DERIVCO

Shaders

ESTONIA

Basic principles of shaders with OPENGL ES 2.0 in WebGL 1.0

Daniel Priori – Derivco Estonia

DERIVCO

Shaders

BASIC PRINCIPLES OF SHADERS WITH OPENGL ES 2.0 IN WEBGL 1.0

Download the package: **https://goo.gl/vxRB9X**

- Presentation
  - Basics of Shaders and some things in particular for OpenGL ES 2.0
- Break
  - ~15 minutes
- Workshop (let’s code something!)
  - We will focus on create direct shader programs without adding code for shader compilation steps (WebGL steps. Because PIXI/ThreeJS or other Engines will do this for us ☺).
- Cherry

Shaders

BASIC PRINCIPLES OF SHADERS WITH OPENGL ES 2.0 IN WEBGL 1.0

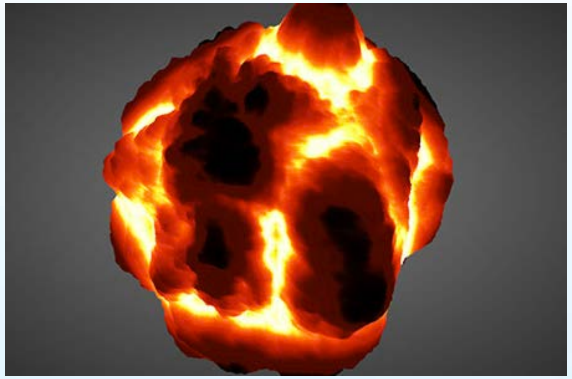
- So, what exactly are shaders?
  - In OpenGL ES 2.0 (for WebGL 1.0) are a form of pairs of functions with the purpose of change and rasterizing (based on the code you supply) the points, lines, triangles and colors.

Shaders

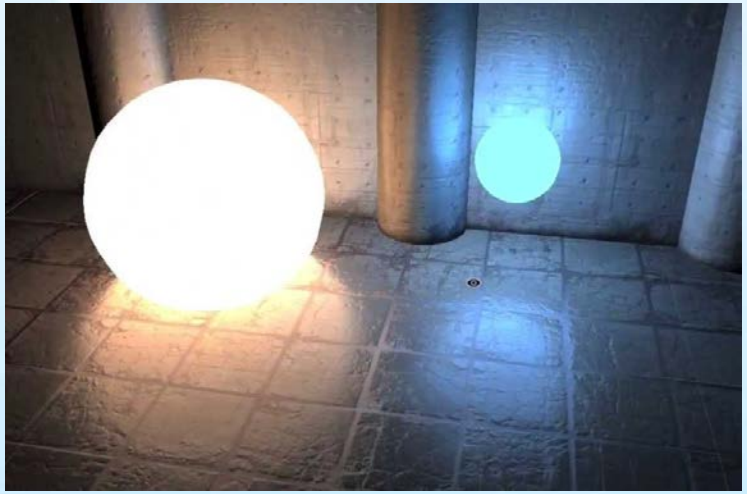
BASIC PRINCIPLES OF SHADERS WITH OPENGL ES 2.0 IN WEBGL 1.0

https://blendermarket.com/products/prism---fast--advanced-glass-shader-for-cycles

http://staggart.xyz/unity/stylized-water-shader/



https://www.clicktorelease.com/blog/vertex-displacement-noise-3d-webgl-glsl-three-js/



Shader Forge for Unity
https://www.assetstore.unity3d.com/#!/content/14147

Shader Forge for Unity
https://www.assetstore.unity3d.com/#!/content/14147

Shader Forge for Unity
https://www.assetstore.unity3d.com/#!/content/14147

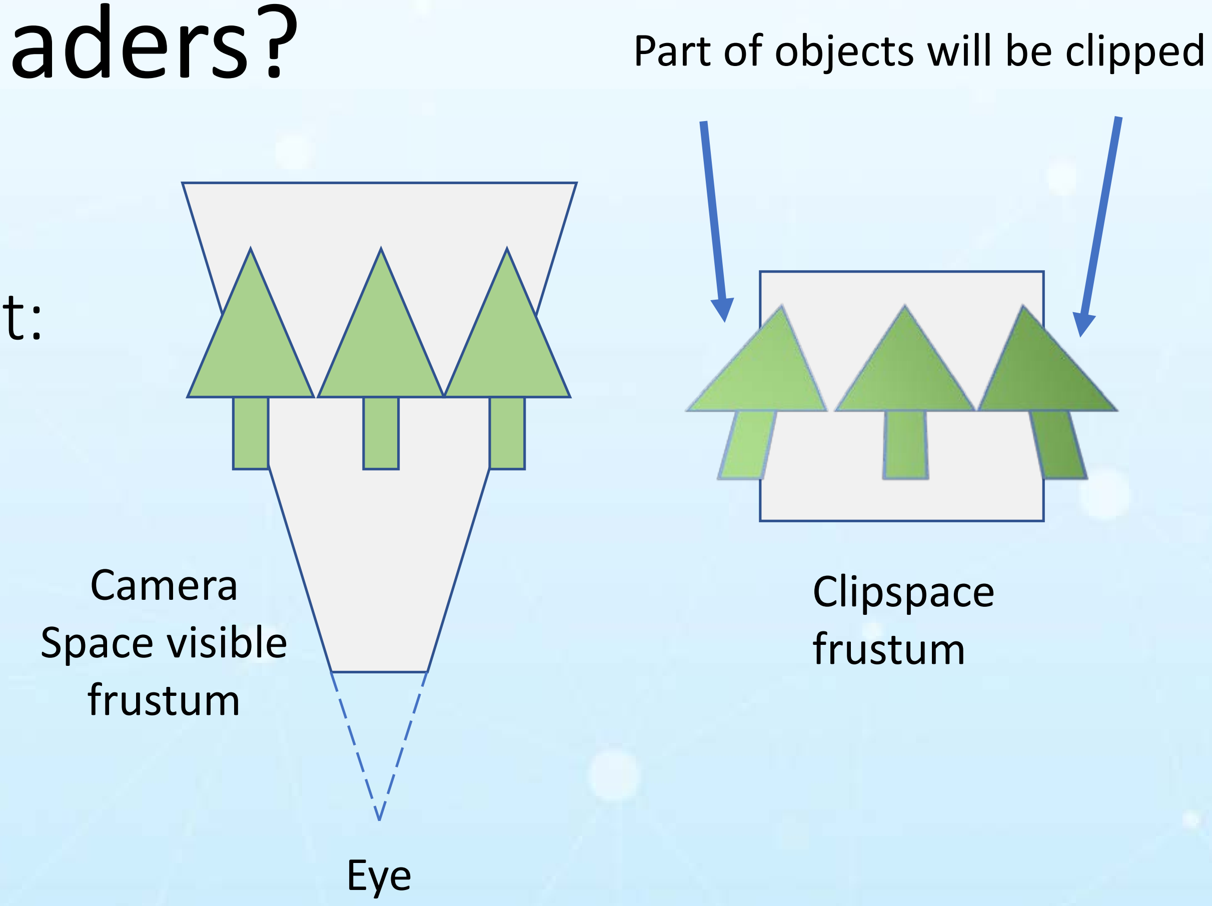
DERIVCO

Shaders

BASIC PRINCIPLES OF SHADERS WITH OPENGL ES 2.0 IN WEBGL 1.0

- So, what exactly are shaders?

  - 2 things we need to care about:
    - clipspace coordinates
    - colors.



Shaders

BASIC PRINCIPLES OF SHADERS WITH OPENGL ES 2.0 IN WEBGL 1.0

- So, what exactly are shaders?
  - How to insert those information?
    - By code – **vertex shader** – clipspace coordinates
    - By code – **fragment shader** – color.

Shaders

BASIC PRINCIPLES OF SHADERS WITH OPENGL ES 2.0 IN WEBGL 1.0

- So, what exactly are shaders?

  - Those 2 functions are each written in a very strictly typed C/C++ like language called GLSL (GL Shader Language).

Shaders

BASIC PRINCIPLES OF SHADERS WITH OPENGL ES 2.0 IN WEBGL 1.0

- So, what exactly are shaders?

