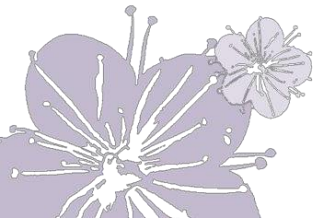
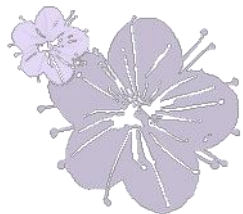
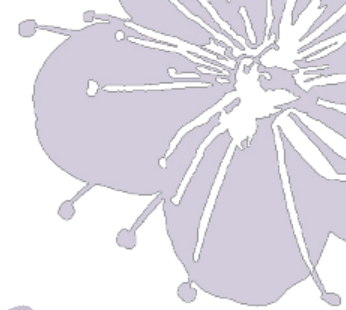


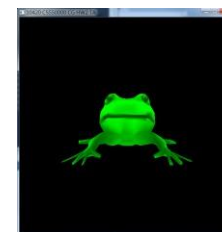
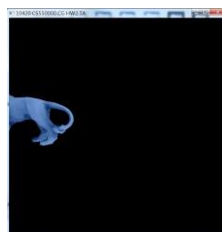
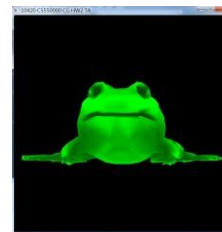
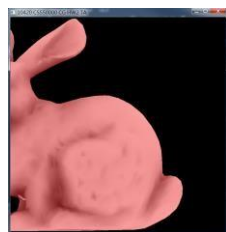
Homework 2

CS 550000 Computer Graphics
CGV Lab, NTHUCS



Goal

- Render and manipulate 3D models by implementing **transformation matrices**.



Goal

- Render and manipulate 3D models by implementing **transformation matrices**.
- Step by step
 - 1) Geometrical transformation
 - 2) Viewing transformation
 - 3) Projection transformation

Reminders

- You **CANNOT** use the existing transform API of OpenGL 1.X.
e.g. glRotate, glTranslate, glScale



Grading Principle

Total score: 100

- Transformation(70%)

- Implement MVP matrices.

Geometrical transformation: 30%

Viewing transformation : 30%

Projection transformation : 10%

- Control (15%)

- Keyboard & Mouse (Appendix)

- Display (5%)

- Display information such as model, mode, operation in the console window

- Report (10%)

- Express your work.

