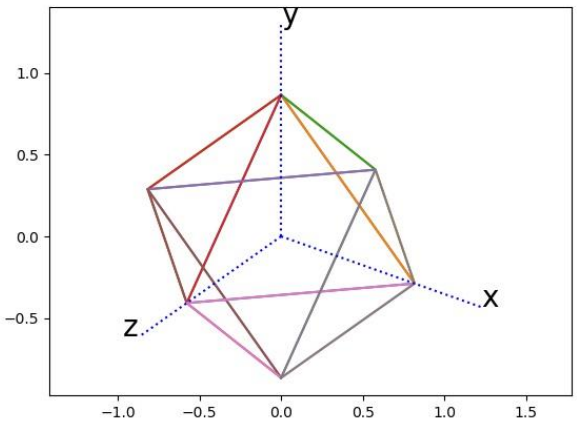
HW3 Report

104070038 陳羿先

1. Design a convex object with at least 7 faces

我設計的是一個以圖形中心為(0,0,0)、稜長為√2的正八面體，如此設計後，正八面體的六個座標點分別為(±1,0,0)、(0,±1,0)、(0,0,±1)。



2. my transformation matrices

(1). r1: 將圖形沿著 xy 平面旋轉 30 度(rotate)

```
r1 = np.array([[m.sqrt(3)/2, 1/2, 0],
               [-1/2, m.sqrt(3)/2, 0],
               [0, 0, 1]])
```

(2). r2: 將八面體在 x 方向扭曲(skew)

```
r2 = np.array([[1, 0, 0.5],
               [0, 1, 0],
               [0, 0, 1]])
```

(3). r3: 將八面體在 x 方向放大(enlarge)

```
r3 = np.array([[4, 0, 0],
               [0, 1, 0],
               [0, 0, 1]])
```

(3). r4: 將八面體在 z 方向扭曲(skew)

```
r4 = np.array([[1, 0, 0],
               [0, 1, 0],
               [2, 0, 1]])
```

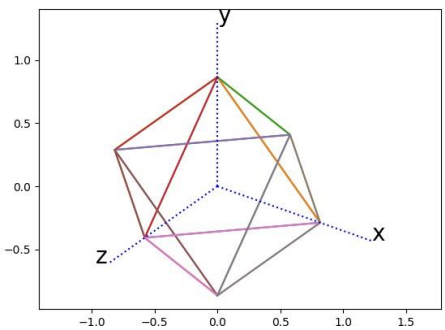
3. 如何消除隱藏面

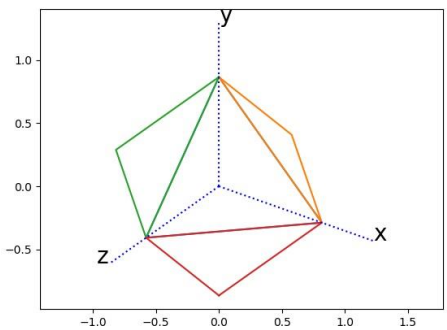
首先，因為正八面體每一面皆為三角形，因此對於每個面來說，我們擁有 p1、p2、p3 三個頂點座標，透過 p2-p1、p3-p1，我們可以得到兩組位於此平面上的 vectors，並將此兩組 vectors 做 cross product，即可得到八面體每個面的法向量。值得注意的是，因為法向量的方向是由右手定則決定，因此在做 cross product 時需要特別注意，我們所需的是指向八面體外側的那個法向量。
而另一方面，我們要在意的就是 view point，將 view point 與法向量做內積的結果，即可決定該平面是否需要被隱藏(從視角來看是看不到的)，如果此內積值小於 0，代表可以被看見；如果大於等於 0，代表這個面無法被看見，如此就可以完成消除隱藏面的工作。

```
def visible(p1, p2, p3):
    """output if the face is visible."""
    # write your code here...

    v = points[:, p2] - points[:, p1]
    w = points[:, p3] - points[:, p1]

    normal = np.cross(w,v)

    view = np.array([[1/2],
                     [1/2],
                     [m.sqrt(2)/2]])
    num = np.dot(normal, view)
    if num < 0:
        return True
    else:
        return False
```

