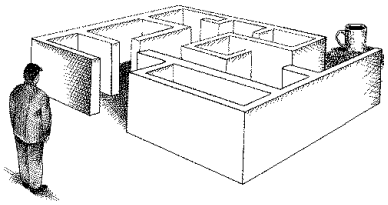
IDES, Sprites, Images, and PixBlts



- Exercise #1
- Java development environments
- Sprites
- Java Images
- PixBlts

Hearn & Baker - pp. 210-211 (Study)

Exercise #1

Set up your course homepage by our next class meeting

The Brain's 6.837 Homepage

Before 6.837 I was a normal lab rat, now I am the BRAIN!

My Projects:

- Project #1
- Project #2
- Project #3
- Project #4
- Project #5

Cool Places:

- MIT's Graphics Group
- Avalon
- My *REAL* homepage
- 6.837's homepage

Narf!

Homepage requirements:

- Locate on imagery in `/mit/imagery4/6.837/F98/username` (you must execute "**add imagetry4**" first)
- Something about you
- Links to all 5 projects
- Area for links to computer graphics sites that you find interesting
- A link back to the 6.837 homepage

```
emacs: Emacs @ eichlid
import java.applet.*;
import java.awt.*;

public class Beno extends Applet {
        Image image;
        int count;

        public void init()
        {
            image = getImage(getDocumentBase(), "World.jpg");
            count = 1;
        }

        public void paint(Graphics g)
        {
            g.drawImage(image, 0, 0, this);
            g.setColor(Color.red);
            for (int y = 15; y < size().height; y += 15) {
                int x = (int) (size().width/2 + 30*Math.cos(Math.PI*y/75));
                g.drawString("Hello", x, y);
            }
            showStatus("Paint called "+count+" time"+((count > 1) ? "s" : ""));
            count += 1;
        }
}
-----Emacs: Beno.java          (Fundamental)----All----
</HEAD>

<BODY>
<H1>Beno Applet</H1>
<P>My favorite class is 6.837</p>
<HR>
<CENTER>
<APPLET code="Beno.class" width=200 height = 200>
</APPLET>
</CENTER>
<HR>
</BODY>
-----Emacs: Beno.html          (Fundamental)----19%----
eichlid5
eichlid5
eichlid5
eichlid5
eichlid5
eichlid5 net

-**-Emacs: *shell*          (Shell: run)----Bot----
```

Integrated Development Environments (IDEs)

What?

Integrated package that includes

- Editor
- Compiler
- Debugger
- Programming Reference
- GUI construction tools
- Code generators

Why?

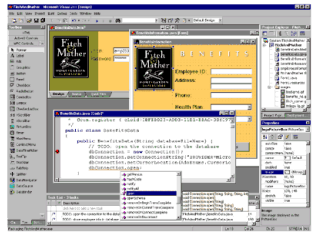
- Java specific hooks
- Interesting developments in *visual* programming aids
- There's more to life than EMACS

Where?, Which?

Microsoft's Visual J++

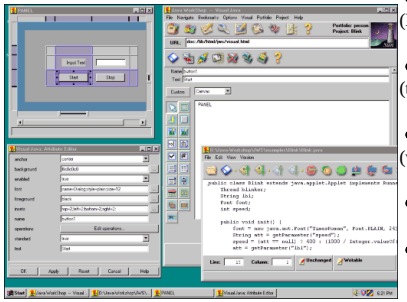
Visual J++ Pro is a popular development environment:

- Visual Studio interface (consistent with all Microsoft compilers)
- Resource Editor & Wizard (for GUI design)
- Good on-line documentation (get to keyword and class docs with a mouse click)
- Good support for large projects
- Weak RAD capabilities
- Inexpensive (with free-trial edition)
- Multi-document interface is awkward
- Poor use of screen space (really needs 1280x1024)
- Win95 and NT platforms only



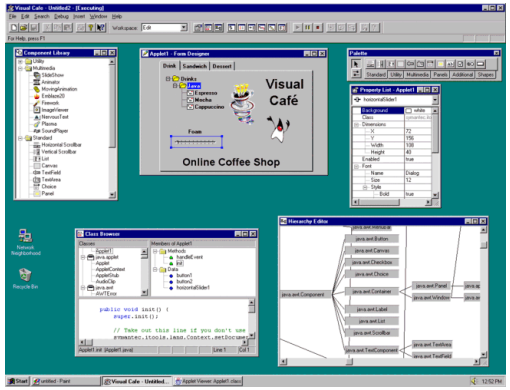
Sun's Java Workshop

The IDE from the folks who brought us Java



- Browser-like interface (HotJava in fact)
- Written in Java (tends to be slow)
- Bulky user interface (wastes screen space)
- Supported by Athena on Sun platforms
- Available on Windows (95 and NT)

