作業二：

機器人大變身

資工三乙-黃品翰 406262163

2019/12/28

● 程式架構

將機器人的身上所有部位分開來渲染，包括頭、脖子、手、腳、身體...等部位，每個部位都有各自的旋轉角度，之後將所有合併。每一次畫面都會有刷新，使用狀態機來控制每一步的動作或是變色。

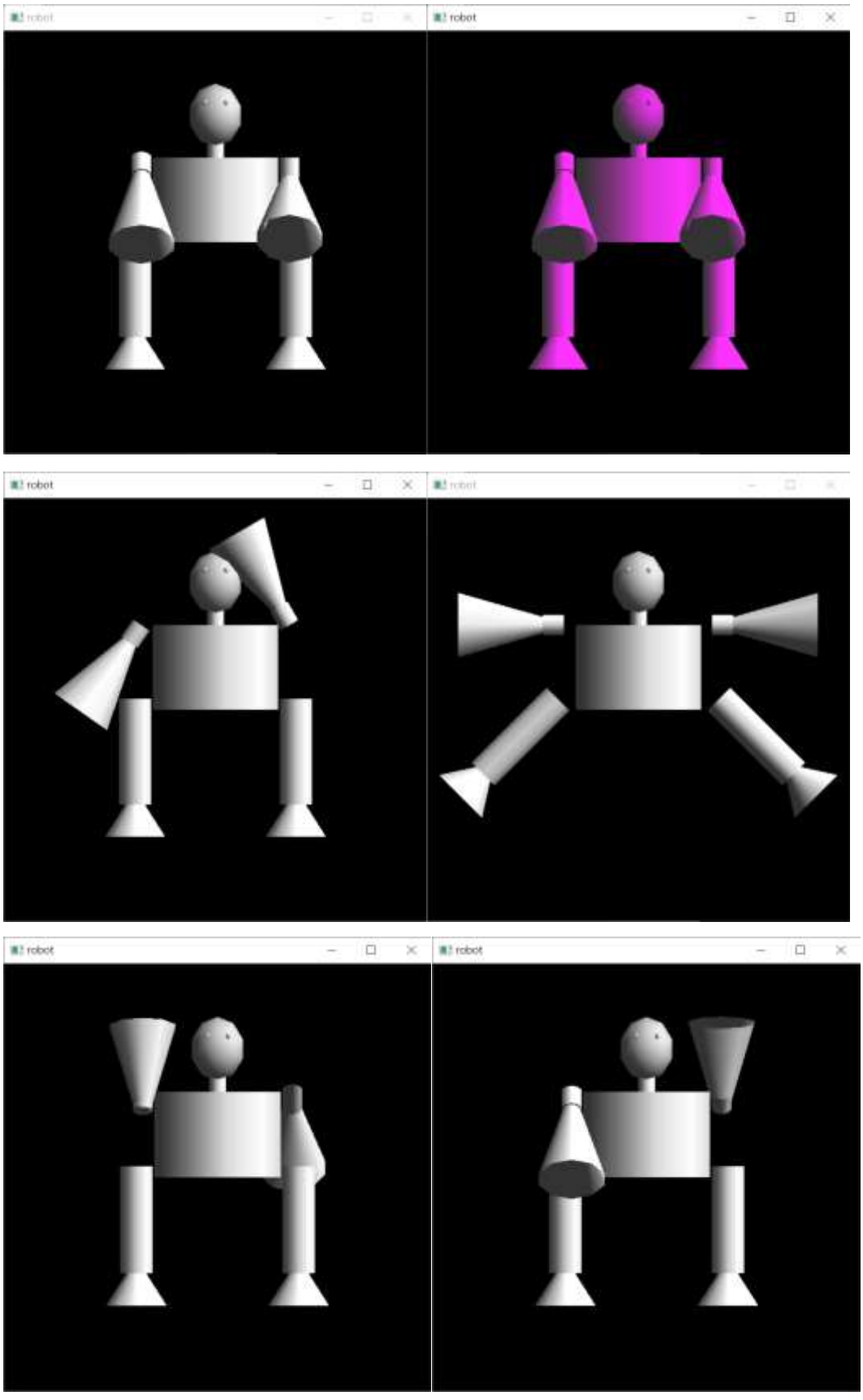
(右鍵點擊)功能則有

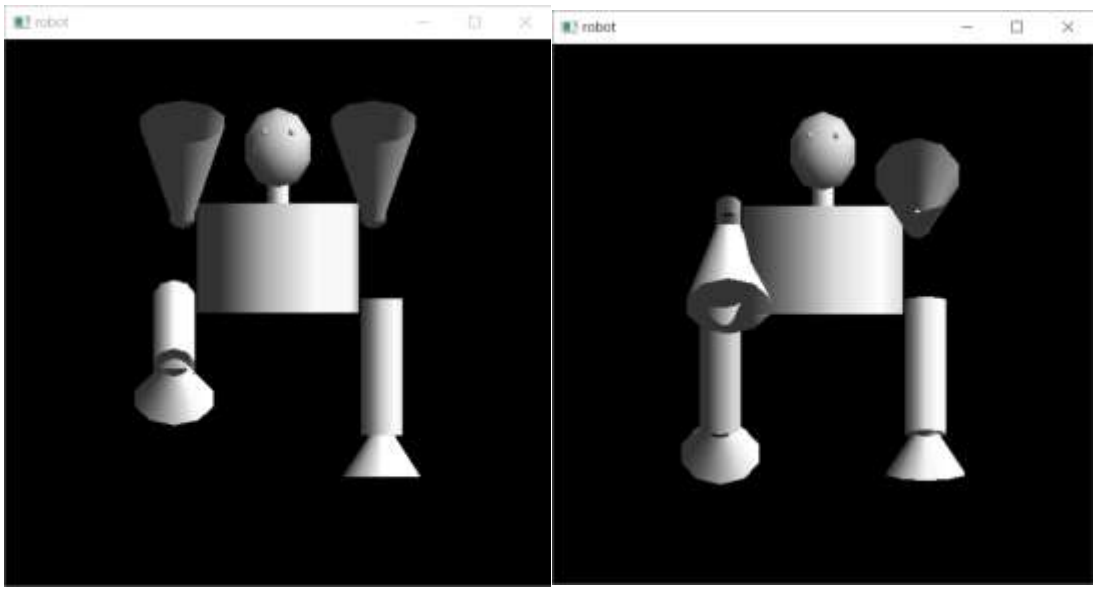
走(WALK)、跑(RUN)、開合跳(NORMAL DANCE)、舉左手(RAISE LEFT HAND)、舉右手(RAISE RIGHT HAND)、開合跳(JUMMPING JACK)、敬禮(SALUTE)、變色(Color_change ON/OFF)、離開(EXIT)

● 討論

- 每個部位的旋轉都與原本想像的不太相同，尤其是在要有動作的情況下，有時候要繞 x 軸旋轉，有時候則是繞 z 軸旋轉，看了參考資料以及網路資訊，才讓我比較習慣整個流程。身體的構造則是用圓柱體來呈現，gluCylinder 可以調整橢圓底部的大小，顏色的部分使用光照來調整機器人的顏色。

● 執行畫面





- 程式碼

```
1. #include <iostream>
2. #include <GL/glut.h>
3.
4. #define TORSO_HEIGHT 4.0
5. #define TORSO_RADIUS 3.0
6. #define UPPER_ARM_RADIUS 0.5
7. #define UPPER_ARM_HEIGHT 1.0
8. #define LOWER_ARM_RADIUS 0.8
9. #define LOWER_ARM_HEIGHT 2.0
10.#define LOWER_LEG_RADIUS 0.5
11.#define LOWER_LEG_HEIGHT 0.5
12.#define UPPER_LEG_RADIUS 0.8
13.#define UPPER_LEG_HEIGHT 5.0
14.#define HEAD_HEIGHT 3.0
15.#define HEAD_RADIUS 2.5
16.
17.typedef float point[3];
18.static GLfloat rotX, rotY, rotZ;
19.static GLfloat lfandrf = 0;
20.static GLfloat upanddown = 0;
21.static int situation = 1;
22.static int index = 9;
23.static int q = 0;
```

```
24.
