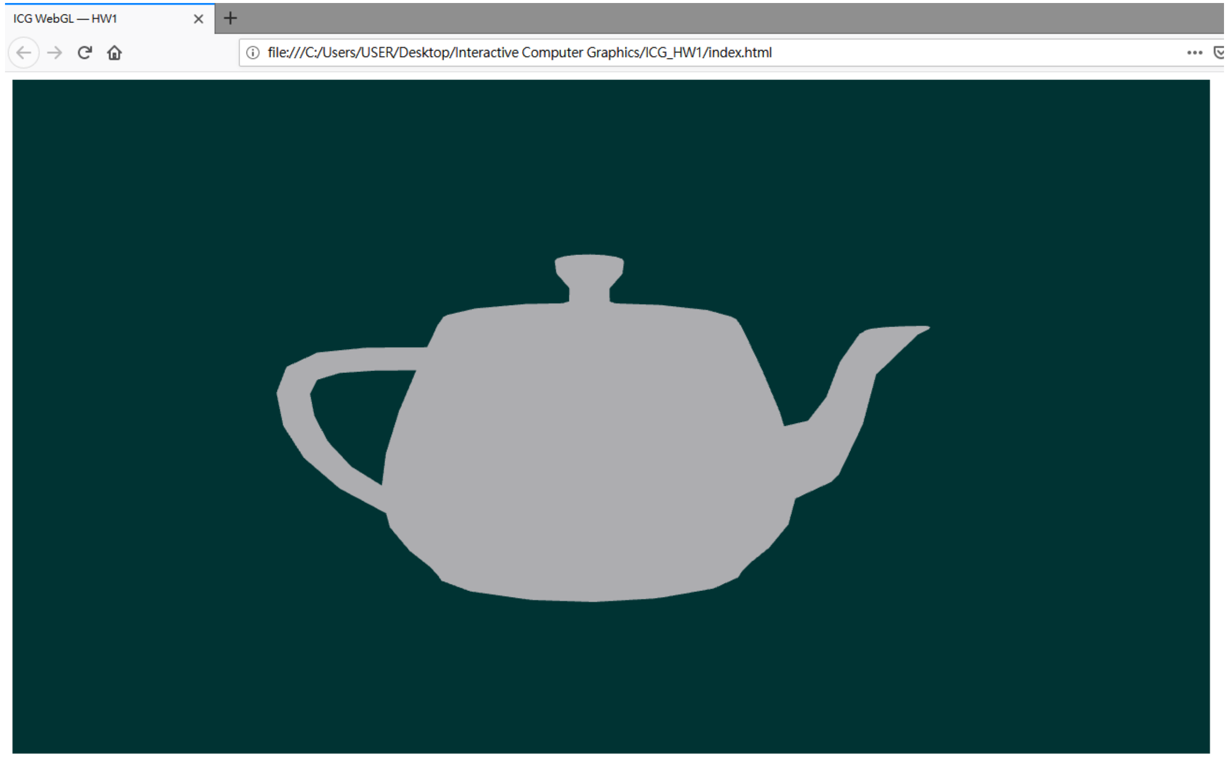
ICG 2020 Spring Homework1 Guidance 2020/03/31

網媒所碩二　周家宇

Environment Setup

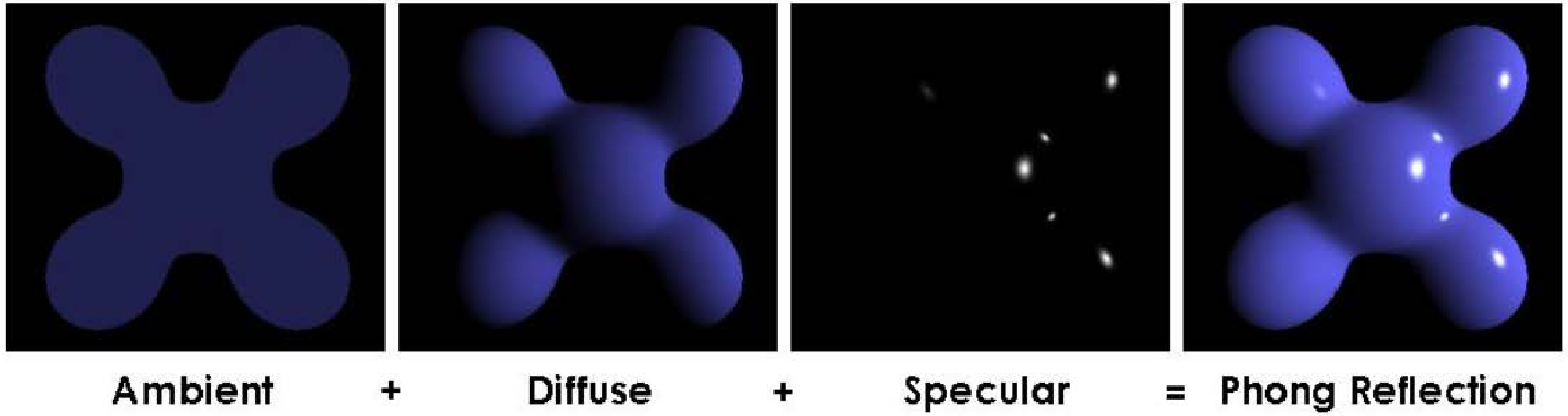
- Download sample code from course website
- Follow steps in **HW1_Guide.pdf** file


