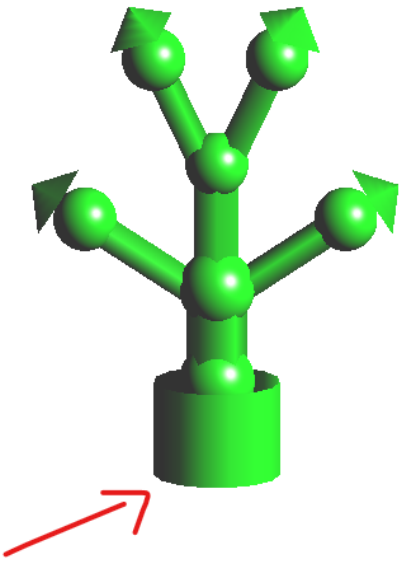
電腦圖學 作業三 機械手臂

資工三乙 408262325 應名宥

date : 2021/12/23

1. 程式架構
    a. 動作常數
        i. 負責在menu的地方決定要進行那些動作
        ii.
```
/*motion 常數*/
#define NONE 0
#define MOTION_1 1
#define MOTION_2 2
#define MOTION_3 3
#define MOTION_4 4
#define MOTION_5 5
#define MOTION_6 6
```
    b. 底座
        i. 物件的底座，以圓柱體組成，使用glPushMatrix與glPopMatrix來確保修改不會改變到未來的物件。
        ii.
```
/*底座*/
void base()
{
    glPushMatrix();
    glRotatef(-90.0, 1.0, 0.0, 0.0);
    gluCylinder(p, BASE_RADIUS, BASE_RADIUS, BASE_HEIGHT, 15, 15);
    glPopMatrix();
}
```
        iii.



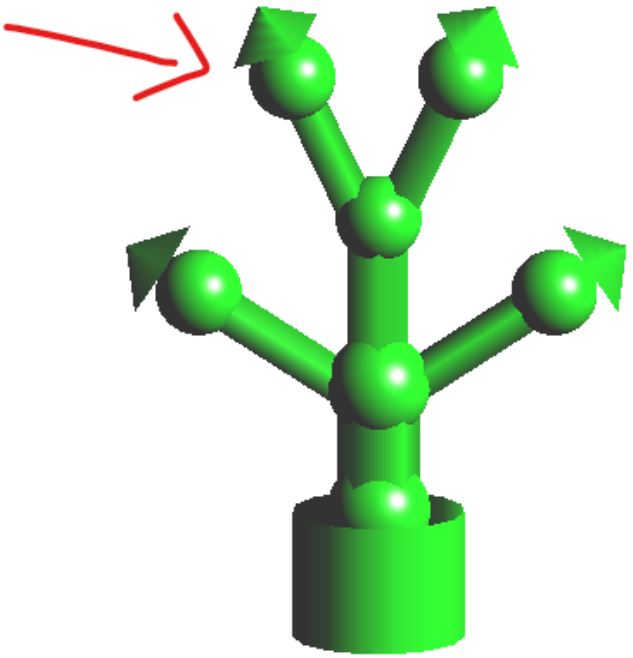
c. 左上臂(外肢臂)

i. 物件的左上臂，以一個圓柱體，一個圓球，一個四角錐組成，使用glPushMatrix與glPopMatrix來確保修改不會改變到未來的物件。

ii.

```
/*左上臂*/
void upper_left_arm()
{
    glPushMatrix();
    glRotatef(theta[6], 1.0, 0.0, 0.0);
    glRotatef(theta[7], 1.0, 0.0, 0.0);
    glRotatef(theta[7], 1.0, 0.0, 1.0);
    glRotatef(theta[7], 1.0, 0.0, 1.0);
    glRotatef(-90.0, 1.0, 0.0, 0.5);
    gluCylinder(p,UPPER_ARM_WIDTH, UPPER_ARM_WIDTH, UPPER_ARM_HEIGHT, 4, 4);
    glTranslatef(0.0, 0.0, UPPER_ARM_HEIGHT);
    gluSphere(p, 1, 100, 100);

    glTranslatef(0.0, 0.0, 1);
    glRotatef(theta[5], 0.0, 0.0, 1.0);
    gluCylinder(p, 1.2, 0, 1, 4, 4);
    glPopMatrix();
}
```



iii.

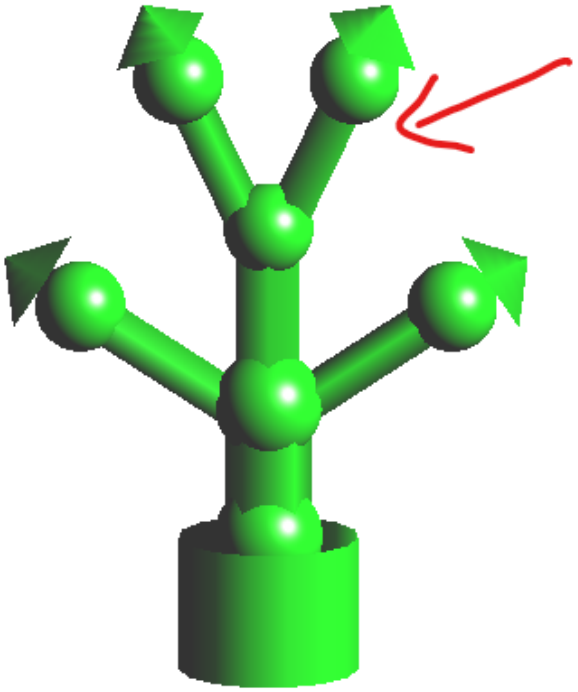
d. 右上臂(外肢臂)

i. 物件的右上臂，以一個圓柱體，一個圓球，一個四角錐組成，使用glPushMatrix與glPopMatrix來確保修改不會改變到未來的物件。

```
/*右上臂*/
void upper_right_arm()
{
    glPushMatrix();
    glRotatef(theta[6], 1.0, 0.0, 0.0);
    glRotatef(theta[7], 1.0, 0.0, 0.0);
    glRotatef(theta[7], 1.0, 0.0, -1.0);
    glRotatef(theta[7], 1.0, 0.0, -1.0);
    glRotatef(-90.0, 1.0, 0.0, -0.5);
    gluCylinder(p, UPPER_ARM_WIDTH, UPPER_ARM_WIDTH, UPPER_ARM_HEIGHT, 4, 4);
    glTranslatef(0.0, 0.0, UPPER_ARM_HEIGHT);
    gluSphere(p, 1, 100, 100);

    glTranslatef(0.0, 0.0, 1);
    glRotatef(theta[5], 0.0, 0.0, 1.0);
    gluCylinder(p, 1.2, 0, 1, 4, 4);
    glPopMatrix();
}
```

ii.



iii.

