TRƯỜNG ĐẠI HỌC THỦY LỢI

**KHOA CÔNG NGHỆ THÔNG TIN**

**--------------------------------------------------------**



**BÁO CÁO BÀI TẬP LỚN MÔN HỌC**

**Lập Trình 3D-Webgl**

***Đề tài***

**3DOVERWEB**

**Nhóm sinh viên thực hiện:** Nhóm 8- lớp 59TH3.01

**Thành viên nhóm:**  Nguyễn Thanh Tùng

Đào Ninh Thái

Vũ Ngọc Anh

Nguyễn Văn Tuấn

**Giảng viên phụ trách môn học:** **Trần Thị Minh Hoàn**

*Hà Nội, tháng 12 năm 2020*

Mục Lục

[I. CƠ SỞ LÝ THUYẾT 3](#_Toc60946618)

[1. Đề tài về 3DOVERWEB 3](#_Toc60946619)

[2. Lý Thuyết 3](#_Toc60946620)

[a. Triển Khai 3](#_Toc60946621)

[b. Biến Attribute, Uniform, Varying 3](#_Toc60946622)

[c. Model Martrix 4](#_Toc60946623)

[d. Hàm Tick 4](#_Toc60946624)

[e. Bộ đệm chỉ số 4](#_Toc60946625)

[f. Điểm Nhìn 5](#_Toc60946626)

[g. Chiếu Sáng 5](#_Toc60946627)

[II. Cấu trúc trong chương trình mẫu 6](#_Toc60946628)

[1. var VSHADER\_SOURCE = 6](#_Toc60946629)

[2. var FSHADER\_SOURCE = 7](#_Toc60946630)

[3. Hàm Main 9](#_Toc60946631)

[4. Các lệnh bổ sung 9](#_Toc60946632)

[III. Kết Luận 10](#_Toc60946633)

[1. Lời Cảm Ơn 10](#_Toc60946634)

[2. Tài Liệu Tham Khảo 10](#_Toc60946635)

# CƠ SỞ LÝ THUYẾT

## Đề tài về 3DOVERWEB

Đề Tài trình bày về việc biểu diện một đối tượng 3D trên môi trường Web thông qua Webgl. Việc làm này là rất đơn giản và không thể chối cãi với webgl.

Ý tưởng của bài tập lớn lần này là thể thiện một đối tượng 3D với những kiến thức được học qua các chương.

## Lý Thuyết

### Triển Khai

- Để triển khai Webgl và thẻ canvas ta sử dụng notepad tạo 1 file bất kỳ và lưu dưới dạng .html, sau đó triển khai môi trường web bằng HTML5. Ta sử dụng lệnh sau để triển khai canvas lưu ý trình duyệt sử dụng phải hỗ trợ canvas:

*<canvas id=”webgl” style=”background: url(‘../resources/633.jpg’)” width=”400” height=”400”></canvas>*

Với việc sử dụng ảnh định dạng cho background canvas đã hiển thị với background bằng texture với ảnh tên “*633.jpg*”

- Tiếp đó ta triển khai các lệnh *button* với điều kiện *onclick* để tương tác với vật thể 3D, xây dựng nội dung cơ bản của web với thẻ *<p>*, tiếp đó liên kết tới thư viện và gọi ra thư mục javascript cần truy vấn tên là *“3DoverWeb.js”.*

- Xây dựng Javascript với *Vshader* và *Fshader* với Vshader là chương trình mô tả tính trạng (màu sắc, vị trí của một đỉnh) còn Fshader là chương trình xử lý tửng mảng ảnh.

### Biến Attribute, Uniform, Varying

- Attribute khác Uniform ở điểm Uniform đều nhận giá trị như nhau còn attribute mỗi giá trị nhận 1 giá trị khác nhau, còn Varying là biến chuyển đổi.

