

# 計算機圖學 作業二報告

103062234 張克齊

## 一、 The method of operating program:

執行程式前，要先確定在 CG\_HW2 資料夾底下有 ColorModels 這個資料夾，裡面用來存放要讀取的.obj 檔；接著，要修改程式中所開的 filename 陣列和 file\_size，若 filename 中寫了 12 個路徑的名稱，則要將 file\_size 改成 12，如下圖所示，這樣就能順利執行程式。而操作如同要求所述，也可按 h 鍵看詳細的按鍵說明。

```
char filename[][50] = { "ColorModels/al7KC.obj", "ColorModels/blitzcrank_incognito.obj",  
                        "ColorModels/texturedknot11KC.obj", "ColorModels/elephant16KC.obj",  
                        "ColorModels/frog2KC.obj", "ColorModels/dragon10KC.obj",  
                        "ColorModels/igea17KC.obj", "ColorModels/Dino20KC.obj",  
                        "ColorModels/lion12KC.obj", "ColorModels/lucy25KC.obj",  
                        "ColorModels/armadillo12KC.obj", "ColorModels/ziggs.obj" };  
int file_size = 12;
```

## 二、 Implementation and problems:

首先是要畫地板，方法是先讀 BoxC 的 OBJ，並存下他的 vertices 和 colors，之後就不會再讀，這樣就能畫出地板；接著主要是實作不同功能的矩陣並透過滑鼠改變矩陣中的值，因此只要我們作對正確的矩陣後，就能輕易實現我們的功能。底下分別討論 M, V, P 的作法，最後的 MVP = P\*V\*M。

### 1. M (Model):

在操縱 model 的部分，總共可以分成 T (translate), S (scale), R (rotation), N (normalization) 四個矩陣相乘後得結果，也就是  $M = T * S * R * N$ 。而這四個的實作矩陣方法都是依照講義，以下用講義與 code 做對照，其中沒有給值的變數是可以利用滑鼠來調整各個軸的值，完成題目要求的功能。

(1) T:

◆ Translation

$P(x, y, z)$  move to  $P'(x', y', z')$

$$x' = x + d_x$$

$$y' = y + d_y$$

$$z' = z + d_z$$

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix} + \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

$$P' = T(d_x, d_y, d_z) \cdot P$$

```
Matrix4(  
    1.0f, 0.0f, 0.0f, t_x,  
    0.0f, 1.0f, 0.0f, t_y,  
    0.0f, 0.0f, 1.0f, t_z,  
    0.0f, 0.0f, 0.0f, 1.0f)
```

(2) S:

◆ Scaling

Scale by  $s_x$ ,  $s_y$ , and  $s_z$  with respect to the  $x$  axis,  $y$  axis, and  $z$  axis

$$x' = s_x \cdot x$$

$$y' = s_y \cdot y$$

$$z' = s_z \cdot z$$

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

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

```
Matrix4(  
    s_x, 0.0f, 0.0f, 0.0f,  
    0.0f, s_y, 0.0f, 0.0f,  
    0.0f, 0.0f, s_z, 0.0f,  
    0.0f, 0.0f, 0.0f, 1.0f)
```

(3) R: 這邊可以透過先求出各個軸的旋轉矩陣後再相乘為最後的 R。

```
R_X = Matrix4(  
    1.0f, 0.0f, 0.0f, 0.0f,  
    0.0f, cos(r_x), (-1)*sin(r_x), 0.0f,  
    0.0f, sin(r_x), cos(r_x), 0.0f,  
    0.0f, 0.0f, 0.0f, 1.0f);
```

```
R_Y = Matrix4(  
    cos(r_y), 0.0f, sin(r_y), 0.0f,  
    0.0f, 1.0f, 0.0f, 0.0f,  
    (-1)*sin(r_y), 0.0f, cos(r_y), 0.0f,  
    0.0f, 0.0f, 0.0f, 1.0f);
```

```
R_Z = Matrix4(  
    cos(r_z), (-1)*sin(r_z), 0.0f, 0.0f,  
    sin(r_z), cos(r_z), 0.0f, 0.0f,  
    0.0f, 0.0f, 1.0f, 0.0f,  
    0.0f, 0.0f, 0.0f, 1.0f);
```

