Gameloft 3D Training
Day 1

thuy.vuthiminh@gameloft.com

GAMELOFT

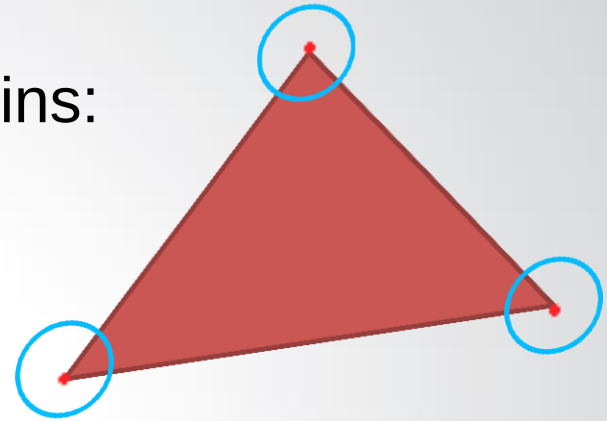
Contents

1. 3D Basic concepts
2. Opengles2.0 begining
3. Texture
4. Practice

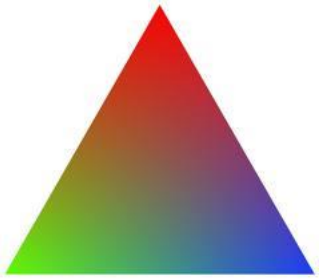
Basic Concepts: Vertex

Vertex

- A vertex is a point in 3D space, contains:
  - Position
  - Color
  - Texture coordinates
  - …



