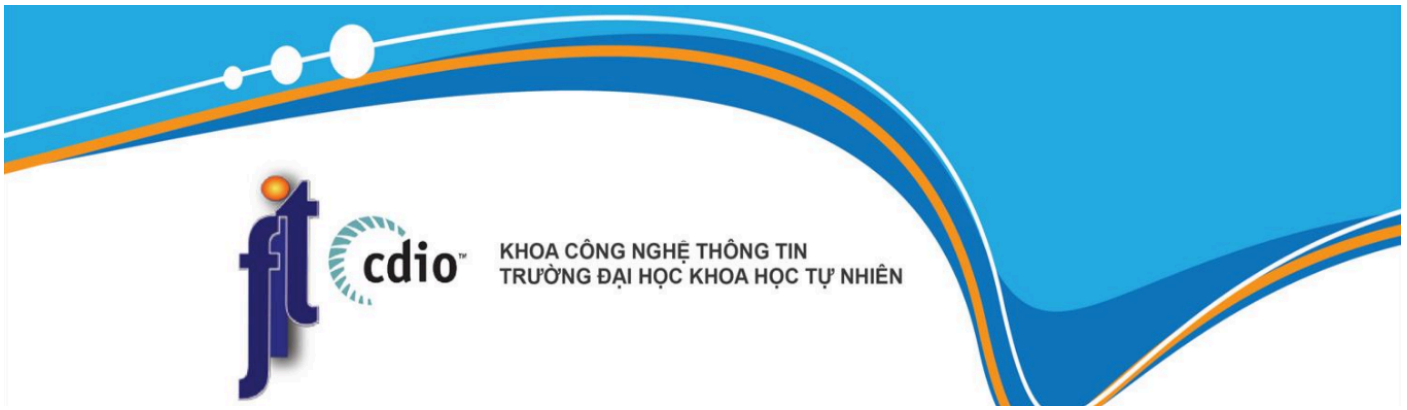


Computer graphics - Practice

Phạm Thanh Tùng - Võ Thế Hào

Week 01 - WebGL Introduction

1. Summary

- **Introduction to WebGL**

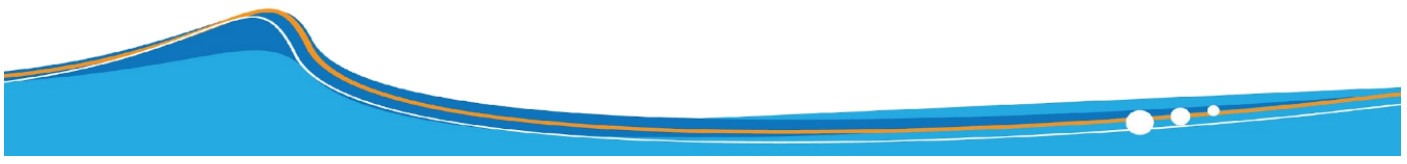
WebGL (Web Graphics Library) is a graphics library for the web, it is derived from OpenGL ES (2D and 3D graphics library on embedded systems: phones, electronics, motor vehicles). WebGL provides the same basic functionality as OpenGL ES and works well on modern 3D graphics hardware.

WebGL is a Javascript API that can be used in HTML5. WebGL is coded in the <canvas> tag of HTML 5, this allows the browser to access and use the GPU to output graphics.

WebGL is supported by most modern browsers: Chrome, FireFox, IE, Opera... without any additional installation required.

