Mensch-Maschine-Interaktion 2
Übung 4

Ludwig-Maximilians-Universität München

Wintersemester 2012/2013

Alexander De Luca, Aurélien Tabard

Benefits and Drawbacks of Eyetrackers

Benefits

1. Which parts of the interface/ website attract attention
2. Flexible tool
3. Quantitative usage data
4. More reliable than mouse movement
5. Enables identifying interaction that the user is not aware of

Drawbacks

1. Expensive hardware => Mostly lab studies
2. Hardware might distract users (not easy to hide)
3. Peripheral attention not measured
4. No qualitative component (what did the user think when looking at point X)
5. Sample rate might be low

Possible Extension for an Eyetracker Experiment

- Think Aloud/Talk Aloud
- Questionnaires
- Interviews
- … and other qualitative measurements

Today

Overview of Multitouch Principles

Outline

- Hardware
- Processing Touch / Tracking
- Hardware Abstraction
- Exercise

Touch hardware available at the lab

- Smart screen
- Smartphones
- Kinects
- Microsoft Surface (1.0 and 2.0)
- Curve

Smart board

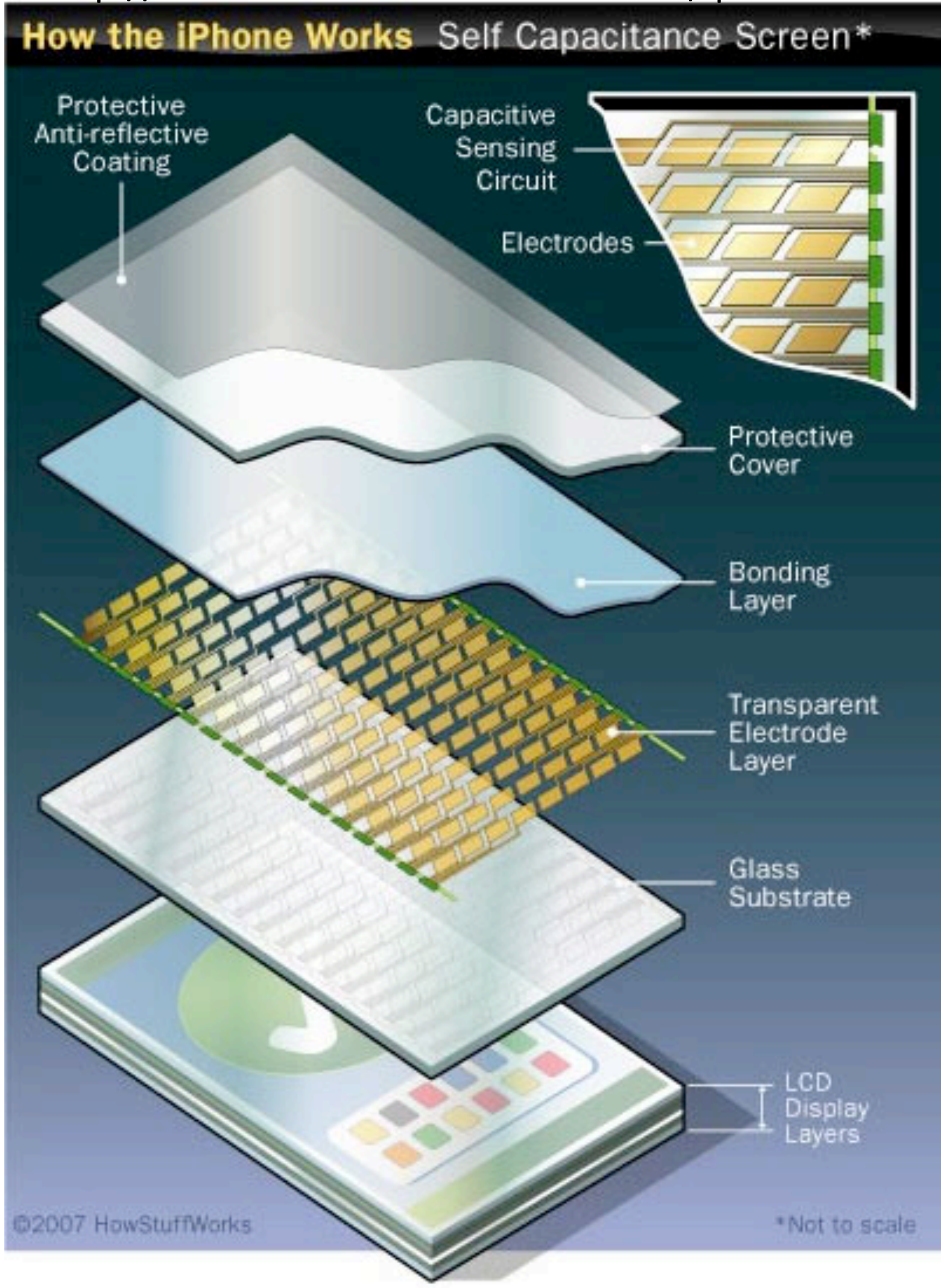
- Projection based
- Vision based tracking
- Single touch and pen tracking
- Cameras placed in the corners.

Smart Board 800 communication material

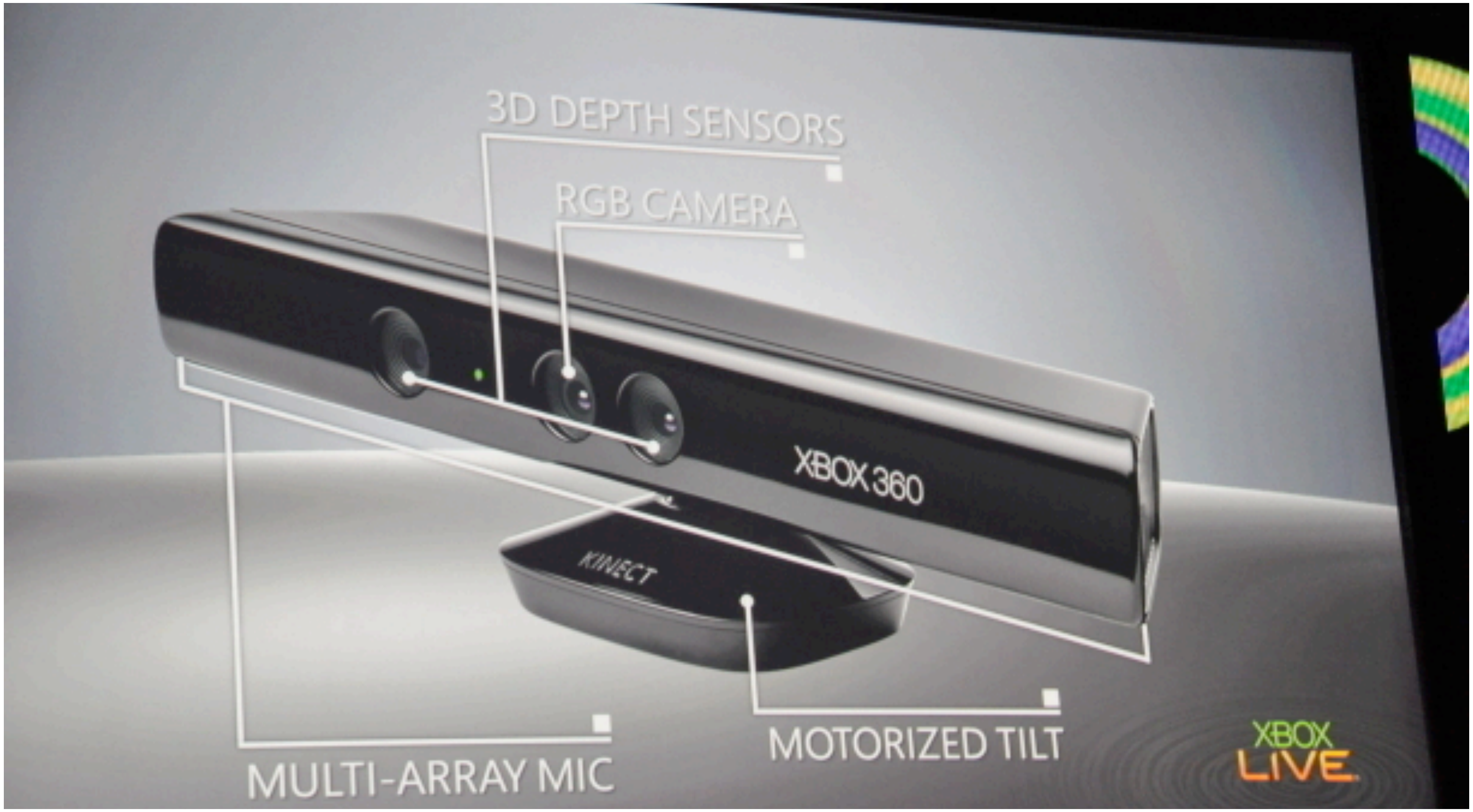
http://electronics.howstuffworks.com/iphone2.htm