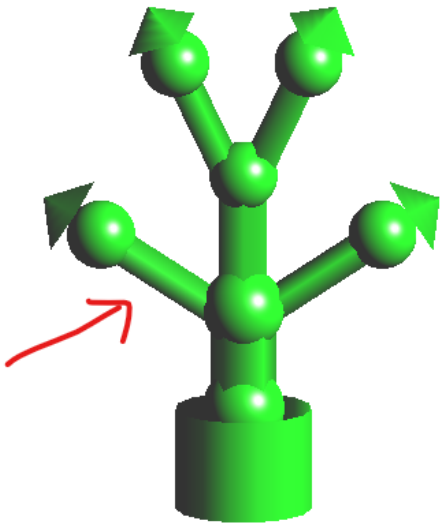
e. 左中臂(外肢臂)

i. 物件的左中臂，以一個圓柱體，一個圓球，一個四角錐組成，使用glPushMatrix與glPopMatrix來確保修改不會改變到未來的物件。

ii.

```
/*左中臂*/
void middle_left_arm()
{
    glPushMatrix();
    glTranslatef(0.7,0, 0);
    glRotatef(theta[3], 1.0, 0.0, 0.0);
    glRotatef(theta[4], 0.0, 0.0, 1.0);
    glRotatef(theta[6], 1.0, 0.0, 0.0);
    glRotatef(theta[7], 1.0, 0.0, 0.0);
    glRotatef(theta[7], 1.0, 0.0, 1.0);
    glRotatef(theta[7], 1.0, 0.0, 1.0);
    glRotatef(theta[7], 1.0, 0.0, 1.0);
    glRotatef(theta[7], 1.0, -1.0, 1.0);
    glRotatef(-90.0, 1.0, 0.0, 0.5);
    glRotatef(-90.0, 1.0, 0.0, 2);
    gluCylinder(p,UPPER_ARM_WIDTH, UPPER_ARM_WIDTH, UPPER_ARM_HEIGHT, 4, 4);
    glTranslatef(0.0, 0.0, UPPER_ARM_HEIGHT);
    gluSphere(p, 1, 100, 100);

    glTranslatef(0.0, 0.0, 1);
    glRotatef(theta[5], 0.0, 0.0, 1.0);
    gluCylinder(p, 1.2, 0, 1, 4, 4);
    glRotatef(theta[3], 1.0, 0.0, 0.0);
    glRotatef(theta[4], 0.0, 0.0, 1.0);
    glPopMatrix();
}
```



iii.

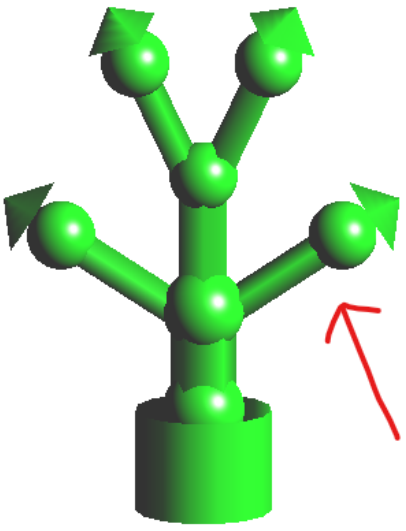
f. 右中臂(外肢臂)

i. 物件的右中臂，以一個圓柱體，一個圓球，一個四角錐組成，使用glPushMatrix與glPopMatrix來確保修改不會改變到未來的物件。

ii.

```
/*右中臂*/
void middle_right_arm()
{
    glPushMatrix();
    glTranslatef(-0.8,0, 0);
    glRotatef(theta[3], 1.0, 0.0, 0.0);
    glRotatef(theta[4], 0.0, 0.0, -1.0);
    glRotatef(theta[4], 0.0, 0.0, -1.0);
    glRotatef(theta[6], 1.0, 0.0, 0.0);
    glRotatef(theta[7], 1.0, 0.0, 0.0);
    glRotatef(theta[7], 1.0, 0.0, -1.0);
    glRotatef(theta[7], 1.0, 0.0, -1.0);
    glRotatef(theta[7], 1.0, 0.0, -1.0);
    glRotatef(theta[7], 1.0, 1.0, -1.0);
    glRotatef(-90.0, 1.0, 0.0, -0.5);
    glRotatef(-90.0, 1.0, 0.0, -2);
    gluCylinder(p,UPPER_ARM_WIDTH, UPPER_ARM_WIDTH, UPPER_ARM_HEIGHT, 4, 4);
    glTranslatef(0.0, 0.0, UPPER_ARM_HEIGHT);
    gluSphere(p, 1, 100, 100);

    glTranslatef(0.0, 0.0, 1);
    glRotatef(theta[5], 0.0, 0.0, 1.0);
    gluCylinder(p, 1.2, 0, 1, 4, 4);
    glPopMatrix();
}
```



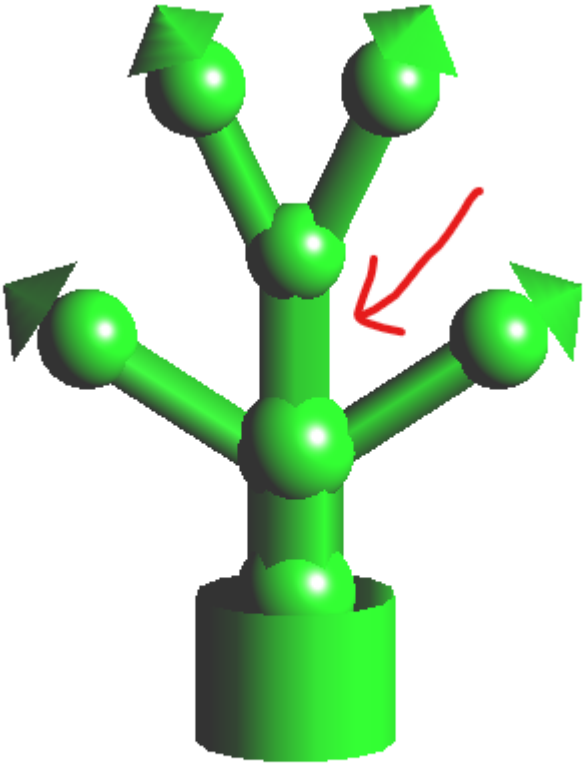
iii.

