Java as a Visualisation Platform

Alexander Streit

What this talk is NOT

- A religious debate about Java vs. C++
- A series of benchmarks on Java's performance
- Does not conclude with "A is better than B"

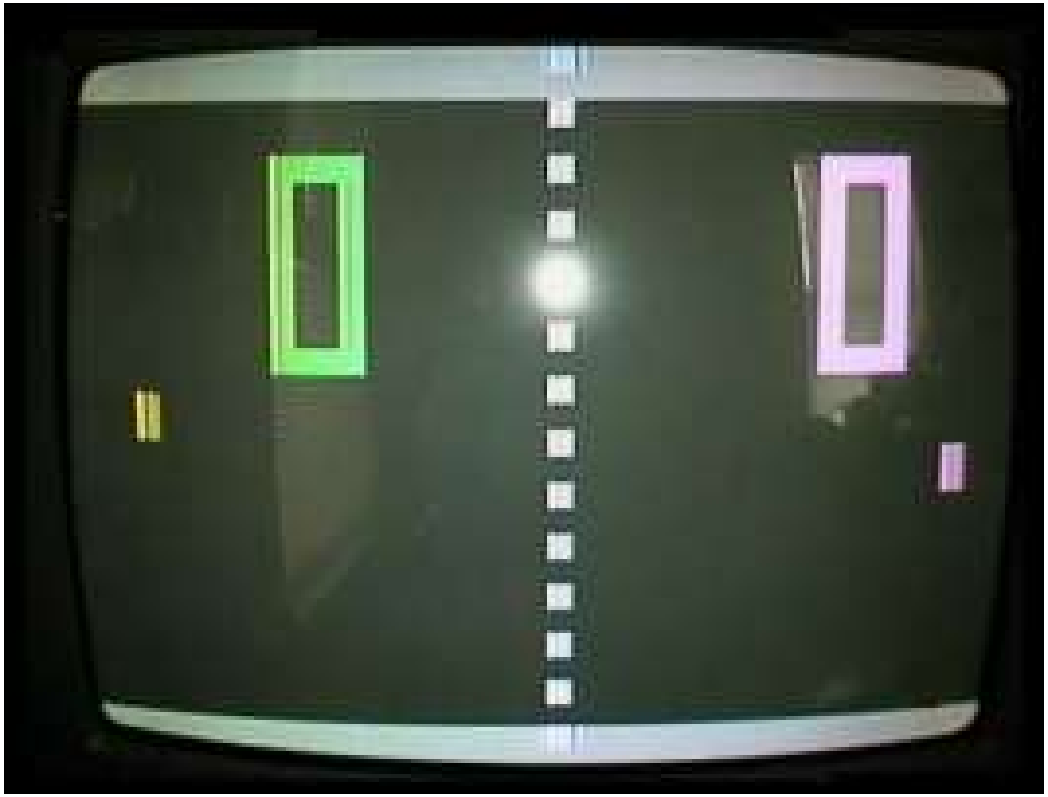
What this talk IS

- (hardware) developments in the field
- issues in graphics
  - how these have been addressed in C++
- What is different under Java?
- a look at various libraries for Java

Graphics

- Traditionally, as fast as fast as fast as possible.
- Most games are optimised like mad.
  - assembly language etc.
  - re-organising data to match cache architecture, etc.
- Ultimately, all about fooling the eye.