Smartphones

- Capacitive screens
- Multitouch
- LCD screen
- Small size

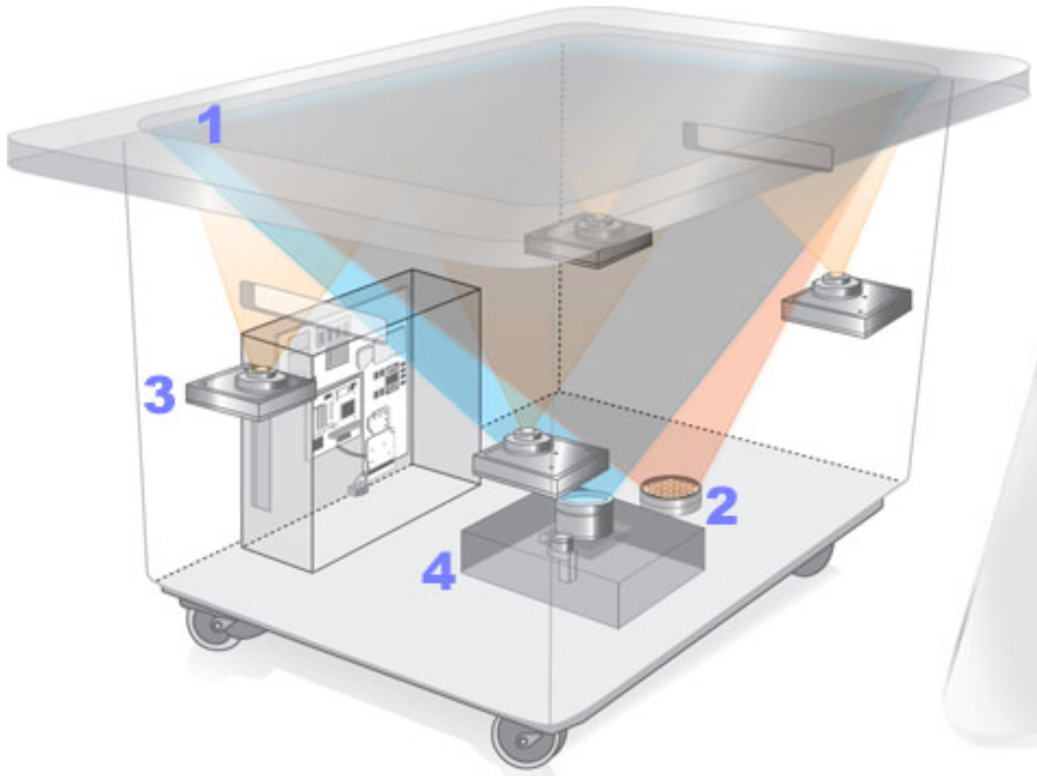

How the iPhone Works Self Capacitance Screen*
Protective Anti-reflective Coating
Capacitive Sensing Circuit
Electrodes
Protective Cover
Bonding Layer
Transparent Electrode Layer
Glass Substrate
LCD Display Layers
©2007 HowStuffWorks
*Not to scale

Kinects



http://itp.nyu.edu/physcomp/sensors/Reports/MicrosoftKinect

Microsoft Surface 1.0

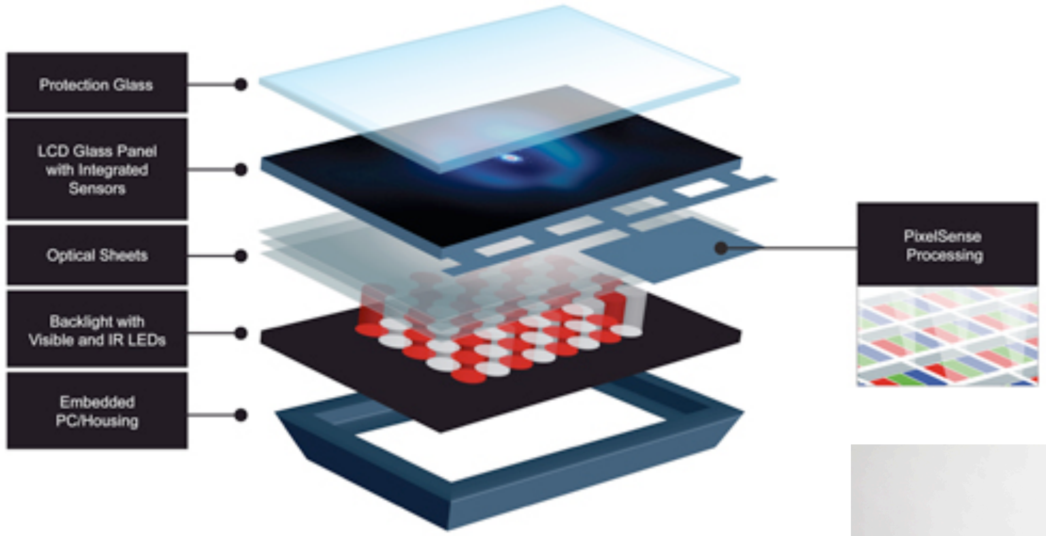


1. Projected surface
2. Infrared projector
3. Cameras
4. Projector

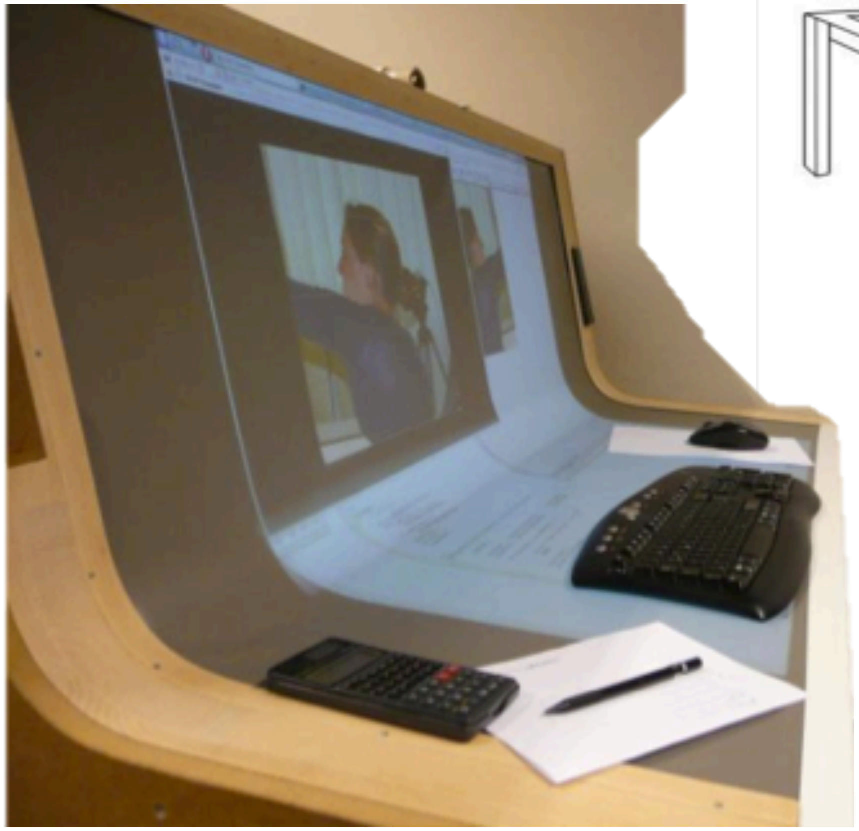
http://www.knowledgebase-script.com/demo/article-420.html

