Lab03. First WebGL Programming

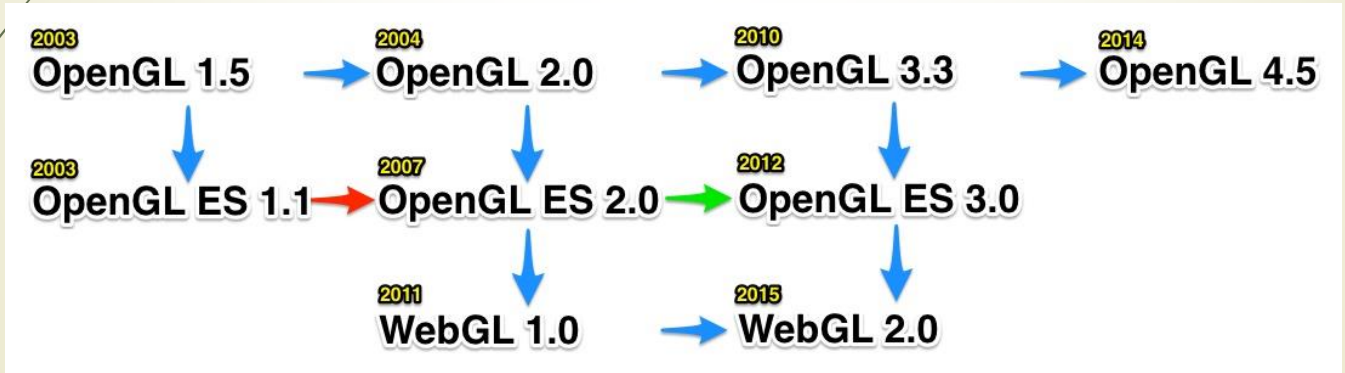
DoHoon Lee Ph.D

Visual & Biomedical Computing(VisBiC) Lab.
School of Computer Science & Engineering
Pusan National University

http://visbic.pusan.ac.kr/

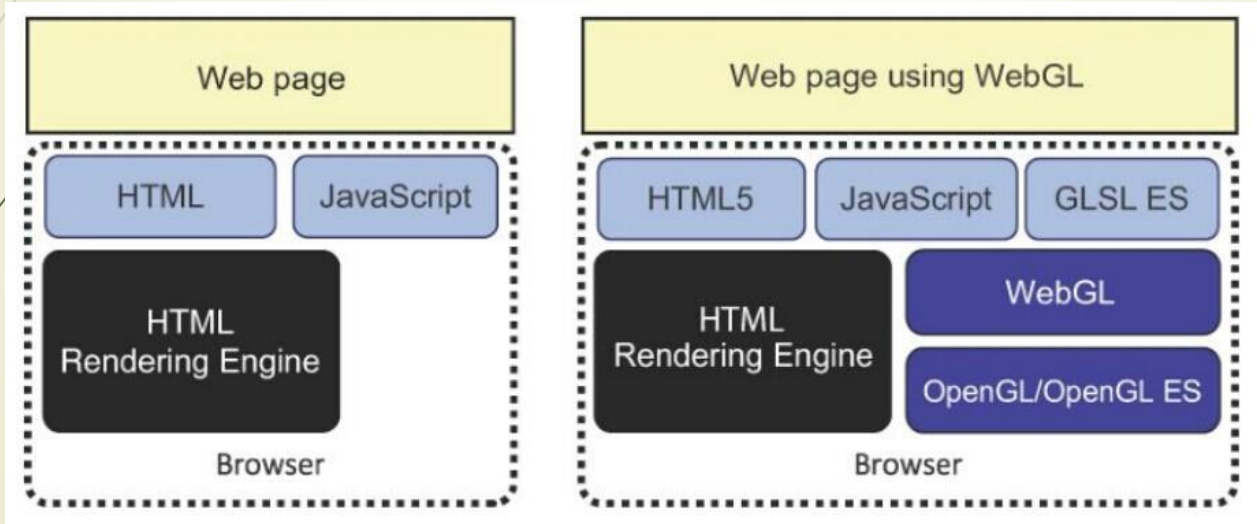
WebGL

- WebGL is a technology that enables drawing, displaying, and interacting with sophisticated interactive three-dimensional computer graphics("3D graphics") from within web browsers.
- Relationship among OpenGL, OpenGL ES and WebGL



Structure of WebGL Applications

- The software architecture of dynamic web pages(left) and web pages using WebGL(right)



