

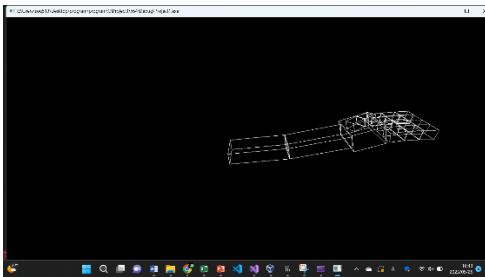
## コンピュータグラフィック前半レポート

### プログラムで工夫したところなど、考察

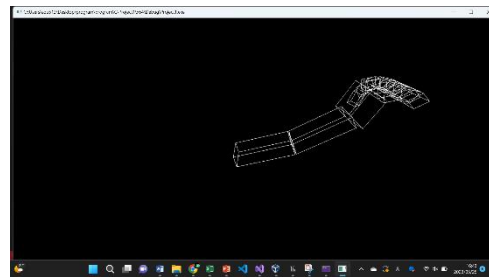
今回の課題では、ロボットアームを改造して手と指を作成した。出力としては作成した手を使ってじゃんけんの手（グー、チョキ、パー）を描画させた。工夫したところとしては2つある。1つ目は指の関節数で合計 14 もの関節を作ったところだ。指の関節を作るにあたって `glPushMatrix()` と `glPopMatrix()` をネスト構造にする必要性があり苦勞した。そして、実際の手では親指のみ他の指と付いている向きが違ふと考え、プログラム内では親指の第 1 関節までを `glRotated()` で回転させ調整した。2つ目の工夫したところは描画するところのプログラムで、`time` ライブラリをインポートしてプログラム実行後の経過時間（秒を）計測し、物体が回転して移動する時間を設定できるようにした。物体の回転では、回転する時間と  $\text{fin\_10} = (\text{fin\_10} - 3) \% 360$  の 3 の値を変更することで、時間あたりに回転する角度を増やして調整した。また、チョキの手やパーの手では指を開いて指の間に間隔を持たせたかったのが各指の付け根にのみ 2 つ目の変数を設けて動かした。

今回の課題では 2 つの問題点があると考えた。1 つ目の問題として、指の関節を 14 も作ったが、じゃんけんのような単調な動きではすべてを活かすことができなかったように感じた。なので後半のレポートではハンドシュミレータのような複雑なものを作るのもいいなと考えた。2 つ目は、動きの滑らかさである。このプログラムでは先に説明したように `time` ライブラリを使って経過時間で動きを時間指定したのだが、`time` ライブラリの時間の単位で秒が最小であるためミリ秒単位での細かい時間指定ができなかった。そこでサンプリングタイムの値を大きくして描画の更新頻度を遅くしたのだが、そのせいで動きにカクつきが出てしまった。これも第 2 回目の課題では改善したいと思う。