Red color at 3 vertices



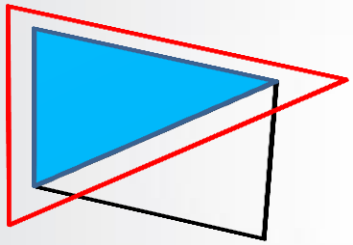
Each vertex is Red, Green, Blue

Triangle with texture

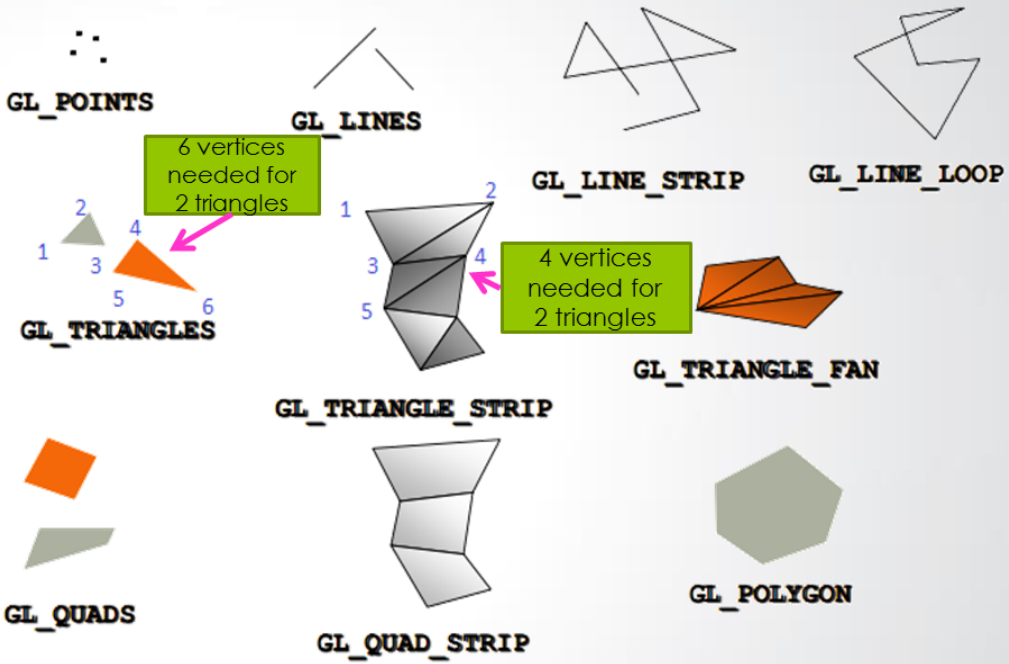
Basic Concepts: Primitives

Triangle

- Defined by 3 vertices



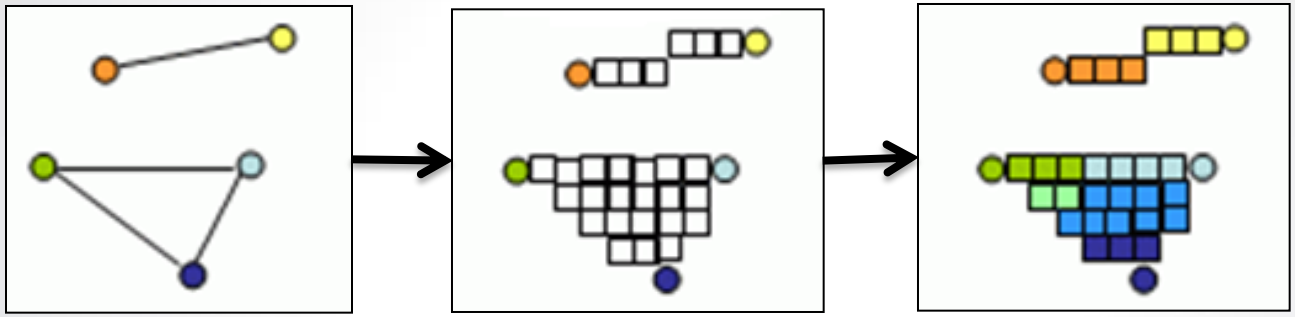
Primitives



GAMELOFT

Basic Concepts: Fragment

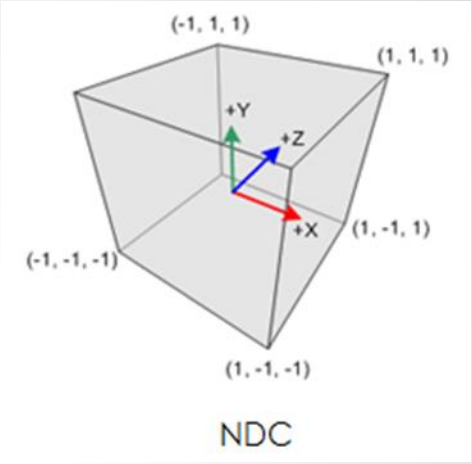
- Fragments are an intermediate between vertices and pixels.



Basic Concepts: NDC

Normal Device Coordination

- 3D point in clip coordinates is mapped to a cube with *left-handed* coordinate system.
- In the bounds of [-1,-1,-1] and [1, 1, 1].
- Everything outside that bound will be clipped.



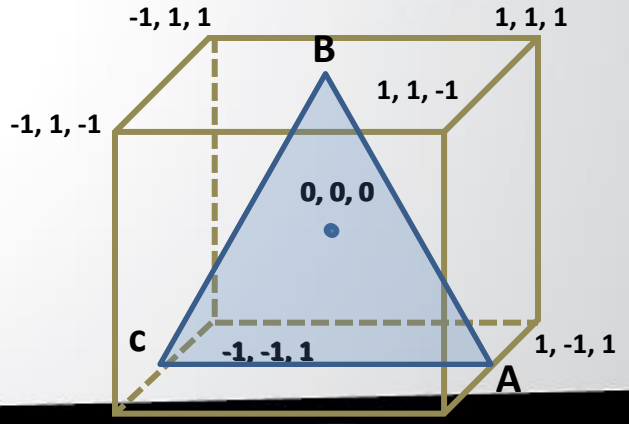
NDC

Ex: The triangle is defined by the points

A (1.0, -1.0, 0.0),

B (0.0, 1.0, 0.0),

C (-1.0, -1.0, 0.0)



