Mobile Application Programming: Android

OpenGL Sprites
Activities

- Apps are composed of activities
- Activities are self-contained tasks made up of one screen-full of information
- Activities start one another and are destroyed commonly
- Apps can use activities belonging to another app
OpenGL ES

- C-Based Performance-Oriented Graphics Library
- Wrapper libraries provided for Java, C#, etc.
- Produces 2D images from 2D or 3D geometric data
- Mobile version of OpenGL
  - Includes nearly all OpenGL functionality
  - Removes seldom-used or legacy features
  - Used by non-mobile platforms also (eg. Playstation 3)
Data read from Scene and OBJ files

OpenGL ES Primitive Processing ➔ Vertex Shader ➔ OpenGL ES Rasterizer

Fragments resulting from rasterization ➔ Frame Buffer

Fragment Shader ➔ OpenGL ES Fragment Processing
Shaders

attribute vec4 position;
attribute vec2 textureCoordinate;

uniform mat4 modelView;

varying lowp vec2 textureCoordinateVarying;

void main()
{
  gl_Position = modelView * position;
  textureCoordinateVarying = textureCoordinate;
}

uniform sampler2D textureUnit;

varying lowp vec2 textureCoordinateVarying;

void main()
{
  gl_FragColor = texture2D(textureUnit, textureCoordinateVarying);
}
Texture Mapping
Turning Texturing On/Off

- `glEnable(GL_TEXTURE_2D);`
- `glEnableVertexAttribArray(TextureCoordAttribLocation);`
Texture Coordinates

(0.15, 0.1)

(0.6, 0.9)

(0.6, 0.2)

(0, 0)

(1, 0)

(0, 1)
Loading a Texture

- Give shaders texture coordinates and sampler
- Load texture into a Bitmap object
- Allocate texture number with `glGenTextures`
- Bind texture to hardware with `glBindTexture`
- Copy texture data onto hardware with `glTexImage2D` and a `ByteBuffer` or use `GLUtils.texImage2D`
- Set texture filters with `glTexParameteri` *(required!)*
Swapping Textures

- Many textures can be loaded at one time
- Switch among loaded textures by calling `glBindTexture`
- Textures can be unloaded if there is insufficient memory

- Calling `glBindTexture` reloads the data automatically if it is not video memory resident
Alpha Blending

```cpp
# Enable blending
glEnable(GL_BLEND);

# Set blending function
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
```
Sprites
Sprites

- Built by texturing a quad (2 triangles)
  - Geometry
    - Translation, Rotation, Scale
  - Texture
    - Bound Texture
    - Texture Coordinates
    - (Optional) Texture Matrix
Sprite - One Implementation

- Sprite
- Position (x,y)
- Animation
- Current Tile List & Animation Time
- Animation
- Texture
- Tile Lists (2D)
Transformations

\[ x_f = x_o \cdot S_x \]
\[ y_f = y_o \cdot S_y \]

\[ x_f = x_i \cdot \cos\theta - y_i \cdot \sin\theta \]
\[ y_f = x_i \cdot \sin\theta + y_i \cdot \cos\theta \]

All Points
Transformations

\[ x_f = x_o \cdot S_x \]
\[ y_f = y_o \cdot S_y \]

\[ x_f = x_o + t_x \]
\[ y_f = y_o + t_y \]

\[ x_f = x_i \cdot \cos \theta - y_i \cdot \sin \theta \]
\[ y_f = x_i \cdot \sin \theta + y_i \cdot \cos \theta \]
Transformations

\[
\begin{bmatrix}
1 & 0 & t_x \\
0 & 1 & t_y \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
1
\end{bmatrix}
\]

\[
\begin{bmatrix}
\cos \theta & -\sin \theta & 0 \\
\sin \theta & \cos \theta & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
1
\end{bmatrix}
\]

All Points

\[
\begin{bmatrix}
s_x & 0 & 0 \\
0 & s_y & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
1
\end{bmatrix}
\]
Matrix Composition

2.1 X
-3.6 Y
0.8 Z
Translate

PI/3 Rotate

2x Scale

Combined Matrix That Does All 3 Ops!
Either This 10,000 Times

Notice the ordering!

And This 10,000 Times