Advanced Computer Graphics Lab 1

Asist. Drd. Ing. Iulia Stănică
Lab Assistant: Alberta Milicu – message me on Teams
iulia.stanica@gmail.com

ACG Grading Scheme

- Laboratory: 60%
  - homework 30%
    - 2 assignments
  - activity + presence at laboratory 30%
- Final exam: 40%
- Bonus: Up to 10% (big hw extra tasks, course activity)
- Re-examination: only final exam

- Requirements:
  - Minimum 4.5 at the final exam
  - Minimum 4.5 in total

ACG Grading Scheme

- The two big assignments should be uploaded on Moodle and presented during the labs (no delay accepted)
- You can work in teams of 2-3 people for the 2 Big Hws
- Deadline & presentation hw1 approx. Week 8
- Deadline & presentation hw2 Week14

ACG Grading Scheme

- You will have to upload on Moodle your lab solution at the end of each lab! Also, the lab solution should be accompanied with a screenshot of your app running (with some personalized feature of your choice – e.g. changed color, different size, different screen parameters etc.)
- Each week you will get a grade between 0 and 10, which you will check individually on Moodle.
- All labs will also be put on moodle, with demo / starting points.

Questions

- For any question, you can contact me on Teams or by email:

  iulia.stanica@gmail.com

What will we use?

- C++ and OpenGL (or you can use Java & OpenGL, but all my materials will be in C++)
- Visual Studio
- Some Unity3D at the end of the semester
- Math! Computer graphics is all about math!

What will we learn?

- Render (and loading) 2D and 3D objects
- Using and customizing the camera in a 3D scene
- Using shaders (vertex and fragment)
- Illumination models
- Textures
- Creating our own game engine
- …and more

Today’s roadmap

- Visual Studio presentation & configuration
- OpenGL basics
- Creating an OpenGL window

Computer Graphics

- A domain of Computer Science
- Manipulation of visual and geometric information using computational techniques
- Term includes both 2D and 3D computer graphics
- Knowledge needed: math, physics, algorithms, UI design etc.

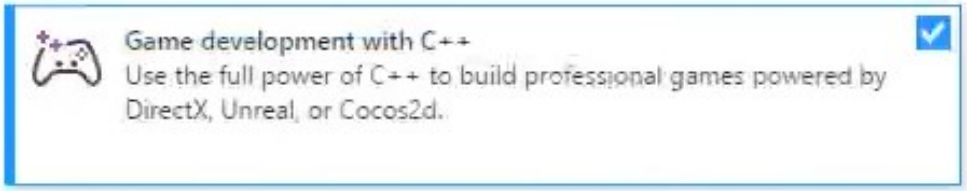
IDE used

- Visual Studio (Community Edition) 2019 (or any other version should be fine too)
- https://visualstudio.microsoft.com/downloads/

- Obs: Visual Studio Code requires additional configuration for using correctly the C++ compiler with OpenGL, so it is not recommended.