1974



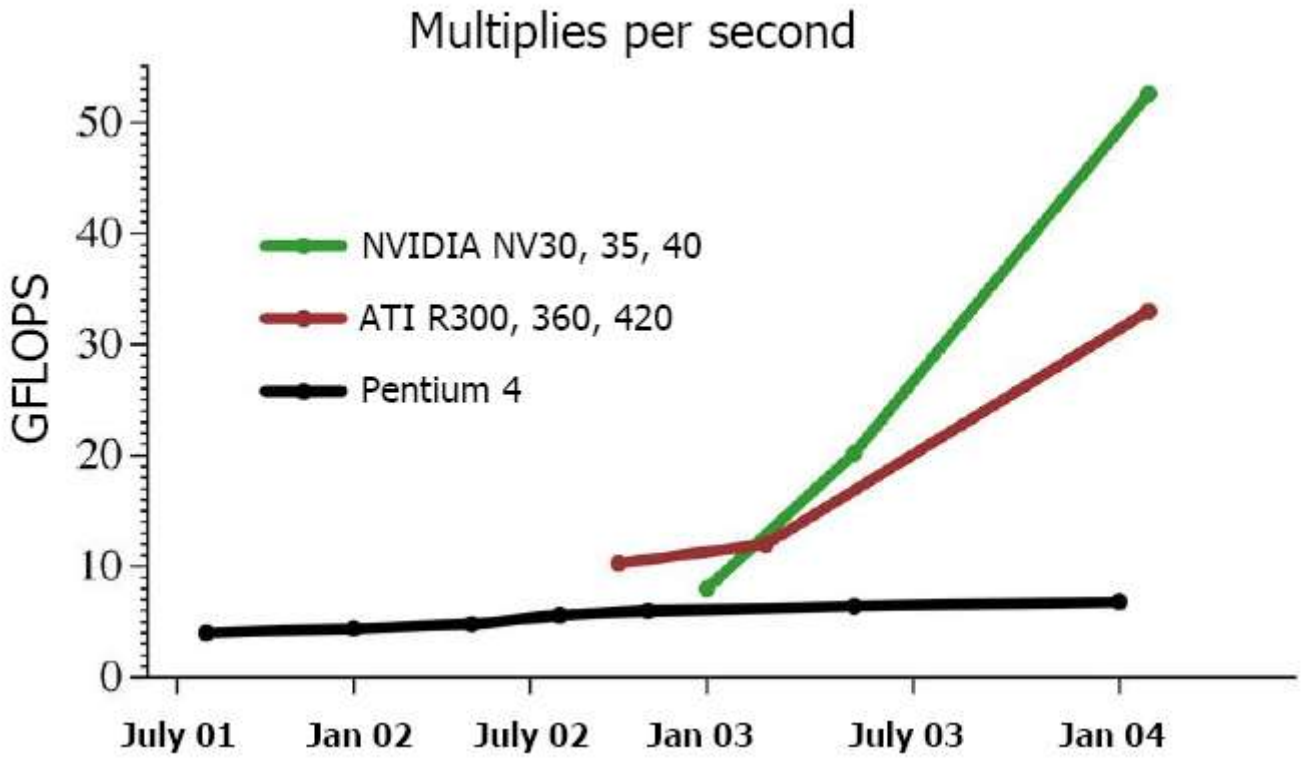
1984


SAGITTARIAN
HI
HI
SCORE
20
20
10
10
10
10
10
BALLS LEFT 5
LO
LO
0050
003775

1994

2004

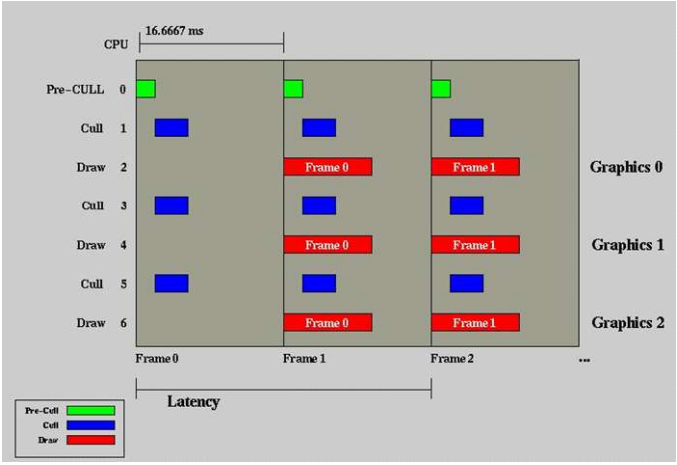
Hardware…

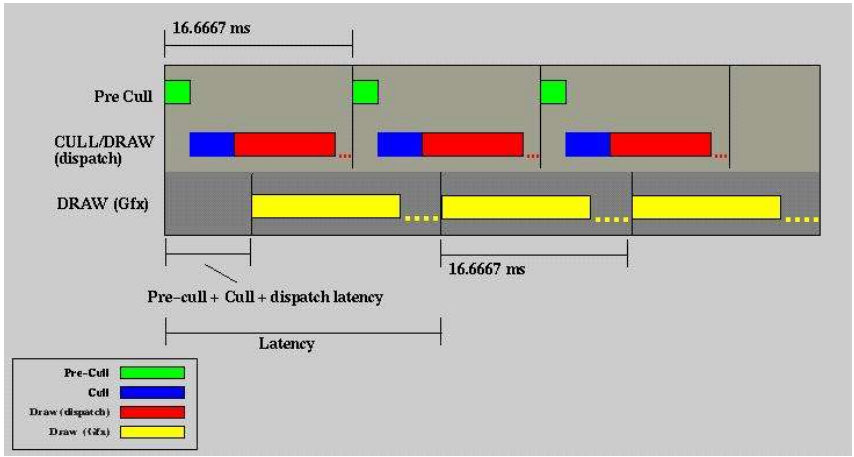


	Instruction Speed	Memory Speed
Intel Pentium 4, 3 GHz	12 GFLOPS	6 GB/sec
NVIDIA GeForce 6800	45 GFLOPS	36 GB/sec

