Computer Graphics

Project #3, Viewing and Projection

姓名：林一
系級：資工 3B
學號：00957101

A、 The manual for controlling the motions

機器人移動

編號	按鍵	功能	附註
1	w	前進(跑步)	踩到地板時有音效
2	a	向右旋轉	
3	s	後退(跑步)	踩到地板時有音效
4	d	向左旋轉	
5	Shift + w	前進(走路)	踩到地板時有音效
6	Shift + s	後退(走路)	踩到地板時有音效
7	兩下 r	機器人傳送至原點	

機器人動作

編號	按鍵	功能	附註
1	5	揮動左手	
2	6	揮動右手	
3	7	揮動雙手	
4	8	跳舞	
5	r	回到預設動作	
6	t	手指運動	手指運動時有機械音效
7	9	蹲下	
8	0	蹲下並跳起	跳起時有音效
9	空白鍵	直接跳起	跳起時有音效

其他

編號	按鍵	功能	附註
1	ESC	關閉程式	

視角與透視模式

編號	按鍵	功能	附註
	`	顯示四種透視頁面	
	1	顯示 X direction parallel viewing	
	2	顯示 Y direction parallel viewing	
	3	顯示 Z direction parallel viewing	
	4	顯示 Perspective viewing	
	tab	開啟與關閉 View volume	顯示可視範圍
	滑鼠任一按鍵長壓	將所有頁面以 Perspective viewing 顯示	
	+	讓 camera 視角跟隨機器人	再按一次回復

Camera 移動

編號	按鍵	功能	附註
	z	向上平移	
	x	向下平移	
	Ctrl + w	前進	
	Ctrl + a	向左平移	
	Ctrl + s	後退	
	Ctrl + d	向右平移	

Camera 視角方向

編號	按鍵	功能	附註
	i	向前翻滾	pitching
	j	向左翻滾	rolling
	k	向後翻滾	pitching
	l	向右翻滾	rolling
	,	向左旋轉	heading
	.	向右旋轉	heading

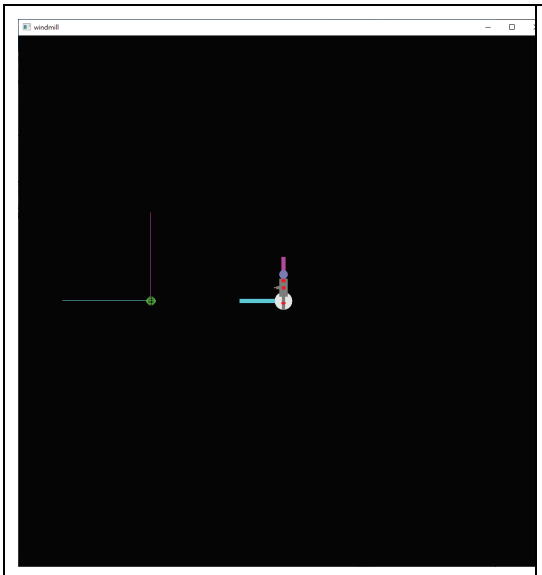
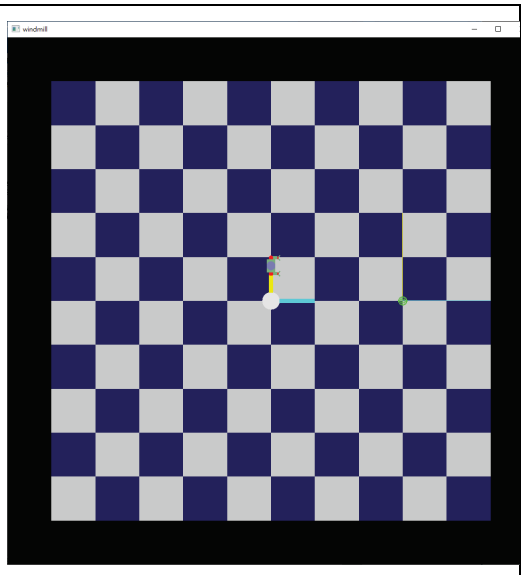
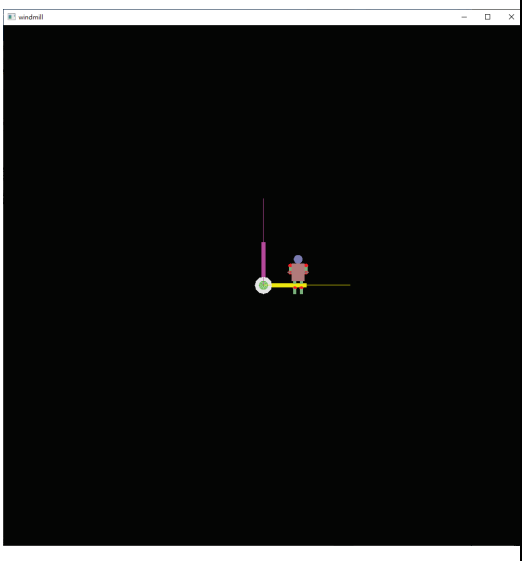
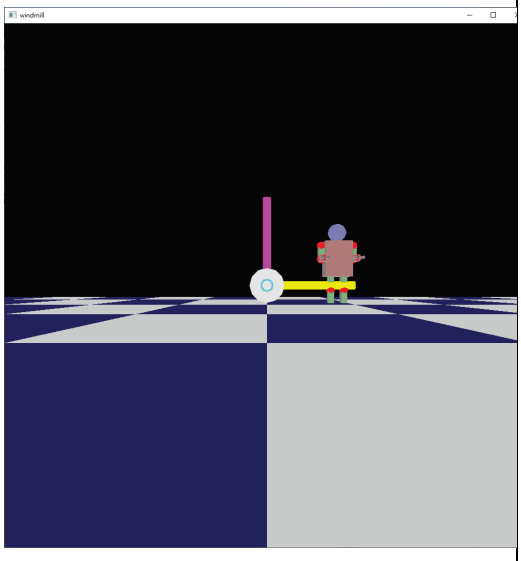
Zoom in and zoom out