How to operate your program

Implementation and problems you met

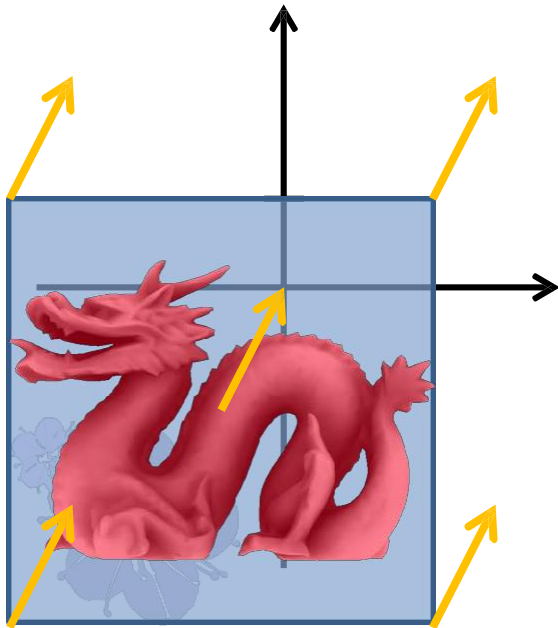
Other efforts you have done

Screenshots

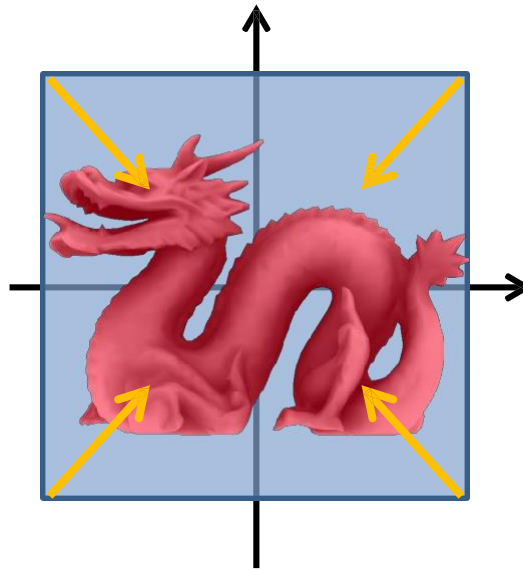


What we have done in HW1

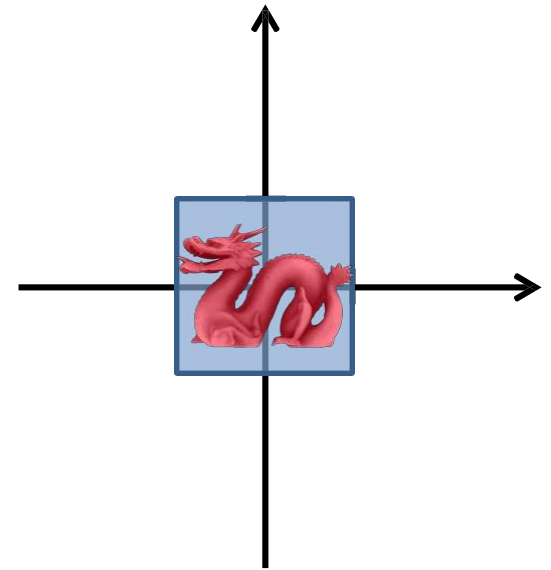
1. Translation



2. Scaling

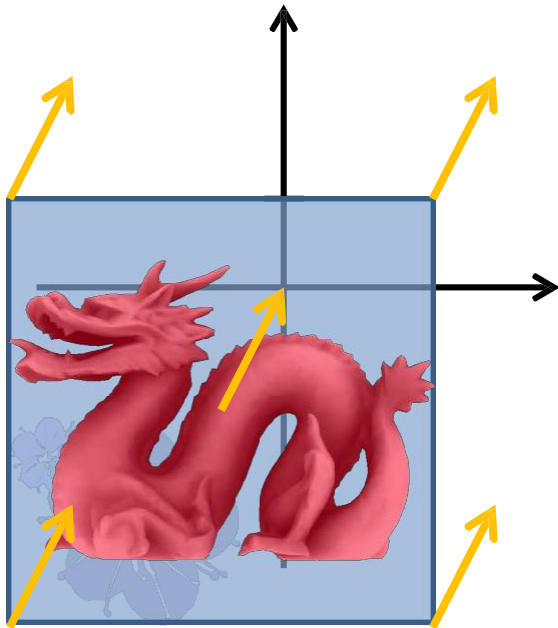


Normalized!

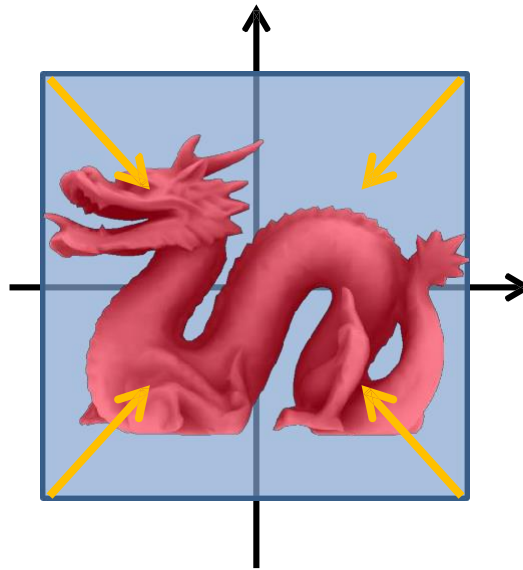


What we will do in HW2

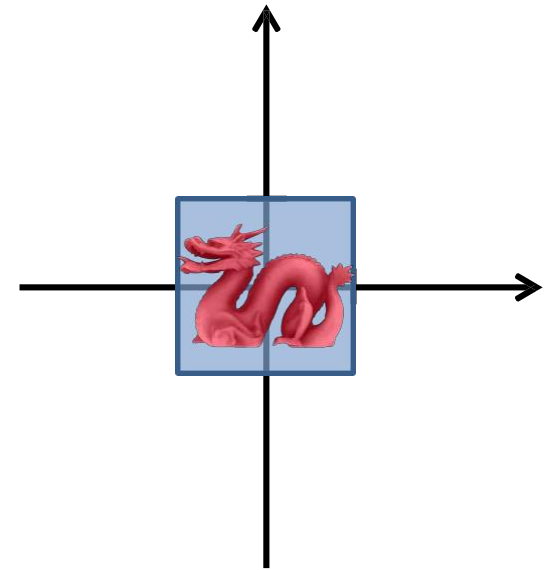
1. Translation



2. Scaling



Normalized!



Use different method to normalize 3D models!