C++: Latency

- Use multi-threading to minimize latency
  - Dispatch and execution done concurrently
  - Pre-cull & cull done before-hand.



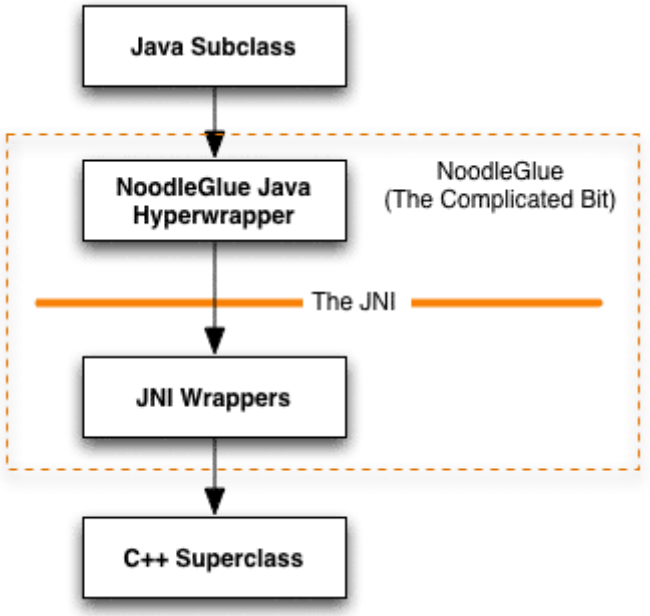


C++: Render Graphs

- A Render Graph is like compiled code
  - It is graphics card ready
  - State sorted
  - Also render bin dependent sort (e.g. α-blend)
- Happens every frame, but frames are coherent, so reuse as much as possible

So, what about Java?

