Sun SPOTs in Action—3D, Virtual Reality and Gaming

Simon Ritter
Technology Evangelist
Angela Caicedo
Technology Evangelist
Sun Microsystems

Session TS-1780

Immersive Java™ Platform Gaming

How to build your own Java technology-based Wii

Learn how to use a range of Java technologies to build compelling, interactive 3D virtual reality games

Agenda

Introduction to Project Sun SPOTs (SPOTs)

Interfacing Useful Hardware to SPOTs

Using Java 3D™ API for Virtual Reality

SPOT Interaction Software

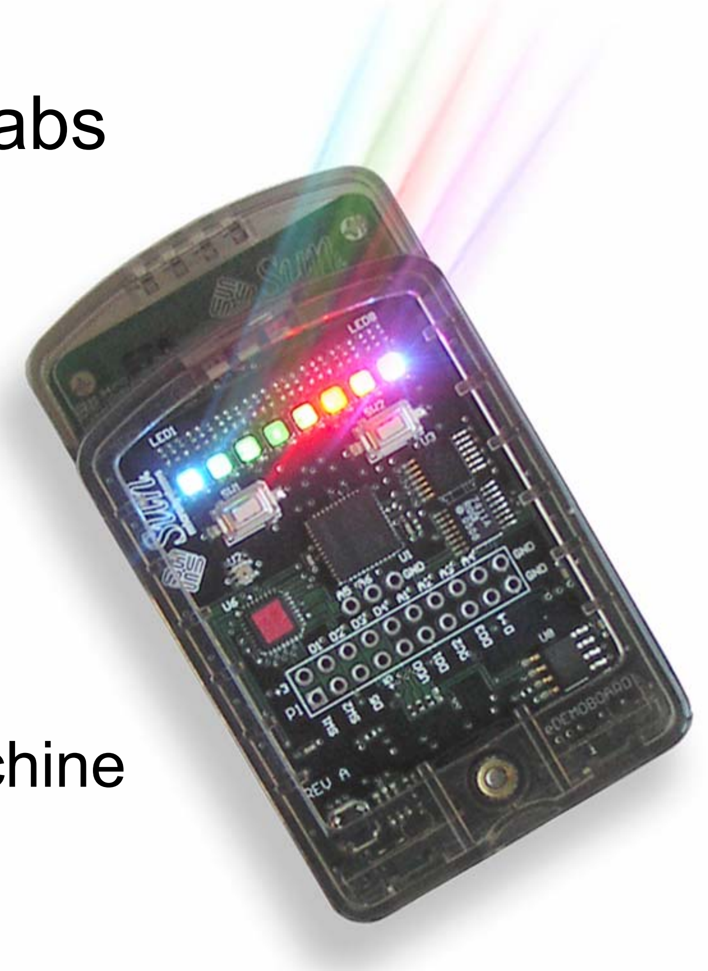
Bringing it All Together

Summary and Resources

Demos

Project Sun SPOTs
Small Programmable Object Technology

- Research project from Sun Labs
- Platform for wireless sensor network applications
- Helping to build the “network of things”
- Several areas of research
  - Java platform on small devices
  - More portable Java Virtual Machine (JVM™ machine)
  - Isolate application model



The terms “Java Virtual Machine” and “JVM” mean a Virtual Machine for the Java™ platform.

SPOT Processor Board

- 180MHz 32-bit ARM 920T CPU
  - 512Kb RAM, 4Mb FLASH
- Chipcon 2420 radio package
  - 2.4GHz frequency
  - IEEE 802.15.4 (Low rate PAN protocol)
- USB interface
- 3.7V 750 mAh Li-Ion battery
- Power consumption 40-100mA
  - Depending on radio/LED/peripheral usage
  - 40 µA deep sleep mode

SPOT Demo Sensor Board

- Accelerometer
- Temperature sensor
- Light sensor
- 8 tri-colour LEDs
- 2 push-button switches
- Analog to digital input pins
- GPIO pins
- High current (100mA) output pins

Squawk Virtual Machine

- Objective: very portable, small footprint JVM machine
  - No underlying OS
  - Runs on “bare metal”
- Most of code written in Java programming language
  - Interpreter and low level I/O code written in C
  - Everything else in Java programming language
- Provides Java Platform, Micro Edition (Java ME) CLDC 1.1 environment
  - Additional libraries for specific functions such as sensors, LEDs, etc.

Agenda

Introduction to Project Sun SPOTs

Interfacing Useful Hardware to SPOTs

Using Java 3D API for Virtual Reality

SPOT Interaction Software

Bringing it All Together

Summary and Resources

Demos

Analog to Digital Converters

- SPOT has six ADC lines accessible via external header pins
  - Firmware currently only supports four
- Apply input that is in range 0-3V
- Read value with 10-bit resolution via **`IScalarInput`** class

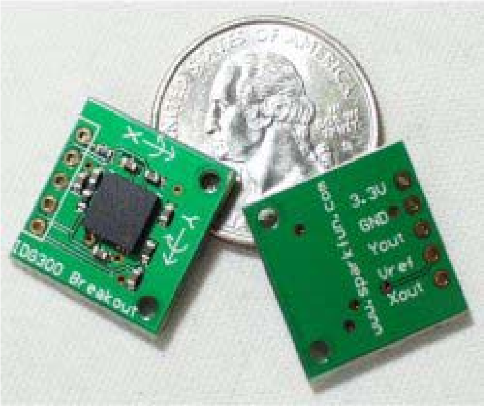
```
EDemoBoard db = EDemoBoard.getInstance();
IScalarInput analog =
    db.bindScalarInput(EDemoBoard.A0);
int analogValue = analog.getValue();
```

Accelerometer

- SPOT has built in 3-axis accelerometer
  - Uses ST-Micro LIS3L02 component
- Scale can be set to 2G or 6G
- Acceleration is measured relative to gravity
  - Tilting the SPOT changes the value

```
EDemoBoard db = EDemoBoard.getInstance();
IAccelerometer3D acc = db.getAccelerometer();
acc.setRange(0);  // 2G
IScalarInput xAccel = acc.getX();
int xa = xAccel.getValue();
```