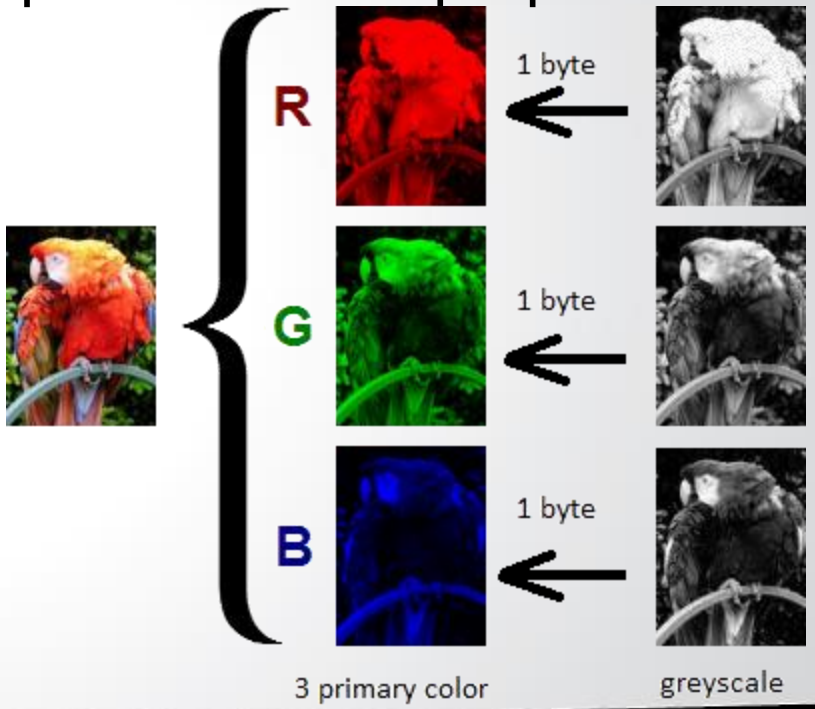
Basic Concepts: Depth-buffer

The Z buffer (depth buffer)

- Contains per-pixel floating-point data for the z depth of each pixel rendered
- Vary size value from 8 → 32 bit

Basic Concepts: Color channel

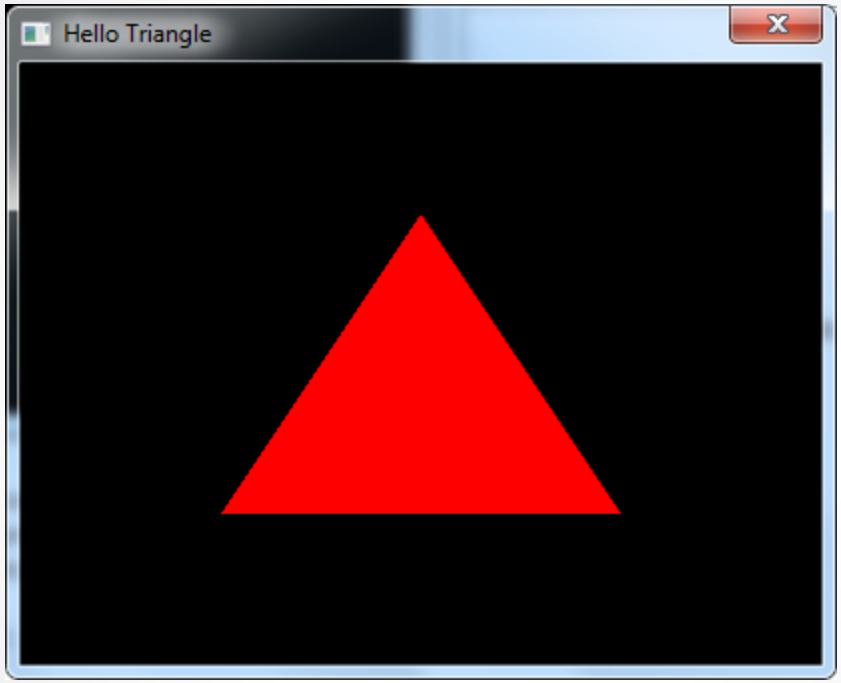
- Each pixel is made of combinations of primary color called Color Channel included: Red, Green, Blue, Alpha (RGBA) (A is optional)
- More higher value of A channel, pixel is more opaque
  - **Color value in GLSL ranging from [0.0, 1.0]**
  - **Example: for red color vec4(1.0, 0.0, 0.0, 1.0)**



