Computer Graphics

MIEI / 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 assignment
- Getting things ready

Libraries

- OpenGL (Open Graphics Library)
  - 3D and 2D graphics (we will use up to 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 a set of window properties (more on this in the theory classes)
- ... meanwhile use the following value as the parameter of the above function:

```
GLUT_DEPTH | GLUT_DOUBLE | GLUT_RGBA
```

GLUT - Initialization

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

- Sets the position of the top left corner of the window, in pixels

```
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 redrawing 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 new window dimensions.

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.

GLUT's Main Cycle

- Inner workings of GLUT main cycle (using GLFW as an example)

```
while (!glfwWindowShouldClose(window)) {


    renderScene();

    glfwSwapBuffers(window);
    glfwPollEvents();
}

glfwDestroyWindow(window);

glfwTerminate();
exit(EXIT_SUCCESS);
```

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);
}
```

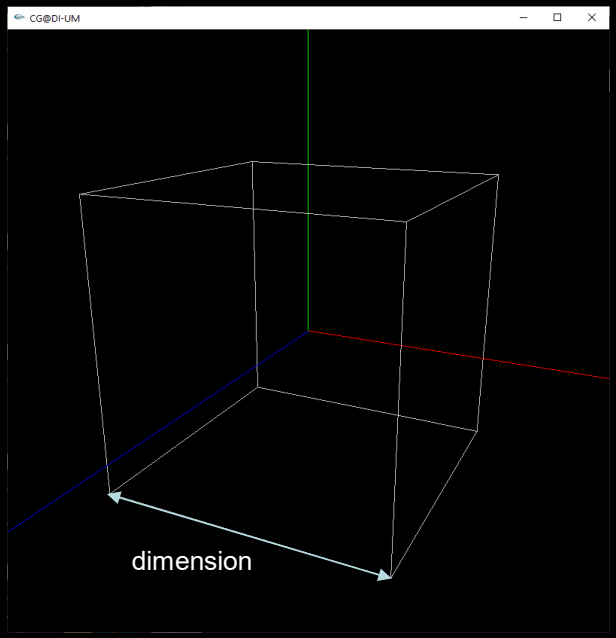
Base Code Skeleton

- 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();
}
```

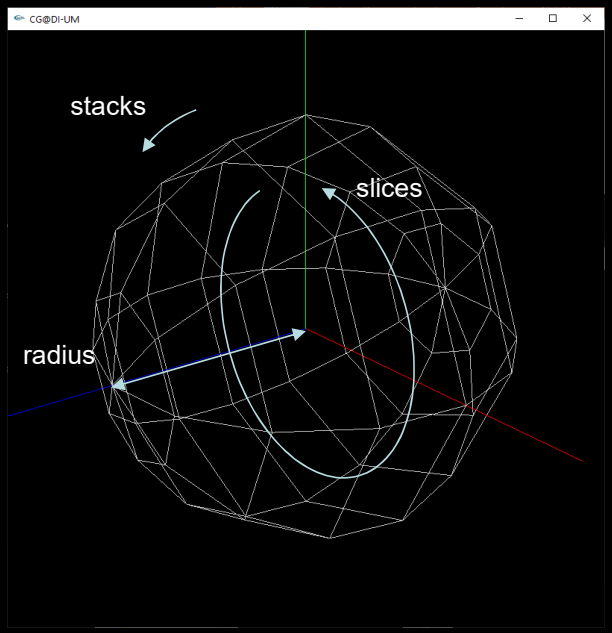
GLUT – Graphical Primitives

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



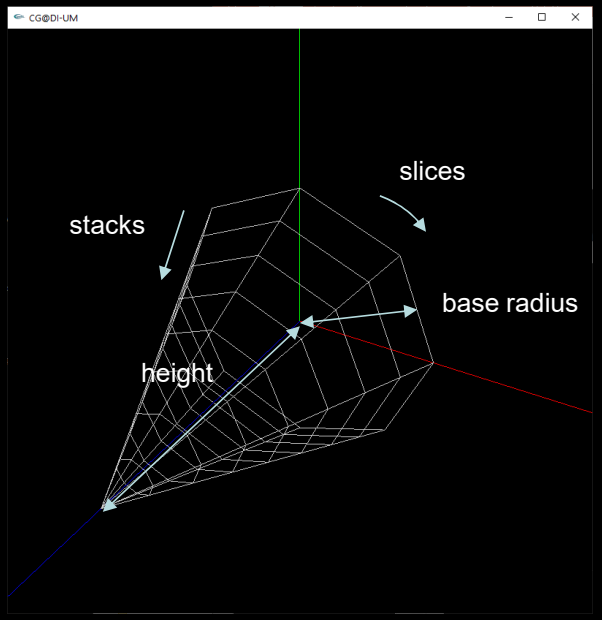
GLUT – Graphical Primitives

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



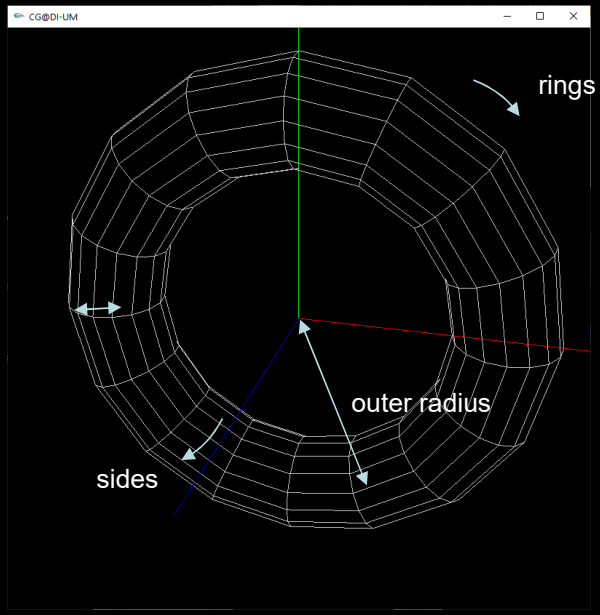
GLUT – Graphical Primitives

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



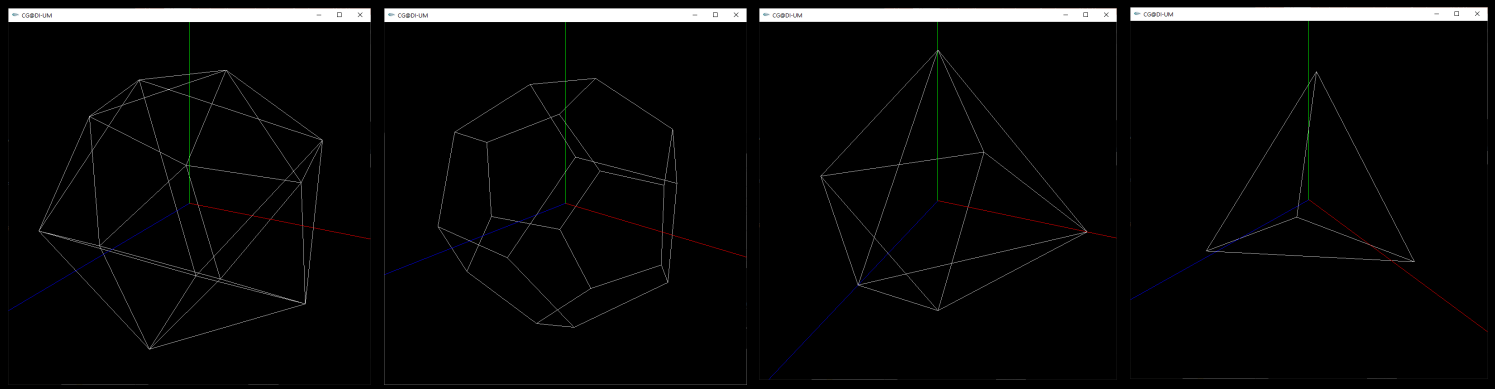
GLUT - Graphical Primitives

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

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

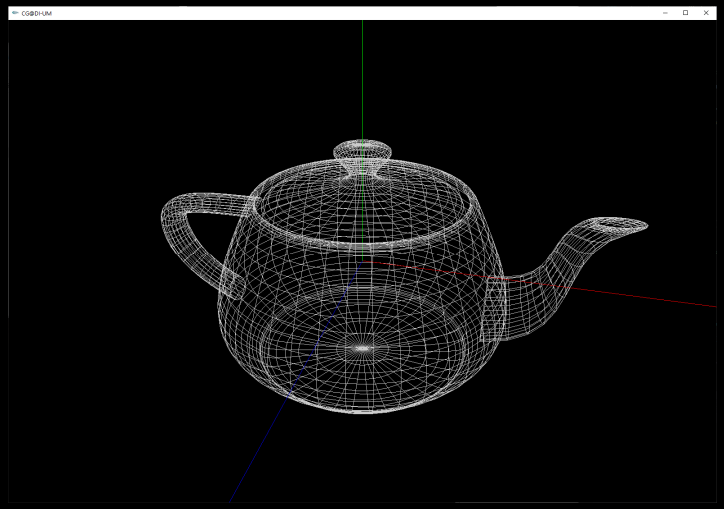
- `glutSolidOctahedron(void);` (8 faces)
- `glutWireOctahedron(void);`

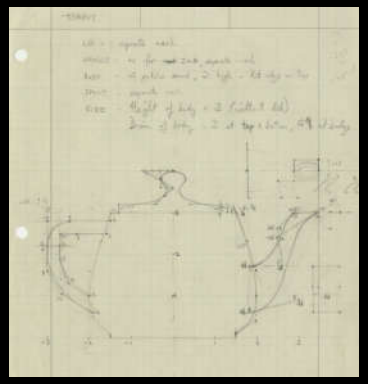
- `glutSolidTetrahedron(void);` (6 faces)
- `glutWireTetrahedron(void);`



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

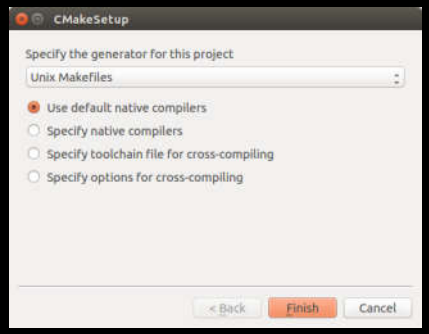
- Install freeglut
  - sudo apt-get install freeglut3-dev

    Note: IF fail to compile freeglut try:
    cd /usr/include/X11/extensions
