Đồ họa máy tính
Giới thiệu về HTML5 và WebGL

GV: Lý Quốc Ngọc

Nhóm: VPN

Lưu Thị Hồng Ngọc	**19127051**
Hồ Thiên Phước	19127517

So sánh WebL, OpenGL ES và OpenGL

	WebGL	OpenGL	OpenGL ES
Khái niệm	Thư viện đồ hoạ cho web được dẫn xuất từ OpenGL ES (thư viện đồ họa 2D và 3D trên hệ thống nhúng: xe cơ giới, điện thoại, đồ điện tử)	Thông qua ngôn ngữ chéo và platfrom API để render vector graphic 2D và 3D	Phiênn bản OpenGL dành cho các hệ thốnng nhúng, tiêu biểu là thiết bị di động
Môi trường	Thường được sử dụng trên nền tảng browser và web	Thường được sử dụng trên nền tảng desktop	Thường được sử dụng trê n nền tảng mobile
Ngôn ngữ	Java Script	C, C++, Python, Java	C/C++, Java, Kotlin
Tính năng	Ít tính năng	Nhiều tính năng hơn 2 cơ chế hoạt động còn lại.	Do là embedded của OpenGL nênn số lượng feature ít hơn. Hỗ trợ 1 số feature khác lập trình mobile tốt hơn
Pipeline	No fixed function pipeline	fixed function pipeline	x

HTML5

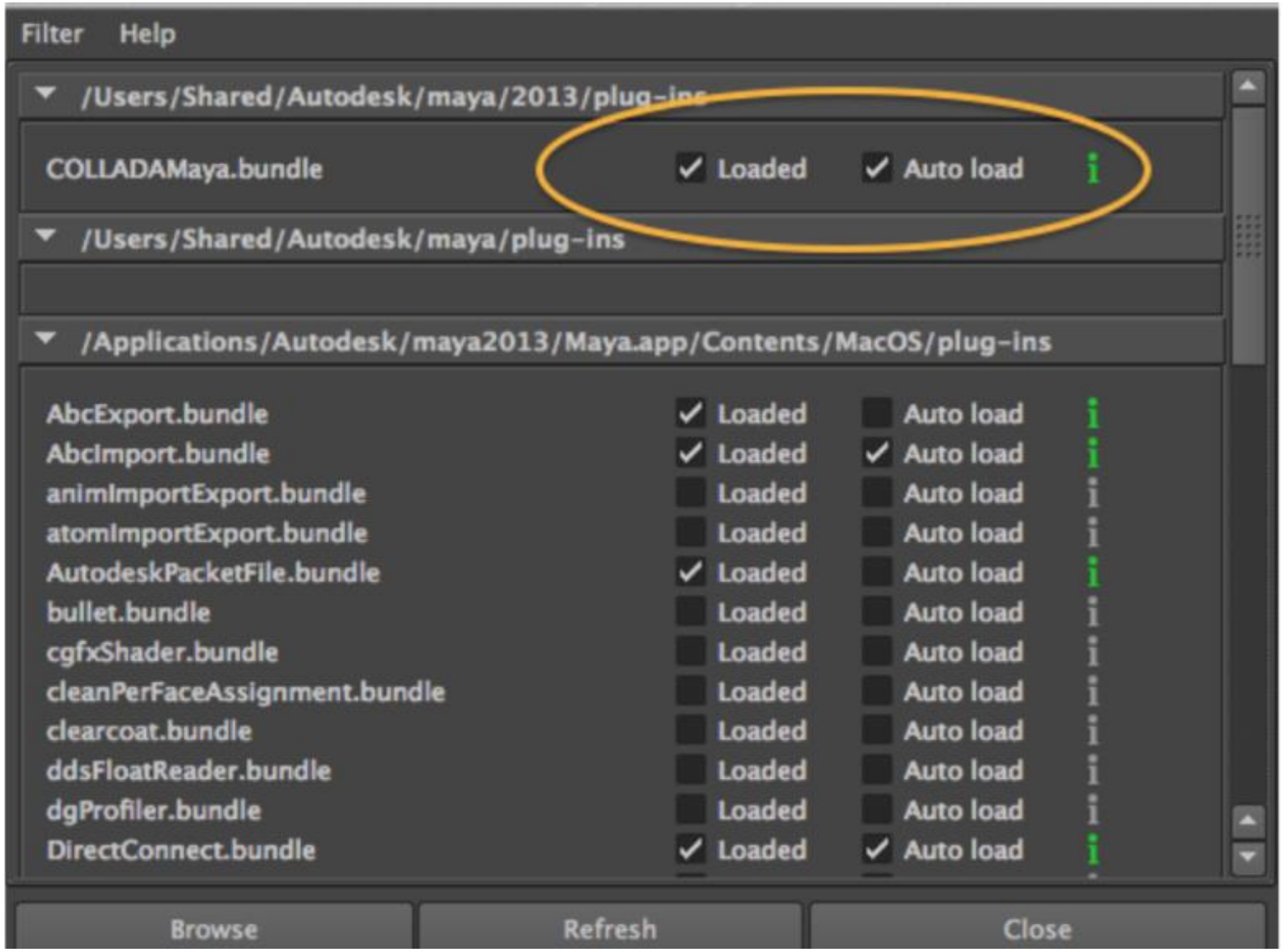
- Thiết kế các trang web trực quan
- Tạo ra các thư viện đồ họa giúp tạo ra các ứng dụng đồ thị, game trong cả môi trường 2D và 3D như những ứng dụng trên desktop thông qua Javascript.
- Không cần sử dụng hay yêu cầu người dùng cài đặt bất kì thư viện nào

