



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 cycle

    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

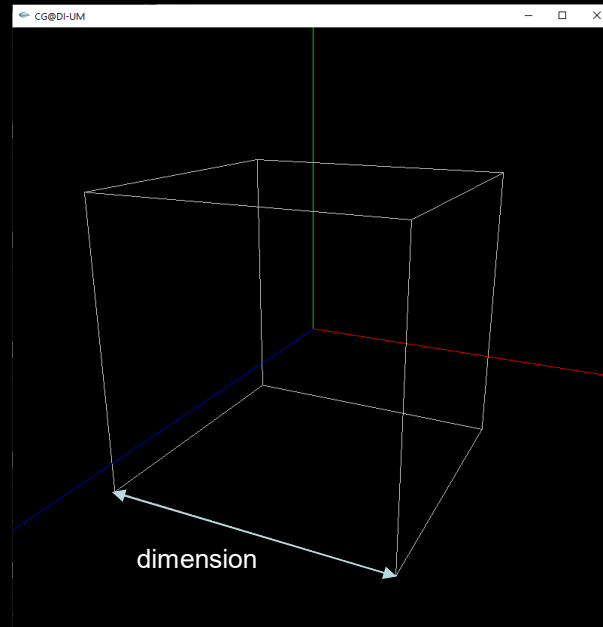
- Display and Idle Func

```
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

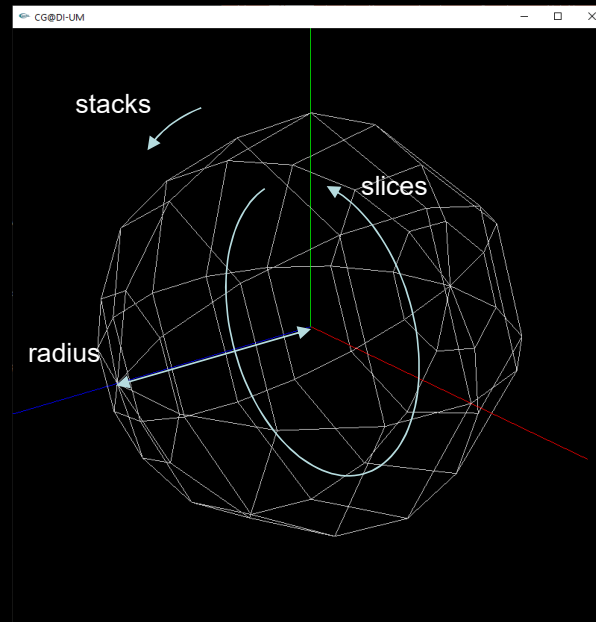
- `glutSolidCube(float dimension);`
- `glutWireCube (float dimension);`





GLUT – Graphical Primitives

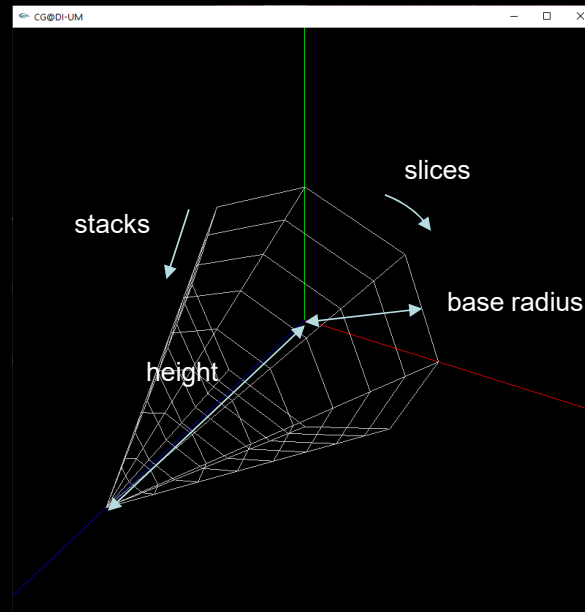
- `glutSolidSphere(float radius, int slices, int stacks);`
- `glutWireSphere (float radius, int slices, int stacks);`





GLUT – Graphical Primitives

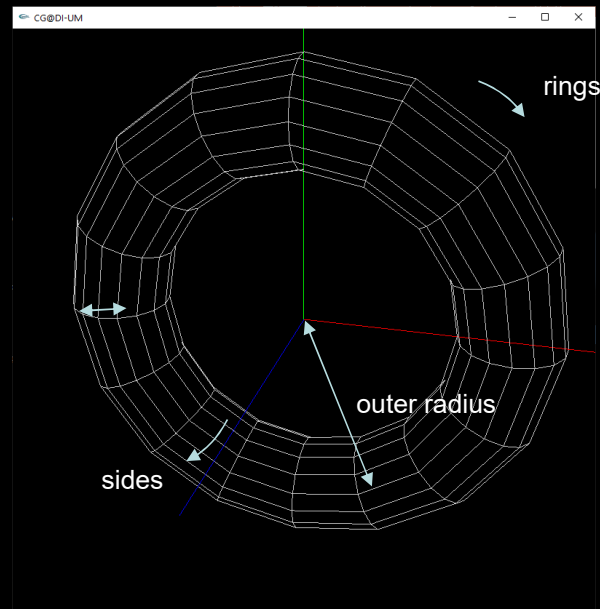
- `glutSolidCone(float baseRadius, float height, int slices, int stacks);`
- `glutWireCone (float baseRadius, float height, int slices, int stacks);`





GLUT - Graphical Primitives

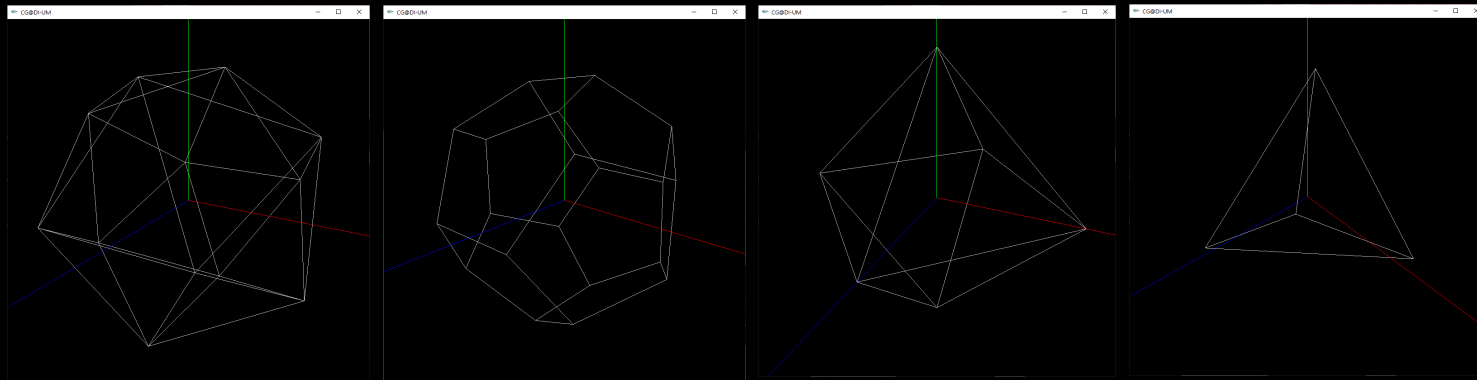