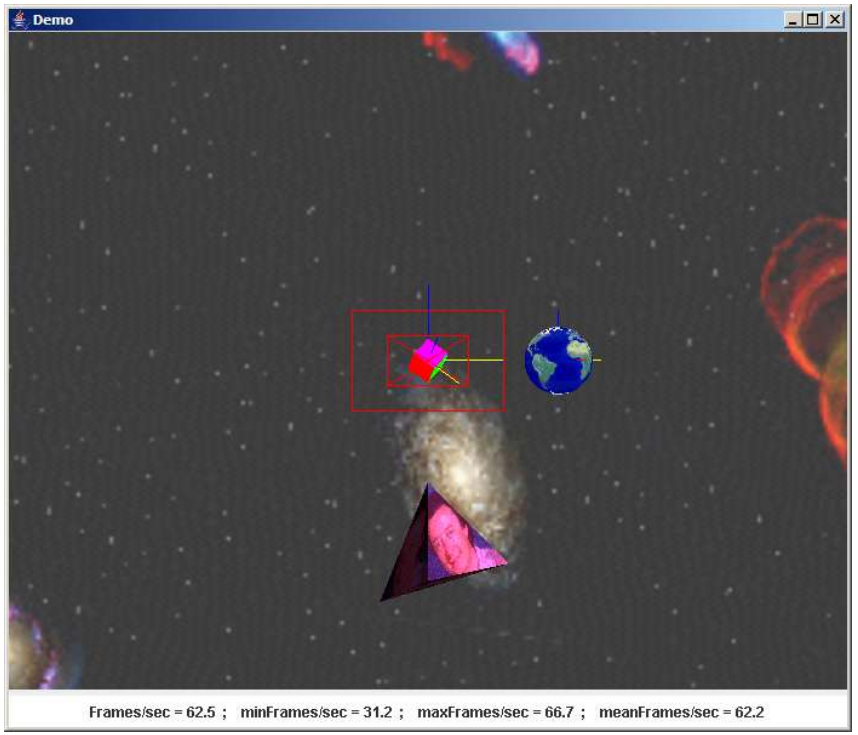
- Java3D
- JOGL
- LWJGL
- JME
- JavaOSG



Java3D

- Scene graph written in Java
- OpenGL, DirectX, Software

- Was canned, now revived.
- GC is an issue



Java3D

```
// Create the BranchGroup brGr1 of the subgraph #1.
brGr1 = new BranchGroup();

// Create and attach the coordinate system to the brGr1.
coordSyst = new CoordSyst(1.0f, 1.0f, 0.0f ,  // Color of the x-axis
                0.0f, 0.0f, 1.0f,  // Color of the y-axis
                1.0f, 0.0f, 0.0f,  // Color of the z-axis
                0.75f);  // Lenght of the 3 axes
brGr1.addChild(coordSyst);

// Background setting for the scene.
newTextureLoader = new NewTextureLoader("Images/Ciel_Out.jpg");
newTextureLoader.setImageObserver(newTextureLoader.getImageObserver());
backGr = new Background(newTextureLoader.getImage());

boundsBackGr = new BoundingSphere(new Point3d(0.0,0.0,0.0), 1000.0);
backGr.setApplicationBounds(boundsBackGr);
brGr1.addChild(backGr);

// A BoundingSphere instance as general bounding region.
boundsGen = new BoundingSphere(new Point3d(0.0,0.0,0.0), 100.0);
```

JOGL

- OpenGL bindings for Java
- NVIDIA's Cg
- OpenGL 2.0 support to come

- Perfect for experimentation

JOGL

```
gl = glDrawable.getGL();

gl.glClear(GL.GL_COLOR_BUFFER_BIT);

gl.glEnable(GL.GL_TEXTURE_2D);
gl.glTexParameteri(GL.GL_TEXTURE_2D, GL.GL_TEXTURE_MIN_FILTER,
        GL.GL_LINEAR);
gl.glTexParameteri(GL.GL_TEXTURE_2D, GL.GL_TEXTURE_MAG_FILTER,
        GL.GL_LINEAR);

gl.glEnable(GL.GL_BLEND);
gl.glBlendFunc(GL.GL_SRC_ALPHA, GL.GL_ONE_MINUS_SRC_ALPHA);

gl.glDisable(GL.GL_LIGHTING);
gl.glColor3f(1f, 1f, 1f);

gl.glMatrixMode(GL.GL_MODELVIEW);
gl.glLoadIdentity();
```

LWJGL

- Abstracts OpenGL, OpenAL, FMOD, Input…
- Made for commercial game development
- Good for games!

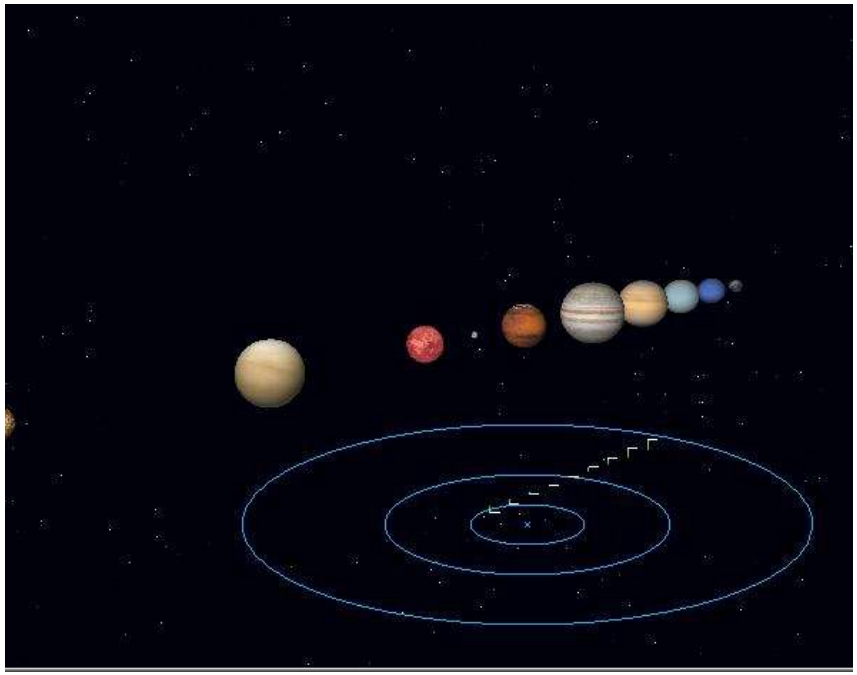
LWJGL

```
// select model view for subsequent transforms
GL11.glMatrixMode(GL11.GL_MODELVIEW);
GL11.glLoadIdentity();

// clear depth buffer and color
GL11.glClear(GL11.GL_COLOR_BUFFER_BIT | GL11.GL_DEPTH_BUFFER_BIT);

// rotate, scale and draw cube
rotation += .3;
GL11.glPushMatrix();
GL11.glLoadIdentity();
GL11.glRotatef(rotation, (float)0, (float)1, (float)0);
GL11.glColor4f(0f, .5f, 1f, 1f);
renderCube();   // draw a cube
GL11.glPopMatrix();

// draw another overlapping cube
GL11.glPushMatrix();
GL11.glLoadIdentity();
GL11.glRotatef(rotation, (float)1, (float)1, (float)1);
GL11.glColor4f(.7f, .5f, 0f, 1f);
renderCube();
```

