School of Data and Computer Science, Sun Yat-sen University

计算机图形学：作业 2

18364066 Yanzuo Lu

November 11, 2020

Contents

1 Introduction

绘制一个沿固定线路运动的机器人，要求如下：

- 线路可以是圆或任意其它闭合路径
- 机器人在任意时刻应面向运动曲线的切线方向
- 机器人应该有头、躯干、四肢等基本部分
- 机器人在运动过程中应具有摆臂及抬脚两个基本动作

2 Robot Structure

机器人构造以游戏我的世界 MineCraft 的主角史蒂夫 Steve 为原型，选取它的主要原因是在我的世界中物体都是由像素方块构成的，在 OpenGL 中能够较为简单地利用基本立方体绘制出大致相似的人物，主要参考图片如下图所示。

Figure 1: robot.png

根据上图我将人物分为 7 个部分，分别是头部 Head、肩部 Shoulder、双臂 Arm、躯干 Torso 和双腿 Leg，双臂和双腿分别可以合并，因此总共需要编写 5 个绘制函数，此外加上一个绘制基本立方体 Cube 的辅助绘制函数，上述 5 个部位都基于该基本立方体开展，划分参考图片如下图所示。

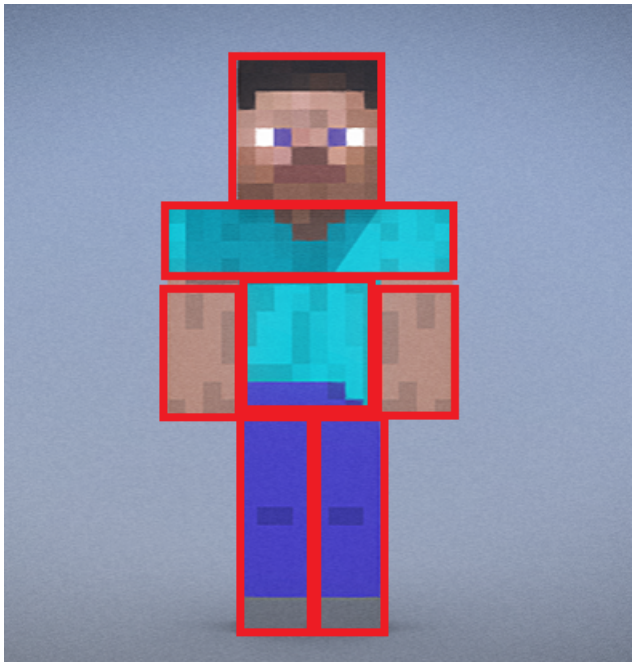

Figure 2: divide.png

3 Head, Shoulder, Torso

本质上绘制头部 Head、肩部 Shoulder 和躯干 Torso 都是类似的，因为它们都不涉及摆臂和抬腿的旋转问题，只需要对基本立方体进行缩放然后平移即可，其中头部的缩放比例在 (x,y,z) 轴上是 (20.0f, 20.0f, 20.0f), 肩部是 (40.0f, 10.0f, 10.0f), 躯干则是 (20.0f, 20.0f, 10.0f)。

绘制完这三个部位后，可以将画面绘制出来进行观察，注意要将深度测试 GL_DEPTH_TEST 打开，否则从其他角度来看十分怪异，效果如下图所示。

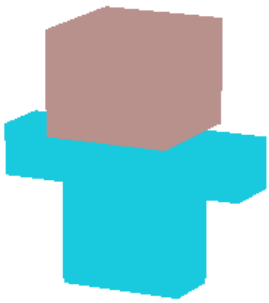

Figure 3: head, shoulder, torso.png

4 Arm, Leg

绘制双臂和双腿时，就要考虑摆臂和抬腿的问题，需要设置的一些参数包括：左臂/右臂运动，左腿/右腿运动，摆动角度增大/缩小等，考虑到 OpenGL 绘制是按帧来运行的，我们只需要在全局变量中保存好上述提到的状态参数，然后在每一帧结束的时候对状态进行更新即可，具体操作就是让左臂 + 右腿或右臂 + 左腿同时运动，每次角度增大或缩小一个较小的数值，然后在绘制时加入旋转 rotate 操作让基本立方体旋转后再进行平移。

添加了双臂和双腿，以及在每一帧结束时状态更新后，效果如下图所示。

Figure 4: arm, leg.png

5 Movement

绘制完整个机器人以后，若要使机器人在固定的闭合路线上运动，同样地只需要设置一些状态参数，保存机器人当前的位置、运动方向和朝向等即可，然后在绘制每一帧时对机器人进行旋转和平移操作使其处于正确的位置，注意到达边界时对朝向和运动方向等进行调整，同时设置好每一帧在坐标系中移动的量，让走路动作看起来更加自然，我设置的曲线是以原点为中心的一个正方形，效果如下图所示。