Within the scope of the subject, students will use the WebGL environment to practice with the following notes

- *WebGL is a standard 3D environment, but most of the practical content will be based on basic 2D theory. The Oz dimension can be ignored by choosing the default value = 0 for the parameters*
- *The WebGL environment was chosen because of its popularity in current applications (popular web environments from the 2010s onwards) and convenience of use (can perform direct programming without complicated installation). junk)*
- *The WebGL environment does not lose the generality of the popular programming environment for computer graphics, OpenGL*



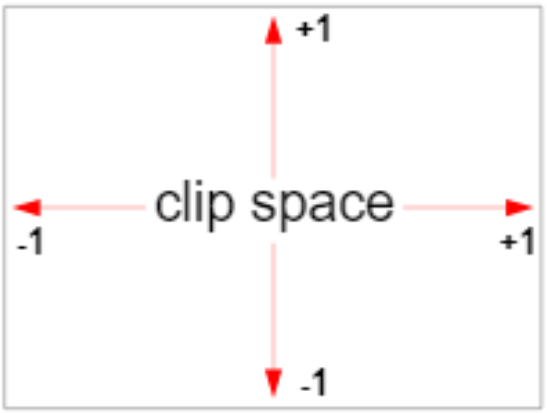
Self-reference material

- Javascript cheatsheet: http://www.cheat-sheets.org/sites/javascript.su/
- Instructions: https://webglfundamentals.org/
- Look up: https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API

- **Basic objects & concepts in WebGL**

Clip coordinate system in WebGL

Coordinates in WebGL have 3 dimensions x, y, z like the 3D system, however in which z represents depth. This coordinate system is limited to (1, 1, 1) to (-1, -1, -1) meaning that if the WebGL project screen is considered a cube, one of its corners has coordinates of (1 , 1, 1) and the angle opposite the center is (-1, -1, -1). WebGL will not display anything drawn outside this limit.



Vertices (vertices): usually to draw a polygon, we connect the points to form the desired polygon. A vertex is a point intersected by the edges of a 3D object. It is represented by 3 real number values corresponding to each axis x, y, z. In WebGL vertices are stored using Javascript arrays.

Indices: are positive integers used to identify vertices. Indices are used to draw meshes in WebGL. In WebGL indices are stored using Javascript arrays.

Buffers: These are areas of GPU memory allocated to WebGL to hold data. There are different types of buffers: drawing buffer, frame buffer, vertex buffer, index buffer.

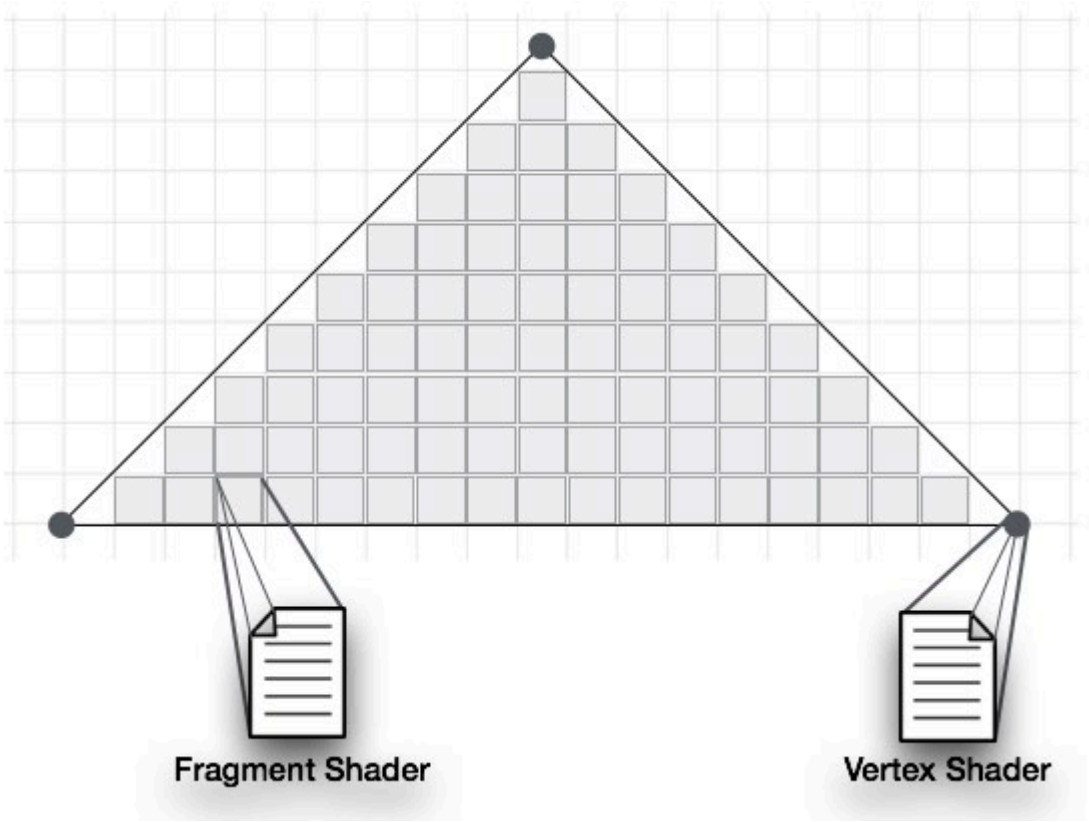
- Vertex buffer object: used to store data corresponding to vertices.
- Index buffer object: used to store data corresponding to the index.
- Frame buffer: is a part of graphics memory used to store data about scenes.