Concept

$$\begin{bmatrix} \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \end{bmatrix} \begin{bmatrix} \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \end{bmatrix} \begin{bmatrix} \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix}$$

Projecting
Transformation

Viewing
Transformation

Model (Geometrical)
Transformation

Coordinates read from .obj



Concept

- Default setting in HW1

$$\begin{matrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\ \mathbf{P} & \mathbf{V} & \mathbf{M} \end{matrix}$$

```
shader.vert x
1 attribute vec4 av4position;
2 attribute vec3 av3color;
3
4 varying vec3 vv3color;
5
6 void main() {
7     // NOTE!! column major
8     mat4 mvp = mat4(
9         vec4( 1, 0, 0, 0),
10        vec4( 0, 1, 0, 0),
11        vec4( 0, 0, -1, 0),
12        vec4( 0, 0, 0, 1)
13    );
14    vv3color = av3color;
15    gl_Position = mvp * av4position;
```

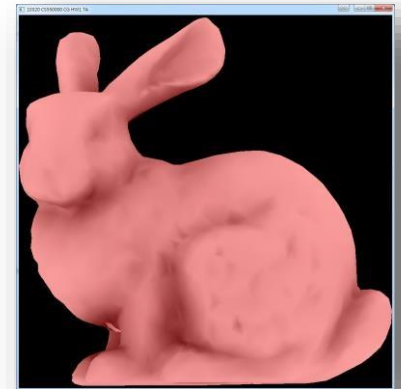
Remember to delete the default setting.
We will implement MVP ourselves in HW2!



Step1: Geometrical Transformation

- Using the **translation** and **scaling** matrices to implement normalization.

$$\begin{matrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \end{bmatrix} \\ \mathbf{P} & \mathbf{V} & \mathbf{M} \end{matrix}$$



- Reference

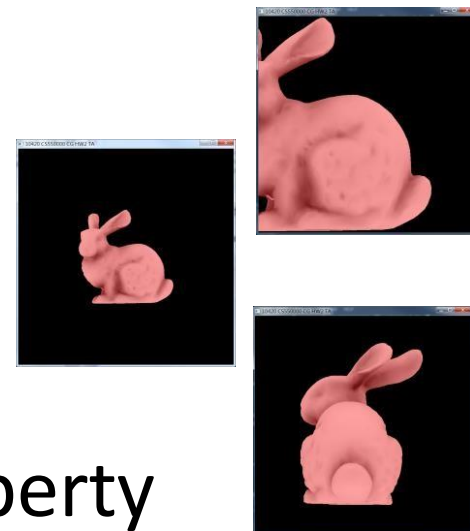
— CG04-Transformation p.39-p.43



Step1: Geometrical Transformation

- Manipulate 3D models
 - Translation, scaling, rotation

$$\begin{array}{c}
 \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \end{bmatrix} \\
 \mathbf{P} \qquad \qquad \mathbf{V} \qquad \qquad \mathbf{M}
 \end{array}$$



- Be careful of the commutative property
 - $T \cdot R \neq R \cdot T$

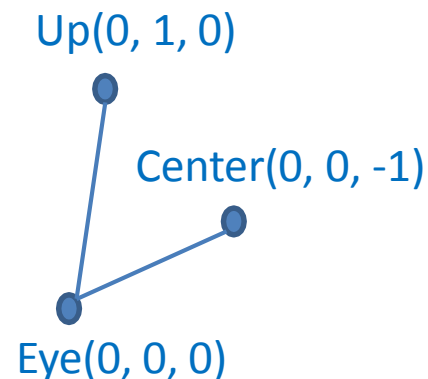


Step2: Viewing Transformation

- Display 3D models from different view.
 - Eye position, center position, up position

$$\begin{matrix}
 \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \end{bmatrix} \\
 \mathbf{P} & \mathbf{V} & \mathbf{M}
 \end{matrix}$$

Default value in HW1



- Reference

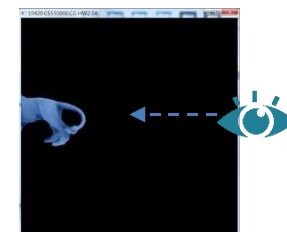
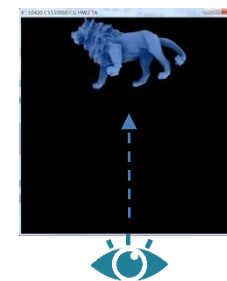
– CG04-Transformation p.72



Step2: Viewing Transformation

- Display 3D models from different view.
 - Eye position, center position, up position

$$\begin{matrix}
 \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \begin{bmatrix} \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \end{bmatrix} & \begin{bmatrix} \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \end{bmatrix} \\
 \text{P} & \text{V} & \text{M}
 \end{matrix}$$



