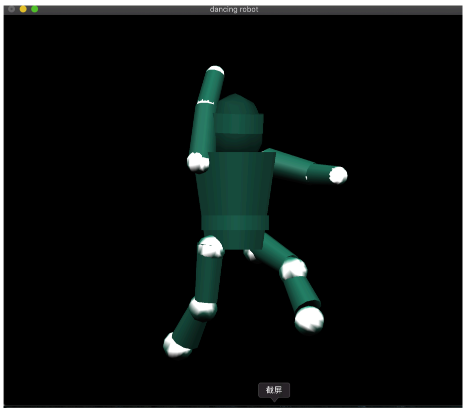
Implementation of dancing robot

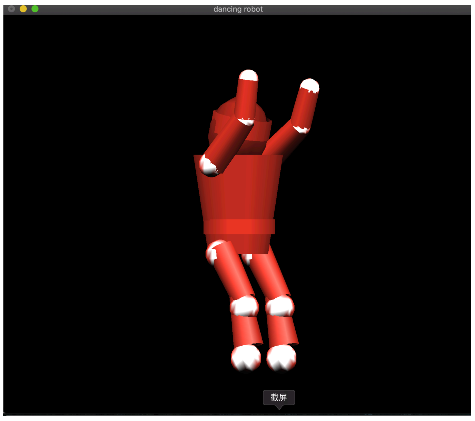
Yaohan Teng 00807306

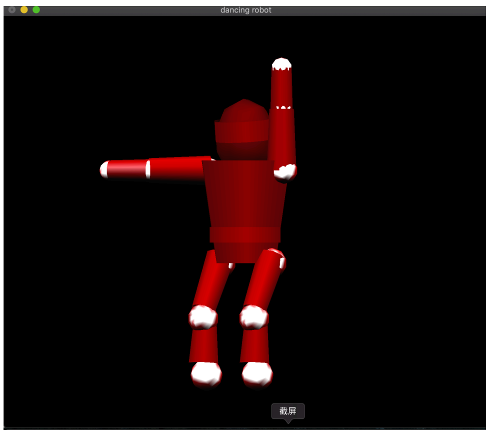
1. abstract:

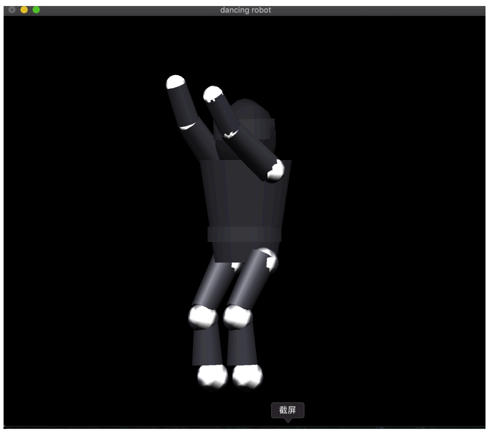
Implement a dancing robot using opengl in this project, every parts of the robot can move automatically according to the time since the program start, just like dancing. By using knowledge from the lectures, like GLUquadricObj, lighting things, glTranslate, glRotate, gluLookat …

See examples as following pictures:









2. important program part:

(1) init objects for every parts of the robot

```
GLUquadricObj*t, /// body
                 *gl, /// glass bot
                 *h,    /// head
                 *lua, /// left    upper aram
                 *lla, /// left    lower aram
                 *rua, /// right upper aram
                 *rla, /// right lower aram
                 *lul, /// left    upper leg
                 *rul, /// right upper leg
                 *lll, /// left    lower leg
                 *rll; /// right lower leg
```

(2) Build body parts:

Then by using gluCylinder() or gluSpherefunction and these objs (in (1)) to build body parts for using in display function.

(3) Combine body parts together

We can use glTranslate and glRotate functions to move these body parts of the robot into the right position to build a robot.

(4) Init lighting things

In light_init() function: Init ambient, diffuse, specular and position arrays. There are two lights in different positions, and we will random choose the arguments of these light arguments to make different color on the robot.

random choose arguments for color:

```
srand(time(NULL));
mat_specular[0] = (rand()% 10) * 0.09;
mat_specular[1] = (rand()% 10) * 0.06;
mat_specular[2] = (rand()% 10) * 0.03;
mat_ambient[0] = (rand()% 10) * 0.09;
mat_ambient[1] = (rand()% 10) * 0.06;
mat_ambient[2] = (rand()% 10) * 0.03;
mat_diffuse[0] = (rand()% 10) * 0.09;
mat_diffuse[1] = (rand()% 10) * 0.06;
mat_diffuse[2] = (rand()% 10) * 0.03;
```

(5) Arguments of rotation angles:

(i) frequency: to control the frequency of the rotation.

(ii) max & min to control the scope of rotation.

(6) Get rotation angle of each body parts:

By using following function:

```
float getAngle(float freq, float min, float max, float time){
    return (max-min) * sin(freq * PI * time) + 0.5 * (max +
min);}
```

This function takes four arguments: frep, min, max and time. The fourth argument time is the time expired the program started. So the angle can be automatically changed by using this function.

(7) Idle function:

So according to (6), we can use getAngle() function in idle(void) function, and use glutIdleFunc(idle) to call the idle() , in this way we can use the change of the time to make the robot dancing automatically.

(8) Implement multiple dancing style:

Besides changing the angle, we can also change the axis these body parts rotate for, including x, y and z axis.

We use a flag called flag_dance to control with axis to rotate for.

So we set each axis to 0 or 1, then we can have 8 dancing style in total. And we can random choose the value of flag_dance to make it do multiple dancing style automatically.

3. conclusion:

I use the knowledge learned from the lectures and assignments that I finished during this semester to finish this project. It gives me a better understanding of what I learned in this course. I think I gained a lot.