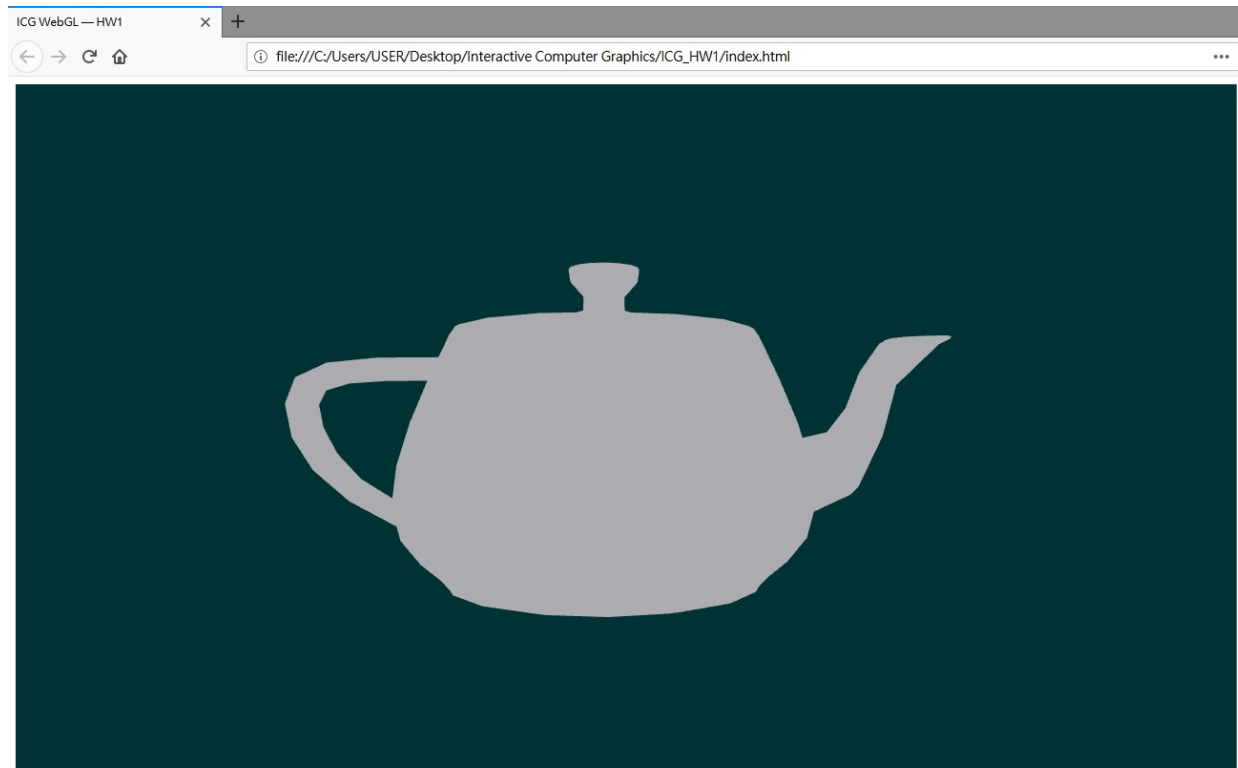


ICG 2022 Spring Homework1 Guidance 2022/03/17

Environment Setup

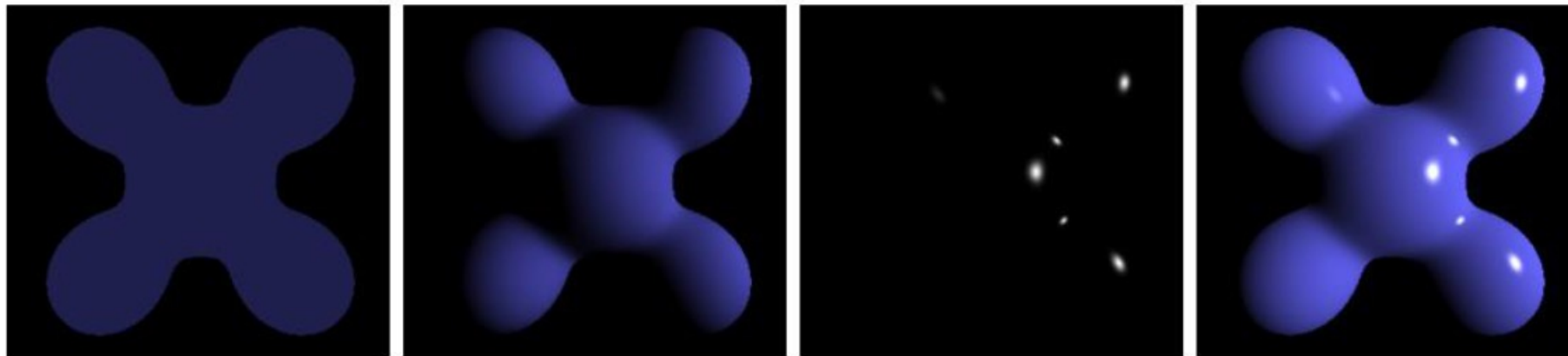
- ▶ Download sample code from course website
- ▶ Follow steps in **HW1_Guide.pdf** file



