LEI / LCC
Departamento de Informática
Universidade do Minho

Practical Class nº 1

OpenGL and GLUT

Summary

- Libraries
- Event oriented programming
- Programming with GLUT
- Base code skeleton
- Geometrical primitives available in GLUT
- Today's assignement
- Crash course in VS

Libraries

- OpenGL (Open Graphics Library)
  - 3D and 2D graphics
- GLU (GL Utilities)
  - Some useful functions we will call repeatedly
- GLUT (GL Utility Toolkit)
  - Building cross platform applications (Win, Xwin, OSX)
- AntTweakBar (User Interface)
  - Simple and intuitive library to design basic user interfaces

Event Oriented Programming

- Define an action for each relevant event

- Event examples:
  - Key pressed
  - Mouse button pressed
  - Mouse movement
  - Window resize
  - Window requires painting

Event Oriented Programming

- The application is controlled by the window manager (GLUT).

- **Define** a *set of functions* to process *events* ...

- and **register** these functions with GLUT
  - Tell GLUT which function to call for each event

Programming with GLUT

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


…
int main(int argc, char **argv) {

   // init GLUT and the window

   // register the functions that will process the events

   // enter GLUT's main cicle

   return 1;
}
```

GLUT - Initialization

```
glutInit(&argc, argv);
```

- This function will init GLUT itself.
- The parameters are the same as in the `main` function.

GLUT - Initialization

`glutInitDisplayMode(...);`

- Defines the window properties (more on this in the theoretical classes)
- ... meanwhile consider the following value as the parameter of the above function:

```
GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA
```

GLUT - Initialization

```
glutInitWindowPosition(100,100);
```

- Top left window position

```
glutInitWindowSize(800,800);
```

- Width and height of the window's client area in pixels.

GLUT - Initialization

```
glutCreateWindow("CG@DI");
```

- Creating the window. The string argument will appear as the window's caption

- Note: the window will only be visible upon entering GLUT's main cycle with
  `glutMainLoop( );`

Programming with GLUT

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

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

   // init GLUT and the window

   // register the functions that will process the events
   (callbacks)

   // enter GLUT's main cicle


   return 1;
}
```

Callback Registry

`glutDisplayFunc( `*`function_name`*` );`

- The callback function responsible for the window's contents.
- GLUT requires the registration of this callback.
- Function signature:

```
void function_name (void);
```

Callback Registry

`glutReshapeFunc( `*`function_name`*` );`

- The registered function will be called when the window is created and when it is resized.

- Function signature:

```
void function_name (int width, int height);
```

Where the input parameters, `width` and `height`, are the window's dimension.

Callback Registry

`glutIdleFunc( `*`function_name`*` );`

- The registered function will be called when the event queue is empty.
- This makes it particularly suitable for situations where repeated redraw is required, for instance in continuous animations.
- Function signature :

```
void function_name(void);
```

Programming with GLUT

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


…
int main(int argc, char **argv) {

   // init GLUT and the window

   // register the functions that will process the events

   // enter GLUT's main cicle

   return 1;
}
```

GLUT's Main Cycle

```
glutMainLoop( );
```

- Calling this function enters GLUT's main cycle.
- The incoming events, such as window resize, paint, keyboard, etc…, are placed in a queue as they arrive and processed in order.
- For each event, GLUT will call the associated registered function.

Base Code Skeleton

- Main

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

// put GLUT's init here



// put callback registry here


// some OpenGL settings
    glEnable(GL_DEPTH_TEST);
    glEnable(GL_CULL_FACE);
    glClearColor(0.0f, 0.0f, 0.0f, 0.0f);

// enter GLUT's main cycle
    glutMainLoop();
    return 1;
}
```

