Computer Games 2014
Game Development Basics

Dr. Mathias Lux

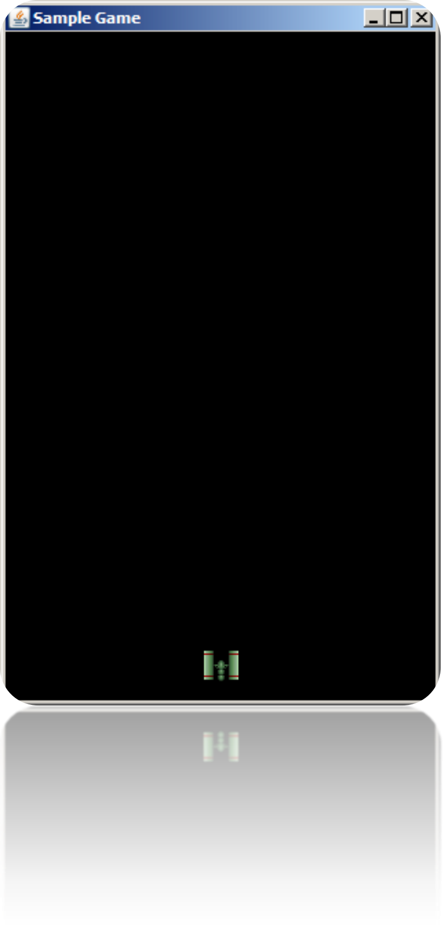
Klagenfurt University

Agenda

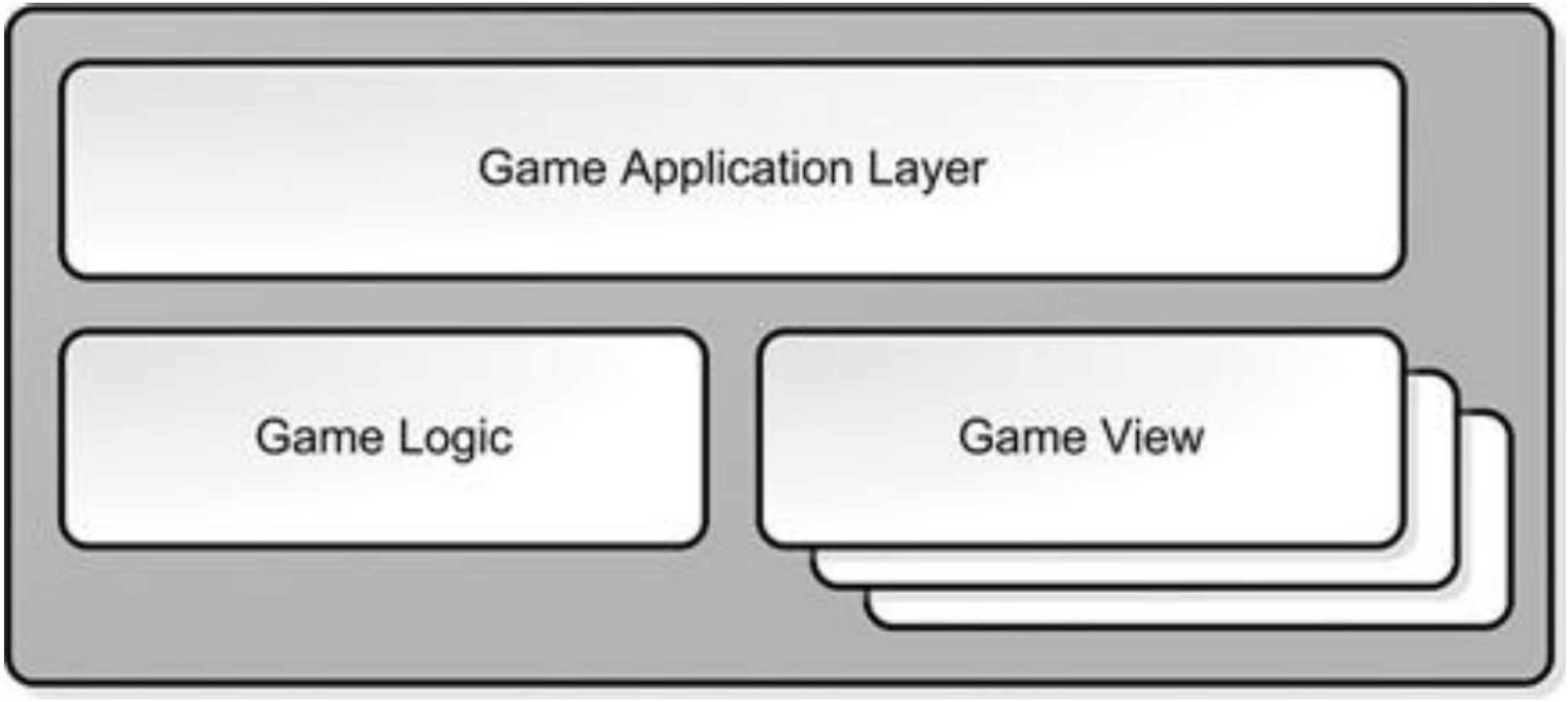
- Game Loop
- Sprites & 2.5D
- Images

Example: Space Ship

- Simple Game:
  - A single space ship
  - Moving left to right
- Advanced Tasks
  - Firing rockets
  - Explosions
  - Sound & music