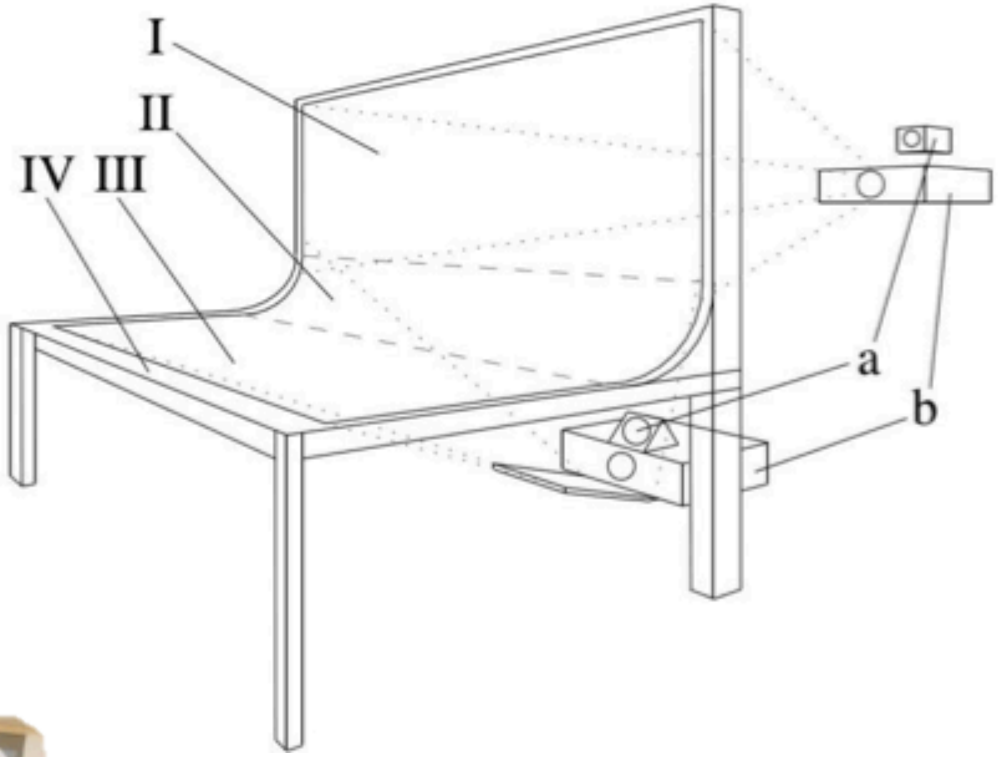
Microsoft Pixelsense (Formerly Surface 2.0)

http://www.microsoft.com/en-us/pixelsense/pixelsense.aspx


Protection Glass
LCD Glass Panel with Integrated Sensors
Optical Sheets
Backlight with Visible and IR LEDs
Embedded PC/Housing
PixelSense Processing