Mesh (mesh)

To draw a 3D object we need to create one or more basic polygons using points, lines, or triangles. Then use these polygons to form a graphical mesh. A 3D object is formed using basic polygons called meshes.

WebGL Shader Program

Shaders are programs for the GPU and are written in GLSL (OpenGL Embedded System Shader Language). These shaders are used to define how vertices transform, lighting, and cameras interact to create a particular image. In other words, it defines and installs algorithms so that from image points we can create an image of an object.



Vertex Shaders

Vertex shader is the program called to transform vertices. It is used to transform geometry from one place to another. It processes data of each vertex such as: coordinates, color, texture (surface structure). Vertex shader tasks include:

- Transform peaks.
- Apply color.
- Create light.
- Generates textures
- Transform textures.

Fragment Shader (Pixel Shader)

A graphic mesh is composed of many triangles, and the surface of each triangle is understood as a fragment. Fragment Shader is the code that runs on all points of a fragment. It is often used to calculate and color points. The tasks of Fragment Shader are:

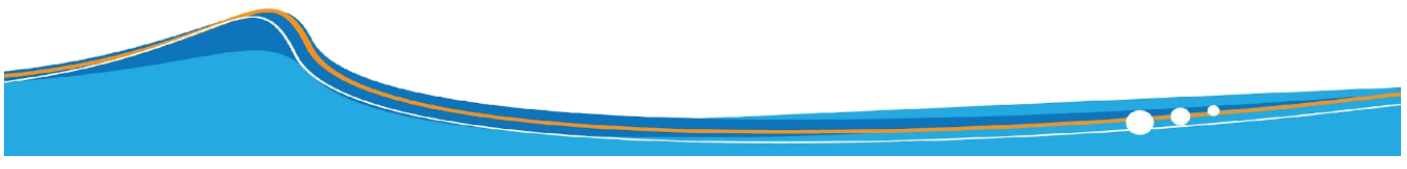
- Access textures.
- Apply texture.
- Create opacity.
- Color calculation.