Solid State Gyroscope



- IDG-300 dual-axis gyroscope
- 3V supply can be taken from SPOT
- X and Y lines connect to ADC pins on SPOT
- Use two mounted orthogonally for full 3D data
- Provides rotational velocity
  - Can be used to calculate change in orientation of SPOT
  - Change is 2mV/degree/second
  - Some drift creeps in—needs to be accounted for

The P5 Data Glove

- Designed for gaming applications
- Uses proprietary hardware and software
- Required modification to work with SPOT
  - Very fiddly soldering to surface-mounted connector
- Two gyros glued inside
- SPOT mounted on top using Velcro

Data Glove Software

- Initialise all inputs from ADC
- Initialise radio communication via broadcast
- Run thread to check values from sensors
  - Use event model to send changes to the PC
- Calibration required to determine appropriate values for bend sensors and accelerometer
- For smooth mouse movements send data every 1/25 second
  - Standard frame refresh rate

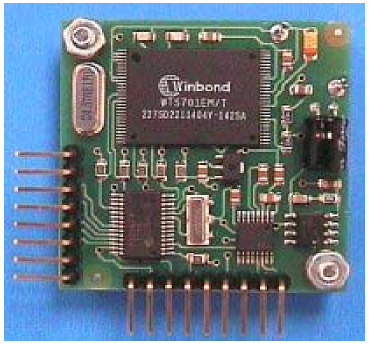
Game Pad Thumb Joystick

- Remove from cheap game pad
- Left-right and back-forward wired to ADC lines
  - Implemented as potentiometers
- Push-button wired to digital input
  - Switch pulls pin from 0 to 3V

Compass Sensor

- Useful for determining orientation
  - Z-axis of accelerometer not really suitable
- CMPS03 magnetic compass module
  - Uses 2 Philips KMZ10A sensors
  - 0.1 degree resolution, 3–4 degrees accuracy
- Separate head-mounted SPOT
- Uses I2C interface
  - Integration took a bit of work
  - Not as simple as other sensor connection

Feedback



- Talking SPOT
- RS-232 interface SP03 board
  - 30 pre-recorded phrases
  - Text to speech capable
- SPOT can drive pins as UART
  - Required modified firmware for demo board
- Use MAX3232 as line driver
  - Convert TTL voltages to RS-232

Agenda

Introduction to Project Sun SPOTs
Interfacing Useful Hardware to SPOTs
Using Java 3D API for Virtual Reality
SPOT Interaction Software
Bringing it All Together
Summary and Resources
Demos

Virtual Reality

- Allows a user to interact with a computer-simulated environment
- Interactively
  - Makes the difference with two- and three-dimensional graphics mediums
  - Gives users some feeling of existence within an artificial world
- User representation
  - Avatar: complete virtual body
  - Part of a body such as a hand or as a controllable viewpoint

VR User Requirements

- Ease-of-use
- Conventions in navigation: be simple and intuitive
- Realism
- Degree of interaction and movement
  - Consider pre-set animations and fly-throughs
- Method of interaction—hardware and software
  - Mouse, keyboard, joystick or a touch-sensitive screen
  - HMD and sensor or data-gloves
- Speed: smooth movement, no long waits

Building the World

- Objects
  - Created using authoring tools, CAD software, 3-D scanners or by stitching images together
  - Use Bubble Worlds
  - Combine polygons to create three-dimensional objects
- Authoring tools
  - Simple as using a text editor (VRML developers)
  - Use a VR authoring tool: AutoCAD, 3D Studio Max, to ease the process
  - Optimize
    - Remove any unnecessary facets that slow down the rendering of the object
    - LOD operations prevent rendering detailed objects that the user cannot “see” from his viewpoint

Bubble Worlds

- Quick, easy and cheap way to present landscapes and indoor environments
- Excellent for guided walks
- Seamless panoramic image projected inside surface of a cylinder or sphere and viewed through an interactive window
- Give the impression of viewing an entire space from the ground to the sky and 360 degrees around through a moveable window
- At a minimum users will be able to pan, tilt and zoom

VRML: Virtual Reality Modeling Language

- Developed by the Web3D Consortium
- Designed for use on the Internet
- Is both a scene description language and a file format for virtual worlds
- The language is used to describe the geometry and behaviour of three-dimensional scenes
- VRML 1.0, VRML 2.0 (ISO), VRML 97 (ISO)
- Transformation, viewpoint setting, definition of lighting within the world and “shapehints”

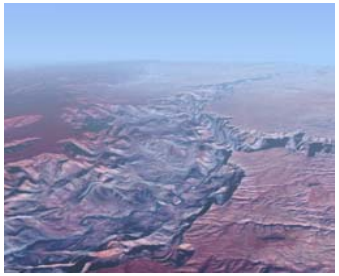
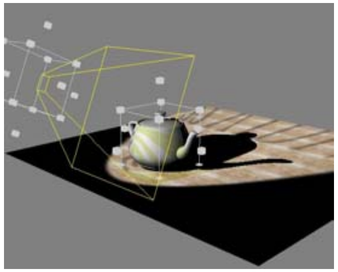
J3D-VRML97: VRML and Java 3D API

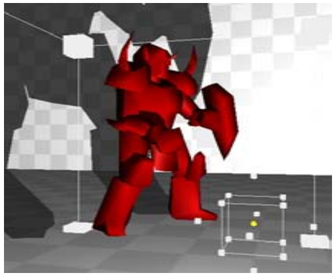
- Java 3D™ API loader for VRML97 models
- https://j3d-vrml97.dev.java.net/

```
import org.jdesktop.j3d.loaders.vrml97.VrmLoader;
   ...

   VrmlLoader loader = new VrmlLoader()
   BufferedReader in = new BufferedReader(new
     InputStreamReader(
       new FileInputStream(filename), "UTF8"));
   Scene scene = loader.load(in);
   BranchGroup branch = scene.getSceneGroup();
   ...
```

JOGL Project: Java Binding for the OpenGL® API (JOGL)

- Java™ Binding for the OpenGL® API (JSR-231)
- Designed to provide hardware-supported 3D graphics to applications written in Java programming language.
- Full access to the APIs in the OpenGL 2.0 specification and integrates with the AWT and Swing widget sets
- https://jogl.dev.java.net/
- JOGL demos: https://jogl-demos.dev.java.net/

