Welcome 😊
• Mobile 3D Graphics API (JSR 184)
3D Mobile APIs for Java ME

• Mobile 3D Graphics API (JSR 184)
• Open GL ES (JSR 239)
Mobile 3D Graphics API; JSR 184

- Java-Specific standard for 3D Graphics.
- JSR 184 is a Java-based library.
- It is an optional package for 3D application development.
Open GL ES; JSR 239

- Provide Java bindings to Open GL ES native 3D graphics library.
- It caters to those developers who are already familiar with OpenGL.
- OpenGL ES is not built for only JAVA ME.
- Can be used also in native code (Symbian OS / C++)

Zülküf Küçükozer
Open GL ES vs. Java ME 3D API

Open GL ES:
- No support for animation and scene management.
- Lack of design efficiency (low level)
- Lots of code even for a simple action.
- Too low level
- Better performance

Java 3D ME:
- Supports a file format (m3g)
- Better design efficiency
- Memory protection which means fewer bugs and better productivity.
- Can be implemented in both low and high level easily.
- Relatively poor performance.
The Basics of JSR 184

• Hardware independent:
  - 3D accelerator is not necessary but it can be used.
  - Java Mobile 3D application can be run in every hardware if JSR 184 is supported.

• Game Design:
  - Game logic in source code plus a m3g file will be enough to build a 3D game in Java ME.

• Floating point hardware needed for calculations. Thus JSR 184 requires version 1.1 of CLDC for its floating-point capability.
The Mobile 3D Graphics API for J2ME is defined in the package `javax.microedition.m3g`, which provides an easy-to-use API for rendering 3D graphics in retained mode and immediate mode.
M3G file for JSR-184

• M3g is a file format for high level design.
• It contains everything (all scheme graph)
• Objects are serialized so that objects can be implemented in source java code.
• Can be compressed.

Zülküf Küçüközer
Tools For 3D JAVA Mobile M3G applications

• Free Tools:

- Blender (www.blender.org)
- M3G exporter plug-in for Blender
  (http://www.nelson-games.de/bl2m3g/default.html )
- Gimp (to design textures) (http://www.gimp.org/)
- ...

Zülfüf Küçüközer
Tools For 3D JAVA Mobile M3G applications

• Non-Free Tools:

- Autodesk 3D Studio Max (http://usa.autodesk.com/)
- Autodesk Maya (http://usa.autodesk.com/)
- M3GExport for Maya (www.m3gexport.com/)
- HI Corporation M3G exporter for 3D Studio Max
  (http://www.mascotcapsule.com/toolkit/m3g/en/index.php)
- ...
Retained and Immediate Mode

• Immediate mode:
  - Mesh objects are created by hardcoding.
  - It is promising for non-complex 3D models.
  - Hard to code for complex textures.
  - Low level

• Retained mode:
  - Mesh objects, models, light, camera and all others are exported to a m3g file via a tool.
  - No need to hardcoding 3D models.
TriangleCanvas.java

import javax.microedition.lcdui.Graphics;
import javax.microedition.lcdui.game.*;
import javax.microedition.m3g.*;

public class TriangleCanvas extends GameCanvas implements Runnable {
    private boolean mRunning = false;
    private Thread mPaintThrd = null;
    private Graphics3D mGraphics3D = Graphics3D.getInstance();
    private Camera mCamera = new Camera();
    private Light mLight = new Light();
    private float mAngle = 0.0f;
    private Transform mTransform = new Transform();
    private Background mBackground = new Background();
    private VertexBuffer mVertexBuffer;
    private IndexBuffer mIndexBuffer;
    private Appearance mAppearance = new Appearance();
    private Material mMaterial = new Material();
    public TriangleCanvas() {
        super(true);
    }
    public void init() {
        short[] vertices = { 0, 0, 0, 3, 0, 0, 0, 3, 0 };
        VertexArray vertexArray = new VertexArray(vertices.length / 3, 3, 2);
        vertexArray.set(0, vertices.length/3, vertices);
        byte[] normals = { 0, 0, 127, 0, 0, 127, 0, 0, 127 };
        VertexArray normalsArray = new VertexArray(normals.length / 3, 3, 1);
        normalsArray.set(0, normals.length/3, normals);
        VertexBuffer verbuf = mVertexBuffer = new VertexBuffer();
        verbuf.setPositions(vertexArray, 1.0f, null);
    }
    public void run() {
        // Runnable implementation
    }
    public void start() {
        mRunning = true;
        mPaintThrd = new Thread(this);
        mPaintThrd.start();
    }
    public void stop() {
        mRunning = false;
        if (mPaintThrd != null) {
            mPaintThrd.interrupt();
            mPaintThrd = null;
        }
    }
    public void paint(Graphics graphics) {
        // Paint implementation
    }
}

