see image in  
progress*

```
<applet code="Monte2.class" height=300 width=300>
  <param name="points" value=20000>
  You must use a Java-enabled browser...
</applet>
```

[view page in browser](#)

## Better buffering

```
public void paint(Graphics g)
{
    if (offScreenImage == null) {
        offScreenImage = createImage(SIZE, SIZE);
        offScreenGraphics = offScreenImage.getGraphics();

        offScreenGraphics.setColor(Color.red);
        g.setColor(Color.red);
        offScreenGraphics.fillOval(0, 0, SIZE, SIZE);
        g.fillOval(0, 0, SIZE, SIZE);
        for (int i = 0; i < NUM_POINTS; i++) {
            int x = randomInRange(0, SIZE);
            int y = randomInRange(0, SIZE);
            if (distance(x, y, SIZE/2, SIZE/2) < SIZE/2) {
                offScreenGraphics.setColor(Color.white);
                g.setColor(Color.white);
            }
            else {
                offScreenGraphics.setColor(Color.black);
                g.setColor(Color.black);
            }
            offScreenGraphics.drawLine(x, y, x, y);
            g.drawLine(x, y, x, y);
        }
    }
    else {
        g.drawImage(offScreenImage, 0, 0, null);
    }
}
```

if want to see image as it is drawn, must be done in paint

when first loaded, have paint draw on the graphics screen and also to an off-screen buffer

on subsequent repaints, simply redraw the contents of the off-screen buffer

[view page in browser](#)

## GUI elements in applets

Java has extensive library support for GUIs (Graphical User Interfaces)

- has elements corresponding to HTML buttons, text boxes, text areas, ...

each element must be created and explicitly added to the applet

```
nameLabel = new Label("User's name");
add(nameLabel);

nameField = new TextField(20);
nameField.setValue("Dave Reed");
add(nameField);
```

by default, GUI elements are placed in the order they were added, with elements moved to the next line as needed to fit

## Text boxes

```
public class Monte4 extends Applet
{
    . . .
    private Label insideLabel;
    private TextField insideField;
    private Label outsideLabel;
    private TextField outsideField;
    . . .

    public void init()
    {
        randy = new Random();
        NUM_POINTS =
            Integer.parseInt(getParameter("points"));
        Dimension dim = getSize();
        SIZE = Math.min(dim.width, dim.height);

        insideLabel = new Label("Inside:");
        add(insideLabel);
        insideField = new TextField(6);
        add(insideField);

        outsideLabel = new Label("Outside:");
        add(outsideLabel);
        outsideField = new TextField(6);
        add(outsideField);
    }
}

public void paint(Graphics g)
{
    . . .

    insideField.setText("0");
    outsideField.setText("0");

    . . .

    if (distance(x, y, SIZE/2, SIZE/2) < SIZE/2) {
        g.setColor(Color.white);
        int value =
            Integer.parseInt(insideField.getText()+1);
        insideField.setText(""+value);
    }
    else {
        g.setColor(Color.black);
        int value =
            Integer.parseInt(outsideField.getText()+1);
        outsideField.setText(""+value);
    }
    . . .
}
}
```

[view page in browser](#)

## GUI layout

```
public class Monte5 extends Applet
{
    . . .

    public void init()
    {
        randy = new Random();
        NUM_POINTS = Integer.parseInt(getParameter("points"));
        Dimension dim = getSize();
        SIZE = Math.min(dim.width, dim.height);

        setLayout(new BorderLayout());

        Panel p = new Panel();
        insideLabel = new Label("Inside:");
        p.add(insideLabel);
        insideField = new TextField(5);
        p.add(insideField);
        outsideLabel = new Label("Outside:");
        p.add(outsideLabel);
        outsideField = new TextField(5);
        p.add(outsideField);

        add(p, BorderLayout.SOUTH);
    }
    . . .
}
```

Java provides several classes for controlling layout

FlowLayout is default

BorderLayout allows placement of elements around the borders of the applet screen

a Panel can contain numerous elements

[view page in browser](#)

## Event handling

in order to handle events (e.g., text changes, button clicks), can use the *event delegation model*

- must specify that the class implements the `ActionListener` interface

```
public class Monte6 extends Applet implements ActionListener
```

- each source of events must be registered within the applet

```
dotButton = new Button("Click to generate dots");  
dotButton.addActionListener();
```

- must have an `actionPerformed` method to handle events

```
public void actionPerformed(ActionEvent e)  
{  
    if (e.getSource() == dotButton) {  
        drawDots();  
    }  
}
```

## ActionListener

```
import java.awt.*;  
import java.applet.*;  
import java.awt.event.*;  
import java.util.Random;  
  
public class Monte6 extends Applet  
    implements ActionListener  
{  
    . . .  
    private Button dotButton;  
  
    public void init()  
    {  
        randy = new Random();  
        NUM_POINTS =  
            Integer.parseInt(getParameter("points"));  
        Dimension dim = getSize();  
        SIZE = dim.width;  
  
        setLayout(new BorderLayout());  
        dotButton =  
            new Button("Click to generate dots");  
        dotButton.addActionListener(this);  
        add(dotButton, BorderLayout.SOUTH);  
    }  
  
    public void drawDots()  
    {  
        offScreenImage = createImage(SIZE, SIZE);  
        offScreenGraphics  
            = offScreenImage.getGraphics();  
  
        Graphics g = getGraphics();  
  
        offScreenGraphics.clearRect(0,0,SIZE,SIZE);  
        g.clearRect(0,0,SIZE,SIZE);  
  
        . . .  
    }  
  
    public void paint(Graphics g)  
    {  
        g.drawImage(offScreenImage, 0, 0, null);  
    }  
  
    public void actionPerformed(ActionEvent e)  
    {  
        if (e.getSource() == dotButton) {  
            drawDots();  
        }  
    }  
}
```

[view page in browser](#)

## Applet examples

The Java Boutique has lots of sample applets with source code

- [Graphing Calculator](#)
- [Mandelbrot Set](#)
- [Email front-end](#)
- [Web search front-end](#)
- [Java Tetris](#)

## Next week...

### More on applets

- JavaBeans & JARs
- integrating Java with JavaScript
  - accessing an applet from JavaScript
  - accessing JavaScript from an applet

read Chapters 22 & 23

as always, be prepared for a quiz on

- today's lecture (moderately thorough)
- the reading (superficial)