  - Programs → override the existing implementation of:
    - **per-vertex** and
    - **per-pixel**
    - behavior handled by the **processor** on **screen**.

Shaders

BASIC PRINCIPLES OF SHADERS WITH OPENGL ES 2.0 IN WEBGL 1.0

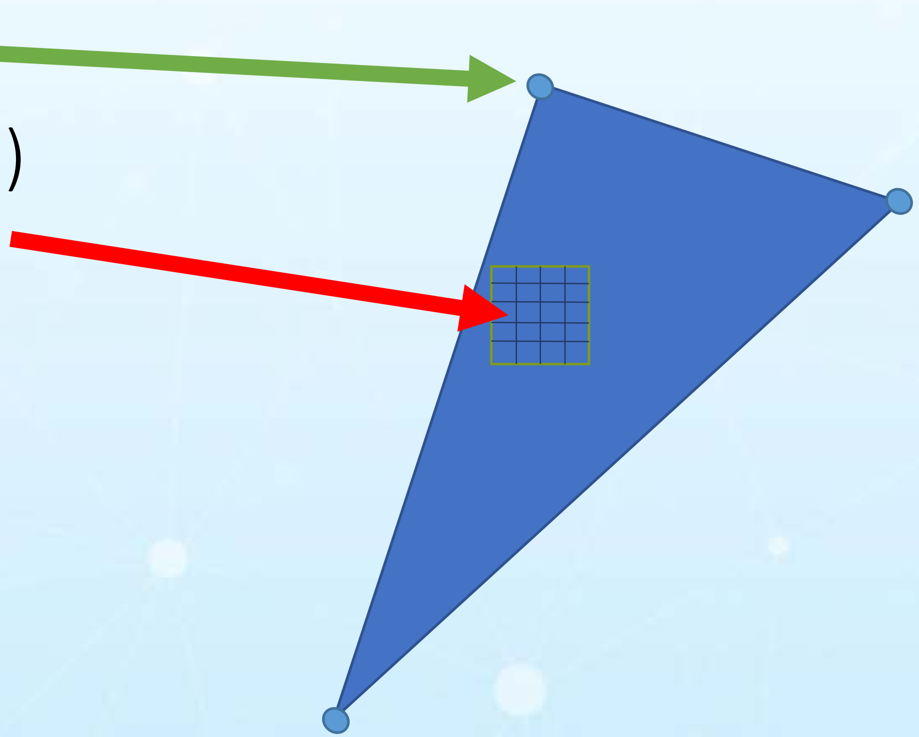
- HLSL, the High Level Shading Language
  - Microsoft
  - DirectX 8+
- Cg
  - Nvidia
- GLSL, the OpenGL Shading Language
  - Khronos Group
    - 3D graphics, Virtual and Augmented Reality, Parallel Computing, Neural Networks, and Vision Processing
      - Members: AMD, 3DLab, Apple, Google, Epic Games, Nvidia, etc.

Shaders

BASIC PRINCIPLES OF SHADERS WITH OPENGL ES 2.0 IN WEBGL 1.0

- OpenGL shaders give the user control over each
  - vertex
  - fragment (each pixel or partial pixel)

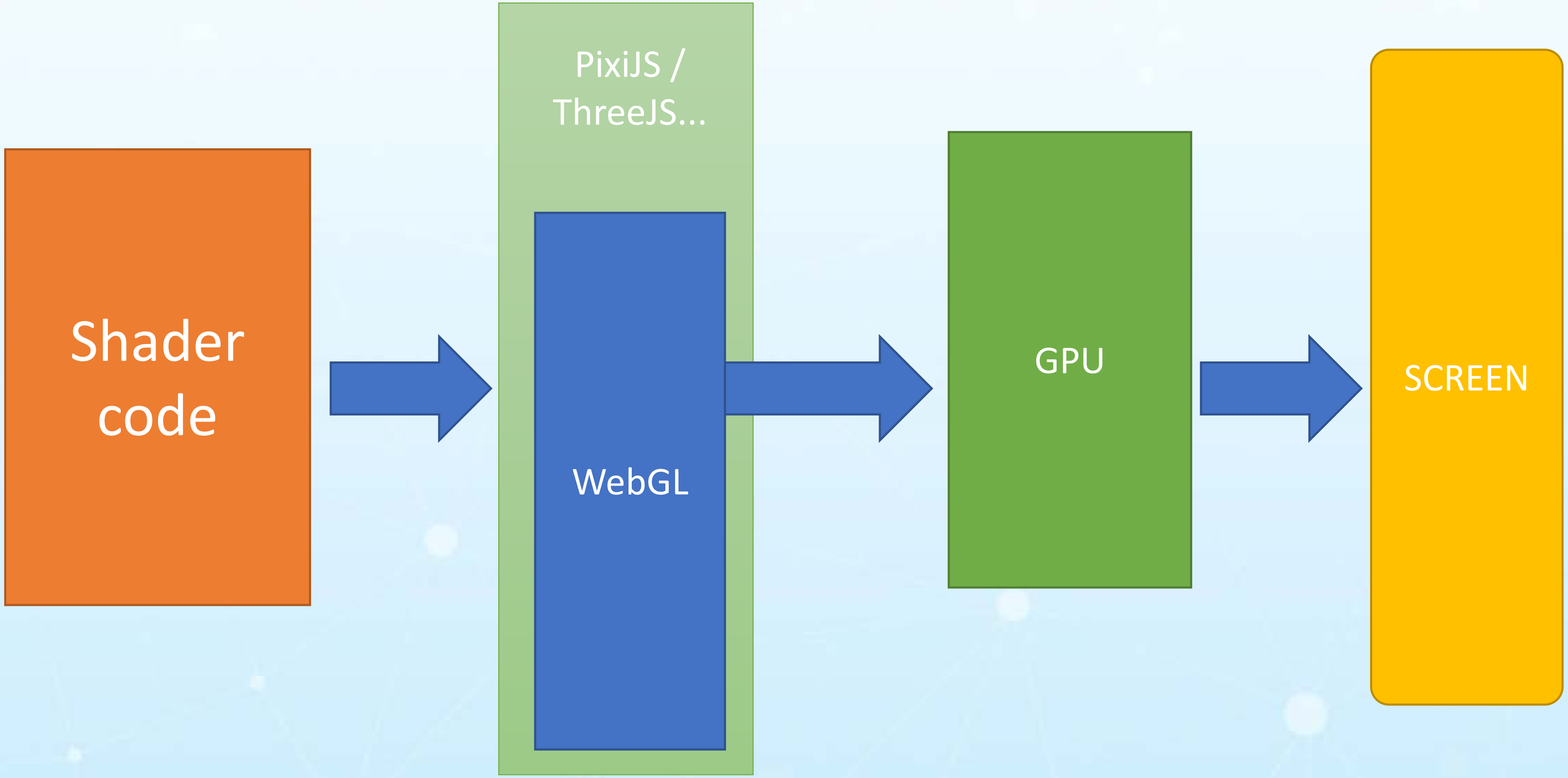
    interpolated between vertices.



