【3D 描画　Step2】マウスで回転できるようにする

ベクトル (a, b, c) を軸として右回りに rot 度の回転は

```
glRotatef(rot, a, b, c);
```

で実行される。従って、座標軸の回転はそれぞれ x 軸まわり,y 軸まわりの回転量(degree で表す)を rotx, roty をとすると、

x 軸まわりの回転は `glRotatef(rotx, 1.0f, 0.0f, 0.0f);`

y 軸まわりの回転は `glRotatef(roty, 0.0f, 1.0f, 0.0f);`

で実現できる。rotx, roty はパネルの端から端までドラッグしたときにちょうど一回転になるようにするためには

$$\mathrm{rotx} = \mathrm{rotx}_{\mathrm{start}} + 360 \times \frac{([\quad] - [\quad])}{[\quad]}$$

$$\mathrm{roty} = \mathrm{roty}_{\mathrm{start}} + 360 \times \frac{([\quad] - [\quad])}{[\quad]}$$

とすればよい。ここで、

$\mathrm{rotx}_{\mathrm{start}}, \mathrm{roty}_{\mathrm{start}}$：ドラッグ開始前の回転量
$x_{\mathrm{start}}, y_{\mathrm{start}}$：ドラッグ開始地点
$x_{\mathrm{end}}, y_{\mathrm{end}}$：現在（ドラッグ中）のマウスの場所
$w_{\mathrm{panel}}, h_{\mathrm{panel}}$：パネルの幅と高さ

である。

問　上の式のボックスの中を埋めなさい。

また、前回 WaterPanel をつくるときにインターフェイス MouseListener、MouseMotionListener を implementes したのは、このステップでマウスが押された時とドラッグのイベント発生時に処理を行うため。

＜作成手順＞

1．WaterPanel.java を編集（次ページ以降を参照）
　空欄部分は、上述の式を参考に埋めなさい。
2．WaterFrame.java を編集
　「元に戻す」ボタンが押された時の処理を記述する

```
resetButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent evt) {
        // 回転量が0になるようフラグをtrueにする
        centerPanel.setDefaultFlg(true);
        // 再描画
        centerPanel.repaint();
    }
});
```