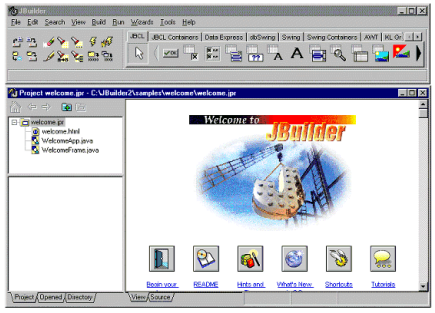
Symantec's Visual Cafe



Visual Cafe

- Earliest IDE
- Consistently top-rated
- Simple visual programming
- Fast compiler
- What I use (now)
- No free download
- Windows and Mac

Borland's JBuilder

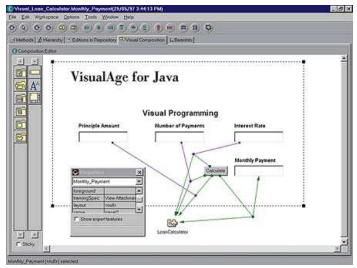


- Another big name IDE
- Handles large projects well
- Good visual programming tools
- Never used it
- Trial version

IBM's Visual Age

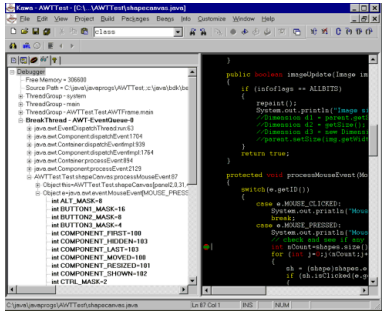
A relatively new player

- Nice visual programming features
- Nice project management tools
- Great hooks for DB programming
- Free evaluation version



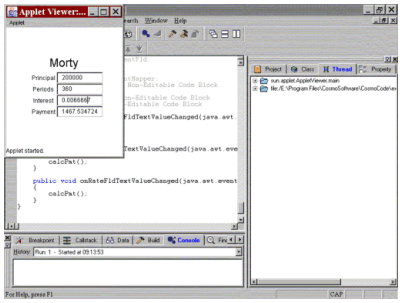
Tek-tool's Kawa

- Best of the Underdogs
- Relies on JDK for compiler
- Very nice IDE environment
- It's free (to you)
- Gaining Momentum



Silicon Graphics CosmoCode

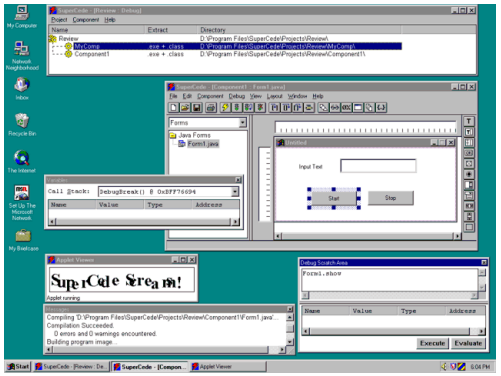
- Supports Windows95, Windows95, & IRIX
- Same folks who did RapidApp for SGI
- Static code analysis
- Trial Version available
- Good visual programming



Asymetrix SuperCede

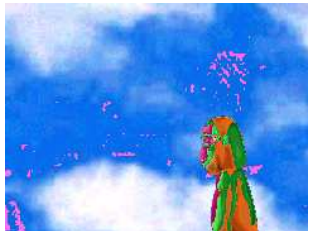
A Glimpse of the Future?

- Creates windows executables (as well as Java class files)
- Fast as native C/C++ implementations
- Multi-window IDE
- Both Java and C++ compilers
- Mix Java and C++ code
- Free download
- I'm going to try it



Sprites

Sprites are graphical objects that move over a background image called a playfield.



Above is a *sub-standard* example of Project #1

A Sprite is a Raster

```
class Sprite extends Raster {
    int x, y;          // current position of sprite on playfield
    int width, height;  // size of a sprite's frame

    public Sprite(Image image);        // initialize sprite with an image
    public void Draw(Raster bgnd);             // draws sprite on a Raster
}
```

Things to consider:

The Draw( ) method must handle transparent pixels, and it must also handle all cases where the sprite overhangs the playfield.

