- INTRODUZIONE
- DRAWING
- EVENT MANAGEMENT
- VIEWING
- DOUBLE BUFFERING
- Z-BUFFERING
- LIGHTING

ESERC.: MODELING

```
void triangle( void )
{
   glBegin( GL_TRIANGLES );
      glVertex2f( -0.5, -0.5 );
      glVertex2f(  0.5, -0.5 );
      glVertex2f(  0.0,  1.0 );
   glEnd();
}
void redraw( void )
{
   glMatrixMode( GL_MODELVIEW );

   glClearColor( 0.0, 0.0, 0.0, 1.0 );
   glClear( GL_COLOR_BUFFER_BIT  );

   glColor3f( 0.0, 1.0, 0.0 );
   triangle();

   glColor3f( 0.0, 0.0, 1.0 );
   glPushMatrix();
   glTranslatef( 1.0, 0.5, 0.0 );
   triangle();
   glPopMatrix();

   glFlush();
}
```

ESERC.: MODELING (2)

```
void redraw( void )
{
   glMatrixMode( GL_MODELVIEW );

   glClearColor( 0.0, 0.0, 0.0, 1.0 );
   glClear( GL_COLOR_BUFFER_BIT  );

   glColor3f( 0.0, 0.0, 1.0 );
   triangle();

   glColor3f( 0.0, 1.0, 0.0 );
   glPushMatrix();
   glTranslatef( 1.5, 0.0, 0.0 );
   glRotatef( 90.0, 0.0, 0.0, 1.0 );
   triangle();
   glPopMatrix();

   glColor3f( 1.0, 0.0, 0.0 );
   glPushMatrix();
   glRotatef( 90.0, 0.0, 0.0, 1.0 );
   glTranslatef( 1.5, 0.0, 0.0 );
   triangle();
   glPopMatrix();

   glFlush();
}
```


ESERC.: ORTHO

```
void resize( int w, int h )
{
   glMatrixMode( GL_PROJECTION );
   glLoadIdentity();

   if( w>=h )
      glOrtho( -3.0*((GLfloat)w/h),
   3.0*((GLfloat)w/h), -3.0, 3.0, -3.0, 3.0 );
   else
      glOrtho( -3.0, 3.0, -3.0*((GLfloat)h/w),
   3.0*((GLfloat)h/w), -3.0, 3.0 );

   glViewport( 0, 0, w, h );
}
void redraw( void )
{
   glMatrixMode( GL_MODELVIEW );
   glClearColor( 0.0, 0.0, 0.0, 1.0 );
   glClear( GL_COLOR_BUFFER_BIT  );

   glPushMatrix();
   glColor3f( 0.0, 1.0, 0.0 );
   glScalef( 2.0, 1.0, 0.7 );
   glutWireCube( 1.0 );
   glPopMatrix();

   glFlush();
}
```

ESERC.: MODELING (3)

- ruotare il parallelepipedo in modo arbitrario

```
glPushMatrix();
glRotatef( 30, 1.0, 0.5, 0.0 );
glScalef( 2.0, 1.0, 0.7 );
glutWireCube( 1.0 );
glPopMatrix();
```

- aggiungerne un altro in una differente posizione dello spazio

```
glColor3f( 1.0, 0.0, 0.0 );
glPushMatrix();
glTranslatef( 1.0, 0.0, -0.5 );
glRotatef( 30, 0.5, 0.3, 0.7 );
glScalef( 1.0, 1.0, 3.0 );
glutWireCube( 1.0 );
glPopMatrix();
```

- …e cosi` via

ESERC.: PERSPECTIVE

```
void resize( int w, int h )
{
   glMatrixMode( GL_PROJECTION );
   glLoadIdentity();
   gluPerspective(60.0,(GLfloat)w/h,1.0,200.0);

   glMatrixMode( GL_MODELVIEW );
   glLoadIdentity();
   gluLookAt( -40.0, 50.0, 100.0, 5.0, 10.0,
   15.0, 0.0, 1.0, 0.0 );
   glViewport( 0, 0, w, h );
}
void redraw( void )
{
   glMatrixMode( GL_MODELVIEW );
   glClearColor( 0.0, 0.0, 0.0, 1.0 );
   glClear( GL_COLOR_BUFFER_BIT );

   glPushMatrix();
   glColor3f( 1.0, 0.0, 0.0 );
   glScalef( 60.0, 30.0, 20.0 );
   glutWireCube( 1.0 );
   glPopMatrix();

   glPushMatrix();
   glColor3f( 0.0, 1.0, 0.0 );
   glScalef( 20.0, 60.0, 30.0 );
   glutWireCube( 1.0 );
   glPopMatrix();

   glFlush();
}
```