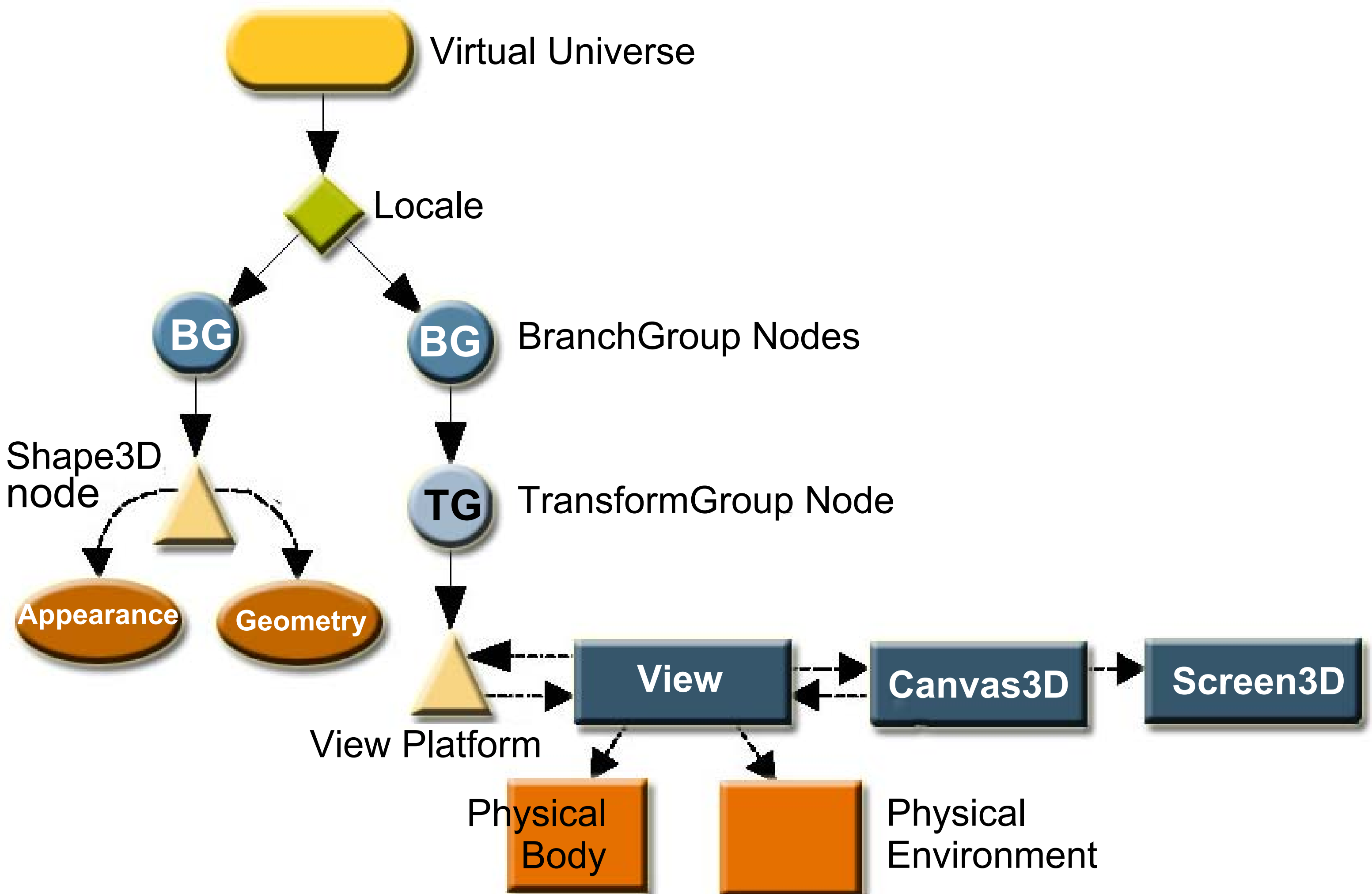





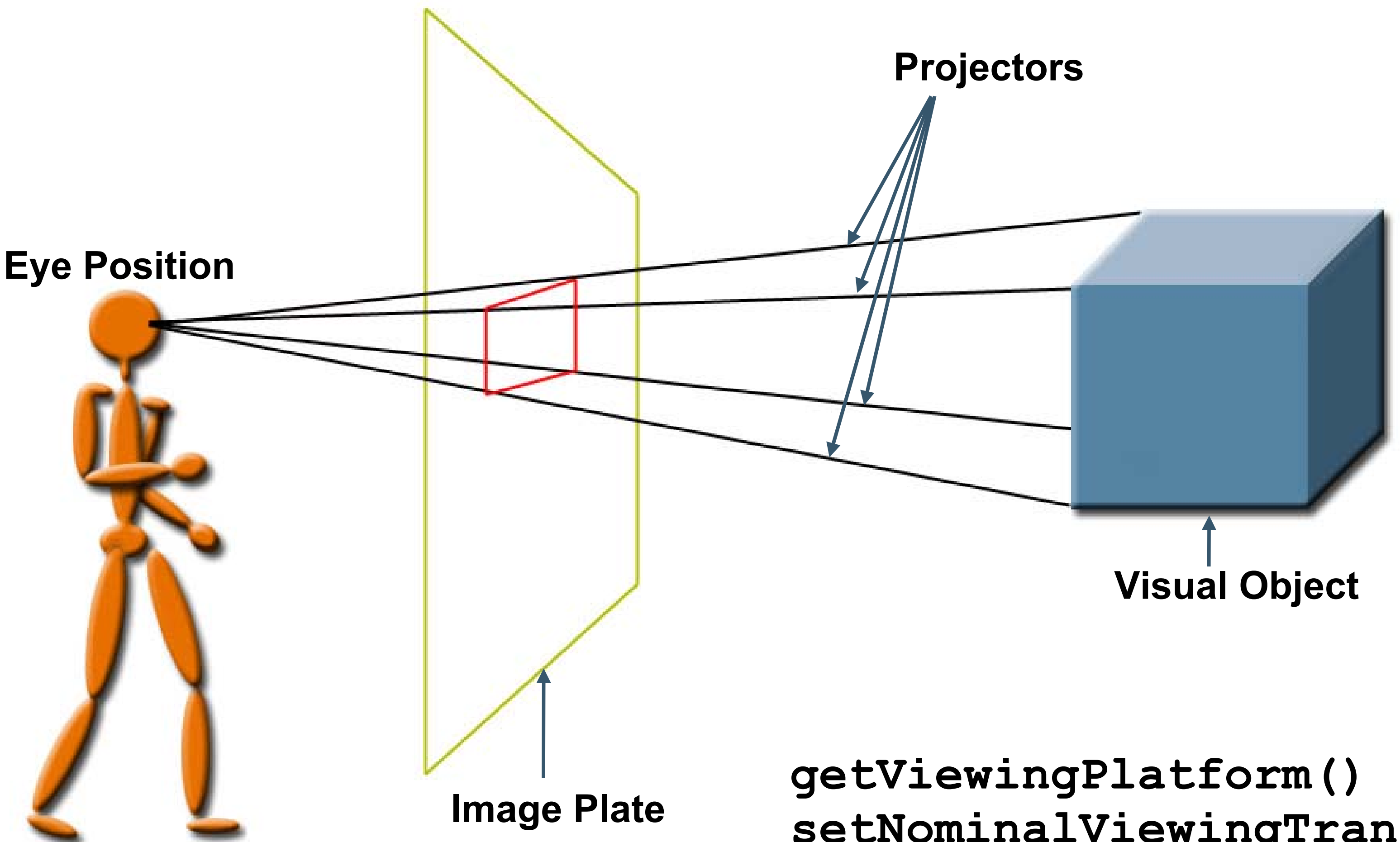
JOAL Project: OpenAL and Java Technology