Cài đặt

Các công cụ sử dụng: Autodesk Maya, Vizi, …

Các cài đặt AutoDesk Maya 2013:

- Tải file AutoDesk Maya 2013
- Exporting the Maya Scene to COLLADA
  - Sau khi cài exporter, vào Plug-in Manager (Window → Settings/Preferences → Plug-in Manager) để xác nhận đã tải lên hay chưa


Filter
Help
/Users/Shared/Autodesk/maya/2013/plug-ins
COLLADAMaya.bundle
Loaded
Auto load
/Users/Shared/Autodesk/maya/plug-ins
/Applications/Autodesk/maya2013/Maya.app/Contents/MacOS/plug-ins
AbcExport.bundle
AbcImport.bundle
animImportExport.bundle
atomImportExport.bundle
AutodeskPacketFile.bundle
bullet.bundle
cgfxShader.bundle
cleanPerFaceAssignment.bundle
clearcoat.bundle
ddsFloatReader.bundle
dgProfiler.bundle
DirectConnect.bundle
Browse
Refresh
Close

- Chuyển COLLADA File sang glTF
  - Mở Terminal, gõ các dòng lệnh như sau

```
 $ <path-to-converter>/collada2gltf -f futurgo.dae -d
    [option] export pass details
    converting:futurgo.dae ... as futurgo.json
    [shader]: futurgo0VS.glsl
    [shader]: futurgo0FS.glsl
    [shader]: futurgo2VS.glsl
    [shader]: futurgo2FS.glsl
    [shader]: futurgo4VS.glsl
    [shader]: futurgo4FS.glsl
    [completed conversion]
```

Cơ chế hoạt động

Bộ hàm thư viện quan trọng

Three.js

Để cài đặt Three.js, vào repository Github để tải về package
https://github.com/mrdoob/three.js/

2D

- Vẽ hình học (đoạn thẳng, cung tròn, Biezer)
- Tô màu
- Phép biến đổi (tịnh tiến, xoay,)

Vẽ đoạn thẳng

Ví dụ: hàm vẽ đoạn thẳng:

```
function draw(canvas, context)
{
    context.lineTo(x2, y2);
}
```

- Trong đó x2, y2 là tọa độ điểm cuối của đoạn thẳng

Vẽ cung tròn

Ví dụ: hàm vẽ cung tròn:

```
    function draw(canvas, context){
  context.arc(x, y, radius, startAngle, endAngle, bool
counterclockwise);
}
```

- Trong đó:
  - x,y là tọa độ tâm cung tròn
  - radius là bán kính cung tròn
  - startAngle, endAngle lần lượt là góc bắt đầu và góc kết thúc của cung tròn trên bán kính, tính từ trục dương của Ox
  - counterclockwise: biến Boolean, xác định chiều vẽ ngược chiều hay cùng chiều kim đồng hồ (mặc định là false, cùng chiều kim đồng hồ)

Vẽ đường cong Biezer

Ví dụ: hàm vẽ đường cong Biezer:

```
        function draw(canvas, context){
context.bezierCurveTo(x2,y2,x3,y3,x4,y4);
}
```