DERIVCO

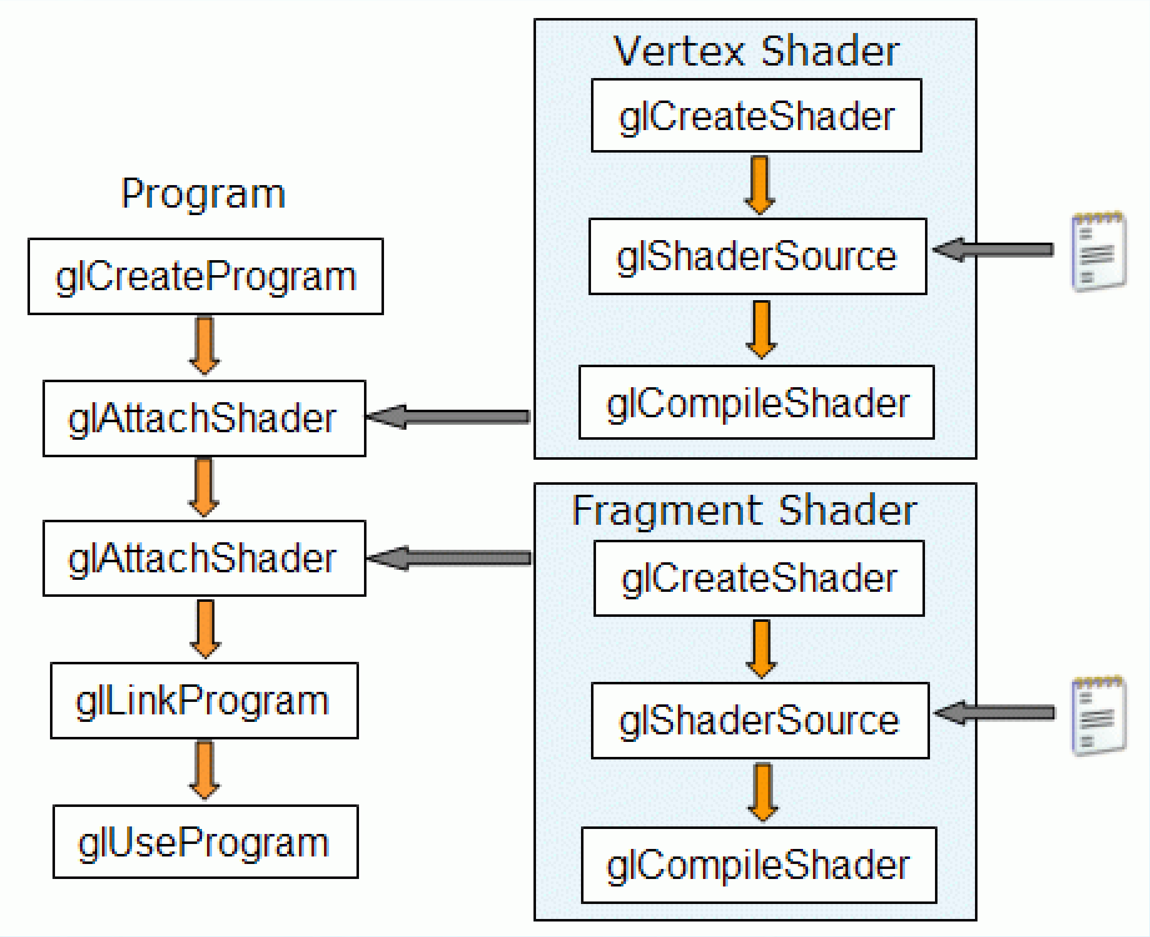
Shaders

BASIC PRINCIPLES OF SHADERS WITH OPENGL ES 2.0 IN WEBGL 1.0



Shaders

BASIC PRINCIPLES OF SHADERS WITH OPENGL ES 2.0 IN WEBGL 1.0

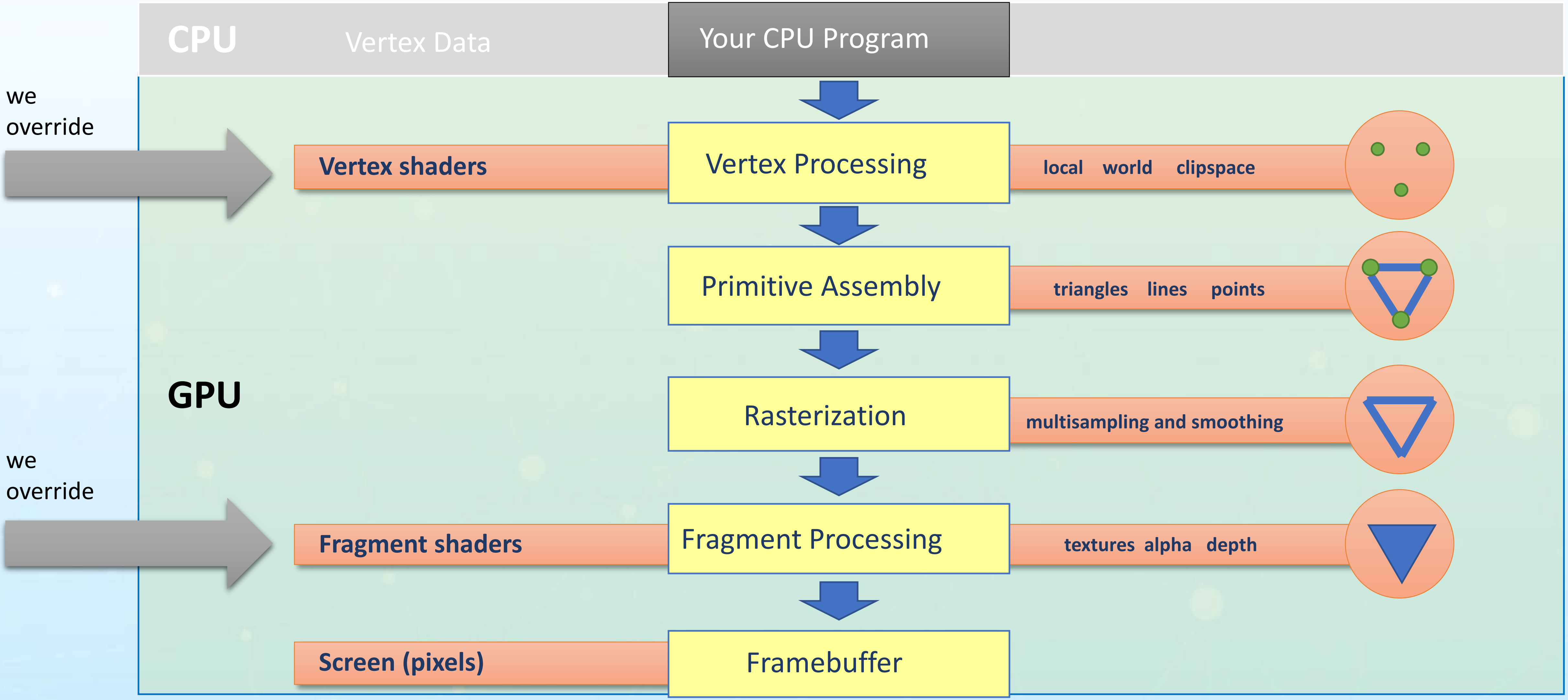


Source: http://www.lighthouse3d.com - opengl-setup-for-glsl

DERIVCO

Shaders

BASIC PRINCIPLES OF SHADERS WITH OPENGL ES 2.0 IN WEBGL 1.0



CPU

Vertex Data

Your CPU Program

we override

Vertex shaders

Vertex Processing

local world clipspace

Primitive Assembly

triangles lines points

GPU

Rasterization

multisampling and smoothing

we override

Fragment shaders

Fragment Processing

textures alpha depth

Screen (pixels)

Framebuffer

Shaders

BASIC PRINCIPLES OF SHADERS WITH OPENGL ES 2.0 IN WEBGL 1.0

- Code structure inside shader
  - Precision and variable declaration (with the [**qualifier**] [**precision**] and **type**)
    - precision highp float
    - attribute vec3 vertexPosition;
    - uniform lowp float time;
  - Functions (custom functions)
    - float calcShadowEffect(vec3 ambientLight){…}
  - Main Function (main() the last function inside code)
    - void main(){ gl_FragColor = vec4(….   }