25./*
26.0 身體
27.1 頭上下
28.2 頭左右
29.3 整隻右手
30.4 右手腕
31.5 整隻左手
32.6 左手腕
33.7 右腳
34.8 右小腿
35.9 左腳
36.10 左小腿
37.*/
38.
39.static GLfloat theta[11] = {
40.    45.0, 90.0, 320.0, 145.0, 0.0,
41.    180.0, -45.0, 180.0, 0.0, 180.0,
42.    0.0};
43.
44.int axis = 0;
45.GLfloat x=1,y=0,z=0;
46.
47.static GLint angle = 2;
48.double size = 1.0;
49.GLUquadricObj *t, *h, *lua, *lla, *rua, *rla, *lll, *rll, *rul, *lul;
50.
51.GLfloat colors[7][3] = {
52.    {1.0, 1.0, 1.0},
53.    {1.0, 0.0, 0.0},
54.    {0.0, 1.0, 0.0},
55.    {0.0, 0.0, 1.0},
56.    {0.0, 1.0, 1.0},
57.    {1.0, 0.0, 1.0},
58.    {1.0, 1.0, 0.0},
59.};
60.GLfloat color_r = 1;
61.GLfloat color_g = 1;
```

```
GLfloat color_b = 1;

void myinit()
{
    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_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_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(0.0, 0.0, 0.0, 1.0);
    glColor3f(0.0, 0.0, 0.0);

    h = gluNewQuadric();
    gluQuadricDrawStyle(h, GLU_FILL);
    t = gluNewQuadric();
    gluQuadricDrawStyle(t, GLU_FILL);
    lua = gluNewQuadric();
    gluQuadricDrawStyle(lua, GLU_FILL);
    lla = gluNewQuadric();
    gluQuadricDrawStyle(lla, GLU_FILL);
```

```
100.     rua = gluNewQuadric();
101.     gluQuadricDrawStyle(rua, GLU_FILL);
102.     rla = gluNewQuadric();
103.     gluQuadricDrawStyle(rla, GLU_FILL);
104.     lul = gluNewQuadric();
105.     gluQuadricDrawStyle(lul, GLU_FILL);
106.     lll = gluNewQuadric();
107.     gluQuadricDrawStyle(lll, GLU_FILL);
108.     rul = gluNewQuadric();
109.     gluQuadricDrawStyle(rul, GLU_FILL);
110.     rll = gluNewQuadric();
111.     gluQuadricDrawStyle(rll, GLU_FILL);
112. }
113.
114. void head()
115. {
116.     glPushMatrix();
117.     glTranslatef(0.0, 2 * HEAD_HEIGHT, 0.0);
118.     glScalef(0.5 * HEAD_RADIUS, 0.5 * HEAD_HEIGHT, 0.5 * HEAD_RADIUS);

119.     gluSphere(h, 1.0, 10, 10);
120.     glPushMatrix();
121.     glTranslatef(0.4, 0.4, 1.0);
122.     gluSphere(h, 0.1, 10, 3);
123.     glPopMatrix();
124.     glPushMatrix();
125.     glTranslatef(-0.4, 0.4, 1.0);
126.     gluSphere(h, 0.1, 10, 3);
127.     glPopMatrix();
128.     glPopMatrix();
129. }
130.
131. void neck()
132. {
133.     glPushMatrix();
134.     glTranslatef(0.0, 0.5, 0.0);
135.     glRotatef(90.0, 1.0, 0.0, 0.0);
136.     gluCylinder(t, 0.4, 0.4, 1.5, 10, 10);
```

```
137.     glPopMatrix();
138. }
139.
140. void torso()
141. {
142.    glPushMatrix();
143.    glRotatef(-90.0, 1.0, 0.0, 0.0);
144.    gluCylinder(t,TORSO_RADIUS, TORSO_RADIUS, TORSO_HEIGHT, 10, 10);
145.    glPopMatrix();
146. }
147.
148. void left_upper_arm()
149. {
150.     glPushMatrix();
151.     glRotatef(-90.0, 1.0, 0.0, 0.0);
152.     gluCylinder(lua, UPPER_ARM_RADIUS, UPPER_ARM_RADIUS, UPPER_ARM_HEIG
    HT, 10, 10);
153.     glPopMatrix();
154. }
155.