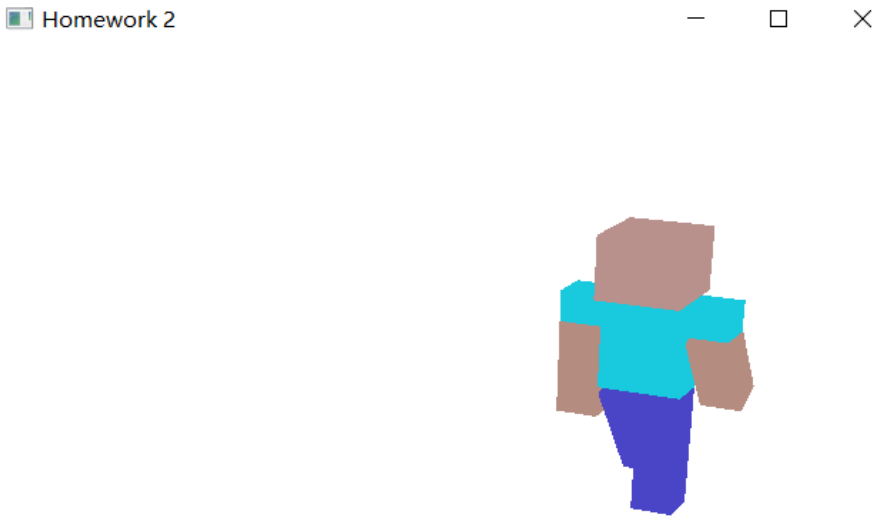

Figure 5: movement.png

6 Codes

"myglwidget.py"

```
#include "myglwidget.h"

float arm_angle = 0.0f; // 左右手臂角度
float leg_angle = 0.0f; // 左右腿角度
int arm_move = 0;       // 0表示左臂在运动，1表示右臂在运动
int leg_move = 1;       // 0表示左腿在运动，1表示右腿在运动
int arm_dire = 1;       // 0表示手臂角度在减小，1表示手臂角度在增大
int leg_dire = 1;       // 0表示腿角度在减小，1表示腿角度在增大
float max_angle = -30.0f;       // 手臂和腿摆动最大逆时针角度
float min_angle = 0.0f; // 手臂和腿摆动最小逆时针角度

float posx = 0.0f;      // 绕正方形闭合曲线运动的x坐标
float posz = 80.0f;     // 绕正方形闭合曲线运动的z坐标
int dire[] = { 1, 0 };  // 坐标在x/z轴上增加1/不变0/减小-1
float max_pos = 80.0f;  // 单一坐标轴上最大坐标值
```

```
16 float min_pos = -80.0f; // 单一坐标轴上最小坐标值
17
18 float angle_step = 1.5f;          // 每一帧摆动手脚的幅度
19 float pos_step = 0.2f;  // 每一帧位置移动的幅度
20
21
22 /*#########################################################
23 ##  函数: MyGLWidget
24 ##  函数描述:  MyGLWidget类的构造函数, 实例化定时器timer
25 ##  参数描述:
26 ##  parent: MyGLWidget的父对象
27 #########################################################*/
28 MyGLWidget::MyGLWidget(QWidget *parent)
29     :QOpenGLWidget(parent)
30 {
31     timer = new QTimer(this); // 实例化一个定时器
32     timer->start(16); // 时间间隔设置为16ms, 可以根据需要调整
33     connect(timer, SIGNAL(timeout()), this, SLOT(update())); // 连接
           update()函数, 每16ms触发一次update()函数进行重新绘图
34 }
35
36
37 /*#########################################################
38 ##  函数: ~MyGLWidget
39 ##  函数描述:  ~MyGLWidget类的析构函数, 删除timer
40 ##  参数描述:  无
41 #########################################################*/
42 MyGLWidget::~MyGLWidget()
43 {
44     delete this->timer;
45 }
46
47
```

```
48 /*#######################################################
49 ##  函数：initializeGL
50 ##  函数描述：  初始化绘图参数，如视窗大小、背景色等
51 ##  参数描述：  无
52 #######################################################*/
53 void MyGLWidget::initializeGL()
54 {
55     glViewport(0, 0, width(), height());
56     glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
57     glEnable(GL_BLEND);             // 打开混合
58     glEnable(GL_DEPTH_TEST);// 打开深度测试
59 }
60
61
62 /*#######################################################
63 ##  函数：drawCube
64 ##  函数描述：  绘制正方体
65 ##  参数描述：  无
66 #######################################################*/
67 void drawCube() {
68     glBegin(GL_TRIANGLE_STRIP);
69     glVertex3f(-0.5f, 0.5f, -0.5f);
70     glVertex3f(0.5f, 0.5f, -0.5f);
71     glVertex3f(-0.5f, -0.5f, -0.5f);
72     glVertex3f(0.5f, -0.5f, -0.5f);
73     glVertex3f(0.5f, -0.5f, 0.5f);
74     glVertex3f(0.5f, 0.5f, -0.5f);
75     glVertex3f(0.5f, 0.5f, 0.5f);
76     glVertex3f(-0.5f, 0.5f, -0.5f);
77     glVertex3f(-0.5f, 0.5f, 0.5f);
78     glVertex3f(-0.5f, -0.5f, -0.5f);
79     glVertex3f(-0.5f, -0.5f, 0.5f);
80     glVertex3f(0.5f, -0.5f, 0.5f);
```

```
81      glVertex3f(-0.5f, 0.5f, 0.5f);
82      glVertex3f(0.5f, 0.5f, 0.5f);
83      glEnd();
84  }
85
86
87  /*#######################################################
88  ##  函数: drawHead
89  ##  函数描述:  绘制机器人头部
90  ##  参数描述:  机器人头部在坐标系中的坐标(x,y,z)
91  #######################################################*/
