VK Computer Games

Mathias Lux & Horst Pichler
Universität Klagenfurt

Agenda

- Game Design – Some Thoughts
- Imaging
  - Imaging Basics, Image Manipulation
  - Sprites & Parallax Scrolling
- Full Screen in Java
- Some more suggestions …

Game Design: Principles

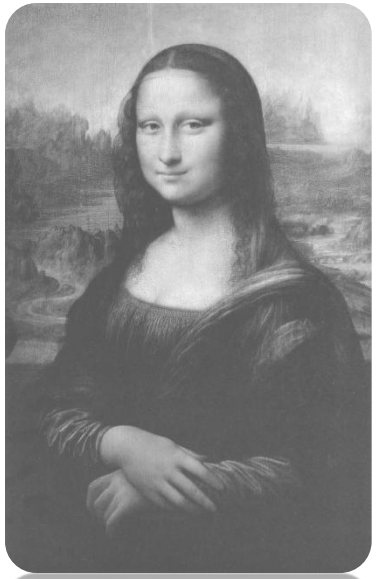
Review game in several aspects:

- Challenge
- Choice
- Clear and Compelling Goals
- Representation
- Conflict
- Feedback

Source: http://www.cs.wisc.edu/graphics/Courses/679-s2007/Main/GameDesign

Game Design: Aesthetics

- What is this needed for?
  - Look / sound / move “nicely”
  - Feel “right”
  - Evoke “right” emotional response
    - Satisfaction, joy, hate, etc.
- Appearance & Mechanics
  - Contribute to a “good” game
  - Eye Candy? – Yes if it contributes to big picture



Aesthetics: Example

Onslaught 2

Challenging Goals

- Premise of the game
  - Story
  - Character
  - Motivation
- Why do I play the game?
- Why do I build towns, jump & run, ...?

Clear Goals

- Different aspects
  - **What** is the goal?
  - **When** is the goal achieved?
- Strongly connected with feedback
  - I need to know when I’m making progress
- Short term vs. long term goals
  - “Get over fire pit” vs. “Rescue princess”

Clear Rules

- Figuring out rules
  - In play: Learning curve?
  - Common sense (gravity, rebound, etc.)
- Unclear rules are frustrating
  - I couldn’t .. because I didn’t know …
- Do not allow workarounds
  - Circumventing != cheating
    - Happens within allowed rule set

Choices

- Player should have meaningful choices
- Consider example choice qualities:
  - Hollow -> No consequence
  - Obvious -> Choice without alternative
  - Informed -> based on provided information cp. guessing
  - Dramatic -> Connects to emotions
  - Weighted -> Both neg. and pos. outcomes
  - Immediate -> Need fast decision
  - Orthogonal -> Choices are independent

Challenge

- Tuning / Balance
  - Make things hard, but not too hard
- Dynamic games
  - Change with game progress / gamers skills
- Challenge from design vs. technical issues
  - Can’t figure out puzzle vs. can’t find button combo

Feedback

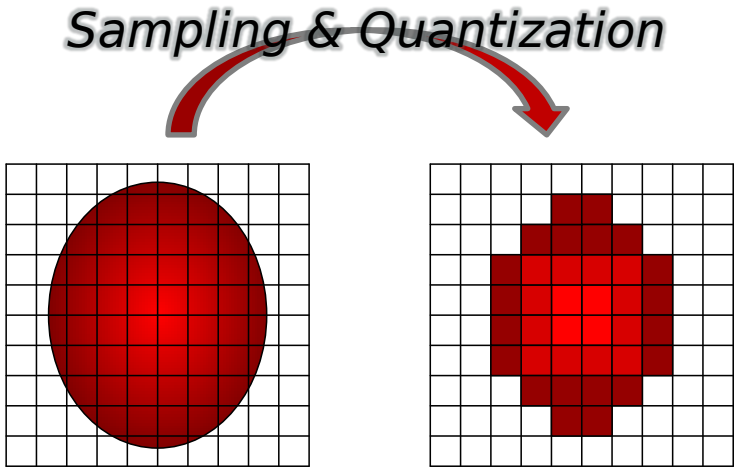
- Action & Reaction
  - Choose a new car and feel the effect ...
  - Buy new clothes & see them on avatar ...
- Gamers need rewards
  - Cp. concept of *highscore* in first course block
- Expericence
  - Buy weapon or skill upgrades
  - Reach new levels & challenges

Agenda

- Game Design – Some Thoughts
- Imaging
  - Imaging Basics, Image Manipulation
  - Sprites & Parallax Scrolling
- Full Screen in Java
- Some more suggestions …

What is an image?

- Basically two types of images:
  - Vector Image
  - Raster Image



Vector Images

- Combination of
  - Atomic elements and
  - Operations
- Example:
  - <circle fill="none" stroke="#000000"
    cx="47.669" cy="47.669" r="41.5"/>
  - <… transform=“matrix(0.24 0 0 0.24 0 0)”/>
- Rendering for presentation
  - Conversion to raster image

Vector Images: Common Formats

- Scalable Vector Graphics
  - Standardized by W3C
  - Supported by QT, Opera, Firefox, Adobe, ...
  - Support in Java by Apache Batik
- Windows Metafile
  - Mostly office clipart

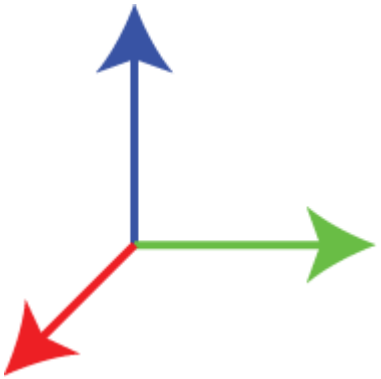
No real support in Java, so we leave it here

Raster Images

- Defined by pixels
  - In rows and columns (e.g. 320x240)
  - Each one has a color value
- Storage Issues:
  - Cp. screen pixels & image pixels
  - Size of raw image
    - 1024 * 768 * 16 = 12.582.912 -> 12 MB
    - Note that 32bit for color are more common -> 24 MB

Color

- Focus on RGB
  - Quantifies red, green and blue parts
  - So each pixel has a
    - Red value
    - Green value
    - Blue value