g. 中臂

i. 物件的中臂，以圓柱體組成，使用glPushMatrix與glPopMatrix來確保修改不會改變到未來的物件。

ii.

```
/*中臂*/
void middle_arm()
{
    glPushMatrix();
    gluSphere(p, 1.2, 100, 100);
    glRotatef(-90.0, 1.0, 0.0, 0.0);
    gluCylinder(p, MIDDLE_ARM_WIDTH, MIDDLE_ARM_WIDTH, MIDDLE_ARM_HEIGHT, 4, 4);
    glPopMatrix();
}
```



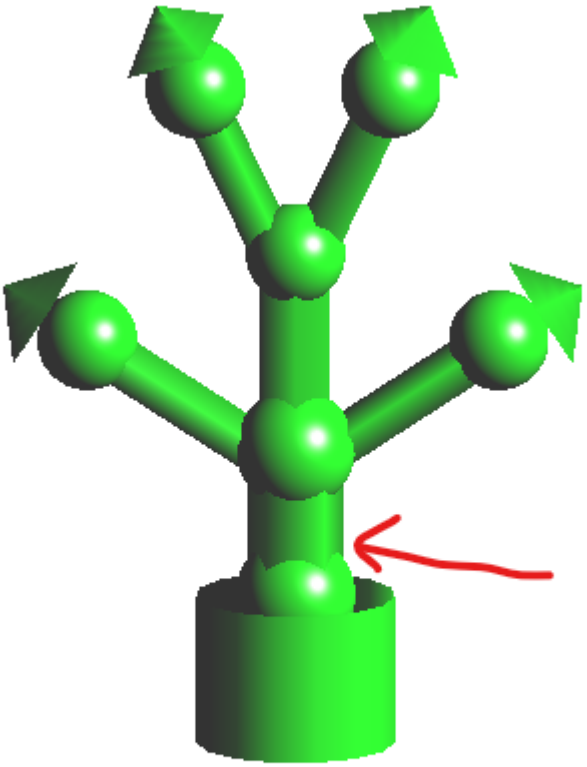
iii.

h. 下臂

i. 物件的下臂，以圓柱體組成，使用glPushMatrix與glPopMatrix來確保修改不會改變到未來的物件。

ii.

```
/*下臂*/
void lower_arm()
{
    glPushMatrix();
    gluSphere(p, 1.2,100, 100);
    glRotatef(-90.0, 1.0, 0.0, 0.0);
    gluCylinder(p, LOWER_ARM_WIDTH, LOWER_ARM_WIDTH, LOWER_ARM_HEIGHT, 5, 5);
    glPopMatrix();
}
```



iii.

i. 點擊事件

i. 可以調整當前選擇的區塊(上臂、中臂、下臂、底座)，進行移動。

ii.

```
void mouse(int btn, int state, int x, int y)
{
    if(btn==GLUT_LEFT_BUTTON && state == GLUT_DOWN)
    {
        theta[axis] += 5.0;
        if( theta[axis] > 360.0 ) theta[axis] -= 360.0;
    }
    if(btn==GLUT_RIGHT_BUTTON && state == GLUT_DOWN)
    {
        theta[axis] -= 5.0;
        if( theta[axis] < 360.0 ) theta[axis] += 360.0;
    }
    display();
}
```

j. 初始化使用的變數

i.

```
/*初始化數值*/
void reset(){
    theta[0] = 0.0;
    theta[1] = 0.0;
    theta[2] = 0.0;
    theta[3] = 0.0;
    theta[4] = 0.0;
    theta[5] = 0.0;
    theta[6] = 0.0;
    theta[7] = 0.0;
    mStep1 = 5;
    mStep2 = 5;
    mStep3 = 3;
    mStep4 = 3;
    mStep5 = 3;
    mStep6 = 2;
}
```

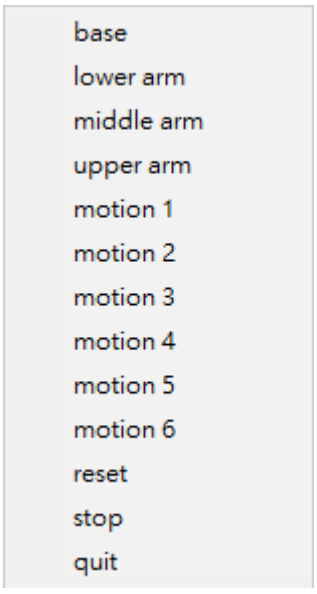
k. menu

i. 可選擇當前調整位置(上臂、中臂、下臂、底座)，以及選擇動作、重製位置、停止與跳出。

ii.

```
void menu(int id)
{
    motion=NONE;
    if(id == 1 ){
        axis=0;
    }
    else if(id == 2 ){
        axis=1;
    }
    else if(id == 3 ){
        axis=2;
    }
    else if(id == 4 ){
        axis=3;
    }
```

iii.