156. void left_lower_arm()
157. {
158.     glPushMatrix();
159.     glRotatef(-90.0, 1.0, 0.0, 0.0);
160.     gluCylinder(lla, LOWER_ARM_RADIUS*0.5, LOWER_ARM_RADIUS*2, LOWER_AR
    M_HEIGHT*2, 10, 10);
161.     glPopMatrix();
162. }
163.
164. void right_upper_arm()
165. {
166.     glPushMatrix();
167.     glRotatef(-90.0, 1.0, 0.0, 0.0);
168.     gluCylinder(rua, UPPER_ARM_RADIUS, UPPER_ARM_RADIUS, UPPER_ARM_HEIG
    HT, 10, 10);
169.     glPopMatrix();
170. }
171.
```

```
172. void right_lower_arm()
173. {
174.     glPushMatrix();
175.     glRotatef(-90.0, 1.0, 0.0, 0.0);
176.     gluCylinder(rla, LOWER_ARM_RADIUS * 0.5, LOWER_ARM_RADIUS * 2, LOWE
     R_ARM_HEIGHT*2, 10, 10);
177.     glPopMatrix();
178. }
179.
180. void left_upper_leg()
181. {
182.     glPushMatrix();
183.     glRotatef(-90.0, 1.0, 0.0, 0.0);
184.     gluCylinder(lul, UPPER_LEG_RADIUS, UPPER_LEG_RADIUS, UPPER_LEG_HEIG
     HT, 10, 10);
185.     glPopMatrix();
186. }
187.
188. void left_lower_leg()
189. {
190.     glPushMatrix();
191.     glRotatef(-90.0, 1.0, 0.0, 0.0);
192.     gluCylinder(lll, LOWER_LEG_RADIUS, LOWER_LEG_RADIUS * 3, LOWER_LEG_
     HEIGHT + 1, 10, 10);
193.     glPopMatrix();
194. }
195.
196. void right_upper_leg()
197. {
198.     glPushMatrix();
199.     glRotatef(-90.0, 1.0, 0.0, 0.0);
200.     gluCylinder(rul, UPPER_LEG_RADIUS, UPPER_LEG_RADIUS, UPPER_LEG_HEIG
     HT, 10, 10);
201.     glPopMatrix();
202. }
203.
204. void right_lower_leg()
205. {
```

```
206.     glPushMatrix();
207.     glRotatef(-90.0, 1.0, 0.0, 0.0);
208.     gluCylinder(rll, LOWER_LEG_RADIUS, LOWER_LEG_RADIUS * 3, LOWER_LEG_
    HEIGHT + 1, 10, 10);
209.     glPopMatrix();
210. }
211.
212. void display(void)
213. {
214.     glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
215.     glLoadIdentity();
216.     glColor3f(0.0, 0.0, 0.0);
217.     glTranslatef(lfandrf, upanddown, 0.0);
218.
219.     if(axis == 0)
220.     {
221.         x = 1, y = 0, z = 0;
222.     }
223.     else if(axis == 1)
224.     {
225.         x = 0, y = 0, z = 1;
226.     }
227.     //HEAD
228.     glPushMatrix();
229.     glTranslatef(0.0, TORSO_HEIGHT + 0.5 * HEAD_HEIGHT, 0.0);
230.     glRotatef(theta[1], 0.0, 0.0, 1.0);
231.     glRotatef(theta[2], 0.0, 0.0, 1.0);
232.     glTranslatef(0.0, -0.5 * HEAD_HEIGHT, 0.0);
233.     glPopMatrix();
234.     head();
235.
236.     //NECK
237.     glPopMatrix();
238.     glPushMatrix();
239.     glTranslatef(0.0, TORSO_HEIGHT + 0.5 * HEAD_HEIGHT, 0.0);
240.     glTranslatef(0.0, -0.5, 0.0);
241.     neck();
242.
```

```
243.     //TORSO
244.     glPopMatrix();
245.     glPushMatrix();
246.     glRotatef(theta[0], 0.0, 1.0, 0.0);
247.     torso();
248.