ICG WebGL — HW1
file:///C:/Users/USER/Desktop/Interactive Computer Graphics/ICG_HW1/index.html

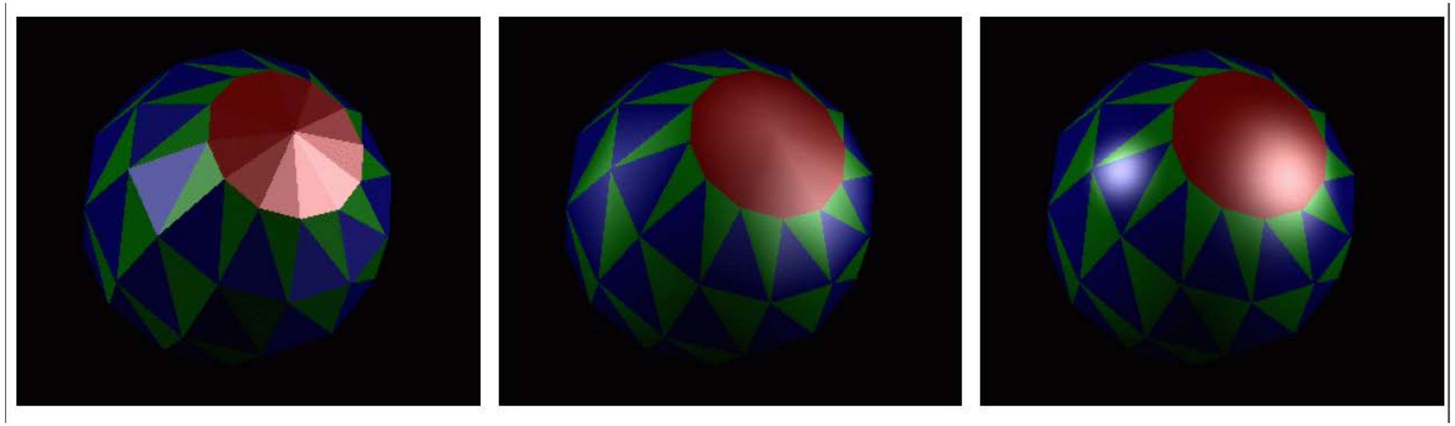
Requirements　(Due to 2020/05/12)

- Implement **Flat, Gouraud, and Phong shading with Phong reflection model** in shaders.
- **Enable multiple transformations (four fundamental transformations) on objects in a scene**. You are free to use those provided model files and arrange them to form the scene on your own style.
- At least **3 objects** & at least **3 light sources**
- Bonus: Special effects on shading / lighting / animation, …

Phong Reflection Model

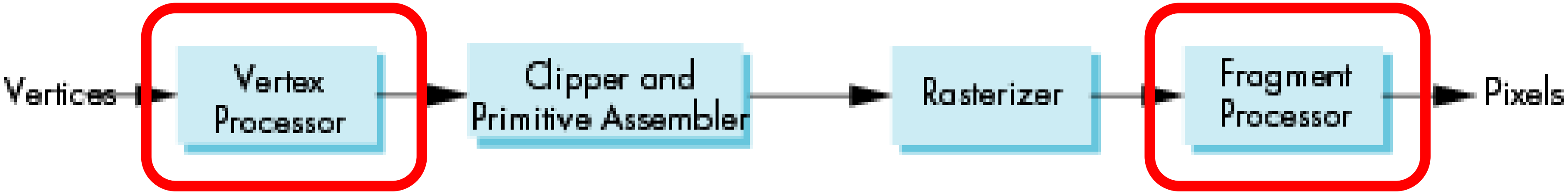


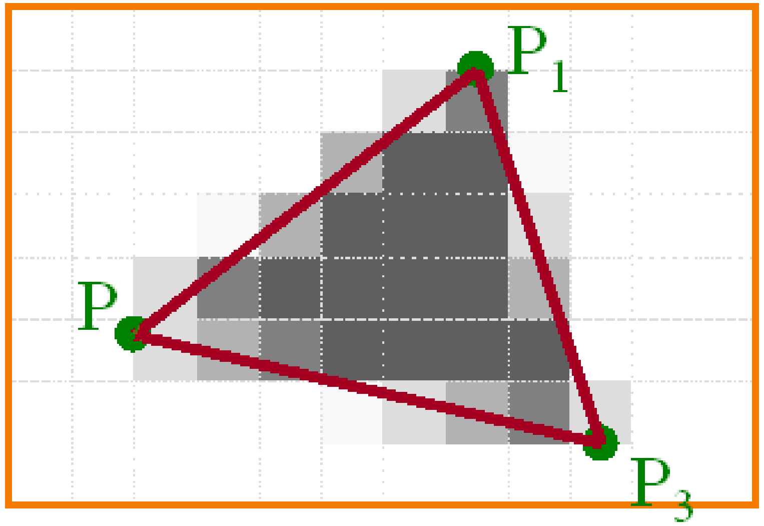
Shading



- Flat Shading: Constant normal on the whole surface
- Gouraud Shading: Different vertex normal, interpolated vertex color on a fragment
- Phong Shading: Different vertex normal, interpolated vertex normal on a fragment