◆ Commutative

$$\begin{aligned} R(\theta) &= R_z(\theta_z) \cdot R_y(\theta_y) \cdot R_x(\theta_x) \\ &= R_x(\theta_x) \cdot R_y(\theta_y) \cdot R_z(\theta_z) \end{aligned}$$

```
Matrix4 R = R_X * R_Y * R_Z;
```

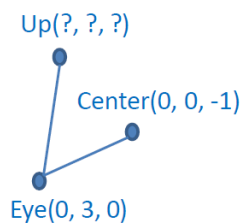
(4) N: 在 Normalization 的部分，跟上次作業調整的方法一樣，只是之前是直接變更 vertice 裡的值，而這邊則是透過矩陣運算得到相同的結果，方法是分成平移和縮放兩個矩陣作用(相乘)，這邊一開始遇到的問題是我將兩個合併成一個矩陣，這樣會因為他同時進行兩個運算而會使最後的結果錯掉，因此分成兩個矩陣作用就會得到正確的結果。

```
N_S = Matrix4(
    scale, 0.0f, 0.0f, 0.0f,
    0.0f, scale, 0.0f, 0.0f,
    0.0f, 0.0f, scale, 0.0f,
    0.0f, 0.0f, 0.0f, 1.0f);
N_T = Matrix4(
    1.0f, 0.0f, 0.0f, (-1)*mid_x,
    0.0f, 1.0f, 0.0f, (-1)*mid_y,
    0.0f, 0.0f, 1.0f, (-1)*mid_z,
    0.0f, 0.0f, 0.0f, 1.0f);
N = N_S * N_T;
```

## 2. V(View):

在實作 View 的部分，首先我們要先自訂三個重要的 Vector3，分別是 eye, center, up，而訂定完後我們可以依據 Supplement 中所敘述的方法來修正 up 的值，這樣，我們就可以透過上課講義中提到的方法 (CG04 #69, #71) 求出我們要的 viewing matrix。

### (1) up 的修正:



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)



```
Vector3 forward = center - eye;
Vector3 right = forward.cross(up);
Vector3 up_vector = right.cross(forward);
up = eye + up_vector;
```

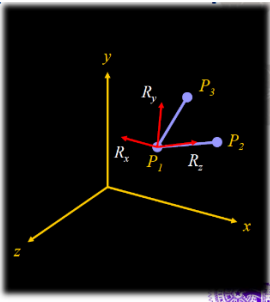
## (2) Viewing matrix 計算:

$$M_{\text{view}} = R \bullet T = \begin{bmatrix} r_{1x} & r_{1y} & r_{1z} & 0 \\ r_{2x} & r_{2y} & r_{2z} & 0 \\ r_{3x} & r_{3y} & r_{3z} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -eye_x \\ 0 & 1 & 0 & -eye_y \\ 0 & 0 & 1 & -eye_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R = \begin{bmatrix} r_{1x} & r_{1y} & r_{1z} \\ r_{2x} & r_{2y} & r_{2z} \\ r_{3x} & r_{3y} & r_{3z} \end{bmatrix}$$

$$R_z = [r_{1z} \ r_{2z} \ r_{3z}]^T = \frac{P_1 P_2}{|P_1 P_2|}$$

$$R_x = [r_{1x} \ r_{2x} \ r_{3x}]^T = \frac{P_1 P_2 \times P_1 P_3}{|P_1 P_2 \times P_1 P_3|}$$

$$R_y = [r_{1y} \ r_{2y} \ r_{3y}]^T = R_x \times R_z$$


```
Vector3 Rz = (center - eye) / (center - eye).length();
Vector3 Rx = (center - eye).cross(up - eye) / ((center - eye).cross(up - eye)).length();
Vector3 Ry = Rx.cross(Rz);

Matrix4 V_R;
V_R.setRow(0, Rx);
V_R.setRow(1, Ry);
V_R.setRow(2, -Rz);
V_R.setRow(3, Vector4(0.0f, 0.0f, 0.0f, 1.0f));

Matrix4 V_T = Matrix4(
    1.0f, 0.0f, 0.0f, (-1)*eye.x,
    0.0f, 1.0f, 0.0f, (-1)*eye.y,
    0.0f, 0.0f, 1.0f, (-1)*eye.z,
    0.0f, 0.0f, 0.0f, 1.0f);

Matrix4 V = V_R * V_T;
```