HTML5

- **HTML5로 통칭되는 요소는 HTML5 뿐만 아니라 CSS3, Javascript API 확장을 포함한 것이다.**

HTML5

- HTML, CSS, Javascript가 합쳐져 웹 어플리케이션을 제공

내용과 태그를 삽입해
홈페이지의 틀을 세움

홈페이지 디자인을 그려
외형을 꾸미는 역할을 함

동적인 기능을 덧입혀
홈페이지의 기능을 제공

Editors

- jEdit
  - http://www.jedit.org/

- gedit
  - 그놈 데스크톱 환경, 마이크로소프트 윈도, 맥 OS X용으로 개발된 자유 소프트웨어인 텍스트 편집기이다. UTF-8과 호환하며, 프로그램 코드, 마크업 언어와 같은 구조화된 텍스트 문서를 편집하는 용도에 중점을 두고 개발했다. 그놈 프로젝트의 철학에 따라 깔끔하고 단순한 GUI가 특징이다.
  - https://wiki.gnome.org/Apps/Gedit

Drawing on the Web

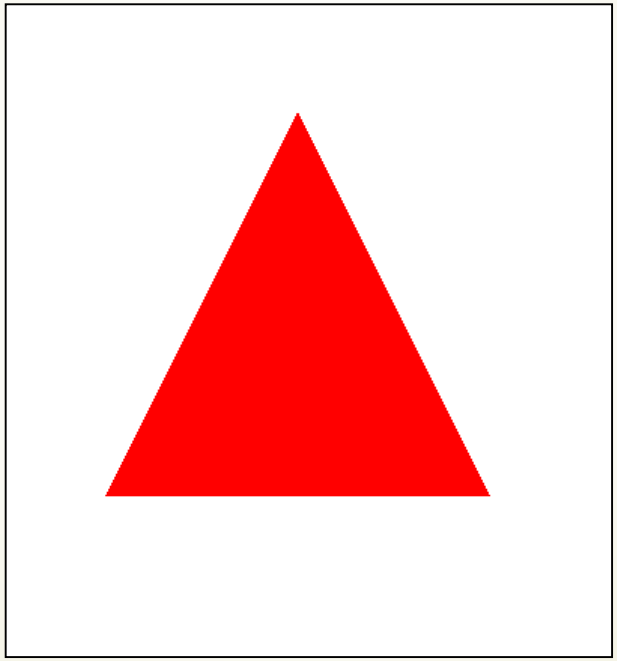
- <canvas> : defines a drawing area on a web page
  - Without WebGL : only allows you to draw 2D graphics using JavaScript
  - With WebGL : can use it for drawing 3D graphics

Drawing a Rectangle with HTML

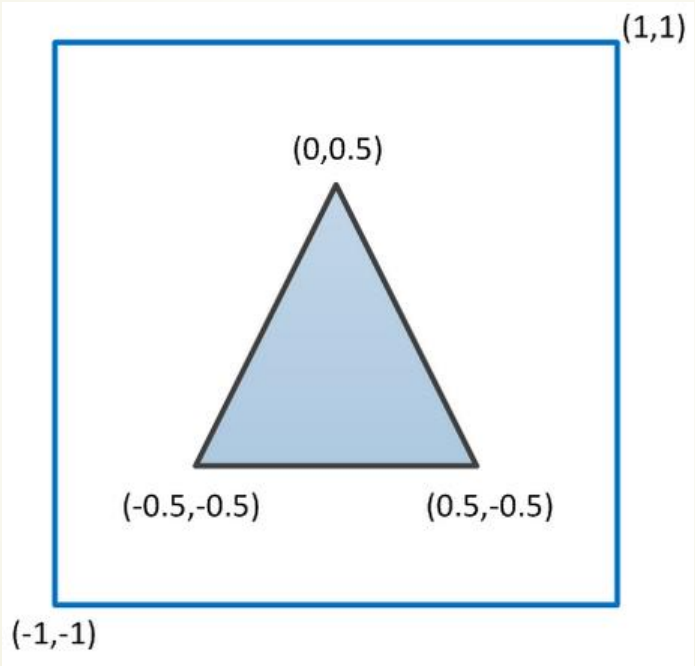
```
<!DOCTYPE html>
<html>
  <head>
    <title>Draw a blue rectangle <canvas version)</title>
  </head>

  <body onload="main()">
    <canvas id="example" width="400" height="400">
    Please use a browser that supports "canvas"
    </canvas>
    <script src="DrawRectangle.js"></script>
  </body>
</html>
```

```
//DrawRectangle.js
function main(){
      //Retrieve <canvas> element
      var canvas = document.getElementById('example');

      if(!canvas){
         console.log('Failed to retrieve the <canvas> element');
         return;
      }

      //Get the rendering context for 2D CG
      var ctx = canvas.getContext('2d');

      //Draw a blue rectangle
      ctx.fillStyle = 'rgba(0,0,255,1.0)'; //Set a blue color
      ctx.fillRect(120, 10, 150, 150);//Fill a rectangle with the color
}
```