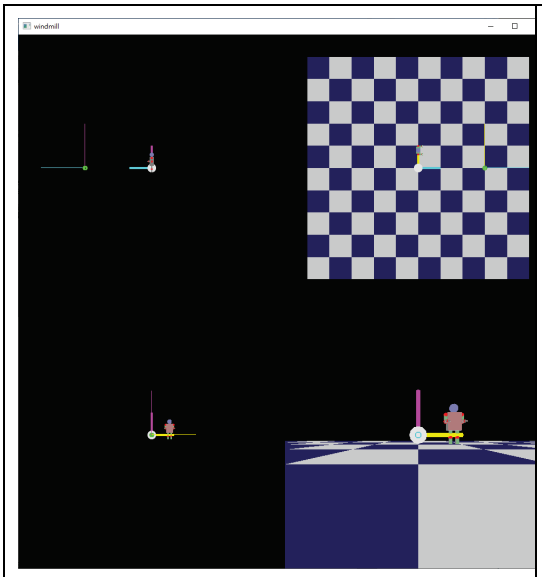
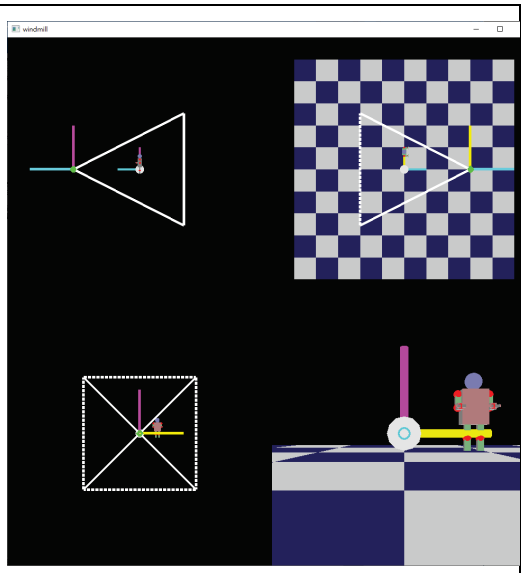
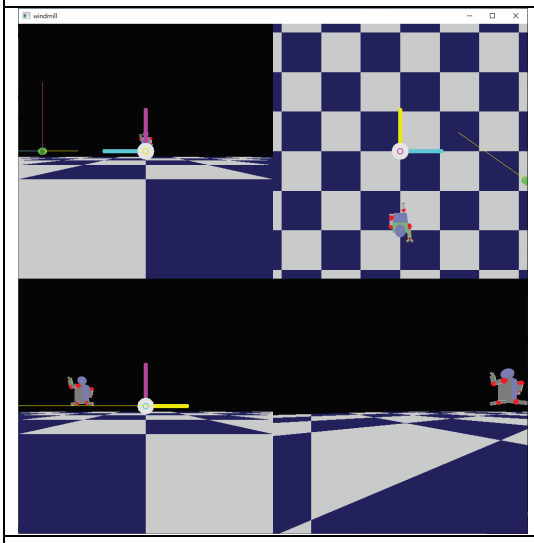
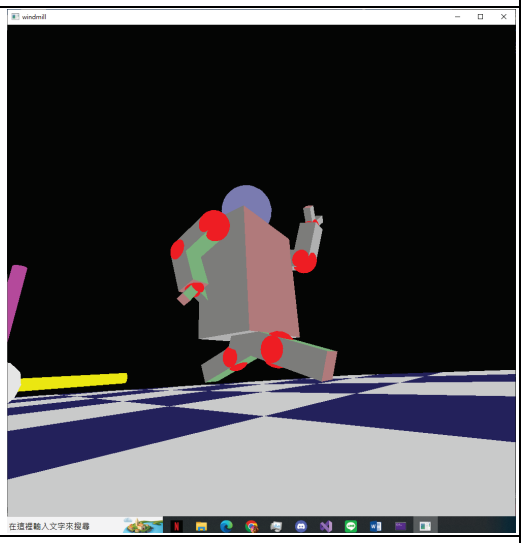
編號	按鍵	功能	附註
	滑鼠向前滾輪	Zoom in	
	滑鼠向後滾輪	Zoom out	

