Mobile Application Programming

OpenGL ES Introduction
OpenGL ES

- C-Based **Performance-Oriented** Graphics Library
- **Wrapper libraries** provided for Swift, 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 4)
Hardware Acceleration
Hardware Acceleration
Hardware Acceleration

2 Processors
Hardware Acceleration

Apple A7

2 Processors

128 Processors!

Image Source: Chipworks
Hardware Acceleration

Apple A7

- 64-bit
- Dual-Core
- Cyclone
- 1.4 GHz
- ARM v8-A
- L1 64+64KB
- L2 1MB
- L3 4MB
- PowerVR G6430 (4x)

Apple A6

- 32-bit
- Dual-Core
- Swift
- 1.3 GHz
- ARM v7
- L1 32+32KB
- L2 1MB
- L3 N/A
- PowerVR SGX543 (3x)

Image Source: Chipworks
# Comparison of Apple GPUs

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>iPad 2/iPhone 4S</td>
<td>-</td>
<td>iPad 5</td>
<td>iPhone 5</td>
<td>iPad 3</td>
<td>-</td>
<td>-</td>
<td>iPad 4</td>
<td>iPhone 5s</td>
</tr>
<tr>
<td>SIMD Name</td>
<td>USSE2</td>
<td>USSE2</td>
<td>USSE2</td>
<td>USSE2</td>
<td>USSE2</td>
<td>USSE2</td>
<td>USSE2</td>
<td>USC</td>
</tr>
<tr>
<td># of SIMDs</td>
<td>4</td>
<td>8</td>
<td>12</td>
<td>16</td>
<td>8</td>
<td>16</td>
<td>32</td>
<td>4</td>
</tr>
<tr>
<td>MADs per SIMD</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>32</td>
</tr>
<tr>
<td>Total MADs</td>
<td>16</td>
<td>32</td>
<td>48</td>
<td>64</td>
<td>32</td>
<td>64</td>
<td>128</td>
<td>128</td>
</tr>
<tr>
<td>GFLOPS @ 300MHz</td>
<td>9.6 GFLOPS</td>
<td>19.2 GFLOPS</td>
<td>28.8 GFLOPS</td>
<td>38.4 GFLOPS</td>
<td>19.2 GFLOPS</td>
<td>38.4 GFLOPS</td>
<td>76.8 GFLOPS</td>
<td>76.8 GFLOPS</td>
</tr>
</tbody>
</table>

www.anantech.com/show/7335/the-iphone-5s-review/7
OpenGL Environment

- UIWindow
- Root VC - GLKViewController
  - GLKView
    - Vertex Shader
    - Fragment Shader
  - Program
    - Uniform Variables
    - Attribute Arrays
Data read from Scene and OBJ files

OpenGL ES Primitive Processing → Vertex Shader → OpenGL ES Rasterizer

Fragments resulting from rasterization

Fragment Shader → OpenGL ES Fragment Processing → Frame Buffer
Data read from Scene and OBJ files

1. OpenGL ES Primitive Processing
2. Vertex Shader
3. OpenGL ES Rasterizer
4. Fragment Shader
5. OpenGL ES Fragment Processing
6. Frame Buffer

Fragments resulting from rasterization
Barycentric Coordinates

\[ \lambda_1 + \lambda_2 + \lambda_3 = 1 \]
and
\[ 0 \leq \lambda_x \leq 1 \]

\[ P = \lambda_1 A + \lambda_2 B + \lambda_3 C \]
Primitives

GL_POINTS

GL_LINES

GL_LINE_STRIP

GL_LINE_LOOP

GL_TRIANGLES

GL_TRIANGLE_STRIP

GL_TRIANGLE_FAN
Vertex Shader

- Takes in *vertex* data and *modifies* it before using it to draw primitives (particularly triangles)
- Each vertex is *processed separately* from others (in parallel)
- Allows you to *position and orient objects* in the scene
- E.g. Load a monster mesh in spread-eagle stance, move him in the world, and position his arms, legs, feet, so he is running
Vertex Shader

```glsl
attribute vec2 position;
uniform vec2 translate;
void main()
{
    gl_Position = vec4(position.x + translate.x, position.y + translate.y, 0.0, 1.0);
}
```
Data read from Scene and OBJ files

OpenGL ES Primitive Processing → Vertex Shader → OpenGL ES Rasterizer

Fragments resulting from rasterization

Fragment Shader → OpenGL ES Fragment Processing

Frame Buffer
Rasterization
Rasterization
Fragment Shader

- Run once for each fragment (or pixel) that a triangle covers in the output pixel buffer to decide its color.
- Each fragment is processed separately from others (again, in parallel).
- Allows you to color pixels to achieve effects (shading).
- E.g. Draw your running monster, but cover him in green skin. Also make it look like the sun is shining on him.
```
void main()
{
    gl_FragColor = vec4(0.0, 1.0, 0.0, 1.0);
}
```
OpenGL Debugging
OpenGL Debugging
A Little Math - Vectors

- Vectors
- Addition & Subtraction
- Scalar Multiplication
- Dot & Cross Product
- Magnitude & Normalization
A Little Math - Matrices

- Matrices
  - Concatenation
  - Vector Multiplication
- Orthographic Transform
- Perspective Transform
- Viewport Transform