- Trong đó:
  - x2,y2,x3,y3,x4,y4 lần lượt là các điểm thứ 2, 3, 4 của đường cong Biezer, điểm đầu tiên được xác định là điểm mới nhất trong thời điểm hiện tại, có thể thay đổi bằng cách sử dụng hàm moveto()

Tô màu

Ví dụ: hàm tô màu:

```
        function draw(canvas, context){
context.fill();
}
```

Phép biến đổi- tịnh tiến

Ví dụ: hàm tịnh tiến:

```
function draw(canvas, context){
context.translate(x, y);
}
```

$$P(x,y) \rightarrow P^{'}(x',y')$$

$$P' = T(t_x, t_y) . P$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Phép biến đổi- xoay

Ví dụ: hàm xoay:

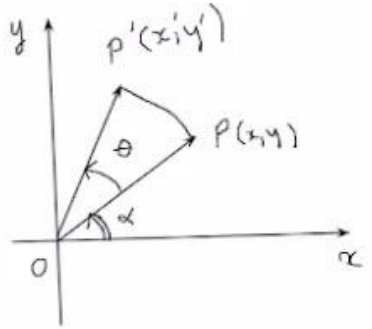
```
function draw(canvas, context){
    context.rotate(angle);
}
```

Pivot point: $O(o,o)$

$P(x,y) \rightarrow P'(x',y')$

$P' = R(\theta).P$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \text{Cos}\theta & -\text{Sin}\theta & 0 \\ \text{Sin}\theta & \text{Cos}\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



Phép biến đổi- co dãn

Ví dụ: hàm co dãn:

```
function draw(canvas, context){
    context.scale(x,y);
}
```

Pivot point: $O(o,o)$

$$P(x,y) \rightarrow P'(x',y')$$

$$P' = S(O, s_x, s_y) \,.\, P$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

3D

Tạo đối tượng 3D

Hiển thị đối tượng 3D

Các phép biến đổi (tịnh tiến, xoay, co dãn)

Các phép chiếu (chiếu song song, chiếu bối cảnh, chiếu sáng, khử mặt khuất)

TẠO VÀ HIỂN THỊ ĐỐI TƯỢNG 3D

- B1: Tạo cấu trúc dữ liệu gồm đỉnh, cạnh và các polygon. (Ví dụ: hàm đặc tả đối tượng)

```
var cube = {buffer:vertexBuffer, colorBuffer:colorBuffer, indices:cubeIndexBuffer,
vertSize:3, nVerts:24, colorSize:4, nColors: 24, nIndices:36, primtype:gl.TRIANGLES};
```

- B2: Xác định mặt phẳng chiếu (hướng nhìn, tọa độ camera).

```
// Add  a camera so we can view the scene
camera = new THREE.PerspectiveCamera( 45,
  canvas.width / canvas.height, 1, 4000 );
scene.add(camera);
```

TẠO VÀ HIỂN THỊ ĐỐI TƯỢNG 3D

- B3: Xác định phép chiếu (song song, phối cảnh)
- B4: Từ tọa độ thế giới thực, mép vào tọa độ màn hình
- B5: Vẽ

3D
Phép biến đổi- tịnh tiến

```
.translate {
    -webkit-transform: translateX(20px) translateY(20px)
translateZ(-100px);
       -moz-transform: translateX(20px) translateY(20px)
translateZ(-100px);
         -o-transform: translateX(20px) translateY(20px)
translateZ(-100px);
            transform: translateX(20px) translateY(20px)
translateZ(-100px);
}
```

$$P(x,y,z) \rightarrow P'(x',y',z')$$

$$P' = T(t_x,t_y,t_z) \cdot P$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

3D
Phép biến đổi - xoay

$$R(\vec{V},\theta) = T^{-1}(P_1\vec{O})\,A(\vec{V})^{-1}\,R(Oz,\theta)\,A(\vec{V})\,T(P_1\vec{O})$$