Zülküf Küçüközer
TriangleCanvas.java

```java
verbuf.setNormals(normalsArray);
int[] stripLength = { 3 };
mIndexBuffer = new TriangleStripArray( 0, stripLength );

mMaterial.setColor(Material.DIFFUSE, 0xFF0000);
mMaterial.setColor(Material.SPECULAR, 0xFF0000);
mMaterial.setShininess(100.0f);
mAppearance.setMaterial(mMaterial);
mBackground.setColor(0x00ee88);
mCamera.setPerspective( 60.0f, (float)getWidth()/ (float)getHeight(), 1.0f, 1000.0f );
mLight.setColor(0xffffff);
mLight.setIntensity(1.25f);
}
public void start() {
  mRunning = true;
mPaintThrd = new Thread(this);
mPaintThrd.start();
}
public void stop() {
  mRunning = false;
  try{
    mPaintThrd.join();
  } catch (InterruptedException ex){}
}
public void run() {
  Graphics g = getGraphics();
  while(mRunning) {
    if (isShown()) {
      mGraphics3D.bindTarget(g);
      mGraphics3D.clear(mBackground);
    }
  }
```
mTransform.setIdentity();
mTransform.postTranslate(0.0f, 0.0f, 10.0f);
mGraphics3D.setCamera(mCamera, mTransform);
mGraphics3D.resetLights();
mGraphics3D.addLight(mLight, mTransform);
mAngle += 1.0f;
mTransform.setIdentity();
mTransform.postRotate(mAngle, 0, 0, 1.0f);
mGraphics3D.render(mVertexBuffer, mIndexBuffer,
mAppearance, mTransform);
mGraphics3D.releaseTarget();
flushGraphics();
try {Thread.sleep(40); }
catch(InterruptedException ie){
}
// of while
} // of run()
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import java.util.*;
public class TriangleMIDlet extends MIDlet implements CommandListener {
    private Display mDisplay = null;
    private TriangleCanvas mCanvas = null;
    private Command exitCommand = new Command("Exit", Command.ITEM, 1);

    public TriangleMIDlet() {
        super();
        mDisplay = Display.getDisplay(this);
        mCanvas = new TriangleCanvas();
        mCanvas.setCommandListener(this);
        mCanvas.addCommand(exitCommand);
    }

    public void startApp() {
        mCanvas.init();
        mDisplay.setCurrent(mCanvas);
        mCanvas.start();
    }

    public void pauseApp() {} 

    public void destroyApp(boolean unconditional) {
        mCanvas.stop();
    }

    public void commandAction(Command cmd, Displayable disp) {
        if (cmd == exitCommand) {
            try {
                destroyApp(false);
                notifyDestroyed();
            } catch(Exception e) {
                e.printStackTrace();
            }
        }
    }
}

Zülkuf Küçüközer
Class Tree of .M3G

- Object3D
- Graphics3D
- Loader
- Transform
- RayIntersection

- AnimationController
- AnimationTrack
- Appearance
- Background
- CompositingMode
- Fog
- Image2D
- IndexBuffer
- KeyFrameSequence
- Material
- PolygonMode
- VertexArray
- VertexBuffer
- Transformable
- Node
- Texture2D
- Camera
- Group
- Light
- Mesh
- Sprite3D
- MorphingMesh
- SkinnedMesh