    sudo ln –s XI.h XInput.h

- Check OpenGL version
  - glxinfo | grep "OpenGL“
  - (sudo apt-get install mesa-utils)

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 apt-get install libxmu-dev libxi-dev`

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

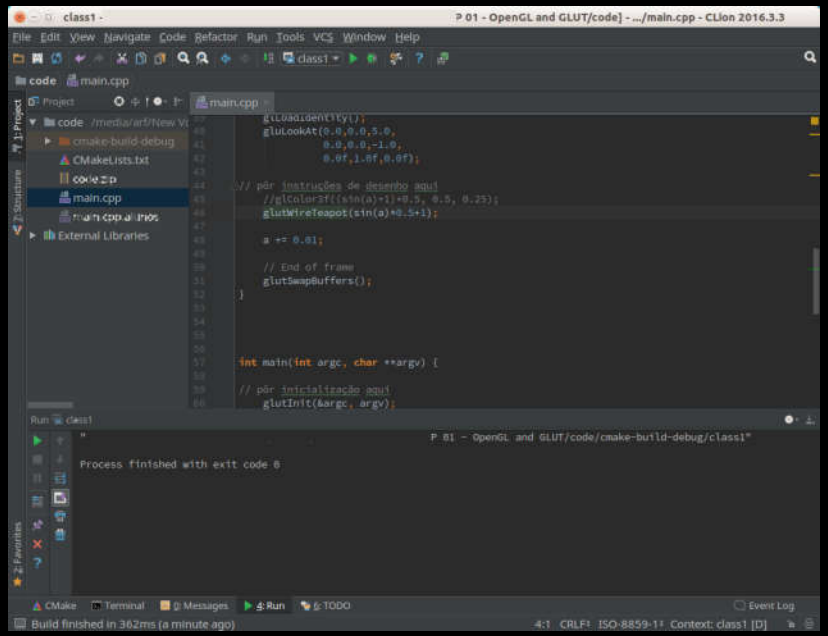
  - Press "Generate"

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.

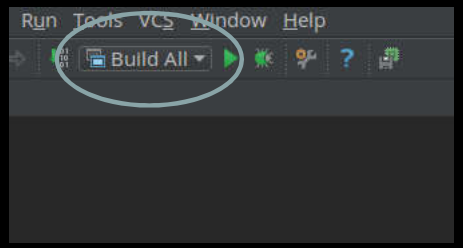
Getting things ready – Linux - CLion

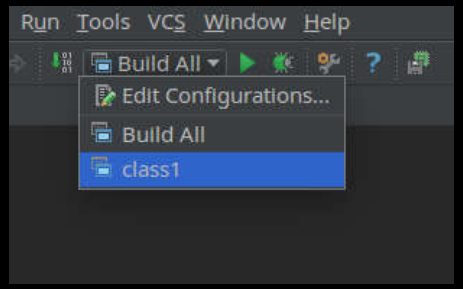
- CLion is a cross platform IDE for C/C++ available for FREE for students
  - https://www.jetbrains.com/student/



Getting things ready – Linux - CLion

- CLion understands CMake files: just open a new project and point to the folder where cMakeLists.txt resides.
  - By default CLion builds all targets
  - press the down arrow and select class1
  - press the green arrow to build and run
  - the “bug” is for debugging ;-)





Getting things ready - Windows

- Download the toolkits folder from the couse page (Conteúdo->Practical Classes) and unzip it to somewhere easily accessible (you'll have to provide its path in CMake.