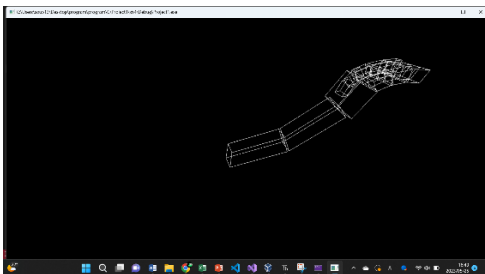
出力結果（動きがわかるように、数枚の絵を提示する）



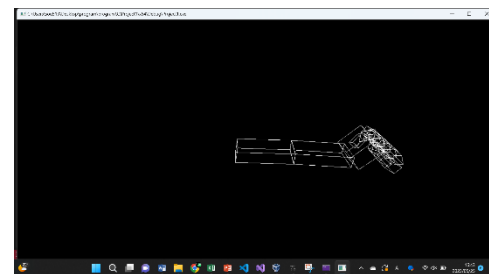
グー 1



グー 3



グー 2



グー 4

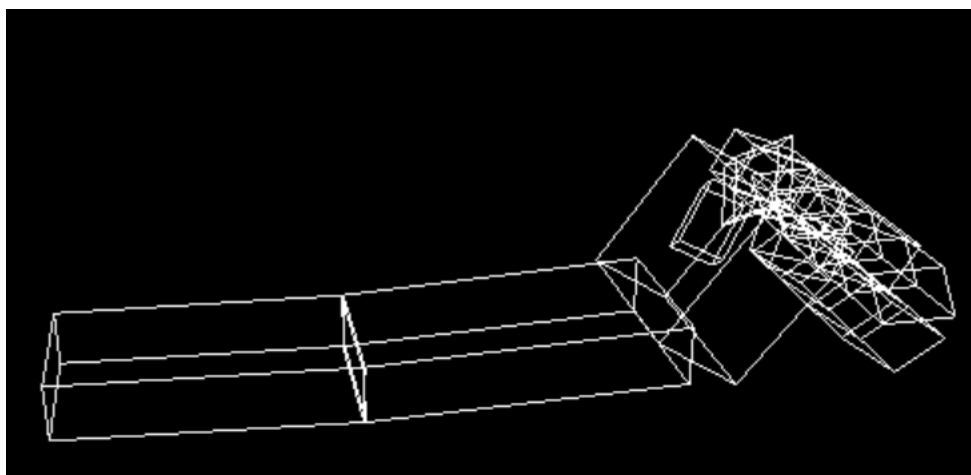
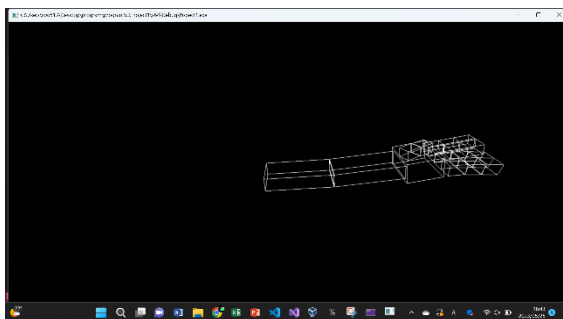
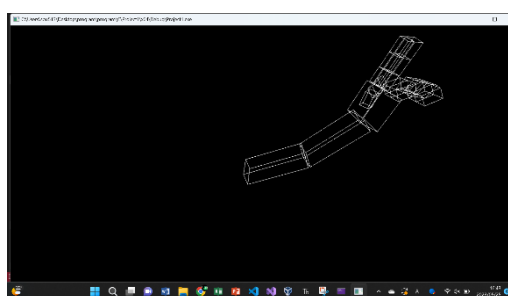


