



Introduction to OpenGL

ENSIM, 5A info – IPS

Academic Year 2022-2023

What is OpenGL?

- Cross-language, cross-platform application programming interface (API) for rendering 2D/3D graphics
- Originally released by Silicon Graphics Inc. (SGI) in 1992
- Currently managed by non-profit technology consortium Khronos Group (<https://www.khronos.org/>)
- The OpenGL API can directly interact with the hardware of the system, specifically the graphics processing unit (GPU) in order to achieve hardware-accelerated rendering

What is OpenGL?

- OpenGL consists of about 150 distinct commands used to specify the objects and operations needed to produce interactive three-dimensional applications
- It doesn't provide high-level commands for describing models of three-dimensional objects (cars, parts of the body, airplanes, or molecules)
- The desired model should be built up from a small set of geometric primitives points, lines, and polygons

What is OpenGL?

- Sophisticated libraries may be built on top of OpenGL to provide more complicated features
- The OpenGL Utility Library (GLU) provides many of the modeling features, such as quadric surfaces and NURBS curves and surfaces
- GLU is a standard part of every OpenGL implementation

What is OpenGL?

- The OpenGL Utility Toolkit (GLUT) is a window system independent toolkit for writing OpenGL programs
- It performs system-level I/O with the host operating system
- It allows the creation of portable code between operating systems
- GLUT is a useful starting point for learning OpenGL and performing easily relatively complex tasks (opening windows, detecting input)
- It allows to create some complicated three-dimensional objects such as a sphere, a torus, and a teapot




Where OpenGL is used?

- Computer-aided design (CAD)
- 3D Modelling
- Virtual reality
- Medical imaging
- Scientific visualization
- Flight simulation
- Video games
- **.....In short, anything that needs to draw graphics**

Color formats in OpenGL

- Red, Green and Blue components of the **RGB** color format belong to the interval $[0, 1]$
- Colors spread out from Black (0.0, 0.0, 0.0) to White (1.0, 1.0, 1.0)
- A fourth component is added to R, G and B and is noted A (alpha). The obtained color format is denoted **RGBA**
- Alpha component is generally used by OpenGL in Blending process to implement transparency within objects
- The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (fully opaque).

Color formats in OpenGL

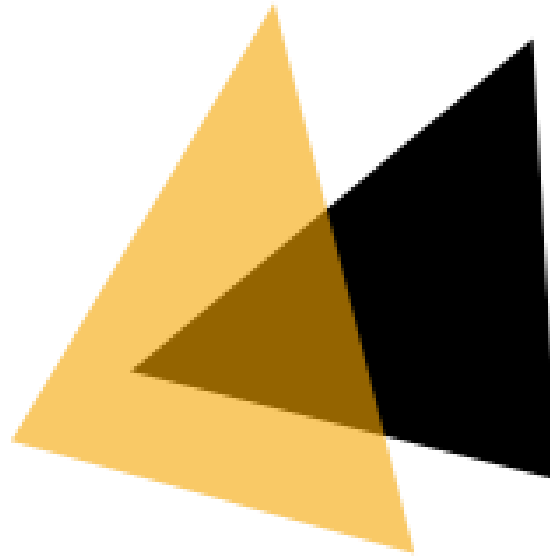
		R	G	B
<i>red</i>		1	0	0
<i>orange</i>		1	0.5	0
<i>yellow</i>		1	1	0
<i>green</i>		0	1	0
<i>blue</i>		0	0	1
<i>violet</i>		0.5	0	1

		R	G	B
<i>black</i>		0	0	0
<i>white</i>		1	1	1
<i>gray</i>		0.5	0.5	0.5
<i>brown</i>		0.5	0.2	0
<i>pink</i>		1	0.5	0.5
<i>cyan</i>		0	1	1