Zülküf I
Key Classes

- **Scene graph root node.**
  - **World**

- **Can load .png, m3g vs. files**
  - **Loader**

- **Performs rendering**
  - **Graphics3D**

- **Abstract class. Can be used for controlling objects**
  - **Node**

- **There is only one active camera for one scene graph. Also can be used for collision detection. Like an eye of a parking sensor.**
  - **Camera**
A Morfing Mesh

Array of vertex buffers, one for each morph target. Resultant positions are a weighted blend of the "base mesh" positions in the standard vertex buffer, and the positions in the morph targets.
Morfing Mesh

Vertex morphing mesh

Base

Target 1
eyes closed

Target 2
mouth closed

Animate eyes and mouth independently
Skinned Mesh

A scene graph node that represents a skeletally animated polygon mesh.

Neutral pose, bones at rest

Bone A

Bone B

shared vertex, weights = (0.5, 0.5)

“skin”

non-shared vertex

Bone B rotated 90 degrees

position in A's coordinate system

interpolated position

position in B's coordinate system

Zülküf Küçüközer
Skinned Mesh

*Skeletally animated mesh:*

- Bones are used for rigging. After connecting the bones to the mesh you can make a skinned mesh.
Typical usage of M3G for 3D graphics rendering:

```java
public class MyCanvas extends Canvas {

  Graphics g = getGraphics(); // 2D graphics instance
  Graphics g3d = Graphics3D.getInstance(); // 3D graphics instance

  while (true) {
    g3d.bind(g); // the frame of the rendered 3D scene will be drawn onto the 2D context
      //...
    // Perform setup of your 3D scene to render
    //...

    g3d.render(); // render the scene

    mGraphics.releaseTarget(); // flush rendered image to the 2D Graphics
    flushGraphics(); // flush the drawing and update the visible components of the rendered frame
  }
}
```
All you need to know well about 3D applications are:

• Rotation
  - Rotating an 3D object in its world or local coordinates around at given coordinate axis.

• Translation
  - Translating an 3D object in world coordinates.
    (Moving an transformable 3D object)

• Scaling
  - Scaling an 3D object in given direction.
Transform Examples

The sample cube

The sample cube: a) Rotated, b) Translated, and c) Scaled
Light Sources

A scene graph node that represents different kinds of light sources.

Light sources are used to determine the color of each object according to its Material attributes, as described in more detail in the Material class documentation.

Light source types:

Four types of light sources are available. In the order of increasing computational complexity, these are the ambient light, directional light, omnidirectional light and spot light.

Cube lit with a) omnidirectional, b) spot, c) ambient, and d) directional light
Loading an M3G file

Object3D[] roots = null;
World iWorld = null;
try {
    // Load the m3g file
    // The loader will return all root level objects of the file,
    // which are not referenced by any other objects.

    roots = Loader.load("/monkey.m3g");
}
catch (IOExceptionex) {
    // couldn't open the file, or invalid data in the file
    ex.printStackTrace();
}

// Generally the world node is the only and first node.
// Save the world node to an instance variable
for (int i = 0; i < roots.length; ++i) {
    if (roots[i] instanceof World) iWorld = (World) roots[i];
}
The content of a M3G File
The content of a M3G File

A world root node without an id number

A camera in the world scheme with id number 33

A mesh object in the world scheme with id number 30

A mesh object that can be seen with its contents of polygon vertices, normals, appearance properties and textures.

Zülküf Küçüközer
Animation in M3G

• An animation may include:
  - Key or keys
  - Property values
  - Time sequence
  - Rotation (it is Rot in Blender)
  - Translation (it is Loc in Blender)
  - Scale (it is Scale in Blender)
How to get objects the in M3G file before performing an action

Node monkey = (Node) iWorld.find(ID_MONKEY);
// gets the object with the given id

// there other methods to get same object

Mesh monkey = (Mesh) iWorld.find(ID_MONKEY);
// gets the mes object with the given id
Controlling the objects with key events

private static final int ID_MONKEY = 1;

// Find searches through all children of this node
Node monkey = (Node) iWorld.find(ID_MONKEY);

switch (getKeyStates())
{
    case LEFT_PRESSED:
        // rotate the monkey about z in world coordinates
        monkey.postRotate(0.3f, 0.0f, 0.0f, 1.0f);