Requirements (Due to 2022/04/20 23:59)

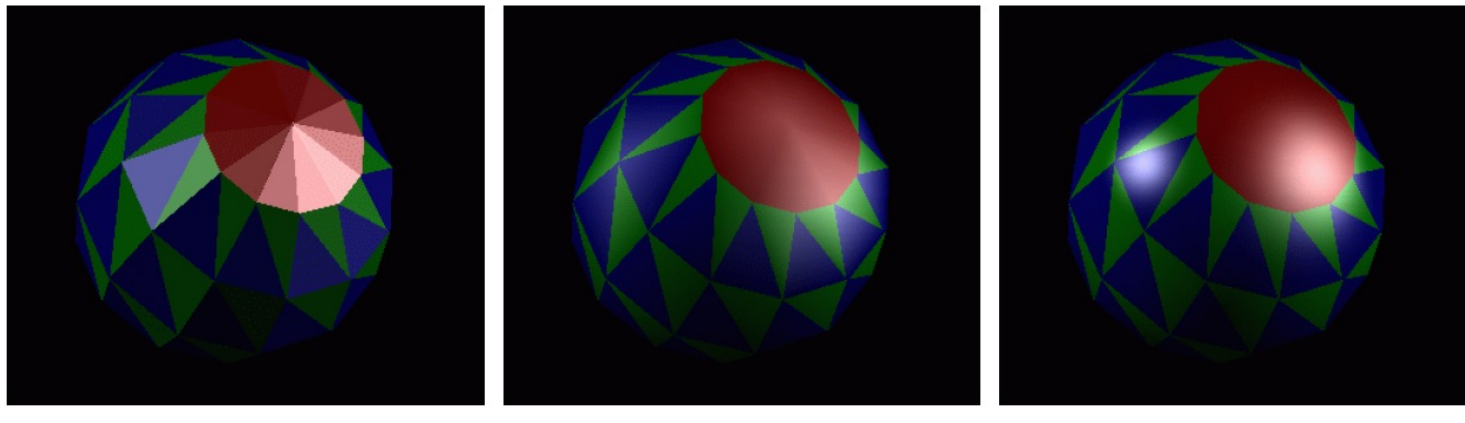
- ▶ Implement **Flat, Gouraud, and Phong shading with Phong reflection model** in shaders.
- ▶ **Enable multiple transformations (four fundamental transformations) on objects in a scene.** You are free to use those provided model files and arrange them to form the scene on your own style.
- ▶ At least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