Base Code Skeleton

- Reshape Func

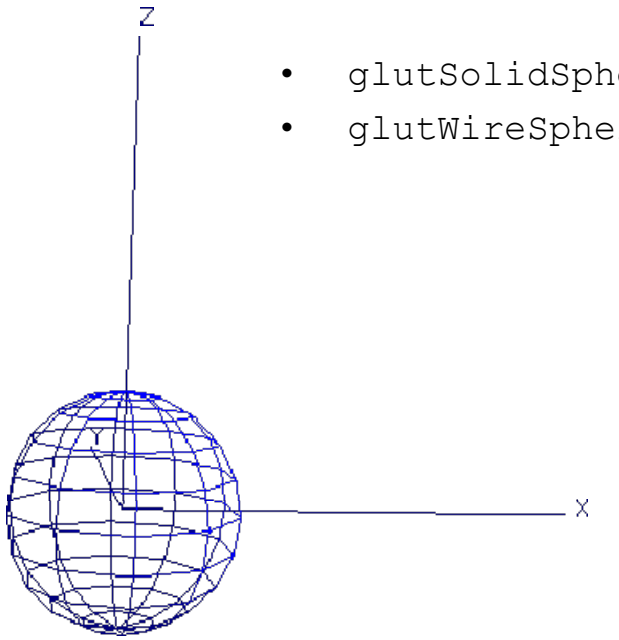
```
void changeSize(int w, int h) {

     // Prevent a divide by zero, when window is too short
     // (you can't make a window with zero width).
     if(h == 0)
             h = 1;

     // compute window's aspect ratio
     float ratio = w * 1.0f / h;

     // Set the projection matrix as current
     glMatrixMode(GL_PROJECTION);
      // Load the identity matrix
     glLoadIdentity();

     // Set the viewport to be the entire window
    glViewport(0, 0, w, h);

     // Set the perspective
     gluPerspective(45.0f, ratio, 1.0f, 1000.0f);

     // return to the model view matrix mode
     glMatrixMode(GL_MODELVIEW);
}
```

Base Code Skeleton

- Display and Idle Func

```
void renderScene(void) {

    // clear buffers
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    // set camera
    glLoadIdentity();
    gluLookAt(0.0f, 0.0f, 5.0f,
              0.0f, 0.0f, -1.0f,
              0.0f, 1.0f, 0.0f);

// put drawing instructions here


    // End of frame
    glutSwapBuffers();
}
```

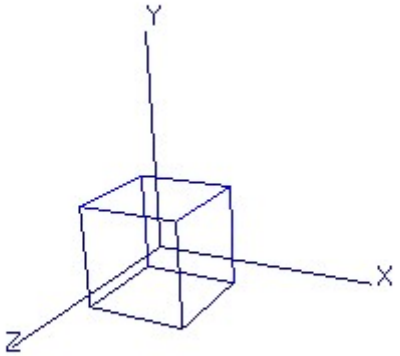
GLUT – Graphical Primitives



- `glutSolidSphere(float radius, int slices, int stacks);`
- `glutWireSphere (float radius, int slices, int stacks);`

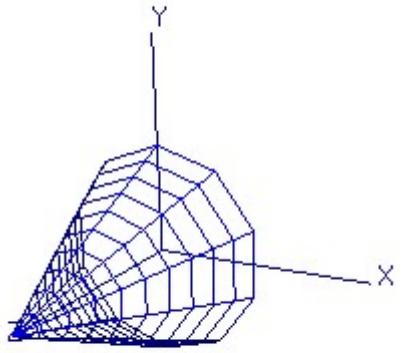
GLUT – Graphical Primitives

- `glutSolidCube(float dimension);`
- `glutWireCube (float dimension);`



GLUT – Graphical Primitives

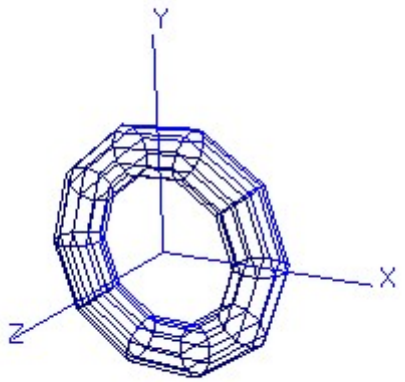
- `glutSolidCone(float baseRadius, float height, int slices, int stacks);`
- `glutWireCone (float baseRadius, float height, int slices, int stacks);`