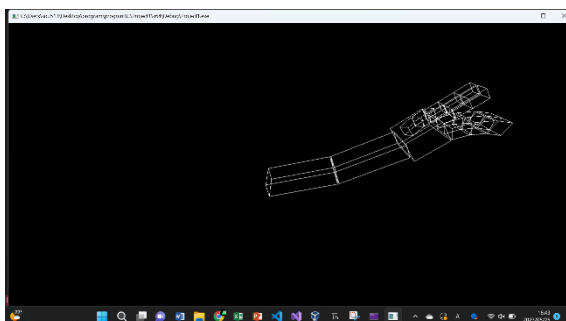
図1.グー



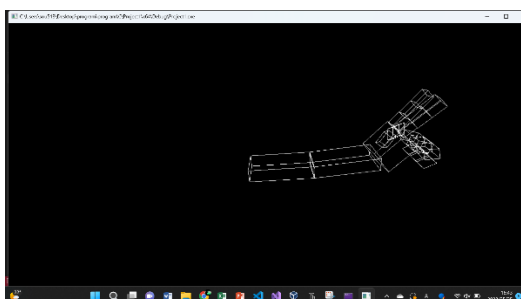
チョキ1



チョキ3



チョキ2



チョキ4

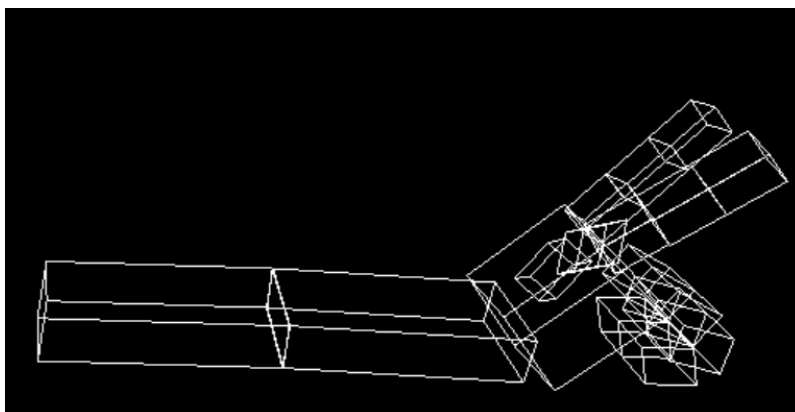
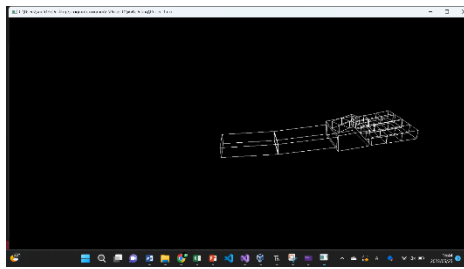
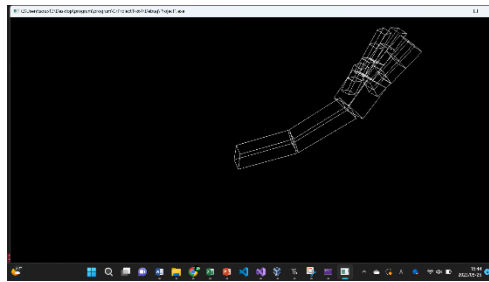