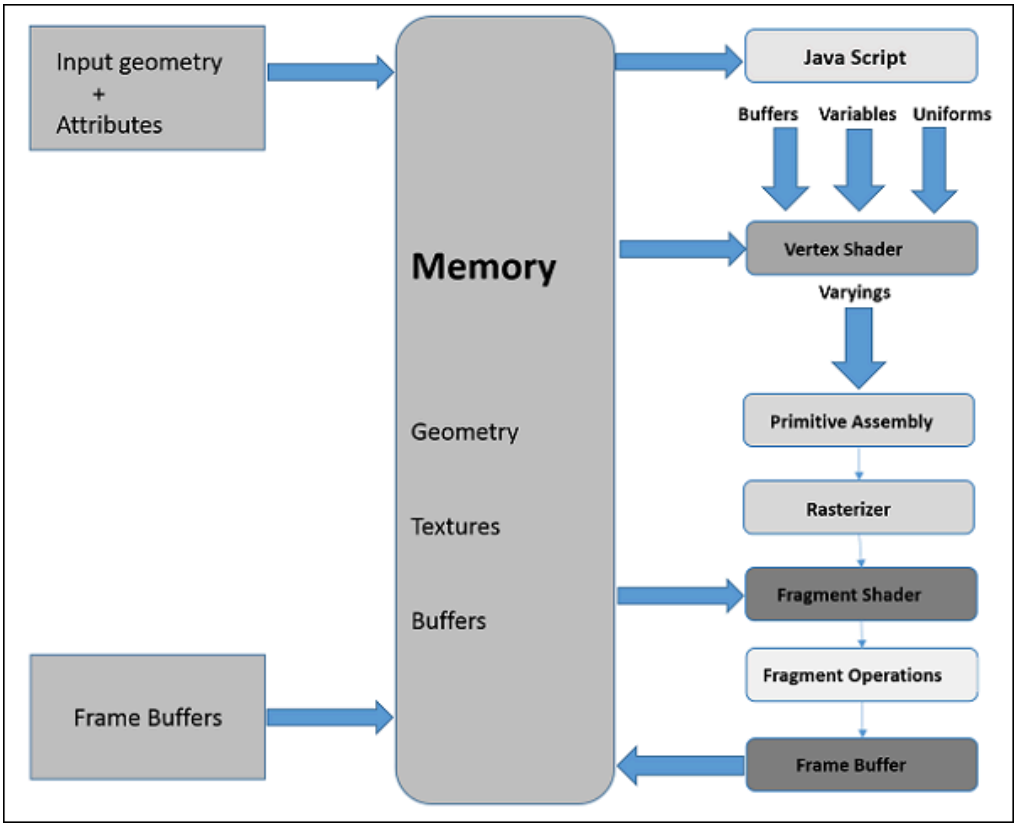
[Read more] Some common variables of OpenGL ES SL (GLSL)

- ***Attributes*** *: are variables that hold the input values of vertex shaders.*
- ***Uniforms*** *: are variables that hold input data common to both vertex and fragment shaders: light position, texture, color.*
- ***Textures*** *: image data in array form (usually loaded from file)*
- ***Varyings*** *: are variables used to pass data from vertex to fragment shader.*

Graphics Pipelines

WebGL renders 3D graphics through a series of calls





1. **Initialize WebGL:** in this step Javascript is used to get the WebGL context
2. **Create a data array:** we use Javascript to create an array to hold the data of the image to draw.
3. **Create buffer object:** we create a buffer object and pass in the data just created above.
4. **Create Shaders:** we create, compile and link shaders using Javascript.
5. **Create Attribute:** create attributes and associate them with buffer objects using Javascript.
6. **Create Uniforms:** create and link Uniforms using Javascript.
7. **Transformation matrix:** using Javascript we create a motion transformation matrix.

When the data of the images to be drawn is created by Javascript, we will pass them to the shaders in the form of buffer objects for processing.

Vertex Shaders

The data passed to the vertex shader as a buffer object is processed for rendering when the following methods drawElements() and drawArray() are called. The task of the vertex shader is to calculate the position of the vertices and then store it in the gl_position variable. In addition, it also calculates the color, texture, and vertices related to each other.

Primitive Assembly

The data we pass into the processing shaders are the vertices of the geometries we want to draw. After being processed by shaders, the primitive assembly process will take place to connect the vertices to form the shape to be drawn.