- Examples:
  - FF0000 (~ 16 Mio. colors, this one is red)
  - EEEEEE (light grey)

Color: Alpha

- In addition the opacity can be quantified
  - Additional channel: Alpha

- Example:
  - FF0000FF (Red, but “invisible”)
  - FF000099 (Red semitransparent)

Alpha: Examples

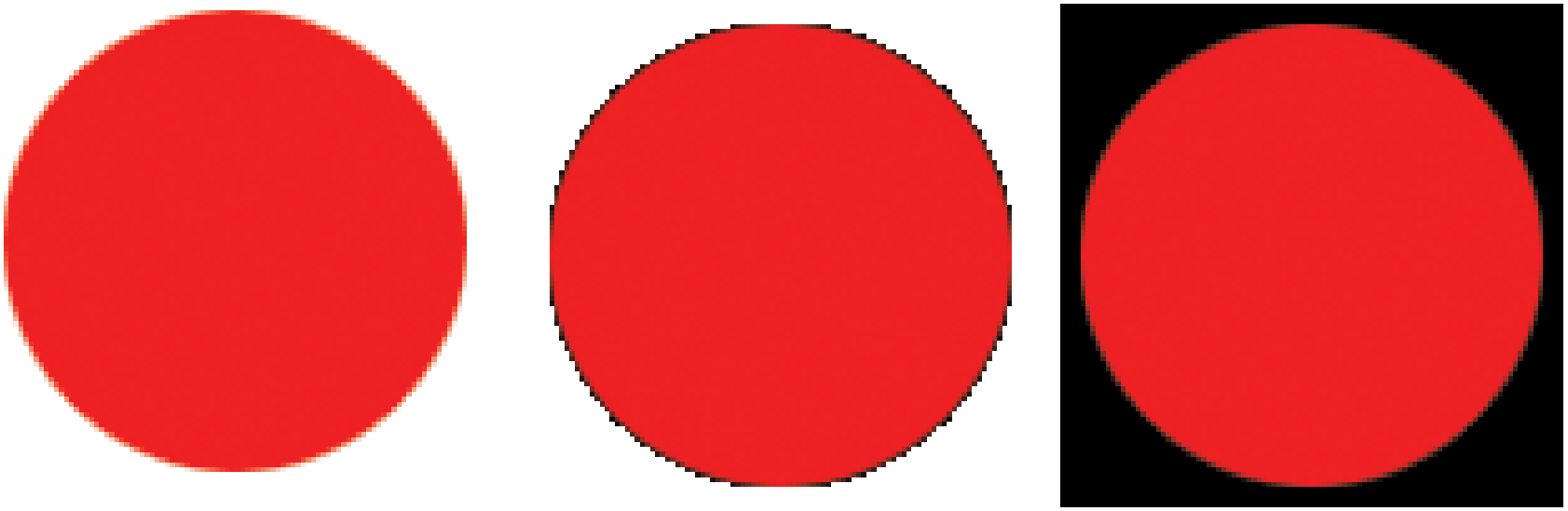


Image Files: Raw Data

- Uncompressed image data
  - PPM, RAW, BMP
  - Benefits:
    - No (de)compression overhead
    - No (de)compression routine needed
  - Drawbacks:
    - File size: $w*h*\log_2(\#colors)$

Image Files: Compressed

- Lossless compression
  - PNG, TIFF are capable of lossless compression
  - No information / quality loss
  - All pixel values can be reconstructed
  - Example: 12.4 kB (PNG) <-> 224 kB (BMP)

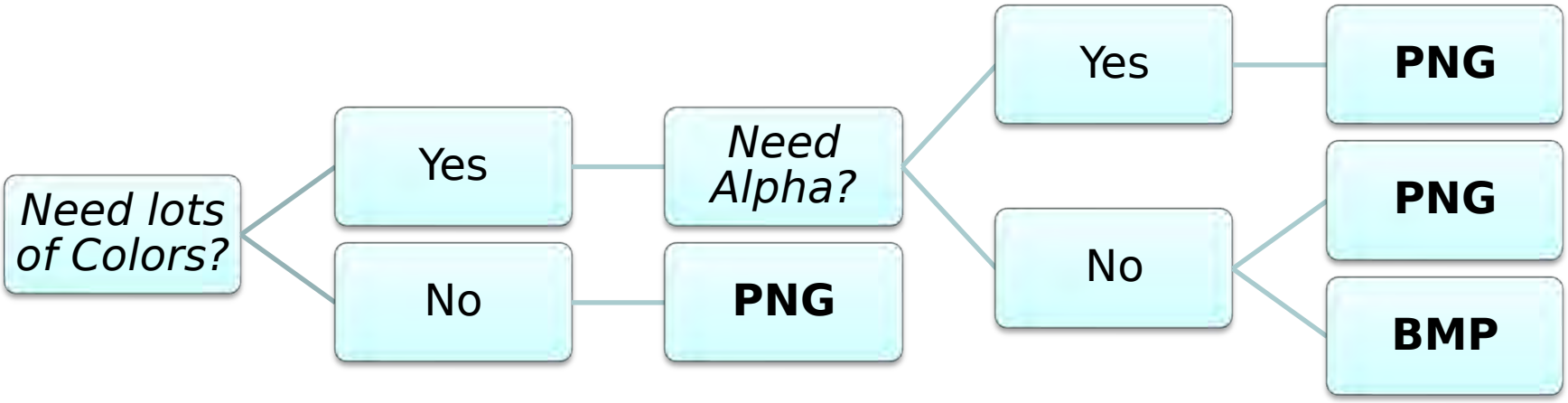
Image Files: Compressed

- Lossy compression
  - JPEG is the most common
  - Trade-off image quality and file size
  - Typical information loss: Block artifacts

Image Files: Compressed

- Reduction of color space
  - PNG (indexed color), GIF (<=256 colors)
  - Minimizes data per pixel

Format Choice for Games?



Format Choice for Games?

- Why not GIF?
  - License issues, PNG does the same and is royalty free.
- Why not JPG?
  - Lossy compression is not needed in domains where one can define graphics.
- Why not TIF?
  - We just need RGB, so there is no need to use anything beside PNG.

