MIEI / LCC
DEPARTAMENTO DE INFORMÁTICA
UNIVERSIDADE DO MINHO

Basic Interactivity with GLUT
Graphical Primitives

GLUT– Mouse, Keyboard and Popup Menus

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 exemples 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 nome_função(int x, int y);
```

The parameters are the window relative mouse coordinates

Interactivity with GLUT

- GLUT allows the definition of pop-up menus.
- The menu items can be further menus providing an hierarchical structure.

Interactivity with GLUT – Pop-up Menus

- First step: create a menu.
  - To create a menu we must tell GLUT which function will process its events.

```
int glutCreateMenu(function_name);
```

  - The return value of `glutCreateMenu` is the menu id.
  - The registered function will be called when the user selects a menu option, and receives the menu option id.
  - Function signature:

```
void function_name (int id_op);
```

Interactivity with GLUT – Pop-up Menus

- Second step: Adding menu entries.

```
void glutAddMenuEntry(char *op, int id_op);
```

  - Ex: `glutAddMenuEntry("Red",1);`
  - The entries are always added to the end of the menu.

- Third step: Bind a mouse button to the menu

```
glutAttachMenu(int button);
```

  - button= `GLUT_LEFT_BUTTON`, `GLUT_RIGHT_BUTTON`, or `GLUT MIDDLE_BUTTON`

Interactivity with GLUT – Pop-up Menus

- Notes:
  - When adding entries to a menu, or binding it to a mouse button there is no reference to which menu. When a menu is created it becomes the “current” menu, and all menu operations are relative to this menu.
  - Using the id returned upon the menu creation we can make a previously created menu the “current” menu
    - **`glutSetMenu(int menuId);`**

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.

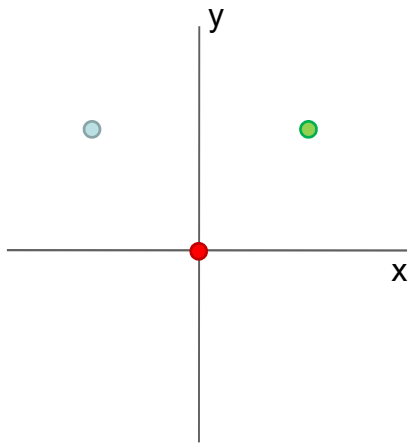
3D Modelling

- 3D vertex definition

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

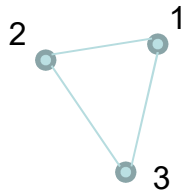
- To draw a triangle:

```
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();
```

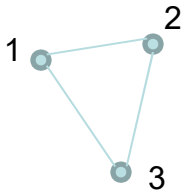


3D Modelling

- Polygon orientation
  - OpenGL allows saving resources 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

- *Face Culling*

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

- Defining default polygon orientation:

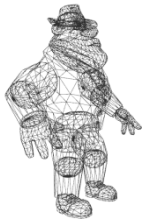
```
glFrontFace(GL_CW ou GL_CCW);
```

3D Modelling

- Drawing polygon mode

```
glPolygonMode(face,mode);
```

- face:
  - `GL_FRONT, GL_BACK, GL_FRONT_AND_BACK`
- 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

glColor3f(r,g,b); // the color in RBG. Each componente varies
   between 0 and 1.

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 color).
- The keyboard should allow to move the pyramid in the XZ plane and change its height.
- Use `glutPostRedisplay`;
- Create a popup menu to select the drawing mode.

The pyramid