COMPUTER GRAPHICS

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 (we will up to use GL 2.1)

- **GLU (GL Utilities)**
  - Some useful functions we will call repeatedly

- **GLUT or FreeGLUT (GL Utility Toolkit)**
  - Building cross platform applications (Win, Xwin, OSX)

- **AntTweakBar (User Interface)**
  - Simple and intuitive library to design basic user interfaces
  - http://anttweakbar.sourceforge.net/doc/tools:anttweakbar:howto

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).

- We only have to:

    - **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 obey the same rules as the arguments from the main function.
  - See https://www.opengl.org/resources/libraries/glut/spec3/node10.html

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();
}
```
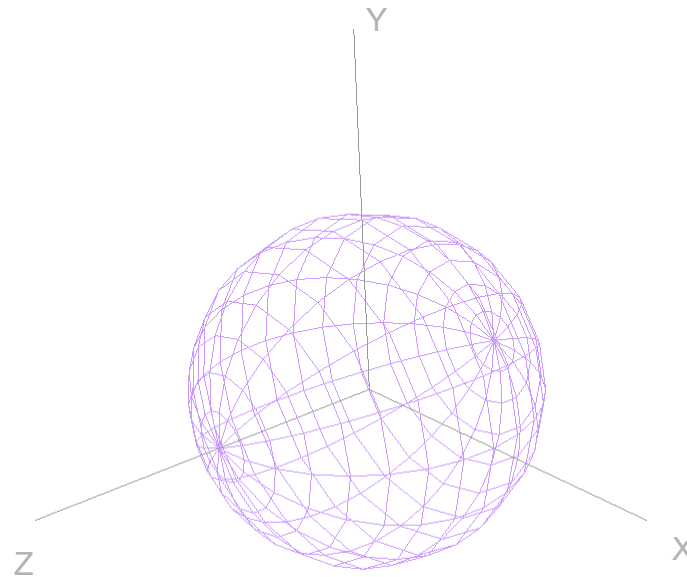
GLUT – Graphical Primitives

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

DI-UM   COMPUTER GRAPHICS 16/17
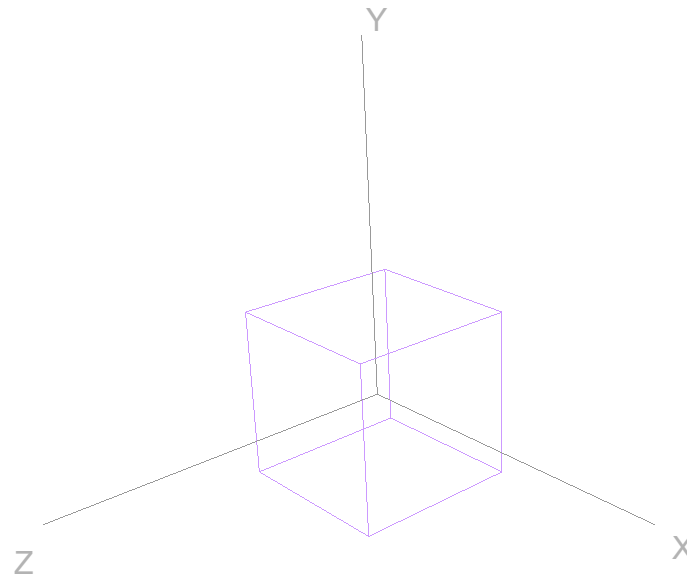
GLUT – Graphical Primitives

- `glutSolidSphere(float radius, int slices, int stacks);`
- `glutWireSphere (float radius, int slices, int stacks);`
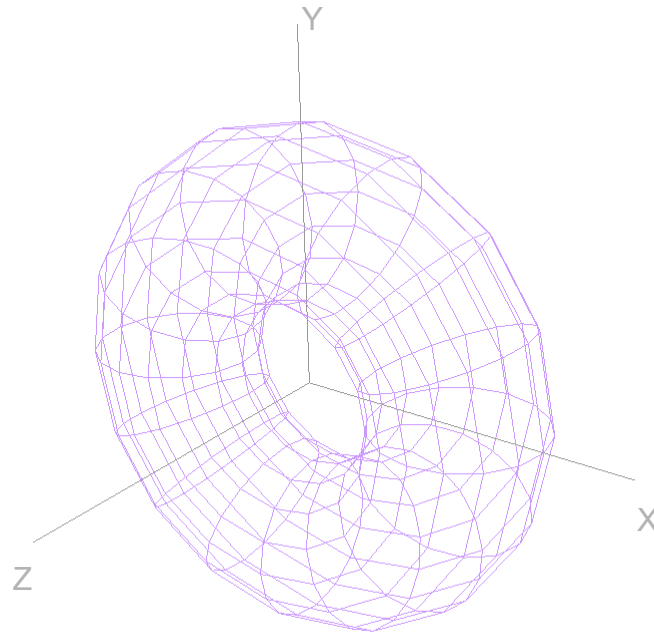
GLUT – Graphical Primitives