Phong Reflection Model



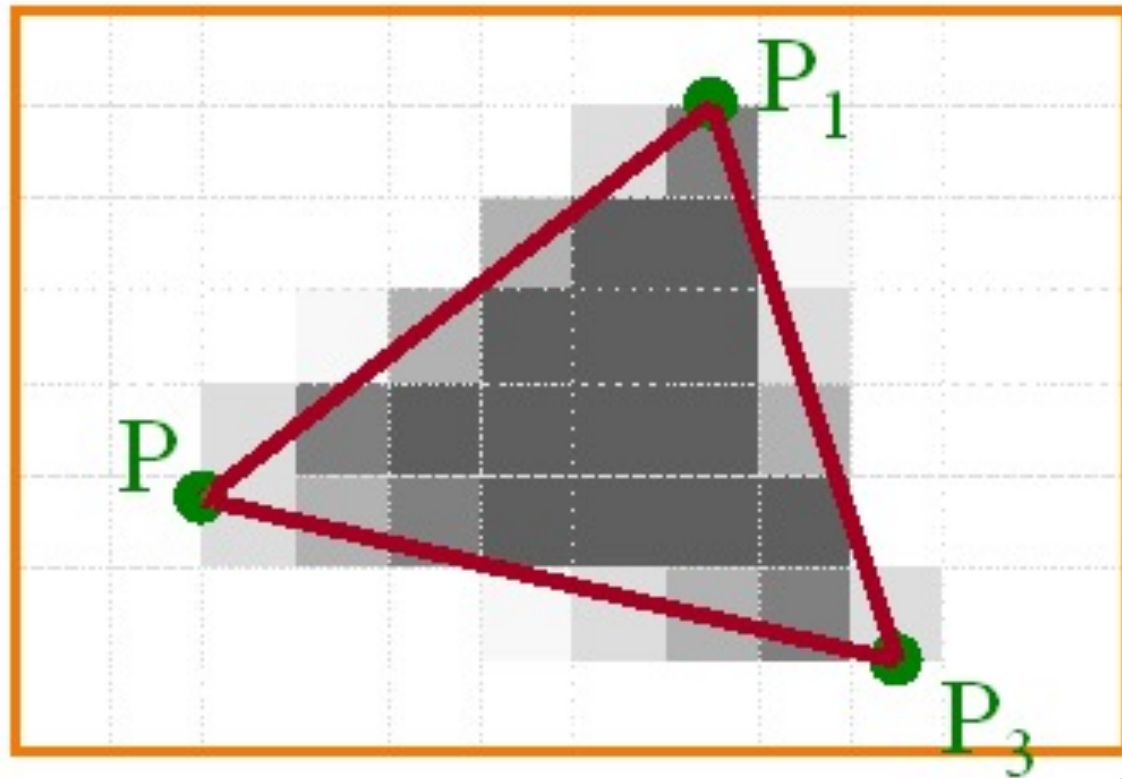
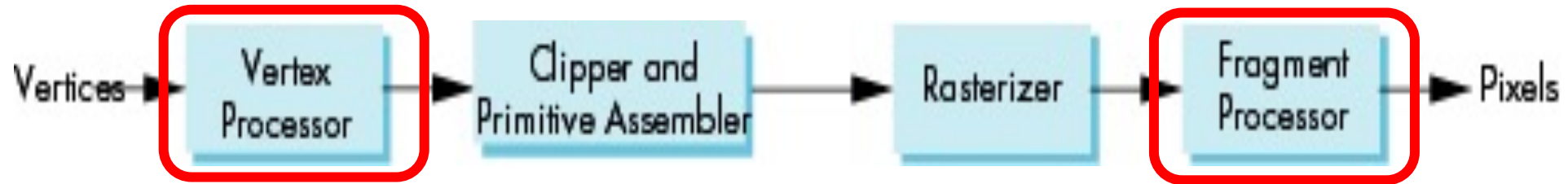
Ambient + Diffuse + Specular = Phong Reflection

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)

```
15 <script id="fragmentShader" type="fragment">
16     precision mediump float;
17
18     varying vec4 fragcolor;
19
20     void main(void) {
21         gl_FragColor = fragcolor;
22     }
23 </script>
```

```
35 <script id="vertexShader" type="vertex">
36     attribute vec3 aVertexPosition;
37     attribute vec3 aFrontColor;
38
39     uniform mat4 uMVMatrix;
40     uniform mat4 uPMatrix;
41
42     varying vec4 fragcolor;
43
44     void main(void) {
45         fragcolor = vec4(aFrontColor.rgb, 1.0);
46         gl_Position = uPMatrix * uMVMatrix * vec4(aVertexPosition, 1.0);
47     }
48 </script>
```


Shader Data (1/2)

```
15 <script id="fragmentShader" type="fragment">
16     precision mediump float;
17
18     varying vec4 fragcolor;
19
20     void main(void) {
21         gl_FragColor = fragcolor;
22     }
23 </script>
```

```
35 <script id="vertexShader" type="vertex">
36     attribute vec3 aVertexPosition;
37     attribute vec3 aFrontColor;
38
39     uniform mat4 uMVMatrix;
40     uniform mat4 uPMatrix;
41
42     varying vec4 fragcolor;
43
44     void main(void) {
45         fragcolor = vec4(aFrontColor.rgb, 1.0);
46         gl_Position = uPMatrix * uMVMatrix * vec4(aVertexPosition, 1.0);
47     }
48 </script>
```

Shader Data (2/2)

SHADER DATA

Vertex Data
= ANYTHING YOU WANT!

Example?
Positions...
Normals...
Colors...
Texture Coordinates...