Curve

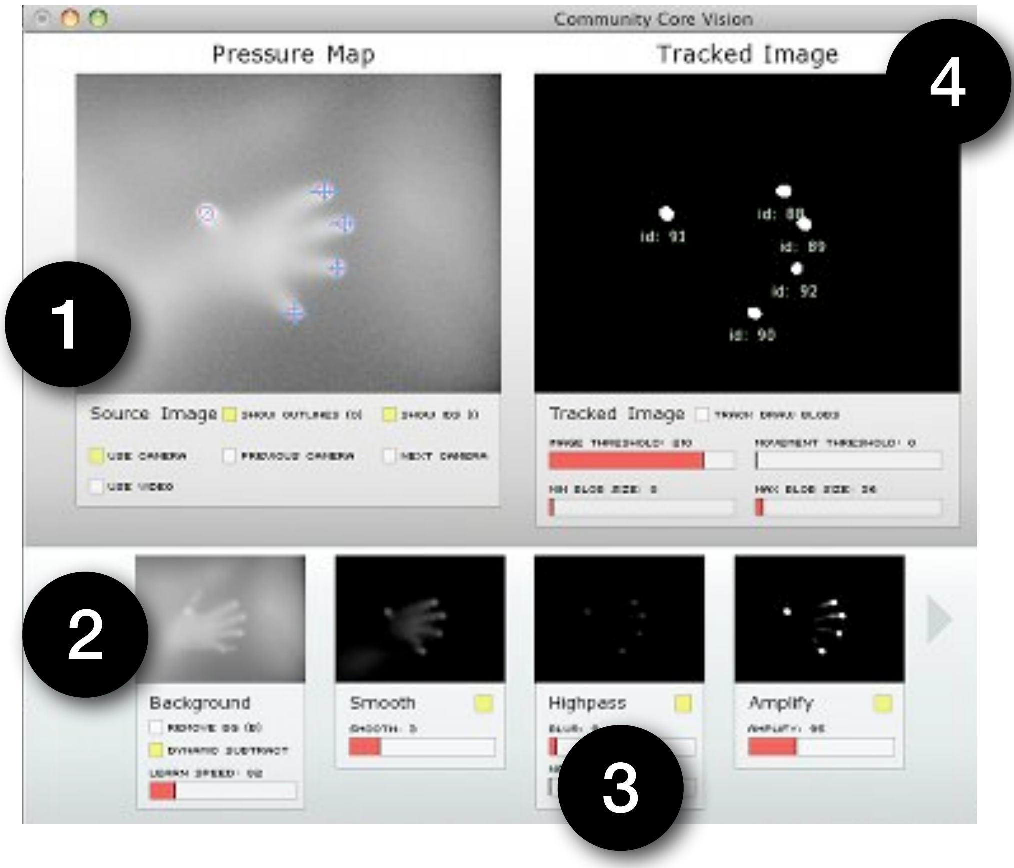




Processing Touch

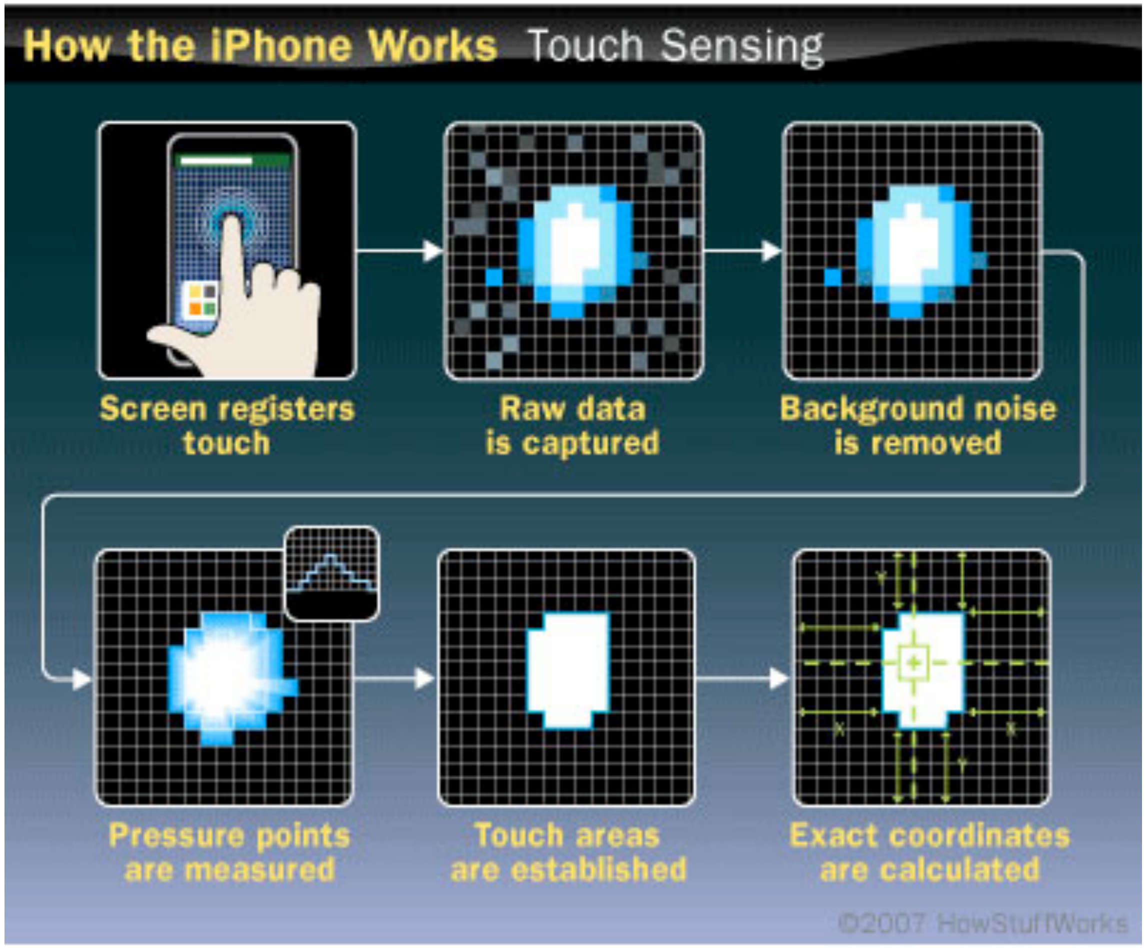
Vision based

1. Capture
2. Background Subtraction
3. Processing / Filtering
4. Recognition
5. Tracking



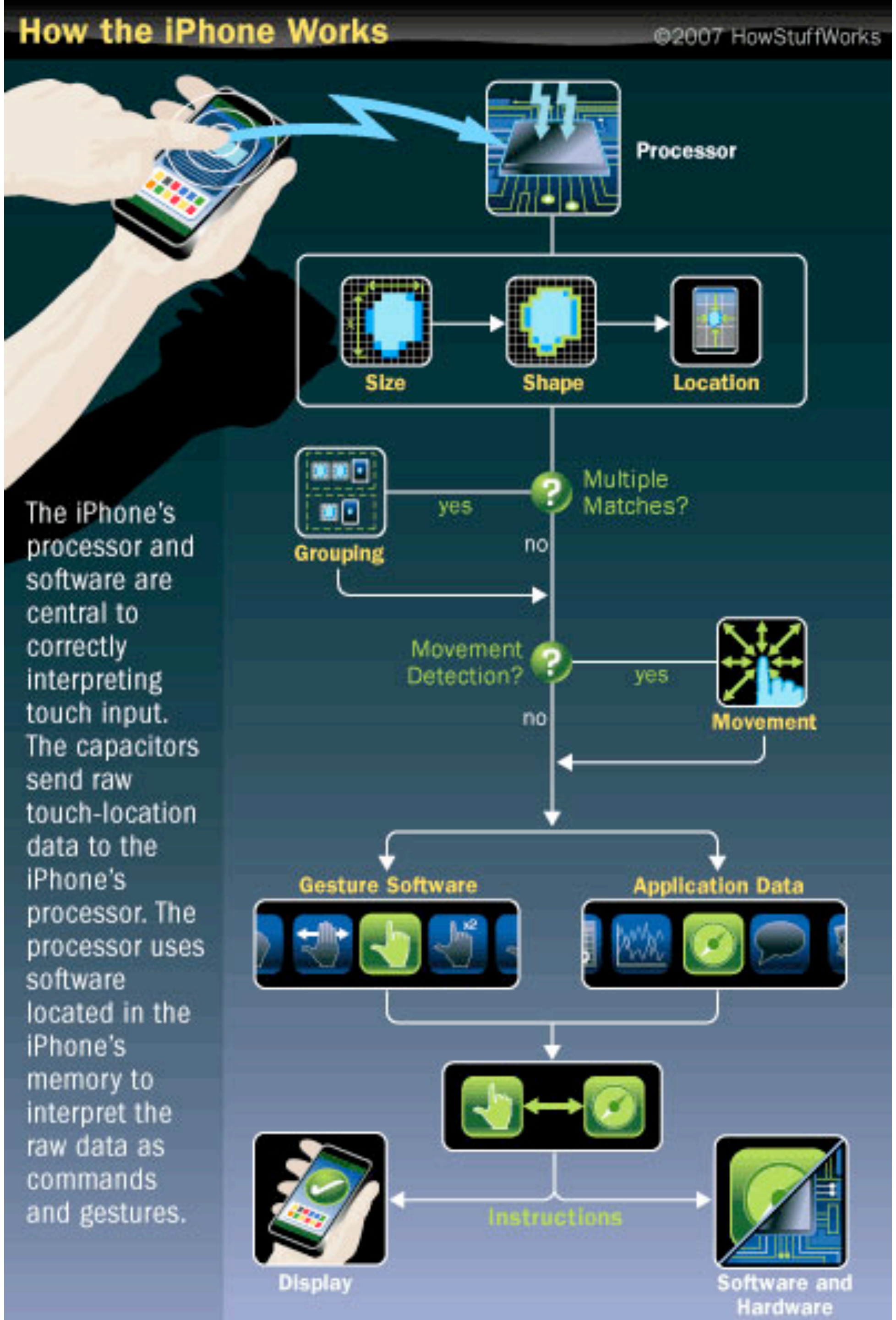
http://ccv.nuigroup.com/

Capacitive screens (1)


How the iPhone Works Touch Sensing
Screen registers touch
Raw data is captured
Background noise is removed
Pressure points are measured
Touch areas are established
Exact coordinates are calculated
©2007 HowStuffWorks

http://electronics.howstuffworks.com/iphone3.htm

Capacitive screens (2)

