Write a 3D Game in the Java™ Programming Language in Less Than Fifty Minutes

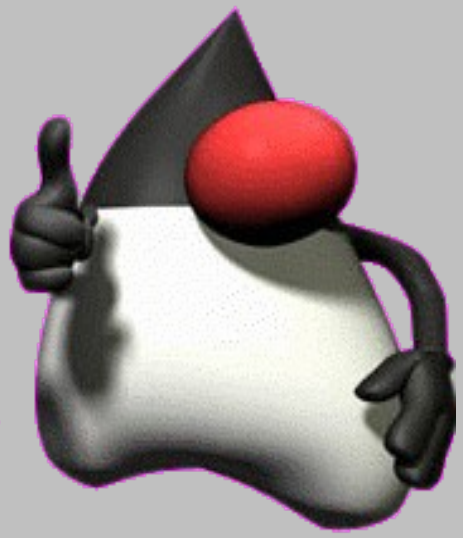
Erik Hellman <erik.hellman@ibs.se>

Consultant
IBS JavaSolutions
http://jsolutions.se

TS-3073

Rapid Game Development

See how easy it is to write games in Java™ code…



Duke Bean'Em!

About the Speaker

- Erik Hellman, Java Technology Consultant from Gothenburg, Sweden
- Has never written a computer game before
- Only basic knowledge of OpenGL
- Lacks artistic talent…
- …and suffers from color blindness :-)

Agenda

The Game
Tools and Libraries
Scene Graph
Complex Models
First-Person Shooter Navigation
Collision Detection
Multi-Player Action

Agenda

The Game

Tools and Libraries

Scene Graph

Complex Models

First-Person Shooter Navigation

Collision Detection

Multi-Player Action

Duke Bean'Em!

- A First-Person Shooter! (sort of…)
- Play as Duke
- Shoot coffee beans at opponents
- Simple flat course surrounded by a skybox
- Simple textured cubes used for obstacles
- Two-player game over network

Things Left Out…

- Computer controlled opponents (i.e., AI players)
- Complex visual effect (shadows…)
- Advanced physics (gravity, realistic collisions…)
- Sound effects

Agenda

The Game

Tools and Libraries

Scene Graph

Complex Models

First-Person Shooter Navigation

Collision Detection

Multi-Player Action

Tools of the Trade…

- Blender—Free 3D modeler
- GIMP—Image manipulation, conversions etc.
- JOGL—Java Binding for the OpenGL® API
- JInput—Game controller
- Vecmath (from Java3D)—Coordinates, etc.
- XMLBeans—For loading models

Agenda

The Game

Tools and Libraries

Scene Graph

Complex Models

First-Person Shooter Navigation

Collision Detection

Multi-Player Action

Transformations

- glTranslatef(x, y, z);
- glRotatef (degrees,x, y, z);
- glScalef(x, y, z)
- glLoadIdentity();

$$\begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Translation (moving)

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation around X-axis

$$\begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Scaling

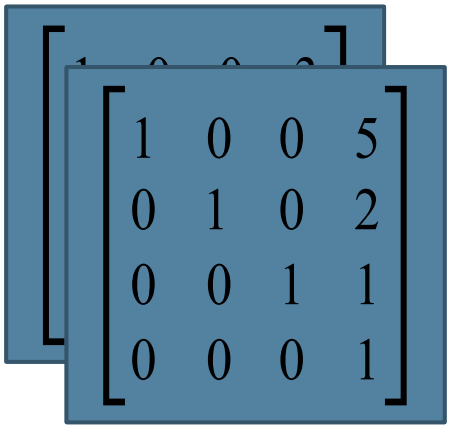
$$\begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation around Y-axis

$$\begin{bmatrix} \cos\gamma & -\sin\gamma & 0 & 0 \\ \sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