- Các bước triển khai 1 biến Uniform

*B1: 'uniform mat4 u\_MvpMatrix;\n' +*

*B2: var u\_MvpMatrix = gl.getUniformLocation(gl.program, 'u\_MvpMatrix');*

*B3: gl.uniformMatrix4fv(u\_MvpMatrix, false, mviewProjMatrix.elements);*

- Các bước triển khai 1 biến Attribute

*B1: 'attribute vec4 a\_Position;\n' +*

*B2: var a\_attribute = gl.getAttribLocation(gl.program, a\_Position);*

*B3: gl.vertexAttribPointer(a\_ Position, num, type, false, 0, 0);*

*B4:* ***gl.enableVertexAttribArray(a\_ Position);***

### Model Martrix

- Ta sử dụng công cụ mà Webgl cung cấp sẵn để thực hiện các phép di chuyển, phóng to, xoay, thông qua các cú pháp trong hàm tick, nhớ rằng việc bổ sung *set* ở đầu sẽ xóa hoàn toàn ma trận trước đó, việc thực hiện nhân theo thứ tự từ dưới lên trên.

*modelMatrix.setRotate(currentAngle, 1, 0, 0);* nhân với ma trận xoay

*modelMatrix.setTranslate(currentAngle, 0, 0);* nhân với ma trận dịch chuyển

*modelMatrix.setScale(currentAngle,0,0);* nhân với ma trận dịch chuyển

### Hàm Tick

- Đối với *Tick* ta cần chú ý tới

*currentAngle = animate(currentAngle); //* dùng để sử dụng hàm *animate*

*draw(gl, n, currentAngle, viewProjMatrix, u\_MvpMatrix); //* dùng để vẽ lại hình

*requestAnimationFrame(tick, canvas); //* yêu cầu trình duyệt gọi *tick*

- Tiếp đó với *animate* đây là hàm để cập nhật góc quay

Chương trình duy trì góc quay hiện tại của tam giác, tam giác đã được xoay bao nhiêu độ từ vị trí ban đầu của nó trong biến *currentAngle* nó tính toán góc tiếp theo giựa trên giá trị này. Cập nhật *currentAngle* được thực trong *animate* biểu diễn góc quay hiện tại và trả về góc mới.

### Bộ đệm chỉ số

- Bộ đệm chỉ số là 1 tính năng trong Webgl giúp người dùng đặt chỉ số từ bộ đệm đối tượng, ở bài tập lớn này bộ đệm chỉ số sẽ biểu diễn cho nhiều điểm để không bị trùng màu do đó sẽ phải khai báo lại các điểm giống nhau, sẽ trình bày tiếp tại II.

- 5 bước để thực hiện bộ đệm đối tượng

*B1: tạo một bộ đệm đối tượng (gl.createBuffer()).*

*B2: gắn bộ đệm đối tượng vào 1 target (gl.blindbuffer()).*

*B3: lưu dữ liệu vào bộ đệm đối tượng (gl.bufferdata()).*

*B4: gán bộ đệm thuộc tính (gl.vertexAttripointer())).*

*B5: kích hoạt lệnh gán (gl.enableVertexAttribArray()).*

### Điểm Nhìn

- Về hướng nhìn ,ta quan tâm đến các thông số sau :Điểm mắt,điểm nhìn ,và hướng đỉnh.Các thông số trên là nhưng yếu tố để hình thành nên hướng nhìn.

Trong WebGl ta có câu lệnh thiết lập hướng nhìn như sau:

*Matrix4.setLookAt(x1,y1,z1,x2,y2,z2,x3,y3,z3)*

Trong đó *x1,y1,z1* là tọa độ của điểm mắt *x2,y2,z2* là điểm mà ta nhìn đến còn ***x3,y3,z3*** là hướng đỉnh đầu.

- Khối quan sát hình chóp tứ giác biểu diễn cho việc hình ảnh ở xa sẽ bị nhỏ đi mặc dù nó to hơn thứ mà chúng ta thấy trước.