Images in Java

- Loading images
  - Use **`javax.imageio.ImageIO.read(…)`**
  - Supports PNG, GIF & JPG
  - Returns a **`BufferedImage`**
- Creating images
  - Use **`new BufferedImage(w,h,type)`**
  - Use **`createGraphics()`** to draw

Image Effects

Java 2D provides extensive image manipulation techniques:

- AffineTransformOp .. spatial transform
- ConvolveOp .. spatial filtering
- RescaleOp .. image scaling

AffineTransformOp

- Employs *AffineTransform* on image
  - 3x3 matrix manually or provided ones:
    - Scale
    - Rotate
    - Shear
    - Translate

ConvolveOp

Spatial Filtering on arbitrary kernel

- What is spatial filtering?
  - Numeric operation on each pixel in an image
- What does this mean?
  - Take for instance a 3x3 matrix (Sobel)

1	0	-1
2	0	-2
1	0	-1

3	4	0	3	3
6	3	0	7	6
2	7	**2**	2	2
4	6	3	3	4
4	6	5	5	4

→

3	4	0	3	3
6	3	0	7	6
2	7	**9**	2	2
4	6	3	3	4
4	6	5	5	4

ConvolveOp

- What does this do?
  - E.g. detect edges ...

ConvolveOp

- Or blur images ...

Gaussian Blur Filter

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

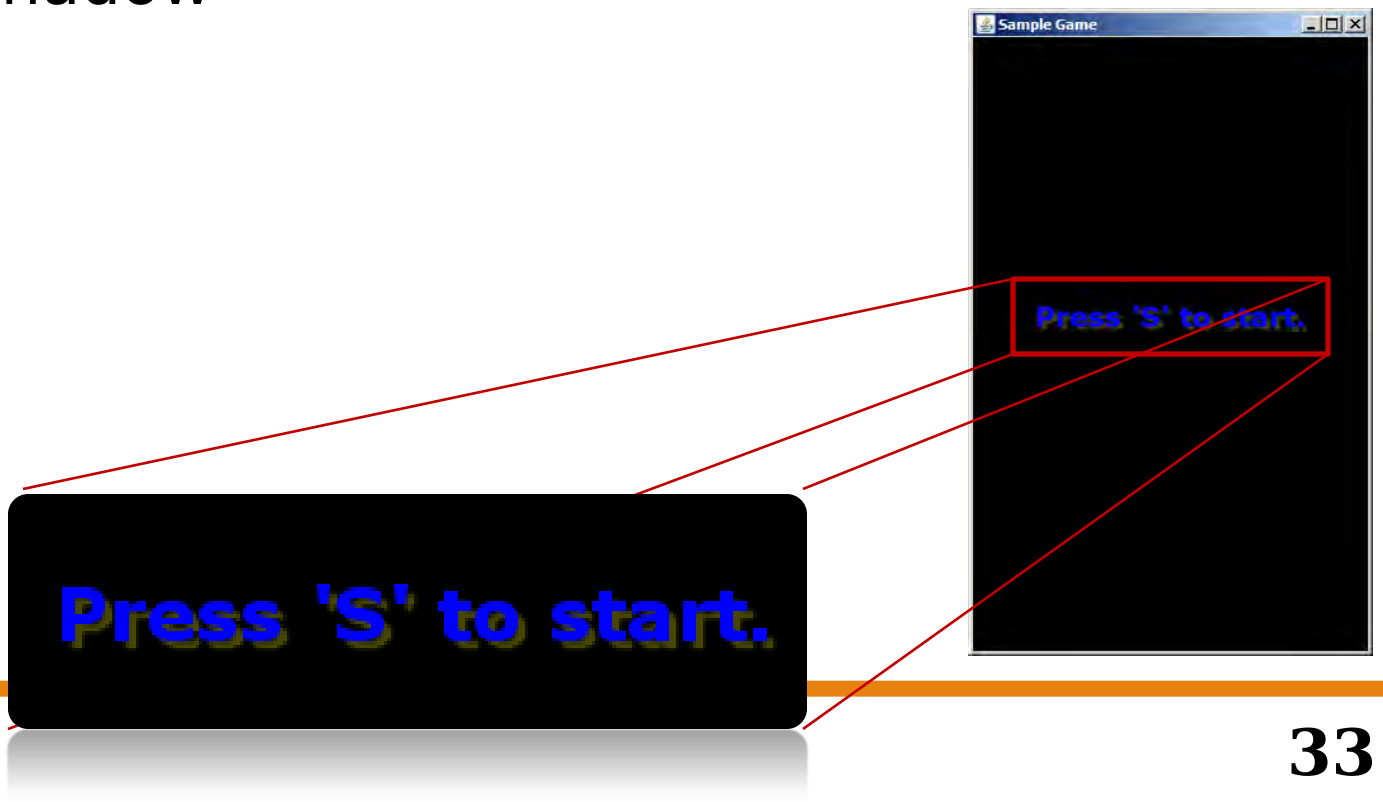
For instance with σ=1

$$\frac{1}{273}$$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

Using Spatial Filtering: Walkthrough ...

- Task: Creating an Info Screen:
  - Display Text
  - Drop Shadow



How to drop shadow ...

- Create a copy of your object
  - Colorize it with your shadow color
  - Move the copy a few pixels
  - Draw and blur the copy
- Draw the actual object

Creating the Kernel ...


```
private static float[] blurKernel;
private static float sigma = 1.2f;
private static int kernelSize = 5;
static {  // creating the blur kernel:
   blurKernel = new float[kernelSize * kernelSize];
   for (int i = 0; i < kernelSize; i++) {
        for (int j = 0; j < kernelSize; j++) {
             blurKernel[i+j* kernelSize] = (float)
                (1/(2*Math.PI*sigma)*Math.exp(-
                (i*i+j*j)/(2*sigma*sigma)));
        }
   }
}
```

Paint the shadow ...


```
private void paintInfo(Graphics2D gra2) {
    BufferedImage binfo = new BufferedImage(getWidth(), getHeight(),
          BufferedImage.TYPE_INT_ARGB);
    Graphics2D g2 = binfo.createGraphics();
    Font myFont = Font.decode("Verdana").deriveFont(Font.BOLD, 22f);
    g2.setFont(myFont);
    infoStr = "Press 'S' to start.";
    Rectangle2D bounds = g2.getFontMetrics().getStringBounds(infoStr, g2);
    g2.setColor(Color.yellow);
    g2.drawString(infoStr,
                  getWidth() / 2 - ((int) bounds.getWidth() / 2 - 4),
                  getHeight() / 2 - ((int) bounds.getHeight() / 2) + 4);
```