GLUT - Graphical Primitives

- `glutSolidTorus(float innerRadius, float outterRadius, int sides, int rings);`

- `glutWireTorus(float innerRadius, float outterRadius, int sides, int rings);`



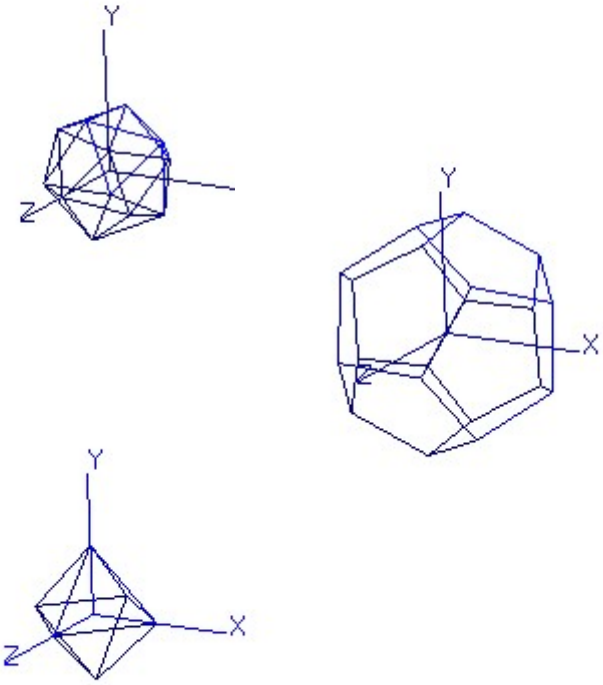
GLUT - Graphical Primitives

- `glutSolidIcosahedron(void);` (20 faces)
- `glutWireIcosahedron(void);`

- `glutSolidDodecahedron(void);` (12 faces)
- `glutWireDodecahedron(void);`

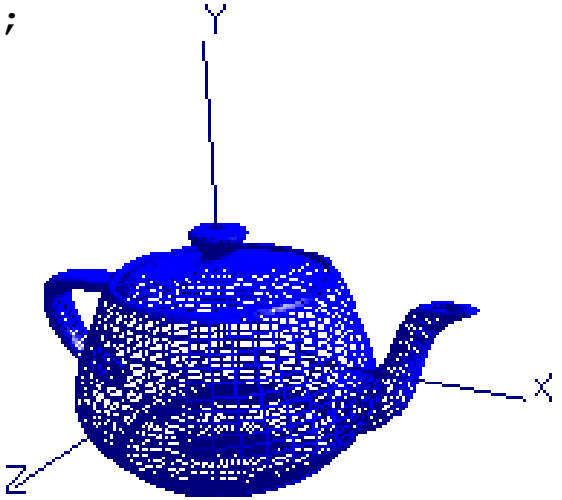
- `glutSolidOctahedron(void);` (8 faces)
- `glutWireOctahedron(void);`

- `glutSolidTetrahedron(void);` (6 faces)
- `glutWireTetrahedron(void);`



GLUT - Graphical Primitives

- `glutSolidTeapot(float dimension);`
- `glutWireTeapot(float dimension);`



Class Practical Assignment

- Fill the provided code skeleton to build an application with OpenGL + GLUT.
- The application should draw a wire frame teapot.
- The teapot's dimension should be used to perform an animation (for instance varying the dimension with a sine function)
- Try with other GLUT's primitives.

Crash Course em VS

- File -> New -> Project

- Project Type: Win32
- Templates: Win32 Console Application
- Name: solution and project's name (note: a solution is a set of projects)
- **OK**

- **Next**

- Additional Options: uncheck everything
- Empty project (check)
- **Finish**

- Mouse right button in"Source Files" in the Solution Window
- Add->New Item
- Select c++ and name the file

- Copy the provided skeleton to this file

Crash Course em VS

- GLUT (Ficheiros: glut.h, glut32.lib, glut32.dll)

Housekeeping suggestion:

Create a folder to place all the third party library files (ex: c:\toolkits)

Create three folders inside this folder (includes, libs e dlls)

Create a folder (c:\toolkits\includes\GL), and add glut.h to it

Place glut32.lib in folder libs, and glut32.dll in folder dlls

Tell Visual Studio where to find the libs and the dlls
Right click on the projects name in the solution window and select Properties
Select VC++ Directories
Add path to libs and includes folders

Add the path to dlls folder to environment variable *Path* (c:\toolkits\dlls)
(Control Panel -> System -> Advanced System Settings -> Environment Variables)

Crash Course in VS

- Add button “Start without Debugging” to the toolbar

- Menu tools -> customize
- Select tab “Commands”
- Select “Toolbar”->”Standard”
- Click “Add Command…”
- On the left side select “Debug”
- On the right side select “Start Without Debugging”
- Click “OK”
- Now the icon should be visible in the toolbar.
- Click “Move Down” to move the icon next to the “Start debugging” icon