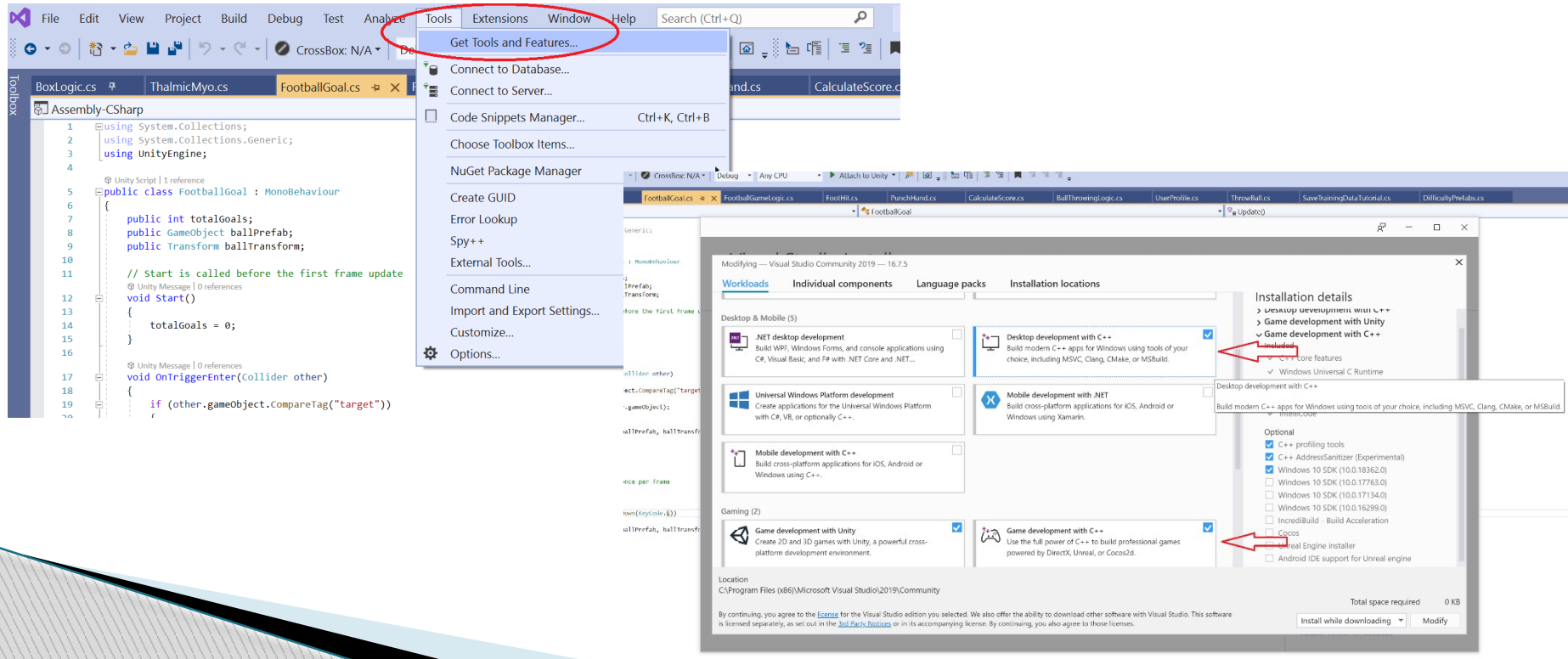
IDE used

- !!! Make sure that the C++ package is included when you install VS (usually, it is not included by default)
- For 2019, choose both Desktop development with C++ and Game development with C++



IDE used

- If you already had VS installed (for C# for instance), you can install the additional C++ plugins by accessing Tools -> Get Tools and Features



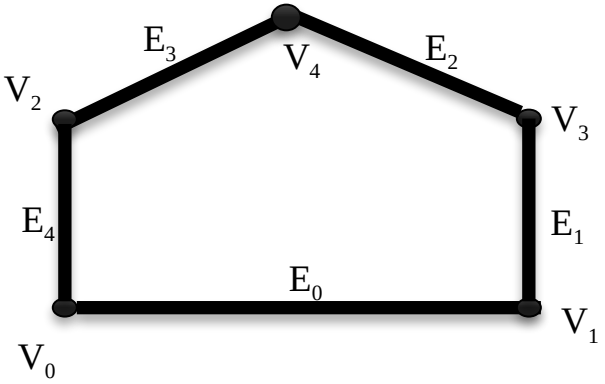
OpenGL

- Graphical Libraries, hardware independent
- Characteristics:
  - Draw objects
  - View transform
  - Colors, illumination
  - Texture mapping
  - Clipping and culling (to eliminate non-visible parts)
- Very helpful for learning the basics principles of computer graphics programming
- We can even build our own game engine