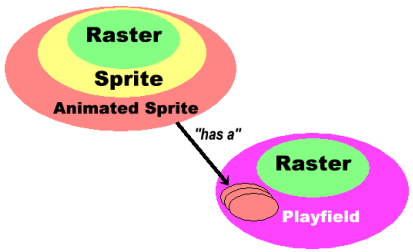
An Animated Sprite is a Sprite

```
class AnimatedSprite extends Sprite {
    int frames;        // frames in sprite
                       // there are other private variables

    public AnimatedSprite(Image images, int frames);
    public void addState(int track, int frame, int ticks, int dx, int dy);
    public void Draw(Raster bgnd);
    public void nextState();
    public void setTrack();
}
```

A track is a series of frames. The *frame* is displayed for *ticks* periods. The sprite's position is then moved (*dx*, *dy*) pixels.

A Playfield is a Raster and has Animated Sprites

```
class Playfield extends Raster {
    Raster background;          // background image
    AnimatedSprite sprite[];    // list of sprites on this playfield

    public Playfield(Image bgnd, int numSprites);
    public void addSprite(int i, AnimatedSprite s);
    public void Draw( );
}
```

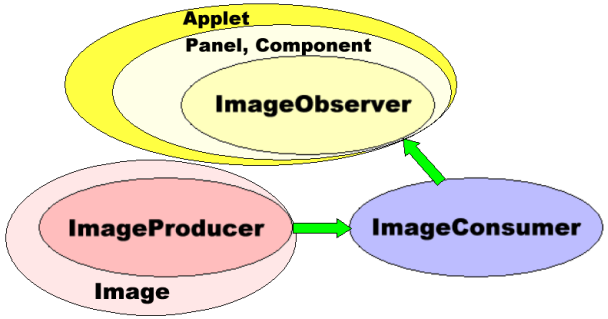


Java Images

The **`java.awt.image`** package is, by any standard, a confusing one. *(Flanagan pg.361)*

There are 3 key interfaces to images in Java

- ImageProducers generate pixels
- ImageConsumers collect pixels into images
- ImageObservers monitor the progress of images



getImage() Methods

- Create Image objects
- Lines up imageProducers who understand the underlying pixel format of the image
- Sets up an imageConsumer to store the image data
- Poorly documented (only supports JPEG and GIF image formats)

getImage() Example

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

public class ImageTest extends Applet {
    Image jpgimg, gifimg;

        public void init() {
        jpgimg = getImage(getCodeBase(),
                "MITdome.jpg");
        gifimg = Toolkit.
                getDefaultToolkit().
                getImage("cowicon.gif");
        }

    public void paint(Graphics g) {
        g.drawImage(jpgimg, 0, 0, this);
        g.drawImage(gifimg, 180, 5, this);
    }
}
```

Download the source here.

Other Image Formats?

What if we want to read in some other image format?

.bmp, .ppm, .tif, .tga, .rgb, .ras, .psd, ...

We must implement an imageProducer to read pixels, an imageConsumer to make the image, and keep imageObservers updated.

Where?
How?

Luckily, We already Have

From our Raster class:

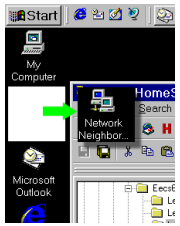
```
public final Image toImage(Component root)
{
    return root.createImage(new MemoryImageSource(width, height, pixel, 0, width));
}
```

The `MemoryImageSource` method is an imageProducer (`root` is an imageObserver).

```
public class ppmDecoder {
    Raster imgRaster;      // declare a Raster

    public ppmDecoder() { }
    public Image getppmImage(URL url) {
        // open url and read image header
            .
            .
            .
        // create imgRaster
            .
            .
            .
        // copy data from file into imgRaster
            .
            .
            .
        return (imgRaster.toImage());
   }
}
```

PixBlts

PixBlts are raster methods for moving and clearing sub-blocks of pixels from one region of a raster to another



Very heavily used by window systems:

- moving windows around
- scrolling text
- copying and clearing

Seems Easy

Here's a PixBlt method:

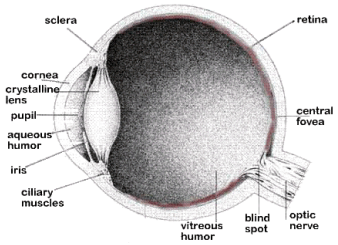
```
public void PixBlt(int xs, int ys, int w, int h, int xd, int yd)
{
    for (int j = 0; j < h; j++) {
        for (int i = 0; i < w; i++) {
            this.setPixel(raster.getPixel(xs+i, ys+j), xd+i, yd+j);
        }
    }
}
```

But does this work?
What are the issues?
How do you fix it?

Next Time

How do we see?

How do we percieve color?

Why do we only need red, green, and blue channels?

What are JPEG and GIF anyway?