3．実行して
　・マウスで回転ができること
　・「元に戻す」ボタンで元の位置にもどって描画されること
　を確認する

WaterPanel.java

```
1   package chemical.water;
2
3   import java.awt.Point;
4   import java.awt.event.MouseEvent;
5   import java.awt.event.MouseListener;
6   import java.awt.event.MouseMotionListener;
7
8   import javax.media.opengl.GL;
9   import javax.media.opengl.GLAutoDrawable;
10  import javax.media.opengl.GLEventListener;
11  import javax.media.opengl.GLJPanel;
12  import javax.media.opengl.glu.GLU;
13
14  import com.sun.opengl.util.GLUT;
15
16  public class WaterPanel extends GLJPanel implements GLEventListener,
17          MouseListener, MouseMotionListener {
18
19      private GL gl;            // OpenGL の関数群をもつオブジェクトその１
20      private GLU glu;      // OpenGL の関数群をもつオブジェクトその２
21      private GLUT glut;        // OpenGL の関数群をもつオブジェクトその３
22      private int div = 20;     // 球、円柱の分割数
23
24      // 軸と O-H のなす角 (※水の O-H がなす角は 104.45 度)
25      private float degree = (180-104.45f)/2.0f;
26
27      private float rotx = 0.0f, roty = 0.0f;    // 回転量
28      private float sRotx, sRoty;                // ドラッグ開始時の回転量
29      private Point startPoint, endPoint;        // ドラッグの開始点と終了点
30      private boolean defaultFlg = true;         // true だったら回転量を 0 にする
31
32      /* 光 ---------------------------------------------------*/
33      // 光の位置 {x 座標, y 座標, z 座標, 光源までの距離(0 は無限円)}
34      private float[] lPosition = { -10.0f, 10.0f, 10.0f, 0.0f };
35      // 光の色 {R, G, B, アルファ(透明度)}  ※数値は 0 から 1 まで
36      private float[] lSpecular = { 0.8f, 0.8f, 0.8f, 1.0f };    // 鏡面
37      private float[] lDiffuse = { 0.8f, 0.8f, 0.8f, 1.0f };     // 拡散
38      private float[] lAmbient = { 0.4f, 0.4f, 0.4f, 1.0f };     // 環境
39
40      /* 物体の反射率 -----------------------------------------*/
41      // {R, G, B, アルファ(透明度)}  ※数値は 0 から 1 まで
42      private float[] mSpecular = { 0.3f, 0.3f, 0.3f, 1.0f };    // 鏡面
43      private float[] mDiffuse = { 0.2f, 0.2f, 0.2f, 1.0f };     // 拡散
44      // 鏡面係数：きらめきの度合い (0～128)
45      private float mShininess = 10.0f;
46
47      /* 色の定義 ---------------------------------------------*/
48      // {R, G, B, アルファ(透明度)}  ※数値は 0 から 1 まで
49      private float[] blue = { 0.0f, 0.0f, 1.0f, 1.0f };     // 青
50      private float[] red = { 1.0f, 0.0f, 0.0f, 1.0f };      // 赤
51      private float[] green = {0.0f, 1.0f, 0.5f, 1.0f };     // 緑
52
53      /**
54       * コンストラクタ
55       */
56      public WaterPanel() {
57          // MouseEvent を受け取れるようにする
58          addMouseListener(this);
59          // MouseMotionEvent を受け取れるようにする
60          addMouseMotionListener(this);
61          // 3D 描画できるようにリスナ登録
62          addGLEventListener(this);
63      }
64
```

```
65      /* 3D 描画 ************************************************************/
66
67      /**
68       * x 軸周りに円柱を描画
69       * @param r 円柱の半径
70       * @param length 円柱の長さ
71       * @param offset offset ≦x≦offset+length に描かれる
72       */
73      public void drawCylinder(float r, float length, float offset) {
74          double t;
75          // 要素として連続する四角形を使う
76          gl.glBegin(GL.GL_QUAD_STRIP);
77          for (int i=0; i<=div; i++) {
78              // 角度を計算（ラジアン）
79              t = 2.0 * Math.PI * i / div;
80              // 光の反射の法線ベクトル
81              gl.glNormal3f(0, (float)Math.cos(t), (float)Math.sin(t));
82              // 頂点を設定
83              gl.glVertex3f(offset, (float)(r*Math.cos(t)), (float)(r*Math.sin(t)));
84              gl.glVertex3f(offset+length, (float)(r*Math.cos(t)), (float)(r*Math.sin(t)));
85          }
86          gl.glEnd();
87      }
88
89      /**
90       * GLEventListener のメソッド：3D 描画メソッド
91       */
92      @Override
93      public void display(GLAutoDrawable drawable) {