Vertices and edges

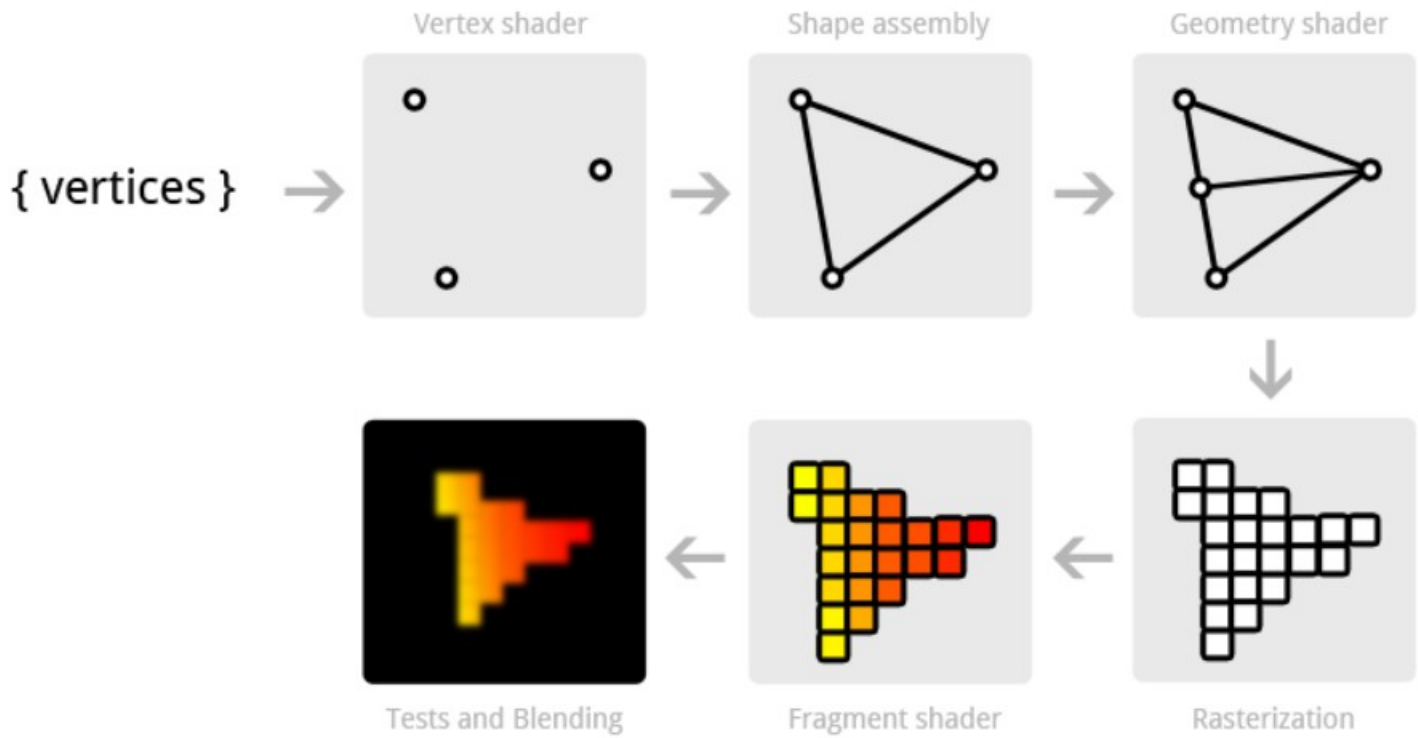
Vertices	
0	(0, 0)
1	(2, 0)
2	(0, 1)
3	(2, 1)
4	(1, 1.5)

Edges	
0	(0, 1)
1	(1, 3)
2	(3, 4)
3	(4, 2)
4	(2, 0)



! In OpenGL, all coordinates are normalized (between [-1 and 1]

Graphics pipeline (more about this at the course)