Shaders

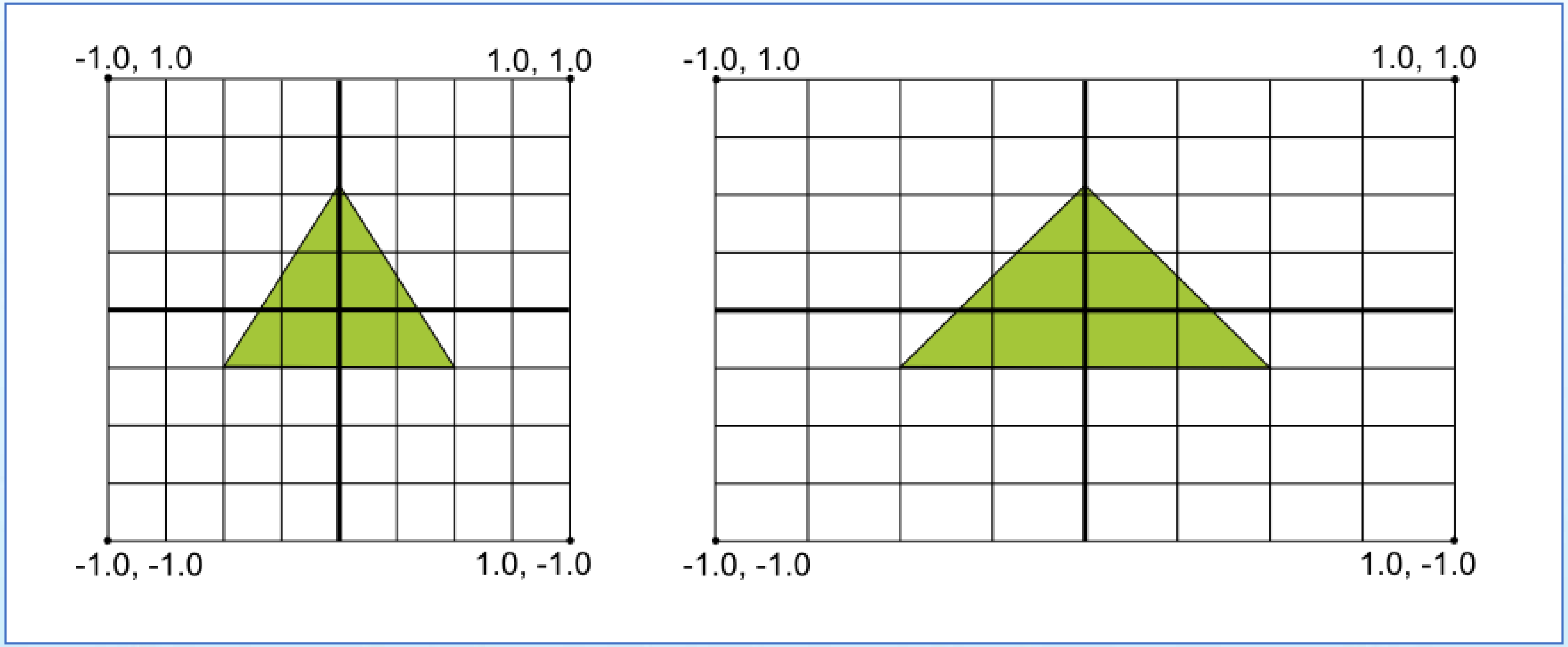
BASIC PRINCIPLES OF SHADERS WITH OPENGL ES 2.0 IN WEBGL 1.0

- Basic types
  - **void** → no function return value or empty parameter list
  - **bool** → Boolean
  - **int** → signed integer
  - **float** → floating scalar
  - **vec2, vec3, vec4** → n-component floating point vector
  - **bvec2, bvec3, bvec4** → Boolean vector
  - **ivec2, ivec3, ivec4** → signed integer vector
  - **mat2, mat3, mat4** → 2x2, 3x3, 4x4 float matrix
  - **sampler2D** → access a 2D texture
  - **samplerCube** → access cube mapped texture

Shaders

BASIC PRINCIPLES OF SHADERS WITH OPENGL ES 2.0 IN WEBGL 1.0

- Coordinates



Source: https://stuff.mit.edu/afs/sipb/project/android/docs/guide/topics/graphics/opengl.html

Shaders

BASIC PRINCIPLES OF SHADERS WITH OPENGL ES 2.0 IN WEBGL 1.0

- Structures and arrays
  - Structure

```
struct Light {
        vec4 position;
        vec4 ambient;
        float attenuation;
};
uniform Light lights[numLights];
```

  - Array
    - structures and blocks can be arrays
    - only 1-dimensional arrays supported
    - structure members can be arrays

```
float elements[42];
```

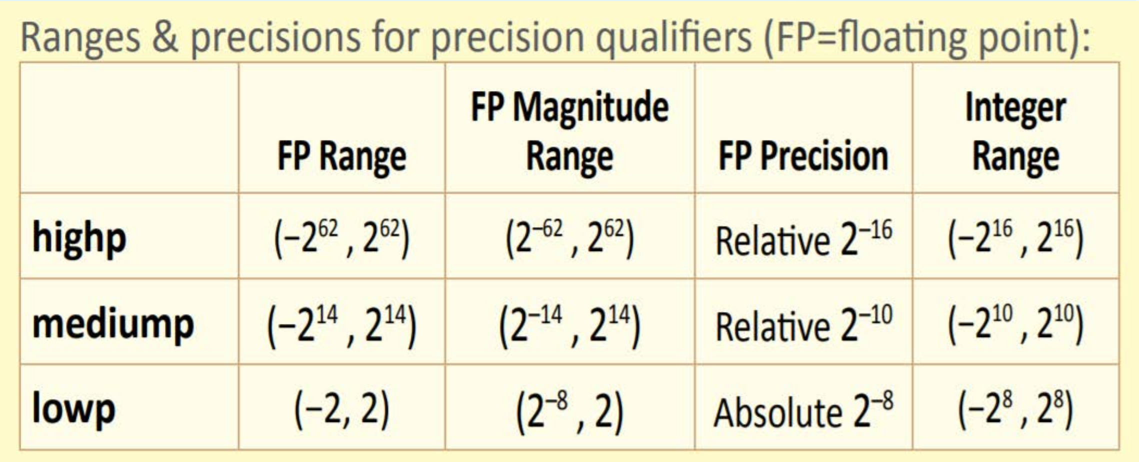
Shaders

BASIC PRINCIPLES OF SHADERS WITH OPENGL ES 2.0 IN WEBGL 1.0

- Precision
  - highp
  - mediump
  - lowp

```
precision highp float; // all float in highp

lowp float color;
highp mat4 someMatrix;
varying mediump vec2 v_someCoordinates;
uniform lowp vec3 u_effectValues;
```



Ranges & precisions for precision qualifiers (FP=floating point):

	FP Range	FP Magnitude Range	FP Precision	Integer Range
highp	$(-2^{62}, 2^{62})$	$(2^{-62}, 2^{62})$	Relative 2^{-16}	$(-2^{16}, 2^{16})$
mediump	$(-2^{14}, 2^{14})$	$(2^{-14}, 2^{14})$	Relative 2^{-10}	$(-2^{10}, 2^{10})$
lowp	$(-2, 2)$	$(2^{-8}, 2)$	Absolute 2^{-8}	$(-2^{8}, 2^{8})$

Source: WebGL Khronos Group Card

Shaders

BASIC PRINCIPLES OF SHADERS WITH OPENGL ES 2.0 IN WEBGL 1.0

- Some bult-in functions →
  - pow()
  - exp()
  - radians()
  - degrees()
  - sin()
  - cos()
  - dot()
  - clamp()
  - texture2D()
  - normalize()
  - Many others (see WebGL Card from Khronos Group)

Shaders

BASIC PRINCIPLES OF SHADERS WITH OPENGL ES 2.0 IN WEBGL 1.0

- Vertex shader
- Fragment shader

Shaders

BASIC PRINCIPLES OF SHADERS WITH OPENGL ES 2.0 IN WEBGL 1.0

- Vertex shader
  - Are run once for each vertex given to the graphics processor.
  - Transform each vertex's 3D position in virtual space to the 2D coordinate.
  - Vertex shaders can manipulate
    - position,
    - texture coordinate.
  - Cannot create new vertices.
  - Output of the vertex shader goes to the next stage in the pipeline →
    - *geometry shader if present* or the rasterizer otherwise.

Shaders

BASIC PRINCIPLES OF SHADERS WITH OPENGL ES 2.0 IN WEBGL 1.0

- Fragment shader
  - Or Pixel Shader, compute color and other attributes of each fragment (each pixel or partial pixel).
  - Fragment shaders range from always outputting:
    - same color,
    - applying a lighting value,
    - bump mapping,
    - shadows,
    - specular highlights,
    - translucency and other phenomena.