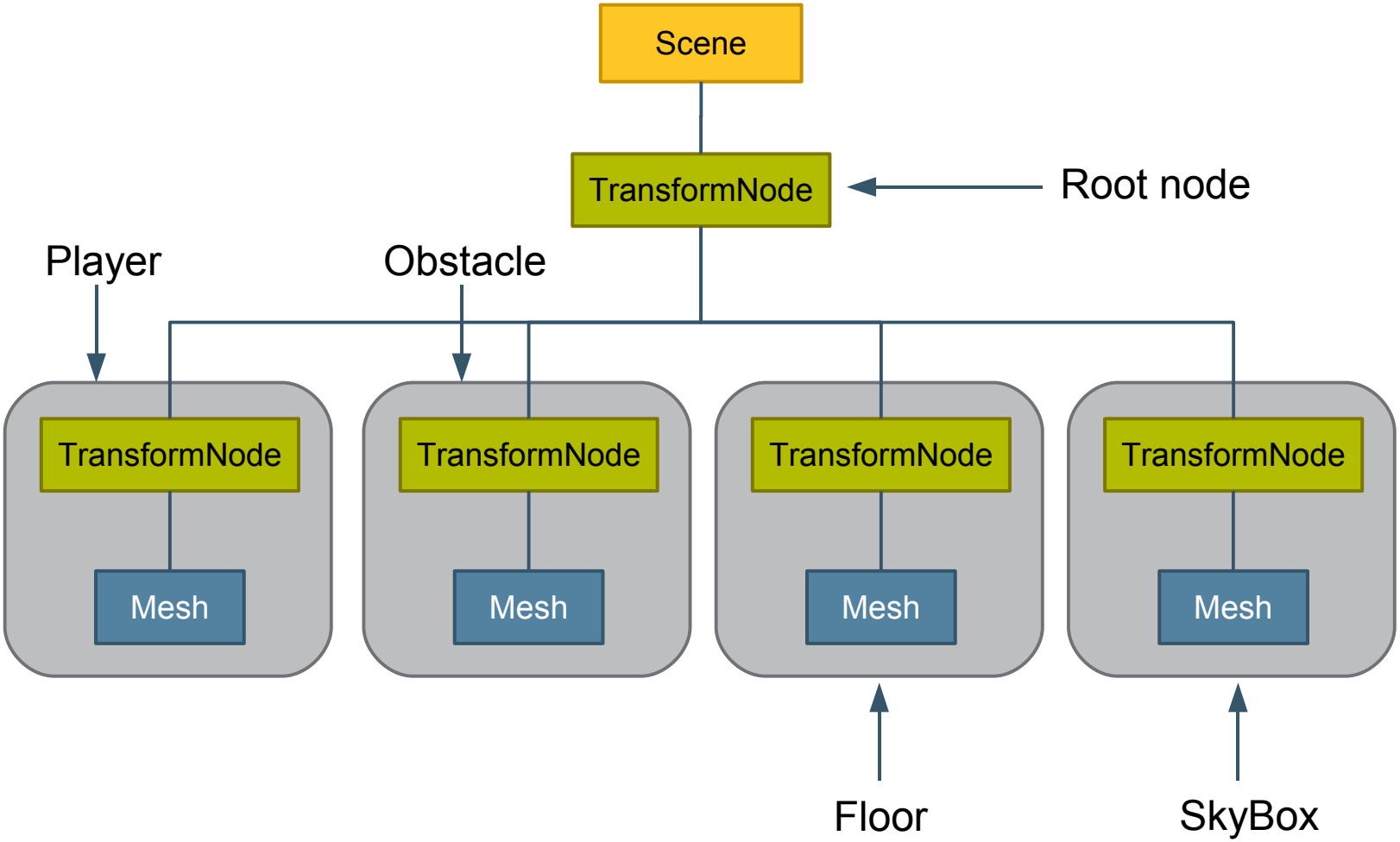
Rotation around Z-axis

Matrix stack




```
gl.glTranslatef(3,2,1)
gl.glPushMatrix()
gl.glTranslatef(2,0,0)
gl.glPopMatrix()
```

Maintaining Order in a 3D World



Node Interface

```
// Common interface for all Scene Graph nodes
public interface Node {
    public void renderNode(GL gl);
}
```

TransformGroup

```
// renderNode() implementation for TransformGroup
public void renderNode(GL gl) {
   gl.glPushMatrix();
   gl.glTranslatef(translation.x, translation.y,
                   translation.z);

   gl.glRotatef(rotation.x,1,0,0);
   gl.glRotatef(rotation.y,0,1,0);
   gl.glRotatef(rotation.z,0,0,1);

   gl.glScalef(scaling.x, scaling.y, scaling.z);

   for (Node child : children) {
      child.renderNode(gl);
   }
   gl.glPopMatrix();
}
```