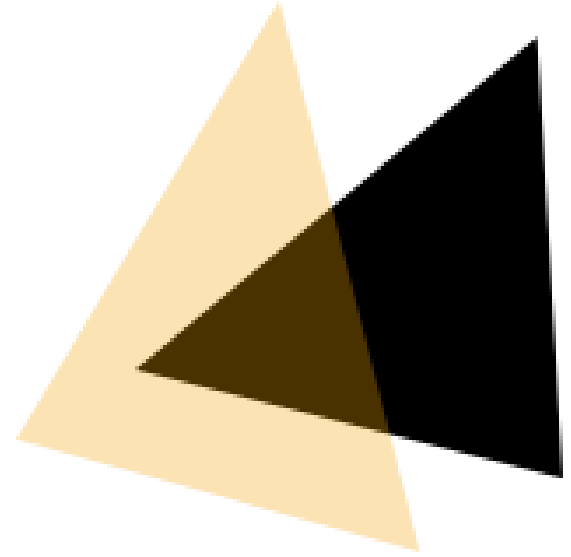
Color formats in OpenGL



Alpha = 1



Alpha = 0.6

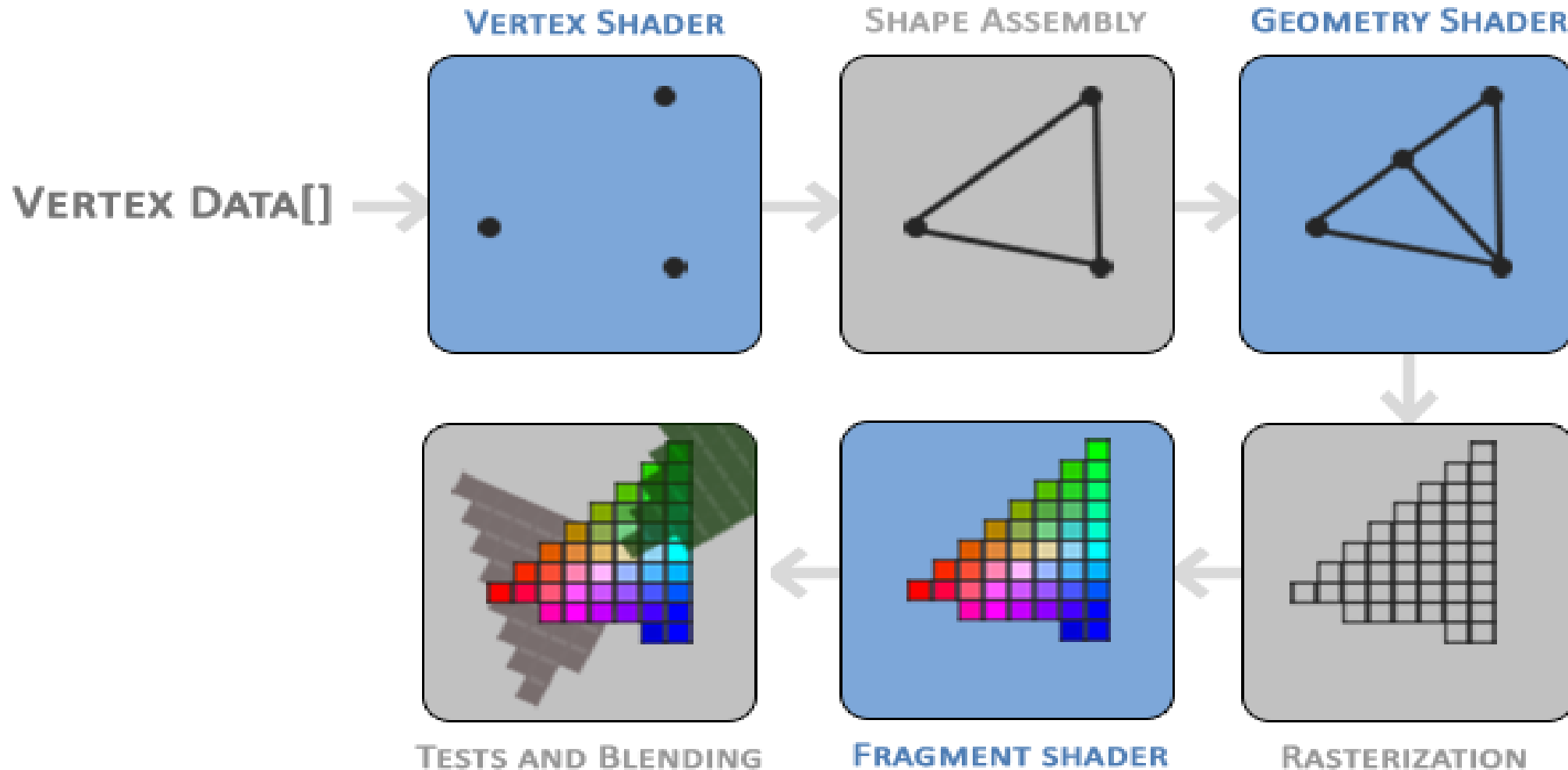


Alpha = 0.3

The graphics pipeline of OpenGL

- A large part of OpenGL's work is to transform all 3D coordinates to 2D pixels that fit on the screen.
- The graphics pipeline can be divided into several steps where each step requires the output of the previous step as its input
- Each of these steps is highly specialized (i.e. has one specific function) and can easily be executed in parallel
- The processing cores run small programs on the GPU for each step of the pipeline (called shaders)