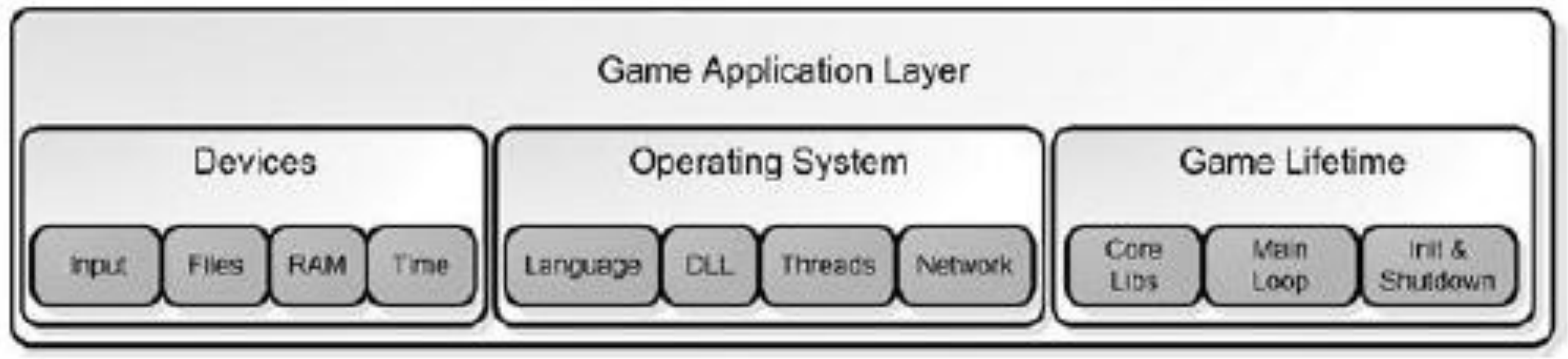


High Level Game Architecture



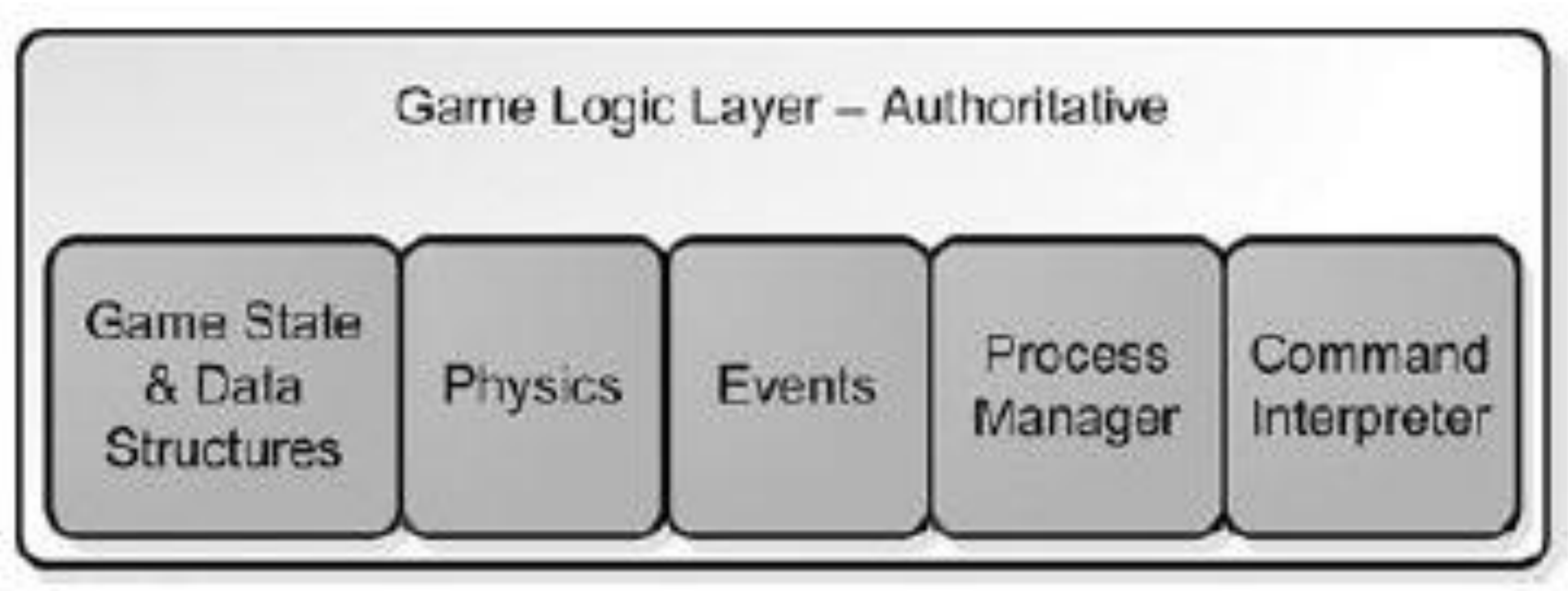
src. Mike McShaffry et al., Game Coding Complete, 3rd Edition

Game Application Layer



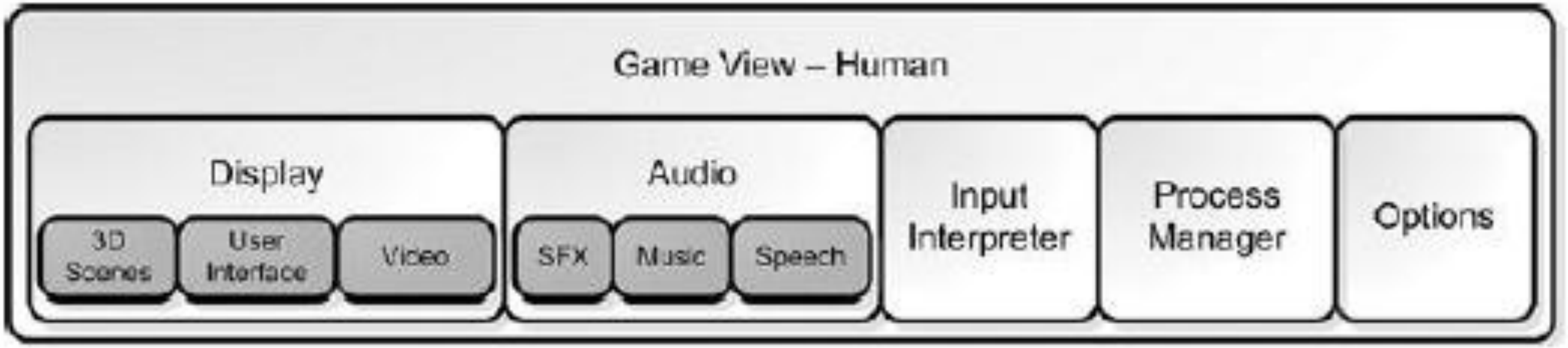
src. Mike McShaffry et al., Game Coding Complete, 3rd Edition

Game Logic


Game Logic Layer – Authoritative
Game State & Data Structures
Physics
Events
Process Manager
Command Interpreter

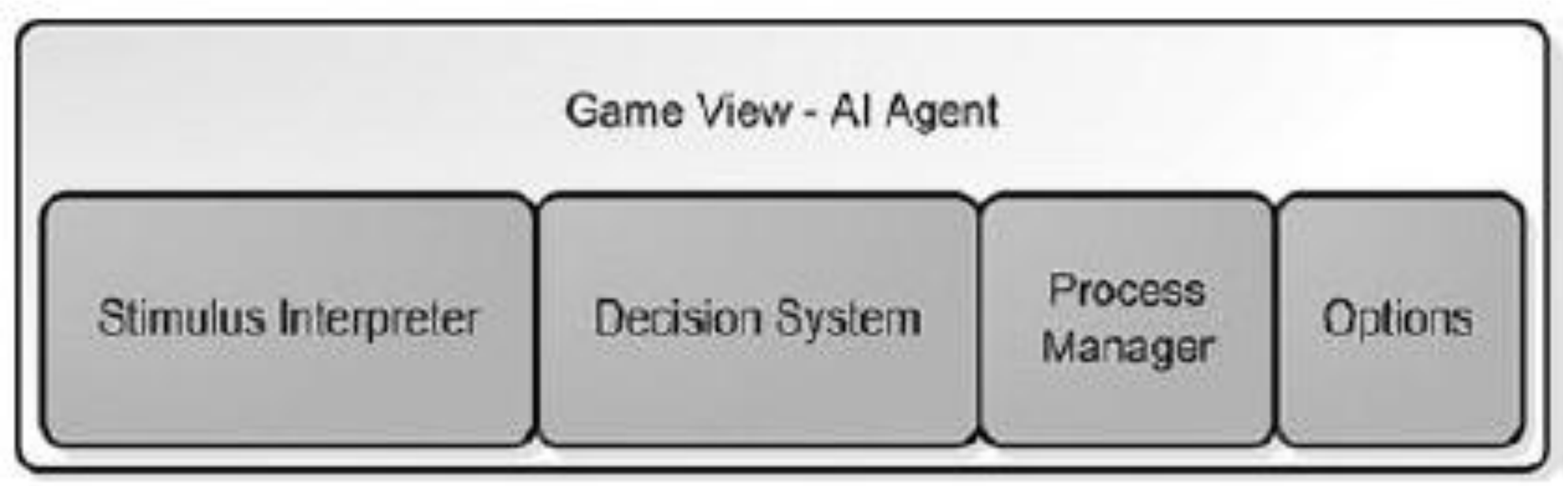
src. Mike McShaffry et al., Game Coding Complete, 3rd Edition