Drawing Triangle

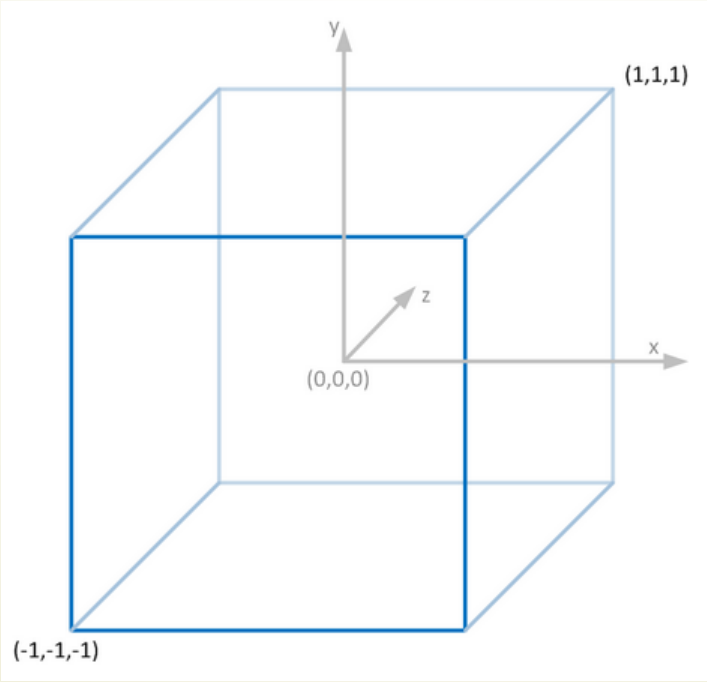


Drawing 2D Triangle



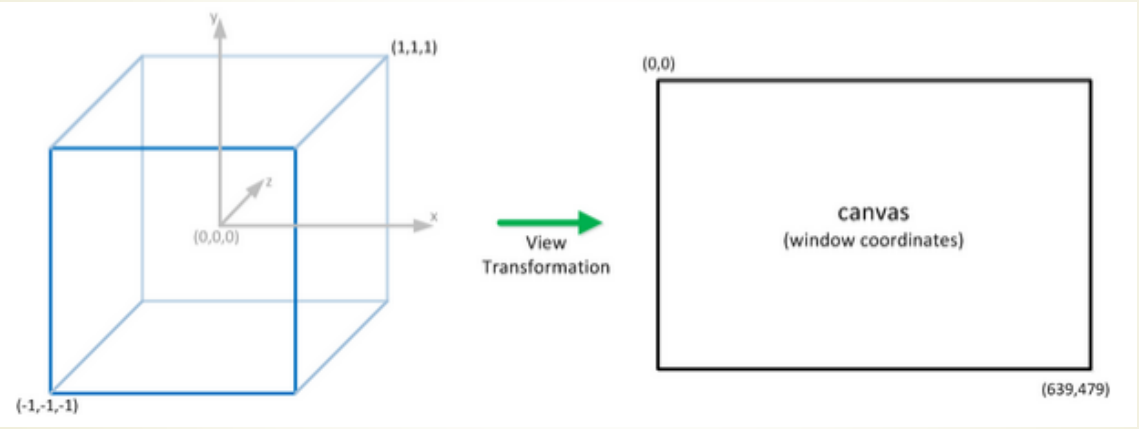
Drawing Object-Coordinate

- Normalized Device Coordinate(NDC)
  - Range (-1, 1) : x, y, z



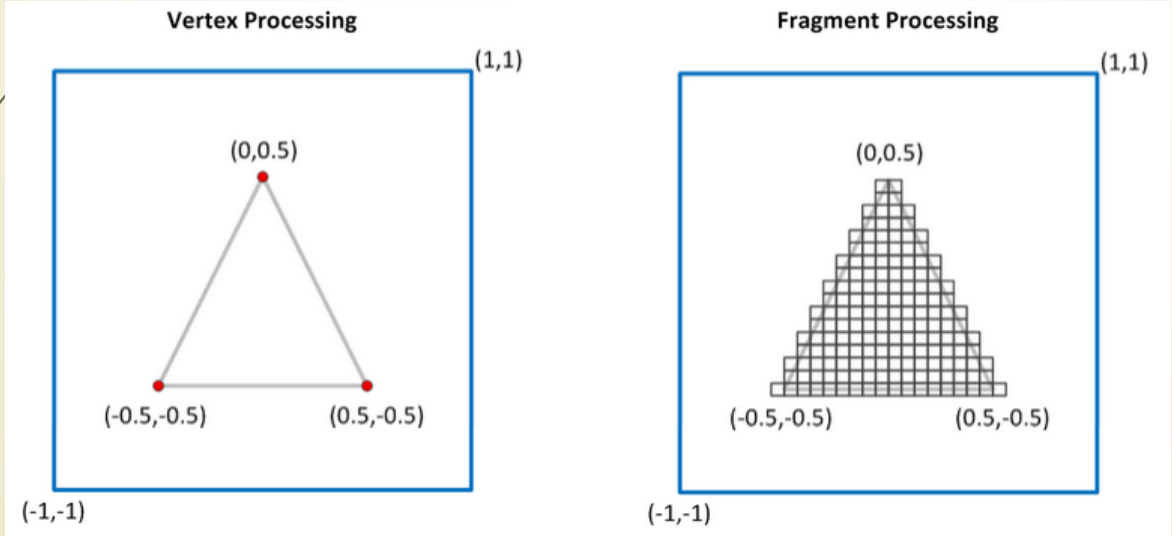
Drawing Object

- NDC->Window Coordinates



Shaders : Processing Vertices & Fragments

- Vertex shader : Transforming vertices
- Fragment shader : output color of the fragment



Start WebGL

1. Initialize WebGL
2. Get the vertex and the fragment shader source code from the DOM
3. Compile the shader
4. Create the program(to attach the shader)
5. Attach shader to program and link
6. WebGL uses the program
7. Setup buffers(data)
8. Draw a scene

Drawing Triangle

- 첨부한 코드에서 각 기능을 함수 형태로 정리해 보자.
  - 소스 프로그램 참조.

```
<html>
<head>

  <script id="vertexshader" type="x-shader">
    attribute vec2 aVertexPosition;

    void main() {
    gl_Position = vec4(aVertexPosition, 0.0, 1.0);
    }
  </script>

  <script id="fragmentshader" type="x-shader">
    #ifdef GL_ES
    precision highp float;
    #endif

    uniform vec4 uColor;

    void main() {
    gl_FragColor = uColor;
    }
  </script>
```

Shader

Drawing Triangle

```
<script type="text/javascript">
    var canvas=null, gl=null, v=null, f=null, vs=null, fs=null,vertices=null;
    function initWebGL()
    {
      canvas = document.getElementById("canvas");
      try{
        gl = canvas.getContext("webgl") || canvas.getContext("experimental-webgl");