- Reference implementation of the Java bindings for OpenAL API
  - Designed to provide hardware-supported 3D spatialized audio for games/apps written in Java platform
  - Make the development of high performance games/apps in Java platform a reality
- Hosts the Sound3D Toolkit
  - High level API for spatialized audio built on top of the OpenAL bindings
  - Provide access to all the features of OpenAL through an intuitive, easy-to-use, object-oriented interface
- https://joal.dev.java.net/

Example of a Scene Graph



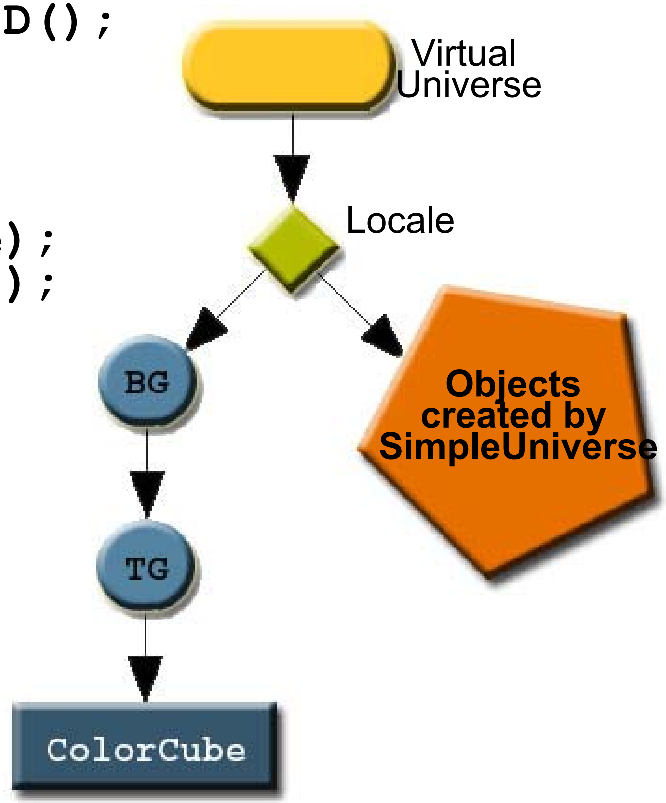
Conceptual Drawing of Image Plate and Eye Position in a Virtual Universe




```
getViewingPlatform()
setNominalViewingTransform()
```

Rotation Transformation Application

```
...
BranchGroup scene = new BranchGroup();
Transform3D rotate = new Transform3D();
Transform3D tempRotate =
                       new Transform3D();
rotate.rotX(Math.PI/4.0d);
tempRotate.rotY(Math.PI/5.0d);
rotate.mul(tempRotate);
TransformGroup objRotate =
            new TransformGroup(rotate);
objRotate.addChild(new ColorCube(0.4));
scene.addChild(objRotate);
simpleU.addBranchGraph(scene);
...
```



Loader Classes

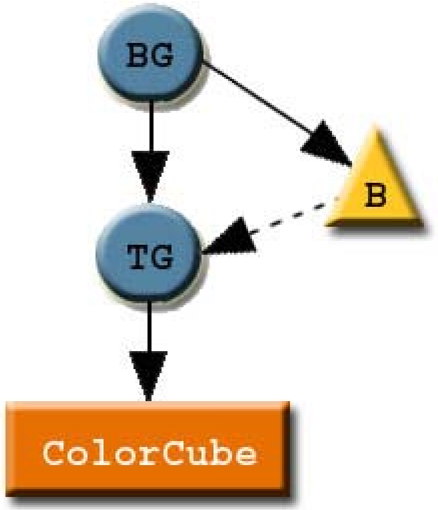
- **`Loader`**: specifies the elements that should be loaded from a file written in a given 3d format
- **`Scene`**: extracts Java 3D API scene graph information from the loaded file
- **`Lw3dLoader`**: for Lightwave 3D scene files
- **`ObjectFile`**: ObjectFile for Wavefront .obj files
- **`LoaderBase`**: implements the ***`Loader`*** interface in a generic way to encourage the building of loaders for other 3D formats through subclassing
- 3DS,COB,DXF,LWS,OBJ,VTK,WRL…