The graphics pipeline of OpenGL

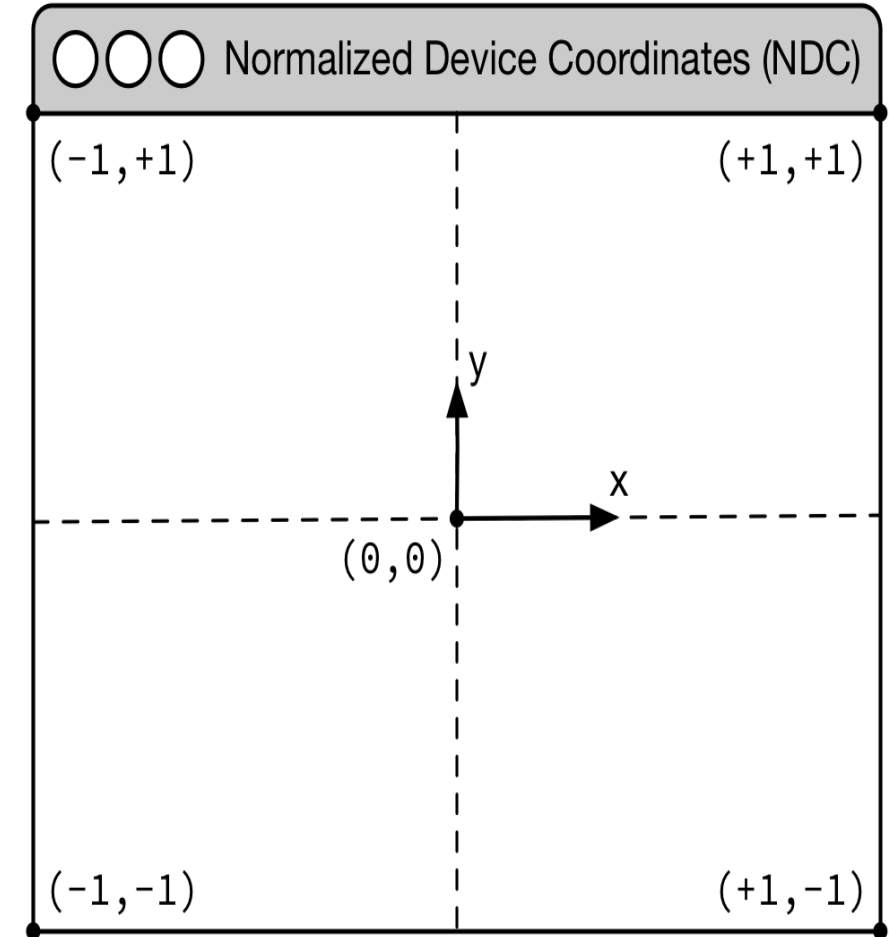


OpenGL syntax notation

- OpenGL **commands** use the prefix “gl” and initial capital letters for each word making up the command name (e.g. *glClearColor*, *glBegin*, *glFlush*)
- OpenGL defined **constants** begin with “GL_”, use all capital letters, and use underscores to separate words (e.g. GL_COLOR_BUFFER_BIT, GL_LINE_LOOP, GL_TRIANGLE_STRIP)