Rendering Pipeline

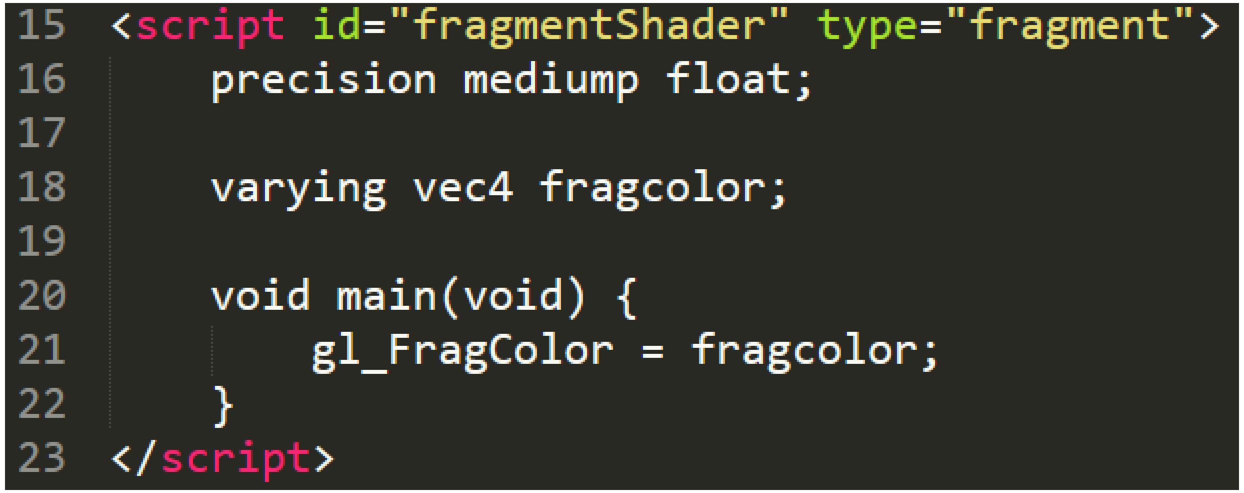




Graphics API & Shader Language

Graphics API	Shader Language
OpenGL / WebGL	GLSL (OpenGL Shading Language)
DirectX	HLSL (High Level Shading Language)
Vulkan	SPIR-V

