WEBGL INTRODUCTION

AGENDA

- GLSL でできることのデモ
- WebGL とは？
- OpenGL というのは...
- ということでWebGL はこんな感じ
- とにかく WebGL を使ってみる
- GPU??
- 三角形を描いてみる
- 2つのシェーダ
- uniform を使うと...
- まとめ
- 参考リンク

GLSL DEMO

http://glsl.heroku.com/

WHAT IS WEBGL?

- brawser で使える OpenGL (OpenGL = OpenなGraphics Library)
- HTML5 Canvas 要素を使います(Document Objec Model interfaces).

WHAT IS OPENGL?

C-lang 向けの 3D graphics API.

ex) 画面消去

```
glClearColor(0, 0, 0, 1); // 消去色のRGBA指定
glClear(GL_COLOR_BUFFER_BIT); // 色バッファを消去実行
```

WHAT IS WEBGL2

browserでもOpenGLが使いたい！ -> JavaScriptからOpenGLを呼び出せるようにした.

```
context.clearColor(0,0,0,1);
context.clear(context.COLOR_BUFFER_BIT);
```

TRY WEBGL

step-1: 描画先になるCanvasオブジェクトを取得する

```
var canvas = document.getElementById('webgl-canvas');
```

step-2: レンダリングコンテキストを取得する

```
var gl = canvas.getContext('webgl');
```

step-3: WebGLの命令を発行する

```
gl.clearColor(0, 0, 0, 1); // 消去色のR,G,B,A指定
gl.clear(gl.COLOR_BUFFER_BIT); // 色バッファを消去実行
```

DEMO TRY WEBGL

Demo http://jsbin.com/marabudovevo/1
http://jsbin.com/gunidu/1

CPU AND GPU

GPU = Graphics Processing Unit

```
  (CPU) <--> (MEM)
    ^
--- | GraphicsAPI ----- 世界の壁
    v
  (GPU) <--> (GPUMEM)
```

GPU

- 周りのデータを参照できない
- 周りにデータを書き込めない
- 演算後のデータを見れない

```
演算元データ          演算後データ
[ ] -> (演算器) -> [ ]
[ ] -> (演算器) -> [ ]
[ ] -> (演算器) -> [ ]
[ ] -> (演算器) -> [ ]
[ ] -> (演算器) -> [ ]
[ ] -> (演算器) -> [ ]
[ ] -> (演算器) -> [ ]
[ ] -> (演算器) -> [ ]
```

DRAW TRIANGLES

```
// 1.描画先になるCanvasオブジェクトを取得する
var canvas = document.getElementById('webgl-canvas');
// 2.レンダリングコンテキストを取得する
var gl = canvas.getContext('webgl');

gl.clearColor(0, 0, 0, 1);
gl.clear(gl.COLOR_BUFFER_BIT);

/*
 * 演算器の準備
 * GPU のプログラムをセットアップする
 */
// シェーダの用意
var vertexShaderSource = '\
  precision mediump float;\
  attribute vec2 vertex;\
  void main() {\
    gl_Position = vec4(vertex, 0.0, 1.0);\
```

DRAW TRIANGLES - COORDINATE SYSTEM

- canvas要素(描画領域)の中心が (0, 0)
- 2つの値で1つの頂点の座標 (x, y)
- 3つの座標で1つの三角形
- 三角形が6つあります

```
var vertexData = [
  -0.6,-0.4, -0.6,0.4, 0.6,-0.4,
  0.6,-0.4, 0.6,0.4, -0.6,0.4,
  -0.6,0.1, -0.7,0, -0.6,-0.1,
  0.6,0.1, 0.7,0, 0.6,-0.1,
  -0.4,-0.4, -0.3,-0.5, -0.2,-0.4,
  0.2,-0.4, 0.3,-0.5, 0.4,-0.4
];
```

DRAW TRIANGLES - VERTEX NUM

```
gl.drawArrays(gl.TRIANGLES, 0, 18);
```

- 0番目の頂点から、18 個の頂点を使う
- 指定するのは "頂点の個数" (三角形の個数ではない)
- TRIANGLES を他の値にすると描画形式が変わる
  - http://www.khronos.org/opengles/sdk/docs/man/xhtml/glDraw
  - http://d.hatena.ne.jp/nakamura001/20081231/1230719279
- 用意した座標データの個数と指定頂点の数をあわせること

DRAW TRIANGLES - COLOR

- (R, G, B, A) で色を指定します
- OpenGL での色指定はRGBのみ

```
var fragmentShaderSource = '\
  precision mediump float;\
  void main() {\
    gl_FragColor = vec4(1.0, 0.7, 0.6, 1.0);\
  }\
';
```

DEMO TRIANGLES

ここ

TWO SHADERS

- 用意したデータの数 = 頂点の数
- 出力の結果 = 絵 = データの数 = pixel数

用意した頂点データと出力データ数が異なる理由

UNIFORM

シェーダプログラムに
外部からパラメータを指定することができる.

1. シェーダソースに uniform を記述する
2. getUniformLocation() でハンドルを取得する
3. uniform*() で値を設定する

UNIFORM NOTATION

uniform の記述例

```
precision mediump float;
uniform vec4 color; // uniform の宣言
void main() {
  gl_FragColor = color; // uniform を使用
}
```

GET UNIFORM VARIABLES

ハンドルを保存する

- シェーダ内の名前と合わせること
- shaderProgram は有効なシェーダプログラムでなければならない

```
var locColor =
    gl.getUniformLocation(
        shaderProgram,
        'color'
    );
```

SET UNIFORM

uniform に値を設定する

- 引数
  - 取得しておいたハンドル
  - 設定したい値

```
gl.useProgram(shaderProgram); //必須

gl.uniform4f(
    locColor,
    0.3, 0.9, 0.3, 1.0
);
```

DRAW SHADERS

```
var canvas = document.getElementById('webgl-canvas');
var gl = canvas.getContext('webgl');

gl.clearColor(0, 0, 0, 1);
gl.clear(gl.COLOR_BUFFER_BIT);

// シェーダの用意
var vertexShaderSource = '\
    precision mediump float;\
    uniform vec2 offset;\
    attribute vec2 vertex;\
    void main() {\
        gl_Position = vec4(vertex + offset, 0.0, 1.0);\
    }\
';

var fragmentShaderSource = '\
    precision mediump float;\
```

DEMO - DRAW SHADERS

ここ

CONCLUSION

- WebGL ≒ OpenGL ES 2.0
- WebGL は canvas オブジェクトからレンダリングコンテキストを取得して使う
- GPU のお仕事は流れ作業
  - 同じ作業を大量のシェーダで並列に行うのが基本(事前準備大事)
  - 周りのデータを参照できない、周りに書き込むことができない、演算後のデータを参照できない
- WebGL で三角形を描画するには座標を指定する
- uniform を使うことでシェーダプログラムに外部から値を与えることができる

SPECIFICATION DOCUMENTS

WEBGL

http://www.khronos.org/webgl/

特に https://www.khronos.org/registry/webgl/specs/1.0/

OPENGL ES 2.0

http://www.khronos.org/opengles/2_X/

特に http://www.khronos.org/opengles/sdk/docs/man/