Blur the shadow and paint the text ...


```
    // now blur:
    ConvolveOp op = new ConvolveOp(new Kernel(kernelSize,
        kernelSize, blurKernel));
    gra2.drawImage(binfo, op, 0, 0);
    gra2.setFont(myFont);
    bounds = …getStringBounds(infoStr, gra2);
    gra2.setColor(Color.blue.brighter());

    gra2.drawString(infoStr,
        getWidth() / 2 - ((int) bounds.getWidth() / 2),
        getHeight() / 2 - ((int) bounds.getHeight() / 2));
}
```

Demo & Code

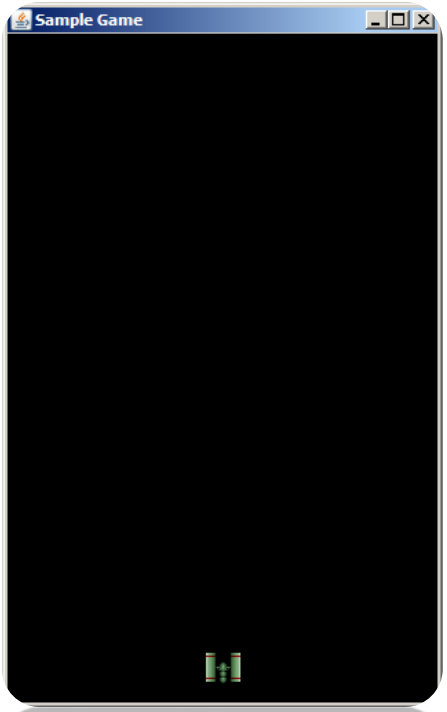
... see Implementation

Creating a Sprite: A Walkthrough

- What is a Sprite?
- Identify resources
  - Select or paint images
- Load and extract images
  - Tiles ...
- Display them
  - Loop, effects

Example: Space Ship

- Simple Task:
  - Draw a space ship
  - Implement movement
- Advanced Tasks
  - Add animation
  - Add zoom
  - Acceleration ...



Loading and drawing …

// Loading file from URL

```
BufferedImage ship = ImageIO.read(…)
```

// In paint method:

```
g.drawImage(ship, …)
```

Implementing a Sprite Object

- Create object ShipSprite
  - Save image(s) in a static variable
  - Store position and movement state
  - Define initialization routine
    - Image loading, calculations, etc.
  - Define *drawSprite(Graphics2D)* method

ShipSprite (1)


```
public class ShipSprite {
    private static BufferedImage ship;
    private int x, y, movement = 4;
    private boolean moveLeft = false, moveRight = false;

    public void setPosition(int x, int y) {
        this.x = x; this.y = y; }

    public void setMoveLeft(boolean moveLeft) {
        this.moveLeft = moveLeft; }

    public void setMoveRight(boolean moveRight) {
        this.moveRight = moveRight; }
    […]
}
```

ShipSprite (2)


```
[…]
public static void initAnimation() {
     try {
          ship = ImageIO.read(
               getClass().class.getResource("./gfx/ship.png"));
     } catch (IOException e) { ... }
 }

 public void drawSprite(Graphics2D g2) {
     if (moveLeft && !moveRight || moveRight && !moveLeft) {
          // implement movement depending on the state:
          if (moveRight) x += movement;
          else if (moveLeft) x -= movement;
     }
     g2.drawImage(ship,
               x-ship.getWidth()/2, y-ship.getHeight()/2, null);
 }
[…]
```

Using ShipSprite

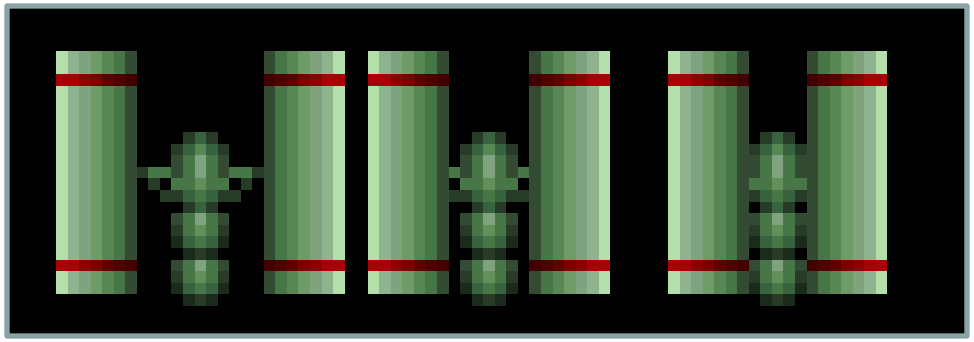

```
// in preload method:
ShipSprite.initAnimation();
ship = new ShipSprite();

// in draw method:
ship.drawSprite(g2);

// in key listener:
if (e.getKeyCode() == KeyEvent.VK_LEFT) {
   ship.setMoveLeft(true);
} else if (e.getKeyCode() == KeyEvent.VK_RIGHT) {
   ship.setMoveRight(true);
}
```

Adding Animation ...