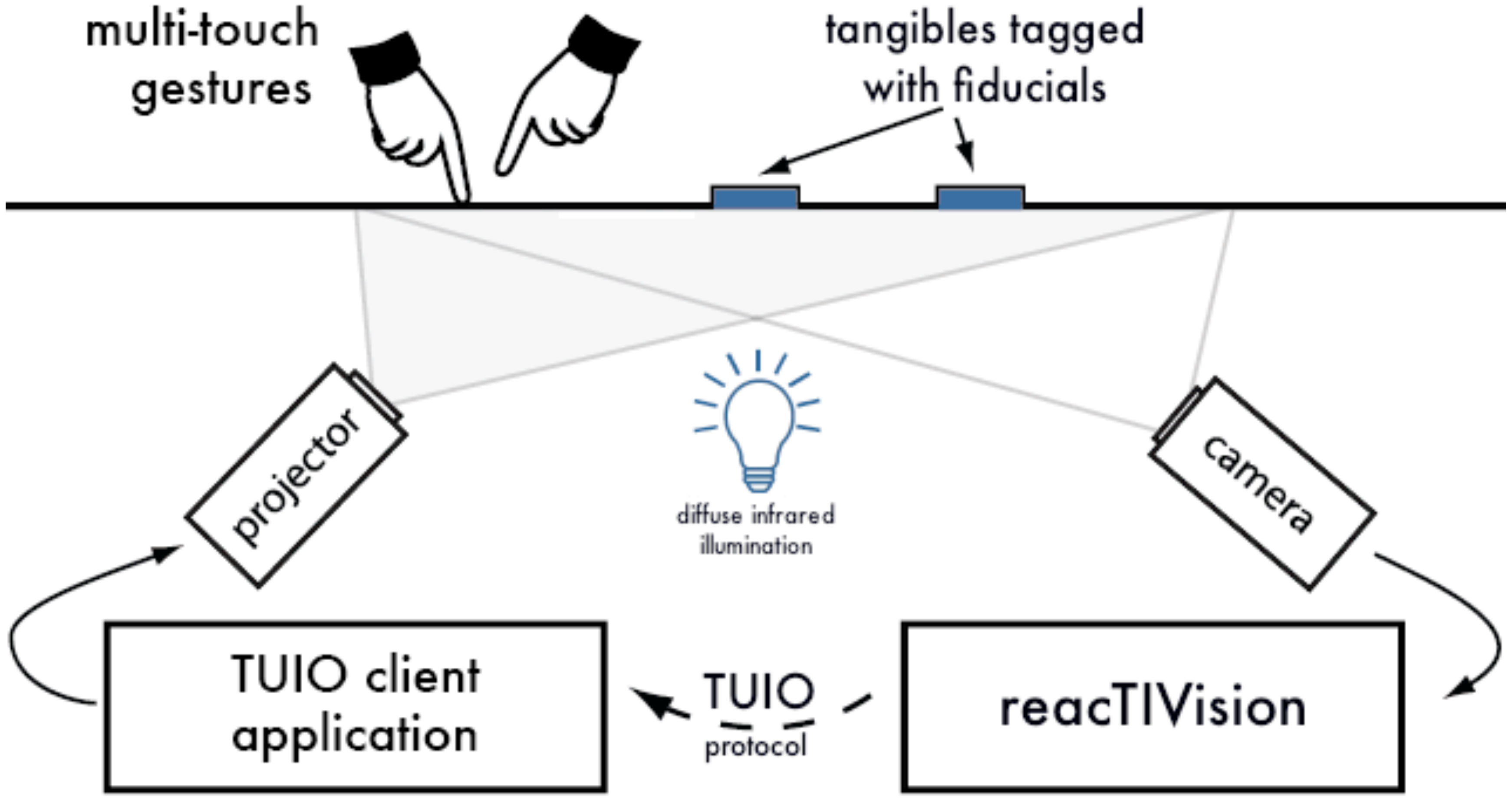


The iPhone's processor and software are central to correctly interpreting touch input. The capacitors send raw touch-location data to the iPhone's processor. The processor uses software located in the iPhone's memory to interpret the raw data as commands and gestures.

http://electronics.howstuffworks.com/iphone3.htm

Hardware Abstraction: TUIO protocol

Hardware abstraction for input



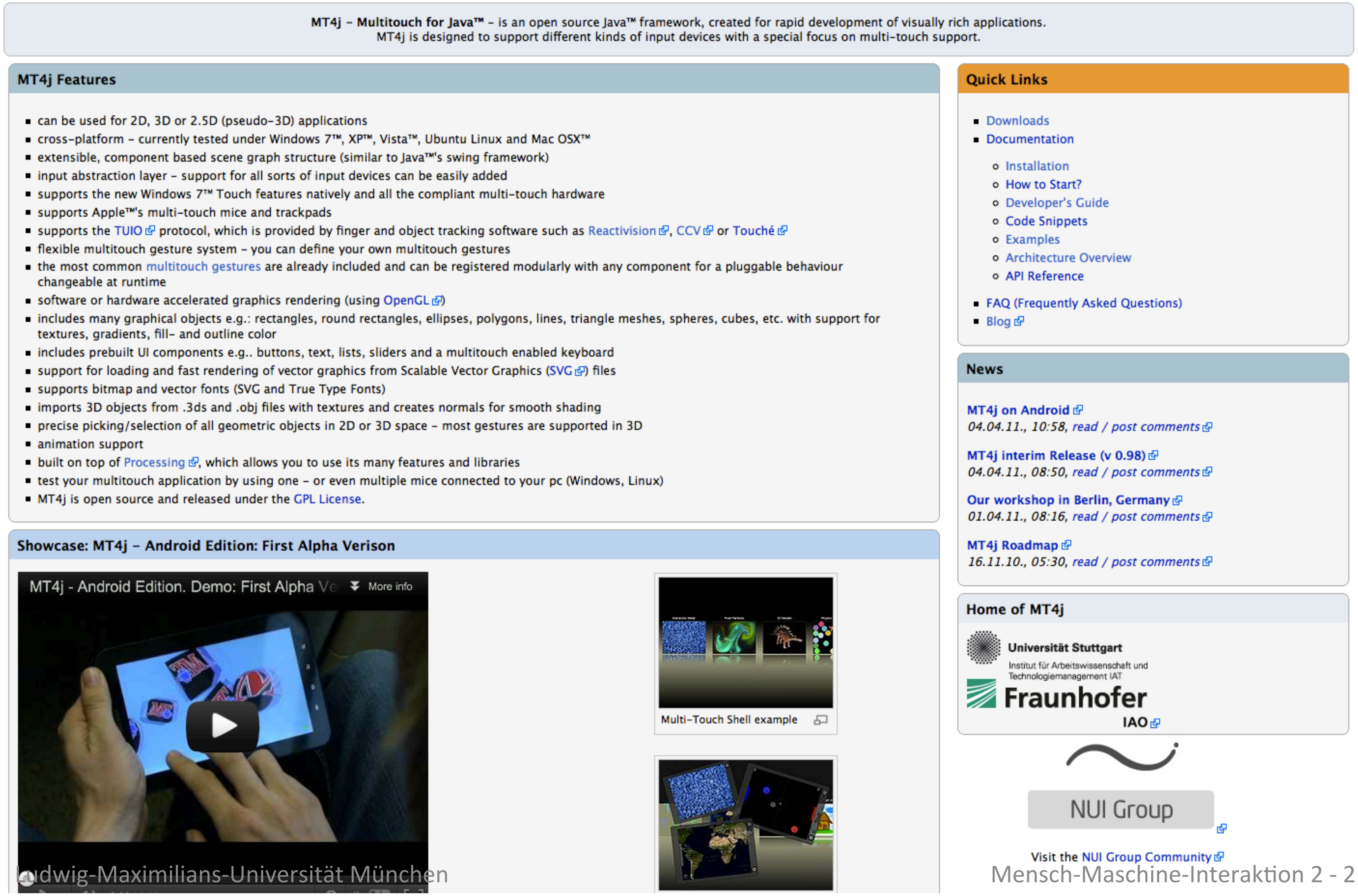
References

1. B. Buxton, Multi-Touch Systems that I Have Known and Loved. http://www.billbuxton.com/multitouchOverview.html
2. How Stuff Works: The iPhone http://electronics.howstuffworks.com/iphone.htm

Exercise 4:

A multitouch photo browser

MT4J: Multi-Touch for Java



MT4j – Multitouch for Java™ – is an open source Java™ framework, created for rapid development of visually rich applications.
MT4j is designed to support different kinds of input devices with a special focus on multi-touch support.

MT4j Features

- can be used for 2D, 3D or 2.5D (pseudo-3D) applications
- cross-platform – currently tested under Windows 7™, XP™, Vista™, Ubuntu Linux and Mac OSX™
- extensible, component based scene graph structure (similar to Java™'s swing framework)
- input abstraction layer – support for all sorts of input devices can be easily added
- supports the new Windows 7™ Touch features natively and all the compliant multi-touch hardware
- supports Apple™'s multi-touch mice and trackpads
- supports the TUIO protocol, which is provided by finger and object tracking software such as Reactivision, CCV or Touché
- flexible multitouch gesture system – you can define your own multitouch gestures
- the most common multitouch gestures are already included and can be registered modularly with any component for a pluggable behaviour changeable at runtime
- software or hardware accelerated graphics rendering (using OpenGL)
- includes many graphical objects e.g.: rectangles, round rectangles, ellipses, polygons, lines, triangle meshes, spheres, cubes, etc. with support for textures, gradients, fill- and outline color
- includes prebuilt UI components e.g.. buttons, text, lists, sliders and a multitouch enabled keyboard
- support for loading and fast rendering of vector graphics from Scalable Vector Graphics (SVG) files
- supports bitmap and vector fonts (SVG and True Type Fonts)
- imports 3D objects from .3ds and .obj files with textures and creates normals for smooth shading
- precise picking/selection of all geometric objects in 2D or 3D space – most gestures are supported in 3D
- animation support
- built on top of Processing, which allows you to use its many features and libraries
- test your multitouch application by using one – or even multiple mice connected to your pc (Windows, Linux)
- MT4j is open source and released under the GPL License.