- `glutSolidTorus(float innerRadius, float outerRadius, int sides, int rings);`
- `glutWireTorus(float innerRadius, float outerRadius, int sides, int rings);`





GLUT - Graphical Primitives

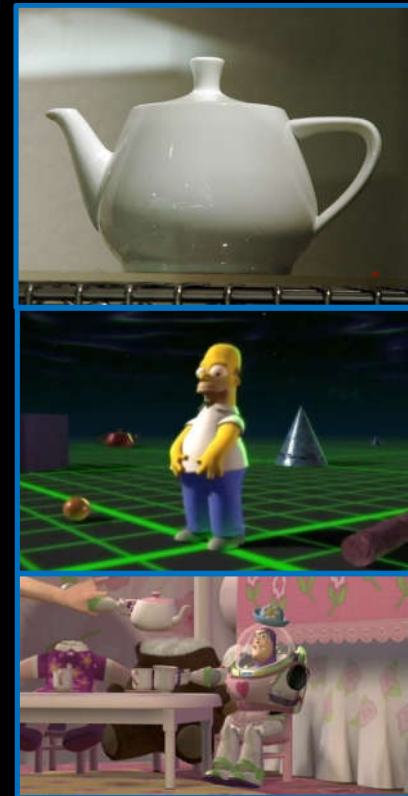
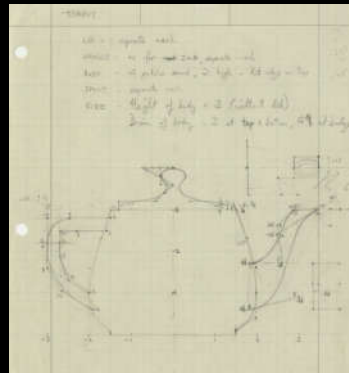
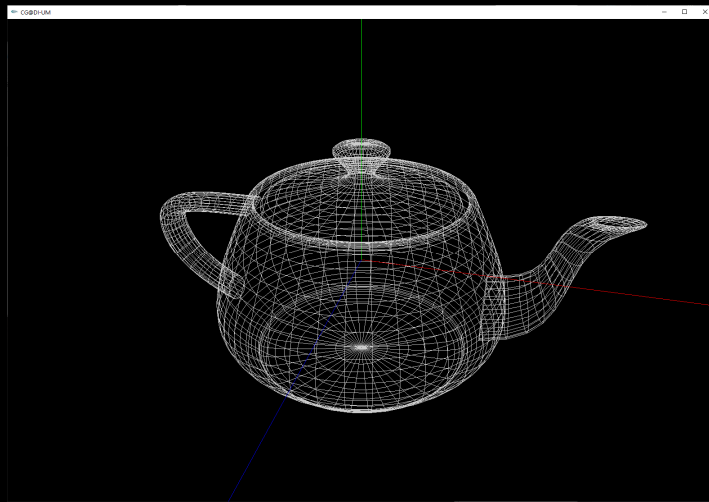
- `glutSolidIcosahedron(void);` (20 faces)
- `glutWireIcosahedron(void);`
- `glutSolidDodecahedron(void);` (12 faces)
- `glutWireDodecahedron(void);`
- `glutSolidOctahedron(void);` (8 faces)
- `glutWireOctahedron(void);`
- `glutSolidTetrahedron(void);` (6 faces)
- `glutWireTetrahedron(void);`





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.



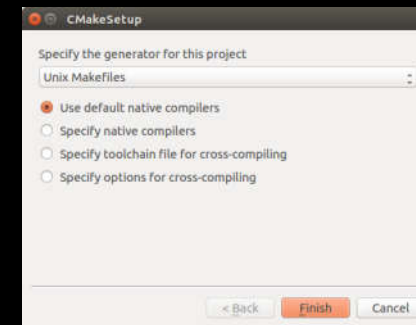
Getting things ready – Linux (Ubuntu)

- 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 X11.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
```