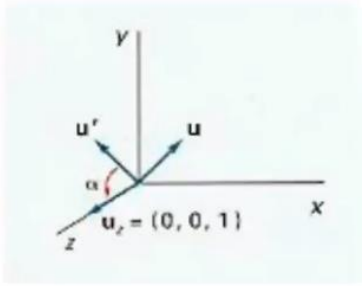
$$A(\vec{V}) = R(Ox,\alpha)R(Oy,\beta)$$

$$A(\vec{V})^{-1} = R(Ox,\alpha)^{-1}R(Oy,\beta)^{-1}$$

$$\cos\alpha = \frac{\vec{u'}.\vec{u_z}}{|u'|.|u_z|} = \frac{c}{d},\quad (\vec{u'} = (0,b,c)), d = \sqrt{b^2+c^2}$$

$$\left.\begin{aligned}\vec{u'}\times\vec{u_z} &= \vec{u_x}.|u'|.|u_z|.\sin\alpha\\ \vec{u'}\times\vec{u_z} &= \vec{u_x}.b\end{aligned}\right\} \Rightarrow \sin\alpha = \frac{b}{d}$$

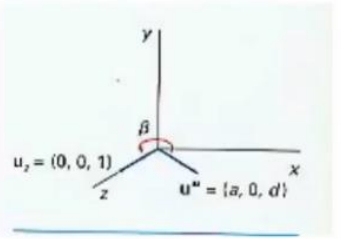
$$R(Ox,\alpha) = \begin{bmatrix}1 & 0 & 0 & 0\\ 0 & c/d & -b/d & 0\\ 0 & b/d & c/d & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$



$$\cos\beta = \frac{\vec{u''}.\vec{u_z}}{|u''|.|u_z|} = d,\quad (\vec{u'} = (a,0,d)), d = \sqrt{b^2+c^2}$$

$$\left.\begin{aligned}\vec{u''}\times\vec{u_z} &= \vec{u_y}.|u''|.|u_z|.\sin\beta\\ \vec{u''}\times\vec{u_z} &= \vec{u_y}.(-a)\end{aligned}\right\} \Rightarrow \sin\beta = -a$$

$$R(Oy,\beta) = \begin{bmatrix}d & 0 & -a & 0\\ 0 & 1 & 0 & 0\\ a & 0 & d & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$



$$R(Oz,\theta) = \begin{bmatrix}\cos\theta & -\sin\theta & 0 & 0\\ \sin\theta & \cos\theta & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

3D
Phép biến đổi- co dãn

```
.scale {
        -webkit-transform: scaleX(1.25) scaleY(.75);
           -moz-transform: scaleX(1.25) scaleY(.75);
             -o-transform: scaleX(1.25) scaleY(.75);
                transform: scaleX(1.25) scaleY(.75);
}
```

Pivot point: $O(o,o,o)$

$$P(x,y,z) \rightarrow P'(x',y',z')$$

$$P' = S(O, s_x, s_y, s_z) \cdot P$$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Projection Line: $\vec{V}(a,b,c)$

View Plane: $(R_0,\vec{N})$

$$P' = Par(\vec{V},(R_0,\vec{N})).P$$

$$Par(\vec{V},(O,Oz) = \begin{bmatrix} 1 & 0 & -\frac{a}{c} & 0 \\ 0 & 1 & -\frac{b}{c} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$A(\vec{N}) = \begin{bmatrix} \frac{\lambda}{|\vec{N}|} & \frac{-n_1n_2}{\lambda|\vec{N}|} & \frac{-n_1n_3}{\lambda|\vec{N}|} & 0 \\ 0 & \frac{n_3}{\lambda} & -\frac{n_2}{\lambda} & 0 \\ \frac{n_1}{|\vec{N}|} & \frac{n_2}{|\vec{N}|} & \frac{n_3}{|\vec{N}|} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \lambda = \sqrt{n_2^2 + n_3^2}$$

$$T(\vec{R_0O}) = \begin{bmatrix} 1 & 0 & 0 & -x_0 \\ 0 & 1 & 0 & -y_0 \\ 0 & 0 & 1 & -z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$Par(\vec{V},(R_0,\vec{N})) = T(\vec{OR_0}).A^{-1}(\vec{N}).Par(\vec{V},(O,Oz).A(\vec{N}).T(\vec{R_0O})$$

3D
Phép chiếu - song song

3D
Phép chiếu - phối cảnh

$$Per(C,(R_0,\vec{N})) = T^{-1}(\vec{CO}).Per(O,(R_0,\vec{N})).T(\vec{CO})$$

$$= \begin{bmatrix} 1 & 0 & 0 & c_x \\ 0 & 1 & 0 & c_y \\ 0 & 0 & 1 & c_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} d & 0 & 0 & 0 \\ 0 & d & 1 & 0 \\ 0 & 0 & d & 0 \\ n_1 & n_2 & n_3 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 1 & -c_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Projection Reference Point: $C(c_x, c_y, c_z)$