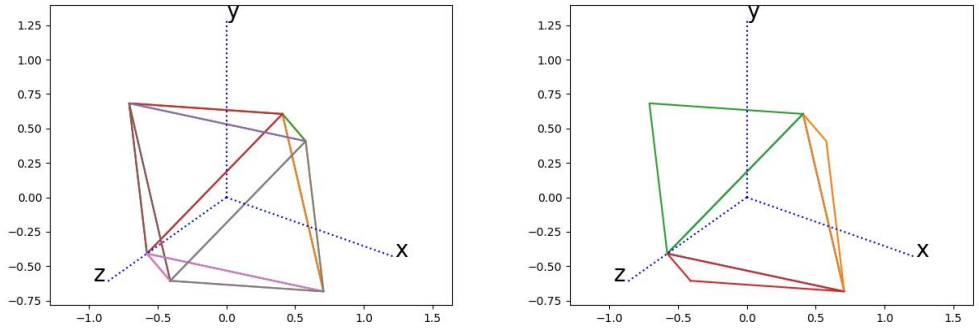
4. 結果圖

(1). 原圖(正八面體)

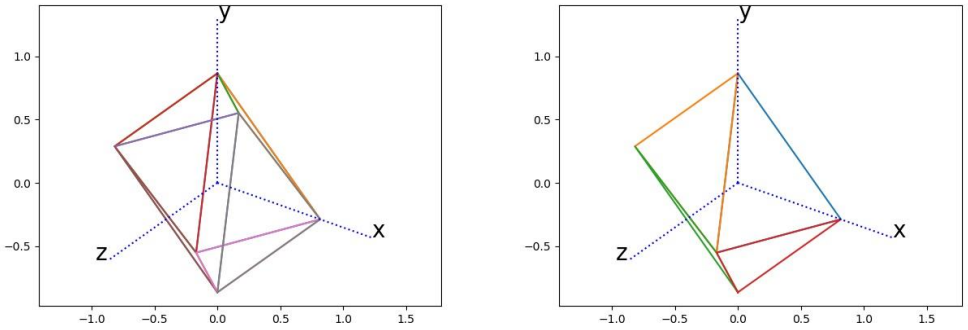




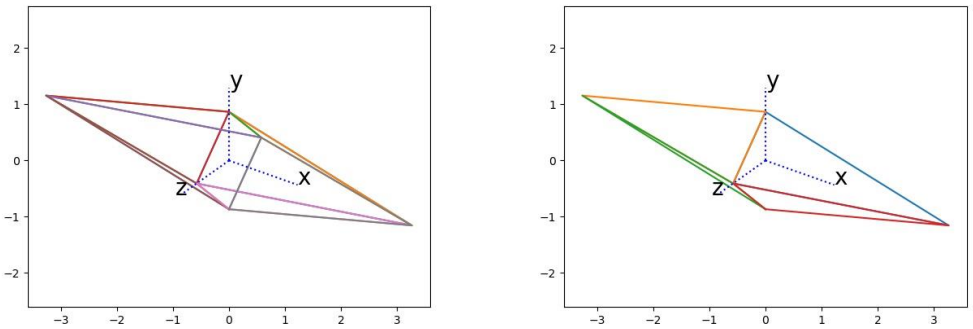
(2). r1：將圖形沿著 xy 平面旋轉 30 度(rotate)



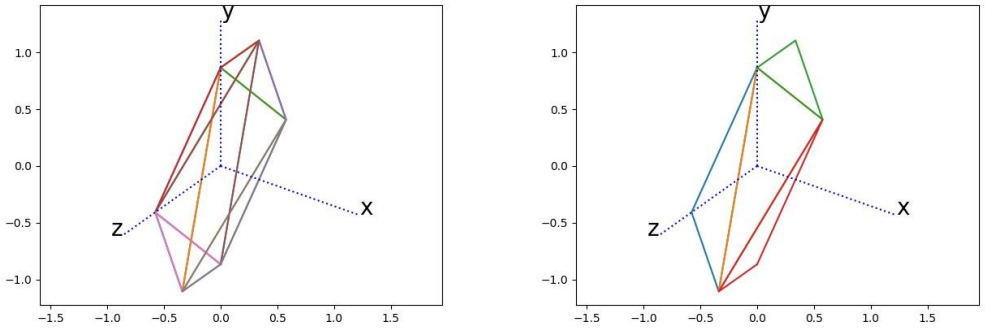
(3). r2：將八面體在 x 方向扭曲(skew)