Using Loaders

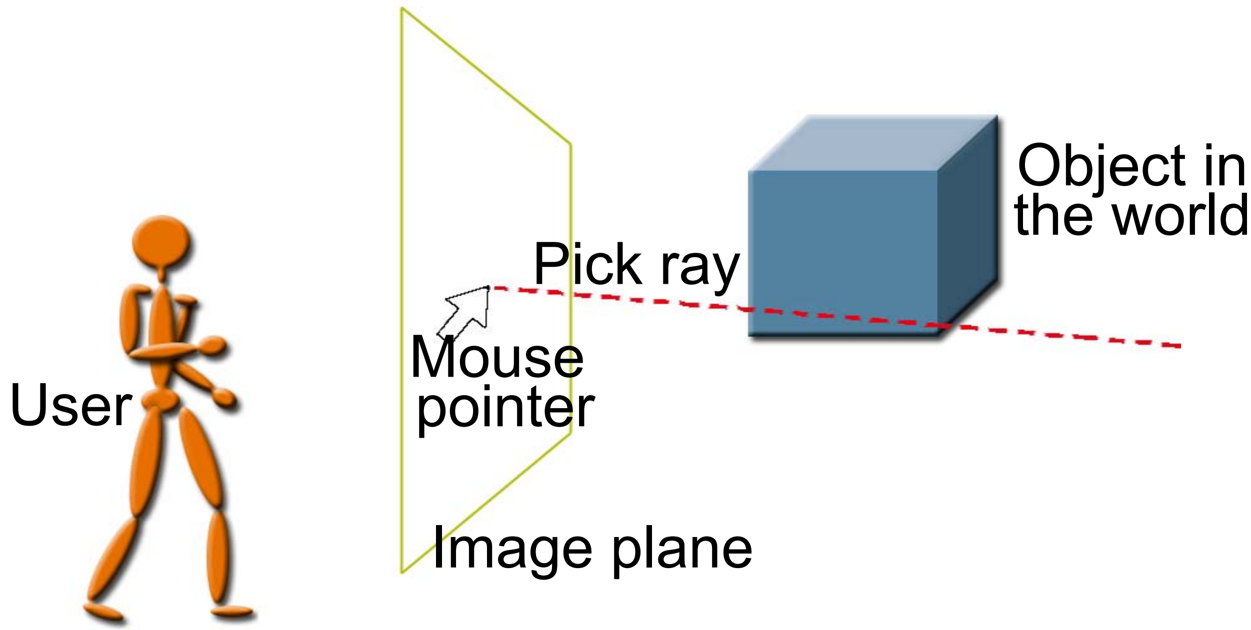
```
import com.sun.j3d.loaders.objectfile.ObjectFile;
import com.sun.j3d.loaders.ParsingErrorException;
import com.sun.j3d.loaders.IncorrectFormatException;
import com.sun.j3d.loaders.Scene;
import javax.media.j3d.*;
import javax.vecmath.*;
...

   BranchGroup objRoot = new BranchGroup();
  ObjectFile f = new ObjectFile();
  Scene s = null;
  try {
      s = f.load(filename);
  }
  catch (FileNotFoundException e){...}
  catch (ParsingErrorException e){...}
  catch (IncorrectFormatException e){...}
  objRoot.addChild(s.getSceneGroup());
...
```

Behavior Objects

- Both interaction and animation are specified with **`Behavior`** objects
- A **`Behavior`** object changes the scene graph in response to events
  - Key presses, mouse moves, object collisions, passage of time, etc.
- User-defined Behavior classes triggered by WakeupCondition objects
- KeyNavigatorBehavior, Mouse Behavior PickMouseBehavior, Interpolator...

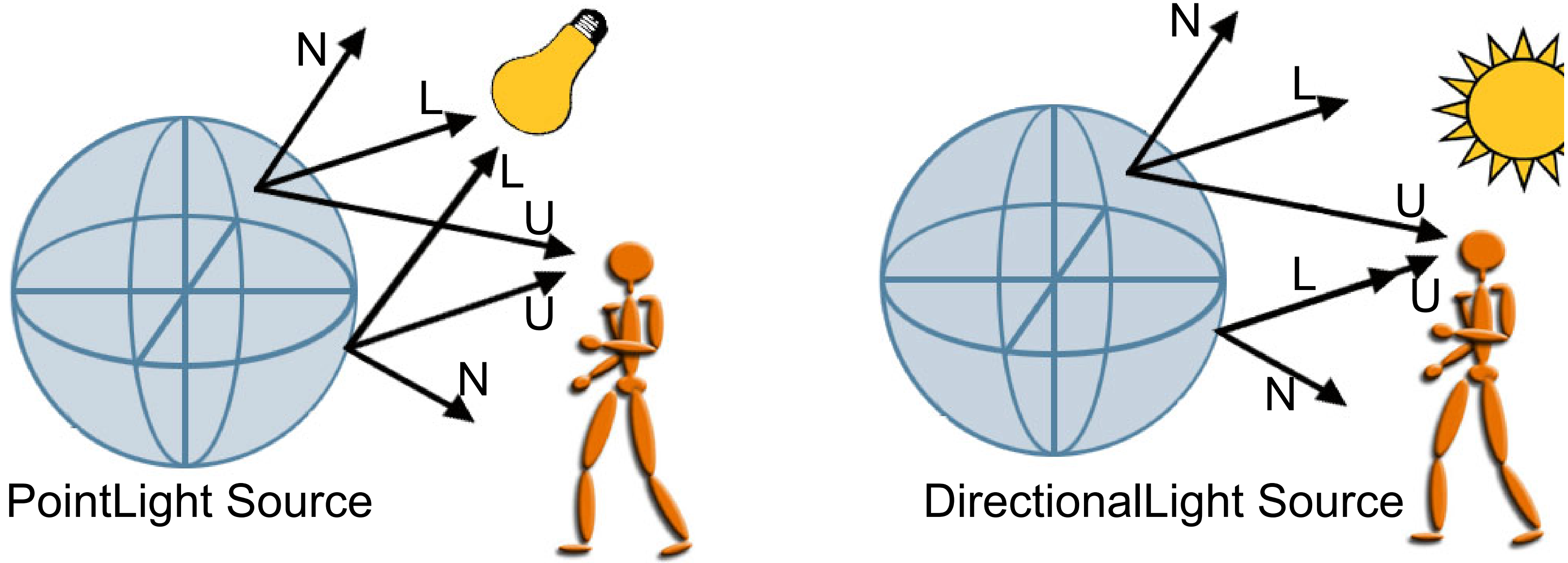


Picking




```
objRotate = new TransformGroup(transform);
objRotate.setCapability(
             TransformGroup.ENABLE_PICK_REPORTING);
objRoot.addChild(objRotate);
objRotate.addChild(new ColorCube(0.4));
pickRotate = new PickRotateBehavior(
                     objRoot,canvas, behaveBounds);
objRoot.addChild(pickRotate);
```