Game View (Human)



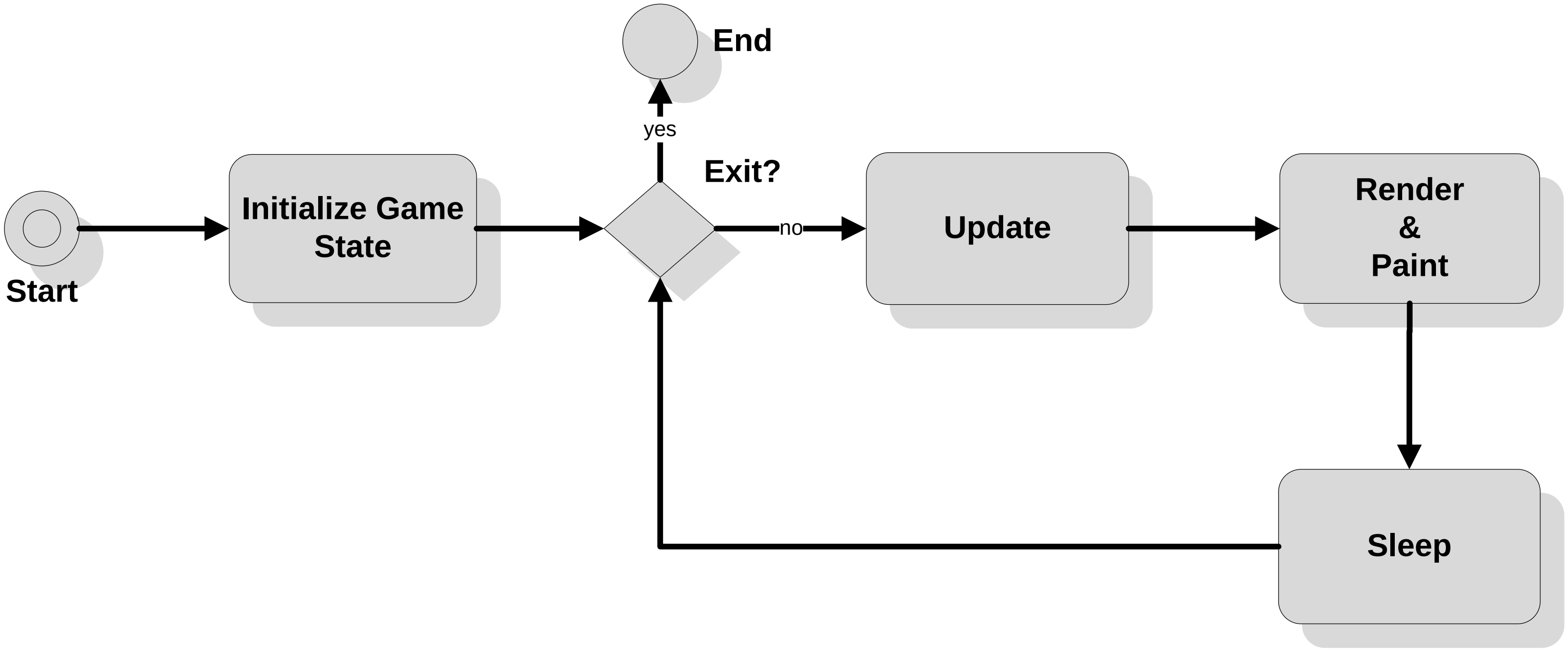
src. Mike McShaffry et al., Game Coding Complete, 3rd Edition

Game View (AI)



src. Mike McShaffry et al., Game Coding Complete, 3rd Edition

Game Loop


End
yes
Exit?
Start
Initialize Game State
no
Update
Render & Paint
Sleep

Game Loop

- while(user doesn't exit)
  - check for user input
  - run AI
  - move objects
  - resolve collisions
  - draw graphics
  - play sounds
- end while

Check for user input

- Get state of keys
  - e.g. is <space> key pressed
- initiate action
  - e.g. spawn rocket

Run AI

- Check current state
- Initiate action
  - spawn UFOs,
  - drop bombs,
  - change paths etc.

Move Objects

- Move objects
  - along their (changed) paths
  - matching their (changed) velocity

Collision Detection

- Check if
  - either there is a crossing in paths
  - or a double setting of pixels
- Pixel based vs. boundary based
- Runtime issues
  - Grid based, data structures etc.

Draw Graphics

- Direct engine
  - to allocate resources
  - to paint the buffer
  - then flip the buffer

Play Sounds

- Decode sounds
  - maintain storage
- Fill buffer
  - to be played
- Trigger events
  - explosions, sounds, etc.

Game Loop

- Frames per second
  - 20 or more are minimum
  - 60+ frames are optimum
  - jitter is a problem (sync to display device)
- Stereoscopic 3D needs double frame rates

Game Loop

- Parallel processing
  - Xbox has 3 cores (with HT)
  - PS3 has 8 cores
- Game loops run in parallel
  - AI loop
  - sound & painting loop
  - control loop

Agenda

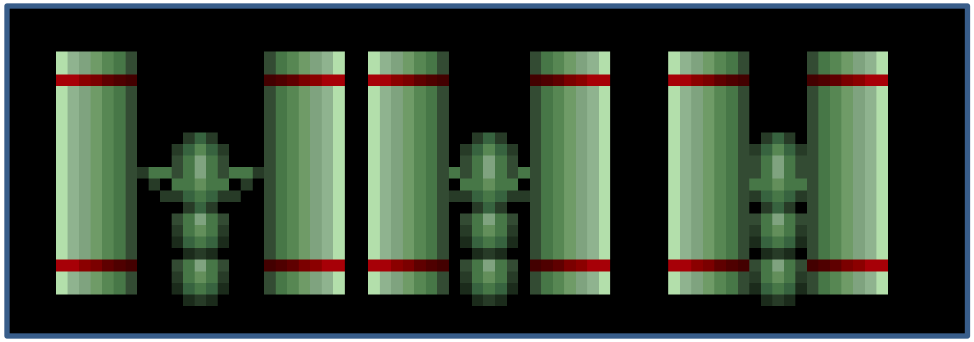
- Game Loop
- Sprites & 2.5D
- Images

Sprites

- What is a sprite?
  - – A (moving) object on the screen
- Resources needed
  - – visuals, audio, state
- Loading and displaying
  - – game loop, effects, resources needed in time

Simple Sprite Animation

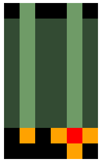
- Image strips …
  - All possible animation frames in one image
  - Cut it in initialization method
  - Display the right one in each state



Features

- Left-right movement
  - spring based physics
  - “feels more real”

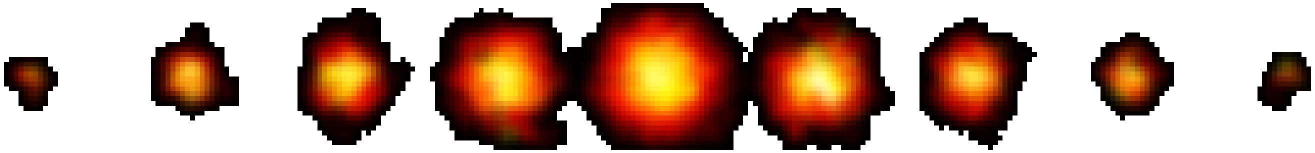
Rocket



- Another sprite
  - 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)
  - Too many sprites consume too much memory
- Simple sprite with 2-frame animation