(4). r3: 將八面體在 x 方向放大(enlarge)

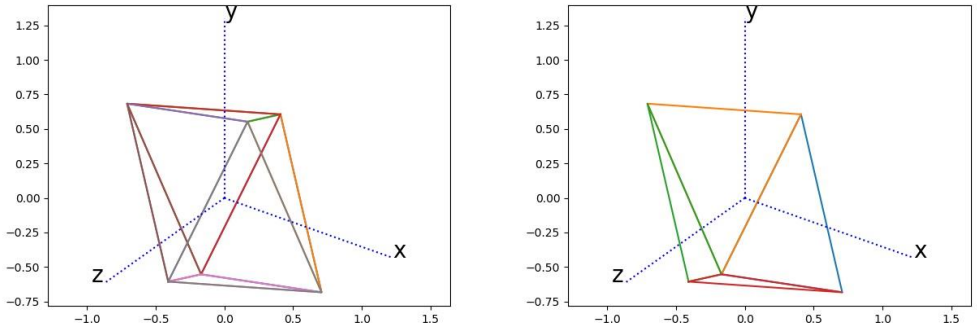


(5). r4: 將八面體在 z 方向扭曲(skew)



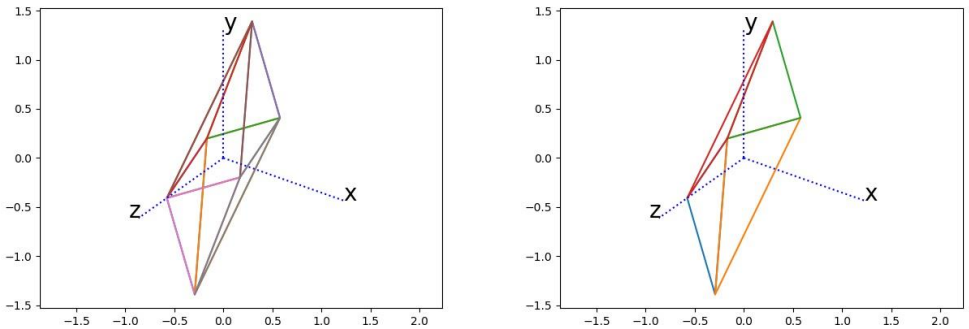
(6). 先將圖形沿著 xy 平面旋轉 30 度，再將八面體在 x 方向扭曲

```
points = np.dot(r1, points)
points = np.dot(r2, points)
```



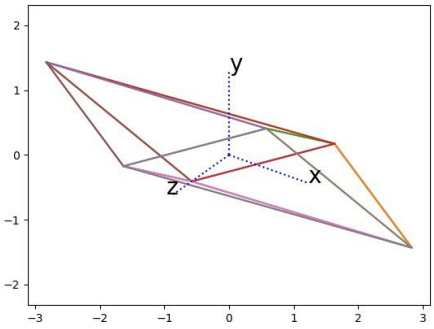
(7). 先將圖形沿著 xy 平面旋轉 30 度，再將八面體在 z 方向扭曲

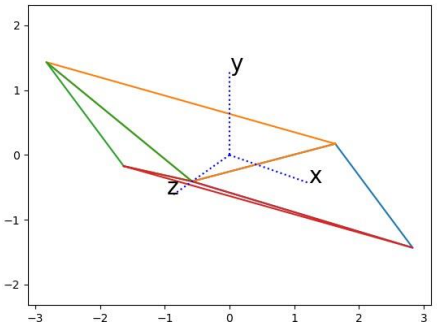
```
points = np.dot(r1, points)
points = np.dot(r4, points)
```



(8). 先將圖形沿著 xy 平面旋轉 30 度，再將八面體在 x 方向放大

```
points = np.dot(r1, points)
points = np.dot(r3, points)
```





(9). 先將八面體在 x 方向扭曲，再將八面體在 z 方向扭曲

```
points = np.dot(r2, points)
points = np.dot(r4, points)
```