I. 改變視角與物件位置

i. 透過鍵盤的輸入來控制使用者的視角，與物件的位置。

ii.

```
void keys(unsigned char key, int x, int y)
{
    /*改變視角*/
    if(key == 'x') viewer[0]-= 0.3;
    else if(key == 'X') viewer[0]+= 0.3;
    else if(key == 'y') viewer[1]-= 0.3;
    else if(key == 'Y') viewer[1]+= 0.3;
    else if(key == 'z') viewer[2]-= 0.3;
    else if(key == 'Z') viewer[2]+= 0.3;

    /*改變模型位置*/
    else if(key == 'w') movefb -= 1.0;
    else if(key == 's') movefb += 1.0;
    else if(key == 'a') movelr -= 1.0;
    else if(key == 'd') movelr += 1.0;

    display();
}
```

m. 動作函數

i. 透過動作函數來決定物件當前該怎麼做動作，當移動超過上下限時就把移動的步伐變號，來達到來回移動的效果，帶入的數字必須被步伐整除。

ii.

```
/*動作函數*/
void TimerFunction(int value)
{
    if (value == MOTION_1)
    {
        if (que(theta[axis],70.0) || que(theta[axis],-70.0))
        {
            mStep1 = -mStep1;
        }

        if (theta[axis] <= 70.0 && theta[axis] >= -70.0)
        {
            theta[axis] += mStep1;
        }
    }
```

n. 初始化函數

i. 控制與燈光有關的屬性(燈光的位置、環境光強度、散數光、反射光等等很多的屬性)，來達到顯示物件不同的樣子，也定義了建立物件的變數。

ii.
```
void myinit()
{
    /*控制燈光*/
    GLfloat mat_specular[]= {1.0, 1.0, 1.0, 1.0};
    GLfloat mat_diffuse[]= {1.0, 1.0, 1.0, 1.0};
    GLfloat mat_ambient[]= {1.0, 1.0, 1.0, 1.0};
    GLfloat mat_shininess= {50.0};
    GLfloat light_ambient[]= {0.0, 0.0, 0.0, 1.0};
    GLfloat light_diffuse[]= {0.0, 1.0, 0.0, 1.0};
    GLfloat light_specular[]= {1.0, 1.0, 1.0, 1.0};
    GLfloat light_position[]= {10.0, 5.0, 10.0, 0.0};
```

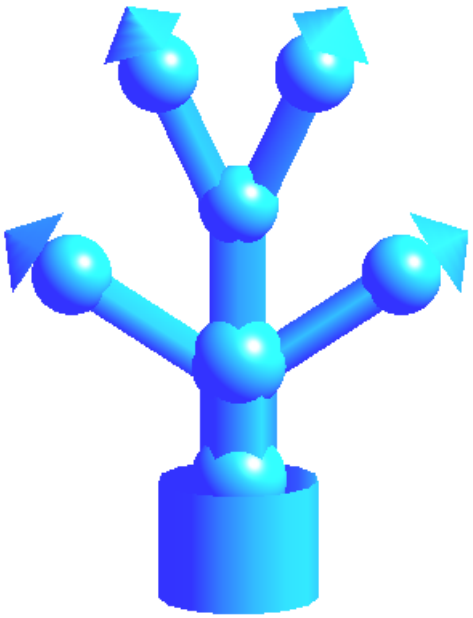
iii.
```
    glClearColor(1.0, 1.0, 1.0, 1.0);
    glColor3f(1.0, 0.0, 0.0);

    p=gluNewQuadric();
    gluQuadricDrawStyle(p, GLU_FILL);
}
```

2. 討論

a. 改變模組的顏色

i. 在用gluNewQuadric生成的物件下單純使用glColor3f(1.0, 0.0, 0.0)無法改變顏色，直接使用的話會是單純的藍色。需要改變燈光位置、環境光強度、反射散射與其他很多的部分才能達到改顏色的目的，在這次作業查了許多資料，但還是搞不太懂怎麼調整會改變那些地方，嘗試了很多次才更改成自己想要的顏色，希望更熟悉後可以簡單一點辦到。

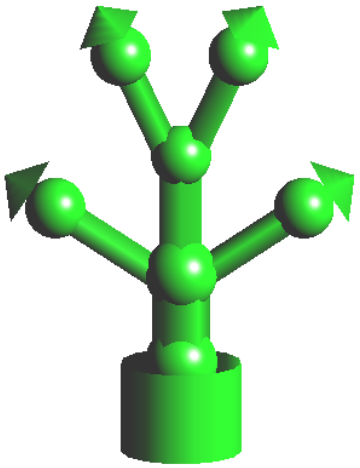


ii.

b. 調整物件的位置
    i. 因為整個畫面變成3維空間了，所以很難在腦中構思怎麼調整可以讓物體出現在想要的位置，也就導致了每個都試試看的情況。
c. reset不完全
    i. 在展示時按下menu的reset鍵後沒辦法把物件很好的變成原本的樣子。明明控制的角度都設定成0了，但結果是總會差一點，猜測是數值還尚未更新完全就display了，是了許多方法但都沒辦法解決，覺得是一個很可惜的地方。
d. 浮點數比較
    i. 在加入新的動作時發現到程式可能會因為浮點數與整數比較的精度問題導致畫面不會動，所以使用fabs函數來解決發生的精度問題，當誤差<=1e-6時就判斷是同一個數字，問題就解決了。
e. 時間差問題
    i. 在選擇一個動作後再點選一個動作時，有可能因為時間差的問題，而導致參數可能尚未初始化完全，就直接進行下一個動作了，所以需要在每次使用前都把參數調整成相應步數的倍數，就可以正常的運作了。

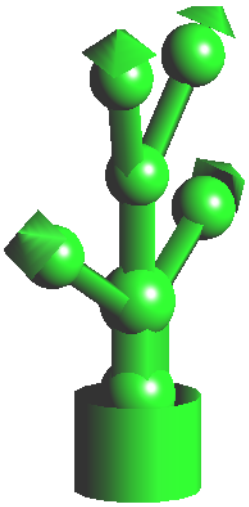
3. 執行畫面

a. 正常情況



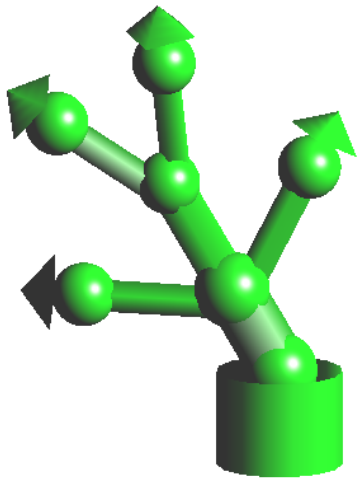
i.