- Image strips ...
  - All possible animation frames in one image
  - Cut it in initialization method
  - Display the right one in each state



SpriteUtils


```
public class SpriteUtils {
    public static BufferedImage[] createSprites(BufferedImage in, int numImages)
    {
        BufferedImage[] result = new BufferedImage[numImages];
        int offset = in.getWidth() / numImages;
        for (int i = 0; i< numImages; i++) {
            BufferedImage bi = new BufferedImage(offset, in.getHeight(),
                    BufferedImage.TYPE_INT_ARGB);
            bi.getGraphics().drawImage(in,
                    0, 0, bi.getWidth(), bi.getHeight(), // destintation rectangle
                    i*offset, 0, (i+1)*(offset), bi.getHeight(), // source rectangle
                    null);
            result[i] = bi;
        }
        return result;
    }
}
```

Animation: ShipSprite


```
private static BufferedImage[] animationSequence;
private int animationState = 0;
[...]

public static void initAnimation() {
    BufferedImage strip = ImageIO.read(
          ExplosionSprite.class.getResource("./gfx/ship.png"));
    animationSequence = SpriteUtils.createSprites(strip, 3);
}
[...]

public void drawSprite(Graphics2D g2) {
    if (moveLeft && !moveRight || moveRight && !moveLeft) {
          // implement animation:
          if (animationState == 0 || animationState == 1)
                animationState++;
```

Zooming the Ship

- Introduce a zoom factor
- Increment and decrement on key release
  - Keys *VK_PLUS* and *VK_MINUS*

- Now draw with scaling:

```
double w = width * zoom;
double h = height * zoom;
g2.drawImage(animationSequence[animationState],
    (int) (x - w / 2), (int) (y - h / 2),
    (int) w, (int) h, null);
```

Acceleration


```
if (moveLeft && !moveRight || moveRight && !moveLeft) {
    // accelerate on button pressed:
    if (moveRight) currentMovement += acceleration;
    else if (moveLeft) currentMovement -= acceleration;
} else {  // decelerate
    if (currentMovement<0) currentMovement += acceleration;
    else if (currentMovement>0) currentMovement -= acceleration;
    if (Math.abs(currentMovement)<1) currentMovement=0;
}
// actually move the ship:
long moveDiscrete = Math.round(currentMovement);
x+= moveDiscrete;
// determine acceleration state:
if (moveDiscrete==0) animationState = 0;
else if (Math.abs(moveDiscrete)>=4) animationState = 2; // fast
else animationState = 1; // accelerating
```

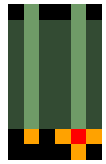
Demo & Code

... see Implementation

Possible additions

- Adding a gravity center
  - Movement proportional to mass & distance
- Adding a dingi
  - Springs: Movement proportional to distance
  - $F = -kx$
    - x … displacement vector
    - k … spring constant
    - F … force

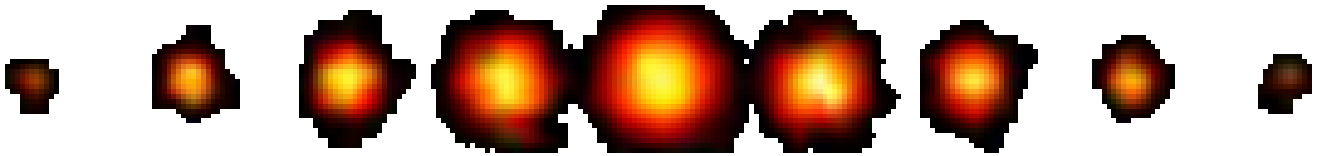
Rocket



- Fired on <space>
- Member of *GamePanel*
  - Only one allowed at a time
- Acceleration
  - The longer it moves the faster it gets
- Removed if out of sight
  - Sprite should be re-used (e.g. ammo)
- Simple Sprite with 2-frame animation

Explosion

- Rocket explodes on <space>
- Animation with 9 different frames
  - No alpha ...
- Member of *GamePanel*
- Removed if burned out



Demo & Code

... see Implementation

Parallax Scrolling

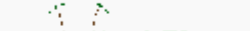
- Common Technique for 2.5D
  - In contrast to “real 3D”
- Simulates depth with multiple layers
  - Each layer moves with different speed
- Side scrollers
  - Games moving from left to right (Mario, etc.)

Parallax Scrolling

Background layer: a starry sky.

Layer 1: a chain of mountains.

Layer 2: background vegetation.



Layer 3: foreground vegetation and path.

Source: http://en.wikipedia.org/wiki/Parallax_scrolling

Starfield Simulation: A Walkthrough

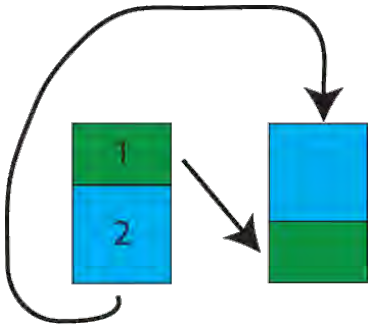
- Create 3 different layers
- Load them during startup
- Display them with wrap around
- Move them in different speeds

Starfield: Loading Images


```
private void initField() {
   stars = new BufferedImage[numberOfLayers];
   position = new int[numberOfLayers];
   for (int i = 0; i < stars.length; i++) {
        try {
               stars[i] = ImageIO.read(...);
               position[i] = 0;
        } catch (IOException e) {
               e.printStackTrace();
        }
   }
}
```

Starfield: Display and Wrap Around


```
public void drawSprite(Graphics2D g2, Rectangle2D dest) {
    int dx = (int) dest.getWidth();
    int dy = (int) dest.getHeight();
    for (int i = 0; i < stars.length; i++) {          // painting the layers:
        g2.drawImage(stars[i],
                     0, position[i], dx, dy,      // destination
                     0, 0, dx, dy - position[i], // source
                     null);
        g2.drawImage(stars[i],
                     0, 0, dx, position[i],        // destination
                     0, dy - position[i], dx, dy, // source
                     null);
        // moving the layers:
        position[i] += (i + 1) * 2;
        position[i] = position[i] % dy;
    }
}
```



Starfield: Performance

- Performance issues with Java
  - Translucent images are not rendered with hardware acceleration.
  - This has to be turned on explicitly on Windows

```
java -Dsun.java2d.translaccel=true …
```