249.     //LEFT HAND
250.     glPopMatrix();
251.     glPushMatrix();
252.     glTranslatef(-
   (TORSO_RADIUS + UPPER_ARM_RADIUS), TORSO_HEIGHT, 0.0);
253.     glRotatef(theta[3], x, y, z);
254.     left_upper_arm();
255.
256.     glTranslatef(0.0, UPPER_ARM_HEIGHT, 0.0);
257.     glRotatef(theta[4], x, y, z);
258.     left_lower_arm();
259.
260.
261.     //RIGHT HAND
262.     glPopMatrix();
263.     glPushMatrix();
264.     glTranslatef(TORSO_RADIUS + UPPER_ARM_RADIUS, TORSO_HEIGHT, 0.0);
265.     glRotatef(theta[5], x, y, z);
266.     right_upper_arm();
267.
268.     glTranslatef(0, UPPER_ARM_HEIGHT, 0.0);
269.     glRotatef(theta[6], x, y, z);
270.     right_lower_arm();
271.
272.
273.     //LEFT LEG
274.     glPopMatrix();
275.     glPushMatrix();
276.     glTranslatef(-
   (TORSO_RADIUS + UPPER_LEG_RADIUS), 0.1 * UPPER_LEG_HEIGHT, 0.0);
277.     glRotatef(theta[7], x, y, z);
278.     left_upper_leg();
```

```
279.
280.     glTranslatef(0.0, UPPER_LEG_HEIGHT, 0.0);
281.     glRotatef(theta[8], x, y, z);
282.     left_lower_leg();
283.
284.     //RIGHT LEG
285.     glPopMatrix();
286.     glPushMatrix();
287.     glTranslatef(TORSO_RADIUS + UPPER_LEG_RADIUS, 0.1 * UPPER_LEG_HEIGH
    T, 0.0);
288.     glRotatef(theta[9], x, y, z);
289.     right_upper_leg();
290.
291.     glTranslatef(0.0, UPPER_LEG_HEIGHT, 0.0);
292.     glRotatef(theta[10], x, y, z);
293.     right_lower_leg();
294.
295.     glPopMatrix();
296.     glFlush();
297.     glutSwapBuffers();
298. }
299.
300. int save_color = 0;
301. int auto_change_color = 0;
302.
303.
304. void get_menu_num(int num)
305. {
306.     index = num;
307.     if(index == 8)
308.         exit(0);
309.     else if(index == 7)
310.     {
311.         auto_change_color ^= 1;
312.     }
313.     std::cout << index << '\n';
314. }
315.
```

```
316. void color_change()
317. {
318.     GLfloat light_diffuse[4];
319.     light_diffuse[0] = color_r;
320.     light_diffuse[1] = color_g;
321.     light_diffuse[2] = color_b;
322.     light_diffuse[3] = 1.0;
323.     glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
324.
325.     if (auto_change_color)
326.     {
327.         if (save_color % 7 == 0)
328.         {
329.             color_r = colors[save_color][0];
330.             color_g = colors[save_color][1];
331.             color_b = colors[save_color][2];
332.         }
333.         else if (save_color % 7 == 1)
334.         {
335.             color_r = colors[save_color][0];
336.             color_g = colors[save_color][1];
337.             color_b = colors[save_color][2];
338.         }
339.         else if (save_color % 7 == 2)
340.         {
341.             color_r = colors[save_color][0];
342.             color_g = colors[save_color][1];
343.             color_b = colors[save_color][2];
344.         }
345.         else if (save_color % 7 == 3)
346.         {
347.             color_r = colors[save_color][0];
348.             color_g = colors[save_color][1];
349.             color_b = colors[save_color][2];
350.         }
351.         else if (save_color % 7 == 4)
352.         {
353.             color_r = colors[save_color][0];
```

```
354.             color_g = colors[save_color][1];
355.             color_b = colors[save_color][2];
356.         }
357.         else if (save_color % 7 == 5)
358.         {
359.             color_r = colors[save_color][0];
360.             color_g = colors[save_color][1];
361.             color_b = colors[save_color][2];
362.         }
363.         else if (save_color % 7 == 6)
364.         {
365.             color_r = colors[save_color][0];
366.             color_g = colors[save_color][1];
367.             color_b = colors[save_color][2];
368.         }
369.