B、 Ideas and algorithms

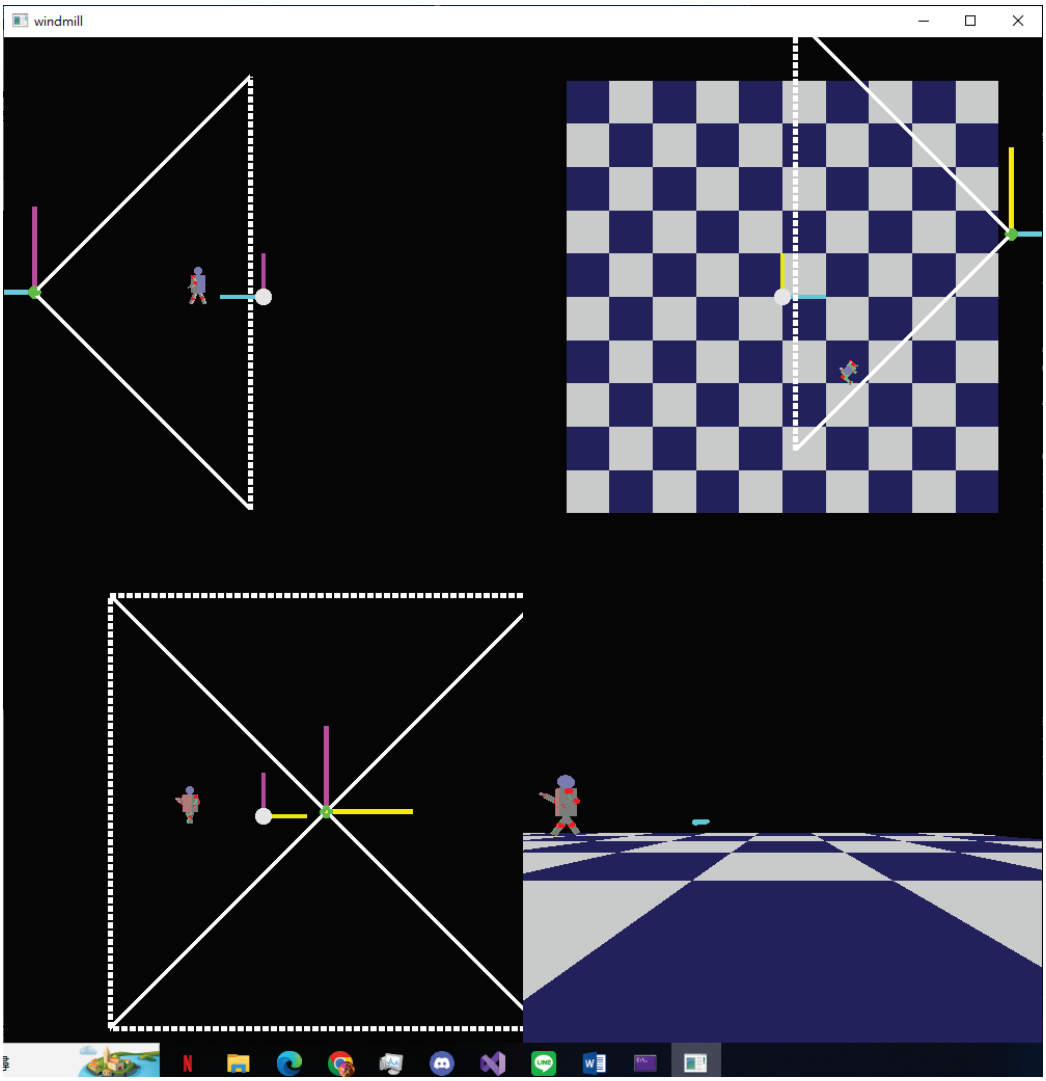
加入多項功能：可改變的視角與透視模式、Camera 移動、Camera 視角方向、Zoom in and zoom out。