Mesh

```
// renderNode() implementation for Mesh
public void renderNode(GL gl) {
   applyMaterials(); // apply all materials for this mesh
   gl.glEnableClientState(GL.GL_NORMAL_ARRAY);
   gl.glEnableClientState(GL.GL_VERTEX_ARRAY);
   gl.glEnableClientState(GL.GL_TEXTURE_COORD_ARRAY);

   gl.glNormalPointer(GL.GL_FLOAT, 0, normals);
   gl.glVertexPointer(3, GL.GL_FLOAT, 0, vertices);
   gl.glTexCoordPointer(3, GL.GL_FLOAT, 0, textureCoords);
   gl.glDrawElements(primitiveType, indices.capacity(),
                     GL.GL_UNSIGNED_INT, indices);

   gl.glDisableClientState(GL.GL_VERTEX_ARRAY);
   gl.glDisableClientState(GL.GL_NORMAL_ARRAY);
   gl.glDisableClientState(GL.GL_TEXTURE_COORD_ARRAY);
}
```

Agenda

The Game

Tools and Libraries

Scene Graph

Complex Models

First-Person Shooter Navigation

Collision Detection

Multi-Player Action

Loading a 3D Model

- Closed, proprietary formats
  - Lightwave, 3D Studio MAX, etc.
- Open formats
  - Collada, X3D (Web 3D), etc.
- Issues with loaders
  - Tightly coupled with scene-graph APIs
  - Semantics differ
  - Unnecessary information
- Duke Bean'Em uses Collada for all models

Build a Collada Loader in 3 Easy Steps

1. Download XMLBeans from xmlbeans.apache.org
2. Compile Collada schema into beans
3. Parse Collada XML file using the beans

Collada 1.4 Example

```
<library_geometries>
  <geometry id="Mesh_004" name="Mesh_004">
    <mesh>
      <source id="Mesh_004-Position">
        <float_array count="540"
id="Mesh_004-Position-array">0.42536 0.85714 -0.59032
0.43183 0.85824 -0.57594 ...</float_array>
        ...
      </source>
      <source id="Mesh_004-Normals">...</source>
      <triangles count="352" material="Hand">
        <input offset="0" semantic="VERTEX"
source="#Mesh_004-Vertex"/><p>1 0 5 5 4...</p></input>
      </triangles>
    </mesh>
  </geometry>
</library_geometries>
```

Using XMLBeans With Collada

```
// Loading our Collada model
public TransformGroup loadCollada(String fileName) {
    COLLADADocument colladaDocument =
        COLLADADocument.Factory.parse(fileName);

// Traverse the object graph and build the TransformGroup
    LibraryGeometries[] geometriesArray =
        colladaDocument.getCOLLADA().
        getLibraryGeometriesArray();
    for(LibraryGeometries geometries : geometriesArray) {
        ...
    }

    ...
    return transformGroup;
}
```