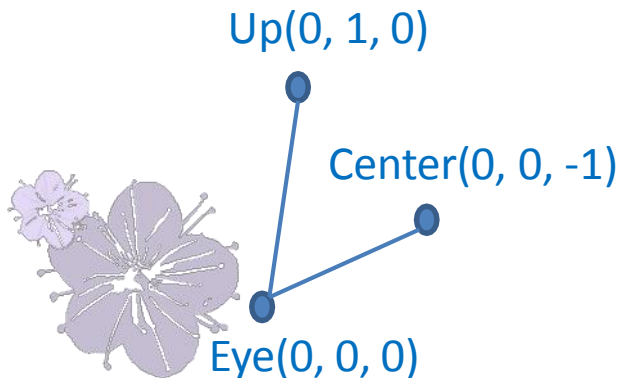
Requirement

- Center position & up position should **also be changed** if we change eye position

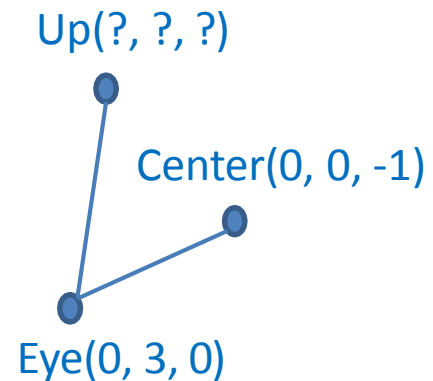


Relationship between Eye, Center, Up

- When change eye(center) position, we have to adjust up position to get a proper result, here is an example, if we move eye from (0,0,0) to (0,3,0)



$$\text{Forward} = \text{center}(0,0,-1) - \text{eye}(0,0,0) = (0,0,-1)$$

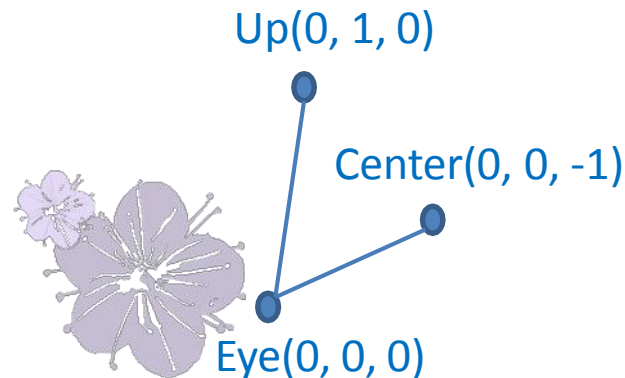


$$\text{Forward} = \text{center}(0,0,-1) - \text{eye}(0,3,0) = (0,-3,-1)$$

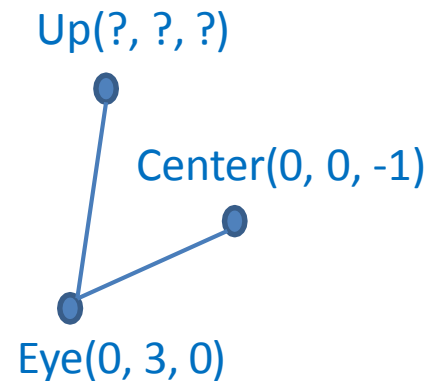


Relationship between Eye, Center, Up

- Because the forward vector and up vector must be perpendicular, now forward vector changed, we need to compute new up vector



$$\text{Forward} = \text{center}(0,0,-1) - \text{eye}(0,0,0) = (0,0,-1)$$

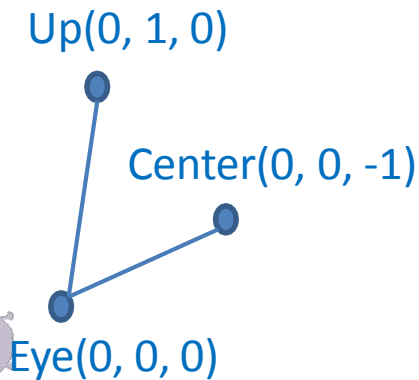


$$\text{Forward} = \text{center}(0,0,-1) - \text{eye}(0,3,0) = (0,-3,-1)$$



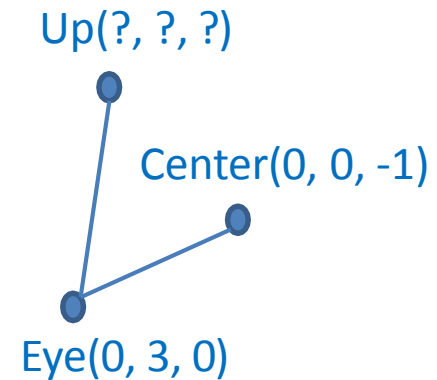
Relationship between Eye, Center, Up

- In this case, we can compute right vector by the cross product of forward and up vector(old one)