Explosion

- Rocket explodes
  - rocket is removed
  - explosion sprite is displayed
- Animation with 9 different frames
  - No alpha ...
- Removed when over

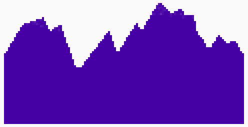


Parallax Scrolling

- 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

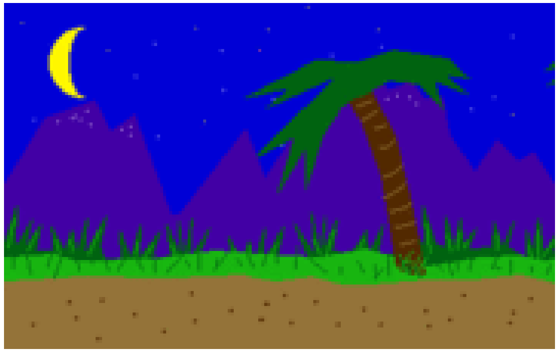
Background layer: a starry sky.



Layer 1: a chain of mountains.

Layer 2: background vegetation.

Layer 3: foreground vegetation and path.



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

Demo-Video

- California Games

Starfield Simulation

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

Starfield: Performance

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

- Better: Draw stars yourself

More 2.5D Tricks

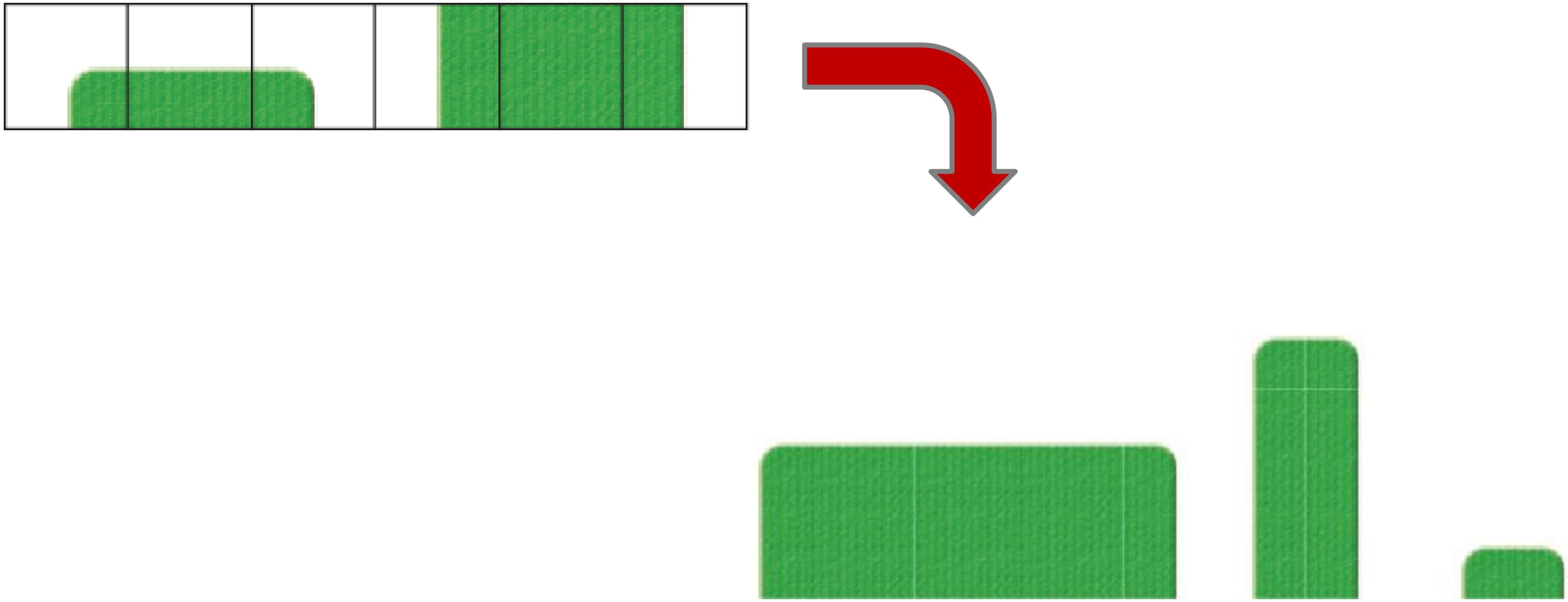
- Assume top-down view on landscape
  - Draw shadow
    - Use translucent color
    - While scrolling move and scale shadow
    - Creates illusion of uneven terrain
  - Implement jump action of sprite:
    - Move and scale shadow
    - Scale sprite