GAMELOFT

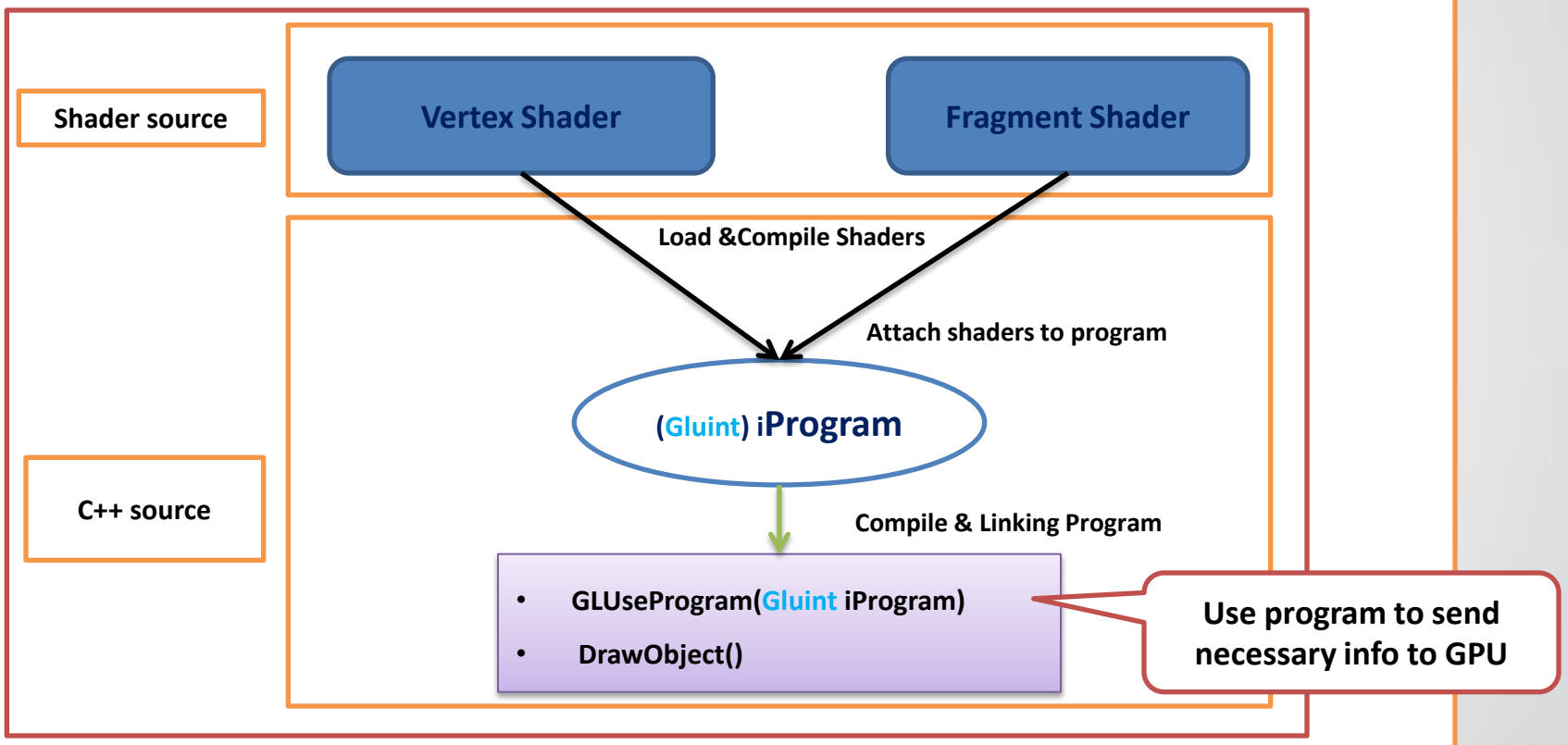
Opengles2.0: Triangle rendering

- How to render a triangle into screen as following?