(1) 視角與透視模式

<table>
<tr><td>  </td><td>  </td></tr>
<tr><td>顯示 X direction parallel viewing</td><td>顯示 Y direction parallel viewing</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td>顯示 Z direction parallel viewing</td><td>顯示 Perspective viewing</td></tr>
</table>

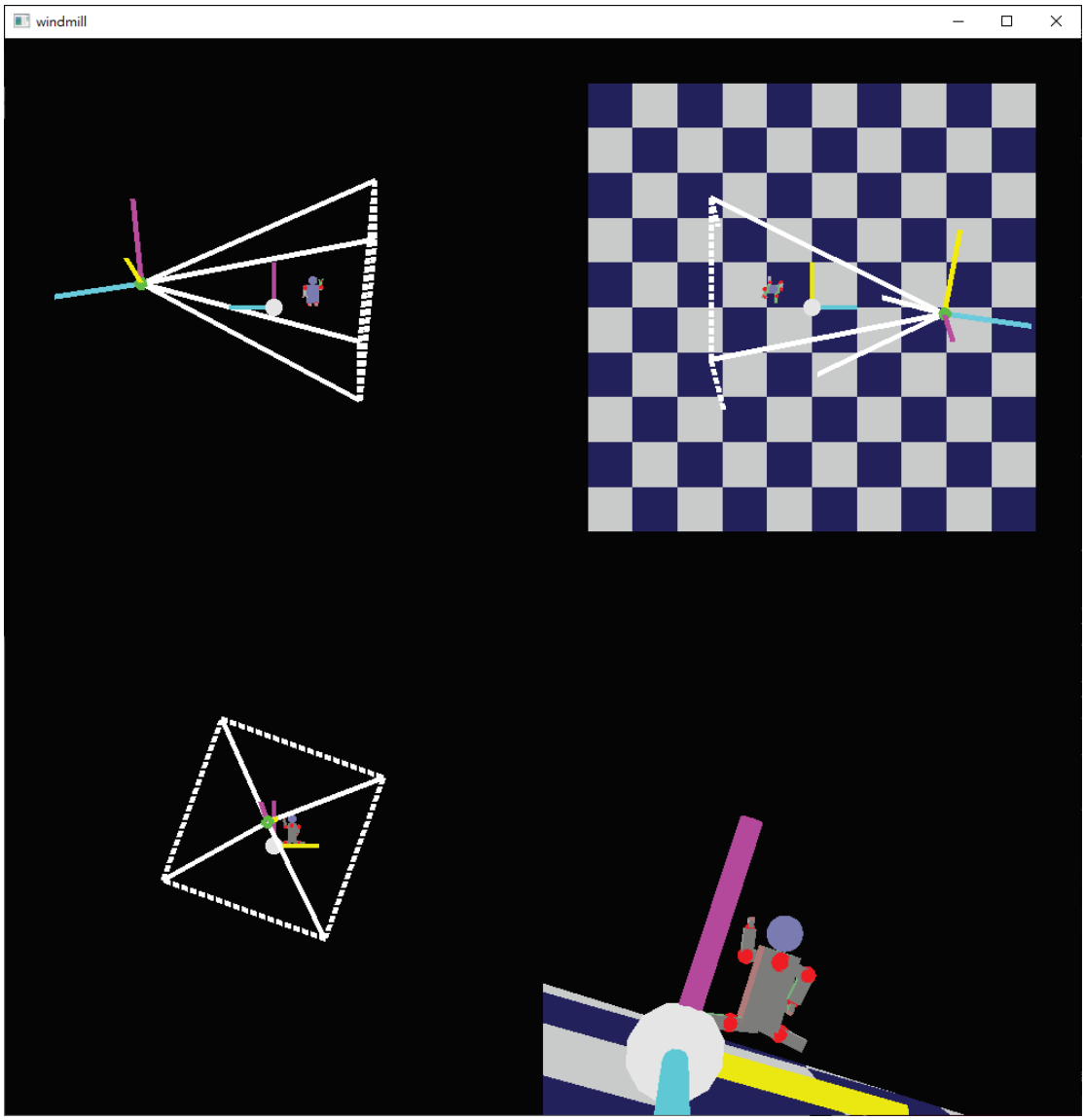
	
四種模式同時顯示。	開啟 View volume 顯示可視範圍
	
將所有模式改為 Perspective viewing	讓 Camera 視線跟隨機器人

(2) Camera 移動



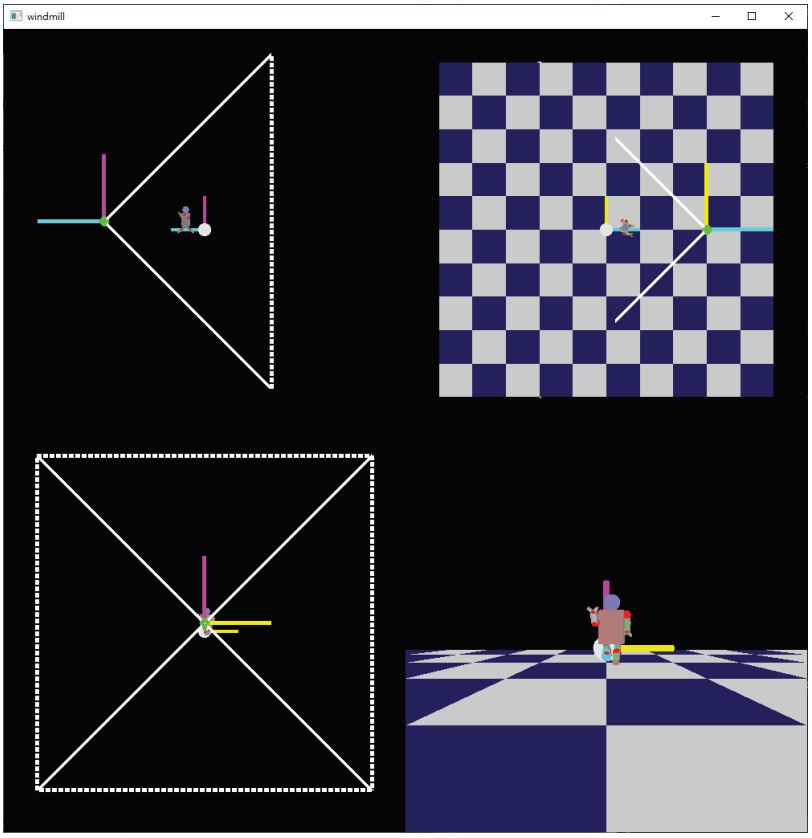
上圖可以觀察到可視範圍會隨著 Camera 移動。

(3) Camera 視角方向

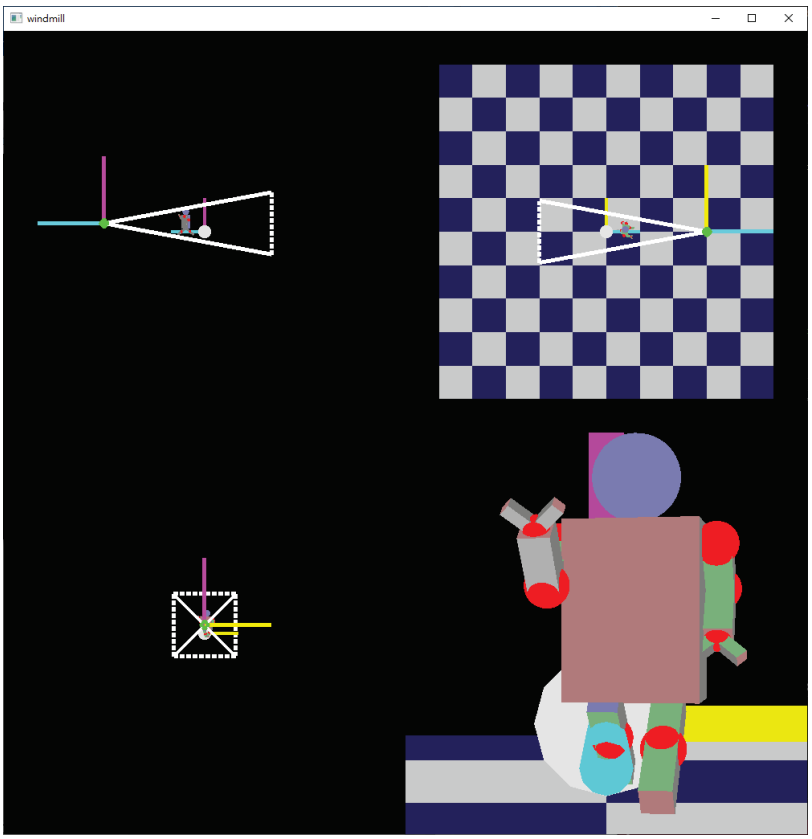


上圖可以觀察到可視範圍會隨著 Camera 的視角改變方向。

(4) Zoom in and zoom out



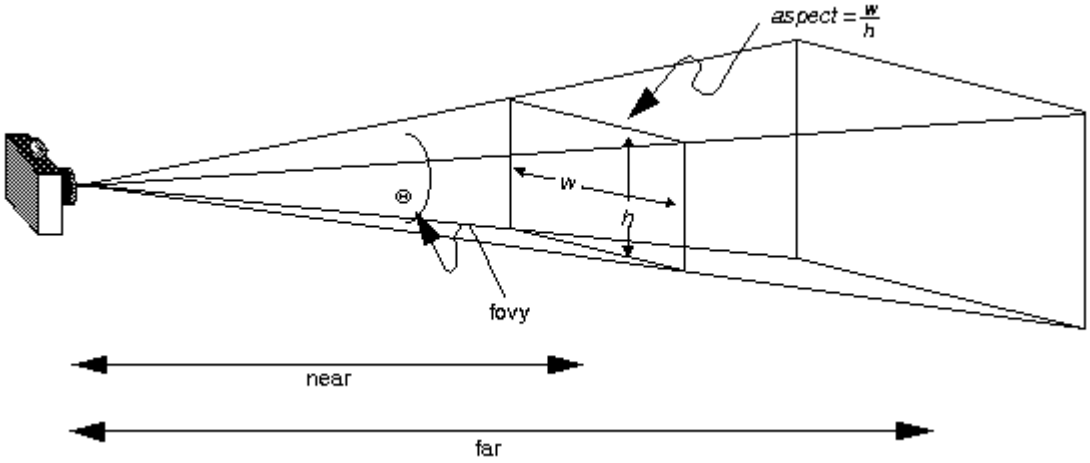
Zoom out
將視線的角度放大，有類似廣角的效果。



Zoom in
將視線的角度縮小，有類似望遠的效果。

(5) 數學