- Better: Draw stars yourself

Demo & Code

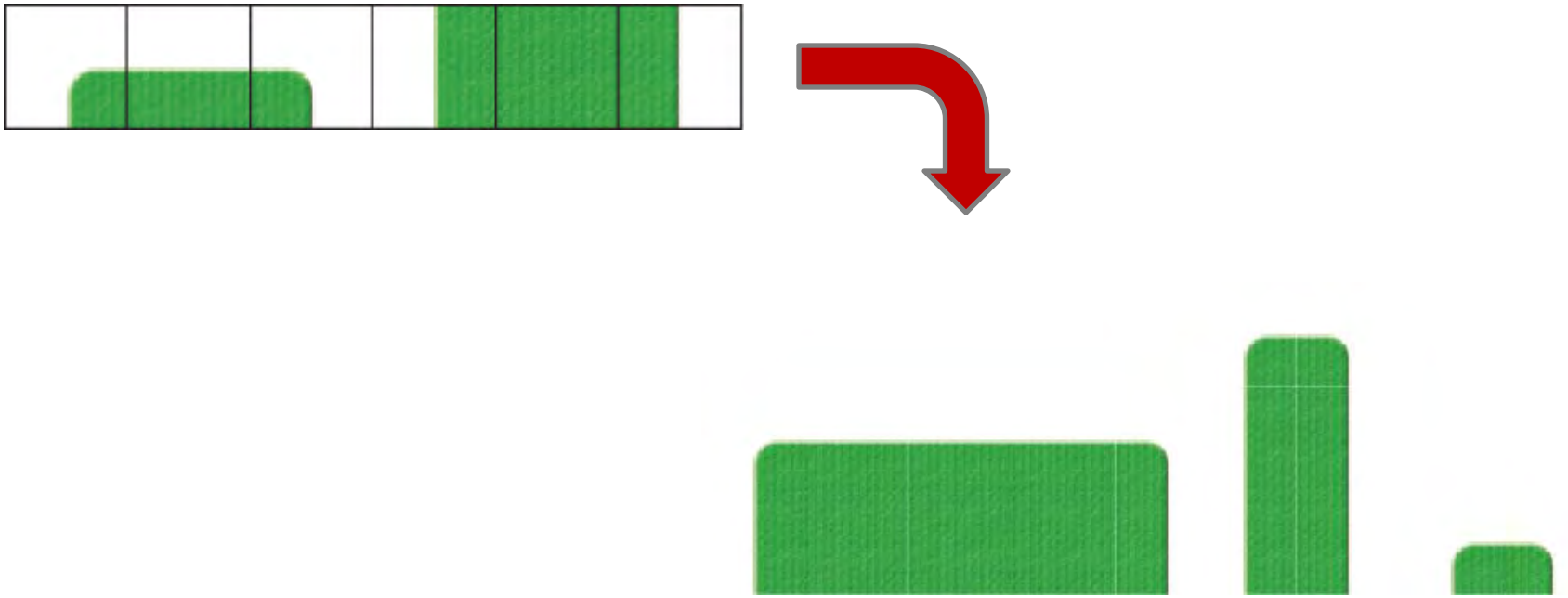
... see Implementation

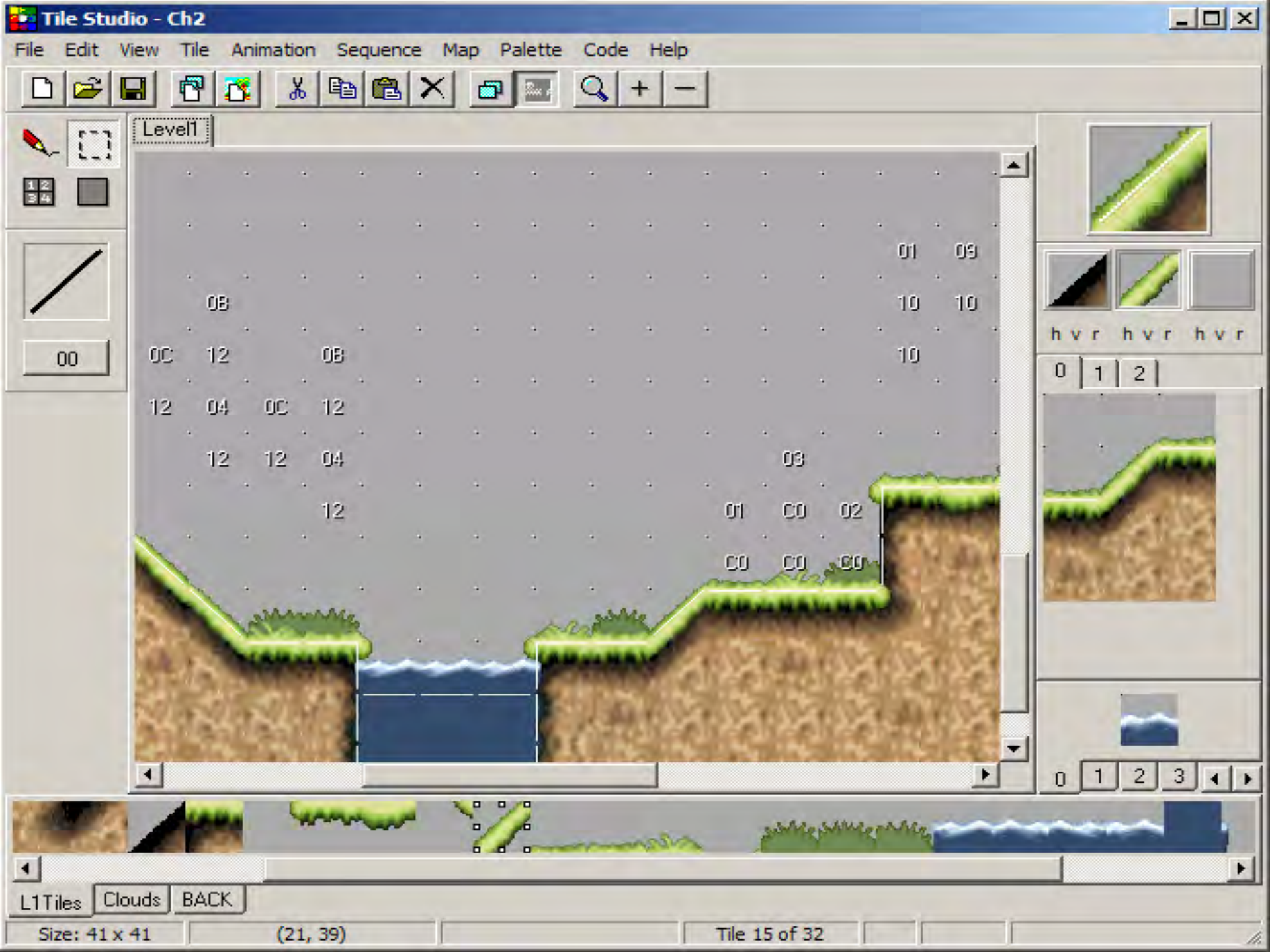
More 2.5D Tricks

- Assume top-down view on landscape
  - Draw shadow
    - Use translucent color -> `new Color(0, 0, 0, 128)`
    - While scrolling move and scale shadow
    - Creates illusion of uneven terrain
  - Implement jump action of sprite:
    - Move and scale shadow
    - Scale sprite