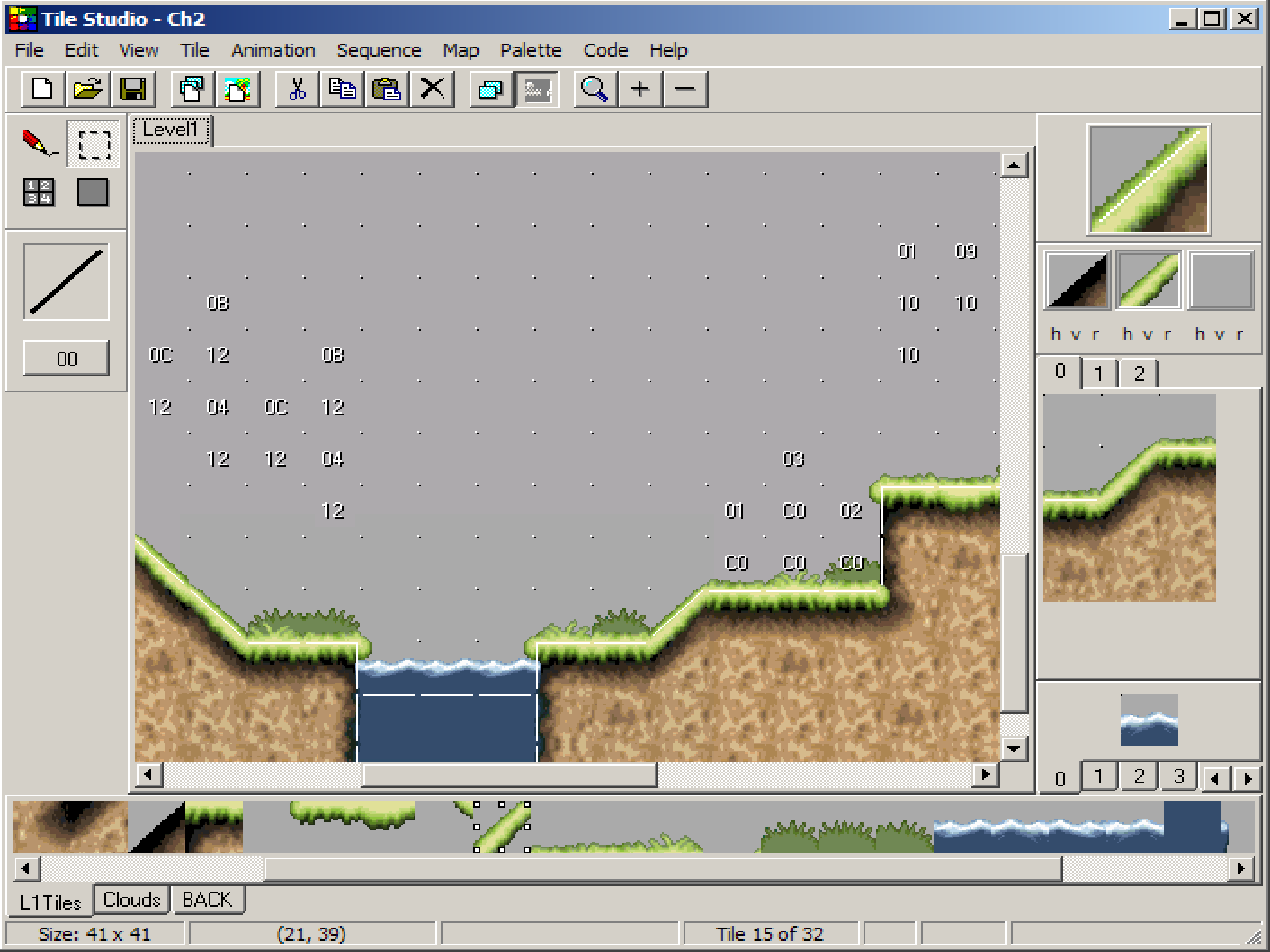
Demo

Video: 1942

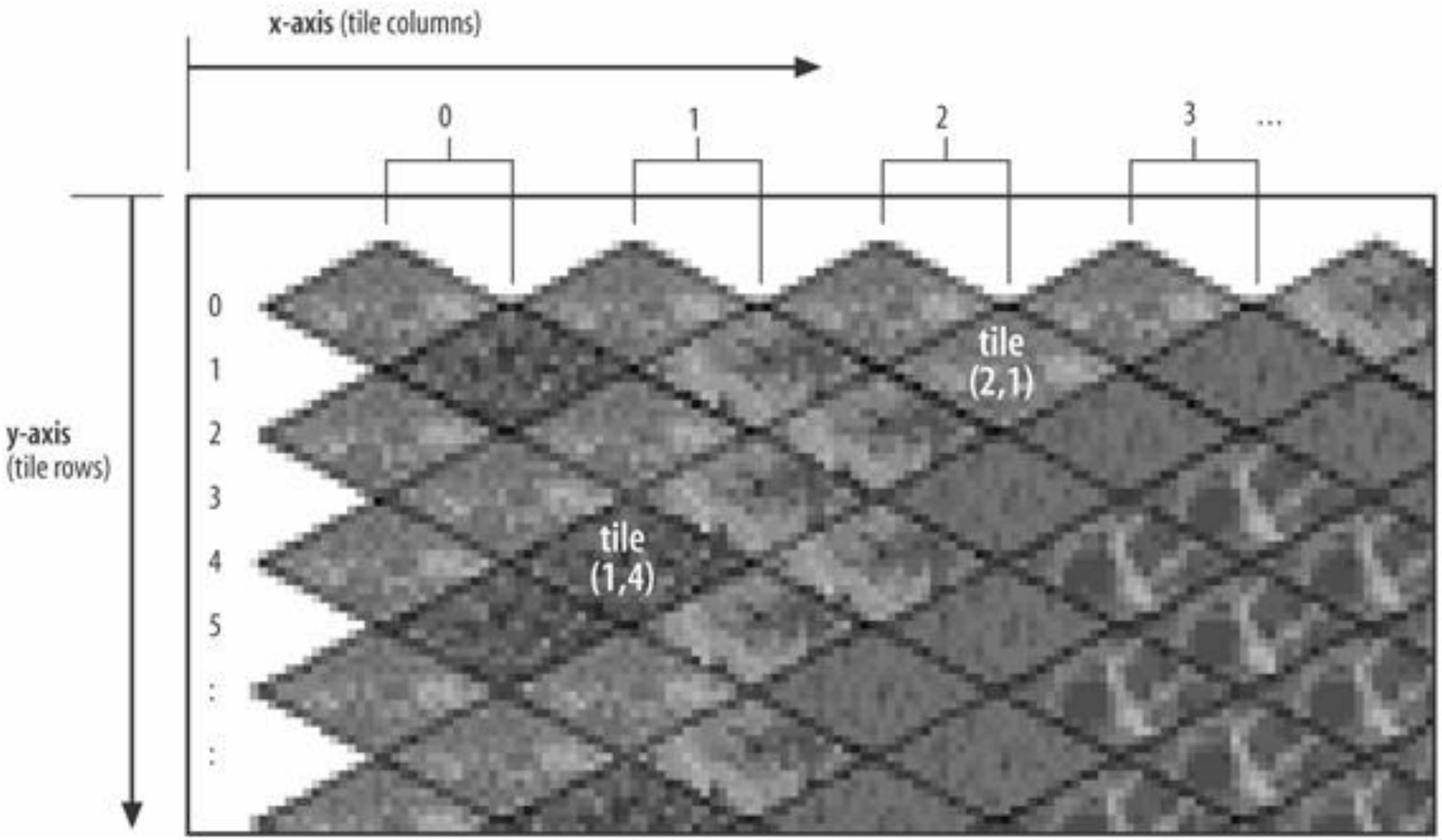
Image Tiles …

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



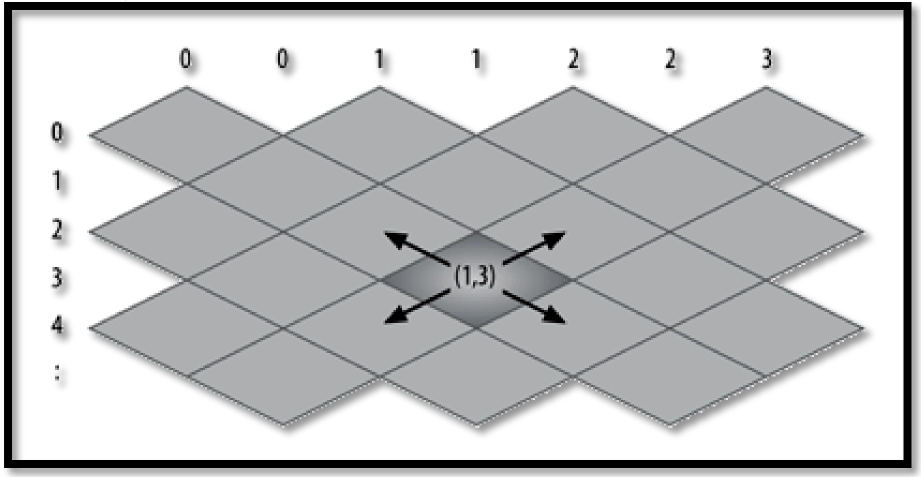

Tile Studio - Ch2
File
Edit
View
Tile
Animation
Sequence
Map
Palette
Code
Help
Level1
00
01
09
0B
10
10
0C
12
0B
10
12
04
0C
12
12
12
04
03
12
01
C0
02
C0
C0
C0
h v r
h v r
h v r
0
1
2
0
1
2
3
L1Tiles
Clouds
BACK
Size: 41 x 41
(21, 39)
Tile 15 of 32