View volume

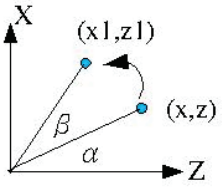


可視範圍 View volume 可以透過三角函數去推算：

```
// asp = w/h
GLfloat nearH = 2 * tan(fovy / 2 * PI / 180.0) * zNear;
GLfloat nearW = nearH * ((double)width / (double)height);
GLfloat farH = 2 * tan(fovy / 2 * PI / 180.0) * zFar;
GLfloat farW = farH * ((double)width / (double)height);
```

只要分別求得 near 與 far 面中 w 與 h 的高度，就能透過座標的平移找到八個目標點，先畫出四條線形成三角錐，接著再將 near 面與 far 面分別用四個點相連畫線，就能畫出面，在空間中這就是一金字塔的樣子。

View volume **的旋轉**



$$\frac{x1}{L} = \frac{x}{L} * \cos\beta + \frac{z}{L} * \sin\beta$$

$$\frac{z1}{L} = \frac{z}{L} * \cos\beta - \frac{x}{L} * \sin\beta$$

View volum 會跟隨著 Camera 視角方向、距離而有所改變，當遇到旋轉的情況就必須去考慮其旋轉軸與旋轉角度，就能推算出旋轉後的位置（如右邊的公視）。

繞Y軸旋轉

