

## 電腦圖學 Project 02

### 目次

電腦圖學 Project 02.....	1
API 文件.....	2
My_GLU.h.....	2
perspective.....	2
lookAt.....	2
Frustum.h.....	3
class Frustum_2D.....	3
演算法.....	4
投影到 2 維座標並繪製.....	4
Cell Portal.....	5
Frustum_2D::clip.....	6
視錐的初始化.....	7

## API 文件

### My\_GLU.h

下列函數都宣告在 namespace My 中。

- **perspective**

自己實作的 gluPerspective。建立一個 perspective 投影矩陣，並用 glmMultMatrixd 和原本的矩陣相乘。

- **lookAt**

自己實作的 gluLookAt。建立一個用來將 world 座標轉成 view 座標的座標轉換矩陣，並用 glmMultMatrixd 和原本的矩陣相乘。

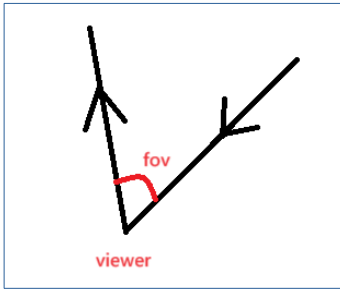
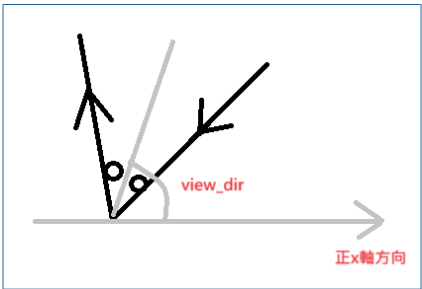
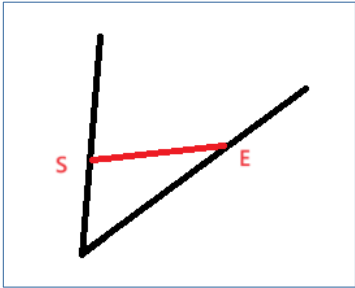
## Frustum.h

下列 class 宣告在 namespace My 中。

- **class Frustum\_2D**

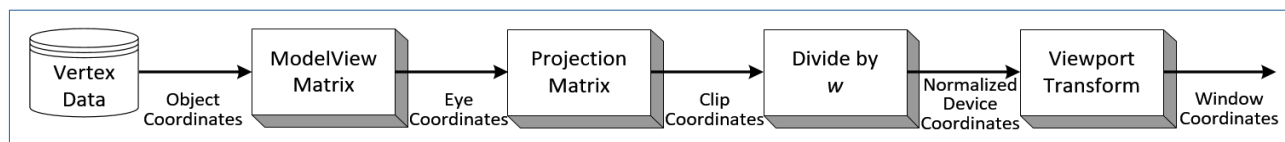
以左、右兩直線來表示 2D 平面上的視錐。

### member function

(constructor)	<p>給定 viewer 的座標、視角 ( fov )、和面向方向的方位角 ( view_dir )，並初始化視錐。</p> <div style="display: flex; justify-content: space-around; align-items: center;">   </div>
restrict (static)	<p>建立一個剛好包住一線段的視錐。( S 和 E 的順序不影響 )</p> <div style="text-align: center;">  </div>
clip	<p>將直線 clip，使其在視錐的範圍內。</p>

## 演算法

### 投影到 2 維座標並繪製



Source: [https://www.songho.ca/opengl/gl\\_transform.html](https://www.songho.ca/opengl/gl_transform.html)

OpenGL 在顯示時會將物體的座標乘上 ModelView Matrix 得到 view 座標、再乘上 Projection Matrix 得到 clip 座標、再將  $(x, y, z)$  除以  $w$  得到 NDC 座標（稱作 perspective division）、最後再做 Viewport Transformation 得到 window 座標。

NDC 座標很類似 window 座標，只不過它是獨立於螢幕設備的表示法—— $x$ 、 $y$  代表要畫在螢幕的哪裡，而  $z$  則代表深度。

所以，對於原本要用 `glVertex3f` 畫的座標，我們可以自己手動轉成 NDC 座標，並用 `glVertex2f` 將 NDC 座標的  $x$ 、 $y$  繪製上去。

【註 1】`GL_MODELVIEW` 和 `GL_PROJECTION` 這兩個矩陣要設成單位方陣，這樣用 `glVertex2f` 畫上去的座標才會原封不動的保留下來變 NDC 座標。