DEMO

Scene Graph and Loading Duke

Agenda

The Game

Tools and Libraries

Scene Graph

Complex Models

First-Person Shooter Navigation

Collision Detection

Multi-Player Action

Using JInput for Navigation

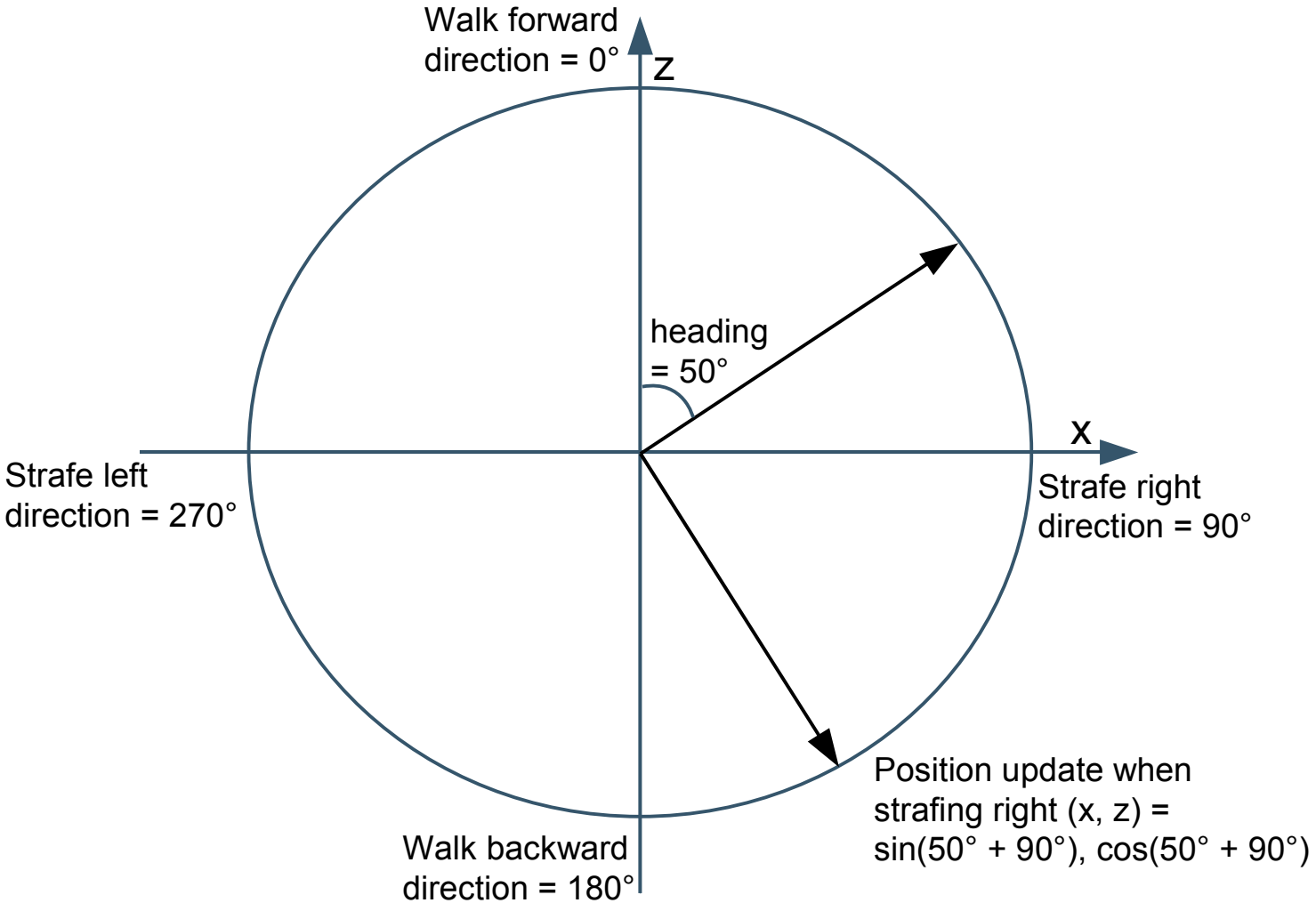
```
// Poll mouse and keyboard every 20 ms
while(doInput) {
    mouse.poll();
    float rotationDiff = mouse.getX().getPollData();

    keyboard.poll();
    boolean wDown = keyboard.isKeyDown(Key.W);
    boolean sDown = keyboard.isKeyDown(Key.S);
    boolean aDown = keyboard.isKeyDown(Key.A);
    boolean dDown = keyboard.isKeyDown(Key.D);

    // Update current player position and rotation and
    // set the new camera position
    ...

    try { Thread.sleep(20); } catch(Exception e) {}
}
```

Determine Movement (1)



Determine Movement (2)

```
// Determine the new position based on
// current heading and direction of movement
// and update the rotation
rotation.y += rotationDiff;
if(moving) {
    position.z += (float) Math.
       sin(Math.toRadians(rotation.y + direction));
    position.x += (float) Math.
       cos(Math.toRadians(rotation.y + direction));
}
```

Setting the Camera

```
// Determine the new camera position based on
// the heading of the player

cameraPosition.x = position.x +
      (float) Math.sin(Math.toRadians(rotation.y + 180))
      * CAMERA_DISTANCE;
cameraPosition.z = position.z +
      (float) Math.cos(Math.toRadians(rotation.y + 180))
      * CAMERA_DISTANCE;

lookAt.x = position.x +
      (float) Math.sin(Math.toRadians(rotation.y))
      * CAMERA_DISTANCE;
lookAt.z = position.z +
      (float) Math.cos(Math.toRadians(rotation.y))
      * CAMERA_DISTANCE;
```