DERIVCO

Shaders

BASIC PRINCIPLES OF SHADERS WITH OPENGL ES 2.0 IN WEBGL 1.0

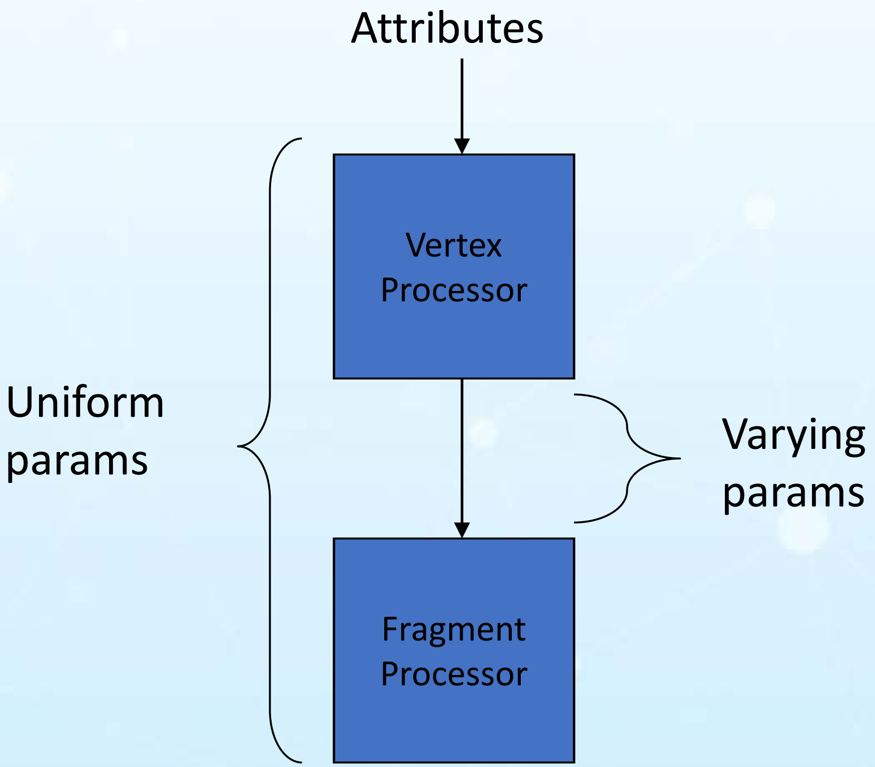
- Fragment shader
  - They can alter the depth of the fragment (for Z-buffering), or output more than one color if multiple render targets are active.
  - In 3D graphics, a fragment shader alone ***cannot*** produce very complex effects.

DERIVCO

Shaders

BASIC PRINCIPLES OF SHADERS WITH OPENGL ES 2.0 IN WEBGL 1.0

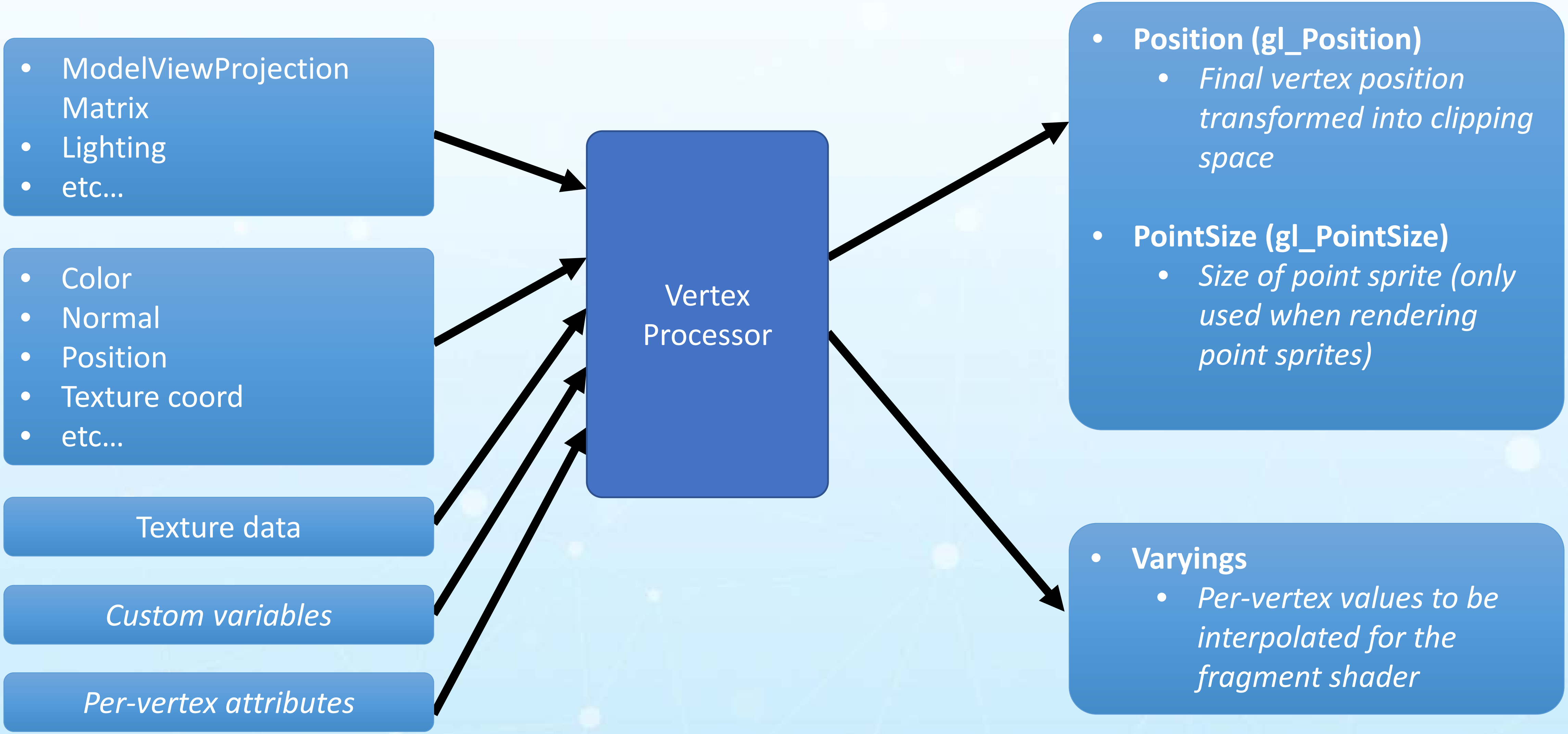
- Four storage qualifiers of shader parameter
  - Const
    - Compile-time constant or read-only function parameters
  - Attribute
    - Set per vertex - comes from outside shaders (your CPU program or framework) or defined by OpenGL engine.
    - Only in vertex shader.
    - Ex.: position, color, texture coordinate(s)
  - Uniform
    - Set throughout execution. Global parameter.
    - Ex.: Model-View-Projection Matrix, time, ambient light.
  - Varying
    - Passed from vertex shader to fragment shader.
    - Per-vertex values to be interpolated (across connected vertices) for the fragment shader