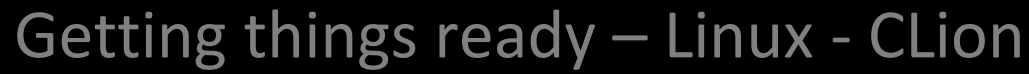
And press “Configure” again (this time there should be no errors)

- 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.



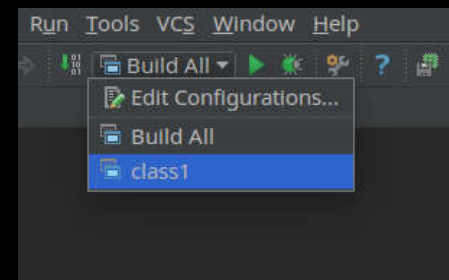
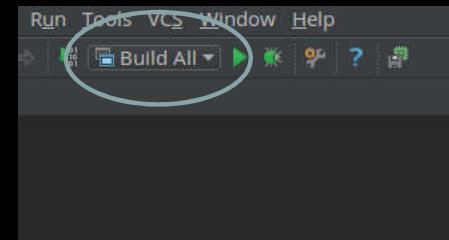
-
- The screenshot displays the CMake GUI interface. The top bar shows the project name "class1" and the current configuration "P 01 - OpenGL and GLUT/code/". The left sidebar contains a "Project" view showing the file structure, including "cmake-build-debug", "CMakeLists.txt", "code", "main.cpp", and "main.cpp.all.h". The central editor shows the contents of "main.cpp", which includes OpenGL and GLUT headers, window creation, and a simple sine wave plot. The bottom status bar indicates "Build finished in 362ms (a minute ago)".
- File Edit View Navigate Code Refactor Run Tools VCS Window Help
- code main.cpp
- Project
- code /media/art/View V
- cmake-build-debug
- CMakeLists.txt
- code
- main.cpp
- main.cpp.all.h
- External Libraries
- ```

1 // CMakeLists.txt
2
3 # CMakeLists.txt
4
5 # CMakeLists.txt
6
7 # CMakeLists.txt
8
9 # CMakeLists.txt
10
11 # CMakeLists.txt
12
13 # CMakeLists.txt
14
15 # CMakeLists.txt
16
17 # CMakeLists.txt
18
19 # CMakeLists.txt
20
21 # CMakeLists.txt
22
23 # CMakeLists.txt
24
25 # CMakeLists.txt
26
27 # CMakeLists.txt
28
29 # CMakeLists.txt
30
31 # CMakeLists.txt
32
33 # CMakeLists.txt
34
35 # CMakeLists.txt
36
37 # CMakeLists.txt
38
39 # CMakeLists.txt
40
41 # CMakeLists.txt
42
43 # CMakeLists.txt
44
45 # CMakeLists.txt
46
47 # CMakeLists.txt
48
49 # CMakeLists.txt
50
51 # CMakeLists.txt
52
53 # CMakeLists.txt
54
55 # CMakeLists.txt
56
57 # CMakeLists.txt
58
59 # CMakeLists.txt
60
61 # CMakeLists.txt
62
63 # CMakeLists.txt
64
65 # CMakeLists.txt
66
67 # CMakeLists.txt
68
69 # CMakeLists.txt
70
71 # CMakeLists.txt
72
73 # CMakeLists.txt
74
75 # CMakeLists.txt
76
77 # CMakeLists.txt
78
79 # CMakeLists.txt
80
81 # CMakeLists.txt
82
83 # CMakeLists.txt
84
85 # CMakeLists.txt
86
87 # CMakeLists.txt
88
89 # CMakeLists.txt
90
91 # CMakeLists.txt
92
93 # CMakeLists.txt
94
95 # CMakeLists.txt
96
97 # CMakeLists.txt
98
99 # CMakeLists.txt
100
101 # CMakeLists.txt
102
103 # CMakeLists.txt
104
105 # CMakeLists.txt
106
107 # CMakeLists.txt
108
109 # CMakeLists.txt
110
111 # CMakeLists.txt
112
113 # CMakeLists.txt
114
115 # CMakeLists.txt
116
117 # CMakeLists.txt
118
119 # CMakeLists.txt
120
121 # CMakeLists.txt
122
123 # CMakeLists.txt
124
125 # CMakeLists.txt
126
127 # CMakeLists.txt
128
129 # CMakeLists.txt
130
131 # CMakeLists.txt
132
133 # CMakeLists.txt
134
135 # CMakeLists.txt
136
137 # CMakeLists.txt
138
139 # CMakeLists.txt
140
141 # CMakeLists.txt
142
143 # CMakeLists.txt
144
145 # CMakeLists.txt
146
147 # CMakeLists.txt
148
149 # CMakeLists.txt
150
151 # CMakeLists.txt
152
153 # CMakeLists.txt
