Going Forward

Final Project Lab

Additional Lecture Topics
Going Forward

https://www.surveymonkey.com/r/CYKD9BN
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
Fragments

- Acts like a sub-activity
- Attached and removed from an activity using the FragmentManager
- Attachment or removal of many fragments with FragmentTransaction
- Lifecycle tied to parent activity
- Adds onAttach / onDetach and onCreateView / onDestroyView
Hardware Acceleration
Hardware Acceleration
Hardware Acceleration

2 Processors
Hardware Acceleration
Hardware Acceleration

192 Processors
Hardware Acceleration
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)
OpenGL Environment

- `android.opengl.GLSurfaceView`
- `GLSurfaceView.Renderer`
- `GLES20` (C Library Wrapper)
  - Program
    - Vertex Shader
    - Fragment Shader
    - Uniform Variables
    - Attribute Arrays
OpenGL ES 1 vs ES 2

- **Fixed-Function Pipeline vs. Programmable Pipeline**

- **ES1** has only fixed processes, manipulating geometry and generating fragments in a standardized process.

- **ES2** has some fixed processes and 2 programmable processes for the data input into OpenGL.

- **ES2** is much simpler than ES1, but requires knowledge of the OpenGL Shader Language.
Data read from Scene and OBJ files

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

OpenGL ES Primitive Processing → Vertex Shader → OpenGL ES Rasterizer

Fragments resulting from rasterization → Frame Buffer

Fragment Shader → OpenGL ES Fragment Processing
**Vertex Shader**

- Receives a *vertex* from OpenGL after minimal processing
- **Modifies** incoming vertex in some way using *uniform variables* where needed
- Outputs the vertex
- May also output *additional data* for the *fragment shader* to use
Data read from Scene and OBJ files

1. OpenGL ES Primitive Processing
2. Vertex Shader
3. OpenGL ES Rasterizer

Fragments resulting from rasterization

4. Fragment Shader
5. OpenGL ES Fragment Processing

Frame Buffer
Fragment Shader

- Receives a fragment from OpenGL resulting from rasterizing a primitive
- **Chooses a color** for the fragment based on data given by vertex shader and uniform variables
- **Outputs** the fragment color
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