b. 使用選單選擇base後可以用左鍵與右鍵讓底座旋轉。



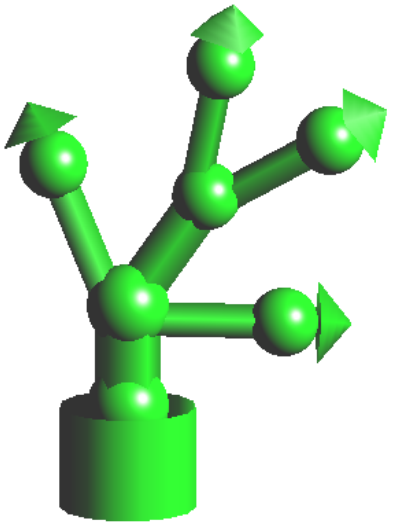
i.

c. 使用選單選擇lower arm後可以用左鍵與右鍵讓下臂往左或右擺。



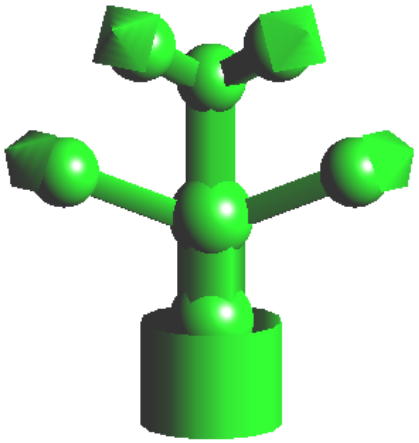
i.

d. 使用選單選擇middle arm後可以用左鍵與右鍵讓中臂往左或右擺。



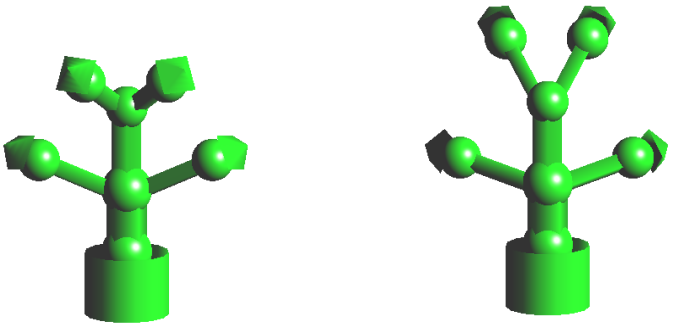
i.

e. 使用選單選擇upper arm後可以用左鍵與右鍵讓上臂(外肢臂)往前或後擺。



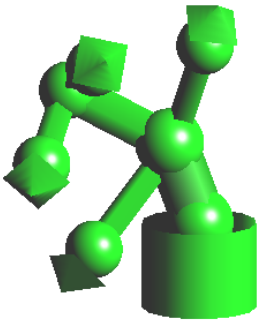
i.

f. 使用選單選擇motion 1後可以讓先前選擇的區塊進行來回動作(base,lower arm...等等)。



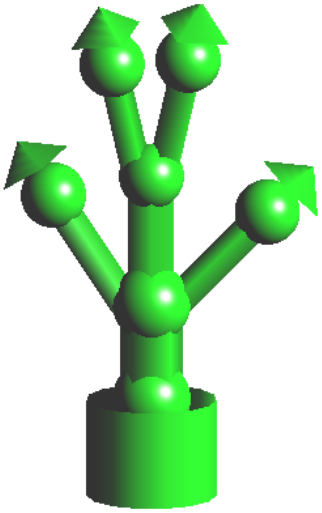
i.

g. 使用選單選擇motion 2後可以讓機械手臂全部的部位都開始擺動。



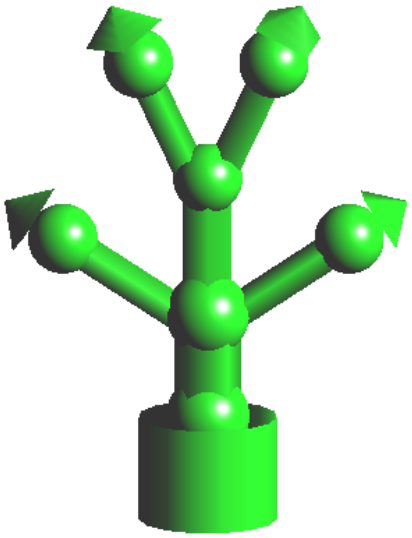
i.

h. 使用選單選擇motion 3後可以讓外肢臂開合動作，跟夾東西類似。



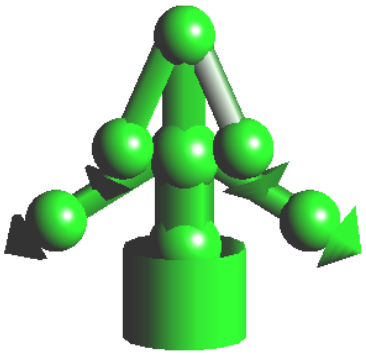
i.

i. 使用選單選擇motion 4後可以讓外肢臂上的鑽頭開始旋轉。



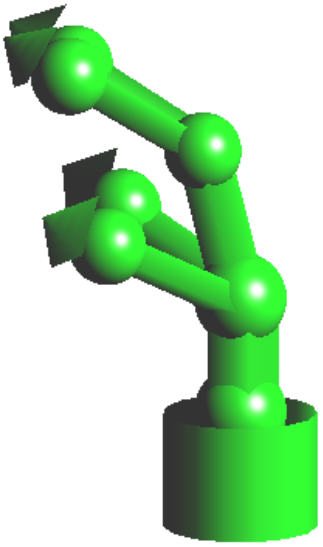
i.

j. 使用選單選擇motion 5後可以讓外肢臂在空中繞圈。



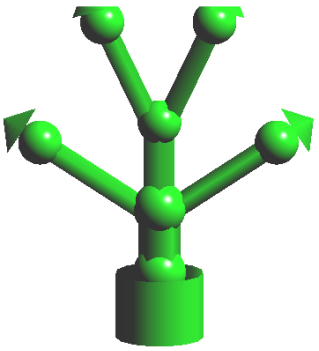
i.