Agenda

The Game

Tools and Libraries

Scene Graph

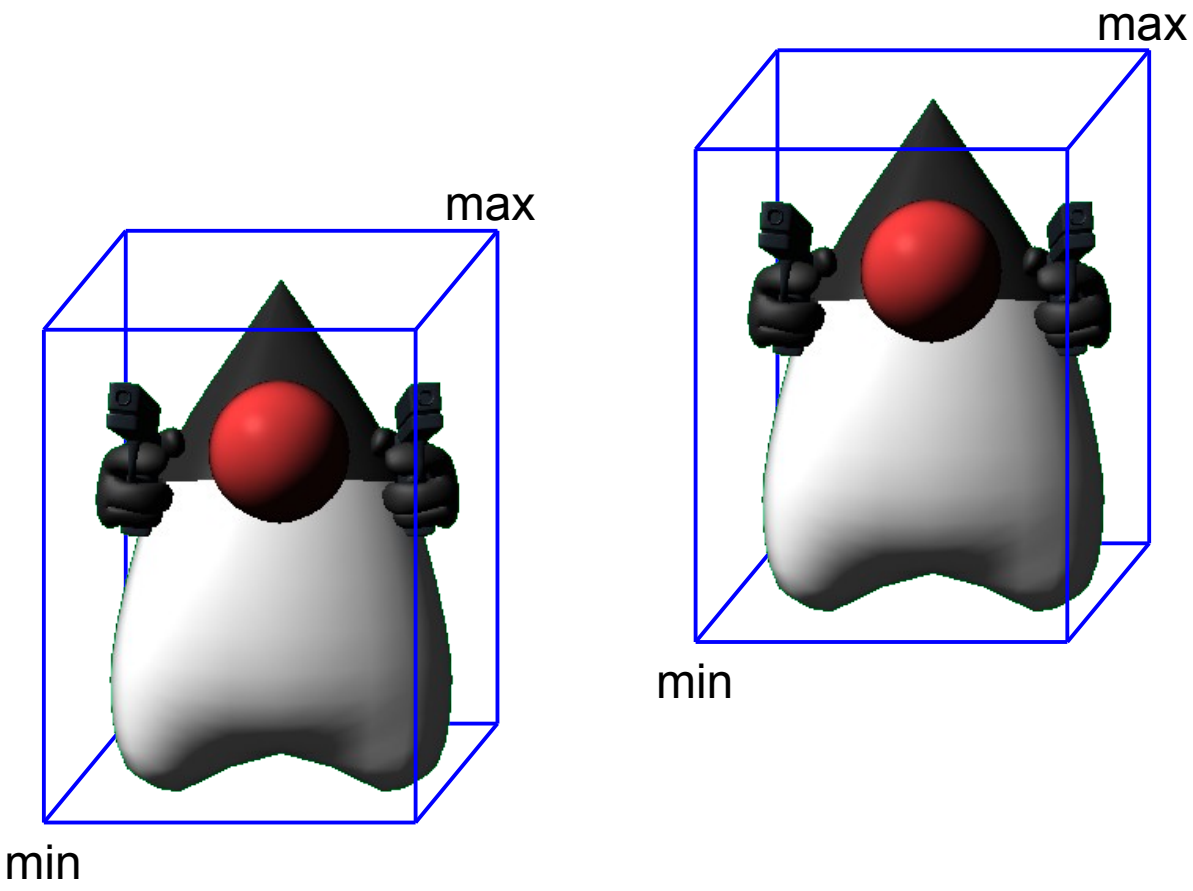
Complex Models

First-Person Shooter Navigation

Collision Detection

Multi-Player Action

AABB Collision Detection



Axis-Aligned Bounding Box

```
// A simple AABB for detecting collisions
public class BoundingBox {
    public Point3f min;
    public Point3f max;

    public boolean isCollide(BoundingBox otherBox) {
        return otherBox != this &&
               otherBox.max.x > min.x &&
               otherBox.min.x < max.x &&
               otherBox.max.y > min.y &&
               otherBox.min.y < max.y &&
               otherBox.max.z > min.z &&
               otherBox.min.z < max.z;
    }
}
```