jME

- Scene graph built on LWJGL (but abstracted)
- Games oriented

- Doesn’t do stereo etc.
- Loads Models
- Provides convenience

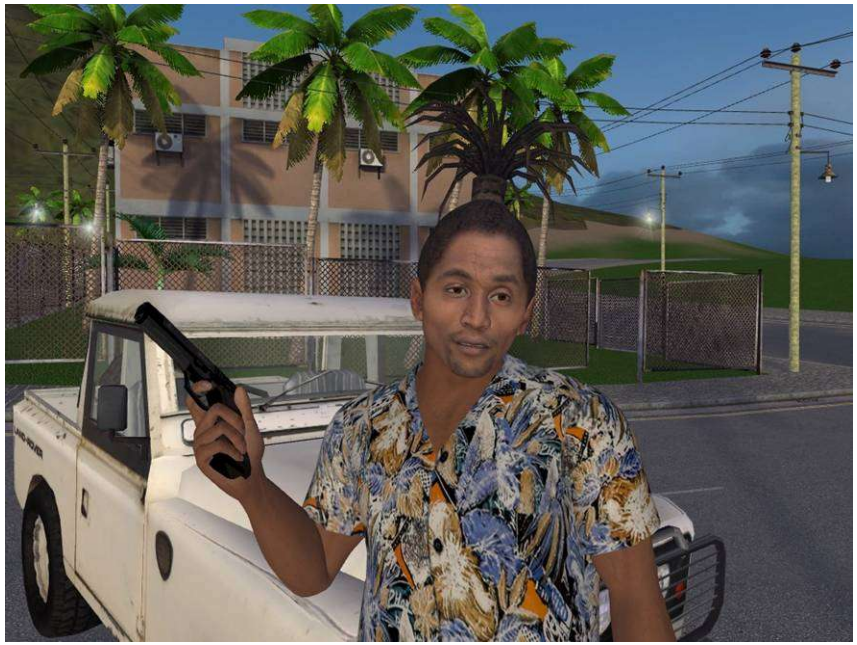


jME

```
Box b=new Box("My Box",new Vector3f(0,0,0),new Vector3f(1,1,1));
// Give the box a bounds object to allow it to be culled
b.setModelBound(new BoundingSphere());
// Calculate the best bounds for the object you gave it
b.updateModelBound();
// Move the box 2 in the y direction up
b.setLocalTranslation(new Vector3f(0,2,0));
// Give the box a solid color of blue.
b.setSolidColor(ColorRGBA.blue);

Sphere s=new Sphere("My sphere",10,10,1f);
// Do bounds for the sphere, but we'll use a BoundingBox this time
s.setModelBound(new BoundingBox());
s.updateModelBound();
```

JavaOSG

- Only in early BETA
- OSG is a high-performance multi-platform scene graph system
- Basically JNI bindings, so same interface as C++ code
- One to watch!



Java - Issues

- Requires some work to ensure data is in correct format
  - Big-endian
  - Uses different color encoding
  - Uses different image data encoding
- Using ByteBuffers etc…

Java - Issues

- High-end requires consistent framerate
  - Military sims & flight sims must guarantee a fixed frame rate!
- Under Java, garbage collection can be an issue
  - Calling the garbage collector all the time does not work (too slow)


My Conclusion

- Using Java for ad-hoc experimentation is great
- Anything that runs mostly on the card is fine
- But we won’t see cutting edge games, yet.