$$\text{Forward} = \text{center}(0,0,-1) - \text{eye}(0,0,0) = (0,0,-1)$$

$$\text{Right} = \text{forward}(0,0,-1) \times \text{up}(0,1,0) = (1,0,0)$$



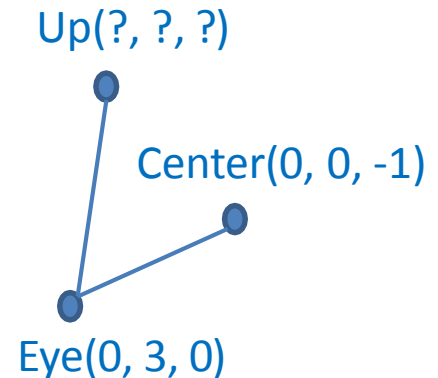
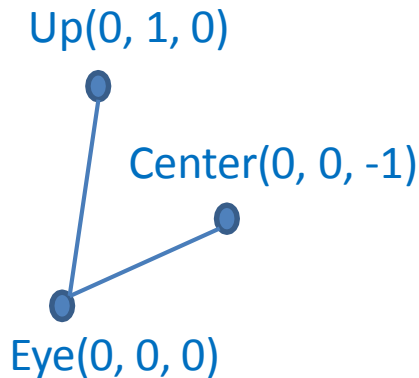
$$\text{Forward} = \text{center}(0,0,-1) - \text{eye}(0,3,0) = (0,-3,-1)$$

$$\text{Right} = \text{forward}(0,-3,-1) \times \text{up}(0,1,0) = (1,0,0)$$



Relationship between Eye, Center, Up

- Finally re-compute new up vector by the cross product of right vector and forward vector, and find new up position!



Forward = center(0,0,-1)-eye(0,0,0) = (0,0,-1)
Right = forward(0,0,-1) X up(0,1,0) = (1,0,0)
Up = right(1,0,0) X forward(0,0,-1) = (0,1,0)

Forward = center(0,0,-1)-eye(0,3,0) = (0,-3,-1)
Right = forward(0,-3,-1) X up(0,1,0) = (1,0,0)
Up vector(new) =
right(1,0,0) X forward(0,-3,-1) = (0,-1,-3)
Up(position) =
Eye(0,3,0) + up vector(0,-1,-3) =
(0, 2, -3)



Step3: Projection Transformation

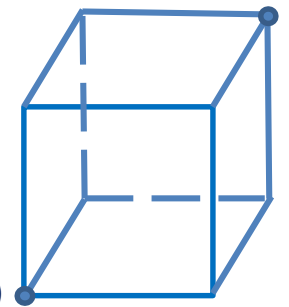
- Project 3D models on screen in different way.
 - Parallel(orthogonal), perspective projection

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \end{bmatrix} \begin{bmatrix} \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \end{bmatrix}$$

P **V** **M**

Default value in HW1
(parallel)

(xmax, ymax, zfar) = (1, 1, -1)



(xmin, ymin, znear) = (-1, -1, 1)

Coordinates!

• Reference

– CG04-Transformation p.121-p.128

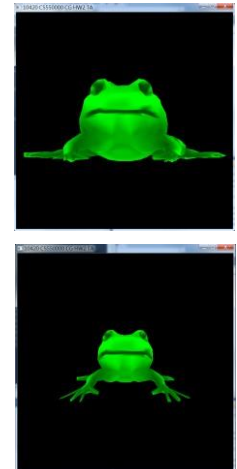


Step3: Projection Transformation

- Project 3D models on screen in different way.
 - Parallel(orthogonal), perspective

$$\begin{bmatrix} \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \end{bmatrix} \begin{bmatrix} \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \end{bmatrix} \begin{bmatrix} \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \\ \square & \square & \square & \square \end{bmatrix}$$

P **V** **M**



- **Reminder**

- OpenGL APIs: znear, zfar are **distances** to the viewpoint instead of the z coordinates.

- The model should appear on the screen directly. Please find the proper values of all the parameters (e.g. znear, zfar, eye).