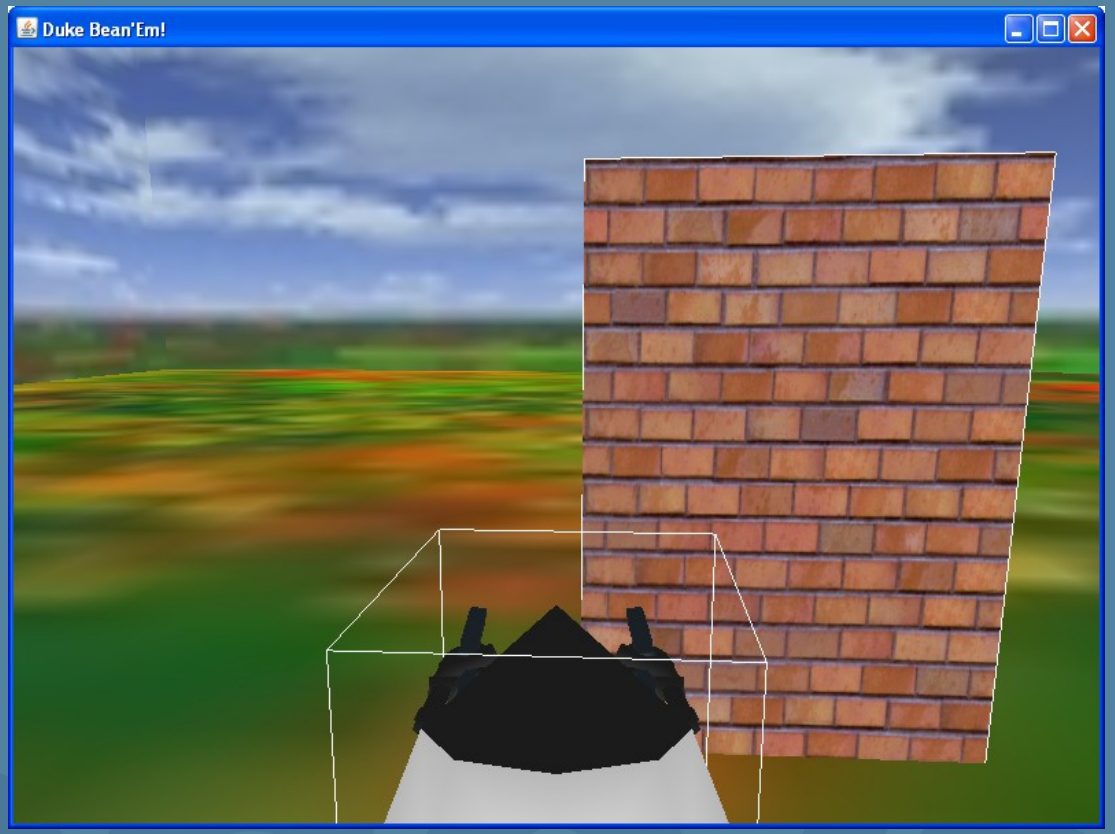
Keeping Track of Collisions

```
// Calculate new position, check for collisions and
// update the players position if no collision occured
public void move() {
    Point3f position = player.getTranslation();
    position.x += (float) Math.sin(....);
    position.z += (float) Math.cos(....);

    // Check for collisions against other players
    for(TransformGroup otherPlayer : players) {
        if(player.getBoundingBox()
            .isCollide(otherPlayer.getBoundingBox())) {
            position.x -= (float) Math.sin(....);
            position.z -= (float) Math.cos(....);
            break;
        }
    }
}
```

DEMO

Movement and Collisions



Agenda

The Game

Tools and Libraries

Scene Graph

Complex Models

First-Person Shooter Navigation

Collision Detection

Multi-Player Action

Multi-Player Issues

- Latency
- Packet loss
- Cheating
- Different bandwidth
- And many, many more…

- Project Darkstar is a nice platform…
  https://games-darkstar.dev.java.net/

Duke Bean'Em Multiplayer

- Each player sends updates to the other
  - New player position and health
  - New CoffeeBeans (sent only once) shot by the player
- CoffeeBeans updated locally only
- CoffeeBeans decay over time

Simple Multi-Player Implementation

```
// Transfer object for sending updates
public class GameUpdate implements Serializable {
    public int health;
    public ObjectType objectType;
    public String objectId;
    public Point3f position;
    public Point3f rotation;
    public BoundingBox boundingBox;
}
```