DERIVCO

Shaders

BASIC PRINCIPLES OF SHADERS WITH OPENGL ES 2.0 IN WEBGL 1.0



- ModelViewProjection Matrix
- Lighting
- etc…

- Color
- Normal
- Position
- Texture coord
- etc…

Texture data

Custom variables

Per-vertex attributes

Vertex Processor

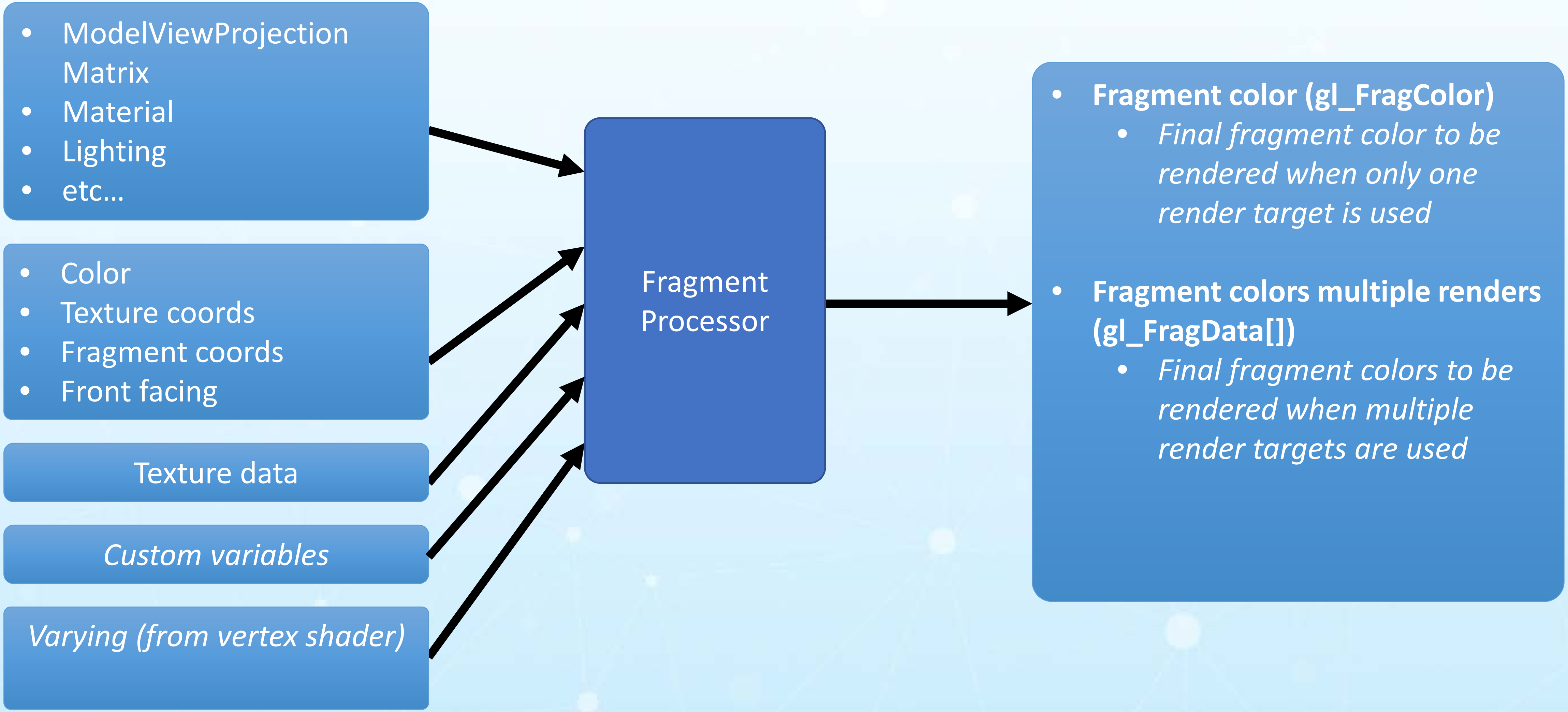
- **Position (gl_Position)**
  - *Final vertex position transformed into clipping space*
- **PointSize (gl_PointSize)**
  - *Size of point sprite (only used when rendering point sprites)*

- **Varyings**
  - *Per-vertex values to be interpolated for the fragment shader*

DERIVCO

Shaders

BASIC PRINCIPLES OF SHADERS WITH OPENGL ES 2.0 IN WEBGL 1.0



- ModelViewProjection Matrix
- Material
- Lighting
- etc…

- Color
- Texture coords
- Fragment coords
- Front facing

Texture data

Custom variables

Varying (from vertex shader)

Fragment Processor

- **Fragment color (gl_FragColor)**
  - *Final fragment color to be rendered when only one render target is used*
- **Fragment colors multiple renders (gl_FragData[])**
  - *Final fragment colors to be rendered when multiple render targets are used*

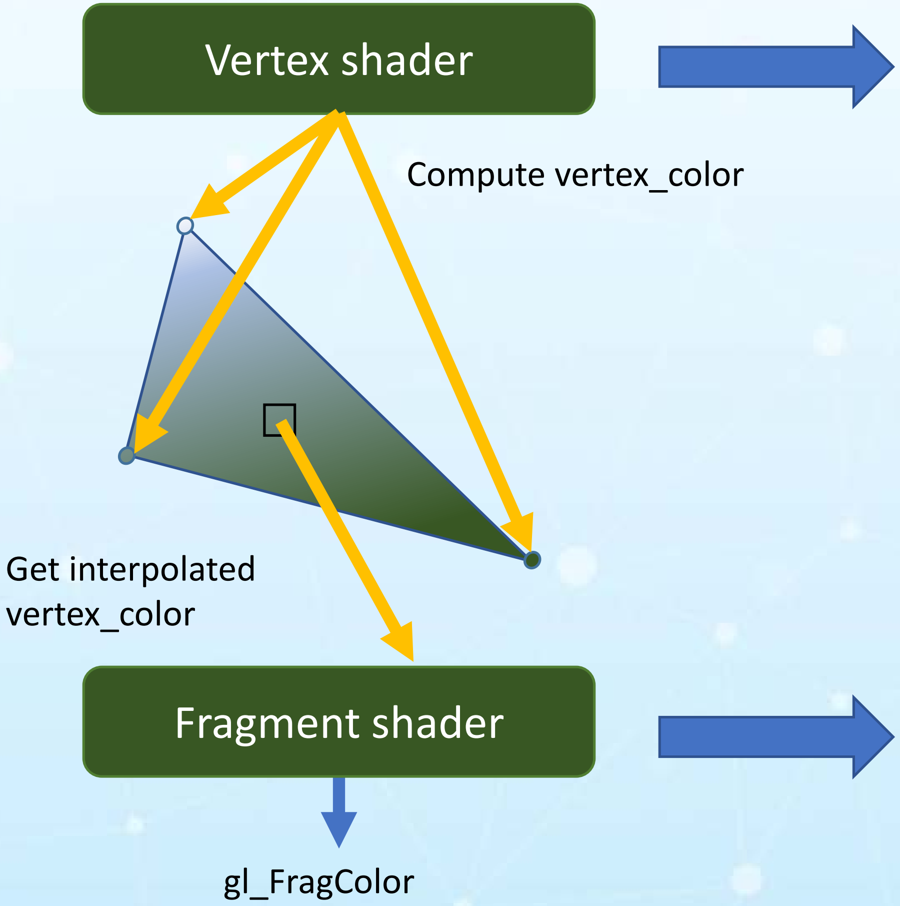
Shaders

BASIC PRINCIPLES OF SHADERS WITH OPENGL ES 2.0 IN WEBGL 1.0

- Let’s talk about:
  - Varying
  - Normal map
  - Model View Projection Matrix
  - Uv coordinates
  - Maks for coordinates

Shaders

- One thing about Varying



Vertex shader

Compute vertex_color

Get interpolated vertex_color

Fragment shader

gl_FragColor

```
varying vec4 v_vertexColor;
attribute vec4 a_vertexPosition;

void main()
{
		v_vertexColor = vec4(a_vertexPosition.xy * 0.5,
a_vertexPosition.zw * 0.2);
		gl_Position = a_vertexPosition;
}
```