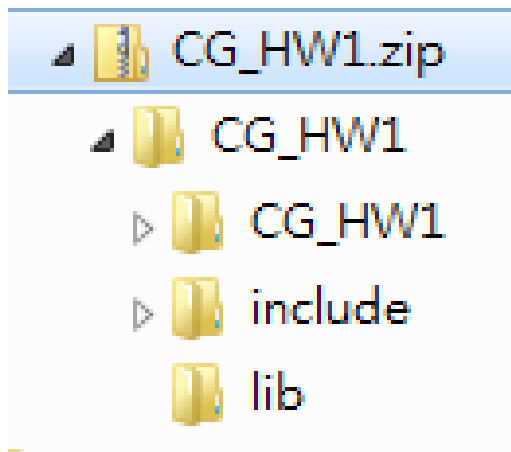
Submission

- Due date: **April 19th, 2017**
- Submit your project to **iLMS**.
- Filename: **HW2_XXXXXXXXX.zip**
- Put both “**lib**” and “**include**” folder in your zip file



位置: 計算機圖學Computer Graphics > 作業 ✓ 新增 | 複製

| 項次 | 標題 | 分組作業 | 已評分 / 繳交 | 期限 | 動作 |
|----|--------------------------------|------|----------|-------------|---------|
| 1 | HW1: Draw Some Geometry Models | | 0 / 0 | 04-06 23:59 | 編輯 刪除 |



| 名稱 | 大小 | 類型 |
|----------------|-----------|--------------------------|
| CG_HW1.sdf | 31,700 KB | SQL Server Compact ... |
| CG_HW1.suo | 44 KB | Visual Studio Solutio... |
| CG_HW1.sln | 1 KB | Microsoft Visual Stud... |
| CG_HW1.opensdf | 0 KB | OPENSDF 檔案 |
| CG_HW1 | | 檔案資料夾 |
| Debug | | 檔案資料夾 |
| ipch | | 檔案資料夾 |

建立日期: 2015/3/17 下午 08:09
大小: 24.0 MB
資料夾: cg_hw1-f20be826

***** Remove “ipch” folder and “.sdf” file. *****



Reminders

- Late submission is accepted. (-10%/week)
- Ask and share information through iLMS



1032 (2015-02-01~2015-07-31)

課程: [計算機圖學Computer Graph](#) ▼

課程資訊 [\[報表\]](#)

訪客: 5518

文章: 269

討論: 190

容量: 剩餘 401.5 MB (2.9 GB)

老師: 李潤容 ☒

助教: 羅逸翔 ☒, 阮維廷 ☒, 蔡繪琦 ☒

閱讀權限: 不開放旁聽 (僅成員可以閱讀)


課程功能 [\[管理\]](#)

 [課程活動\(公告\)](#)

 [上課教材 \(18\)](#)

 [課堂整理](#)

 [課程說明](#)

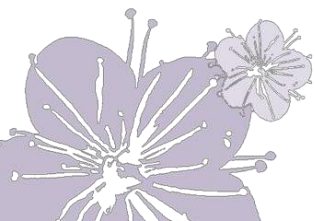
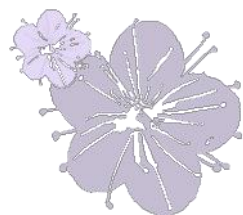
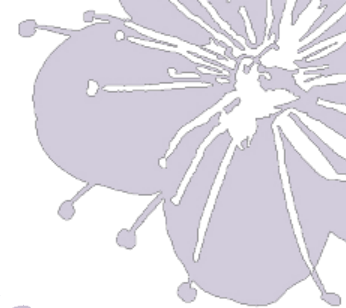
 [課程行事曆](#)

 [討論區 \(190\)](#)

 [小組專區](#)



DEMO



Appendix

CS 550000 Computer Graphics

April 13, 2016

Department of Computer Science

National Tsing Hua University



HW2 Framework

main.cpp

- Global variable

```
28 // Shader attributes
29 GLint iLocPosition;
30 GLint iLocColor;
31 GLint iLocMVP;
```

- void setShaders()

```
245 // link program
246 glLinkProgram(p);
247
248 iLocPosition = glGetUniformLocation(p, "av4position");
249 iLocColor = glGetUniformLocation(p, "av3color");
250 iLocMVP = glGetUniformLocation(p, "mvp");
```

- void onDisplay()

```
178 // bind array pointers to shader
179 glVertexAttribPointer(iLocPosition, 3, GL_FLOAT, GL_FALSE, 0, triangle_vertex);
180 glVertexAttribPointer(iLocColor, 3, GL_FLOAT, GL_FALSE, 0, triangle_color);
181
182 // bind uniform matrix to shader const GLfloat*
183 glUniformMatrix4fv(iLocMVP, 1, GL_FALSE, mvp);
```