Showcase: MT4j – Android Edition: First Alpha Verison

MT4j - Android Edition. Demo: First Alpha Ve More info

Multi-Touch Shell example

Quick Links

- Downloads
- Documentation
  - Installation
  - How to Start?
  - Developer's Guide
  - Code Snippets
  - Examples
  - Architecture Overview
  - API Reference
- FAQ (Frequently Asked Questions)
- Blog

News

MT4j on Android
04.04.11., 10:58, read / post comments

MT4j interim Release (v 0.98)
04.04.11., 08:50, read / post comments

Our workshop in Berlin, Germany
01.04.11., 08:16, read / post comments

MT4j Roadmap
16.11.10., 05:30, read / post comments

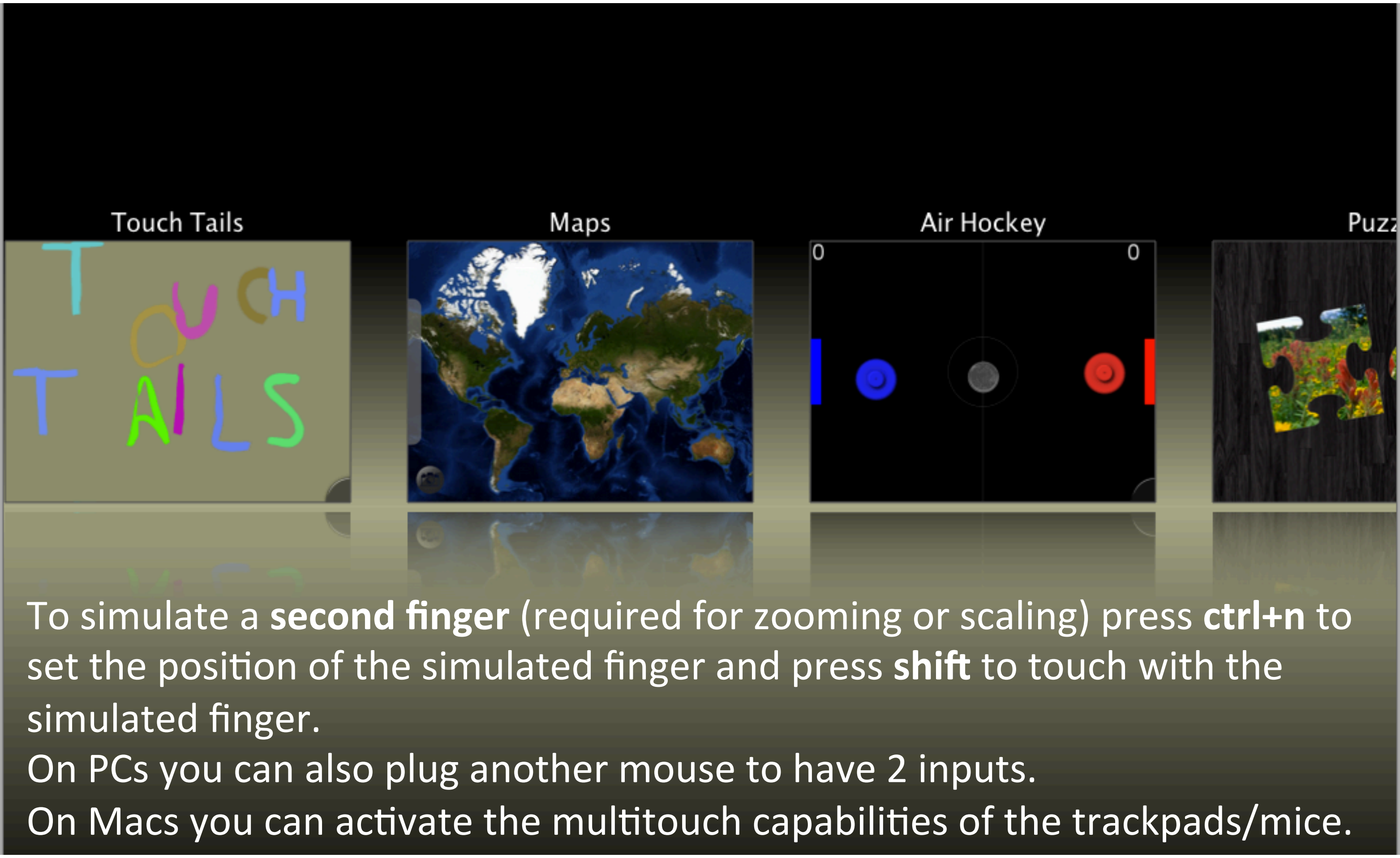
Home of MT4j

Universität Stuttgart
Institut für Arbeitswissenschaft und Technologiemanagement IAT
Fraunhofer IAO

NUI Group

Visit the NUI Group Community

Demo, mtShell



To simulate a **second finger** (required for zooming or scaling) press **ctrl+n** to set the position of the simulated finger and press **shift** to touch with the simulated finger.
On PCs you can also plug another mouse to have 2 inputs.
On Macs you can activate the multitouch capabilities of the trackpads/mice.

MT4J Principles

- Full screen (= single window) applications
- Application composed of scenes

```
public class StartHelloWorld extends MTApplication {
      private static final long serialVersionUID = 1L;

      public static void main(String[] args) {
            initialize();
      }
      @Override
      public void startUp() {
            addScene(newHelloWorldScene(this,"HelloWorld"));
      }
}
```

