MIEI / LCC
Departamento de Informática
Universidade do Minho

Basic Interactivity with GLUT
Graphical Primitives

GLUT– Mouse and Keyboard

OpenGL – Drawing with triangles

Basic GLUT Interactivity

- GLUT supports a range of input devices:
  - Mouse
  - Keyboard
  - Trackball
  - Tablet
- Using these devices implies **writing functions to process the respective events**, and
- **Registering these functions** with GLUT.

Keyboard – Callback Registry

- Regular keys (letters, numbers , etc...)

To register the callback use:

```
glutKeyboardFunc(function_name);
```

Function signature:

```
void function_name (unsigned char key, int x, int y);
```

This function will be called by GLUT when a regular key is pressed. The parameters are the key itself and the actual mouse coordinates relative to the window.

Keyboard – Callback Registry

- Special Keys (F1..F12, Home, End, Arrows, etc...)

To register the callback use:

```
glutSpecialFunc(function_name);
```

Function signature:

```
void function_name(int key_code, int x, int y);
```

The key codes are constants defined in glut.h. Some examples are: `GLUT_KEY_F1` and `GLUT_KEY_UP`.

Mouse – Callback Registry

- Mouse: pressing and releasing a button

To register the callback use:

```
glutMouseFunc(function_name);
```

Function signature:

```
void function_name (int button, int state, int x, int y);
```

The parameters are:

- Which button (`GLUT_LEFT_BUTTON`, `GLUT_MIDDLE_BUTTON`, `GLUT_RIGHT_BUTTON`);
- Button state (`GLUT_UP`, `GLUT_DOWN`);
- Mouse position in window relative coordinates.

Mouse – Callback Registry

- Mouse: passive and active motion

To register the callback use:

```
glutMotionFunc(function_name);
glutPassiveMotionFunc(function_name);
```

Function signature:

```
void function_name(int x, int y);
```

The parameters are the window relative mouse coordinates.

Resource Management

- When using the idle function, GLUT is constantly redrawing the scene.
- For static scenes we only need to redraw when the camera moves.
- To avoid unnecessary redraw we can call the following function when the camera moves

```
glutPostRedisplay()
```

- `glutPostRedisplay` generates an event stating that the window needs to be redrawn. The event will be placed in the event queue for later processing.

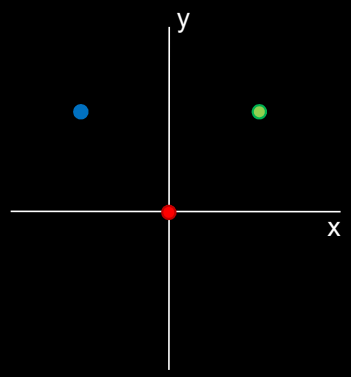
3D Modelling

- 3D vertex definition

```
glVertex3f(x,y,z);
```

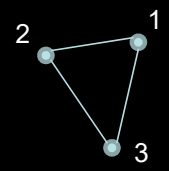
- To draw a triangle:

```
glBegin(GL_TRIANGLES);
     glVertex3f(0.0f, 0.0f, 0.0f);
     glVertex3f(1.0f,  1.0f, 0.0f);
     glVertex3f(-1.0f, 1.0f, 0.0f);
glEnd();
```

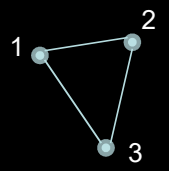


3D Modelling

- Polygon orientation
  - OpenGL allows for optimization by only drawing polygons which are facing the camera. To define the front face of a polygon we use the right hand rule.



Poligon facing forward

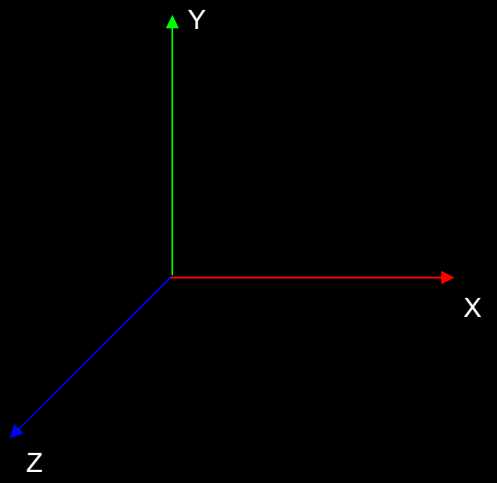


Poligon facing backward

3D Modelling

- Drawing Axis with lines

```
glBegin(GL_LINES);
      // X axis in red
      glColor3f(1.0f, 0.0f, 0.0f);
      glVertex3f(-100.0f, 0.0f, 0.0f);
      glVertex3f( 100.0f, 0.0f, 0.0f);
      // Y Axis in Green
      glColor3f(0.0f, 1.0f, 0.0f);
      glVertex3f(0.0f, -100.0f, 0.0f);
      glVertex3f(0.0f,  100.0f, 0.0f);
      // Z Axis in Blue
      glColor3f(0.0f, 0.0f, 1.0f);
      glVertex3f(0.0f, 0.0f, -100.0f);
      glVertex3f(0.0f, 0.0f,  100.0f,);
glEnd();
```



3D Modelling

- Face Culling

```
glEnable(GL_CULL_FACE);
glCullFace(GL_FRONT ou GL_BACK);
```

- Defining default polygon orientation:

```
glFrontFace(GL_CW ou GL_CCW);
```

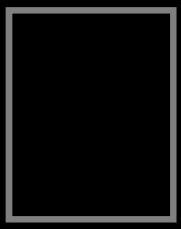
3D Modelling

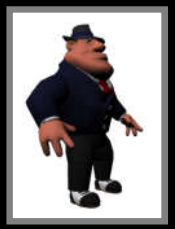
- Drawing polygon mode

  `glPolygonMode(face,mode);`

- possible values for face:
  - `GL_FRONT, GL_BACK, GL_FRONT_AND_BACK`

- possible values for mode:
  - `GL_FILL, GL_LINE, GL_POINT`





Required Functions

- OpenGL and GLU

```
glTranslatef(x,y,z); // moves the object

glRotatef(angle,x,y,z); // angle is in degrees

glScalef(x,y,z); // scale factors for each axis

glColor3f(r,g,b); // color in RGB. Each component varies between 0 and 1. (1,1,1) is
  white, (0,0,0) is black.

gluLookAt(px,py,pz,  lx,ly,lz,  ux,uy,uz);
  // px,py,pz - camera position
  // lx,ly,lz - look at point
  // ux,uy,uz - camera tilt, by default use (0.0, 1.0, 0.0)
```

Assignment

- Complete the provided code skeleton to create an interactive application with a pyramid (each face with a different colour).
- The keyboard should allow to move the pyramid in the XZ plane, rotate it around its vertical axis, and scale its height.
  - Try swapping the order of the geometric transformations. Interpret the result.
- Use the keyboard to select the drawing mode (`GL_FILL, GL_LINE, GL_POINT`).
- Use `glutPostRedisplay;`

The Pyramid