What you need to define in HW2

shader.vert

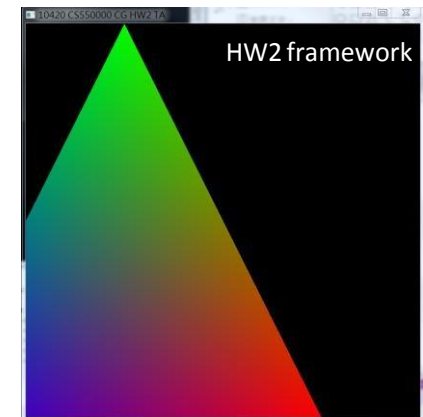
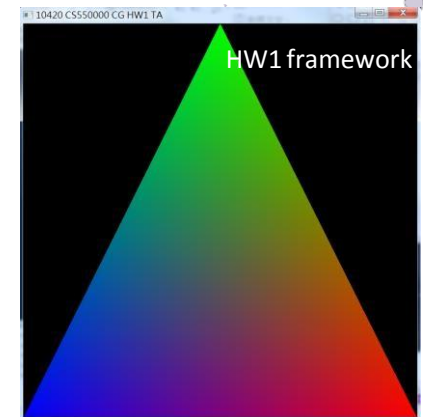
```
1 attribute vec4 av4position;
2 attribute vec3 av3color;
3
4 varying vec3 vv3color;
5
6 uniform mat4 mvp;
7
8 void main() {
9     // NOTE!! column major
10    /*mat4 mvp = mat4(
11        vec4( 1, 0, 0, 0),
12        vec4( 0, 1, 0, 0),
13        vec4( 0, 0, -1, 0),
14        vec4( 0, 0, 0, 1)
15    );*/
16    vv3color = av3color;
17    gl_Position = mvp * av4position;
18 }
19
```



HW2 Framework

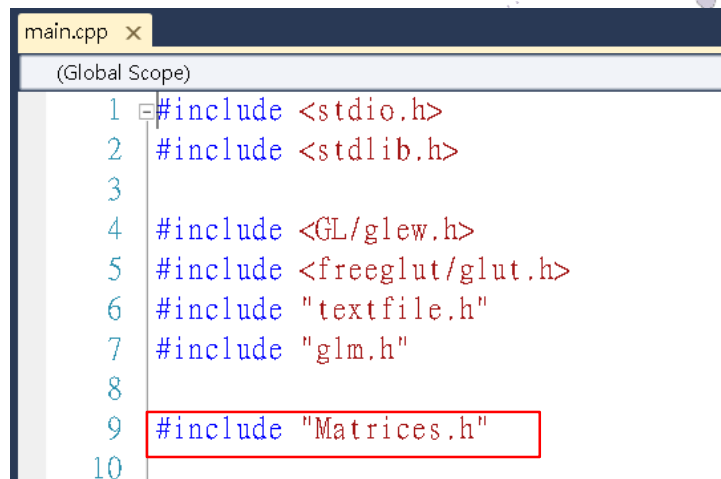
```
main.cpp x
(Global Scope) onDisplay(void)
148 //MVP
149 Matrix4 T;
150 Matrix4 S;
151 Matrix4 R;
152
153 Matrix4 M = Matrix4(
154     1, 0, 0, -0.5,
155     0, 1, 0, 0,
156     0, 0, 1, 0,
157     0, 0, 0, 1);
158 Matrix4 V = Matrix4(
159     1, 0, 0, 0,
160     0, 1, 0, 0,
161     0, 0, 1, 0,
162     0, 0, 0, 1);
163 Matrix4 P = Matrix4(
164     1, 0, 0, 0,
165     0, 1, 0, 0,
166     0, 0, -1, 0,
167     0, 0, 0, 1);
168
169 Matrix4 MVP = P*V*M;
170
171 GLfloat mvp[16];
172 // row-major ---> column-major
173 mvp[0] = MVP[0]; mvp[4] = MVP[1]; mvp[8] = MVP[2]; mvp[12] = MVP[3];
174 mvp[1] = MVP[4]; mvp[5] = MVP[5]; mvp[9] = MVP[6]; mvp[13] = MVP[7];
175 mvp[2] = MVP[8]; mvp[6] = MVP[9]; mvp[10] = MVP[10]; mvp[14] = MVP[11];
176 mvp[3] = MVP[12]; mvp[7] = MVP[13]; mvp[11] = MVP[14]; mvp[15] = MVP[15];
177
```

move -0.5 in x direction



Useful Library

- Files
 - Matrices.cpp
 - Matrices.h
 - Vectors.h

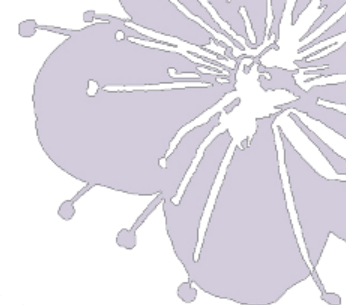


```
main.cpp x
(Global Scope)
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 #include <GL/glew.h>
5 #include <freeglut/glut.h>
6 #include "textfile.h"
7 #include "glm.h"
8
9 #include "Matrices.h"
10
```