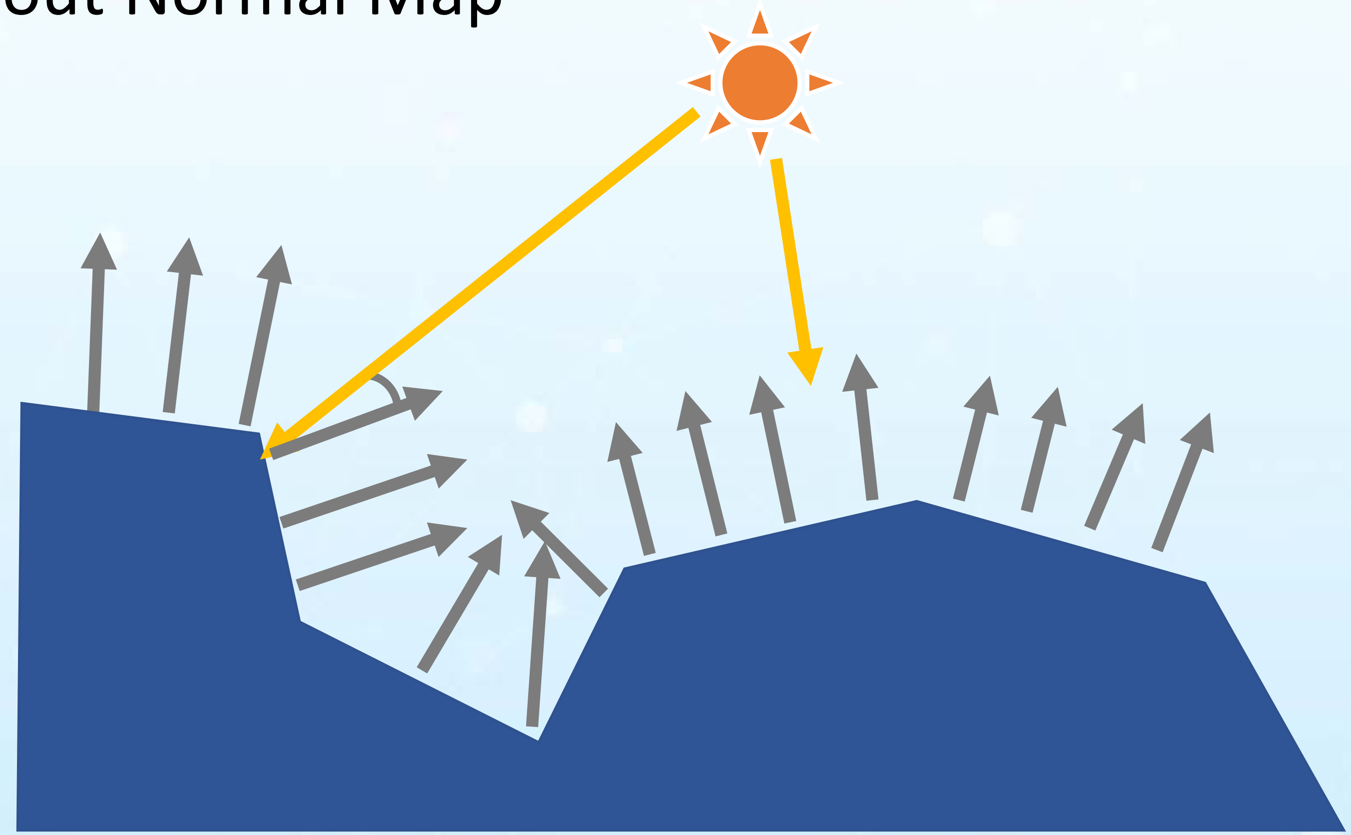
```
varying vec4 v_vertexColor; // from vertex shader

void main()
{
		gl_FragColor = v_vertexColor;
}
```

Shaders

BASIC PRINCIPLES OF SHADERS WITH OPENGL ES 2.0 IN WEBGL 1.0

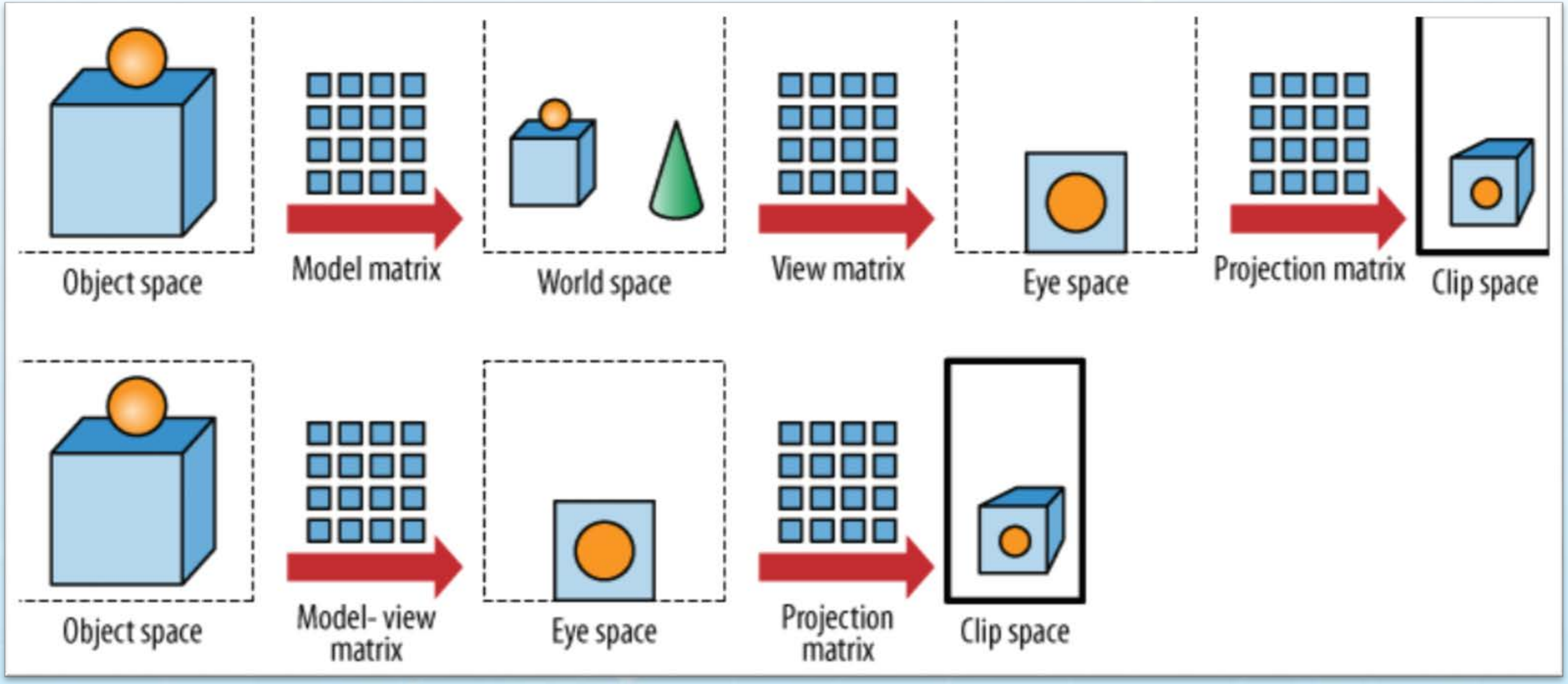
- One thing about Normal Map



Shaders

BASIC PRINCIPLES OF SHADERS WITH OPENGL ES 2.0 IN WEBGL 1.0

- One thing about Model-View-Projection Matrix



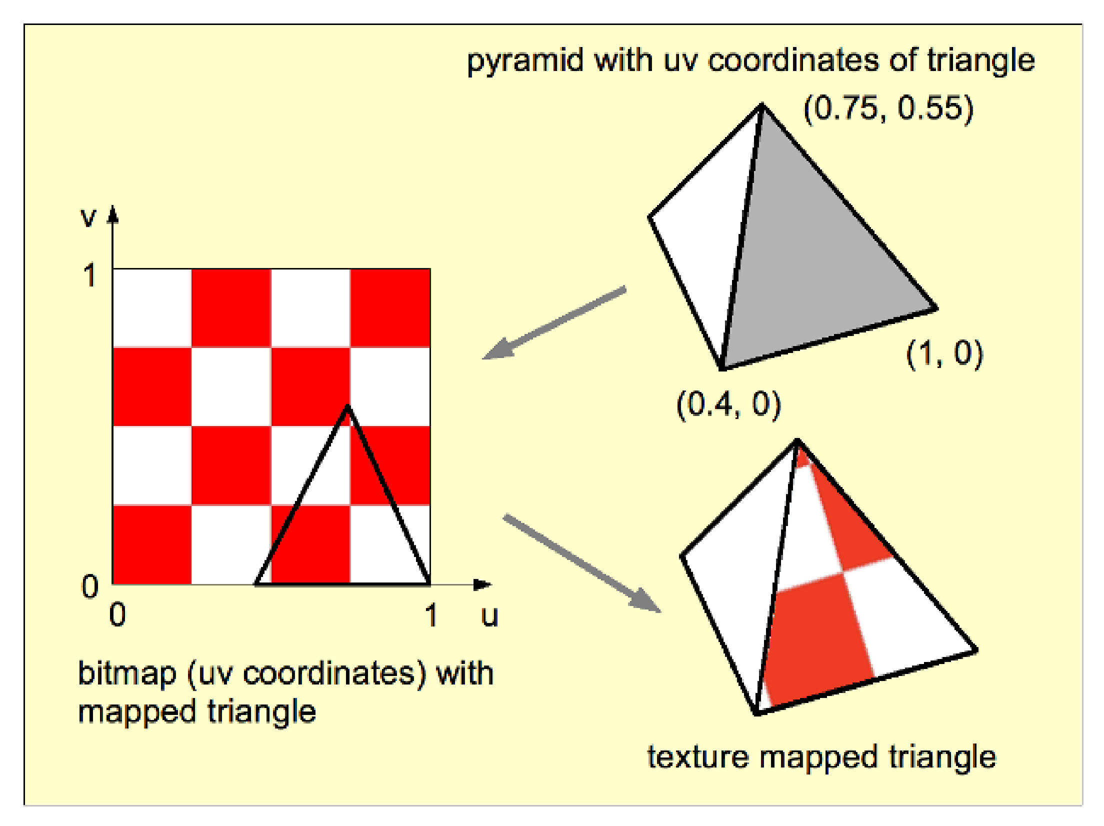
Source: https://www.safaribooksonline.com/library/view

