TD 1 - Découvrir OpenGL - Notes & Solutions

Xavier Décoret

1 Notes diverses

- `glClearColor()` peut se mettre dans `initializeGL()` car OpenGL est une machine à état
- Attention au division entière en C ! Si vous calculez le ratio de la fenêtre avec `width()/height()`, vous aurez des résultats "quantiques" (avec des sauts !) car le / est une division entière. Pour forcer la division réelle, utilisez une des formes suivantes :

```
1 float ratio = float(width())/height();
2 float ratio = width()*1.0f/height();
3 float ratio =
4 static_cast<float>(width())/static_cast<float>(height()); // Best!
```

- Attention à avoir autant de `glPushAttrib()` que de *glPopMatrix()*
- Par défaut, une fenêtre ne se réaffiche (appelle `paintGL()` que quand son contenu est invalidé, ce qui arrive si la fenêtre est redimensionné ou si elle est couverte (par une autre fenêtre) puis découverte. Pour forcer le rafraîchissemen, utiliser la méthode `updateGL()` de `QGLWidget`. Pensez-y notamment dans les réponses aux évènements *keyPressEvent()*.
- La méthode *resizeGL(int,int)* est appelé une fois après le *initializeGL()* est puis chaque fois que la fenêtre est redimensionné. On y met typiquement les fonctions qui changent le viewport (plutôt que d'appeller la fonction *glViewport()* à chaque *paintGL()*.
- L'orientation d'une face (qui influencera par exemple l'interprétation de `GL_FRONT`, `GL_BACK`, `GL_FRONT_AND_BACK` comme premier paramètre de `glPolygonMode()`) est déterminé par l'ordre d'envoi des sommets (l'ordre des appels `glVertex()`). Pensez à être cohérent (dans le cours, l'IFS du cube par exemple n'a pas une bonne orientation de ses faces !).
- Pour dessiner des cubes, sphères, cylindre, utilisez les fonctions `glut(Solid Wire)(Cube Sphere Cylinder)` du *GL Utility Toolkit* (GLUT). Cela nécessite d'inclure `#include <GL/glut.h>`, d'appeller `glutInit(&argc,argv) ;` dans la fonction *main()*, et de rajouter `LIBS *= -lglut` dans le `.pro`.

Solution de la section 3

```
#include <QtOpenGL>

class Viewer : public QGLWidget
{
public:
  Viewer()
    : QGLWidget()
    {
    }
protected:
  virtual void initializeGL()
    {
      glClearColor(0.2f,0.2f,0.2f,0.0f);
    }
  virtual void paintGL()
    {
      glPushAttrib(GL_ALL_ATTRIB_BITS);
      glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

      glMatrixMode(GL_PROJECTION);
      glLoadIdentity();
      gluPerspective(40.0f,float(width())/height(),0.1f,10.0f);
      glMatrixMode(GL_MODELVIEW);
      glLoadIdentity();
      glTranslatef(0.0f,0.0f,-4.0f);

      for (int n=0;n<4;++n)
      {
        int w = width()/2;
        int h = height()/2;
        glViewport((n%2)*w,(n/2)*h,w,h);
        glBegin(GL_QUADS);
        glVertex3f(-0.5f,-0.5f, 0.0f);
        glVertex3f( 0.5f,-0.5f, 0.0f);
        glVertex3f( 0.5f, 0.5f, 0.0f);
        glVertex3f(-0.5f, 0.5f, 0.0f);
        glEnd();
      }
      glPopAttrib();
    }
  virtual void keyPressEvent(QKeyEvent* e)
    {
      switch (e->key())
      {
      case Qt::Key_Escape:
        close();
        break;
      }
    }
};

int main(int argc,char** argv)
{
```

```
54    QApplication application(argc,argv);
55
56    Viewer v;
57    v.show();
58
59    return application.exec();
60  }
```

Solution de la section 4

```
1   #include <QtOpenGL>
2   #include <cmath>
3
4   using namespace std;
5
6   class Viewer : public QGLWidget
7   {
8   public:
9     Viewer()
10      : QGLWidget()
11      , _angle(0.0f)
12      , _wireframe(true)
13      {
14      }
15  protected:
16    virtual void initializeGL()
17      {
18        glClearColor(0.2f,0.2f,0.2f,0.0f);
19      }
20    virtual void resizeGL(int w,int h)
21      {
22        glViewport(0,0,w,h);
23      }
24    virtual void paintGL()
25      {
26        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
27
28        glMatrixMode(GL_PROJECTION);
29        glLoadIdentity();
30        gluPerspective(40.0f,float(width())/height(),0.1f,10.0f);
31        glMatrixMode(GL_MODELVIEW);
32        glLoadIdentity();
33        const float radius = 4.0f;
34        float x = radius*cos(_angle*180.0f/M_PI);
35        float y = radius*sin(_angle*180.0f/M_PI);
36        float z = 1.0f;
37        gluLookAt(x,y,z,0.0f,0.0f,0.0f,0.0f,0.0f,1.0f);
38
39        glPolygonMode(GL_FRONT_AND_BACK, _wireframe?GL_LINE:GL_FILL);
40        float vertices[8][3] =
41        {
42          { -0.5f,-0.5f,-0.5f },
```

```
43          { +0.5f,-0.5f,-0.5f },
44          { -0.5f,+0.5f,-0.5f },
45          { +0.5f,+0.5f,-0.5f },
46          { -0.5f,-0.5f,+0.5f },
47          { +0.5f,-0.5f,+0.5f },
48          { -0.5f,+0.5f,+0.5f },
49          { +0.5f,+0.5f,+0.5f },
50        };
51        int faces[6][4] =
52        {
53          { 0,4,6,2 }, // Face pointing -x
54          { 1,3,7,5 }, // Face pointing +x
55          { 0,1,5,4 }, // Face pointing -y
56          { 3,2,6,7 }, // Face pointing +y
57          { 0,2,3,1 }, // Face pointing -z
58          { 4,5,7,6 }, // Face pointing +z
59        };
60        glBegin(GL_QUADS);
61        for (int i=0;i<6;++i)
62        {
63          glVertex3fv(vertices[faces[i][0]]);
64          glVertex3fv(vertices[faces[i][1]]);
65          glVertex3fv(vertices[faces[i][2]]);
66          glVertex3fv(vertices[faces[i][3]]);
67        }
68        glEnd();
69      }
70    virtual void keyPressEvent(QKeyEvent* e)
71      {
72        switch (e->key())
73        {
74        case Qt::Key_Escape:
75          close();
76          break;
77        case Qt::Key_W:
78          _wireframe = !_wireframe;
79          updateGL();
80          break;
81        case Qt::Key_Left:
82          _angle += 2.0f;
83          updateGL();
84          break;
85        case Qt::Key_Right:
86          _angle -= 2.0f;
87          updateGL();
88          break;
89        }
90      }
91  private:
92    float _angle;
93    bool  _wireframe;
94  };
95
96  int main(int argc,char** argv)
97  {
98    QApplication application(argc,argv);
99
```

```
100     Viewer v;
101     v.show();
102
103     return application.exec();
104   }
```