*viewProjMatrix.setPerspective(fov, aspect, near, far);*

* *Fov* : phạm vi quan sát góc tạo bởi mặt phẳng đỉnh top và bottom lớn hơn 0
* *Aspect*: chỉ ra tỉ lệ cảnh của mặt cắt gần (width/height)
* *Near, far*: Chỉ ra khoảng cách cho mặt cắt gần và xa dọc theo đường ngắm.

Vì đối tượng được đặt trong 3D ,nên sẽ xảy ra tình trạng vật ở trước vật ở sau,nếu chúng ta không thực hiện khử mặt khất thì sẽ dẫn đến hiện tượng xung đột z

Các bước khử mặt khuất

- Kích hoạt hàm khử mặt khuất

*gl.enable(gl.DEPTH\_TEST)*

- Xóa bộ đệm chiều sâu được sử dụng để khử mặt khuất trước khi vẽ *gl.clear(gl.DEPTH\_BUFFER\_BIT)*

### Chiếu Sáng

- Có 3 loại nguồn sáng cơ bản sau:

* *Ánh sáng hướng*: tức là nguồn sáng lúc này ở xa vô tận, hướng sáng sẽ là các đường song song với nhau, VD: ánh sáng từ mặt trời
* *Ánh sáng nguồn*: nguồn sáng lúc này sẽ là 1 điểm xác định trong không gian, hướng sáng sẽ là các đường đi từ điểm sáng hướng ra mọi phía được xác định qua nguồn sáng và vị trí chiếu trên bề mặt, VD: bóng đèn, cây nến,
* *Ánh sánh xung quanh*: còn gọi là ánh sáng gián tiếp, là mô hình ánh sáng được phát ra từ nguồn sáng khác, được phản chiếu bởi các đối tượng như tường, và đi đến đối tượng 1 cách gián tiếp, ánh sáng này ko có hướng và vị trí xác định.

- Các kiểu ánh sáng phản chiếu

* Phản chiếu khếch tán: là sự phản chiếu của đối tượng từ ánh sáng hướng hoặc ánh sáng điểm
* Phản chiếu xung quanh: là sự phản chiếu của đối tượng từ ánh sáng xung quanh
* Khi 1 đối tượng được chiếu sáng bởi ánh sáng hướng, diểm, xung quanh thì màu sắc của nó chính là màu của phản chiếu khuếch tán và phản chiếu xung quanh hợp lại

- chúng ta sử dụng công thức

1. *<Surface color by diffuse refelection> = <light color> x <base color or surface> x <light direction>. <orientation of a surface>*
2. *<Surface color by ambient refelection> = <light color> x <base color or surface>*
3. *<Surface color by diffuse and ambient refelection> = <Surface color by diffuse*

*refelection>+<Surface color by ambient refelection>*

sẽ bổ sung chứng minh tiếp với II

# Cấu trúc trong chương trình mẫu

## var VSHADER\_SOURCE =

***'attribute vec4 a\_Position;\n' +***

*- Khai báo biến thuộc tính a\_Position,biến này lưu trữ các đỉnh của hình lập phương*

***'attribute vec4 a\_Color;\n' +***

*- Khai báo biến thuộc tính a\_Color,biến này lưu trữ màu sắc của các đỉnh hình lập phương*

***'attribute vec4 a\_Normal;\n' +***

*- Khai báo biến thuộc tính a\_Normal,biến này là ma trận thẻ hiện vecto pháp tuyến của đỉnh hình lập phương*

***'uniform mat4 u\_MvpMatrix;\n' +***

*- Khai báo biến đồng nhất u\_MvpMatrix,biến này là 1 ma trận là tích của ma tra biến đổi ,ma trận điểm nhìn ,và ma trận không gian nhìn*

***'uniform mat4 u\_ModelMatrix;\n' +***

*- Khai báo biến đồng nhất u\_Matrix,biến này là 1 ma trận thể hiện cho các phép biến đổi của hình lập phương*