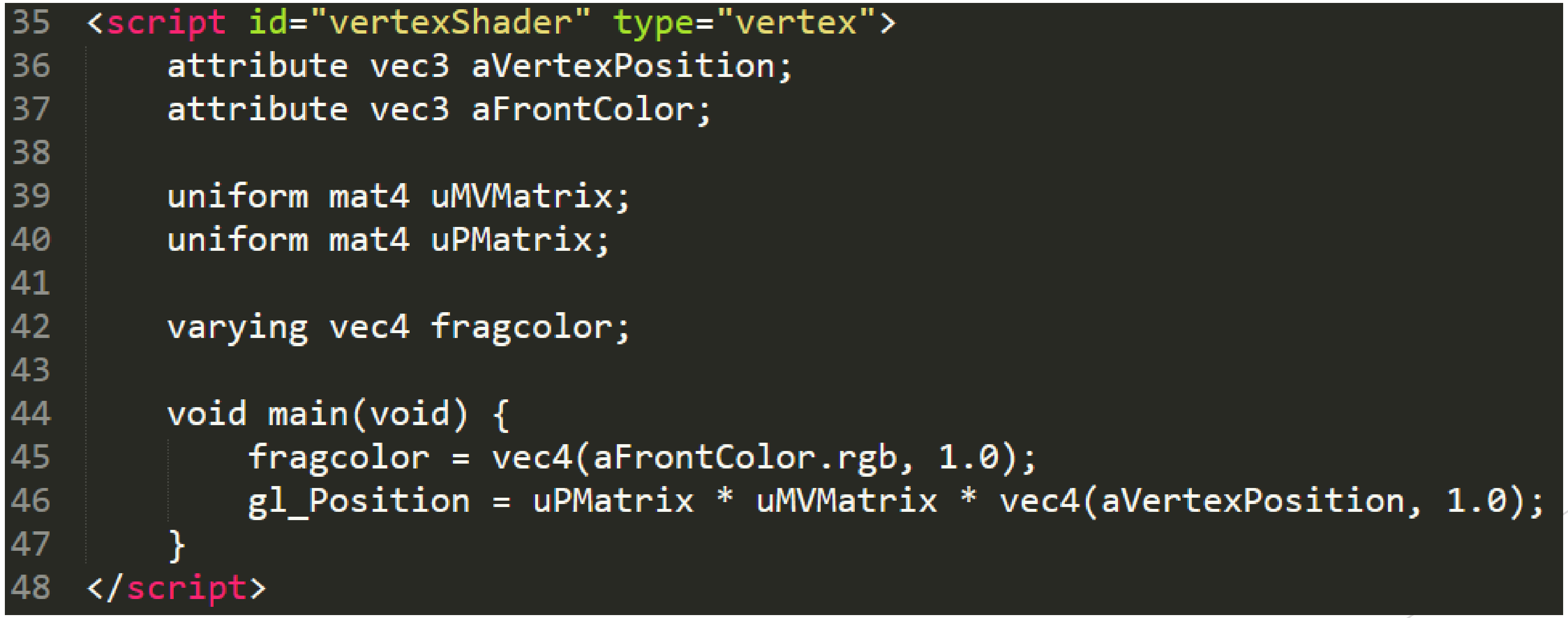
Shader (GLSL)



```
<script id="fragmentShader" type="fragment">
    precision mediump float;

    varying vec4 fragcolor;

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



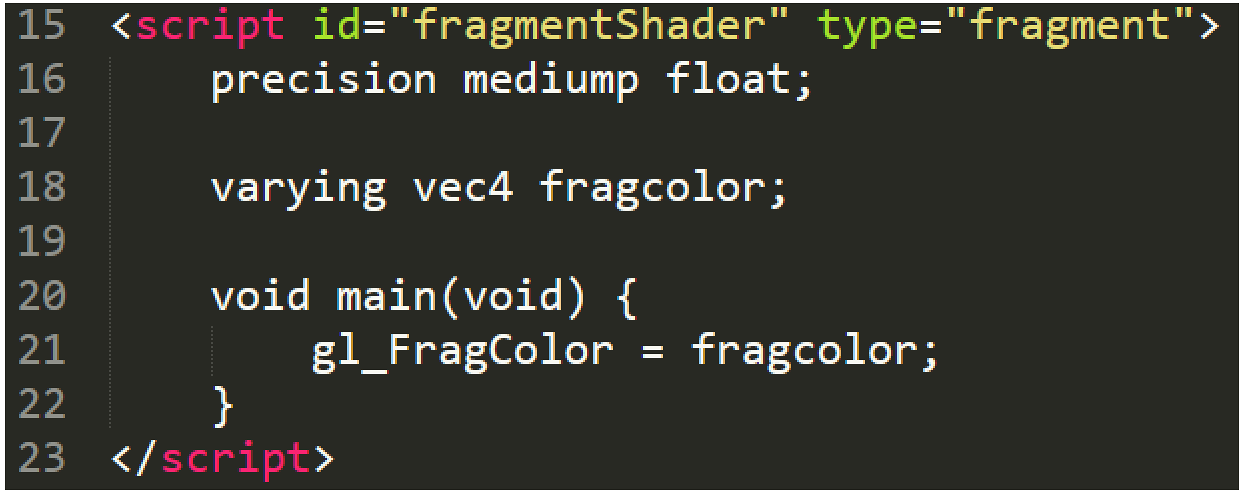
```
<script id="vertexShader" type="vertex">
    attribute vec3 aVertexPosition;