        // rotate the monkey about z in local coordinates
        Transform tra = new Transform();
        monkey.getTransform(tra);
        tra.postRotate(-0.3f, 0.0f, 0.0f, 1.0f);
        monkey.setTransform(tra);

        break;
}

Zülküf Küçüközer
void run()
{
    // Get the singleton Graphics3D instance that is associated with this midlet.
    Graphics3D g3d = Graphics3D.getInstance();
    // Measure the time that has passed since the prev. frame
    long start, elapsed = 0;
    int time = 0;

    while (iIsActive)
    {
        start = System.currentTimeMillis();
        try {
            // Bind the 3D graphics context to the given MIDP Graphics object.
            g3d.bindTarget(getGraphics());
            // Update the world [...] 
            iWorld.animate(time);
            // Animate the world
            // Render the view from the active camera
            g3d.render(iWorld);
        } finally {
            // Release the graphics context
            g3d.releaseTarget();
        }
        flushGraphics();
        // Flush the rendered image to the screen
        // Give other threads a chance to run
        Thread.sleep(20);
        // This sample omits try and catch
        elapsed = System.currentTimeMillis() - start;
        time += elapsed; // Time that has passed since the last frame
    }
}
Animation in M3G

**keyframe**: time and property values

**sequence**: different types of sequences (linear, nonlinear)
- can be looping
- can include more than one keyframes

Zülküf Küçükozer
BLENDER INTERFACE

3D VIEW

SAMPLE CUBE OBJECT

SAMPLE LIGHT OBJECT

SAMPLE CAMERA OBJECT

PRIMARY CONTROL PANEL BUTTONS

SECONDARY CONTROL PANEL BUTTONS

CONTROL PANELS

BUTTONS WINDOW
Using M3G Exporter for Blender

1) To give an ID number for any object in Blender:

- Press Object Button
- In Objects and Links pane change the Datablock name like this.

Camera == Camera#1

2) File ==> export ==> M3G Export

3) That's all.
Transformation example in Blender

1. Select the cube
2. press r button.
3. rotate the by dragging the mouse.

Note: Rotation is done around z axis. You can see it if you look at the view coordinates of the cube below.
Transformation example in Blender

1. Select cube
2. Press editing button
1. Change 3D cursor location to the left of the cube.
2. Then press Center Cursor button. Now rotate the cube again.
The center of the cube is changed now. So we can rotate the cube differently. This is very crucial in all 3D Game Applications!!!
_press i key
select Loc
Animating objects in Blender-2
Animating objects in Blender-3

1. Change time bar by dragging to right.

2. Drag the cube to up.

3. Press i key.
Animating objects in Blender-4

the time sequence with different keys are constructed as you see it at above graph.
Textures in Blender

- Textures can be exported if you use UV Mapping in Blender.
- Texture or textures are mapped to texture coordinates.
- Use UV mapping.
- Select TextFace

Select text face
Select Shadeless if you want your object to be shadeless
An example of texturing in Blender
Collision Detection

javax.microedition.m3g
Class RayIntersection

A RayIntersection object is filled in by the pick methods in Group. A camera can be used for collision detection.

//checkGroup is a group object that includes a 3D mesh inside.
//checkCamera is a camera object

RayIntersection ri = new RayIntersection();

if(checkGroup.pick(9, 0.5f, 0.5f, checkCamera, ri )
   System.out.println( "intersection at distance: " +ri.getDistance() );
else
   System.out.println("No intersection");
Collision Detection

Intersection!

No Intersection!
My Project

• I developed a car game.
• Collect the check points to finish the game.
• Keys are: up, down, left and right.
• Do not hit the wall.
My Project
Sources:

- http://developers.sun.com/mobility/apis/articles/3dgraphics/
- www.squidoo.com
- Beginnig J2ME: From Novice to Professional 3rd
  - http://blogs.forum.nokia.com/blog/andreas-jakls-forum-nokia-blog
  - www.blender.org
Questions?