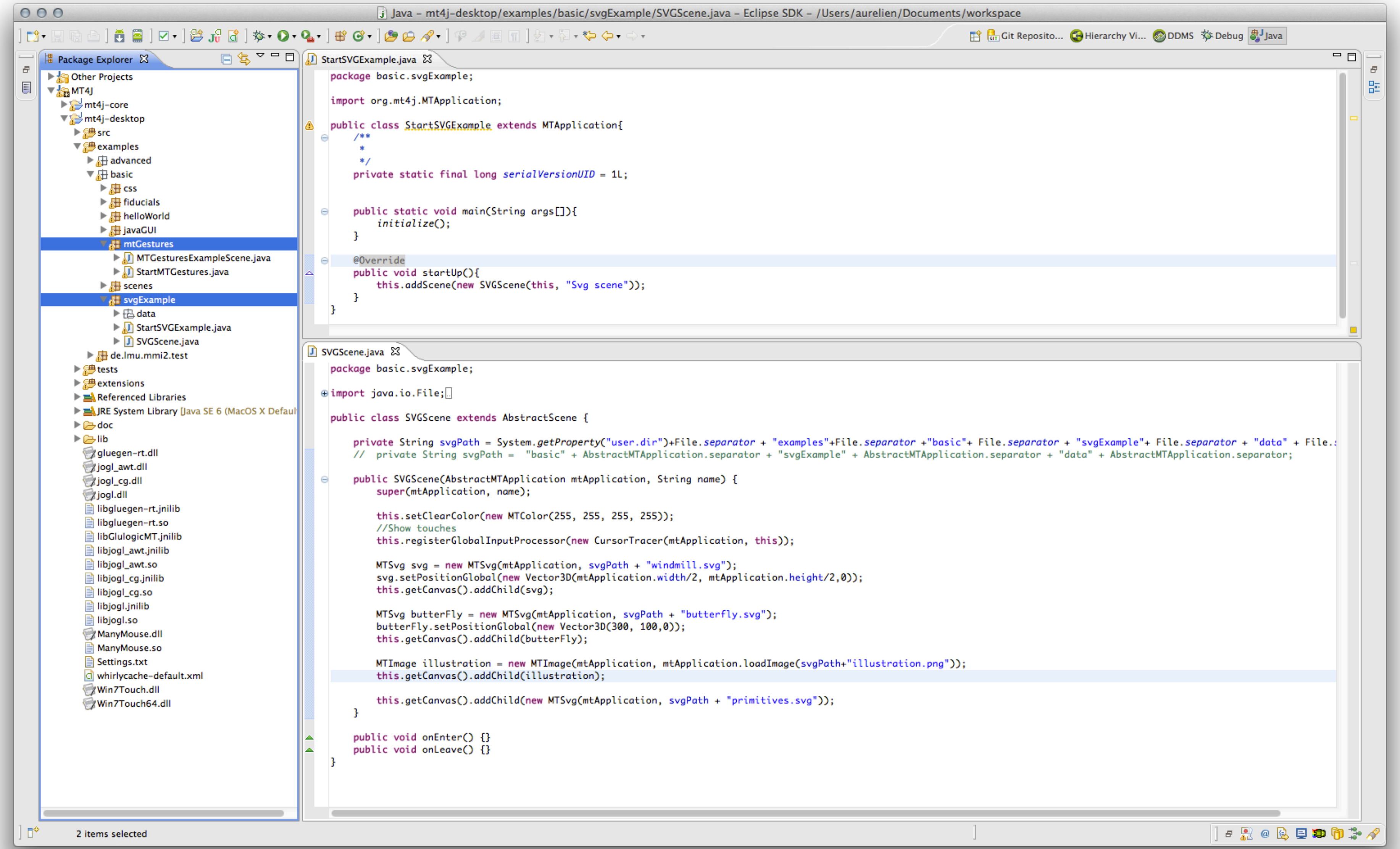
Hello World Scene

```
public class HelloWorldScene extends AbstractScene {

   public HelloWorldScene( AbstractMTApplication mtApplication,
                           String name) {
     super(mtApplication, name);

     IFont fontArial= FontManager.getInstance().createFont(
       mtApplication, "arial.ttf", 50, new MTColor(255,255,255));

     //Create a textfield
     MTTextArea textField = new MTTextArea(  mtApplication,
                                             fontArial);
     textField.setNoStroke(true);
     textField.setNoFill(true);
     textField.setText("Hello World!");
```

Hello World Scene

```
    //Center the textfield on the screen
    textField.setPositionGlobal(new Vector3D(
           mtApplication.width/2f, mtApplication.height/2f));
    //Add the textfield to our canvas
    this.getCanvas().addChild(textField);

    //Show touches
    this.registerGlobalInputProcessor(
             new CursorTracer(mtApplication, this));
  }

  public void onEnter() {}
  public void onLeave() {}
}
```

Eclipse walkthrough



Java – mt4j-desktop/examples/basic/svgExample/SVGScene.java – Eclipse SDK – /Users/aurelien/Documents/workspace

```
Other Projects
MT4J
  mt4j-core
  mt4j-desktop
    src
    examples
      advanced
      basic
        css
        fiducials
        helloWorld
        javaGUI
        mtGestures
          MTGesturesExampleScene.java
          StartMTGestures.java
        scenes
        svgExample
          data
          StartSVGExample.java
          SVGScene.java
      de.lmu.mmi2.test
    tests
    extensions
    Referenced Libraries
    JRE System Library [Java SE 6 (MacOS X Defaul
    doc
    lib
    gluegen-rt.dll
    jogl_awt.dll
    jogl_cg.dll
    jogl.dll
    libgluegen-rt.jnilib
    libgluegen-rt.so
    libGlulogicMT.jnilib
    libjogl_awt.jnilib
    libjogl_awt.so
    libjogl_cg.jnilib
    libjogl_cg.so
    libjogl.jnilib
    libjogl.so
    ManyMouse.dll
    ManyMouse.so
    Settings.txt
    whirlycache-default.xml
    Win7Touch.dll
    Win7Touch64.dll
```

StartSVGExample.java

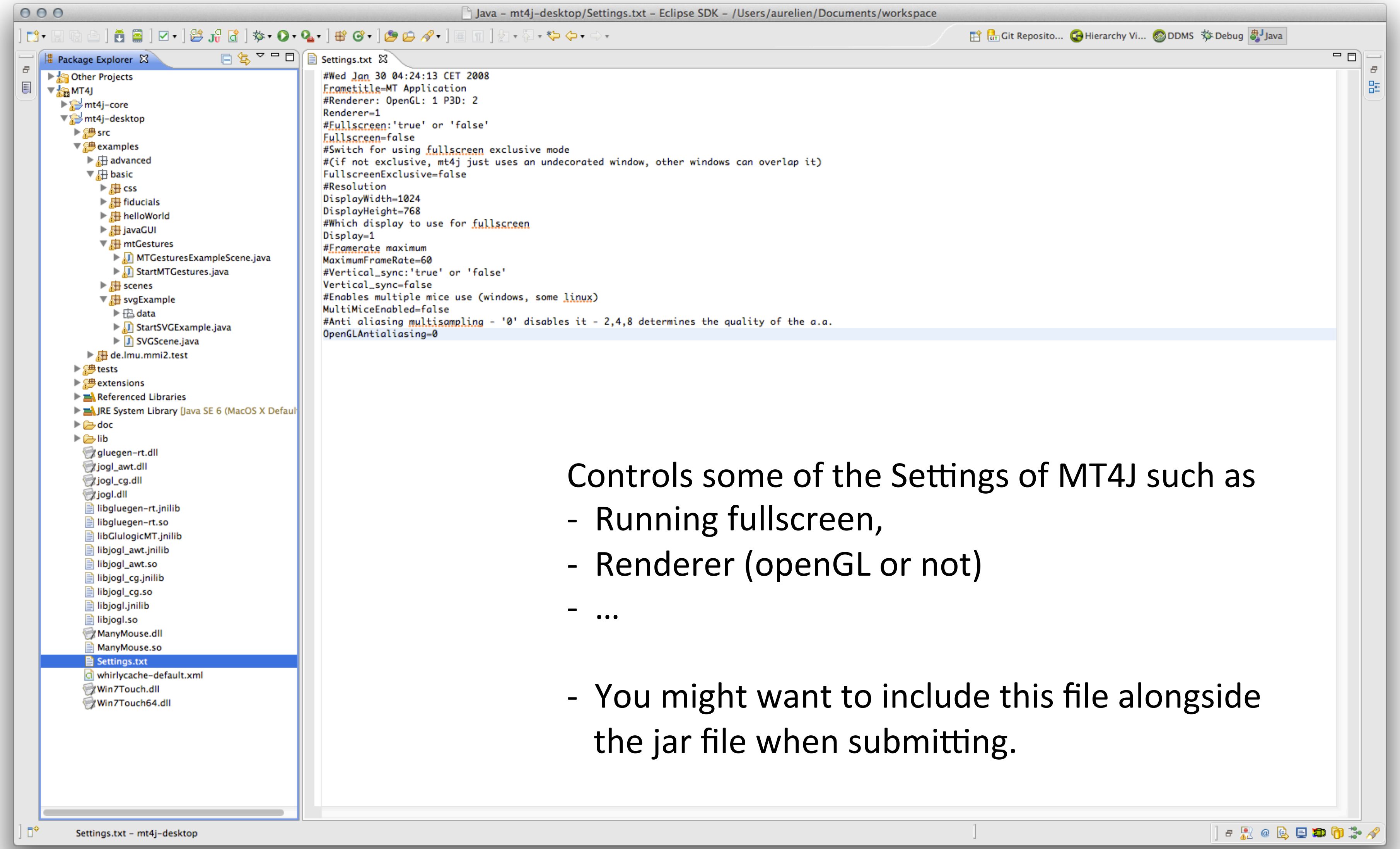
```java
package basic.svgExample;

import org.mt4j.MTApplication;

public class StartSVGExample extends MTApplication{
	/**
	 *
	 */
	private static final long serialVersionUID = 1L;


	public static void main(String args[]){
		initialize();
	}

	@Override
	public void startUp(){
		this.addScene(new SVGScene(this, "Svg scene"));
	}
}
```