    attribute vec3 aFrontColor;

    uniform mat4 uMVMatrix;
    uniform mat4 uPMatrix;

    varying vec4 fragcolor;

    void main(void) {
        fragcolor = vec4(aFrontColor.rgb, 1.0);
        gl_Position = uPMatrix * uMVMatrix * vec4(aVertexPosition, 1.0);
    }
</script>
```

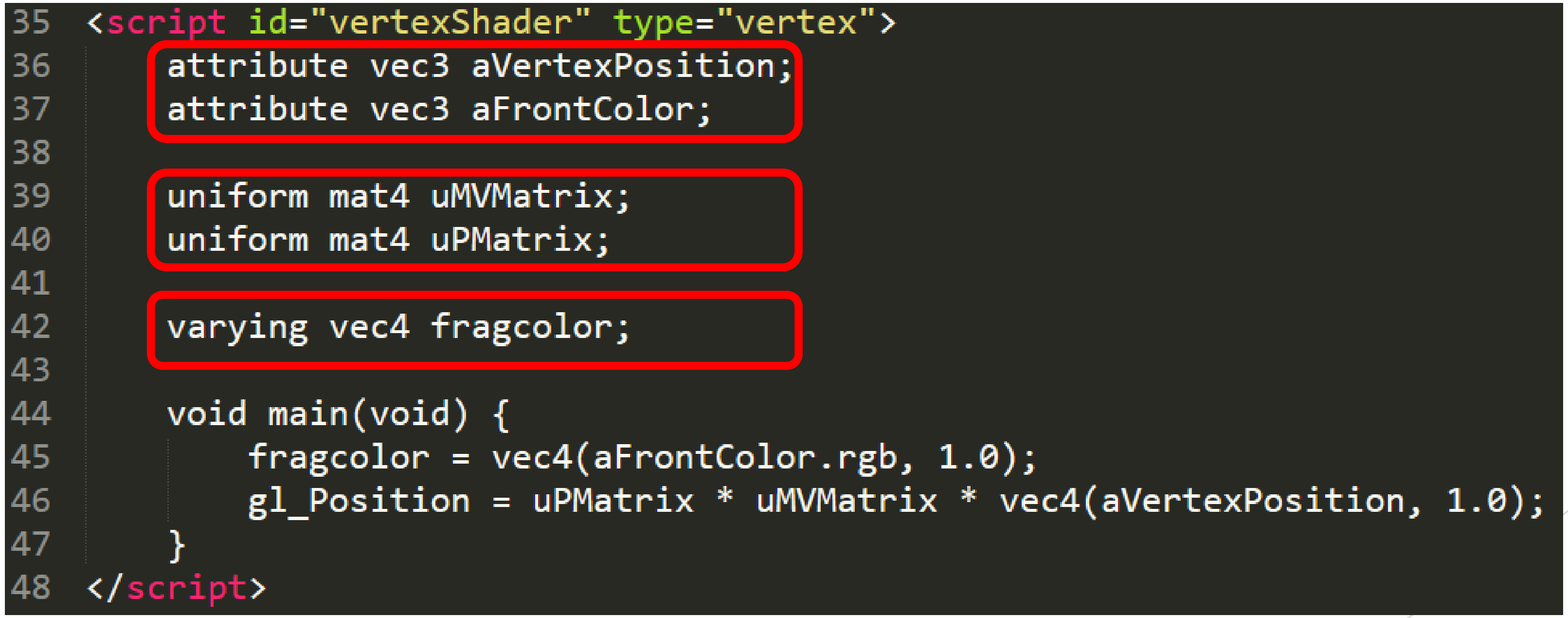
Shader Data (1/2)