      }catch(e){
      }
    }

    function main()
    {
      initWebGL();

      if(gl){
        setupWebGL(); //WebGL환경설정
        initShaders();     //Shader 초기화
        setupBuffers();   // drawing을 위한 버퍼 설정
        drawScene();    // 원하는 object drawing
      }else{
        alert("Error:Your browser does not appear to support WebGL.");
      }
    }

   function setupWebGL()
   {
        // Viewport
      gl.viewport(0, 0, canvas.width, canvas.height);
     // gl.clearColor(0, 0, 0, 1);
      gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
   }
```

Drawing Triangle

```
function initShaders()
{

  // Setup Shaders:문서(DOM)로 부터 vertex/fragment shader source code를 가져
  v = document.getElementById("vertexshader").firstChild.nodeValue;
  f = document.getElementById("fragmentshader").firstChild.nodeValue;
  //정점 세이더 생성
  vs = gl.createShader(gl.VERTEX_SHADER);
  //문서에서 정점세이더 소스코드 연결
  gl.shaderSource(vs, v);
  //세이더를 컴파일함
  gl.compileShader(vs);

  if (!gl.getShaderParameter(vs, gl.COMPILE_STATUS))
     alert(gl.getShaderInfoLog(vs));
  fs = gl.createShader(gl.FRAGMENT_SHADER);
  gl.shaderSource(fs, f);
  gl.compileShader(fs);

  if (!gl.getShaderParameter(fs, gl.COMPILE_STATUS))
     alert(gl.getShaderInfoLog(fs));


 //프로그램을 생성하고 그것을 세이더와 연결함
  program = gl.createProgram();
  gl.attachShader(program, vs);
  gl.attachShader(program, fs);
  gl.linkProgram(program);

}
```