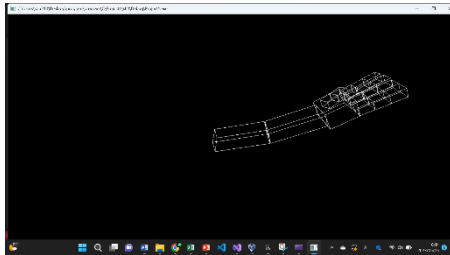
図2.チョキ



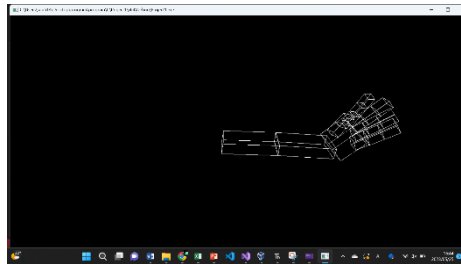
パー 1



パー 3



パー 2



パー 4

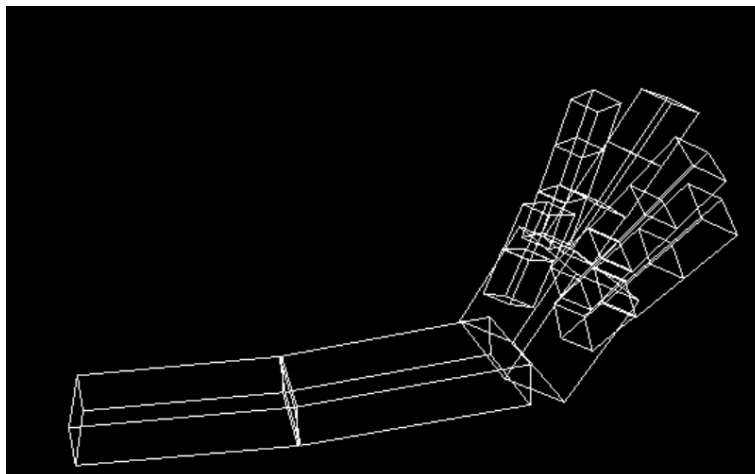


図 3.パー

#### 4. ソース

```

1  /* List: p3-robot.c
2  /* Robot arm with two links and two joints.
3  /* Examination of world-local coordinates, modeling transformation and
4  /* operation of matrix stack.
5  /*
6  #include <stdlib.h>
7  #include <GL/glut.h>
8  #include <math.h>
9  #include <time.h>
10 #include <stdio.h>
11 int    timenow=0,samplingTime = 200;
12 time_t start_time;
13 int rps_type;
14 static int shoulder = 0, elbow = 0,hand=0, fin_4 = 0, fin_5 = 0, fin_6 = 0,fin_7 = 0, fin_8 = 0, fin_9 = 0
15 , fin_10 = 0, fin_11 = 0 , fin_12 = 0 , fin_13 = 0, fin_14 = 0 , fin_15 = 0 , fin_16 = 0, fin_17 = 0,fin_4_y=0,fin_7_y=0
16 , fin_10_y = 0, fin_13_y = 0;
17
18
19 void myInit(char *programe)
20 {
21     glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA);
22     glutInitWindowSize(1600, 900);
23     glutInitWindowPosition(0, 0);
24     glutCreateWindow(programe);
25     glClearColor(0.0, 0.0, 0.0, 0.0);
26 }
27

```

```

68         glutWireCube(1.0);
69         glPushMatrix();/* 6th link *人指し指爪*/
70         glTranslated(0.5, 0.0, 1.0);
71         glRotated((double)fin_6, 0.0, 0.0, 1.0);
72         glTranslated(0.5, 0.0, -1.0);
73         glutWireCube(1.0);
74         glPopMatrix();
75     glPopMatrix();
76     glPopMatrix();
77     /* 7th link *中指第2関節まで*/
78     glPushMatrix();
79     glTranslated(0.5, 0.0, -0.5);
80     glRotated((double)fin_7, 0.0, 0.0, 1.0);
81     glRotated((double)fin_7_y, 0.0, 1.0, 0.0);
82     glTranslated(0.25, 0.0, 0.3);
83     glScaled(0.45, 0.3, 0.4);
84     glutWireCube(1.0);
85     glPushMatrix();/* 8th link *中指第1関節まで*/
86     glTranslated(0.5, 0.0, 0.5);
87     glRotated((double)fin_8, 0.0, 0.0, 1.0);
88     glTranslated(0.5, 0.0, -0.45);
89     glutWireCube(1.0);
90     glPushMatrix();/* 9th link *中指爪*/
91     glTranslated(0.5, 0.0, 1.0);
92     glRotated((double)fin_9, 0.0, 0.0, 1.0);
93     glTranslated(0.5, 0.0, -1.0);
94     glutWireCube(1.0);
95     glPopMatrix();
96     glPopMatrix();
97     glPopMatrix();
98     /* 10th link *薬指第2関節まで*/
99     glPushMatrix();
100     glTranslated(0.5, 0.0, -0.5);
101     glRotated((double)fin_10, 0.0, 0.0, 1.0);
102     glRotated((double)fin_10_y, 0.0, 1.0, 0.0);
103     glTranslated(0.25, 0.0, 0.7);
104     glScaled(0.45, 0.3, 0.4);
105     glutWireCube(1.0);
106     glPushMatrix();/* 11th link *薬指第1関節まで*/
107     glTranslated(0.5, 0.0, 0.5);
108     glRotated((double)fin_11, 0.0, 0.0, 1.0);