Rasterization

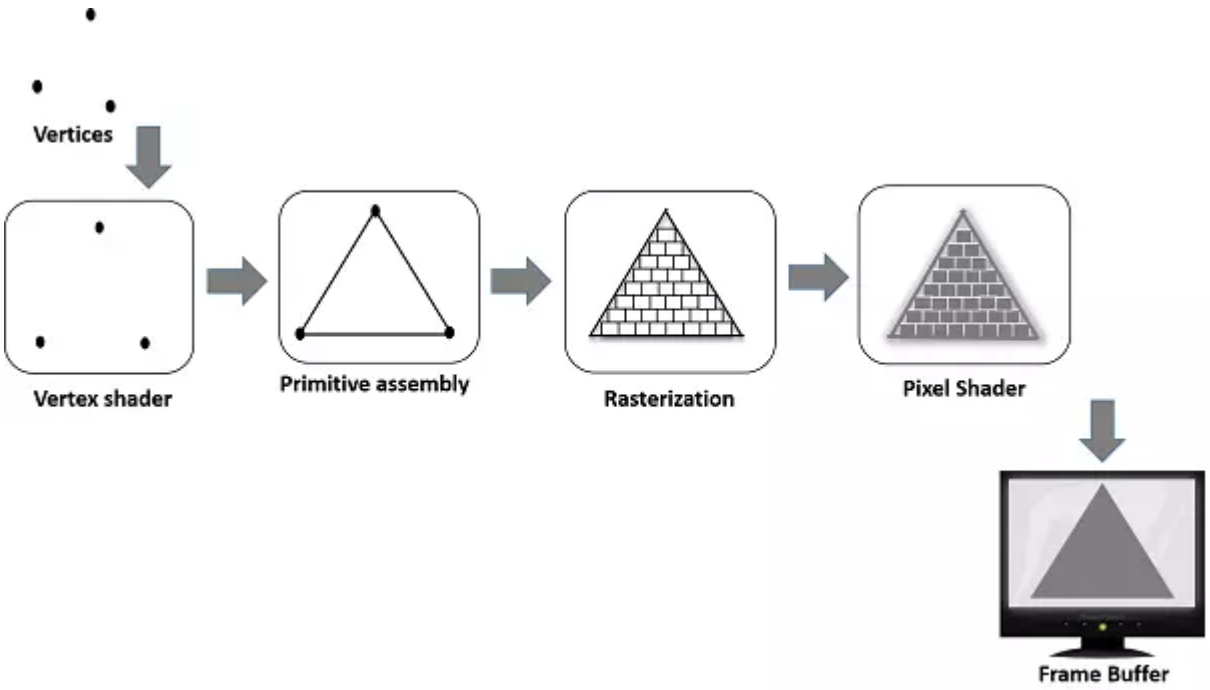
Rastersizer will receive input vertex coordinates (transformed from previous steps) and generate fragment(s) for those vertices.

Fragment shaders

In this step, the fragment shader will take each fragment from the rastersizer and calculate the depth, stencil and color values of that fragment. In this step, the color of a fragment will be decided.

Frame buffer

The final step is for the image you want to draw to be processed in terms of color, texture and displayed on the screen.



2. Case Studies

- Through the sample program about WebGL, learn the principles and how it works
- Based on the sample program, refactor auxiliary functions to optimize programming and design of more complex programs.

3. Specific instructions

Refer to the basic sample program that works with WebGL

- https://jsfiddle.net/royalgarter/xdz9pqag/

Jsfilddle is an emulation platform that allows you to easily test HTML, CSS and Javascript code quickly without using any other tools.

4. Exercise

Based on the sample program provided, look up the functions used and answer the following questions

1. The goal of the " **createShader** " function? Why do we have to pass code as a string to create shaders?
2. Explain **the meaning & parameters** (including value type & data range) of the following functions
    a. gl.bufferData
    b. gl.vertexAttribPointer
    c. gl.viewport
3. In what form are colors in WebGL transmitted/used? What is the range of values of color parameters / color channels?
4. In the function " **clearGL** " what will happen when the line **gl.enable(gl.DEPTH_TEST)" is uncommented.**
5. When using **Jsfilddle,** if the program has an error (the code cannot run), how can we find out which line has the error?

5. Note

Homework should be submitted in the following format

- 01 report file in PDF format
- Required name: **DHMT_TH_01_<MSSV>.pdf**
    - Write in correct capitalization
    - Do not add any other characters
    - Do not compress into zip, rar, tar files...