“Per-object constant”

Uniform
= Shared Constant

Load Models

- ▶ 已經將大部分課程網的 **tri** 模型轉成 **json** 檔

Example Csie.json

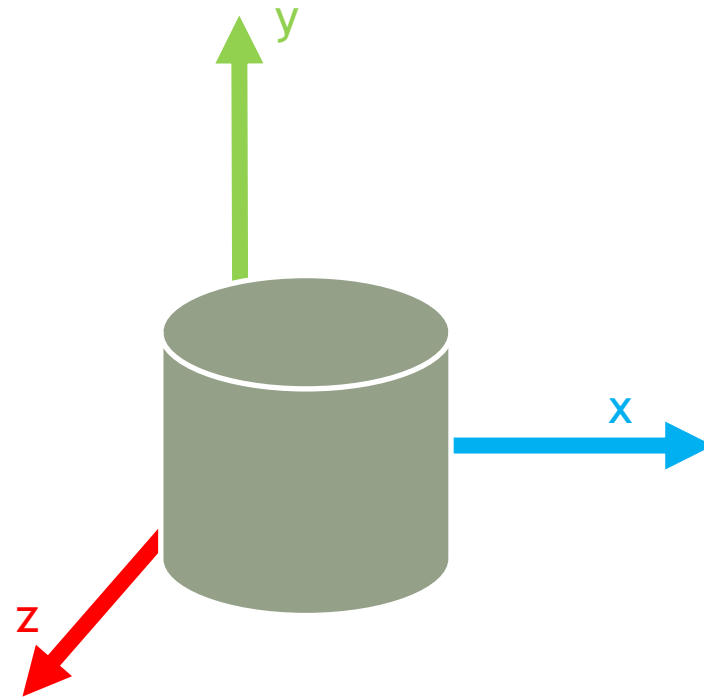
```
1 {  
2   "vertexPositions" : [0.85,0.6471428571428571,0.0571428  
3   "vertexNormals" : [0.000000,1.000000,0.000000,0.000000  
4   "vertexFrontcolors" : [1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0  
5   "vertexBackcolors" : [0.9803921568627451,0.0,0.0,0.980  
6 }
```

Load Models

- ▶ 將範例code的茶壺路徑替換成其他想要的模型即可。

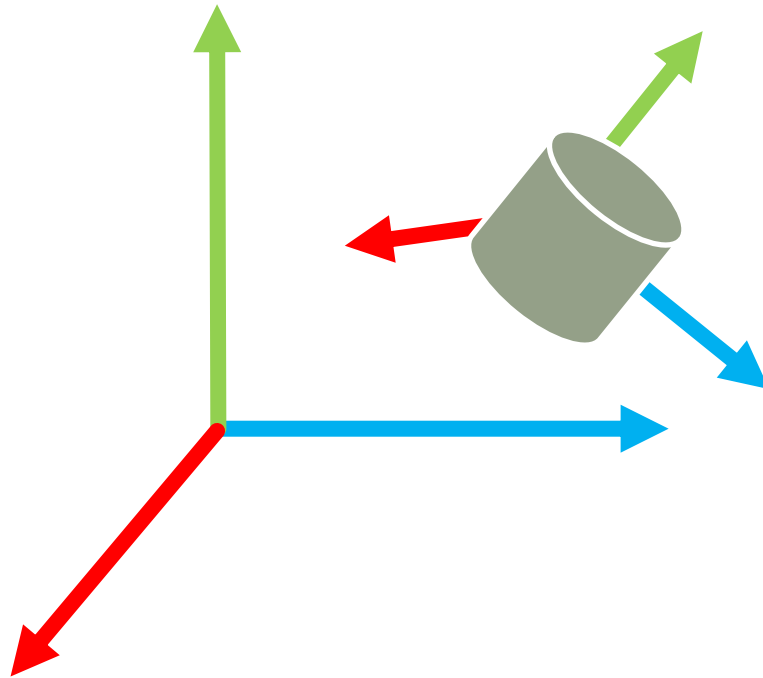
```
170     function loadTeapot() {  
171         var request = new XMLHttpRequest();  
172         request.open("GET", "./model/Teapot.json");  
173         request.onreadystatechange = function () {  
174             if (request.readyState == 4) {  
175                 handleLoadedTeapot(JSON.parse(request.responseText));  
176             }  
177         }  
178         request.send();  
179     }
```