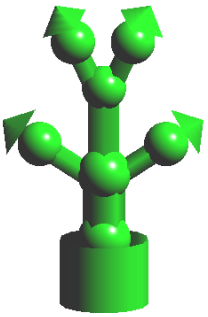
k. 使用選單選擇motion 6後可以讓機械手臂作出類似抓取物品的動作(側面觀看), 側面觀看可以先選選單上的base, 然後選擇motion 6, 再使用滑鼠左右鍵來調整位置。



i.

l. 使用選單選擇motion 7後可以讓機械手臂的外肢臂縮短與伸長





i.

4. 程式碼

```
#include <windows.h>
#include <GL/glut.h>
#include <bits/stdc++.h>


/*模型參數*/
#define BASE_HEIGHT 3.0
#define BASE_RADIUS 2.0
#define LOWER_ARM_HEIGHT 3.0
#define LOWER_ARM_WIDTH 1.0
#define MIDDLE_ARM_HEIGHT 4.0
#define MIDDLE_ARM_WIDTH 0.7
#define UPPER_ARM_WIDTH 0.5
double UPPER_ARM_HEIGHT = 4.0;

/*motion 常數*/
#define NONE 0
#define MOTION_1 1
#define MOTION_2 2
#define MOTION_3 3
#define MOTION_4 4
#define MOTION_5 5
#define MOTION_6 6
#define MOTION_7 7

using namespace std;

/*調整角度*/
static GLfloat theta[] = {0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0};

/*目前控制的區塊，下臂、中臂、上臂*/
static GLint axis = 0;

/*負責製作許多模型物件*/
GLUquadricObj  *p;
```

```
/*觀看視角*/
static GLdouble viewer[]= {0.0, 0.7, 3.0};

/*動作*/
static GLint motion = 0;

/*調整角度移動速度*/
static GLint mStep1 = 5, mStep2 = 5, mStep3 = 3, mStep4 = 3,mStep5 = 3,mStep6 =
2;
static double siz = 0.5;



/*控制物體位置*/
static GLfloat movefb = 0.0;
static GLfloat movelr = 0.0;

/*底座*/
void base()
{
   glPushMatrix();
   glRotatef(-90.0, 1.0, 0.0, 0.0);
   gluCylinder(p, BASE_RADIUS, BASE_RADIUS, BASE_HEIGHT, 15, 15);
   glPopMatrix();
}

/*左上臂*/
void upper_left_arm()
{
   glPushMatrix();
   glRotatef(theta[6], 1.0, 0.0, 0.0);
   glRotatef(theta[7], 1.0, 0.0, 0.0);
   glRotatef(theta[7], 1.0, 0.0, 1.0);
   glRotatef(theta[7], 1.0, 0.0, 1.0);
   glRotatef(-90.0, 1.0, 0.0, 0.5);
   gluCylinder(p,UPPER_ARM_WIDTH, UPPER_ARM_WIDTH,
UPPER_ARM_HEIGHT, 4, 4);
   glTranslatef(0.0, 0.0, UPPER_ARM_HEIGHT);
   gluSphere(p, 1, 100, 100);
```

```
    glTranslatef(0.0, 0.0, 1);
    glRotatef(theta[5], 0.0, 0.0, 1.0);
    gluCylinder(p, 1.2, 0, 1, 4, 4);
    glPopMatrix();
}

/*右上臂*/
void upper_right_arm()
{
    glPushMatrix();
    glRotatef(theta[6], 1.0, 0.0, 0.0);
    glRotatef(theta[7], 1.0, 0.0, 0.0);
    glRotatef(theta[7], 1.0, 0.0, -1.0);
    glRotatef(theta[7], 1.0, 0.0, -1.0);
    glRotatef(-90.0, 1.0, 0.0, -0.5);
    gluCylinder(p, UPPER_ARM_WIDTH, UPPER_ARM_WIDTH,
UPPER_ARM_HEIGHT, 4, 4);
    glTranslatef(0.0, 0.0, UPPER_ARM_HEIGHT);
    gluSphere(p, 1, 100, 100);

    glTranslatef(0.0, 0.0, 1);
    glRotatef(theta[5], 0.0, 0.0, 1.0);
    gluCylinder(p, 1.2, 0, 1, 4, 4);
    glPopMatrix();
}

/*左中臂*/
void middle_left_arm()
{
    glPushMatrix();
    glTranslatef(0.7,0, 0);
    glRotatef(theta[3], 1.0, 0.0, 0.0);
    glRotatef(theta[4], 0.0, 0.0, 1.0);
    glRotatef(theta[6], 1.0, 0.0, 0.0);
    glRotatef(theta[7], 1.0, 0.0, 0.0);
    glRotatef(theta[7], 1.0, 0.0, 1.0);
    glRotatef(theta[7], 1.0, 0.0, 1.0);
    glRotatef(theta[7], 1.0, 0.0, 1.0);
    glRotatef(theta[7], 1.0, -1.0, 1.0);
```

```
    glRotatef(-90.0, 1.0, 0.0, 0.5);
    glRotatef(-90.0, 1.0, 0.0, 2);
    gluCylinder(p,UPPER_ARM_WIDTH, UPPER_ARM_WIDTH,
UPPER_ARM_HEIGHT, 4, 4);
    glTranslatef(0.0, 0.0, UPPER_ARM_HEIGHT);
    gluSphere(p, 1, 100, 100);

    glTranslatef(0.0, 0.0, 1);
    glRotatef(theta[5], 0.0, 0.0, 1.0);
    gluCylinder(p, 1.2, 0, 1, 4, 4);
    glRotatef(theta[3], 1.0, 0.0, 0.0);
    glRotatef(theta[4], 0.0, 0.0, 1.0);
    glPopMatrix();
}

/*右中臂*/
void middle_right_arm()
{
    glPushMatrix();
    glTranslatef(-0.8,0, 0);
    glRotatef(theta[3], 1.0, 0.0, 0.0);
    glRotatef(theta[4], 0.0, 0.0, -1.0);
    glRotatef(theta[4], 0.0, 0.0, -1.0);
    glRotatef(theta[6], 1.0, 0.0, 0.0);
    glRotatef(theta[7], 1.0, 0.0, 0.0);
    glRotatef(theta[7], 1.0, 0.0, -1.0);
    glRotatef(theta[7], 1.0, 0.0, -1.0);
    glRotatef(theta[7], 1.0, 0.0, -1.0);
    glRotatef(theta[7], 1.0, 1.0, -1.0);
    glRotatef(-90.0, 1.0, 0.0, -0.5);
    glRotatef(-90.0, 1.0, 0.0, -2);
    gluCylinder(p,UPPER_ARM_WIDTH, UPPER_ARM_WIDTH,
UPPER_ARM_HEIGHT, 4, 4);
    glTranslatef(0.0, 0.0, UPPER_ARM_HEIGHT);
    gluSphere(p, 1, 100, 100);

    glTranslatef(0.0, 0.0, 1);
    glRotatef(theta[5], 0.0, 0.0, 1.0);
    gluCylinder(p, 1.2, 0, 1, 4, 4);
```

```
    glPopMatrix();
}

/*中臂*/
void middle_arm()
{
    glPushMatrix();
    gluSphere(p, 1.2, 100, 100);
    glRotatef(-90.0, 1.0, 0.0, 0.0);
    gluCylinder(p, MIDDLE_ARM_WIDTH, MIDDLE_ARM_WIDTH,