```
<script id="fragmentShader" type="fragment">
    precision mediump float;

    varying vec4 fragcolor;

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



```
<script id="vertexShader" type="vertex">
    attribute vec3 aVertexPosition;
    attribute vec3 aFrontColor;

    uniform mat4 uMVMatrix;
    uniform mat4 uPMatrix;

    varying vec4 fragcolor;

    void main(void) {
        fragcolor = vec4(aFrontColor.rgb, 1.0);
        gl_Position = uPMatrix * uMVMatrix * vec4(aVertexPosition, 1.0);
    }
</script>
```

Shader Data (2/2)

SHADER DATA

“Per-object constant”

Uniform

= Shared Constant

Vertex Data

= ANYTHING YOU WANT!

Example?

Positions…

Normals…

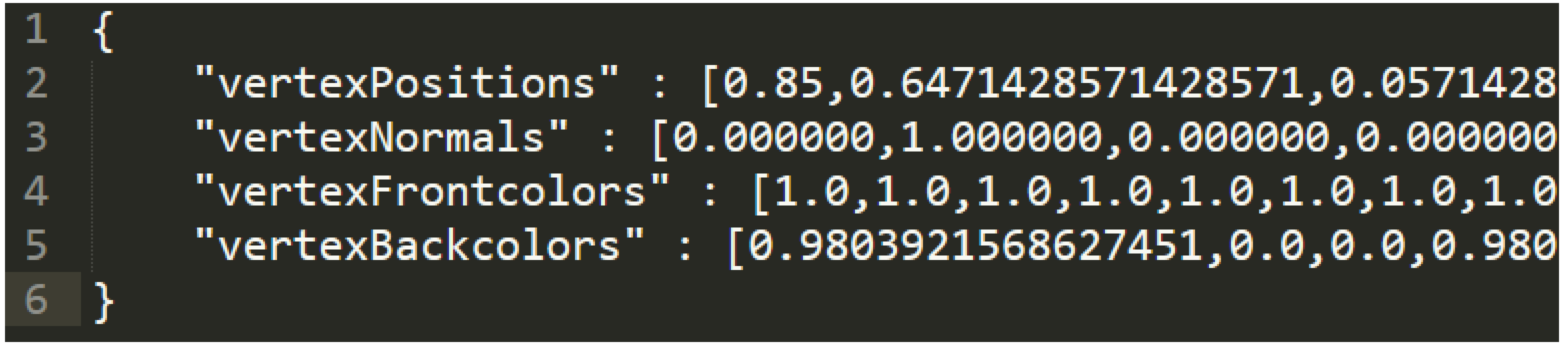
Colors…

Texture Coordinates…

Load Models

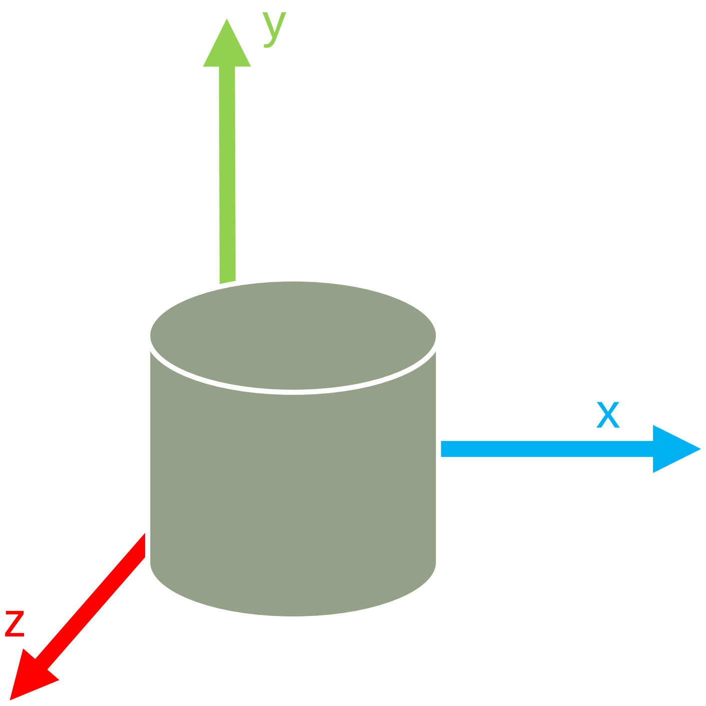
- 已經將大部分課程網的 tri 模型轉成 json 檔

Example　Csie.json



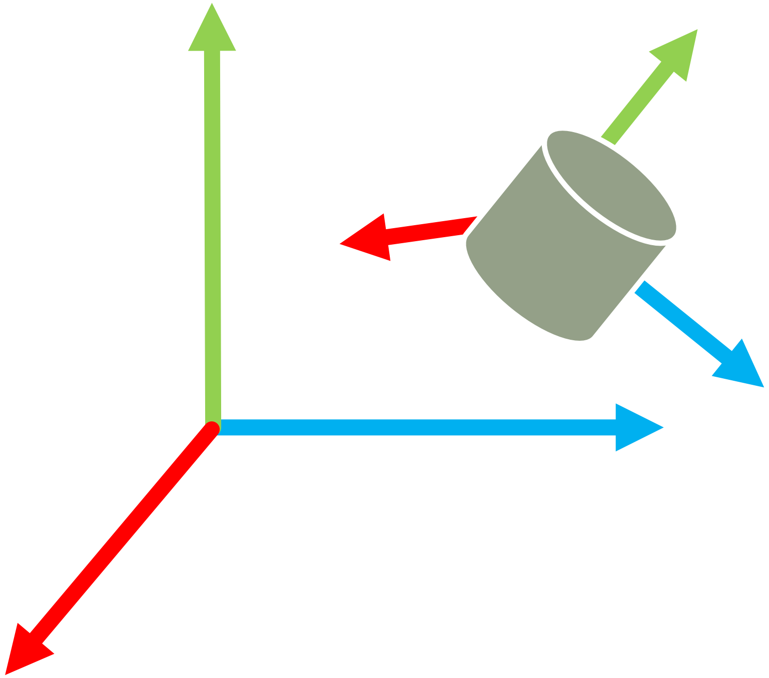
```
{
    "vertexPositions" : [0.85,0.6471428571428571,0.0571428
    "vertexNormals" : [0.000000,1.000000,0.000000,0.000000
    "vertexFrontcolors" : [1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0
    "vertexBackcolors" : [0.9803921568627451,0.0,0.0,0.980
}
```