View Plane: $(R_0, \vec{N}), R_0(x_0, y_0, z_0), \vec{N}(n_1, n_2, n_3)$

$$P' = Per(C,(R_0,\vec{N})).P$$

$$\begin{bmatrix} x_H \\ y_H \\ z_H \\ H \end{bmatrix} = \begin{bmatrix} d + c_x.n_1 & c_x.n_2 & c_x.n_3 & -c_x.d_0 \\ c_y.n_1 & d + c_y.n_2 & c_y.n_3 & -c_y.d_0 \\ c_z.n_1 & c_z.n_2 & d + c_z.n_3 & -c_z.d_0 \\ n_1 & n_2 & n_3 & -d_1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$d_0 = n_1x_0 + n_2y_0 + n_3z_0 \quad d_0 = n_1.c_x + n_2.c_y + n_3.c_z$$

$$d = d_0 - d_1$$

WinToView

$$P(x,y) \rightarrow P'(x',y')$$
$$P' = WTV.P$$
$$s_x = \frac{v_{max}.x - v_{min}.x}{w_{max}.x - w_{min}.x}$$
$$s_y = \frac{v_{max}.y - v_{min}.y}{w_{max}.y - w_{min}.y}$$
$$t_x = v_{min}.x - w_{min}.x$$
$$t_y = v_{min}.y - w_{min}.y \Rightarrow WTV = S(v_{min}, s_x, s_y).T(t_x, t_y)$$
$$\Rightarrow WTV = \begin{pmatrix} s_x & 0 & v_{min}.x - s_x.v_{min}.x \\ 0 & s_y & v_{min}.y - s_y.v_{min}.y \\ 0 & 0 & 1 \end{pmatrix} . \begin{pmatrix} 1 & 0 & v_{min}.x - w_{min}.x \\ 0 & 1 & v_{min}.y - w_{min}.y \\ 0 & 0 & 1 \end{pmatrix}$$
$$\Rightarrow \begin{pmatrix} s_x & 0 & v_{min}.x - s_x.w_{min}.x \\ 0 & s_y & v_{min}.y - s_y.w_{min}.y \\ 0 & 0 & 1 \end{pmatrix}$$

Môi trường đồ họa

Render WebGL vào một trang hoặc ứng dụng

Gồm các bước:

- B1: Tạo canvas và vẽ bằng context (Khởi động môi trường đồ họa)

```
function initWebGL(canvas) {
        var gl = null;
        var msg = "Your browser does not support WebGL, " +
            "or it is not enabled by default.";
        try
        {
            gl = canvas.getContext("experimental-webgl");
}
catch (e) {
            msg = "Error creating WebGL Context!: " + e.toString();
        }
if (!gl) {
alert(msg);
            throw new Error(msg);
        }
return gl; }
```

Render WebGL vào một trang/ứng dụng

- B2: Khởi tạo Viewport

```
function initViewport(gl, canvas)
    {
        gl.viewport(0, 0, canvas.width, canvas.height);
    }
```

Render WebGL vào một trang/ứng dụng

- B3: Tạo các buffer để render (Chuẩn bị dữ liệu tronng buffer)

```
// CREATE THE VERTEX DATA FOR A SQUARE TO BE DRAWN
    FUNCTION CREATESQUARE(GL) {
        VAR VERTEXBUFFER;
        VERTEXBUFFER = GL.CREATEBUFFER();
        GL.BINDBUFFER(GL.ARRAY_BUFFER, VERTEXBUFFER);
        VAR VERTS = [
             .5,  .5,  0.0,
            -.5,  .5,  0.0,
             .5, -.5,  0.0,
            -.5, -.5,  0.0
        ];
        GL.BUFFERDATA(GL.ARRAY_BUFFER, NEW
FLOAT32ARRAY(VERTS), GL.STATIC_DRAW);
        VAR SQUARE = {BUFFER:VERTEXBUFFER, VERTSIZE:3,
NVERTS:4,
            PRIMTYPE:GL.TRIANGLE_STRIP};
        RETURN SQUARE;
}
```