***'uniform mat4 u\_NormalMatrix;\n' +***

*- Khai báo biến đồng nhất u\_NormarMatrix,biến này là 1 ma trận là chuyển vị nghịch đảo của các phép biến đổi*

***'uniform bool u\_Clicked;\n' +***

***-*** *Kiểm tra khi click chuột*

***'varying vec4 v\_Color;\n' +***

*- Khai báo biến thay đổi v\_Color ,biến này ta sử dụng để như trung gian để truyền a\_Color sang Fshader*

***'varying vec3 v\_Normal;\n' +***

*- Khai báo biến thay đổi v\_Normal,biến này ta sử dụng để như trung gian để truyền a\_Normal sang Fshader*

***'varying vec3 v\_Position;\n' +***

*- Khai báo biến thay đổi v\_Posiotion,biến này ta sử dụng để như trung gian để truyền a\_Position sang Fshader*

***'void main() {\n' +***

***' gl\_Position = u\_MvpMatrix \* a\_Position;\n' +***

*- Ta gán đỉnh của hình lập phương sau khi biến đổi cho gl.Position*

***' v\_Position = vec3(u\_ModelMatrix \* a\_Position);\n' +***

*- Tính v-Position ,đây là vecto thể hiện tọa độ của đỉnh sau khi thực hiện các phép biến đổi*

***' v\_Normal = normalize(vec3(u\_NormalMatrix \* a\_Normal));\n' +***

*- Tính v\_Normal ,đây là vecto thể hiện tọa độ của vecto pháp tuyến sau khi thực hiện các phép biến đổi*

***' if (u\_Clicked) {\n' +***

***' v\_Color = vec4(0.0, 0.0, 0.0, 1.0);\n' +***

*- Khi click tự động chuyển màu đen*

## var FSHADER\_SOURCE =

***'#ifdef GL\_ES\n' +***

***'precision mediump float;\n' +***

*- Chỉ ra độ chính xác*

***'#endif\n' +***

***'uniform vec3 u\_LightColor;\n' +***

*- Khai báo biến đồng nhất biểu diễn cho màu của ánh sáng nguồn*

***'uniform vec3 u\_LightPosition;\n' +***

*- Khai báo biến đồng nhất thể hiện vị trí của nguồn sáng*

***'uniform vec3 u\_AmbientLight;\n' +***

*- Khai báo biến đồng nhất thể hiện màu của ánh sáng xung quanh*

***'varying vec3 v\_Normal;\n' +***

*- Khai báo biến thay đổi thể sử dụng v\_normal được truyền từ Vshader*

***'varying vec3 v\_Position;\n' +***

*- Khai báo biến thay đổi để sử dụng v\_position được truyền từ Vshader*

***'varying vec4 v\_Color;\n' +***

*- Khai báo biến thay đổi để sử dụng v\_Color được truyền từ Vshader*

***'void main() {\n' +***

***' vec3 normal = normalize(v\_Normal);\n' +***

*- Chuẩn hóa normal thành vecto đơn vị*

***' vec3 lightDirection = normalize(u\_LightPosition - v\_Position);\n' +***

*- Tính hướng sáng của nguồn sáng dựa theo các đỉnh của hình lập phương sau khi được biến đổi*

***' float nDotL = max(dot(lightDirection, normal), 0.0);\n' +***

*- Tính tích vô hướng của hướng sáng và pháp tuyến của đỉnh,nếu ấm sẽ cho kết quả bằng 0*

***' vec3 diffuse = u\_LightColor \* v\_Color.rgb \* nDotL;\n' +***

*- Tính ánh sáng của phản xạ khếch tán*

***' vec3 ambient = u\_AmbientLight \* v\_Color.rgb;\n' +***

*- Tính ánh sáng của phản xạ xung quanh*

***' gl\_FragColor = vec4(diffuse + ambient, v\_Color.a);\n' +***

*- Tính màu của đối tượng sau khi được tô bóng ,màu này sẽ là hợp của màu ánh sáng phản xạ khếch tán và ánh sáng phản xạ xung quanh*