SCHEMA PROGRAMMA

```
#include <GL/glut.h>

void resize( int w, int h )
{
   glMatrixMode( GL_PROJECTION );
   glLoadIdentity();

   /* manipolazione projection matrix */

   glMatrixMode( GL_MODELVIEW );
   glLoadIdentity();

   /* manipolazione modelview matrix */

   /* definizione viewport transformation */
}
void redraw( void )
{
   glMatrixMode( GL_MODELVIEW );

   /* modeling & drawing */

   glFlush();
}
void main( int argc, char** argv )
{
   ...
   glutReshapeFunc ( resize );
   glutDisplayFunc ( redraw );
   glutMainLoop    (        );
}
```


RESIZE ORTHO2D

```
/* proiezione ortogonale parallela "2D" */

void resize( int w, int h )
{
GLfloat
   Xmin = ... , Xmax = ... ,
   Ymin = ... , Ymax = ... ;

  glMatrixMode( GL_PROJECTION );
  glLoadIdentity();

  if( w>=h )
     gluOrtho2D( Xmin*((GLfloat)w/h),
      Xmax*((GLfloat)w/h), Ymin, Ymax );
  else
     gluOrtho2D( Xmin, Xmax,
      Ymin*((GLfloat)h/w),Ymax*((GLfloat)h/w));

  glViewport( 0, 0, w, h );
}
```

RESIZE ORTHO (3D)

```
/* proiezione ortogonale parallela (3D) */

void resize( int w, int h )
{
GLfloat
   Xmin = ... , Xmax = ... ,
   Ymin = ... , Ymax = ... ,
   near = ... , far  = ... ;

    glMatrixMode( GL_PROJECTION );
    glLoadIdentity();

    if( w>=h )
       glOrtho(Xmin*((GLfloat)w/h),
       Xmax*((GLfloat)w/h),Ymin,Ymax,near,far);
   else
      glOrtho(Xmin,Xmax,Ymin*((GLfloat)h/w),
       Ymax*((GLfloat)h/w),near,far);

   glViewport( 0, 0, w, h );
}
```


RESIZE PERSPECTIVE

```
/* proiezione prospettica */

void resize( int w, int h )
{
GLfloat
   fovY  = ... , near  = ... , far   = ... ,
   eyeX  = ... , eyeY  = ... , eyeZ  = ... ,
   targX = ... , targY = ... , targZ = ... ,
   vwupX = ... , vwupY = ... , vwupZ = ... ;

   glMatrixMode( GL_PROJECTION );
   glLoadIdentity();
   gluPerspective(fovY,(GLfloat)w/h,near,far);

   glMatrixMode( GL_MODELVIEW );
   glLoadIdentity();
   gluLookAt(eyeX,eyeY,eyeZ,targX,targY,targZ,
             vwupX,vwupY,vwupZ);

   glViewport( 0, 0, w, h );
}
```

GLUT PRIMITIVES

- wireframe:

```
glutWireSphere(GLdouble,GLint,GLint);
glutWireCone(GLdouble,GLdouble,GLint,GLint);
glutWireCube(GLdouble);
glutWireTorus(GLdouble,GLdouble,GLint,GLint);
glutWireDodecahedron(void);
glutWireTeapot(GLdouble);
glutWireOctahedron(void);
glutWireTetrahedron(void);
glutWireIcosahedron(void);
```

- solid:

```
glutSolidSphere(GLdouble,GLint,GLint);
glutSolidCone(GLdouble,GLdouble,GLint,GLint);
glutSolidCube(GLdouble);
glutSolidTorus(GLdouble,GLdouble,GLint,GLint);
glutSolidDodecahedron(void);
glutSolidTeapot(GLdouble);
glutSolidOctahedron(void);
glutSolidTetrahedron(void);
glutSolidIcosahedron(void);
```


ESERC.: MODELING (4)

- modellare una scena a piacere utilizzando le primitive di glut