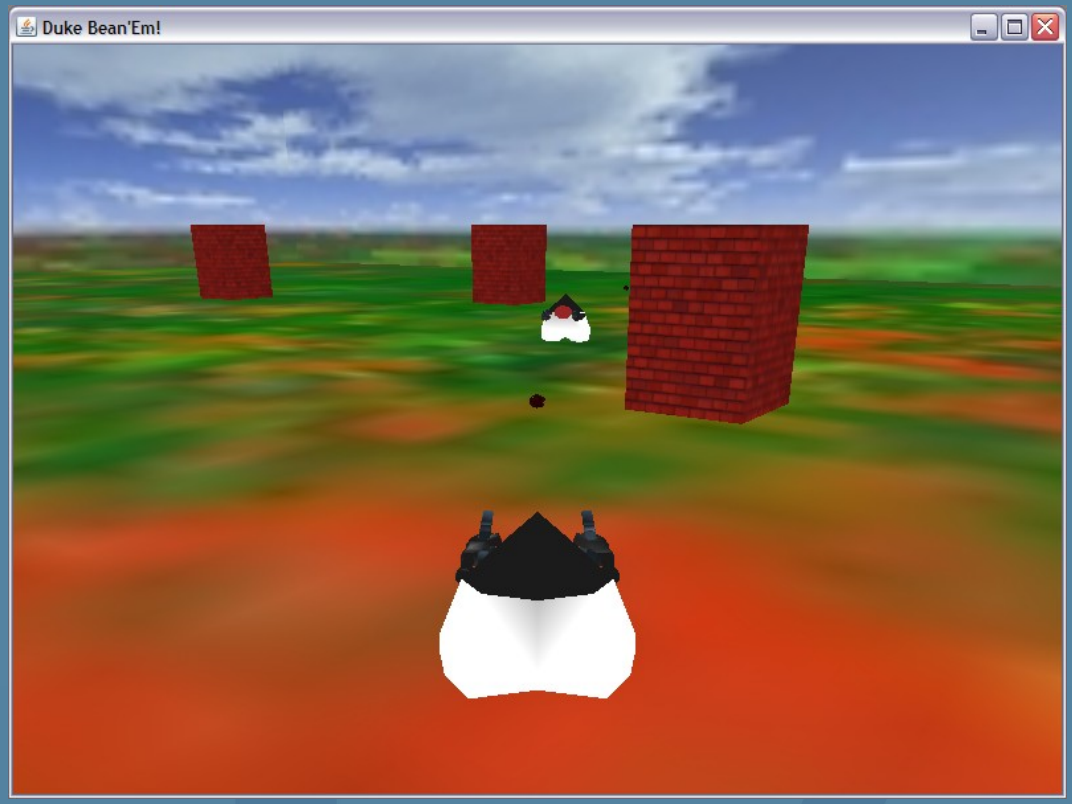
Receiving updates

```
// Thread for listening for incoming updates
public void run() {
  ...
  while(doListen) {
    GameUpdate gameUpdate =
      (GameUpdate) input.readObject();

    if(gameUpdate.objectType.equals(ObjectType.player)) {
      TransformGroup player = players.
          getPlayer(gameUpdate.objectId)
      player.setTranslation(gameUpdate.
                            getTranslation());
      player.setRotation(gameUpdate.getRotation());
    }
    ...
  }
}
```

DEMO

Final Game



Summary

- All you need is…
  - A simple scene graph
  - A model loader
  - Navigation using “unit circle” math
  - Collision detection using AABB
  - Multi-player using ObjectInput/OutputStream
- 9 classes and 1 interface, 1500 lines of code
  - ~900 lines written “manually” using a smart IDE with code completion and refactoring features...
  - Takes less than 50 minutes to write…

For More Information

Resources online

- Duke Bean'Em—http://jsolutions.se/DukeBeanEm
- JOGL—http://jogl.dev.java.net
- JInput—http://jinput.dev.java.net
- Vecmath—http://vecmath.dev.java.net
- XMLBeans—http://xmlbeans.apache.org
- Blender 3D—http://www.blender.org
- JavaGaming.org—http://www.javagaming.org
- Collada—http://www.khronos.org/collada/

Q&A

Write a 3-D Game in the Java™ Programming Language in Less Than 50 Minutes

Erik Hellman <erik.hellman@ibs.se>

Consultant
IBS JavaSolutions
http://jsolutions.se

TS-3073