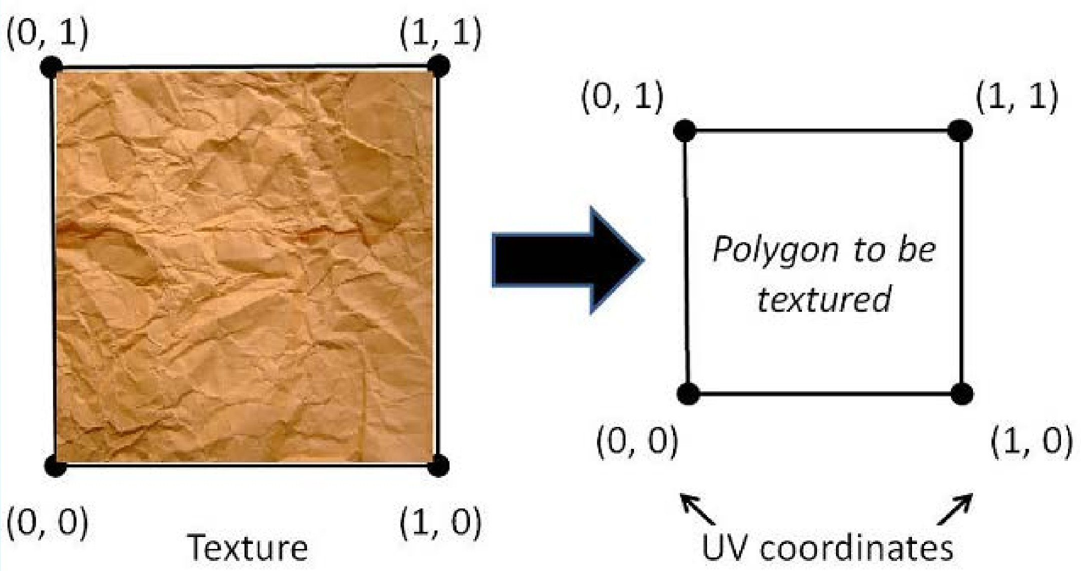
DERIVCO

Shaders

BASIC PRINCIPLES OF SHADERS WITH OPENGL ES 2.0 IN WEBGL 1.0

- One thing about UV coordinates





Source: http://www.c-jump.com/bcc/common/Talk3/OpenGL/Wk07_texture/Wk07_texture.html

Shaders

BASIC PRINCIPLES OF SHADERS WITH OPENGL ES 2.0 IN WEBGL 1.0

- One thing about masks for coordinates
  - XYZW, STPQ, or RGBA
    - Common name for texture coordinates are:
      - U and V.
    - Then came 3D Textures and should be used for 3D textures the W letter.
    - That causes a conflict with position:
      - X, Y, Z, and W.
    - To avoid such conflicts, OpenGL's convention is that the components of texture coordinates are named **S, T, and R.**
    - Came GLSL and this swizzle masks came around → conflicts with **RGBA**
    - So, they decided to use **STPQ** for textures coordinates.
    - XYZW, STPQ, or RGBA have the same origin.
      - But we cannot use **position.xt**

Shaders

BASIC PRINCIPLES OF SHADERS WITH OPENGL ES 2.0 IN WEBGL 1.0

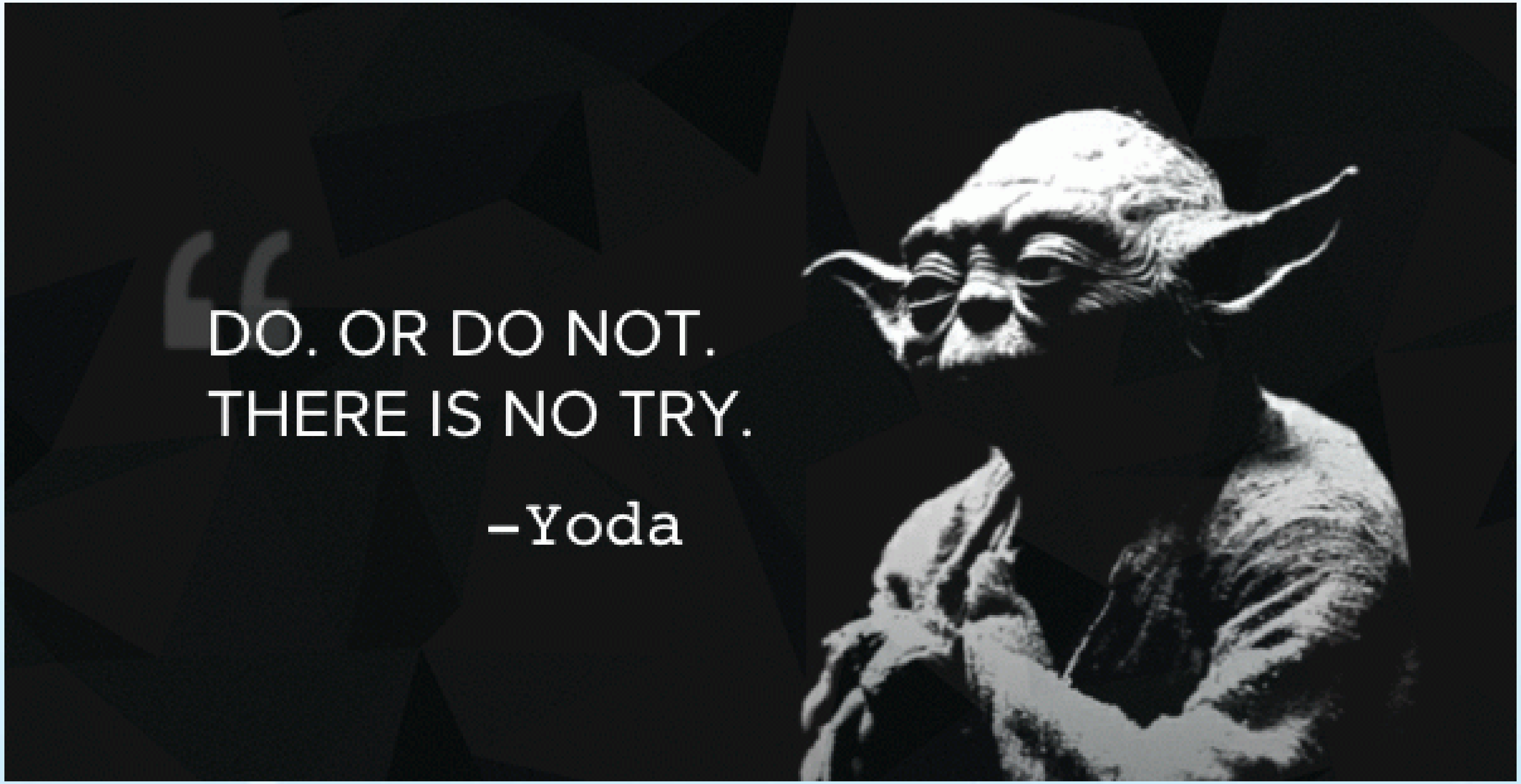
References to follow

https://www.khronos.org/files/webgl/webgl-reference-card-1_0.pdf
http://www.opengl-tutorial.org/
https://thebookofshaders.com/

Shaders

BASIC PRINCIPLES OF SHADERS WITH OPENGL ES 2.0 IN WEBGL 1.0

Let’s try!



Shaders

BASIC PRINCIPLES OF SHADERS WITH OPENGL ES 2.0 IN WEBGL 1.0

Thank you!

https://80.lv/articles/unreal-engine-4-stylized-rendering-workflow/