옴

Drawing Triangle

```
function setupBuffers()
{
      if (!gl.getProgramParameter(program, gl.LINK_STATUS))
     alert(gl.getProgramInfoLog(program));

  // Setup Geometry : 삼각형을 위한 정점 정보 - 배열에 저장
  vertices = new Float32Array([
    -0.5,-0.5,0.5,-0.5,0.0,0.5  // Triangle-Coordinates
  ]);

  vbuffer = gl.createBuffer();
  gl.bindBuffer(gl.ARRAY_BUFFER, vbuffer);
  gl.bufferData(gl.ARRAY_BUFFER, vertices, gl.STATIC_DRAW);

}

function drawScene()
{
      itemSize = 2; // we have 2 coordinates (x,y)
  numItems = vertices.length / itemSize; // number of triangles


  // Setup Geometry:
  gl.useProgram(program);

  program.uColor = gl.getUniformLocation(program, "uColor");
  gl.uniform4fv(program.uColor, [1.0, 0.0, 0.0, 1.0]);

  program.aVertexPosition = gl.getAttribLocation(program, "aVertexPosition");
  gl.enableVertexAttribArray(program.aVertexPosition);
  gl.vertexAttribPointer(program.aVertexPosition, itemSize, gl.FLOAT, false, 0, 0);

  // Draw:
  gl.drawArrays(gl.TRIANGLES, 0, numItems);

}
</script>
```

Drawing Triangle

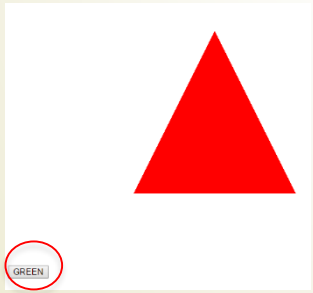
```
    </head>
<body onload="main()">
  <canvas id="canvas" width="512" height="512">
                Your browser does not support the  HTML5 canvas element.
  </canvas>
</body>
</html>
```

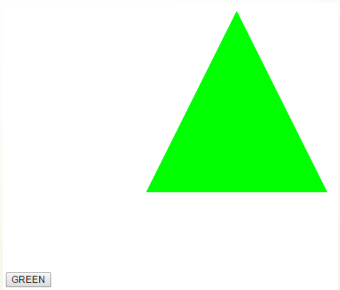
WebGL with Button

- Button을 만들어 보자.

```
<body onload="main()">
    <input type="button" name="GREEN" value = "GREEN" onclick="setColor('GREEN')"/>
    <canvas id="canvas" width="512" height="512">
                Your browser does not support the  HTML5 canvas element.
    </canvas>
</body>
```

- 버튼의 색깔로 바꾸기

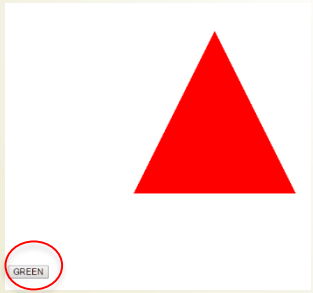


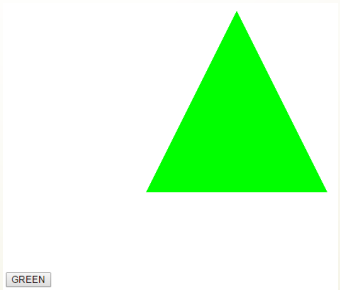


WebGL with Button

- 버튼의 색깔로 바꾸기 : color를 지정하는 변수 선언

```
// 삼각형 Color 변경 함수
var setColor=function(c){

   if(c=="GREEN"){
        T_color =[0.0, 1.0, 0.0, 1.0];
   }else{

   }

   drawScene();
}
```





Drawing Triangle

- 실습문제
  - 버튼을 클릭하면 색깔이 변하는 코드를 완성하시오.
- 실습문제
  - 버튼을 Red, Blue, Black를 추가하여 구현하시오.
- 도전문제
  - 서로 다른 삼각형을 추가해 보자.
  - 4변체를 그려보시오.