- Put these files under the same folder as main.cpp

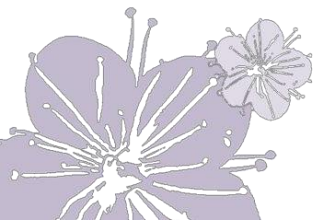
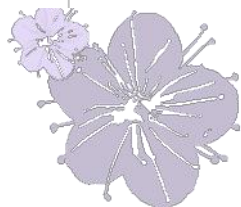


Useful Library



```
Matrices.h x
(Global Scope)
131
132 ///////////////////////////////////////////////////
133 // 4x4 matrix
134 ///////////////////////////////////////////////////
135 class Matrix4
136 {
137 public:
138     // constructors
139     Matrix4(); // init with identity
140     Matrix4(const float src[16]);
141     Matrix4(float xx, float xy, float xz, float xw,
142             float yx, float yy, float yz, float yw,
143             float zx, float zy, float zz, float zw,
144             float wx, float wy, float wz, float ww);
145
146     void set(const float src[16]);
147     void set(float xx, float xy, float xz, float xw,
148             float yx, float yy, float yz, float yw,
149             float zx, float zy, float zz, float zw,
```

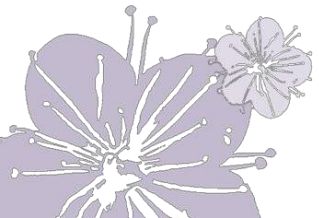
```
Vectors.h x
Vector3
67 struct Vector3
68 {
69     float x;
70     float y;
71     float z;
72
73     // ctors
74     Vector3() : x(0), y(0), z(0) {};
75     Vector3(float x, float y, float z) : x(x), y(y), z(z) {};
76
77     // utils functions
78     void set(float x, float y, float z);
79     float length() const; //
80     float distance(const Vector3& vec) const; // distance between two vectors
81     Vector3& normalize(); //
82     float dot(const Vector3& vec) const; // dot product
83     Vector3 cross(const Vector3& vec) const; // cross product
84     bool equal(const Vector3& vec, float e) const; // compare with epsilon
```



Control (Keyboard)



| Mode | Manipulation | Key |
|---------------------------|-----------------------------------------------------------------|-------|
| Translation | Change (x ,y, z) | T |
| Scaling | | S |
| Rotation | | R |
| Eye position | | E |
| Center position | | L |
| Help menu | Show help menu | H |
| Switch Projection | Switch between parallel / perspective projection | O |
| Change model | Previous / next model as hw1 | Z / X |
| Projection transformation | Change up, down, right, left, near, far of projection parameter | P |



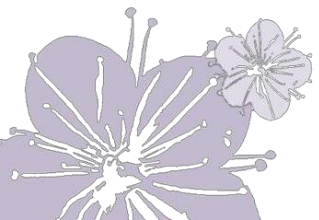
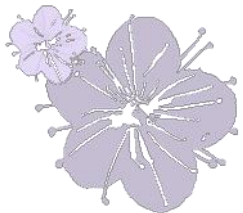
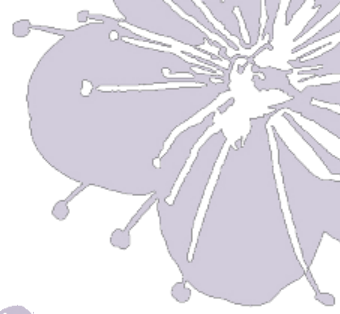
Control (Mouse)

Geometrical / Viewing transformation mode

| Button | Manipulation |
|--------|----------------------------------------------------------------------------|
| Left | Drag horizontal for x offset Drag vertically for y offset |
| Right | Same as left |
| Middle | Wheel up/down for z offset |

Projection transformation mode

| Button | Manipulation |
|--------|------------------------------------------------------------------------------------------------------------------|
| Left | Drag horizontal for left-right boundary scaling Drag vertically for bottom-top boundary scaling |
| Right | Drag horizontal for moving near clipping plane Drag vertically for moving far clipping plane |
| Middle | none |



Q & A