World transform

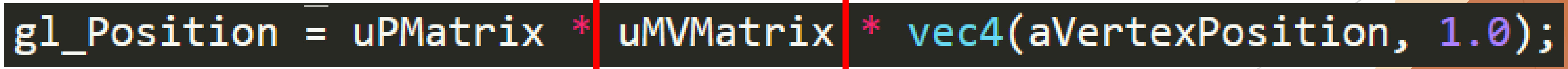


Model coordinates

World transform

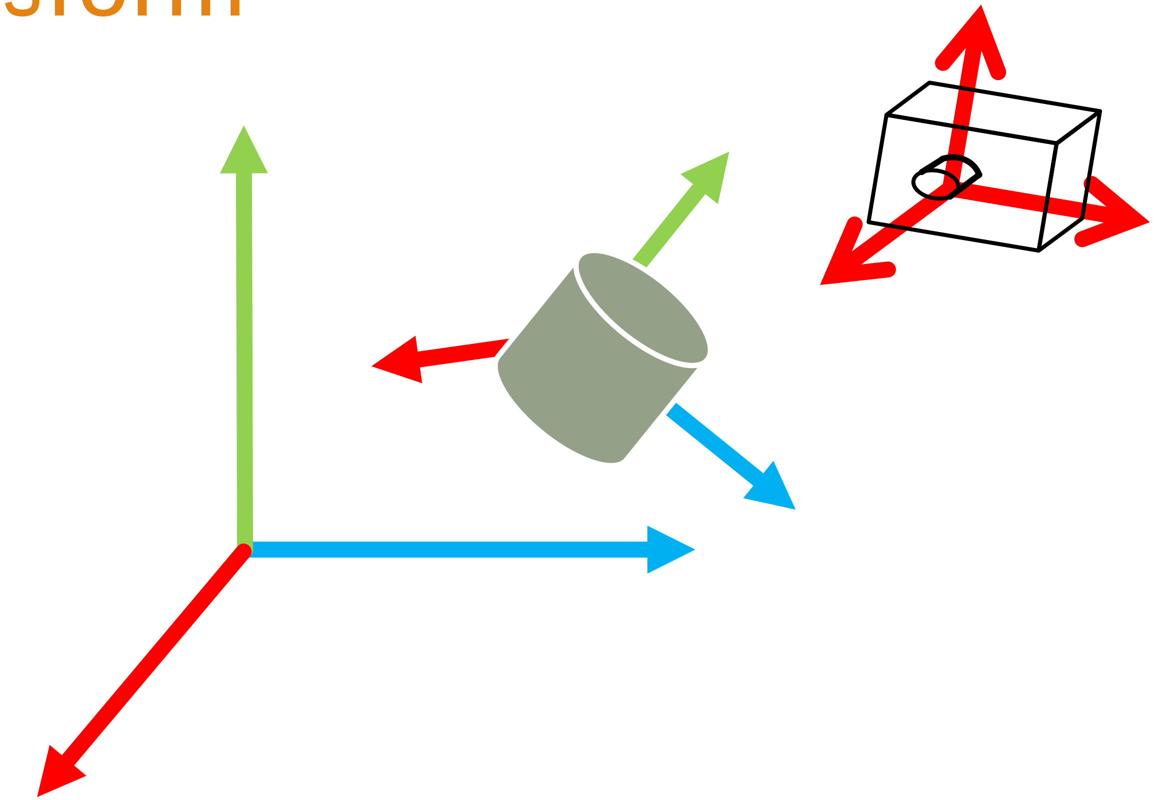


World coordinates

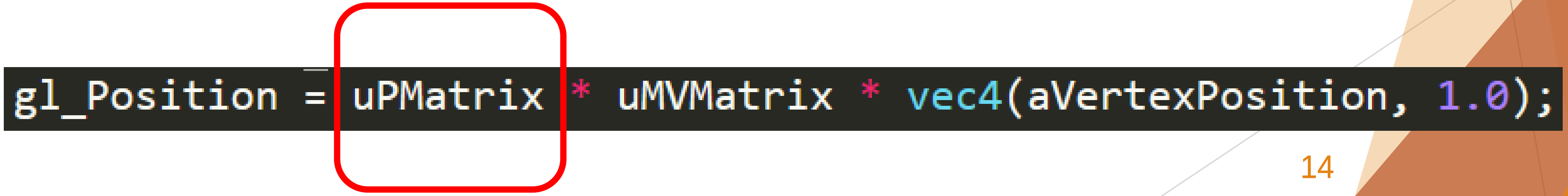


```
gl_Position = uPMatrix * uMVMatrix * vec4(aVertexPosition, 1.0);
```

World transform



Camera coordinates



```
gl_Position = uPMatrix * uMVMatrix * vec4(aVertexPosition, 1.0);
```

Transformations

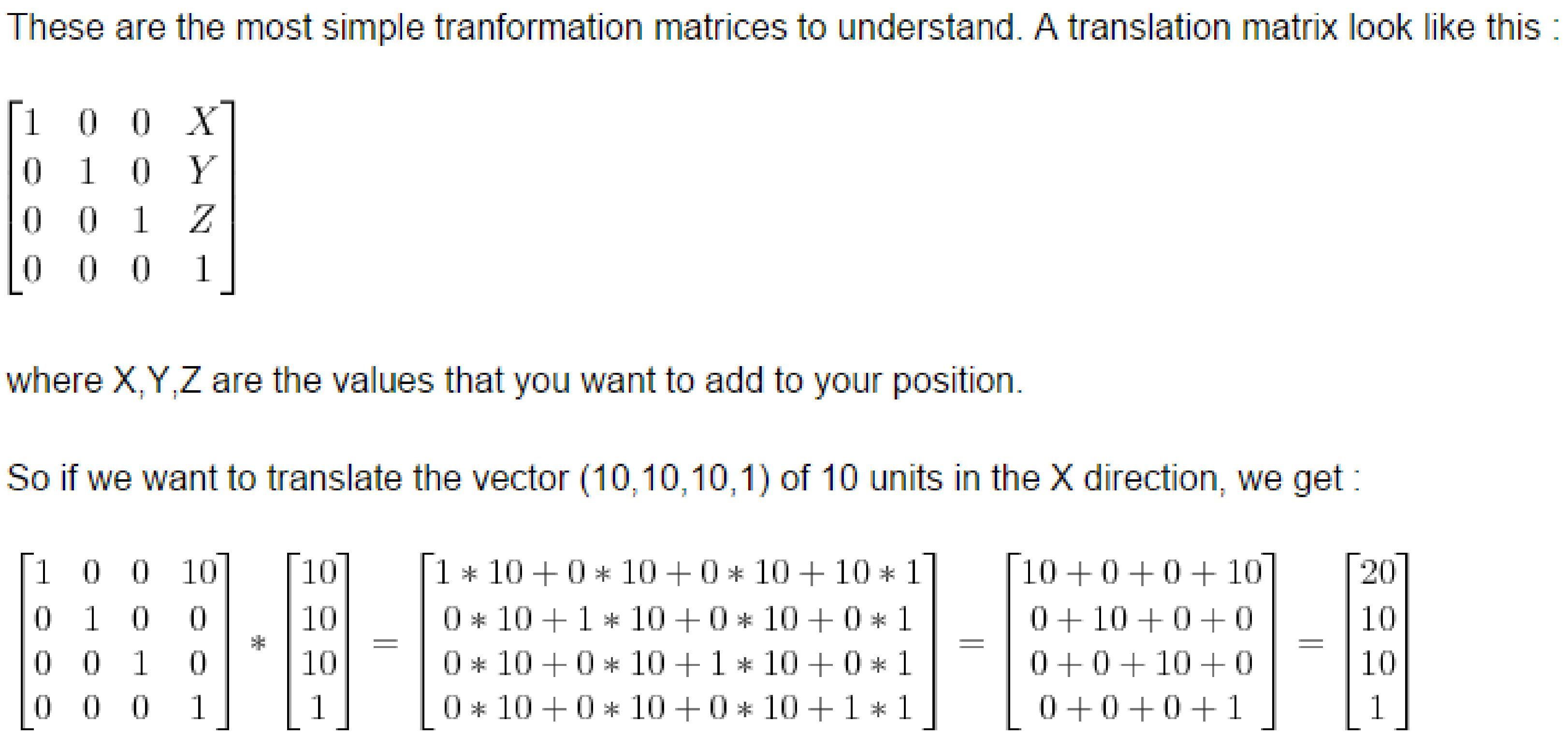
- Fundamental Transformations:
  - Translation、Scale、Rotation、Shear
  - Order of matrix multiplication may affect final result