Isometric Tiles



Isometric Tile Games

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



Demo

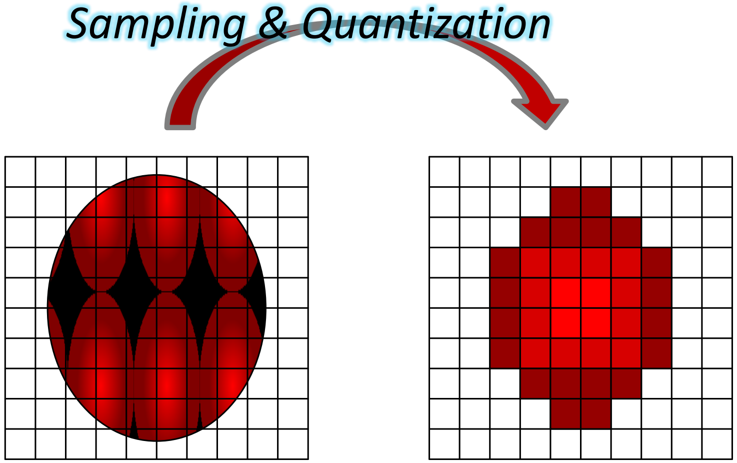
- Diablo

Agenda

- Game Loop
- Sprites & 2.5D
- Images

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
- Adobe Flash

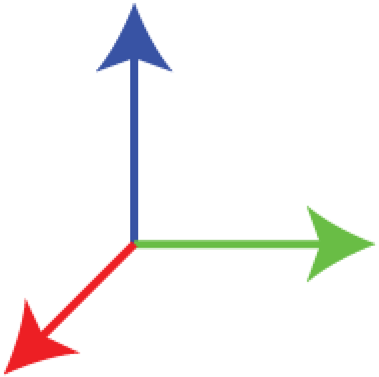
Raster Images

- 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 ~> 1.5 MB
    - Note that 32bit for color are more common -> ???
    - HDMI: 8bit (v1.3 – 10, 12 & 16 bit)

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

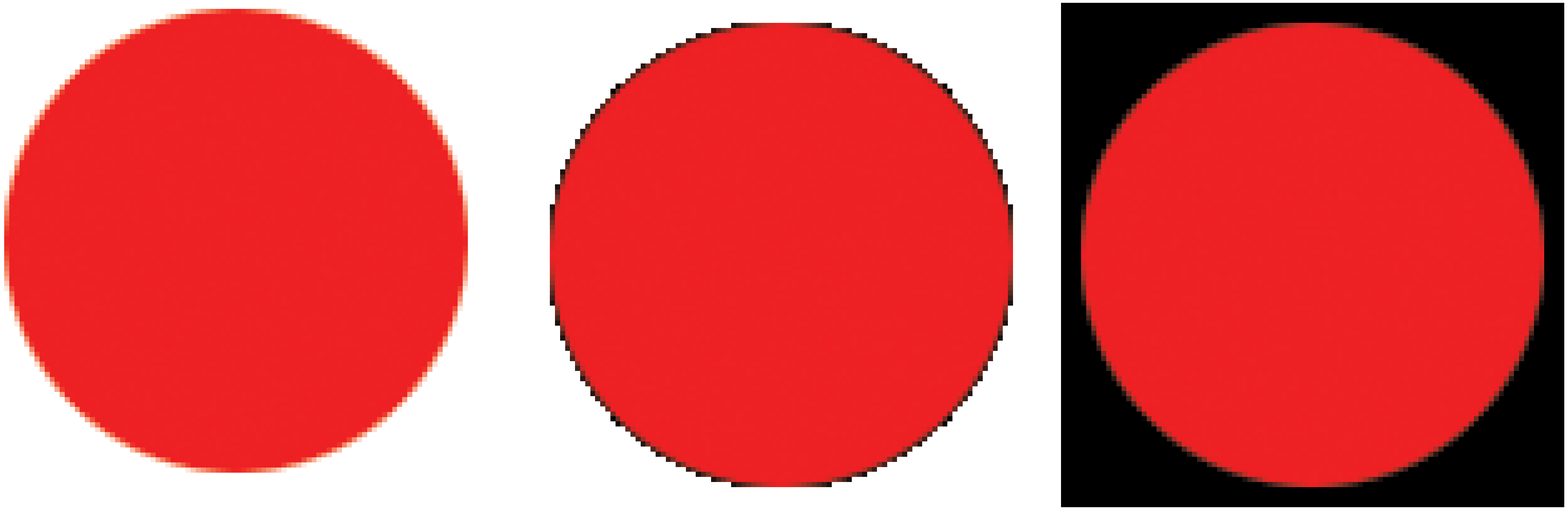


Image Files: Raw Data

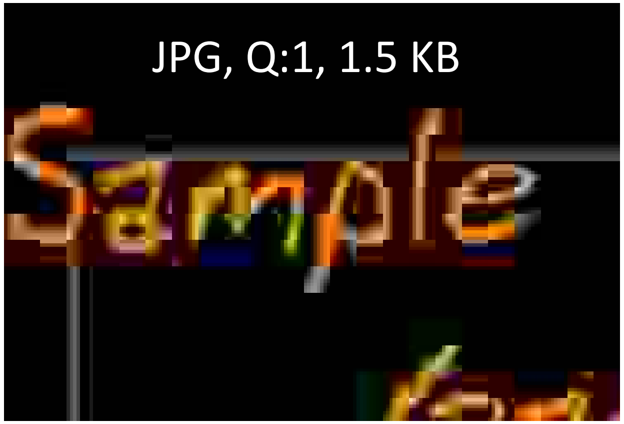
- Uncompressed image data
  - PPM, RAW, BMP
  - Benefits:
    - No (de)compression overhead
    - No (de)compression routine needed
      - Patents, additional code, licenses, etc.
  - Drawbacks:
    - File size: w*h*$\log_2$(#colors)

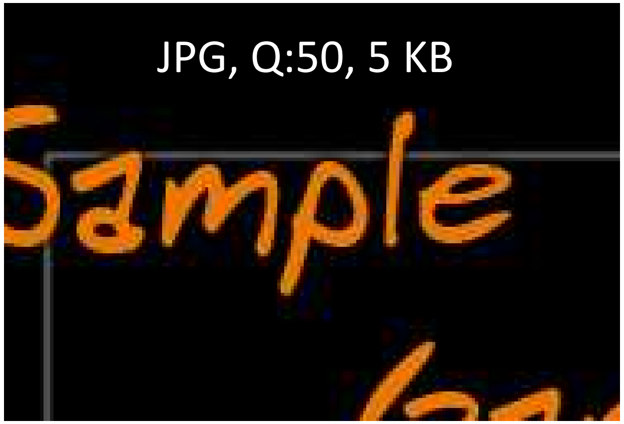
Image Files: Compressed

- 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
- Example: Note anti-aliasing and outer glow


