Modelització i Visualització

Exercici 3. Simulant l'especificació bàsica d'OpenGL

David Domènech Vilà
2018-2019

Introducció

En aquesta pràctica ens demanen simular l'especificació bàsica en OpenGl per poder pintar figures geomètriques.

Respostes

-Identifica quines matrius has de guardar i les operacions que has de poder fer sobre elles. Planteja't la necessitat de tenir una classe auxiliar per a gestionar aquesta informació.

Has de guardar diferents matrius per poder realitzar les operacions, primer guardem les matrius GL_MODELVIEW i GL_PROJECTION, en una pila per poder realitzar les operacions.
La matriu ModelView serveix per poder aplicar les transformacions escalat translació i rotació en les figures geomètriques.
La matriu Projection serveix per fer les projeccions de la escena de 3 a 2 dimensions.
Per guardar aquestes matrius, tindrem una classe auxiliar Matriu que guardarà sobre un vector de 16 posicions tots els punts de les matrius.

També hem de guardar la matriu identityMatrix per poder resetejar l'estat de la matriu a l'estat inicial, tindrem una funció iden(), que s'encarregarà de donar aquesta matriu cada cop que cridem al mètode.

-Analitza quina és la forma de definir un objecte i poder-lo pintar per a que hi intervinguin el mínim número de passos, i per tant, crides a mètodes. Pista: imagina que pintes el polígon definit en el mètode drawRectangle, defineixes el window en la franja (-1,-1) i (1,1) i el viewport que ocupa tota la finestra. Respon a la següent pregunta: quins són els mètodes que has d'implementar en aquest cas?

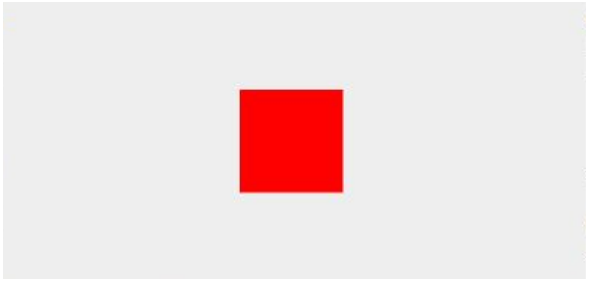
Els mètodes per poder pintar el rectangle són:

```
glViewport(-1, -1, width, height);

glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective();

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glTranslatef();

glPushMatrix();
glBeginPolygon();
        glVertex3f(-1f, -1f, 0.0f);
        glVertex3f(-1f,  1f, 0.0f);
        glVertex3f( 1f,  1f, 0.0f);
        glVertex3f( 1f, -1f, 0.0f);
glEndPolygon();
glPopMatrix();
```



-Complementa el codi anterior per afegir alguna transformació geomètrica, translació, escalat, rotació. Valida cada tg per separat.

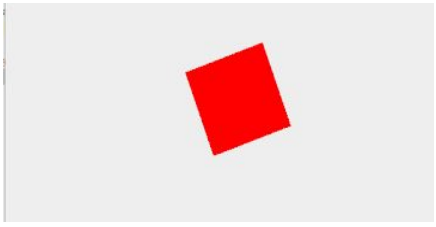
-Rotació:

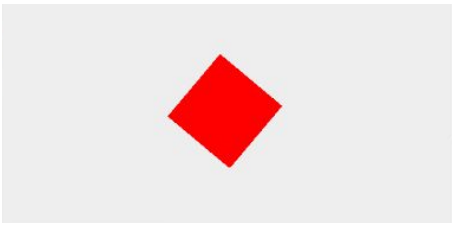
```
glViewport(0, 0, width, height);

glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective();

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glTranslatef();


glPushMatrix();
glRotatef();
glBeginPolygon();
glVertex3f(-1f, -1f, 0.0f);
glVertex3f(-1f,  1f, 0.0f);
glVertex3f( 1f,  1f, 0.0f);
glVertex3f( 1f, -1f, 0.0f);
glEndPolygon();
glPopMatrix();
```





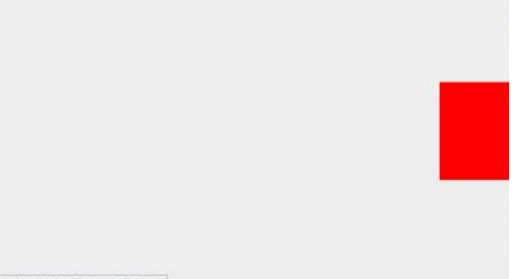
-Translació:

```
glViewport(0, 0, width, height);

glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective();

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glTranslatef();


glPushMatrix();
glTranslatef();
glBeginPolygon();
glVertex3f(-1f, -1f, 0.0f);
glVertex3f(-1f,  1f, 0.0f);
glVertex3f( 1f,  1f, 0.0f);
glVertex3f( 1f, -1f, 0.0f);
glEndPolygon();
glPopMatrix();
```



-Escalat:

```
glViewport(0, 0, width, height);

glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective();

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glTranslatef();


glPushMatrix();
glTranslatef();
glScalef();

glBeginPolygon();
glVertex3f(-1f, -1f, 0.0f);
glVertex3f(-1f,  1f, 0.0f);
glVertex3f( 1f,  1f, 0.0f);
glVertex3f( 1f, -1f, 0.0f);
glEndPolygon();
glPopMatrix();
```

-Acaba el codi per afegir la implementació de tots els mètodes demanats.