92  void drawHead(float x, float y, float z) {
93      glPushMatrix();
94      glTranslatef(x, y, z);
95      glScalef(20.0f, 20.0f, 20.0f);
96      glColor3f(0.72f, 0.57f, 0.55f);
97      drawCube();
98      glPopMatrix();
99  }
100
101
102 /*#######################################################
103 ##  函数: drawShoulder
104 ##  函数描述:  绘制机器人肩部
105 ##  参数描述:  机器人肩部在坐标系中的坐标(x,y,z)
106 #######################################################*/
107 void drawShoulder(float x, float y, float z) {
108     glPushMatrix();
109     glTranslatef(x, y, z);
110     glScalef(40.0f, 10.0f, 10.0f);
111     glColor4f(0.10f, 0.79f, 0.87f, 1.0f);
112     drawCube();
113     glPopMatrix();
```

```
114 }
115
116
117 /*#####################################################
118 ##  函数: drawTorso
119 ##  函数描述:  绘制机器人躯干
120 ##  参数描述:  机器人躯干在坐标系中的坐标(x,y,z)
121 #####################################################*/
122 void drawTorso(float x, float y, float z) {
123     glPushMatrix();
124     glTranslatef(x, y, z);
125     glScalef(20.0f, 20.0f, 10.0f);
126     glColor4f(0.10f, 0.79f, 0.87f, 1.0f);
127     drawCube();
128     glPopMatrix();
129 }
130
131
132 /*#####################################################
133 ##  函数: drawArm
134 ##  函数描述:  绘制机器人手臂
135 ##  参数描述:  机器人手臂在坐标系中的坐标(x,y,z)和绕x轴正方向旋转的角度
136 #####################################################*/
137 void drawArm(float x, float y, float z, float rotatex) {
138     glPushMatrix();
139     glTranslatef(x, y, z);
140     glRotatef(rotatex, 1.0f, 0.0f, 0.0f);
141     glTranslatef(0.0f, -15.0f, 0.0f);
142     glScalef(10.0f, 30.0f, 10.0f);
143     glColor4f(0.71f, 0.55f, 0.50f, 1.0f);
144     drawCube();
145     glPopMatrix();
146 }
```

```
147
148
149 /*#######################################################
150 ##  函数: drawLeg
151 ##  函数描述:  绘制机器人腿部
152 ##  参数描述:  机器人腿部在坐标系中的坐标(x,y,z)和绕x轴正方向旋转的角度
153 #######################################################*/
154 void drawLeg(float x, float y, float z, float rotatex) {
155     glPushMatrix();
156     glTranslatef(x, y, z);
157     glRotatef(rotatex, 1.0f, 0.0f, 0.0f);
158     glTranslatef(0.0f, -20.0f, 0.0f);
159     glScalef(10.0f, 40.0f, 10.0f);
160     glColor4f(0.29f, 0.27f, 0.78f, 1.0f);
161     drawCube();
162     glPopMatrix();
163 }
164
165
166 /*#######################################################
167 ##  函数: paintGL
168 ##  函数描述:  绘图函数, 实现图形绘制, 会被update()函数调用
169 ##  参数描述:  无
170 #######################################################*/
171 void MyGLWidget::paintGL() {
172     // Your Implementation
173     glMatrixMode(GL_PROJECTION);
174     glLoadIdentity();
175     gluPerspective(30.0f, 1.0f, 0.1f, 1000.0f);
176     glMatrixMode(GL_MODELVIEW);
177     glLoadIdentity();
178     gluLookAt(200.0f, 200.0f, 400.0f, 0.0f, 0.0f, 0.0f, 0.0f, 1.0f, 0.0f
        );
```

```
179
180     glTranslatef(posx, 0.0f, posz);
181     if (dire[0] == 1) {
182         // 朝向x轴正向
183         glRotatef(90.0f, 0.0f, 1.0f, 0.0f);
184         posx += pos_step;
185         if (posx > max_pos) {
186             // 到达最右
187             dire[0] = 0;