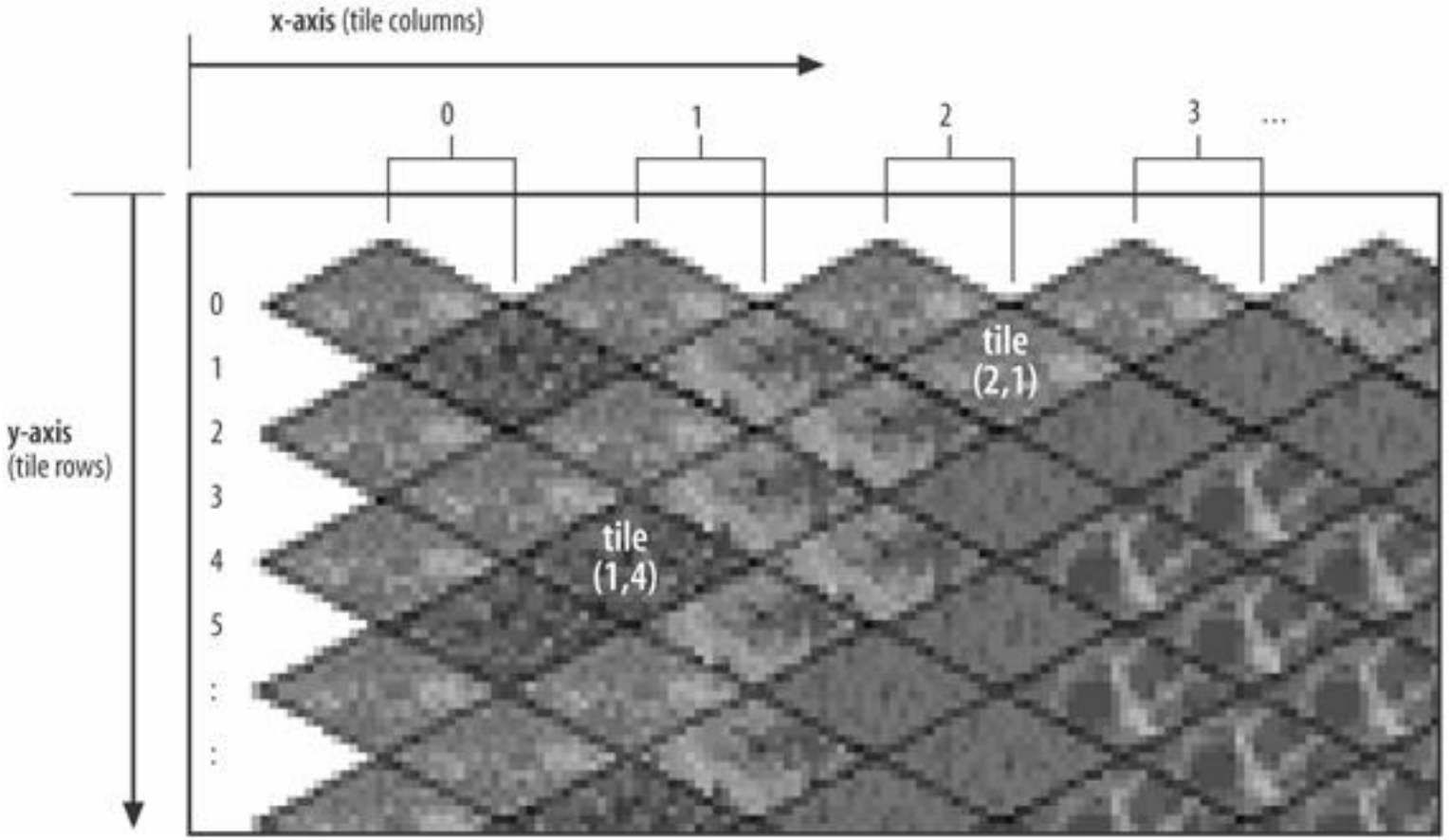
Image Tiles ...

- Common technique to “create worlds”
- Add up small tiles to big picture



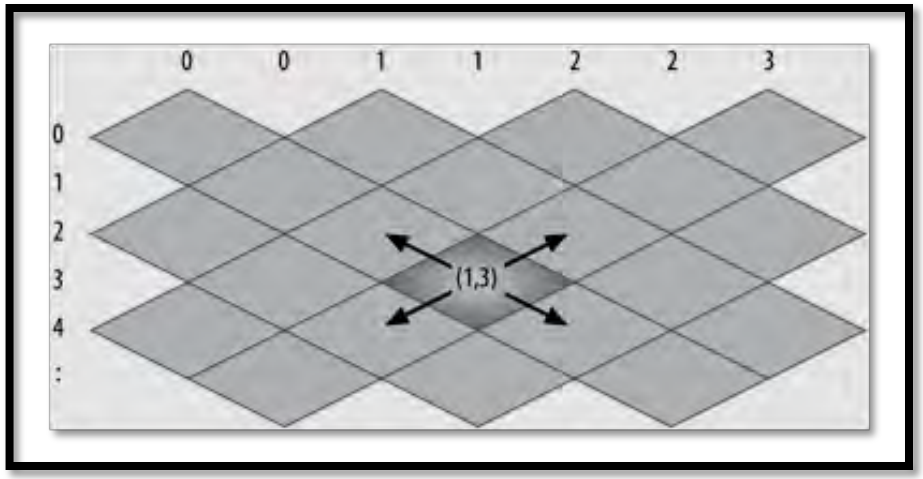

Tile Studio - Ch2
File Edit View Tile Animation Sequence Map Palette Code Help
Level1
00
01 09
08 10 10
0C 12 08 10
12 04 0C 12
12 12 04 03
12 01 C0 02
C0 C0 C0
h v r h v r h v r
0 1 2
0 1 2 3
L1Tiles Clouds BACK
Size: 41 x 41
(21, 39)
Tile 15 of 32

Isometric Tiles



Isometric Tile Games

- Render back to front
  - Support for sprites (trees, characters, etc.)
- Movement
  - From tile to tile (animated?)
  - World “moves”



Demo ...