Render WebGL vào một trang/ứng dụng

- B4: Tạo một hoặc nhiều ma trận để xác định phép biến đổi từ vùng đệm đỉnh sang không gian màn hình (đẩy ra memory card, buffer màn hình)

```
    function initMatrices(canvas)
    {
        // Create a model view matrix with camera at 0, 0, -3.333
        var modelViewMatrix = mat4.create();
        mat4.translate(modelViewMatrix, modelViewMatrix, [0, 0, -3.333]);
        // Create a project matrix with 45 degree field of view
        var projectionMatrix = mat4.create();
        mat4.perspective(projectionMatrix, Math.PI / 4,
            canvas.width / canvas.height, 1, 10000);
}
```

```
      function createShader(gl, str, type) {
       var shader;
       if (type == "fragment") {// pixel màu của mỗi đỉnh
sau khi biến đổi
 shader = gl.createShader(gl.FRAGMENT_SHADER);
       } else if (type == "vertex") {// chuyển toạ độ object
thành kgian hiển thị 2D
           shader = gl.createShader(gl.VERTEX_SHADER);
       } else {
           return null;
       }
       gl.shaderSource(shader, str);
       gl.compileShader(shader);
       if (!gl.getShaderParameter(shader,
gl.COMPILE_STATUS)) {
           alert(gl.getShaderInfoLog(shader));
           return null;
}return shader;
    }
```

Render WebGL vào một trang/ứng dụng

- B5: Tạo một hoặc nhiều shader để thực hiện thuật toán vẽ.

Render WebGL vào một trang/ứng dụng

- B6: Khởi tạo shader với các tham số.

```
    FUNCTION INITSHADER(GL) {
        // LOAD AND COMPILE THE FRAGMENT AND VERTEX
SHADER
        VAR FRAGMENTSHADER = CREATESHADER(GL,
FRAGMENTSHADERSOURCE,
           "FRAGMENT");
        VAR VERTEXSHADER = CREATESHADER(GL,
VERTEXSHADERSOURCE,
"VERTEX");
        // LINK THEM TOGETHER INTO A NEW PROGRAM
       VAR SHADERPROGRAM = GL.CREATEPROGRAM();
        GL.ATTACHSHADER(SHADERPROGRAM, VERTEXSHADER);
        GL.ATTACHSHADER(SHADERPROGRAM, FRAGMENTSHADER);
        GL.LINKPROGRAM(SHADERPROGRAM);
        // GET POINTERS TO THE SHADER PARAMS
        VAR SHADERVERTEXPOSITIONATTRIBUTE =
           GL.GETATTRIBLOCATION(SHADERPROGRAM,
"VERTEXPOS");
        GL.ENABLEVERTEXATTRIBARRAY(SHADERVERTEXPOSITIONA
TTRIBUTE);
        VAR SHADERPROJECTIONMATRIXUNIFORM =
          GL.GETUNIFORMLOCATION(SHADERPROGRAM,
"PROJECTIONMATRIX");
        VAR SHADERMODELVIEWMATRIXUNIFORM =
          GL.GETUNIFORMLOCATION(SHADERPROGRAM,
"MODELVIEWMATRIX");
        IF (!GL.GETPROGRAMPARAMETER(SHADERPROGRAM,
           GL.LINK_STATUS)) {
           ALERT("COULD NOT INITIALISE SHADERS");
} }
```

Render WebGL vào một trang/ứng dụng

- B7: Vẽ

```
function draw(gl, obj) {
    // clear the background (with black)
    gl.clearColor(0.0, 0.0, 0.0, 1.0);
    gl.clear(gl.COLOR_BUFFER_BIT);
    // set the vertex buffer to be drawn
    gl.bindBuffer(gl.ARRAY_BUFFER, obj.buffer);
    // set the shader to use
    gl.useProgram(shaderProgram);
    // connect up the shader parameters: vertex position
    // and projection/model matrices
    gl.vertexAttribPointer(shaderVertexPositionAttribute,
        obj.vertSize, gl.FLOAT, false, 0, 0);
    gl.uniformMatrix4fv(shaderProjectionMatrixUniform, false,
        projectionMatrix);
    gl.uniformMatrix4fv(shaderModelViewMatrixUniform, false,
        modelViewMatrix);
    // draw the object
    gl.drawArrays(obj.primtype, 0, obj.nVerts);
  }
```

Cảm ơn thầy và các bạn đã lắng nghe