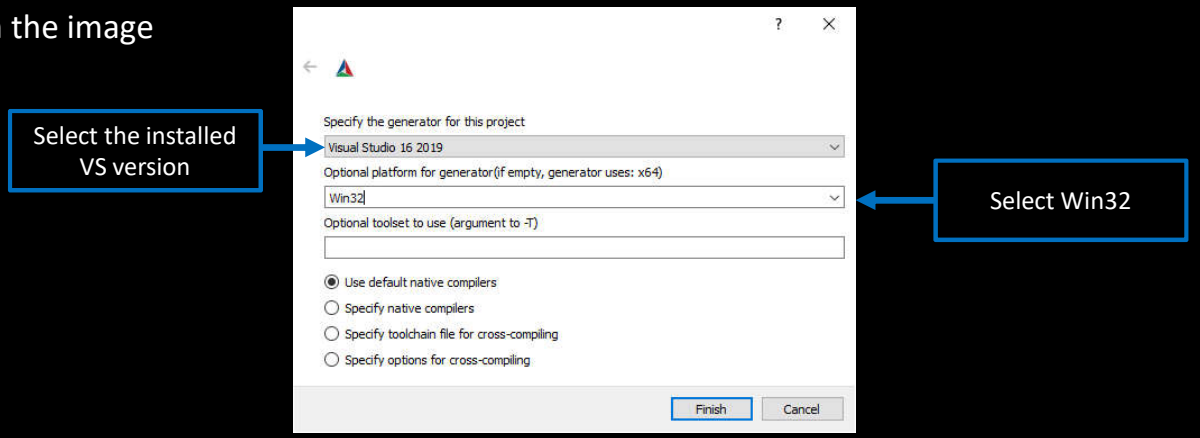
- This folder contains all the APIs that will be used in this course, namely: GLUT, GLEW and DevIL

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 “build”)
  - Press “Configure”

Getting things ready - Windows

- Select the generator as shown in the image
- Press “Finish”

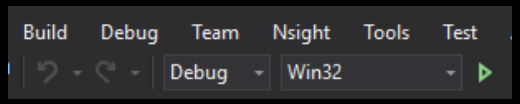


- You’ll get an error – press OK
- Look for the last red line

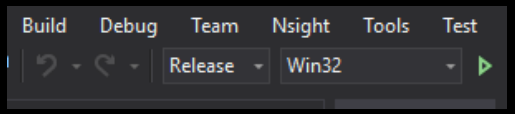
- Press the line and select the folder where toolkits were placed
- Press “Configure” again
  - The bottom window should display “Configuring done”
- Press “Generate” and then press “Open Project”

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



- 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)

Getting things ready - MacOS

- Tips for MacOS, kindly provided by João Luís Martins:
- Download and install CMake (https://cmake.org/files/v3.8/cmake-3.8.0-rc1-Darwin-x86_64.dmg);
- Download and install XQuartz (https://dl.bintray.com/xquartz/downloads/XQuartz-2.7.11.dmg;
-
  - **Note:** Freeglut requires X11 libs. Although X11 is no longer available by default on MacOS, these libs are part of projet XQuartz.