94          /* deaultFlg=true なら最初かリセットボタンが押された状態
95           * なので、回転量を 0 にするｓ */
96          if (defaultFlg) {
97              rotx = 0.0f;
98              roty = 0.0f;
99              defaultFlg = false;
100         }
101
102         // 背景色で塗りつぶし
103         gl.glClear(GL.GL_COLOR_BUFFER_BIT | GL.GL_DEPTH_BUFFER_BIT);
104         // 現時点のマトリクス（Modelview 行列）を保存　※必ず glPopMatrix()と対で使用する
105         /* glPushMatrix と glPopMatrix()は、座標を回転したり平行移動したり
106          * したものを、する前の状態を覚えておくために使う。
107          * これを使わないと、回転などがどんどん重なっていってしまう。 */
108         gl.glPushMatrix();
109         // ｚ軸上 15 の位置から原点を見る,上方向はｙ軸
110         glu.gluLookAt(0.0, 0.0, 15.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0);
111         // マウスの移動量に応じて回転
112         gl.glRotatef(rotx, 1.0f, 0.0f, 0.0f);
113         gl.glRotatef(roty, 0.0f, 1.0f, 0.0f);
114         // y 軸正方向に少しずらす
115         gl.glTranslatef(0.0f, 0.4f, 0.0f);
116
117         /* 棒 1 を描画(z 軸周りに-degree 度回転) ----------*/
118         // ModelView 行列保存
119         gl.glPushMatrix();
120         // 色指定
121         // ※環境光の反射率とは、つまり物体の色になる
122         gl.glMaterialfv(GL.GL_FRONT, GL.GL_AMBIENT, green, 0);
123         // 座標軸を z 軸周りに-degree 度回転
124         // ※2, 3, 4 番目の引数は回転軸を表す（順に x, y, z）
125         gl.glRotatef(-degree, 0.0f, 0.0f, 1.0f);
126         // 半径 0.07 の円柱を 0≦x≦1 に描画
127         drawCylinder(0.07f, 1.0f, 0.0f);
128         // 行列を保存した時の状態に戻す
```

```
129             gl.glPopMatrix();
130
131             /* 棒 2 を描画(z 軸周りに degree 度回転) ----------*/
132             gl.glPushMatrix();
133             gl.glMaterialfv(GL.GL_FRONT, GL.GL_AMBIENT, green, 0);
134             gl.glRotatef(degree, 0.0f, 0.0f, 1.0f);
135             drawCylinder(0.07f, 1.0f, -1.0f); // -1≦x≦0
136             gl.glPopMatrix();
137
138             /* O 原子を原点に描画 -------------------------*/
139             gl.glMaterialfv(GL.GL_FRONT, GL.GL_AMBIENT, red, 0);
140             // 塗りつぶしの球を描画
141             // glutSolidSphere(半径, Z 軸まわりの分割数, Z 軸に沿った分割数)
142             glut.glutSolidSphere(0.3, div, div);
143
144             // H 原子 1 を描画 ----------------------------*/
145             gl.glPushMatrix();
146             gl.glMaterialfv(GL.GL_FRONT, GL.GL_AMBIENT, blue, 0);
147             gl.glRotatef(-degree, 0.0f, 0.0f, 1.0f);
148             gl.glTranslatef(1.0f, 0.0f, 0.0f);    // x 方向に 1 だけ平行移動
149             glut.glutSolidSphere(0.2, div, div);
150             gl.glPopMatrix();
151
152             // H 原子 2 を描画 ----------------------------*/
153             gl.glPushMatrix();
154             gl.glMaterialfv(GL.GL_FRONT, GL.GL_AMBIENT, blue, 0);
155             gl.glRotatef(degree, 0.0f, 0.0f, 1.0f);
156             gl.glTranslatef(-1.0f, 0.0f, 0.0f);   // x 方向に-1 だけ平行移動
157             glut.glutSolidSphere(0.2, div, div);
158             gl.glPopMatrix();
159
160             // glPushMatrix()を呼ぶ前の行列の状態に戻す
161             gl.glPopMatrix();
162         }
163
164         /**
165          * GLEventListener のメソッド：表示の切り替えが発生した場合に呼ばれる
166          * 今回は特に記述することはない
167          */
168         @Override
169         public void displayChanged(GLAutoDrawable drawable, boolean modeChanged,
170                 boolean deviceChanged) {
171         }
172
173         /**
174          * GLEventListener のメソッド：初期化時に呼ばれる
175          */
176         @Override
177         public void init(GLAutoDrawable drawable) {
178             // OpenGL の関数群をもつオブジェクトを取得
179             gl = drawable.getGL();
180             glu = new GLU();
181             glut = new GLUT();
182             // 背景色を黒に設定
183             gl.glClearColor(0.0f, 0.0f, 0.0f, 0.0f);
184             // 影に隠れて見えないものは表示しない
185             gl.glEnable(GL.GL_DEPTH_TEST);