Lights




```
AmbientLight lightA = new AmbientLight();
lightA.setInfluencingBounds(
                        new BoundingSphere());
scene.addChild(lightA);
```

Agenda

Introduction to Project Sun SPOTs

Interfacing Useful Hardware to SPOTs

Using Java 3D API for Virtual Reality

SPOT Interaction Software

Bringing it All Together

Summary and Resources

Demos

Robot Class

- Part of AWT, designed for writing tests
- Emulates control of mouse from within application
- Movement is absolute, rather than relative

```
mousePress(int buttons)
mouseRelease(int buttons)
mouseMove(int x, int y)
```

Determining Position

- ***Simple physics***

$$d = \int_0^t v \, dt \qquad v = \int_0^t a \, dt$$

- Not so simple with available data
- Remember, accelerometer value changes with tilt
- Need to combine gyro data with calibrated accelerometer data
- Ultimately get position relative to start point

Radio Positioning

- SPOT sends “ping” radio signal
- APIs provide simple radio signal strength access
  - **`Radiogram.getRssi()`**
- Take signal strength from multiple basestations
  - Inverse square law for distance
  - Triangulate position
  - More basestations means more degrees of accuracy
- Stability of signal strength is not high
  - Resolution of position changes is therefore low
  - Good enough for some situations

Networking with Dead Reckoning

- Technique of calculating your present position from past position and your speed
- Updating your position transmit only velocities changes
- Exact position of object extrapolated from its last known location and velocity
- Exchange simple deterministic instructions:
  - "Follow this ship," "Orbit that planet"
- Communicated in very little data