- Sometimes, numbers and letters are appended to some command names (e.g. *glVertex2f*, *glVertex2i*, *glUniform3f*, *glColor4f*)
 - The numerical part (2, 3, 4, ...) indicates the number of arguments
 - The letter part (b, i, f, d, ...) indicates the type of the arguments

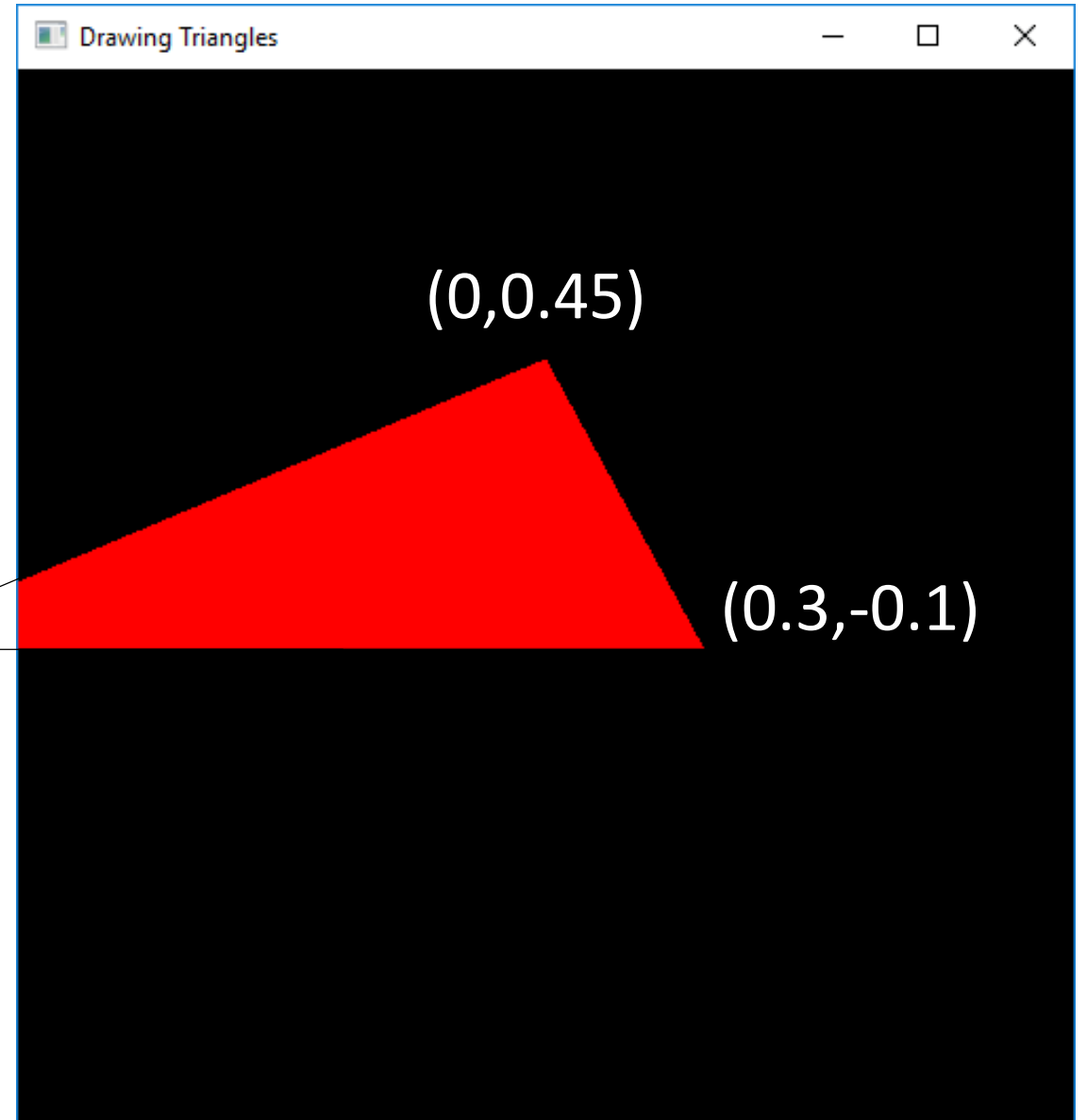
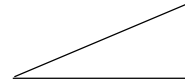
Normalized Device Coordinates – NDC