Codi de les funcions demanades:

```
  public void defineGlMatrixMode(int model) {
     this.mode=mode;
      this.PilaProj.push(this.iden());
      this.PilaMV.push(this.iden());
  }

public void defineGlViewport(int x, int y, int width, int height) {
      this.x = x;
      this.y = y;
      this.width = width;
      this.height = height;
  }

public void defineGluPerspective(float fovy, float aspect, float zNear, float zFar) {
      Matriu m = new Matriu();

      double ymax, xmax;
      double temp, temp2, temp3, temp4;

      ymax = zNear * Math.tan( (double) fovy * 3.1415926535897930D / 360.0);
      xmax = ymax * aspect;
      Perspective(m, -xmax, xmax, -ymax, ymax, zNear, zFar);

      pilaActual().push((this.pilaActual().pop()).multMatriu(m));

  }

public void defineGlLoadIdentity() {

      pilaActual().push(this.iden());
      this.glScalef(a,b,c);
  }

public void defineGlPushMatrix() {

      Matriu m = this.pilaActual().lastElement();
      this.pilaActual().push(m);
  }
```

```
public void defineGlPopMatrix() {
     this.pilaActual().pop();
  }

   public void defineGlTranslatef(float x, float y, float z) {

      Matriu m = this.iden();
      float[] punts=m.valors();
      punts[3]=x;
      punts[7]=y;
      punts[11]=z;
      m.nousValors(punts);
      m = (this.pilaActual().pop()).multMatriu(m);
      this.pilaActual().push(m);
  }

public void defineGlScalef(float x, float y, float z) {

      Matriu m = this.iden();
      float[] punts=m.valors();
      punts[0]=x;
      punts[5]=y;
      punts[10]=z;
      punts[15]=1.0F;

      m.nousValors(punts);
      m = (this.pilaActual().pop()).multMatriu(m);
      this.pilaActual().push(m);

  }

public void defineGlRotatef(float angle, float x, float y, float z) {

         double radians =angle*3.14/180;
         double cos =Math.cos(radians);
         double sin =Math.sin(radians);

         Matriu m = new Matriu();
         float[] punts=m.valors();

      punts[0]=(float)((x*x)*(1-cos)+cos);
```

```
        punts[1]=(float)((x*y)*(1-cos) - z*sin);
        punts[2]=(float)((x*z)*(1-cos) + y*sin);
        punts[3]=0.0F;
        punts[4]=(float)((y*x)*(1-cos) + z*sin);
        punts[5]=(float)((y*y)*(1-cos) +cos);
        punts[6]=(float)((y*z)*(1-cos) -x*sin);
        punts[7]=0.0F;
        punts[8]=(float)((x*z)*(1-cos) -y*sin);
        punts[9]=(float)((y*z)*(1-cos) +x*sin);
        punts[10]=(float)((z*z)*(1-cos) +cos);
        punts[11]=0.0F;
        punts[12]=0.0F;
        punts[13]=0.0F;
        punts[14]=0.0F;
        punts[15]=1.0F;

        m.nousValors(punts);

        m = pilaActual().pop().multMatriu(m);
        this.pilaActual().push(m);

    }

    public void defineGlBeginPolygon() {
        triangle++;
        if(triangle==1){
            if(restar==true){
                d=d-0.025f;
                e=e-0.025f;
                f=f-0.025f;
                if(d<0.2){
                    restar=false;
                }
            }
            if(restar==false){
                d=d+0.025f;
                e=e+0.025f;
                f=f+0.025f;
                if(d>0.7){
                    restar=true;
                }
            }
```

```
            this.glScalef(d,e,f);
        }
        if(triangle==3){
            triangle=0;
        }
        this.PuntsFinal= new ArrayList<>();
    }

public void defineGlVertex3f(float x, float y, float z) {

        Matriu modelview = this.PilaMV.lastElement();
        Matriu projection = this.PilaProj.lastElement();
        Matriu projxmodelview = projection.multMatriu(modelview);

        Vertex v =new Vertex(x, y, z, 1.0F);
        v=projxmodelview.multVertex(v);

        v.transform(this.height,this.width,this.x,this.y);

        PuntsFinal.add(v.getX());
        PuntsFinal.add(v.getY());

    }

public void defineGlEndPolygon() {
        defineFillPolygon(PuntsFinal.toArray(new Integer[0]));
    }
```

Joc de proves

En aquets joc de proves es mostraran diferents imatges de el resultat del programa.