Opengles2.0: **Shader calling structure**

- Diagram



Opengles2.0: **Triangle Rendering step by step**

- **Step 1: Declare vertices**
- **Step 2: Send vertices to GPU by VBO and use a variable to handle it**
- **Step 3: Use a shader to receive and process these vertices; and use a PROGRAM to handle it.**

Vertex shader	Fragment shader
attribute vec4 a_position; void main() { gl_Position = a_position; }	precision lowp float; void main() { gl_FragColor = vec4(1.0, 0.0, 0.0, 1.0); }

Opengles2.0: **Triangle Rendering step by step**

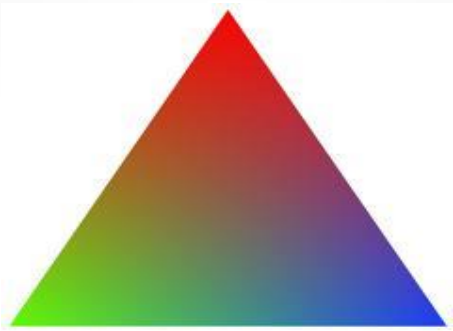
- **Step 4: Create the connection between vertices in C++ and variables in shaders (verticesData vs attribute vec4 a_position)**
- **Step 5: Rendering into screen**

```
glDrawArrays(GL_TRIANGLES, 0, 3);

glBindBuffer(GL_ARRAY_BUFFER, 0);
```

Opengles2.0: Practice 1

- **How to render this gradient triangle?**



Each vertex is Red, Green, Blue

Opengles2.0: **Practice 1 (cont.)**

- **Pseudo:**

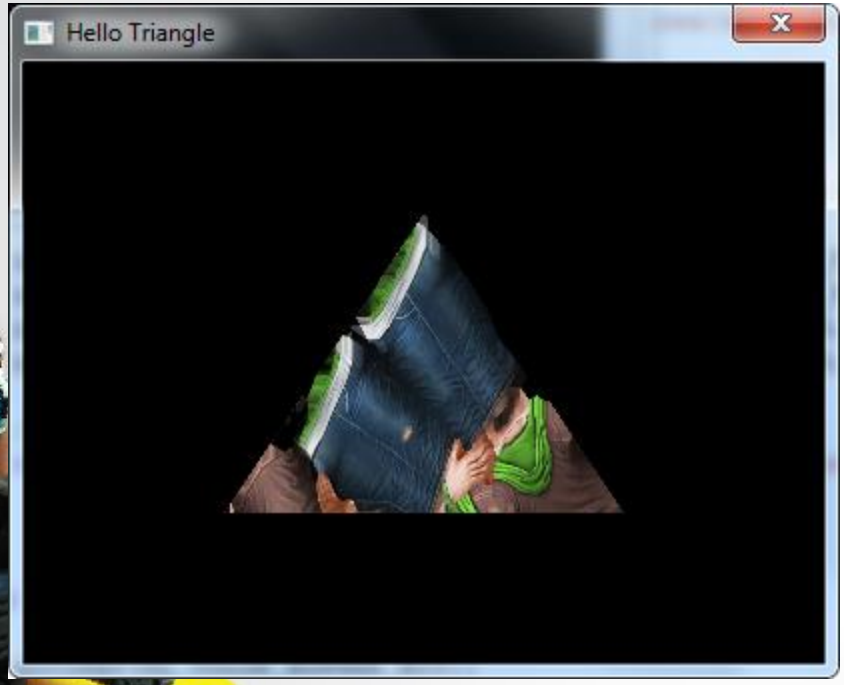
When shaders are compiled, the varying, attributes, uniforms or any other unused local variable will be removed