154
155 # CMakeLists.txt
156
157 # CMakeLists.txt
158
159 # CMakeLists.txt
160
161 # CMakeLists.txt
162
163 # CMakeLists.txt
164
165 # CMakeLists.txt
166
167 # CMakeLists.txt
168
169 # CMakeLists.txt
170
171 # CMakeLists.txt
172
173 # CMakeLists.txt
174
175 # CMakeLists.txt
176
177 # CMakeLists.txt
178
179 # CMakeLists.txt
180
181 # CMakeLists.txt
182
183 # CMakeLists.txt
184
185 # CMakeLists.txt
186
187 # CMakeLists.txt
188
189 # CMakeLists.txt
190
191 # CMakeLists.txt
192
193 # CMakeLists.txt
194
195 # CMakeLists.txt
196
197 # CMakeLists.txt
198
199 # CMakeLists.txt
200
201 # CMakeLists.txt
202
203 # CMakeLists.txt
204
205 # CMakeLists.txt
206
207 # CMakeLists.txt
208
209 # CMakeLists.txt
210
211 # CMakeLists.txt
212
213 # CMakeLists.txt
214
215 # CMakeLists.txt
216
217 # CMakeLists.txt
218
219 # CMakeLists.txt
220
221 # CMakeLists.txt
222
223 # CMakeLists.txt
224
225 # CMakeLists.txt
226
227 # CMakeLists.txt
228
229 # CMakeLists.txt
230
231 # CMakeLists.txt
232
233 # CMakeLists.txt
234
235 # CMakeLists.txt
236
237 # CMakeLists.txt
238
239 # CMakeLists.txt
240
241 # CMakeLists.txt
242
243 # CMakeLists.txt
244
245 # CMakeLists.txt
246
247 # CMakeLists.txt
248
249 # CMakeLists.txt
250
251 # CMakeLists.txt
252
253 # CMakeLists.txt
254
255 # CMakeLists.txt
256
257 # CMakeLists.txt
258
259 # CMakeLists.txt
260
261 # CMakeLists.txt
262
263 # CMakeLists.txt
264
265 # CMakeLists.txt
266
267 # CMakeLists.txt
268
269 # CMakeLists.txt
270
271 # CMakeLists.txt
272
273 # CMakeLists.txt
274
275 # CMakeLists.txt
276
277 # CMakeLists.txt
278
279 # CMakeLists.txt
280
281 # CMakeLists.txt
282
283 # CMakeLists.txt
284
285 # CMakeLists.txt
286
287 # CMakeLists.txt
288
289 # CMakeLists.txt
290
291 # CMakeLists.txt
292
293 # CMakeLists.txt
294
295 # CMakeLists.txt
296
297 # CMakeLists.txt
298
299 # CMakeLists.txt
300
301 # CMakeLists.txt
302
303 # CMakeLists.txt
304
305 # CMakeLists.txt
306
307 # CMakeLists.txt
308
309 # CMakeLists.txt
310
311 # CMakeLists.txt
312
313 # CMakeLists.txt
314
315 # CMakeLists.txt
316
317 # CMakeLists.txt
318
319 # CMakeLists.txt
320
321 # CMakeLists.txt
322
323 # CMakeLists.txt
324
325 # CMakeLists.txt
326
327 # CMakeLists.txt
328
329 # CMakeLists.txt
330
331 # CMakeLists.txt
332
333 # CMakeLists.txt
334
335 # CMakeLists.txt
336
337 # CMakeLists.txt
338
339 # CMakeLists.txt
340
341 # CMakeLists.txt
342
343 # CMakeLists.txt
344
345 # CMakeLists.txt
346
347 # CMakeLists.txt
348
349 # CMakeLists.txt
350
351 # CMakeLists.txt
352
353 # CMakeLists.txt
354
355 # CMakeLists.txt
356
357 # CMakeLists.txt
358
359 # CMakeLists.txt
360
361 # CMakeLists.txt
362
363 # CMakeLists.txt
364
365 # CMakeLists.txt
366
367 # CMakeLists.txt
368
369 # CMakeLists.txt
370
371 # CMakeLists.txt
372
373 # CMakeLists.txt
374
375 # CMakeLists.txt
376
377 # CMakeLists.txt
378