... show Video

Agenda

- Game Design – Some Thoughts
- Imaging
  - Imaging Basics, Image Manipulation
  - Sprites & Parallax Scrolling
- **Full Screen in Java**
- Sound
  - Sound Basics & Effects
- Some more suggestions …

Full Screen Games

- Why full screen?
  - Screens bigger than possible game resolutions
  - Locking out other programs
  - Use as much screen as possible (small devices)

- Full screen modes
  - Full screen is not the same as full screen
  - Using multiple monitors?

Almost Full Screen (ASF)

- Maximized window
  - Title bar and border are visible
  - Task bar (Windows, KDE, Gnome, ..) is visible
- Common window interactions disabled
  - No moving (snaps back to original position)
  - No resizing

Almost Full Screen (ASF)


```
setSize(…) // get from Toolkit.getDefaultToolkit( );
addComponentListener(
   new ComponentAdapter( ) {
        public void componentMoved(ComponentEvent e) {
              setLocation(0,0);
        }
   });

setResizable(false);
```

Undecorated Full Screen (USF)

- Window decorations are removed
  - No title bar and no border
  - Task bar is “behind” the game
- Do not forget …
  - Means to exit and pause the game

Undecorated Full Screen (USF)


```
// in game panel:
Toolkit tk = Toolkit.getDefaultToolkit( );
Dimension scrDim = tk.getScreenSize( );
setPreferredSize(scrDim);     // set JPanel size

// in game frame
setUndecorated(true);     // no borders or titlebar
setIgnoreRepaint(true);   // turn off paint events
```

Full Screen Exclusive Mode (FSEM)

- Suspends most of window-environment
  - Bypassing AWT & Swing
  - Almost direct access to screen
  - Possible to change resolution
- Issues with
  - Possibly limited RAM (VRAM)
  - Resource management of OS (VRAM)
- Typically increases framerate

See also: http://java.sun.com/docs/books/tutorial/extra/fullscreen/

Full Screen Exclusive Mode (FSEM)

- Get **`GraphicsDevice`** (screen) instance
- Remove borders & title:

```
setUndecorated(true)
```

- Check support:

```
gd.isFullScreenSupported( )
```

- Go to full screen:

```
gd.setFullScreenWindow(this)
```

- Eventually change display mode

```
setDisplayMode(800, 600, 8);
```

- Code? Consult tutorial on java.sun.com/docs

Agenda

- Game Design – Some Thoughts
- Imaging
  - Imaging Basics, Image Manipulation
  - Sprites & Parallax Scrolling
- Full Screen in Java
- Some more suggestions …

Some additional notes ...

- Splash screens
- Physics
- Resources in Java
- Distribution & Packaging
- Java Web Start

Splash Screens

- Supported by VM since Java 1.6
  - Prepare a splash screen image
    - Alpha channel supported -> PNG
  - Start application with
    `java -splash:image.png <class-name>`
- Modify splash screen like this:

```
SplashScreen splash = SplashScreen.getSplashScreen();
Graphics2D g = (Graphics2D)splash.createGraphics();
Dimension size = splash.getDimension();
```

Physics

- Physics
  - Use trigonometric functions for arcs
    - Also simulates quadratic dependencies
    - Can be “prepared” in a table
  - Acceleration is linear
    - Movement += acceleration
    - Movement -= acceleration
  - Compute no discrete movements

Resources in Java

- Resources in Java
  - Loaded via URL
  - Works in classpath, network, filesystem, etc.

```
// relative to class
ImageIO.read(getClass().getResource("gfx/1.png"));
Applet.newAudioClip(getClass().getResource("sfx/1.mid"));

// from root classpath
ImageIO.read(getClass().getResource("/resources/gfx/1.png"));
Applet.newAudioClip(getClass().getResource("/resources/sfx/1.mid"));
```

Distribution & Packaging

- Creating a jar file
  - Write a manifest
    - `Main-Class: uniklu.games.spaceship.GameFrame`
      `SplashScreen-Image: splash.png`
  - Jar the classes
    - `jar cmf manifestFile myFile.jar -c classes *.class`
  - Eventually use Ant

Deploy by Ant Build


```
<target name="compile" depends="init">
    <echo file="default.mf" append="false">Main-Class: uniklu.games.spaceship.GameFrame
SplashScreen-Image: splash.png
    </echo>
    <copy todir="${build}/uniklu">
           <fileset dir="${src}/uniklu">
                      <exclude name="**/*.java"/>
                      <exclude name="**/CVS/*"/>
           </fileset>
    </copy>
    <!-- actual compile -->
    <javac srcdir="${src}" destdir="${build}" optimize="on" verbose="true" listfiles="true"/>
</target>

<target name="clean" >
           <delete dir="${build}"/>

</target>

<target name="dist" depends="clean,compile">
    <jar jarfile="SampleGame.jar" basedir="${build}" manifest="default.mf"/>
</target>
```

Java Web Start

- Allows to provides Java apps on the web
- Description file is used for start
  - JNLP file
  - Linked from website, etc.
- Java Web Start client checks
  - Version
  - Libraries
  - Security

Sample JNLP


```
<?xml version="1.0" encoding="UTF-8"?>
<jnlp spec="1.0+" codebase="dist" href="launch.jnlp">
    <information>
        <title>Some game</title>
        <vendor>Mathias Lux</vendor>
        <description>Fly the space ship</description>
        <description kind="short">fly</description>
        <homepage href=""/>
    </information>
    <security>
        <all-permissions/>
    </security>
    <resources>
        <j2se version="1.5+"/>
        <jar href="Game.jar" main="true" download="eager"/>
        <jar href="lib/Jama-1.0.2.jar" download="eager"/>
        <jar href="lib/looks-2.1.4.jar" download="eager"/>
    </resources>
    <application-desc main-class="start.GameFrame">
    </application-desc>
</jnlp>
```

Thanks ...

... for your attention