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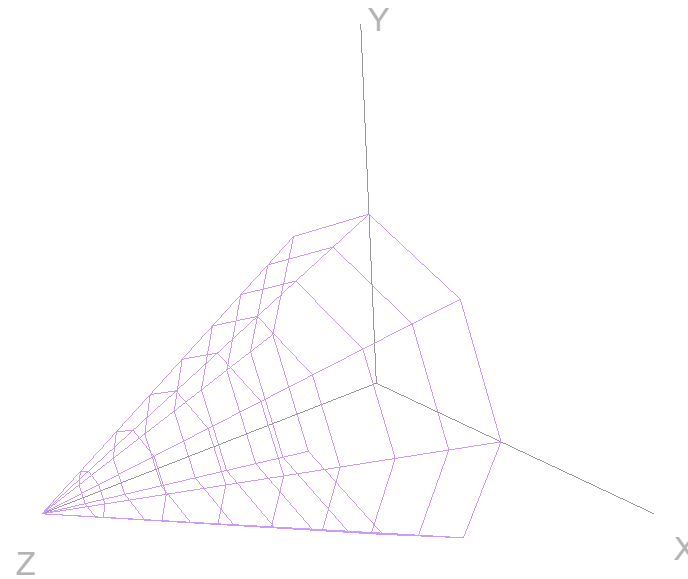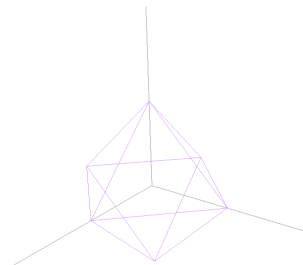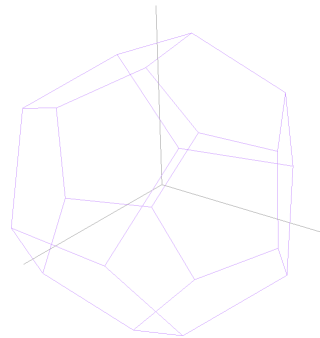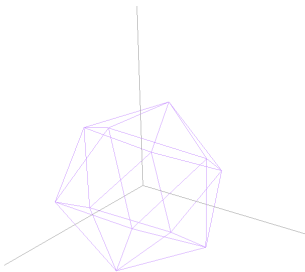
GLUT - Graphical Primitives

- 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);

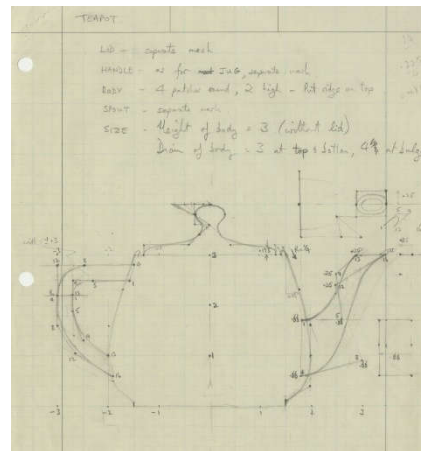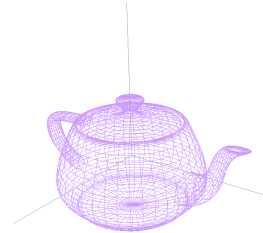- glutSolidTetrahedron(void); (6 faces)
- glutWireTetrahedron(void);
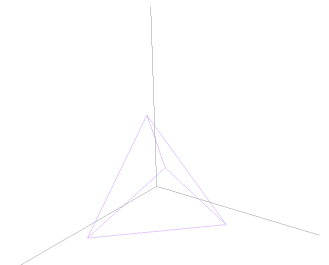
GLUT - Graphical Primitives

- `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.

Getting things ready – Linux (Ubuntu)

- Install cmake and cmake-qt gui
  - sudo apt-get install cmake
  - sudo apt-get install cmake-qt-gui

- Install freeglut
  - sudo apt-get install freeglut3-dev

- Check OpenGL version
  - glxinfo | grep "OpenGL"

Getting things ready - Linux

- Get the zip in the course page and decompress the zip file to a folder (the project folder)

- Open CMake from a terminal window: `cmake-gui &`

- In the CMake window:
    - "Where is the source code": input the project folder
    - "Where to build the sources": a new subfolder
    - Press "Configure"
    - If errors appear such as:
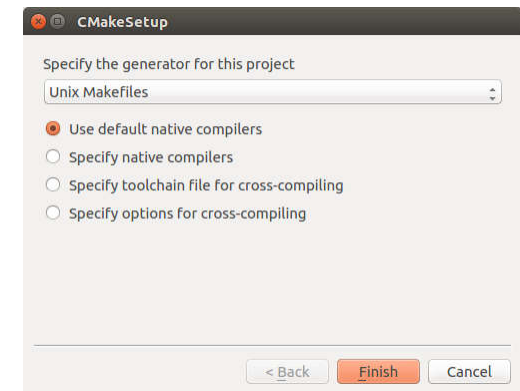
```
CMake Error: The following variables are used in this project, but they are set to NOTFOUND.
Please set them or make sure they are set and tested correctly in the CMake files:
GLUT_Xi_LIBRARY (ADVANCED) linked by target "class1" in directory …
GLUT_Xmu_LIBRARY (ADVANCED) linked by target "class1" in directory ...
```

Try : `sudo get-apt install libxmu-dev libxi-dev`

And press "Configure" again (this time there should be no errors)
   - Press "Generate"

Getting things ready - Linux

- Open a terminal in the build folder and type:
  - `make class1`

- To run the app write
  - `./class1`

- Note: if you run the app now you'll get the following message:
  - freeglut ERROR: Function <glutMainLoop> called without first calling 'glutInit'.

- This is because the code is incomplete. To show a window you must complete at least the glut initialization and callback registration. To show something in the window you'll need to complete the render function.

Getting things ready - Windows

- Create a folder for the toolkits you'll be using in the course, for instance "toolkits"
- Create a subfolder named glut
- Download the GLUT zip file and copy the files to the glut folder
- Create a subfolder called GL and place glut.h inside this folder
- The picture shows the proposed folder structure