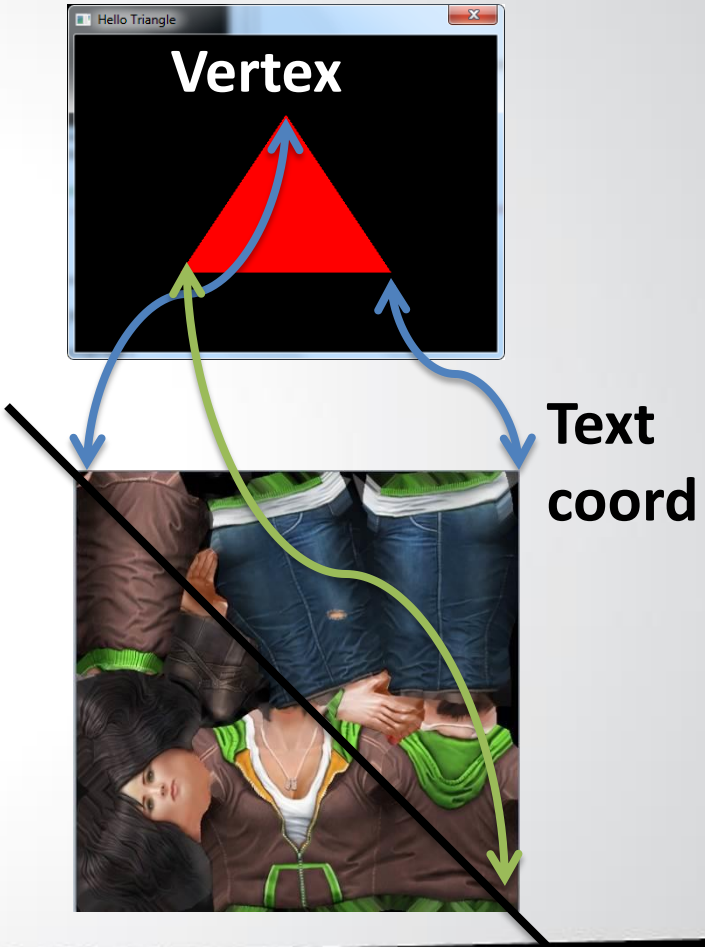
Vertex shader	Fragment shader
attribute vec4 a_position; attribute vec4 a_color; varying vec4 v_color; void main() { *//gl_Position must be set on* *//every vertex shader* gl_Position = a_position; v_color = a_color; }	precision lowp float; varying vec4 v_color; void main() { *//gl_FragColor must be set on* *//every vertex shader* gl_FragColor = v_color ; *//gl_FragColor = vec4(1.0, 0.0, 0.0, 1.0);* }

Why need precision?