***'}\n';***

*\*lưu ý 1*: bởi vì ánh sảng điểm phát ra nguồn sáng theo mọi hướng từ vị trí của nó, hướng ánh sáng tại một đỉnh là kết quả của việc trừ đi vì trí đỉnh từ vị trí nguồn sáng. Bởi vì vị trí nguồn sáng được truyền cho biến ***u\_LightPosition***, hướng ánh sáng được tính lại bởi ***lightDirection = normalize(u\_LightPosition - v\_Position);*** và sau đó màu bề mặt của mỗi đỉnh được tính toán dựa trên hướng ánh sáng này.

***\*****lưu ý 2:* hệ thống Webgl sẽ nội suy màu giữa các đỉnh giựa trên màu mà bạn cung cấp cho các đỉnh. Tuy nhiên vì hướng chiếu sáng từ nguồn sáng điểm thay đổi theo vị trí để tô bóng tự nhiên, bạn phải tính màu ở mọi vị trí mà ánh sáng chiếu tới thay vì chỉ ở đỉnh.

## Hàm Main

//tại hàm tick

***mviewProjMatrix.set(viewProjMatrix).multiply(modelMatrix);***

*- Dùng để kết hợp giữa phép chiếu phối cảnh và ma trận biến đổi*

***normalMatrix.setInverseOf(modelMatrix);***

*- Tính toán nghịch đảo của ma trận được lưu trữ trong m và lưu trữ kết quả trong đối tượng* ***Martrix4*** *trong đó* ***modelMatrix*** *là* ***Matrix4***

***normalMatrix.transpose();***

*- Chuyển ma trận được lưu trữ trong đối tượng* ***Matrix4*** *và ghi kết quả trở lại đối tượng* ***Matrix4***

*//tại initVertexBuffers*

***if (!initArrayBuffer(gl, 'a\_Position', vertices, 3, gl.FLOAT)) return -1;***

***if (!initArrayBuffer(gl, 'a\_Color', colors, 3, gl.FLOAT)) return -1;***

***if (!initArrayBuffer(gl, 'a\_Normal', normals, 3, gl.FLOAT)) return -1;***

*- Dữ Liệu được gán tại 3 phần* ***vertices, colors, normals*** *với biến attribute đã khai báo bên trên lấy 3 dữ liệu.*

## Các lệnh bổ sung

***canvas.onmousedown = function(ev)***

***function check***

*- Hai lệnh trên sử dụng để khi làm event click chuột*

***document.getElementById('change\_view').addEventListener('click',function(){***

*- Hàm dùng để chỉnh hướng nhìn*

***document.getElementById('change\_light').addEventListener('click',function(){***

*- Hàm dùng để chỉnh hướng chiếu sáng*

***document.getElementById('change\_hc').addEventListener('click',function(){***

*- Hàm dùng để thay đổi hoạt cảnh xoay*

***document.getElementById('change\_hc1').addEventListener('click',function(){***

*- Hàm dùng để thay đổi hoạt cảnh tịnh tiến*

***document.getElementById('change\_hc2').addEventListener('click',function(){***

*- Hàm dùng để thay đổi hoạt cảnh scale*

# Kết Luận

## Lời Cảm Ơn

Kết thúc môn đồ họa 3D trên webgl nhóm chúng em xin chân thành cảm ơn cô Hoàn đã hướng dẫn chi tiết môn học khiến chúng em năm được kiến thức cơ bản để có thể hoàn thành bài tập lớn này rất mong trong tương lai bọn em sẽ tiếp tục được đồng hành cùng cô.

## Tài Liệu Tham Khảo

* giáo trình môn học
* tài liệu sách lập trình webgl
* vd code ch10 và các bài tập thực hành