- Homogeneous Coordinates

Matrix x Vertex (in this order !!) = TransformedVertex

$$\begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} ax + by + cz + dw \\ ex + fy + gz + hw \\ ix + jy + kz + lw \\ mx + ny + oz + pw \end{bmatrix}$$

Translation



These are the most simple tranformation matrices to understand. A translation matrix look like this :

$$\begin{bmatrix} 1 & 0 & 0 & X \\ 0 & 1 & 0 & Y \\ 0 & 0 & 1 & Z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where X,Y,Z are the values that you want to add to your position.

So if we want to translate the vector (10,10,10,1) of 10 units in the X direction, we get :

$$\begin{bmatrix} 1 & 0 & 0 & 10 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 10 \\ 10 \\ 10 \\ 1 \end{bmatrix} = \begin{bmatrix} 1*10+0*10+0*10+10*1 \\ 0*10+1*10+0*10+0*1 \\ 0*10+0*10+1*10+0*1 \\ 0*10+0*10+0*10+1*1 \end{bmatrix} = \begin{bmatrix} 10+0+0+10 \\ 0+10+0+0 \\ 0+0+10+0 \\ 0+0+0+1 \end{bmatrix} = \begin{bmatrix} 20 \\ 10 \\ 10 \\ 1 \end{bmatrix}$$

Scale

$$\begin{bmatrix} x & 0 & 0 & 0 \\ 0 & y & 0 & 0 \\ 0 & 0 & z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

So if you want to scale a vector (position or direction, it doesn’t matter) by 2.0 in all directions :

$$\begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} 2*x+0*y+0*z+0*w \\ 0*x+2*y+0*z+0*w \\ 0*x+0*y+2*z+0*w \\ 0*x+0*y+0*z+1*w \end{bmatrix} = \begin{bmatrix} 2*x+0+0+0 \\ 0+2*y+0+0 \\ 0+0+2*z+0 \\ 0+0+0+1*w \end{bmatrix} = \begin{bmatrix} 2*x \\ 2*y \\ 2*z \\ w \end{bmatrix}$$

Rotate

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

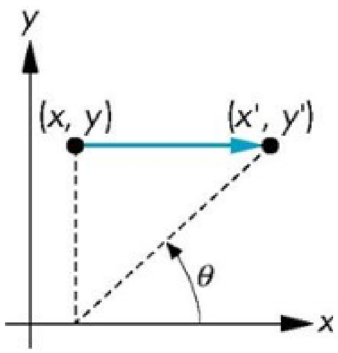
$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

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

Shear

Consider simple shear along x axis

$$x' = x + y \cot \theta$$
$$y' = y$$
$$z' = z$$



$$\mathbf{H}(\theta) = \begin{bmatrix} 1 & \cot \theta & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Requirements Again (Due to 2020/05/12)

- Implement **Flat, Gouraud, and Phong shading with Phong reflection model** in shaders.
- **Enable multiple transformations (four fundamental transformations) on objects in a scene**. You are free to use those provided model files and arrange them to form the scene on your own style.
- At least **3 objects** & at least **3 light sources**
- Bonus: Special effects on shading / lighting / animation, …

Result Example

Reference

- https://webglfundamentals.org/
- http://learningwebgl.com/blog/?page_id=1217
- https://learnopengl.com/

TA Hours

- 周家宇 (CSIE R505)
  - r07944038@csie.ntu.edu.tw
  - Thursday 14:00 ~ 16:00

- 李建德 (CSIE R506)
  - r08922180@ntu.edu.tw
  - Tuesday 15:00 ~ 17:00

Q & A