379 # CMakeLists.txt
380
381 # CMakeLists.txt
382
383 # CMakeLists.txt
384
385 # CMakeLists.txt
386
387 # CMakeLists.txt
388
389 # CMakeLists.txt
390
391 # CMakeLists.txt
392
393 # CMakeLists.txt
394
395 # CMakeLists.txt
396
397 # CMakeLists.txt
398
399 # CMakeLists.txt
400
401 # CMakeLists.txt
402
403 # CMakeLists.txt
404
405 # CMakeLists.txt
406
407 # CMakeLists.txt
408
409 # CMakeLists.txt
410
411 # CMakeLists.txt
412
413 # CMakeLists.txt
414
415 # CMakeLists.txt
416
417 # CMakeLists.txt
418
419 # CMakeLists.txt
420
421 # CMakeLists.txt
422
423 # CMakeLists.txt
424
425 # CMakeLists.txt
426
427 # CMakeLists.txt
428
429 # CMakeLists.txt
430
431 # CMakeLists.txt
432
433 # CMakeLists.txt
434
435 # CMakeLists.txt
436
437 # CMakeLists.txt
438
439 # CMakeLists.txt
440
441 # CMakeLists.txt
442
443 # CMakeLists.txt
444
445 # CMakeLists.txt
446
447 # CMakeLists.txt
448
449 # CMakeLists.txt
450
451 # CMakeLists.txt
452
453 # CMakeLists.txt
454
455 # CMakeLists.txt
456
457 # CMakeLists.txt
458
459 # CMakeLists.txt
460
461 # CMakeLists.txt
462
463 # CMakeLists.txt
464
465 # CMakeLists.txt
466
467 # CMakeLists.txt
468
469 # CMakeLists.txt
470
471 # CMakeLists.txt
472
473 # CMakeLists.txt
474
475 # CMakeLists.txt
476
477 # CMakeLists.txt
478
479 # CMakeLists.txt
480
481 # CMakeLists.txt
482
483 # CMakeLists.txt
484
485 # CMakeLists.txt
486
487 # CMakeLists.txt
488
489 # CMakeLists.txt
490
491 # CMakeLists.txt
492
493 # CMakeLists.txt
494
495 # CMakeLists.txt
496
497 # CMakeLists.txt
498
499 # CMakeLists.txt
500
501 # CMakeLists.txt
502
503 # CMakeLists.txt
504
505 # CMakeLists.txt
506
507 # CMakeLists.txt
508
509 # CMakeLists.txt
510
511 # CMakeLists.txt
512
513 # CMakeLists.txt
514
515 # CMakeLists.txt
516
517 # CMakeLists.txt
518
519 # CMakeLists.txt
520
521 # CMakeLists.txt
522
523 # CMakeLists.txt
524
525 # CMakeLists.txt
526
527 # CMakeLists.txt
528
529 # CMakeLists.txt
530
531 # CMakeLists.txt
532
533 # CMakeLists.txt
534
535 # CMakeLists.txt
536
537 # CMakeLists.txt
538
539 # CMakeLists.txt
540
541 # CMakeLists.txt
542
543 # CMakeLists.txt
544
545 # CMakeLists.txt
546
547 # CMakeLists.txt
548
549 # CMakeLists.txt
550
551 # CMakeLists.txt
552
553 # CMakeLists.txt
554
555 # CMakeLists.txt
556
557 # CMakeLists.txt
558
559 # CMakeLists.txt
560
561 # CMakeLists.txt
562
563 # CMakeLists.txt
564
565 # CMakeLists.txt
566
567 # CMakeLists.txt
568
569 # CMakeLists.txt
570
571 # CMakeLists.txt
572
573 # CMakeLists.txt
574
575 # CMakeLists.txt
576
577 # CMakeLists.txt
578
579 # CMakeLists.txt
580
581 # CMakeLists.txt
582
583 # CMakeLists.txt
584
585 # CMakeLists.txt
586
587 # CMakeLists.txt
588
589 # CMakeLists.txt
590
591 # CMakeLists.txt
592
593 # CMakeLists.txt
594
595 # CMakeLists.txt
596
597 # CMakeLists.txt
598
599 # CMakeLists.txt
600
601 # CMakeLists.txt
602
603 #
```