World transform



Model coordinates

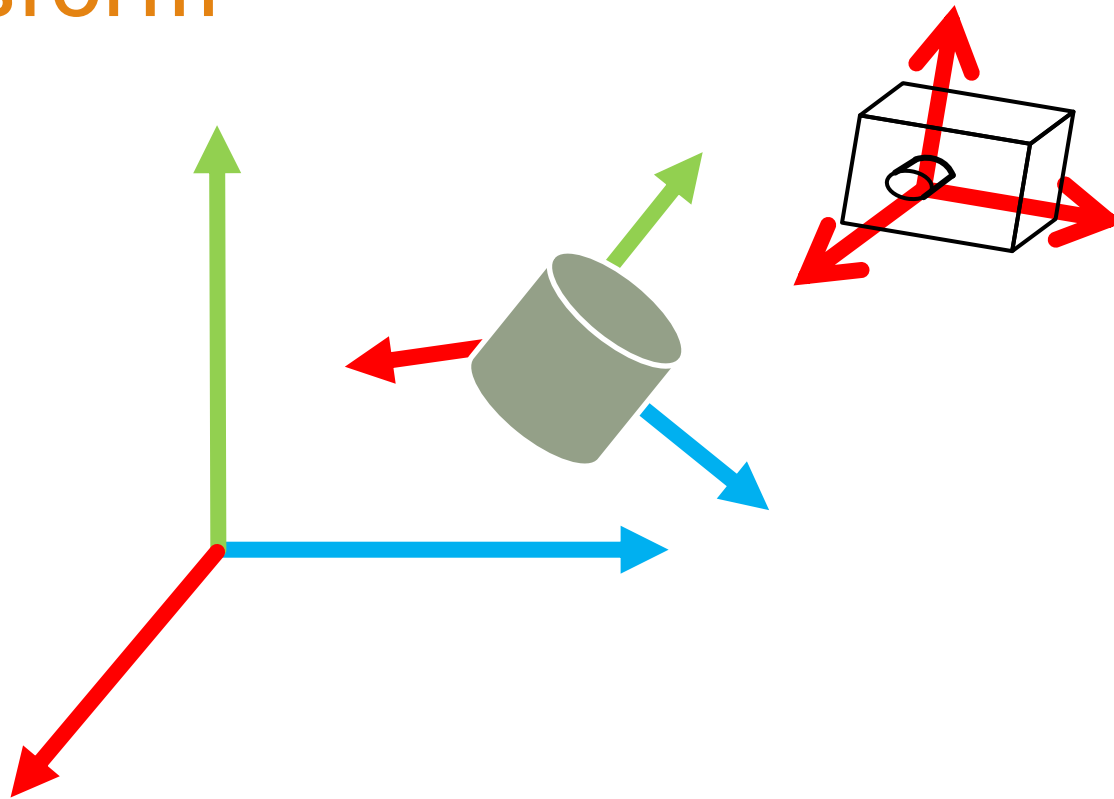
World transform



World coordinates

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

World transform



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

Transformations

- ▶ Fundamental Transformations:

- ▶ Translation 、 Scale 、 Rotation 、 Shear Shear 的 api 自己寫
- ▶ 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 transformation 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 :

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

```
200 // Setup Model-View Matrix
201 mat4.identity(mvMatrix);
202 mat4.translate(mvMatrix, [0, 0, -40]);
203 mat4.rotate(mvMatrix, degToRad(teapotAngle), [0, 1, 0]);
```

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

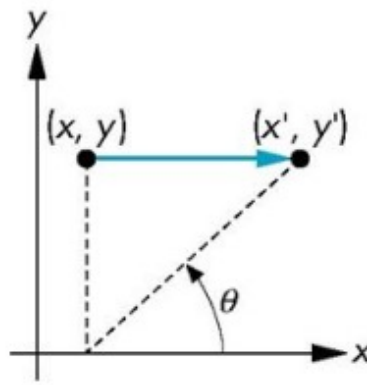
Consider simple shear along x axis

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

$$y' = y$$

$$z' = z$$

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



Requirements Again (Due to 2022/04/20)

- ▶ 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



TA Hours

- ▶ 李謙 (CSIE R403)
 - ▶ r10922001@ntu.edu.tw
 - ▶ Wednesday 13:00 ~ 14:00
- ▶ 洪佳生 (CSIE R506)
 - ▶ r10944051@ntu.edu.tw
 - ▶ Wednesday 13:20 ~ 14:20
- ▶ 張凱華 (CSIE R506)
 - ▶ r10922131@ntu.edu.tw
 - ▶ Wednesday 13:00 ~ 14:00

Reference

- ▶ <https://webglsfundamentals.org/>
- ▶ http://learningwebgl.com/blog/?page_id=1217
- ▶ <https://learnopengl.com/>

Q & A