JPG, Q:1, 1.5 KB


JPG, Q:50, 5 KB

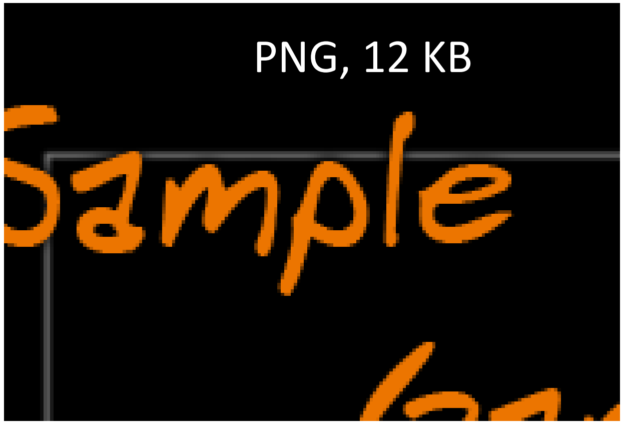
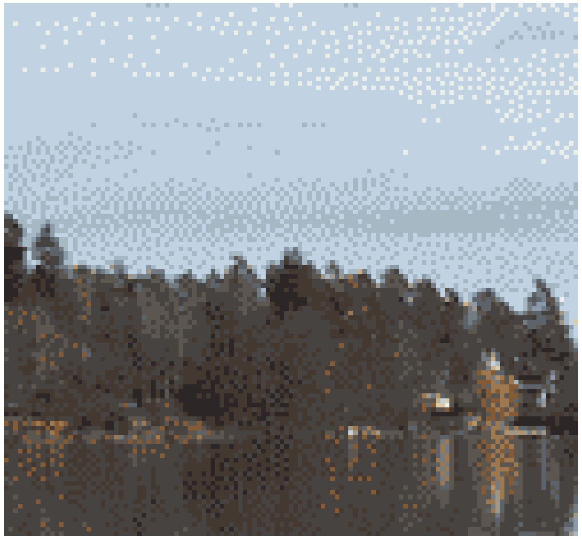
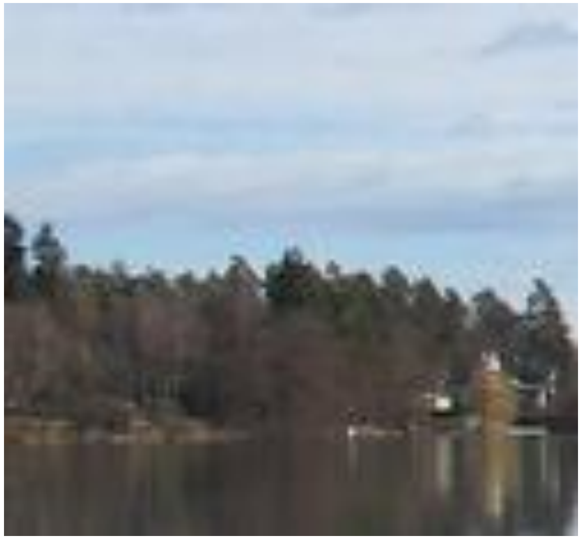

PNG, 12 KB

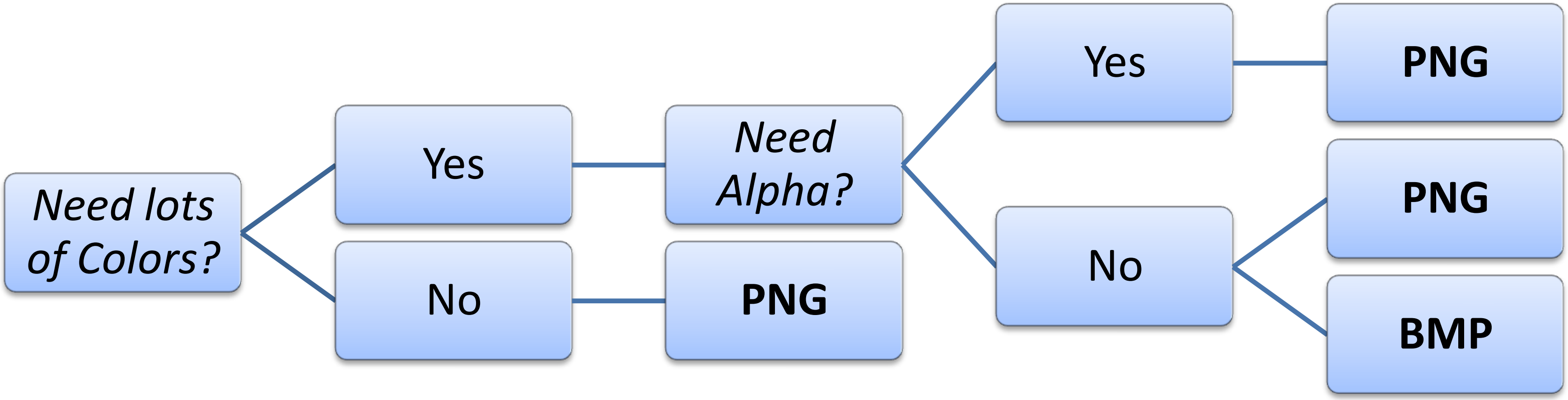
Image Files: Compressed

- 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?
  - – If we just need RGB, 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

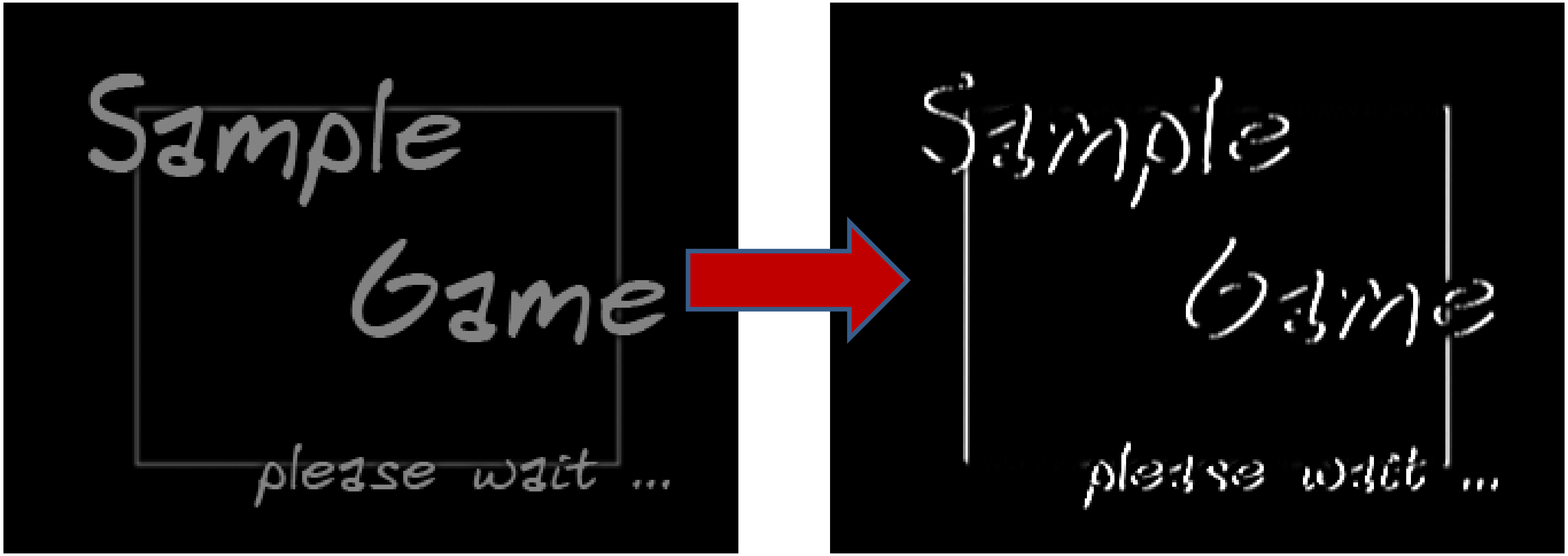
3	4	0	3	3
6	3	0	7	6
2	7	**2**	2	2
4	6	3	3	4
4	6	5	5	4