370.         save_color += 1;
371.         if (save_color >= 7)
372.             save_color = 0;
373.         std::cout << save_color << "\n";
374.     }
375.     else
376.     {
377.         color_r = colors[save_color][0];
378.         color_g = colors[save_color][1];
379.         color_b = colors[save_color][2];
380.     }
381. }
382.
383. void update(int a)
384. {
385.     color_change();
386.     switch (index)
387.     {
388.         case 0:
389.             axis = 0;
390.             if (q >= 2)
391.             {
```

```
392.                 theta[3] = 200.0;
393.                 theta[4] = -20.0;
394.                 theta[5] = 170.0;
395.                 theta[6] = -20.0;
396.                 theta[7] = 150.0;
397.                 theta[8] = 60.0;
398.                 theta[9] = 180.0;
399.                 theta[10] = 0.0;
400.             }
401.             else
402.             {
403.                 theta[3] = 170.0;
404.                 theta[4] = -20.0;
405.                 theta[5] = 200.0;
406.                 theta[6] = -20.0;
407.                 theta[7] = 180.0;
408.                 theta[8] = 0.0;
409.                 theta[9] = 150.0;
410.                 theta[10] = 60.0;
411.             }
412.             q = (q + 1) % 4;
413.             break;
414.
415.         case 1:
416.             axis = 0;
417.             if (q == 1)
418.             {
419.                 theta[3] = 230.0;
420.                 theta[4] = -90.0;
421.                 theta[5] = 150.0;
422.                 theta[6] = -90.0;
423.                 theta[7] = 200.0;
424.                 theta[8] = 30.0;
425.                 theta[9] = 170.0;
426.                 theta[10] = 20.0;
427.             }
428.             else
429.             {
```

```
430.                theta[3] = 130.0;
431.                theta[4] = -90.0;
432.                theta[5] = 200.0;
433.                theta[6] = -90.0;
434.                theta[7] = 170.0;
435.                theta[8] = 30.0;
436.                theta[9] = 210.0;
437.                theta[10] = 20.0;
438.            }
439.            q = q ? 0 : 1;
440.            break;
441.
442.        case 2:
443.            if (q % 4 == 3)
444.            {
445.                theta[0] = 45.0;
446.                theta[3] = 120.0;
447.                theta[4] = -90.0;
448.                theta[5] = 120.0;
449.                theta[6] = -90.0;
450.                theta[7] = 180.0;
451.                theta[8] = 0.0;
452.                theta[9] = 120.0;
453.                theta[10] = 100.0;
454.            }
455.            else if (q % 4 == 2)
456.            {
457.                theta[0] = 45.0;
458.                theta[3] = 180.0;
459.                theta[4] = 0.0;
460.                theta[5] = 180.0;
461.                theta[6] = 0.0;
462.                theta[7] = 180.0;
463.                theta[8] = 0.0;
464.                theta[9] = 120.0;
465.                theta[10] = 0.0;
466.            }
467.            else if (q % 4 == 1)
```

```
468.             {
469.                 theta[0] = 315.0;
470.                 theta[3] = 120.0;
471.                 theta[4] = -90.0;
472.                 theta[5] = 120.0;
473.                 theta[6] = -90.0;
474.                 theta[7] = 120.0;
475.                 theta[8] = 100.0;
476.                 theta[9] = 180.0;
477.                 theta[10] = 0.0;
478.             }
479.             else if(q % 4 == 0)
480.             {
481.                 theta[0] = 315.0;
482.                 theta[3] = 180.0;
483.                 theta[4] = 0.0;
484.                 theta[5] = 180.0;
485.                 theta[6] = 0.0;