MIDDLE_ARM_HEIGHT, 4, 4);
    glPopMatrix();
}

/*下臂*/
void lower_arm()
{
    glPushMatrix();
    gluSphere(p, 1.2,100, 100);
    glRotatef(-90.0, 1.0, 0.0, 0.0);
    gluCylinder(p, LOWER_ARM_WIDTH, LOWER_ARM_WIDTH,
LOWER_ARM_HEIGHT, 5, 5);
    glPopMatrix();
}


/*顯示*/
void display(void)
{
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    gluLookAt(viewer[0],viewer[1],viewer[2], 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);

    /*底座*/
    glTranslatef(movelr, 0.0, movefb);
    glRotatef(theta[0], 0.0, 1.0, 0.0);
    base();
    /*下臂*/
    glTranslatef(0.0, BASE_HEIGHT, 0.0);
```

```
    glRotatef(theta[1], 0.0, 0.0, 1.0);
    lower_arm();

    /*中臂*/
    glTranslatef(0.0, LOWER_ARM_HEIGHT, 0.0);
    glRotatef(theta[7], 1.0, 0.0, 0.0);
    glRotatef(theta[2], 0.0, 0.0, 1.0);
    middle_arm();

    glRotatef(theta[9], 0.0, 1.0, 0.0);
    /*左中臂*/
    middle_left_arm();

    /*右中臂*/
    middle_right_arm();
    glRotatef(theta[9], 0.0, -1.0, 0.0);

    /*接和點*/
    glTranslatef(0.3, MIDDLE_ARM_HEIGHT, 0.0);
    glTranslatef(-0.3, 0.0, 0);
    gluSphere(p, 1, 100, 100);

    glTranslatef(0.3, 0.0, 0);

    /*左上臂*/
    glRotatef(theta[3], 1.0, 0.0, 0.0);
    glRotatef(theta[4], 0.0, 0.0, 1.0);
    upper_left_arm();

    /*右上臂*/
    glTranslatef(-0.7,0, 0.0);
    glRotatef(theta[4], 0.0, 0.0, -1.0);
    glRotatef(theta[4], 0.0, 0.0, -1.0);
    upper_right_arm();

    glFlush();
    glutSwapBuffers();
}
```

```
void mouse(int btn, int state, int x, int y)
{
   if(btn==GLUT_LEFT_BUTTON && state == GLUT_DOWN)
   {
      theta[axis] += 5.0;
      if( theta[axis] > 360.0 ) theta[axis] -= 360.0;
   }
   if(btn==GLUT_RIGHT_BUTTON && state == GLUT_DOWN)
   {
      theta[axis] -= 5.0;
      if( theta[axis] < 360.0 ) theta[axis] += 360.0;
   }
   display();
}

/*初始化數值*/
void reset(){
   motion = NONE;
   theta[axis] = 0.0;
   theta[0] = 0.0;
   theta[1] = 0.0;
   theta[2] = 0.0;
   theta[3] = 0.0;
   theta[4] = 0.0;
   theta[5] = 0.0;
   theta[6] = 0.0;
   theta[7] = 0.0;
   mStep1 = 5;
   mStep2 = 5;
   mStep3 = 3;
   mStep4 = 3;
   mStep5 = 3;
   mStep6 = 2;
   UPPER_ARM_HEIGHT = 4;
}


void menu(int id)
```

```
{
    motion=NONE;
    if(id == 1 ){
        axis=0;
    }
    else if(id == 2 ){
        axis=1;
    }
    else if(id == 3 ){
        axis=2;
    }
    else if(id == 4 ){
        axis=3;
    }
    else if(id == 5 ){
        reset();
        motion=MOTION_1;
    }
    else if(id == 6 )
    {
        reset();
        motion=MOTION_2;
    }
    else if(id == 7){
        reset();
        motion = MOTION_3;
    }
    else if(id == 8){
        reset();
        motion = MOTION_4;
    }
    else if(id == 9){
        reset();
        motion = MOTION_5;
    }
    else if(id == 10){
        reset();
        motion = MOTION_6;
    }
```

```
    else if(id == 11){
        reset();
        motion = MOTION_7;
    }
    else if(id == 12 ){
        reset();
    }
    else if(id == 13 ){
        motion=0;
    }
    else if(id == 14 ){
        exit(0);
    }
}

void keys(unsigned char key, int x, int y)
{
    /*改變視角*/
    if(key == 'x') viewer[0]-= 0.3;
    else if(key == 'X') viewer[0]+= 0.3;
    else if(key == 'y') viewer[1]-= 0.3;
    else if(key == 'Y') viewer[1]+= 0.3;
    else if(key == 'z') viewer[2]-= 0.3;
    else if(key == 'Z') viewer[2]+= 0.3;

    /*改變模型位置*/
    else if(key == 'w') movefb -= 1.0;
    else if(key == 's') movefb += 1.0;
    else if(key == 'a') movelr -= 1.0;
    else if(key == 'd') movelr += 1.0;

    display();
}

/*比較兩個浮點數*/
bool que(double a,double b){
    return fabs(a-b)<=1e-6;
}
```

```
/*動作函數*/
void TimerFunction(int value)
{
    if (value == MOTION_1)
    {
        if((GLint)theta[axis]%mStep1!=0){
            theta[axis] = (GLint)((GLint)theta[axis]/mStep1)*mStep1;
        }
        if (que(theta[axis],70.0) || que(theta[axis],-70.0))
        {
            mStep1 = -mStep1;
        }

        if (theta[axis] <= 70.0 && theta[axis] >= -70.0)
        {
            theta[axis] += mStep1;
        }
    }
    else if (value == MOTION_2)
    {
        if((GLint)theta[1]%mStep2!=0){
            theta[1] = (GLint)((GLint)theta[1]/mStep2)*mStep2;
        }
        if((GLint)theta[2]%mStep3!=0){
            theta[2] = (GLint)((GLint)theta[2]/mStep3)*mStep3;