```
toolkits
    glut
        GL
            glut.h
        glut32.dll
        glut32.lib
```
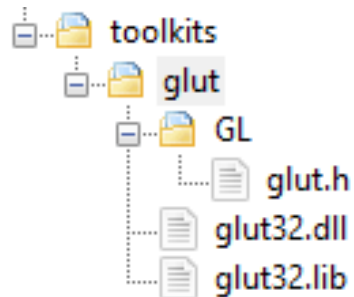
# Getting things ready - Windows

- Get Cmake (https://cmake.org/download/) and install it

- Get the zip in the course page and decompress the zip file to a folder (the project folder)

- Open Cmake
  - Where is the source code: input the project folder
  - Where to build the sources: commonly set to be a subfolder of the project folder (for instance "bin")
  - Press "Configure"

Getting things ready - Windows

- Select the generator as shown in the image
- Press "Finish"
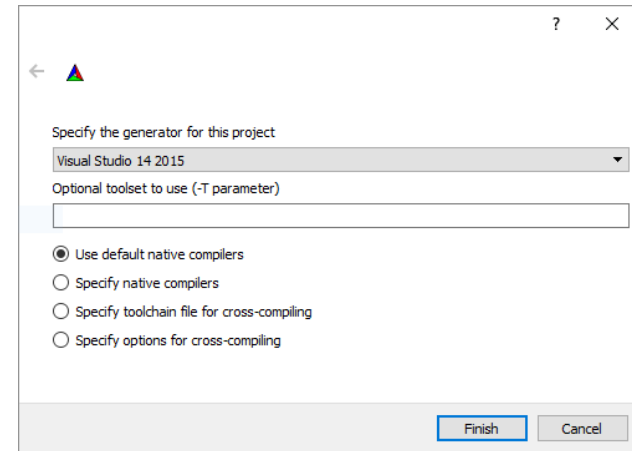
- You'll get an error – press OK
- Look for the red lines that say NOTFOUND

| GLUT_INCLUDE_DIR	GLUT_INCLUDE_DIR-NOTFOUND
GLUT_glut_LIBRARY	GLUT_glut_LIBRARY-NOTFOUND

- Press the first line and select the folder where GLUT was placed
- Press the second line and select the file glut32.lib
- Press "Configure" again
  - The bottom window should display "Configuring done"
- Press "Generate" and then press "Open Project"
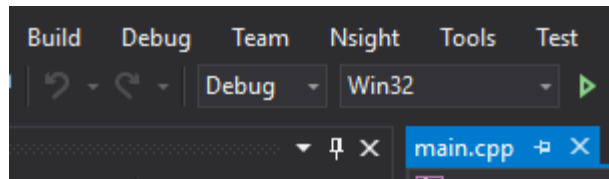
Getting things ready - Windows

- Right mouse click on "class1" in Solution Explorer
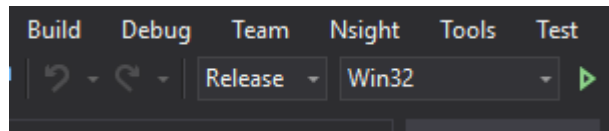
- Select "Set as StartUp Project"
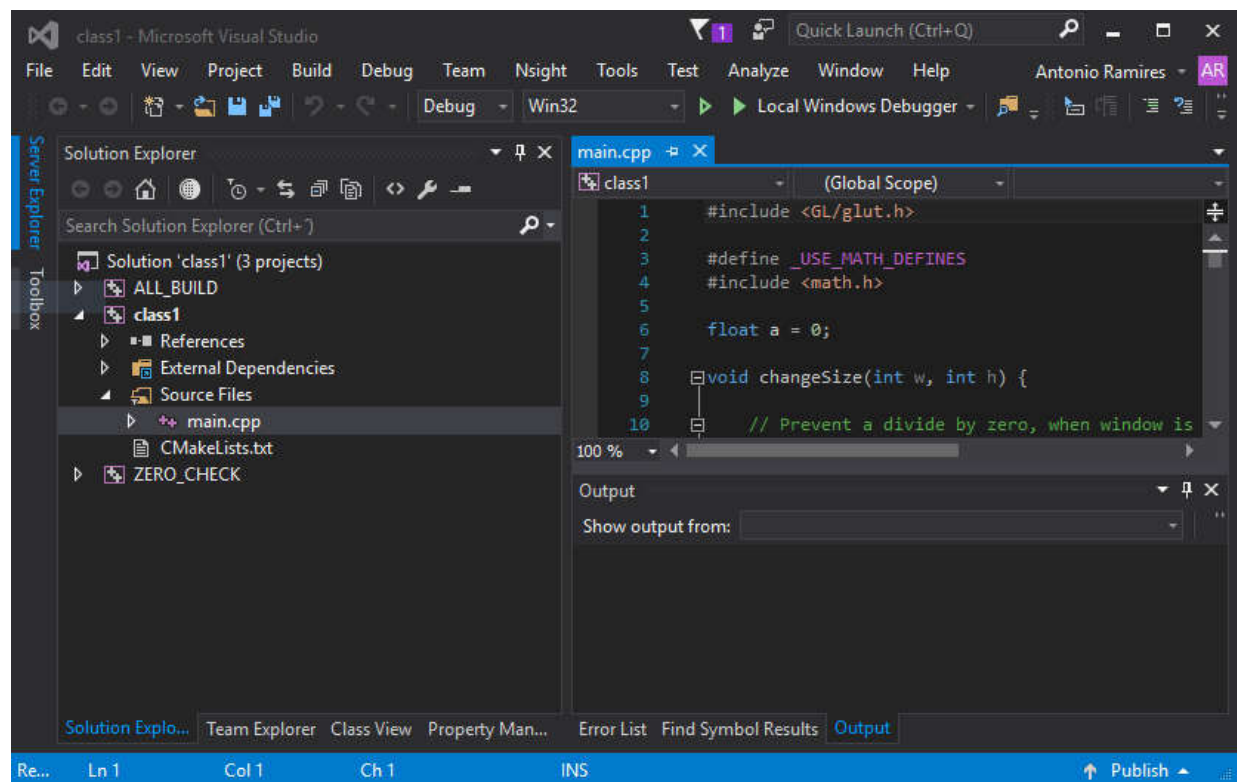
Getting things ready - Windows

- A few more things:
  - VS by default starts with the debug configuration active

  - As a rule we should work in the release configuration

- Before trying to run the project copy glut32.dll to the folder where the sources were built

- Press Ctrl-F5 to run the project (note: as is the provided code is incomplete and hence it will not produce the desired result. You must at least perform the GLUT initialization and call back registration to see the OpenGL window)