```

```

109     glTranslated(0.5, 0.0, -0.45);
110     glutWireCube(1.0);
111     glPushMatrix();/* 12th link *薬指爪*/
112     glTranslated(0.5, 0.0, 1.0);
113     glRotated((double)fin_12, 0.0, 0.0, 1.0);
114     glTranslated(0.5, 0.0, -1.0);
115     glutWireCube(1.0);
116     glPopMatrix();
117     glPopMatrix();
118     glPopMatrix();
119     /* 13th link *小指第2関節まで*/
120     glPushMatrix();
121     glTranslated(0.5, 0.0, -0.5);
122     glRotated((double)fin_13, 0.0, 0.0, 1.0);
123     glRotated((double)fin_13_y, 0.0, 1.0, 0.0);
124     glTranslated(0.25, 0.0, 1.1);
125     glScaled(0.45, 0.3, 0.4);
126     glutWireCube(1.0);
127     glPushMatrix();/* 14th link *小指第1関節まで*/
128     glTranslated(0.5, 0.0, 0.5);
129     glRotated((double)fin_14, 0.0, 0.0, 1.0);
130     glTranslated(0.5, 0.0, -0.45);
131     glutWireCube(1.0);
132     glPushMatrix();/* 15th link *小指爪*/
133     glTranslated(0.5, 0.0, 1.0);
134     glRotated((double)fin_15, 0.0, 0.0, 1.0);
135     glTranslated(0.5, 0.0, -1.0);
136     glutWireCube(1.0);
137     glPopMatrix();
138     glPopMatrix();
139     glPopMatrix();
140     /* 16th link *親指第一関節まで*/
141     glTranslated(0.3, 0.0, 0.1);
142     glPushMatrix();
143     glTranslated(0.5, 0.0, -0.5);
144     glRotated(20.0, 0.0, 1.0, 0.0);
145     glRotated((double)fin_16, 1.0, 0.0, 0.0);
146     glTranslated(-0.6, 0.0, -0.5);

```

```

147     glScaled(0.45, 0.3, 0.4);
148     glutWireCube(1.0);
149     glPushMatrix(); /* 17th link *親指爪*/
150     glTranslated(0.5, 0.0, 0.5);
151     glRotated((double)fin_17, 0.0, 0.0, 1.0);
152     glTranslated(0.5, 0.0, -0.45);
153     glutWireCube(1.0);
154     glPopMatrix();
155     glPopMatrix();
156     glPopMatrix();
157     glPopMatrix();
158     glutSwapBuffers();
159 }
160
161 void myReshape (int width, int height)
162 {
163     glViewport(0, 0, width, height);
164     glMatrixMode(GL_PROJECTION);
165     glLoadIdentity();
166     gluPerspective(60.0, (double)width/(double)height, 0.1, 20.0);
167     glMatrixMode(GL_MODELVIEW);
168     glLoadIdentity();
169     //glTranslated(-2.0, 3.0, -5.0);
170     gluLookAt(0.0, 4.0, 7.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0); // move to enable viewing
171 }
172
173 void myTimer(int value)
174 {
175     if (value == 1)
176     {
177         if (rps_type == 0) { //グー
178             timenow = time(NULL) - start_time;
179             glutTimerFunc(samplingTime, myTimer, 1);
180             if (timenow <= 2) {
181                 shoulder = (shoulder + 2) % 360; //肩を振り上げる
182                 elbow = (elbow + 2) % 360; //肘を振り上げる
183                 glutPostRedisplay();
184             }