- All the vertices, that we want to make visible, should be in normalized device coordinates after each vertex shader run, i.e., the x, y and z coordinates of each vertex should be between -1.0 and 1.0; coordinates outside this range will not be visible (clipped)
- For example, in the 2D x-y plane, the lower left corner corresponds to (-1.0, -1.0), the upper right corner corresponds to (1.0, 1.0) and the center of the window corresponds to (0.0, 0.0).



Clipped triangle

$(-1.3, -0.1)$



pyOpenGL

- OpenGL is a graphics API and not a platform of its own => it requires a language to operate with
- There are many bindings from programming languages to OpenGL and related APIs (C++, Java, C#, Python, ...)
- The binding we are going to use is “pyOpenGL” and is intended to use OpenGL APIs from within Python code
- PyOpenGL is a cross-platform open source Python binding to the standard OpenGL API providing 2-D and 3-D graphic drawing
- PyOpenGL supports the GL, GLU, and GLUT libraries

Python Integrated Development Environment - IDE

- An Integrated Development Environment (IDE) is a software application that provides comprehensive facilities to computer programmers for software development
- An IDE normally consists of a source code editor, interpretation and/or compiling/building automation tools and a debugger
- Many modern IDEs also have a class browser, an object browser, and a class hierarchy diagram for use in object-oriented software development
- Examples of Python IDEs: Sublime Text, PyCharm, Atom and Spyder

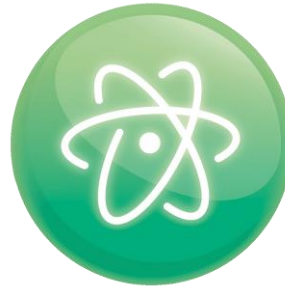
Python Integrated Development Environment - IDE



Sublime Text



PyCharm



Atom



Spyder (Anaconda)

Packages to install for OpenGL

- The minimum of packages needed to run pyOpenGL under Python are: PyOpenGL and PyOpenGL_accelerate
- Other packages may be needed depending on the target application (numpy, scipy, pygame, etc.)
- Install: `pip install PyOpenGL PyOpenGL_accelerate`
or
- Download the corresponding wheel packages from the repository
<https://www.lfd.uci.edu/~gohlke/pythonlibs/>

Packages to install for OpenGL

- Be careful to download the packages corresponding to your hardware (CPU), and to the current version of Python installed on your machine
- Run `python --version` to know your version of Python
- For example, if you have a windows 64-bit machine with Python 3.9 installed on it, you should download and install the packages
`PyOpenGL-3.1.5-cp39-cp39-win_amd64.whl` and
`PyOpenGL_accelerate-3.1.5-cp39-cp39-win_amd64.whl`
`glumpy-1.2.0-cp39-cp39-win_amd64.whl`

How to install PyOpenGL downloaded packages?