【註 2】Viewport Transformation 是由 `glViewport` 定義的，這部分讓 OpenGL 自己做就好。

【註 3】在 NDC 座標中，只有  $x$ 、 $y$ 、 $z$  都在  $[-1, 1]$  間的點才會被畫在螢幕上。因此照理說，在 `glVertex2f` 前要先判斷  $z$  的大小才決定要不要畫。不過，如果 clip 和 cell portal 都有做對的話，那麼畫出的點都會是螢幕上會顯示的點，所以不需要去檢查  $z$  的大小。

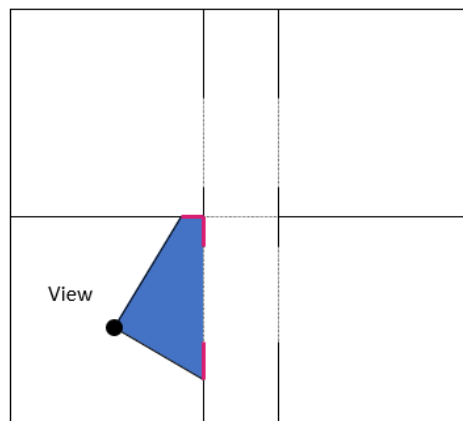
## Cell Portal

```

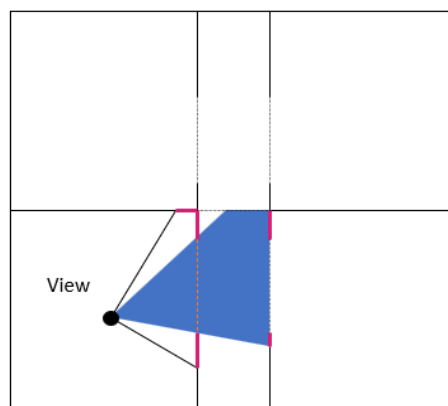
Draw_Cell(cell C, frustum F) {
  for each cell edge E {
    if E is opaque {
      E' = clip E to F
      draw E'
    }
    if E is transparent {
      E' = clip E to F
      F' = F restricted to E'
      Draw_Cell(neighbor(C, E), F')
    }
  }
}

```

if E is opaque  
E' = clip E to F  
draw E'



if E is transparent  
E' = clip E to F  
F' = F restricted to E'



和投影片講得差不多。

## Frustum\_2D::clip

對於視錐的每個邊界：

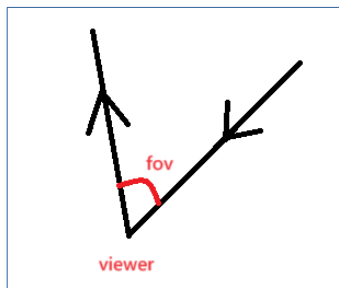
if 線段的兩端都在邊界的右側： 保持不變

else if 兩端都在左側： return false

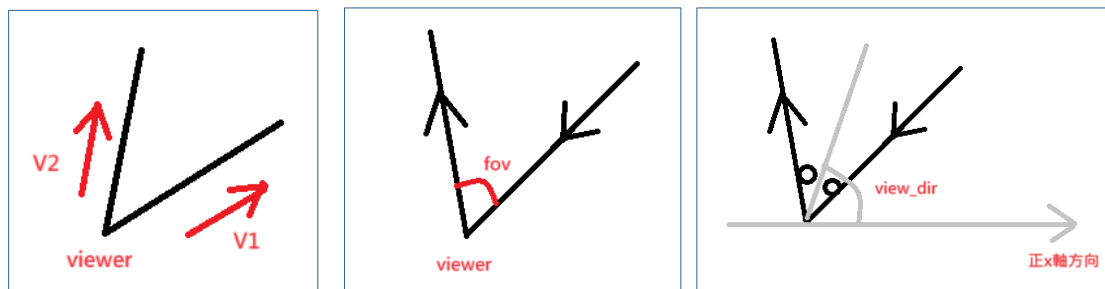
else： 將邊界左側的端點設成邊界和線段的交點

return true

邊界的方向性如下：



## 視錐的初始化



假設 $\vec{V}_1$ 和 $\vec{V}_2$ 兩向量分別平行視錐的右邊和左邊的兩直線。

則兩向量的方位角分別是 $\theta_1 = \text{view\_dir} - \text{fov} / 2$  和  $\theta_2 = \text{view\_dir} + \text{fov} / 2$ 。

所以可以定 $\vec{V}_1 = (\cos(\theta_1), \sin(\theta_1))$  ·  $\vec{V}_2 = (\cos(\theta_2), \sin(\theta_2))$ 。

因此：

右邊的直線為 點  $\text{viewer} + \vec{V}_1$  和 點  $\text{viewer}$  所連直線。

左邊的直線為 點  $\text{viewer}$  和 點  $\text{viewer} + \vec{V}_2$  所連直線。