- End session and restart;
- Execute `brew install freeglut` (HomeBrew required);
- Open CMake GUI and follow the Linux instructions starting from “In the CMake window” (slide 28).
- **Note**: These sequence of steps worked on Mac OS X (version 10.12.3)

Getting things ready - MacOS

- Tips for MacOS, kindly provided by Elísio Freitas Fernandes:

- When running Cmake we may get the following error: "xcode-select: error: tool 'xcodebuild' requires Xcode, but active developer directory '/Library/Developer/CommandLineTools' is a command line tools instance".
- The following command fixes this issue:

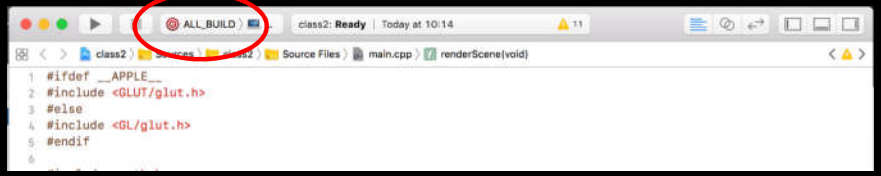
```
sudo xcode-select -s /Applications/Xcode.app/Contents/Developer
```

- tested in version macOS 10.13.3.

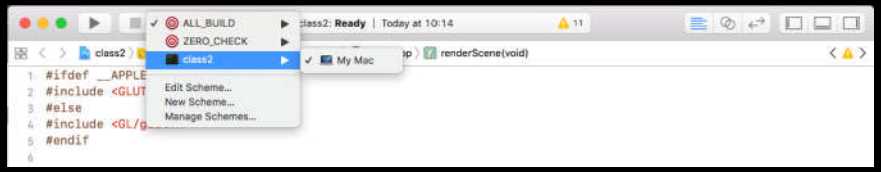
Getting things ready - MacOS

- To compile the project it is necessary to change the predefined target from “ALL_BUILD” to “class*N*”, where N is the class number (images supplied by Elísio Fernandes).

- Click in ALL_BUILD



- Select class*N*



- Click on the arrow to compile