## 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 course 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”

Select the installed  
VS version

Specify the generator for this project

Visual Studio 16 2019

Optional platform for generator (if empty, generator uses: x64)

Win32

Optional toolset to use (argument to -T)

☒ Use default native compilers  
☐ Specify native compilers  
☐ Specify toolchain file for cross-compiling  
☐ Specify options for cross-compiling

Finish Cancel

Select Win32

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

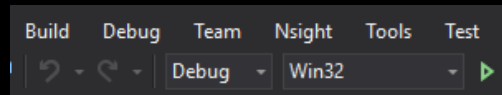
TOOLKITS\_FOLDER

- 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”

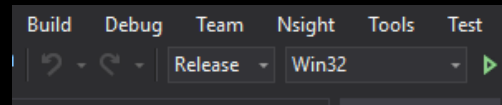


## Getting things ready - Windows

- 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](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
```

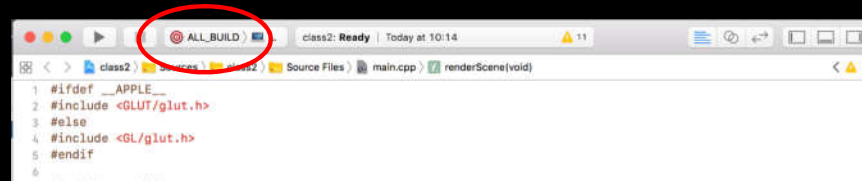
- tested in version macOS 10.13.3.



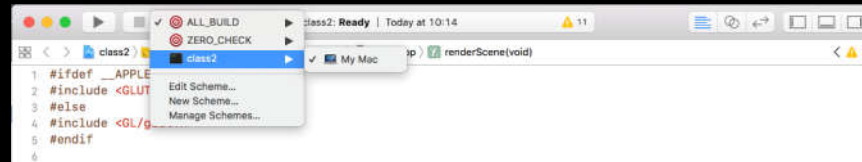
## Getting things ready - MacOS

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

- Click in ALL\_BUILD



- Select classN



- Click on the arrow to compile