□

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

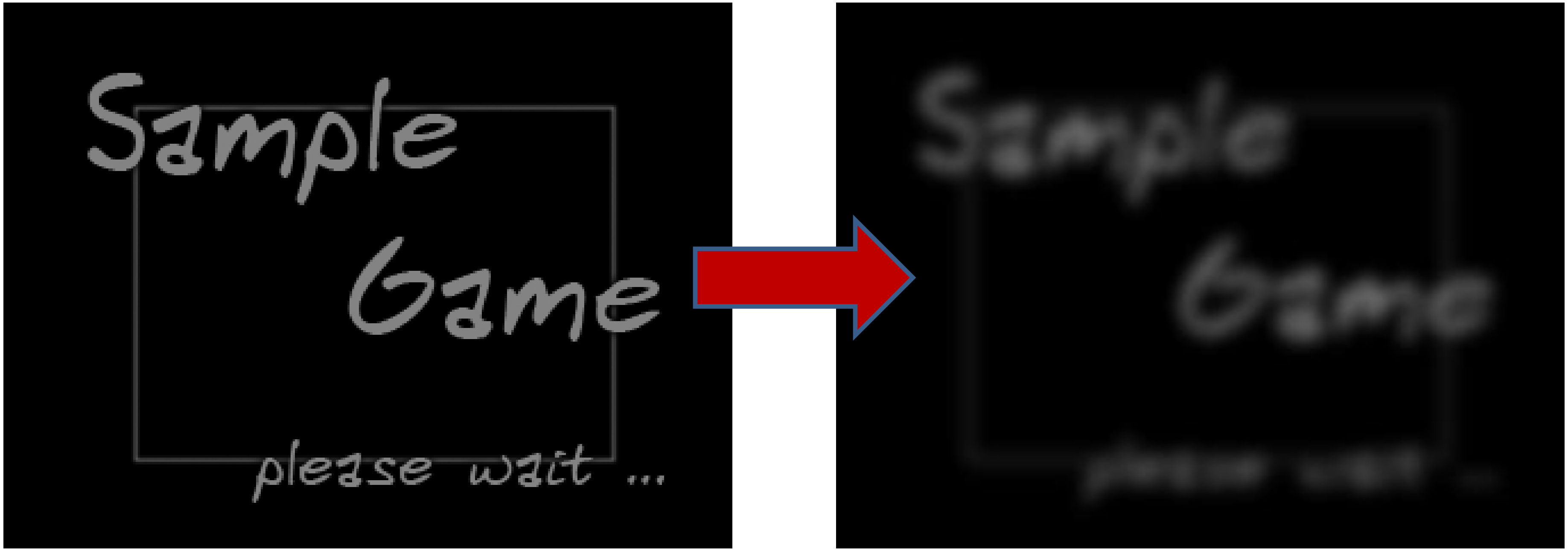
ConvolveOp

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



ConvolveOp

- Or blur images …



Gaussian Blur Filter

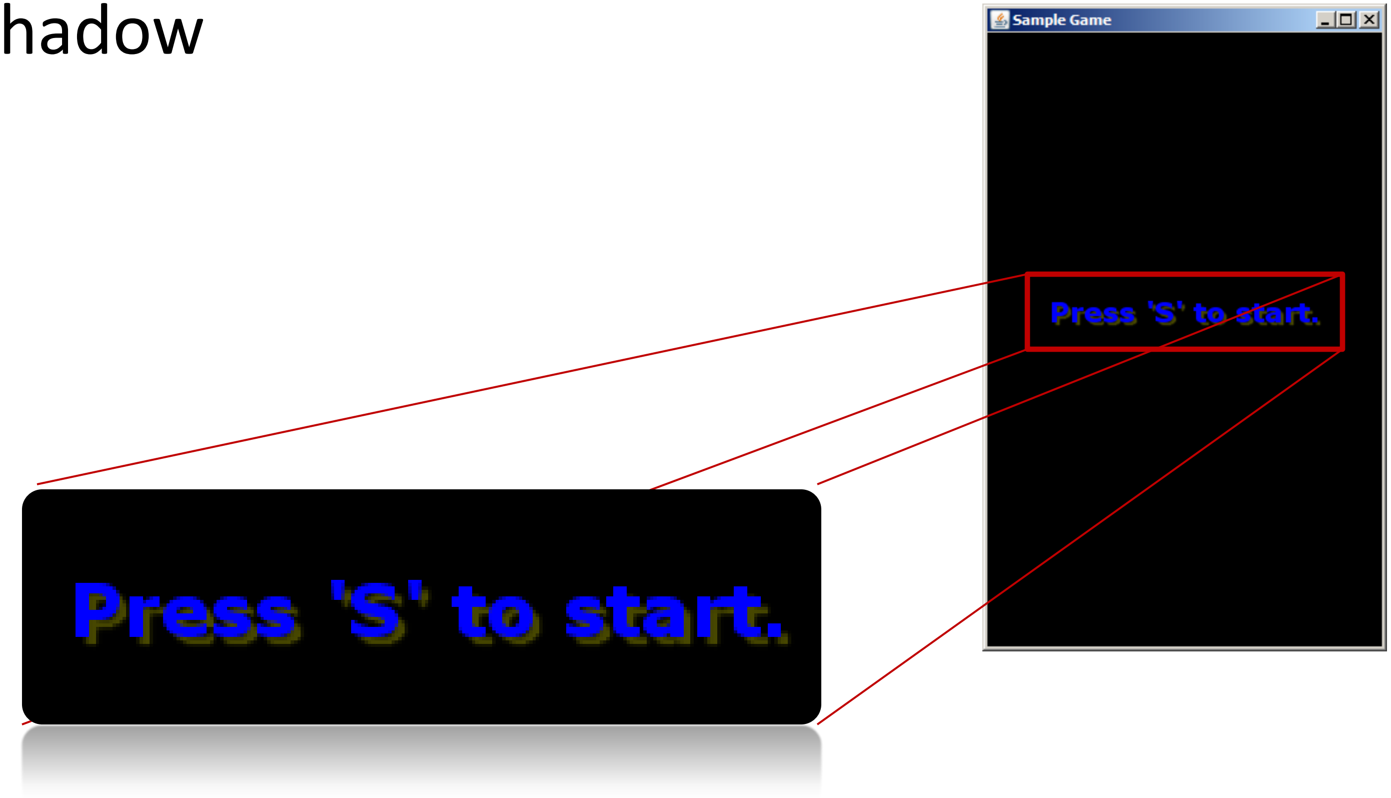
$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

For instance with □=1

$$\frac{1}{273}\begin{array}{|c|c|c|c|c|}\hline 1 & 4 & 7 & 4 & 1 \\ \hline 4 & 16 & 26 & 16 & 4 \\ \hline 7 & 26 & 41 & 26 & 7 \\ \hline 4 & 16 & 26 & 16 & 4 \\ \hline 1 & 4 & 7 & 4 & 1 \\ \hline \end{array}$$

Using Spatial Filtering: Walkthrough …

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



Press 'S' to start.

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));
}
```

Vielen Dank …

… für die Aufmerksamkeit