C:\Windows\system32\cmd.exe

```
Microsoft Windows [Version 10.0.14393]  
(c) 2016 Microsoft Corporation. All rights reserved.
```

```
C:\Users\User>cd Downloads
```

```
C:\Users\User\Downloads>python -m pip install PyOpenGL-3.1.5-cp39-cp39-win_amd64.whl
```

```
C:\Users\User\Downloads>python -m pip install PyOpenGL_accelerate-3.1.5-cp39-cp39-win_amd64.whl
```

```
C:\Users\User\Downloads>python -m pip install glumpy-1.2.0-cp39-cp39-win_amd64.whl
```

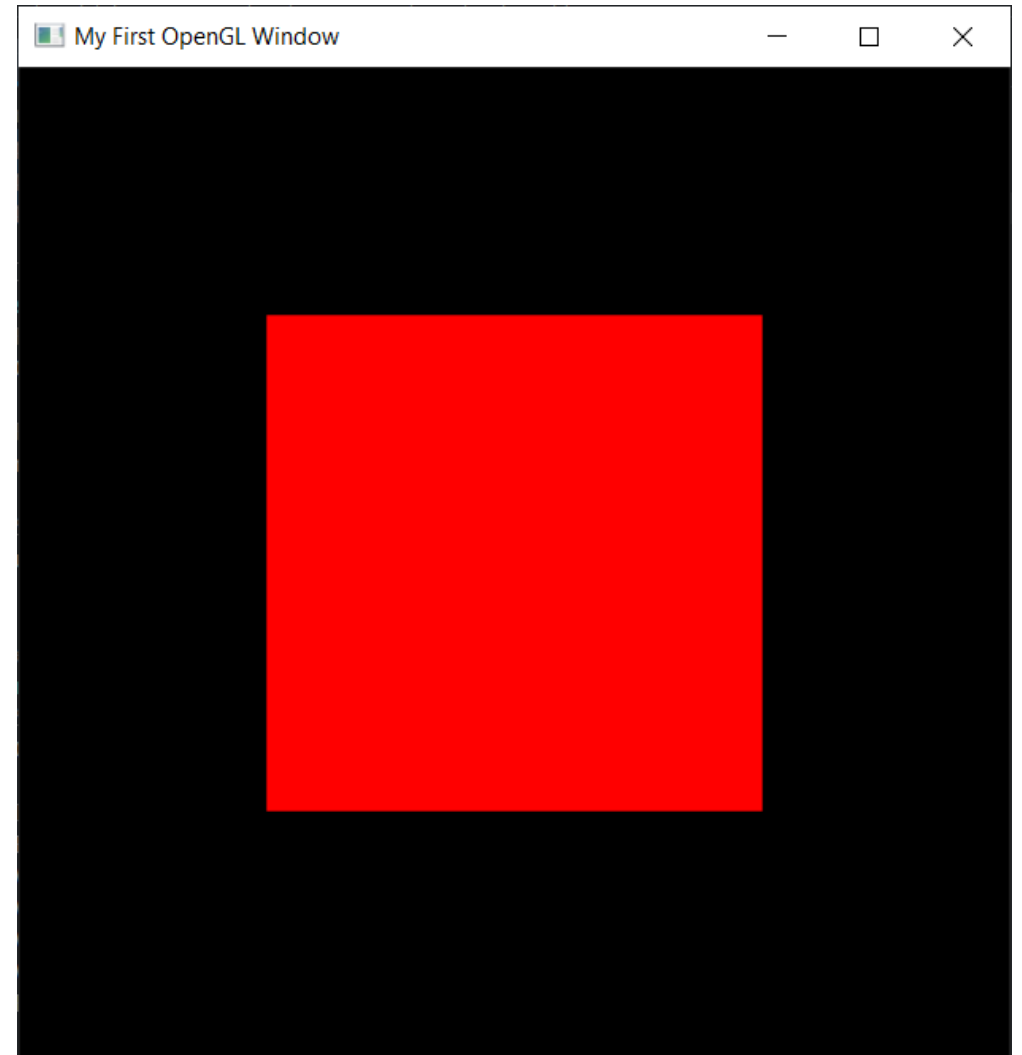
How to test that everything is OK?

```
from OpenGL.GL import *
```

```
from OpenGL.GLUT import *
```

```
from OpenGL.GLU import *
```

Open the file `template.py` and run it



How will we work in the lab sessions?

- I will explain some commands/functionalities in OpenGL
- You will directly apply these commands/functionalities
- I will give you exercises to practice during the lab sessions
- I will give you homework to see what you learnt