188             dire[1] = -1;
189         }
190     }
191     else if (dire[0] == -1) {
192         // 朝向x轴负向
193         glRotatef(-90.0f, 0.0f, 1.0f, 0.0f);
194         posx -= pos_step;
195         if (posx < min_pos) {
196             // 到达最左
197             dire[0] = 0;
198             dire[1] = 1;
199         }
200     }
201     else if (dire[1] == 1) {
202         // 朝向z轴正向
203         posz += pos_step;
204         if (posz > max_pos) {
205             // 到达最外
206             dire[0] = 1;
207             dire[1] = 0;
208         }
209     }
210     else if (dire[1] == -1) {
211         // 朝向z轴负向
```

```
212         glRotatef(180.0f, 0.0f, 1.0f, 0.0f);
213         posz -= pos_step;
214         if (posz < min_pos) {
215             // 到达最里
216             dire[0] = -1;
217             dire[1] = 0;
218         }
219     }
220
221     drawHead(0.0f, 80.0f, 0.0f);
222     drawShoulder(0.0f, 65.0f, 0.0f);
223     drawTorso(0.0f, 50.0f, 0.0f);
224
225     if (arm_move == 0) {
226         // 左臂运动, 右臂不变
227         drawArm(15.0f, 60.0f, 0.0f, arm_angle);
228         drawArm(-15.0f, 60.0f, 0.0f, 0.0f);
229     }
230     else {
231         // 右臂运动, 左臂不变
232         drawArm(15.0f, 60.0f, 0.0f, 0.0f);
233         drawArm(-15.0f, 60.0f, 0.0f, arm_angle);
234     }
235
236     if (arm_dire == 1) {
237         // 角度增大 (对应逆时针减小)
238         arm_angle -= angle_step;
239         if (arm_angle < max_angle) {
240             // 增加角度达到最大值
241             arm_dire = 0;
242         }
243     }
244     else {
```

```
245             // 角度减小（对应逆时针增大）
246             arm_angle += angle_step;
247             if (arm_angle > min_angle) {
248                 // 减小角度达到最大值
249                 arm_dire = 1;
250                 // 换手臂
251                 arm_move = 1 - arm_move;
252             }
253         }
254 
255         if (leg_move == 0) {
256             // 左腿运动，右腿不变
257             drawLeg(5.0f, 40.0f, 0.0f, leg_angle);
258             drawLeg(-5.0f, 40.0f, 0.0f, 0.0f);
259         }
260         else {
261             // 右腿运动，左腿不变
262             drawLeg(5.0f, 40.0f, 0.0f, 0.0f);
263             drawLeg(-5.0f, 40.0f, 0.0f, leg_angle);
264         }
265 
266         if (leg_dire == 1) {
267             // 角度增大（对应逆时针减小）
268             leg_angle -= angle_step;
269             if (leg_angle < max_angle) {
270                 // 增加角度达到最大值
271                 leg_dire = 0;
272             }
273         }
274         else {
275             // 角度减小（对应逆时针增大）
276             leg_angle += angle_step;
277             if (leg_angle > min_angle) {
```

```
278                 // 减小角度达到最大值
279                 leg_dire = 1;
280                 // 换腿
281                 leg_move = 1 - leg_move;
282             }
283         }
284 }
285
286
287 /*#######################################################
288 ##  函数: resizeGL
289 ##  函数描述:  当窗口大小改变时调整视窗尺寸
290 ##  参数描述:  无
291 #######################################################*/
292 void MyGLWidget::resizeGL(int width, int height)
293 {
294     glViewport(0, 0, width, height);
295     update();
296 }
```