        }
        if((GLint)theta[3]%mStep4!=0){
            theta[3] = (GLint)((GLint)theta[3]/mStep4)*mStep4;
        }
        if (que(theta[1],60) || que(theta[1],-60)){
            mStep2 = -mStep2;
        }
        if (que(theta[2],69) || que(theta[2],-69)){
            mStep3 = -mStep3;
        }
        if (que(theta[3],81) || que(theta[3],-81)){
            mStep4 = -mStep4;
        }
        if (theta[1] <= 60 && theta[1] >= -60){
```

```
        theta[1] += mStep2;
    }
    if (theta[2] <= 69 && theta[2] >= -69){
        theta[2] += mStep3;
    }
    if (theta[3] <= 81 && theta[3] >= -81){
        theta[3] += mStep4;
    }
}
else if(value == MOTION_3){
    if((GLint)theta[4]%mStep5!=0){
        theta[4] = (GLint)((GLint)theta[4]/mStep5)*mStep5;
    }
    if (que(theta[4],18) || que(theta[4],-18)){
        mStep5 = -mStep5;
    }
    if (theta[4] <= 18 && theta[4] >= -18){
        theta[4] += mStep5;
    }
}
else if(value == MOTION_4){
    theta[5] += 20;
    if(theta[5]>360){
        theta[5]=0;
    }
}
else if(value == MOTION_5){
    theta[6] += 20;
    if(theta[6]>=360){
        theta[6]=0;
    }
}
else if(value == MOTION_6){
    if((GLint)theta[7]%mStep6!=0){
        theta[7] = (GLint)((GLint)theta[7]/mStep6)*mStep6;
    }
    if (theta[7] <= 18 && theta[7] >= 0){
        theta[7] += mStep6;
    }
```

```
        if (que(theta[7],18) || que(theta[7],0)){
            mStep6 = -mStep6;
        }
    }
    else if(value == MOTION_7){
        if(UPPER_ARM_HEIGHT <= 6 && UPPER_ARM_HEIGHT >= 2){
            UPPER_ARM_HEIGHT += siz;
        }
        if (que(UPPER_ARM_HEIGHT,6) || que(UPPER_ARM_HEIGHT,2)){
            siz=-siz;
        }
    }
    glutPostRedisplay();
    glutTimerFunc(50, TimerFunction, motion);
}

/*Reshape函數*/
void myReshape(int w, int h)
{
    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    if (w <= h)
        glOrtho(-10.0, 10.0, -5.0 * (GLfloat) h / (GLfloat) w,
                15.0 * (GLfloat) h / (GLfloat) w, -10.0, 10.0);
    else
        glOrtho(-10.0 * (GLfloat) w / (GLfloat) h,
                10.0 * (GLfloat) w / (GLfloat) h, -5.0, 15.0, -10.0, 10.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}


void myinit()
{
    /*控制燈光*/
    GLfloat mat_specular[]= {1.0, 1.0, 1.0, 1.0};
    GLfloat mat_diffuse[]= {1.0, 1.0, 1.0, 1.0};
    GLfloat mat_ambient[]= {1.0, 1.0, 1.0, 1.0};
```

```
    GLfloat mat_shininess= {100.0};
    GLfloat light_ambient[]= {0.0, 0.0, 0.0, 1.0};
    GLfloat light_diffuse[]= {0.0, 1.0, 0.0, 1.0};
    GLfloat light_specular[]= {1.0, 1.0, 1.0, 1.0};
    GLfloat light_position[]= {10.0, 10.0, 10.0, 0.0};

    glLightfv(GL_LIGHT0, GL_POSITION, light_position);
    glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
    glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);

    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
    glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
    glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
    glMaterialf(GL_FRONT, GL_SHININESS, mat_shininess);

    glShadeModel(GL_SMOOTH);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    glDepthFunc(GL_LEQUAL);
    glEnable(GL_DEPTH_TEST);

    glClearColor(1.0, 1.0, 1.0, 1.0);
    glColor3f(1.0, 0.0, 0.0);

    p=gluNewQuadric();
    gluQuadricDrawStyle(p, GLU_FILL);
}

int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(500, 500);
    glutCreateWindow("robot");
    myinit();
    glutReshapeFunc(myReshape);
    glutDisplayFunc(display);
    glutMouseFunc(mouse);
```

```
    glutKeyboardFunc(keys);
    glutCreateMenu(menu);
    glutAddMenuEntry("base", 1);
    glutAddMenuEntry("lower arm", 2);
    glutAddMenuEntry("middle arm", 3);
    glutAddMenuEntry("upper arm", 4);
    glutAddMenuEntry("motion 1", 5);
    glutAddMenuEntry("motion 2", 6);
    glutAddMenuEntry("motion 3", 7);
    glutAddMenuEntry("motion 4", 8);
    glutAddMenuEntry("motion 5", 9);
    glutAddMenuEntry("motion 6", 10);
    glutAddMenuEntry("motion 7", 11);
    glutAddMenuEntry("reset", 12);
    glutAddMenuEntry("stop", 13);
    glutAddMenuEntry("quit", 14);
    glutAttachMenu(GLUT_MIDDLE_BUTTON);
    glutTimerFunc(33, TimerFunction, motion);
    glutMainLoop();
}
```