Texture: Concept

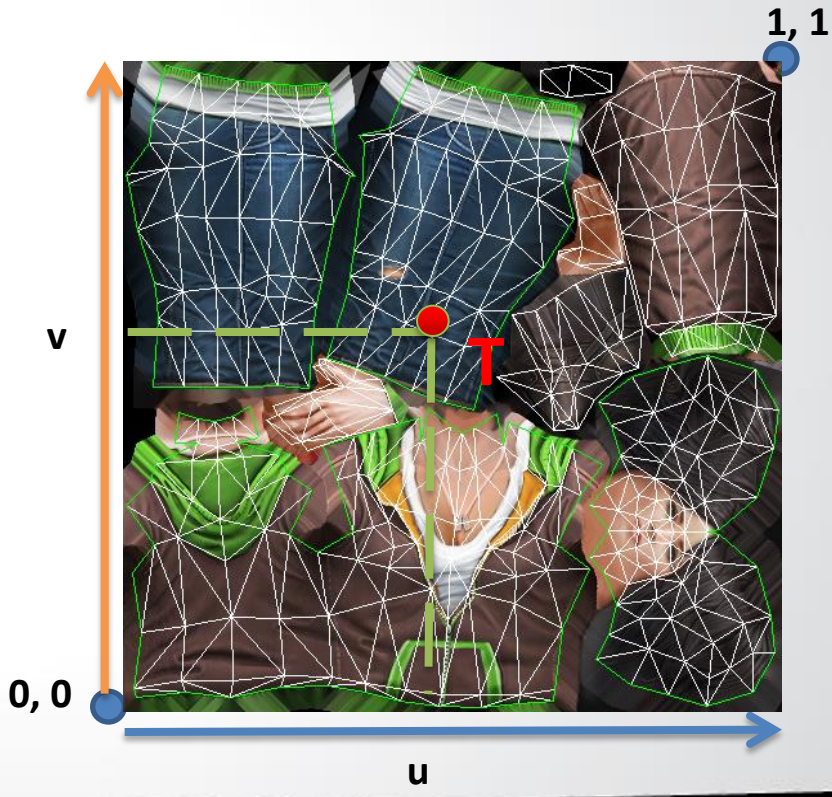
- How to render this triangle?