486.                 theta[7] = 120.0;
487.                 theta[8] = 0.0;
488.                 theta[9] = 180.0;
489.                 theta[10] = 0.0;
490.             }
491.             q++;
492.             break;
493.         case 3:
494.             axis = 0;
495.             theta[0] = 45.0, theta[1] = 90.0, theta[2] = 320.0, theta[3
    ] = 145.0, theta[4] = 0.0, theta[5] = 225.0, theta[6] = 145.0, theta[7]
    = 180.0, theta[8] = 0.0, theta[9] = 180.0, theta[10] = 0.0;
496.             break;
497.         case 4:
498.             axis = 0;
499.             theta[0] = 45.0, theta[1] = 90.0, theta[2] = 320.0, theta[3
    ] = 145.0, theta[4] = 200.0, theta[5] = 225.0, theta[6] = 0.0, theta[7]
    = 180.0, theta[8] = 0.0, theta[9] = 180.0, theta[10] = 0.0;
500.             break;
501.         case 5:
```

```
502.             axis = 1;
503.             if (q % 3 == 0)
504.             {
505.                 theta[0] = 45.0, theta[1] = 90.0, theta[2] = 320.0, the
    ta[3] = 145.0, theta[4] = 0.0, theta[5] = 225.0, theta[6] = 0.0, theta[7
    ] = 180.0, theta[8] = 0.0, theta[9] = 180.0, theta[10] = 0.0;
506.             }
507.             else if (q % 3 == 1)
508.             {
509.                 theta[3] = 90;
510.                 theta[5] = 270;
511.                 theta[6] = 0;
512.                 theta[7] = 135;
513.                 theta[9] = 225;
514.             }
515.             else if (q % 3 == 2)
516.             {
517.                 theta[3] = 0;
518.                 theta[5] = 0;
519.                 theta[7] = 180;
520.                 theta[9] = 180;
521.             }
522.             q++;
523.             break;
524.         case 6:
525.             axis = 1;
526.             theta[0] = 45.0, theta[1] = 90.0, theta[2] = 320.0, theta[3
    ] = 145.0, theta[4] = 0.0, theta[5] = 30.0, theta[6] = 0.0, theta[7] = 1
    80.0, theta[8] = 0.0, theta[9] = 180.0, theta[10] = 0.0;
527.             break;
528.         case 7:
529.             break;
530.         case 8:
531.             break;
532.         case 9:
533.             break;
534.     }
535.
```

```
536.     display();
537.     glutPostRedisplay();
538.     glutTimerFunc(200, update, 0);
539. }
540.
541.
542. void myReshape(int w, int h)
543. {
544.     glViewport(0, 0, w, h);
545.     glMatrixMode(GL_PROJECTION);
546.     glLoadIdentity();
547.     if (w <= h)
548.         glOrtho(-10.0, 10.0, -10.0 * (GLfloat)h / (GLfloat)w,
549.                 10.0 * (GLfloat)h / (GLfloat)w, -10.0, 10.0);
550.     else
551.         glOrtho(-10.0 * (GLfloat)w / (GLfloat)h,
552.                 10.0 * (GLfloat)w / (GLfloat)h, 0.0, 10.0, -
   10.0, 10.0);
553.     glMatrixMode(GL_MODELVIEW);
554.     glLoadIdentity();
555. }
556.
557.
558.
559. int main(int argc, char *argv[])
560. {
561.     glutInit(&argc, argv);
562.     glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
563.     glutInitWindowSize(500, 500);
564.     glutCreateWindow("robot");
565.
566.     //menu
567.     glutCreateMenu(get_menu_num);
568.     glutAddMenuEntry("WALK", 0);
569.     glutAddMenuEntry("RUN", 1);
570.     glutAddMenuEntry("NORMAL DANCE", 2);
571.     glutAddMenuEntry("RAISE LEFT HAND", 3);
572.     glutAddMenuEntry("RAISE RIGHT HAND", 4);
```

```
573.     glutAddMenuEntry("JUMPING JACK", 5);
574.     glutAddMenuEntry("SALUTE", 6);
575.     glutAddMenuEntry("Color_change ON/OFF",7);
576.     glutAddMenuEntry("EXIT", 8);
577.     glutAttachMenu(GLUT_RIGHT_BUTTON);
578.
579.     myinit();
580.
581.     glutReshapeFunc(myReshape);
582.     glutDisplayFunc(display);
583.     glutTimerFunc(200, update, 0);
584.
585.     glutMainLoop();
586. }
```