Solution de la section 5

```
1   #include <QtOpenGL>
2   #include <GL/glut.h>
3   #include <cmath>
4
5   using namespace std;
6
7   namespace
8   {
9     /*!
10     * Renders an uncapped (no top and bottom disks) cylinder of height 1 and
11     * diameter 1, with axis z and basis at origin (not centered on origin).
12     *
13     * \p tesselation controls the number of facets used to approximate
14     * cylinder by a polyhedron.
15     */
16    void cylinder(int tesselation)
17    {
18      float h = 1.0f;
19      float r = 0.5f;
20      glBegin(GL_QUAD_STRIP);
21      for (int i=0;i<tesselation+1;++i) // use +1 to loop back to first points
22      {
23        float a = i*2.0f*M_PI/tesselation;
24        float x = r*cos(a);
25        float y = r*sin(a);
26        glVertex3f(x,y,0.0f);
27        glVertex3f(x,y,h);
28      }
29      glEnd();
30    }
31  };
32
33  class Viewer : public QGLWidget
34  {
35  public:
36    Viewer()
37      : QGLWidget()
38      , _angle(0.0f)
39      , _wireframe(true)
40      {
41      }
42  protected:
43    virtual void initializeGL()
44      {
```

```
45          glClearColor(0.2f,0.2f,0.2f,0.0f);
46        }
47      virtual void resizeGL(int w,int h)
48        {
49          glViewport(0,0,w,h);
50        }
51      virtual void paintGL()
52        {
53          glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
54
55          glMatrixMode(GL_PROJECTION);
56          glLoadIdentity();
57          gluPerspective(40.0f,float(width())/height(),0.1f,40.0f);
58          glMatrixMode(GL_MODELVIEW);
59          glLoadIdentity();
60          const float radius = 10.0f;
61          float x = radius*cos(_angle*180.0f/M_PI);
62          float y = radius*sin(_angle*180.0f/M_PI);
63          float z = 10.0f;
64          gluLookAt(x,y,z,0.0f,0.0f,0.0f,0.0f,0.0f,1.0f);
65
66          glPolygonMode(GL_FRONT_AND_BACK, _wireframe?GL_LINE:GL_FILL);
67          glPushMatrix();
68          glScalef(0.2f,0.4f,4.0f);
69          cylinder(10);
70          glPopMatrix();
71          glPushMatrix();
72          glTranslatef(0.0f,0.0f,4.0f);
73          glutSolidSphere(0.5f,10,10);
74          glPopMatrix();
75          glPushMatrix();
76          glTranslatef(0.0f,-0.5f,-5.0f);
77          glScalef(0.2f,0.2f,5.0f);
78          cylinder(10);
79          glPopMatrix();
80          glPushMatrix();
81          glTranslatef(0.0f,+0.5f,-5.0f);
82          glScalef(0.2f,0.2f,5.0f);
83          cylinder(10);
84          glPopMatrix();
85          glPushMatrix();
86          glTranslatef(0.0f,0.0f,3.5f);
87          glScalef(0.2f,3.0f,0.2f);
88          glRotatef(90.0f,1.0f,0.0f,0.0f);
89          glTranslatef(0.0f,0.0f,-0.5f);
90          cylinder(10);
91          glPopMatrix();
92
93        }
94      virtual void keyPressEvent(QKeyEvent* e)
95        {
96          switch (e->key())
97          {
98          case Qt::Key_Escape:
99            close();
100           break;
101         case Qt::Key_W:
```

```
102           _wireframe = !_wireframe;
103           updateGL();
104           break;
105         case Qt::Key_Left:
106           _angle += 2.0f;
107           updateGL();
108           break;
109         case Qt::Key_Right:
110           _angle -= 2.0f;
111           updateGL();
112           break;
113         }
114     }
115 private:
116   float _angle;
117   bool  _wireframe;
118 };
119
120 int main(int argc,char** argv)
121 {
122   QApplication application(argc,argv);
123   glutInit(&argc,argv);
124
125   Viewer v;
126   v.show();
127
128   return application.exec();
129 }
```