SVGScene.java

```java
package basic.svgExample;

import java.io.File;

public class SVGScene extends AbstractScene {

	private String svgPath = System.getProperty("user.dir")+File.separator + "examples"+File.separator +"basic"+ File.separator + "svgExample"+ File.separator + "data" + File.
//	private String svgPath =  "basic" + AbstractMTApplication.separator + "svgExample" + AbstractMTApplication.separator + "data" + AbstractMTApplication.separator;

	public SVGScene(AbstractMTApplication mtApplication, String name) {
		super(mtApplication, name);

		this.setClearColor(new MTColor(255, 255, 255, 255));
		//Show touches
		this.registerGlobalInputProcessor(new CursorTracer(mtApplication, this));

		MTSvg svg = new MTSvg(mtApplication, svgPath + "windmill.svg");
		svg.setPositionGlobal(new Vector3D(mtApplication.width/2, mtApplication.height/2,0));
		this.getCanvas().addChild(svg);

		MTSvg butterFly = new MTSvg(mtApplication, svgPath + "butterfly.svg");
		butterFly.setPositionGlobal(new Vector3D(300, 100,0));
		this.getCanvas().addChild(butterFly);

		MTImage illustration = new MTImage(mtApplication, mtApplication.loadImage(svgPath+"illustration.png"));
		this.getCanvas().addChild(illustration);

		this.getCanvas().addChild(new MTSvg(mtApplication, svgPath + "primitives.svg"));
	}

	public void onEnter() {}
	public void onLeave() {}
}
```

2 items selected

Settings.txt

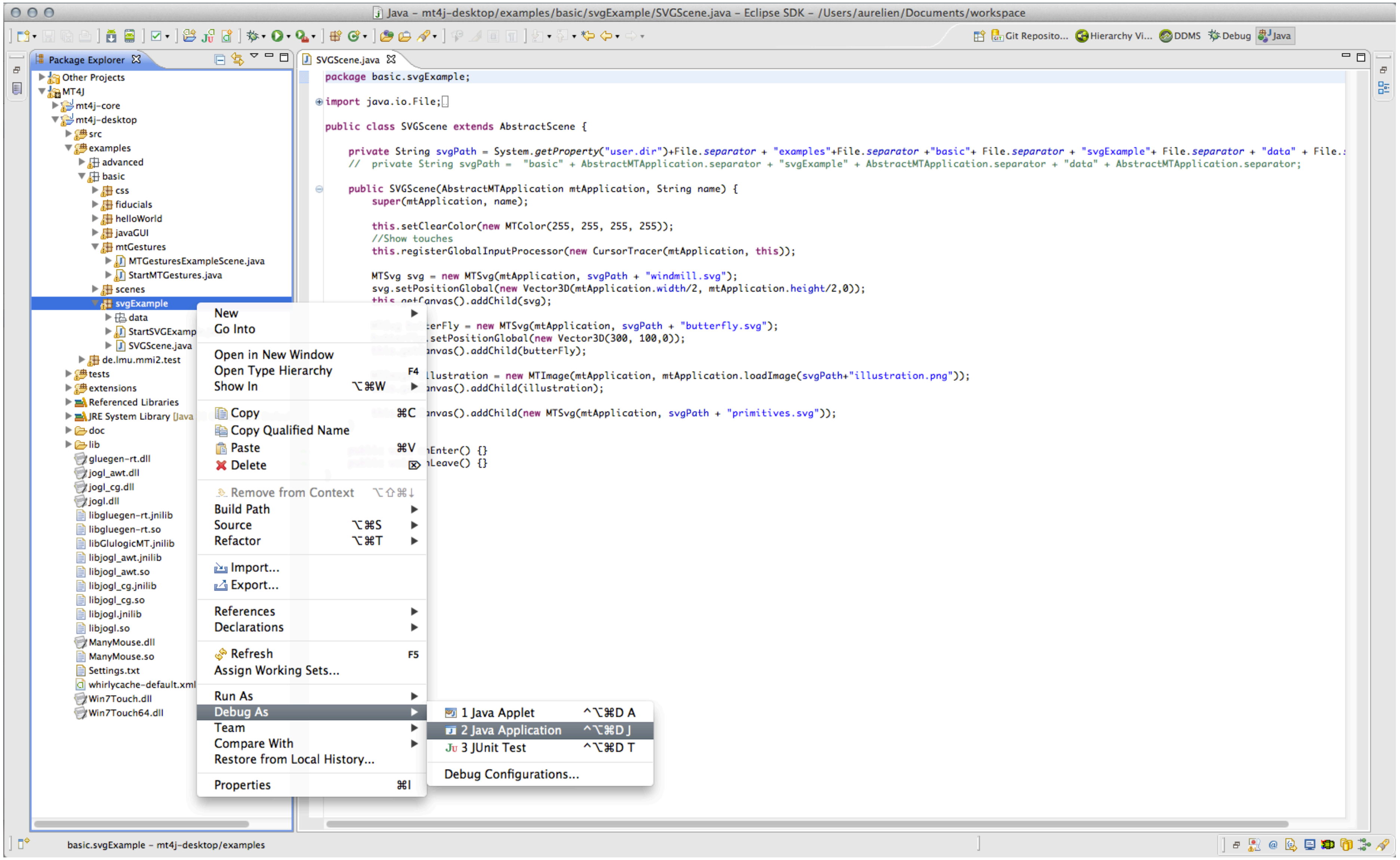


```
#Wed Jan 30 04:24:13 CET 2008
Frametitle=MT Application
#Renderer: OpenGL: 1 P3D: 2
Renderer=1
#Fullscreen:'true' or 'false'
Fullscreen=false
#Switch for using fullscreen exclusive mode
#(if not exclusive, mt4j just uses an undecorated window, other windows can overlap it)
FullscreenExclusive=false
#Resolution
DisplayWidth=1024
DisplayHeight=768
#Which display to use for fullscreen
Display=1
#Framerate maximum
MaximumFrameRate=60
#Vertical_sync:'true' or 'false'
Vertical_sync=false
#Enables multiple mice use (windows, some linux)
MultiMiceEnabled=false
#Anti aliasing multisampling - '0' disables it - 2,4,8 determines the quality of the a.a.
OpenGLAntialiasing=0
```

Controls some of the Settings of MT4J such as
- Running fullscreen,
- Renderer (openGL or not)
- …

- You might want to include this file alongside the jar file when submitting.

Debugging



```java
package basic.svgExample;

import java.io.File;

public class SVGScene extends AbstractScene {

    private String svgPath = System.getProperty("user.dir")+File.separator + "examples"+File.separator +"basic"+ File.separator + "svgExample"+ File.separator + "data" + File.
//  private String svgPath =  "basic" + AbstractMTApplication.separator + "svgExample" + AbstractMTApplication.separator + "data" + AbstractMTApplication.separator;

    public SVGScene(AbstractMTApplication mtApplication, String name) {
        super(mtApplication, name);

        this.setClearColor(new MTColor(255, 255, 255, 255));
        //Show touches
        this.registerGlobalInputProcessor(new CursorTracer(mtApplication, this));

        MTSvg svg = new MTSvg(mtApplication, svgPath + "windmill.svg");
        svg.setPositionGlobal(new Vector3D(mtApplication.width/2, mtApplication.height/2,0));
        this.getCanvas().addChild(svg);

                  erFly = new MTSvg(mtApplication, svgPath + "butterfly.svg");
                  .setPositionGlobal(new Vector3D(300, 100,0));
                  anvas().addChild(butterFly);

                  llustration = new MTImage(mtApplication, mtApplication.loadImage(svgPath+"illustration.png"));
                  anvas().addChild(illustration);

                  anvas().addChild(new MTSvg(mtApplication, svgPath + "primitives.svg"));


                  nEnter() {}
                  nLeave() {}
```