## 3. P (Projection):

Projection 是指我們的投影方式，根據要求我們要能在 Orthographic 和 Perspective 作切換，所以就可以依據講義 (CG04 #123, #125) 的公式實作這兩種方法的 matrix，其中的 top, right 我給 1.0f，bottom, left 我給 -1.0f，這樣符合我們 window 是在 [-1, 1] 間，而 near 跟 far 經過調整分別給 1.0f, 5.0f。底下實作 code 圖中變數對應的就是公式中的值。

### (1) Orthographic:

$$\begin{pmatrix} \frac{2}{\text{Right}-\text{Left}} & 0 & 0 & t_x \\ 0 & \frac{2}{\text{Top}-\text{Bottom}} & 0 & t_y \\ 0 & 0 & \frac{-2}{\text{Far}-\text{Near}} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{matrix} t_x = -\frac{\text{Right}+\text{Left}}{\text{Right}-\text{Left}} \\ t_y = -\frac{\text{Top}+\text{Bottom}}{\text{Top}-\text{Bottom}} \\ t_z = -\frac{\text{Far}+\text{Near}}{\text{Far}-\text{Near}} \end{matrix}$$

```
Matrix4(
    orthox, 0.0f, 0.0f, tx,
    0.0f, orthoy, 0.0f, ty,
    0.0f, 0.0f, orthoz, tz,
    0.0f, 0.0f, 0.0f, 1.0f);
```

## (2) Perspective:

$$\begin{pmatrix} \frac{2 \text{ Near}}{\text{Right} - \text{Left}} & 0 & A & 0 \\ 0 & \frac{2 \text{ Near}}{\text{Top} - \text{Bottom}} & B & 0 \\ 0 & 0 & C & D \\ 0 & 0 & -1 & 0 \end{pmatrix} \begin{matrix} A = \frac{\text{Right} + \text{Left}}{\text{Right} - \text{Left}} \\ B = \frac{\text{Top} + \text{Bottom}}{\text{Top} - \text{Bottom}} \\ C = -\frac{\text{Far} + \text{Near}}{\text{Far} - \text{Near}} \\ D = -\frac{2 \text{ Far Near}}{\text{Far} - \text{Near}} \end{matrix}$$

```
Matrix4(
    perx, 0.0f, A, 0.0f,
    0.0f, pery, B, 0.0f,
    0.0f, 0.0f, C, D,
    0.0f, 0.0f, -1.0, 0.0f);
```

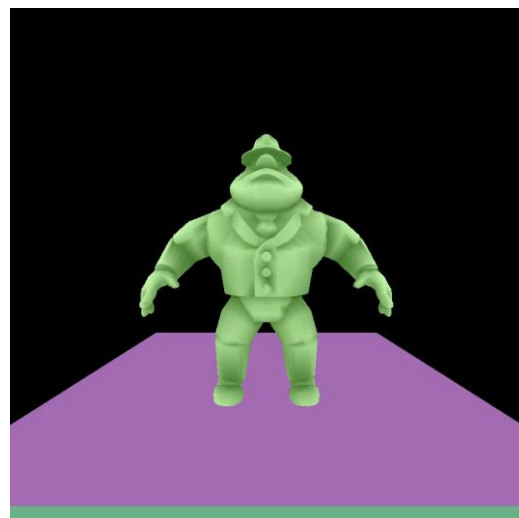
## 四、 Other efforts or worth-mentioned:

這次的 code 中，因為有很多是可以經過人工微調的變數，因此我盡量將所有變數集合在一起，這樣方便作控管和知道其中的差異性；而在作按鍵；也就是印出現在資訊的功能時，我是利用再開一個 2D char array 存每個 mode 的名字，這樣就能很清楚的知道現在的 filename 和 mode。最後，在滑鼠的控制部分，因為他傳入的位置是整數，因此我們算出來的差一開始也會是整數，但因為我們 normalized 到  $[-1, 1]$  間所以會使一次改變太多，所以要除上一定的數使功能看起來是明顯的。