186             // 光を有効にする
187             gl.glEnable(GL.GL_LIGHTING);
188             gl.glEnable(GL.GL_LIGHT0);
189             // 光の反射の法線ベクトルを正規化
190             gl.glEnable(GL.GL_NORMALIZE);
191             // 光の位置を設定
192             /* 4 番目の数字はオフセット。
```

```
193             * JOGL は c++のソースを JAVA に自動変換している。
194             * c++では配列はポインタなので、オフセットの概念がある
195             * JAVA にポインタはないので引数が増やされているのだが、
196             * オフセットの数字は 0 でよい。 */
197             gl.glLightfv(GL.GL_LIGHT0, GL.GL_POSITION, lPosition, 0);
198             // 光の色を設定
199             gl.glLightfv(GL.GL_LIGHT0, GL.GL_SPECULAR, lSpecular, 0);  // 鏡面
200             gl.glLightfv(GL.GL_LIGHT0, GL.GL_DIFFUSE, lDiffuse, 0);        // 拡散
201             gl.glLightfv(GL.GL_LIGHT0, GL.GL_AMBIENT, lAmbient, 0);        // 環境
202             // 物体の反射率を設定
203             gl.glMaterialfv(GL.GL_FRONT, GL.GL_SPECULAR, mSpecular, 0);// 鏡面
204             gl.glMaterialfv(GL.GL_FRONT, GL.GL_DIFFUSE, mDiffuse, 0);  // 拡散
205             gl.glMaterialf(GL.GL_FRONT, GL.GL_SHININESS, mShininess);  // 鏡面係数
206         }
207
208         /**
209          * GLEventListener のメソッド：表示領域が変更されたときに呼ばれる
210          */
211         @Override
212         public void reshape(GLAutoDrawable drawable, int x, int y, int width,
213                 int height) {
214             // ビューポート(描画領域)を設定
215             gl.glViewport(0, 0, width, height);
216             // マトリクスとして Projection 行列を対象にする
217             gl.glMatrixMode(GL.GL_PROJECTION);
218             // マトリクスを単位行列で初期化
219             gl.glLoadIdentity();
220             // 視点の設定 (参考資料を参照)
221             // gluPerspactive(視野角, 縦横比, near, far);
222             glu.gluPerspective(10.0, (double)width/(double)height, 1.0, 100.0);
223             // マトリクスとして ModelView 行列を対象にする
224             gl.glMatrixMode(GL.GL_MODELVIEW);
225         }
226
227         /* MouseListener のメソッド ************************************************/
228
229         @Override
230         public void mouseClicked(MouseEvent e) {
231             // TODO Auto-generated method stub
232
233         }
234
235         @Override
236         public void mouseEntered(MouseEvent e) {
237             // TODO Auto-generated method stub
238
239         }
240
241         @Override
242         public void mouseExited(MouseEvent e) {
243             // TODO Auto-generated method stub
244
245         }
246
247         /**
248          * マウスを押したらスタートポイントに設定
249          */
250         @Override
251         public void mousePressed(MouseEvent e) {
252             // マウスを押した場所を取得
253             startPoint = e.getPoint();
254             // 現時点での回転量を保存
255             sRotx = rotx;
256             sRoty = roty;
```

```
257         }
258
259         @Override
260         public void mouseReleased(MouseEvent e) {
261             // TODO Auto-generated method stub
262
263         }
264
265         /* MouseMotionListener のメソッド ************************************************/
266
267         /**
268          * ドラッグで回転量を計算
269          */
270         @Override
271         public void mouseDragged(MouseEvent e) {
272             // マウスの場所を取得
273             endPoint = e.getPoint();
274             // 回転量を計算(パネルの端から端で一回転)
275             rotx =
276             roty =
277             // 再描画
278             repaint();
279         }
280
281         @Override
282         public void mouseMoved(MouseEvent e) {
283             // TODO Auto-generated method stub
284
285         }
286
287         /* defaultFlg の setter ********************************************************/
288         public void setDefaultFlg(boolean defaultFlg) {
289             this.defaultFlg = defaultFlg;
290         }
291
292
293     }
```