$$x1 = x*\cos\beta + z*\sin\beta$$
$$y1 = y$$
$$z1 = -x*\sin\beta + z*\cos\beta$$

繞X軸旋轉

$$x2 = x1$$
$$y2 = y1*\cos\alpha - z1*\sin\alpha$$

繞Z軸旋轉

$$x3 = x2*\cos\gamma - y2*\sin\gamma$$
$$y3 = x2*\sin\gamma + y2*\cos\gamma$$

```
struct rotate_node {
    GLfloat x, y, z;
    rotate_node() {}
    rotate_node(GLfloat _x, GLfloat _y, GLfloat _z) {
        x = _x; y = _y; z = _z;
    }
};
// 矩陣旋轉
rotate_node rotate(GLfloat x, GLfloat y, GLfloat z) {
    // y
    GLfloat x1 = x * cos(eyeAngy * PI / 180.0) + z * sin(eyeAngy * PI / 180.0);
    GLfloat y1 = y;
    GLfloat z1 = -x * sin(eyeAngy * PI / 180.0) + z * cos(eyeAngy * PI / 180.0);

    // x
    GLfloat x2 = x1;
    GLfloat y2 = y1 * cos(eyeAngx * PI / 180.0) - z1 * sin(eyeAngx * PI / 180.0);
    GLfloat z2 = y1 * sin(eyeAngx * PI / 180.0) + z1 * cos(eyeAngx * PI / 180.0);

    // z
    GLfloat x3 = x2 * cos(eyeAngz * PI / 180.0) - y2 * sin(eyeAngz * PI / 180.0);
    GLfloat y3 = x2 * sin(eyeAngz * PI / 180.0) + y2 * cos(eyeAngz * PI / 180.0);
    GLfloat z3 = z2;

    // result
    return rotate_node(x3, y3, z3);
```

參考資料：

https://blog.csdn.net/wu4long/article/details/6126408?utm_medium=distribute.pc_relevant.none-task-blog-2~default~baidujs_baidulandingword~default-4-6126408-blog-53106457.pc_relevant_multi_platform_whitelistv3&spm=1001.2101.3001.4242.3&utm_relevant_index=7

https://openhome.cc/Gossip/ComputerGraphics/Rotate3Dimension.htm