```

```

185         }
186         if (timenow > 3 && timenow <= 5) {
187             shoulder = (shoulder - 2) % 360; //肩を振り下げる
188             elbow = (elbow - 2) % 360; //肘を振り下げる
189             glutPostRedisplay();
190         }
191         if (timenow <= 5) {
192             hand = (hand + 2) % 360; //手を上げる
193             fin_4 = (fin_4 - 3) % 360; //人差し指曲げる
194             fin_5 = (fin_5 - 2) % 360;
195             fin_6 = (fin_6 - 1) % 360;
196             fin_7 = (fin_7 - 3) % 360; //中指曲げる
197             fin_8 = (fin_8 - 2) % 360;
198             fin_9 = (fin_9 - 2) % 360;
199             fin_10 = (fin_10 - 3) % 360; //薬指曲げる
200             fin_11 = (fin_11 - 2) % 360;
201             fin_12 = (fin_12 - 2) % 360;
202             fin_13 = (fin_13 - 3) % 360; //小指曲げる
203             fin_14 = (fin_14 - 2) % 360;
204             fin_15 = (fin_15 - 2) % 360;
205             fin_16 = (fin_16 - 1) % 360; //親指曲げる
206             fin_17 = (fin_17 - 2) % 360;
207             glutPostRedisplay();
208         }
209     }
210 }
211 if (rps_type == 1) { //チョキ
212     timenow = time(NULL) - start_time;
213     glutTimerFunc(samplingTime, myTimer, 1);
214     if (timenow <= 2) {
215         shoulder = (shoulder + 2) % 360; //肩を振り上げる
216         elbow = (elbow + 2) % 360; //肘を振り上げる
217         glutPostRedisplay();
218     }
219     if (timenow > 3 && timenow <= 5) {
220         shoulder = (shoulder - 2) % 360; //肩を振り下げる
221         elbow = (elbow - 2) % 360; //肘を振り下げる
222     }

```

```

223     fin_4_y = (fin_4_y + 1) % 360; //人差し指の第2関節までを開く
224     fin_7_y = (fin_7_y - 1) % 360; //中指の第2関節までを開く
225     glutPostRedisplay();
226 }
227
228     if (timenow <= 5) { //kusu
229         hand = (hand + 2) % 360; //手を上げる
230         fin_10 = (fin_10 - 3) % 360; //薬指曲げる
231         fin_11 = (fin_11 - 3) % 360;
232         fin_12 = (fin_12 - 3) % 360;
233         fin_13 = (fin_13 - 3) % 360; //小指曲げる
234         fin_14 = (fin_14 - 3) % 360;
235         fin_15 = (fin_15 - 3) % 360;
236         fin_16 = (fin_16 - 1) % 360; //親指曲げる
237         fin_17 = (fin_17 - 2) % 360;
238         glutPostRedisplay();
239     }
240 }
241     if (rps_type == 2) { //パー
242         timenow = time(NULL) - start_time;
243         glutTimerFunc(samplingTime, myTimer, 1);
244         if (timenow <= 2) {
245             shoulder = (shoulder + 2) % 360; //肩を振り上げる
246             elbow = (elbow + 2) % 360; //肘を振り上げる
247             glutPostRedisplay();
248         }
249         if (timenow <= 3) {
250             fin_10_y = (fin_10_y - 1) % 360; //薬指を開く
251             glutPostRedisplay();
252         }
253         if (timenow > 3 && timenow <= 5) {
254             shoulder = (shoulder - 2) % 360; //肩を振り下げる
255             elbow = (elbow - 2) % 360; //肘を振り下げる
256             glutPostRedisplay();
257         }
258         if (timenow <= 4) {
259             fin_13_y = (fin_13_y + -1) % 360; //小指を開く
260             glutPostRedisplay();

```

```

261     }
262     if (timenow <= 5) {
263         hand = (hand + 2) % 360; //手を上げる
264         fin_4_y = (fin_4_y + 1) % 360; //人差し指を開く
265         glutPostRedisplay();
266     }
267 }
268
269     glutPostRedisplay();
270 }
271 }
272 }
273
274
275 int main(int argc, char** argv){
276     start_time = time(NULL);
277     rps_type = 2; //0の場合グー、1の場合チョキ、2の場合パー
278     glutInit(&argc, argv);
279     myInit(argv[0]);
280     glutTimerFunc(samplingTime, myTimer, 1);
281     glutReshapeFunc(myReshape);
282     glutDisplayFunc(myDisplay);
283     glutMainLoop();
284     return 0;
285 }
286

```