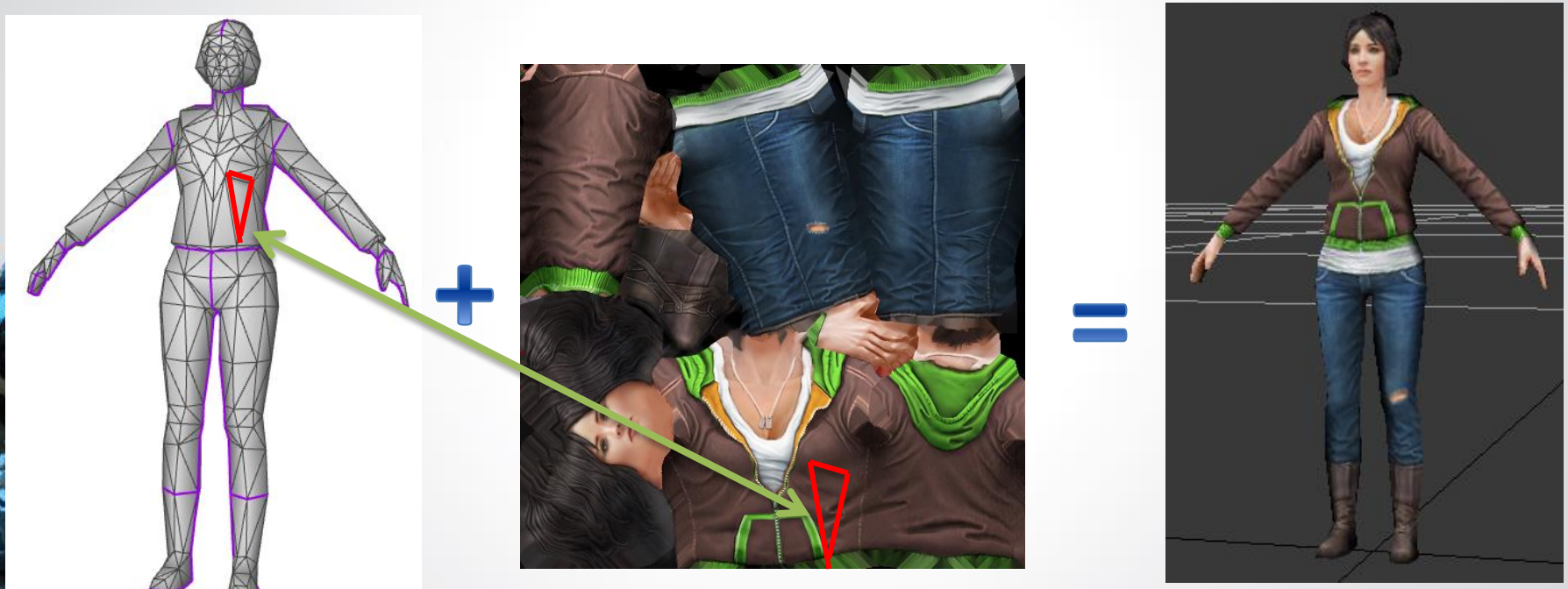
Texture: Text coordinate & Texel

- UV or texture coordinate:
  - ✓ An attribute to describe the position of that vertex on the image
    - **Texel is a pixel on the texture.**
    - **T is a texel:**
      - ✓ **The coordinate of T on the image**
      - ✓ **Defined by (u, v)**
      - ✓ **The u, v is in [0, 1] range**



Textures

- Texture is 2D Image applied on a 3D Object.
- Each primitive on the 3D object will be map to a 2D Image



Texture: **Steps to render with textures**

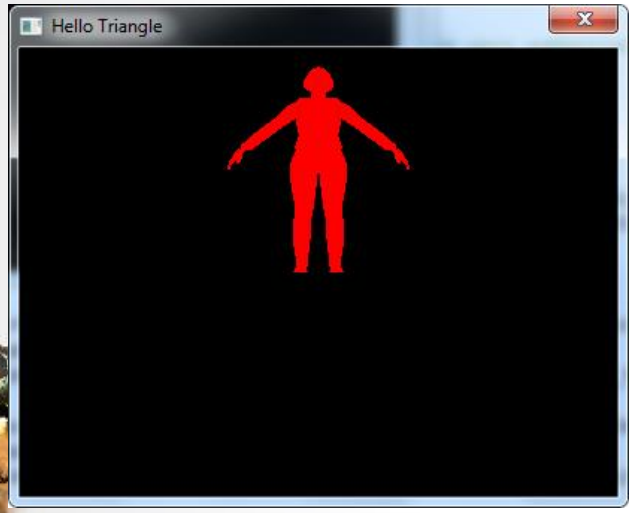
- **Step 1: Declare texcoord**
- **Step 2: Send texcoord to GPU by VBO and use a variable to handle it**
- **Step 3: Modify shader to receive and process these texcoords.**

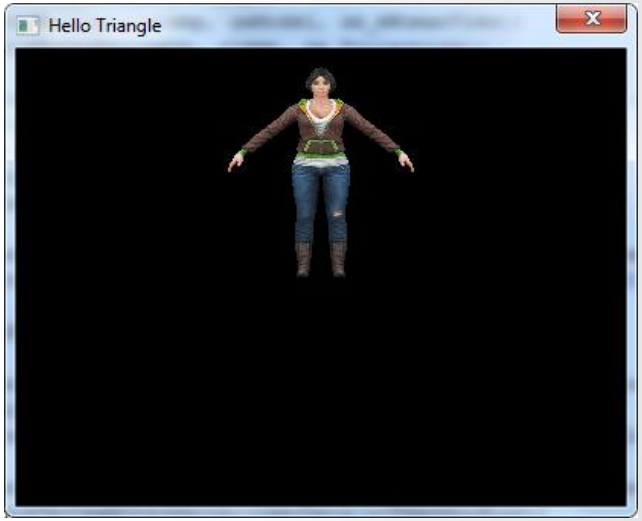
Vertex shader	Fragment shader
attribute vec4 a_position; attribute vec2 a_uv; varying vec2 v_uv; void main() { gl_Position = a_position; v_uv = a_uv; }	precision lowp float; uniform sampler2D u_texture; varying vec2 v_uv; void main() { gl_FragColor = texture2D(u_texture, v_uv); }

- **Step 4: Create the connection between texcoords in C++ and shaders variables (TextCoord vs attribute vec2 a_uv)**
- **Step 5: Rendering into screen**

Practice 2

1. **Load model to render a girl.**
2. **Load texture to that girl.**





Practice 2: Hint

- **NFG file format**
- **Use indices array with elements VBO and glDrawElements**

Thank you!

GAMELOFT