Agenda

Introduction to Project Sun SPOTs

Interfacing Useful Hardware to SPOTs

Using Java 3D API for Virtual Reality

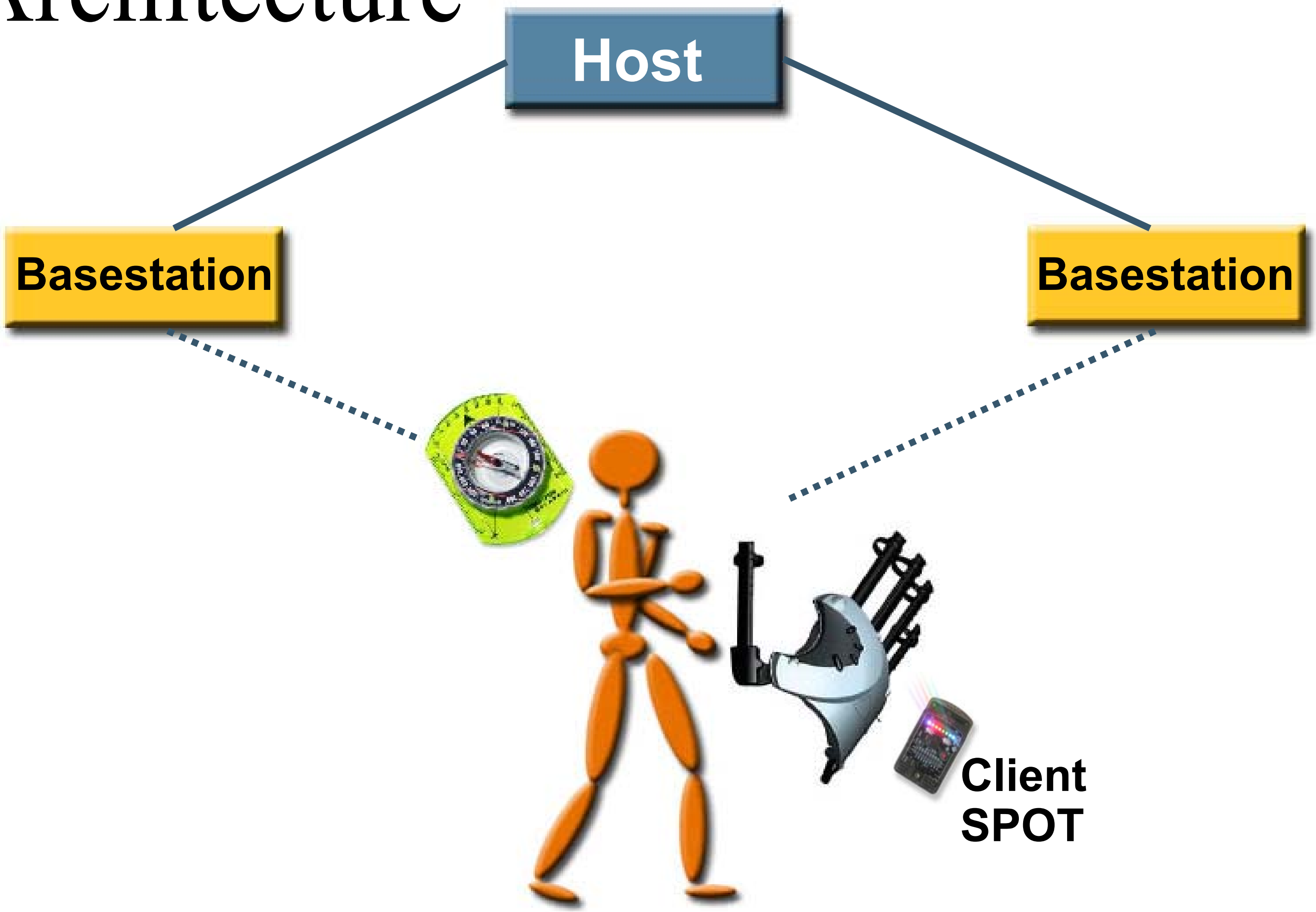
SPOT Interaction Software

Bringing it All Together

Summary and Resources

Demos

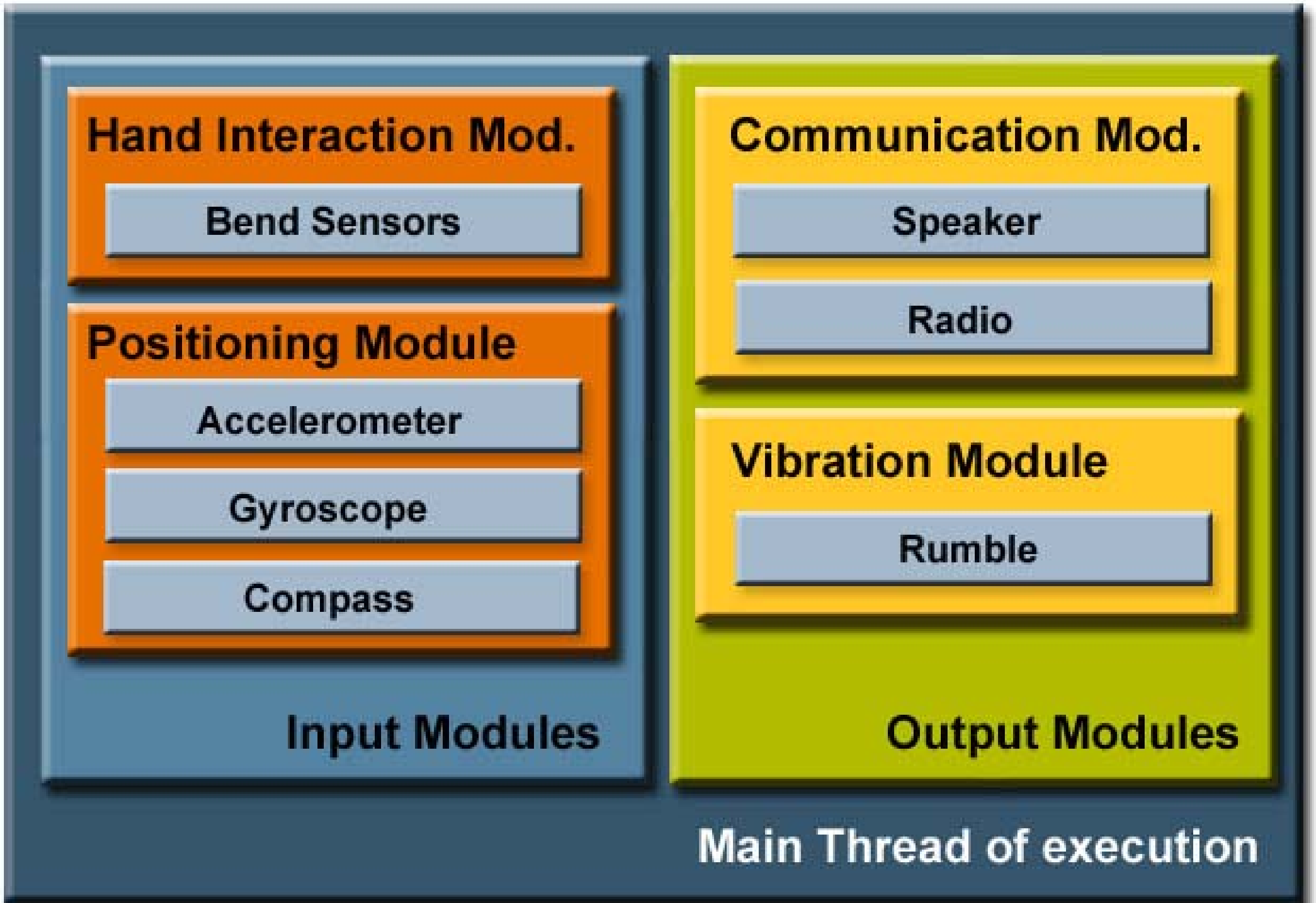
Architecture



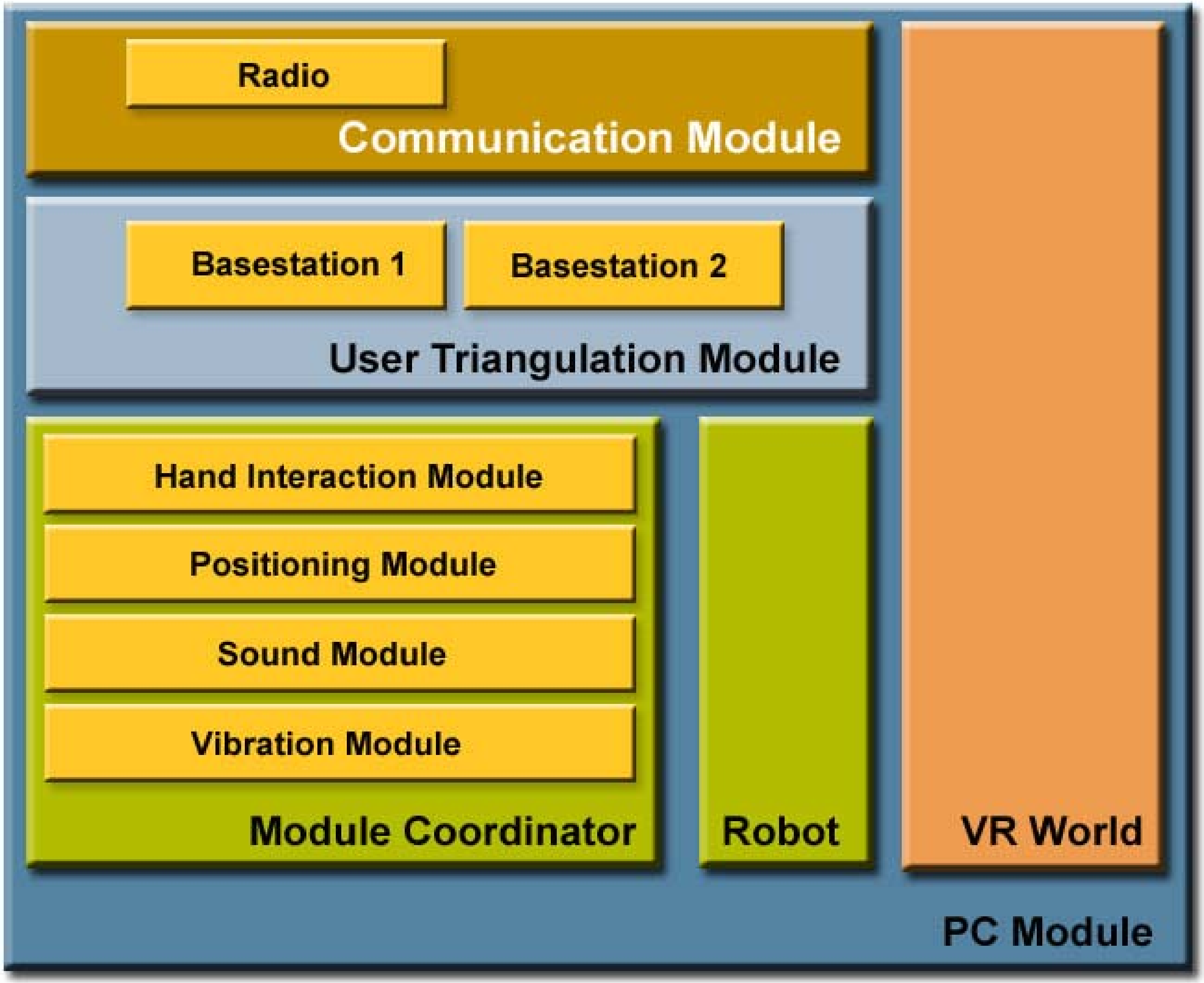
Human Game Interface

- Position of player is determined via radio signal strength to multiple basestations
- Direction player is looking comes from compass sensor
  - Changes view in 3D environment
- Movement of hand comes from data glove
  - Position
  - Tilt
  - Finger movements

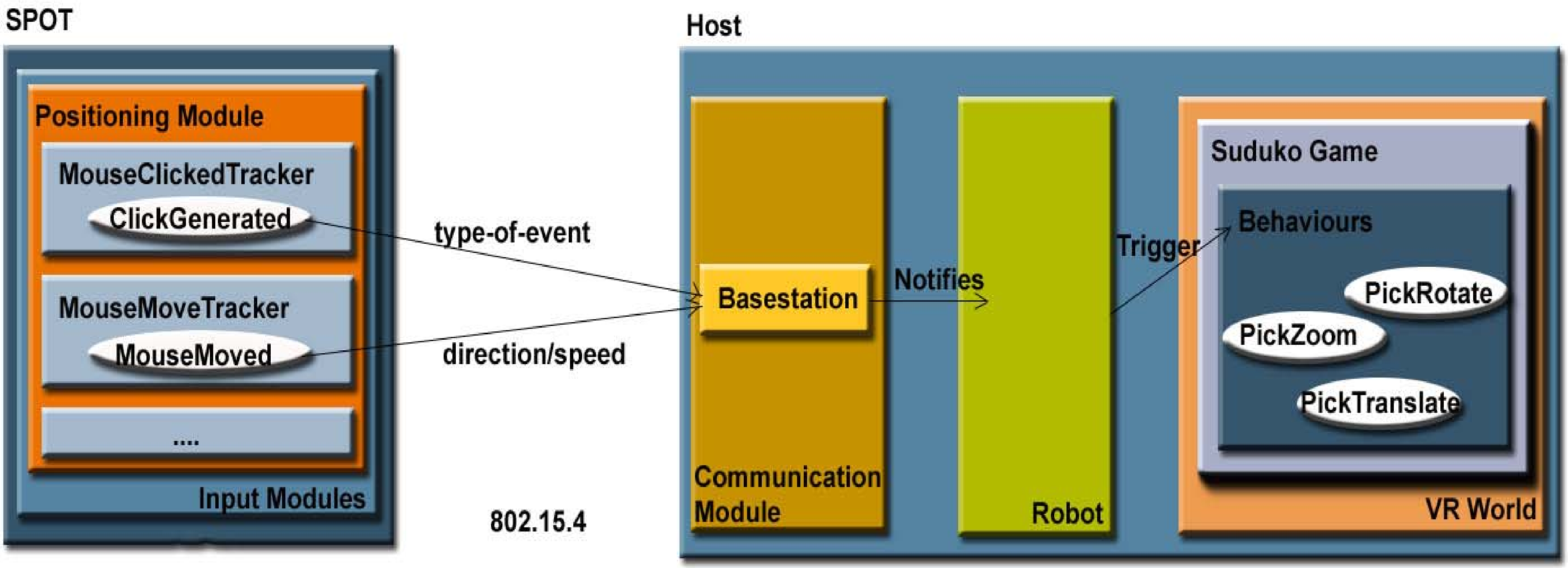
Client SPOT Architecture



Host Module Architecture



Control Flow



Agenda

Introduction to Project Sun SPOTs

Interfacing Useful Hardware to SPOTs

Using Java 3D API for Virtual Reality

SPOT Interaction Software

Bringing it All Together

Summary and Resources

Demos

Summary

- SPOTs are a small easy to use platform for embedded programming applications
- Programming in Java platform makes life much easier
- Integration with most peripheral hardware is a breeze
- Can be used to build sophisticated interactive games
- Lots of fun!

Further Information

- Web resources
  - `http://www.sunspotworld.com`
  - `http://java3d.dev.java.net`
  - `http://j3d-vrml97.dev.java.net`
  - `http://jogl.dev.java.net`
  - `http://joal.dev.java.net`
- Other sessions at 2007 JavaOne[SM] conference
  - BOF-1692: Introducing the Sun SPOT
  - BOF-1892: SPOTBot, Turning a Sun SPOT into a rugged and affordable robot
  - TS-1786: Writing Darkstar Applications

DEMOs

3D Virtual Gaming

Q&A

Simon Ritter
Angela Caicedo

Sun SPOTs in Action: 3D, Virtual Reality and Gaming

Simon Ritter
Technology Evangelist
Angela Caicedo
Technology Evangelist
Sun Microsystems

Session TS-1780