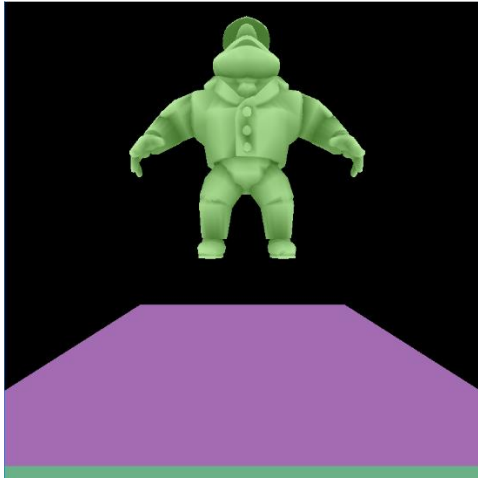
## 五、 Results:



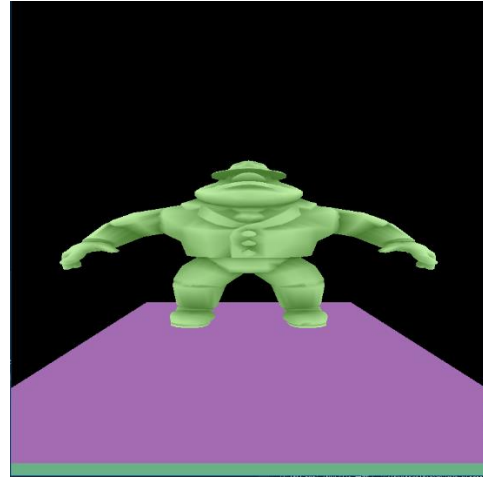
Orthographic



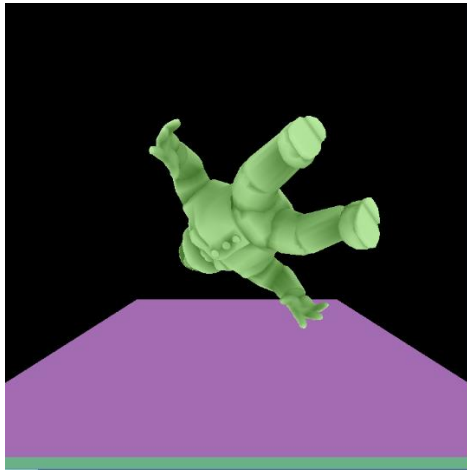
Perspective



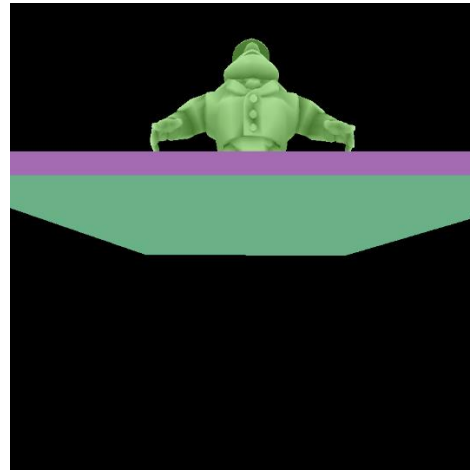
**Translate**



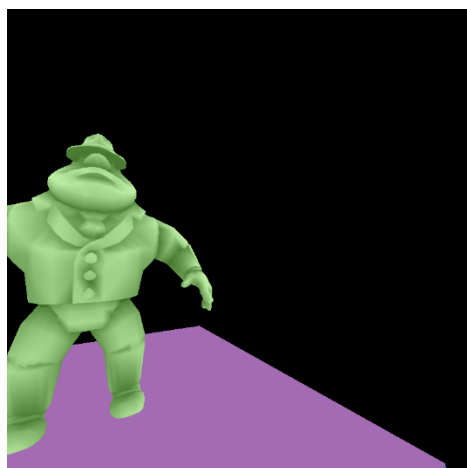
**Scale**



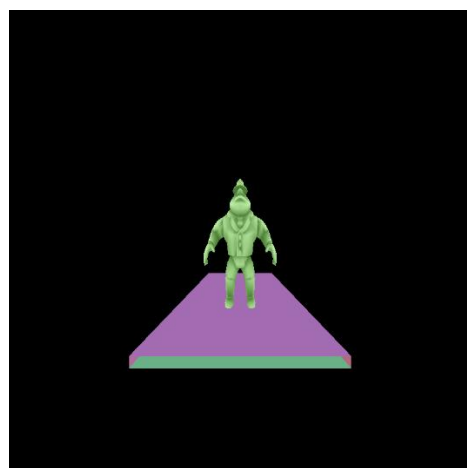
**Rotate**



**Eye translate**



**Look at translate**



**Projection transformation**