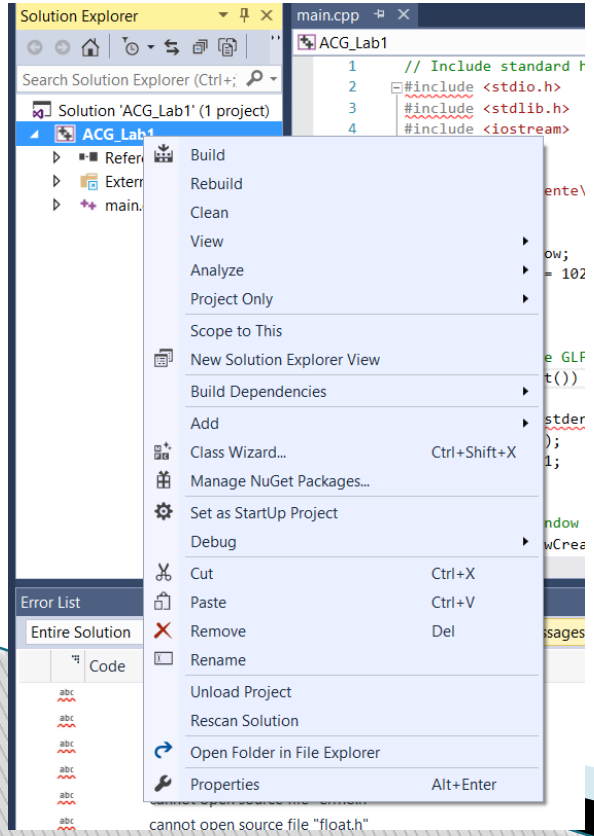


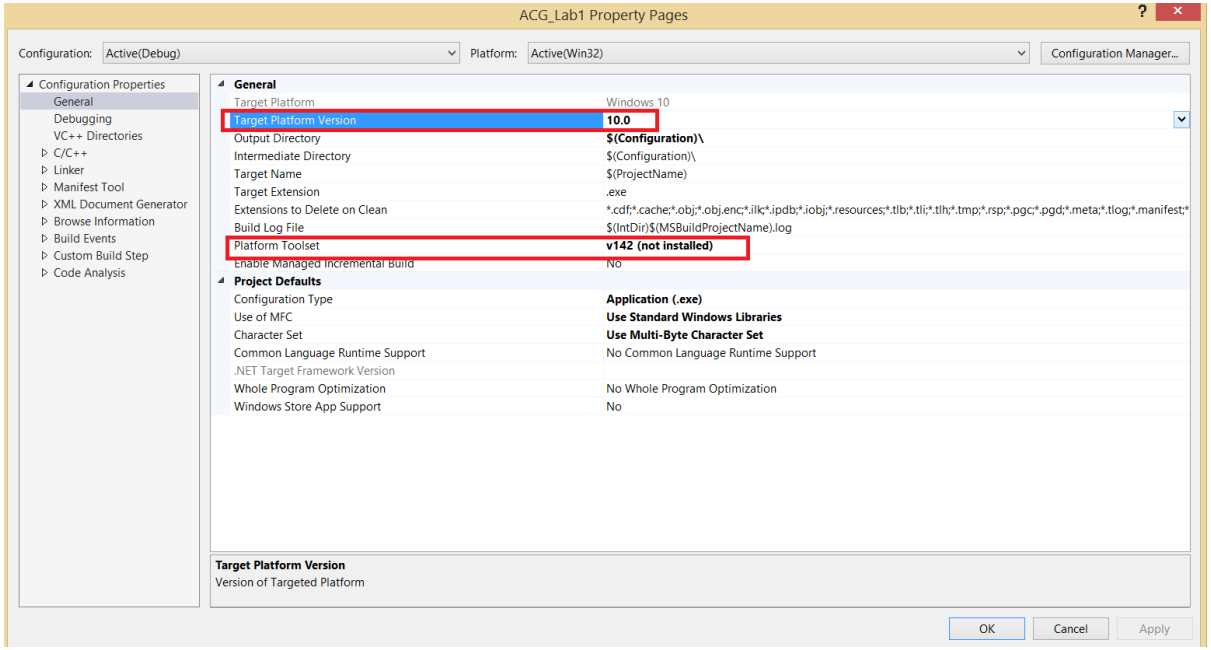
Our First OpenGL program

- Each week, you will have a “starting point” – an OpenGL project developed by me, where you can add your functionalities
- It has all the necessary libraries included locally, so it should work directly in your VS.
- Attention! The demos are developed using VS 2019, so you might be asked when you open the project with an older version to retarget the platform version and / or the platform toolset. Agree to make those changes.

Our First OpenGL program

- If you did not make those changes when you were asked, you can go to the Property Page of the solution (Right click on the solution name -> Properties -> General -> Change Target Platform Version and / or Platform Toolset)





Create a Window in OpenGL

- First thing to do: create an application window to draw in
- Available libraries: GLFW, GLUT, FreeGLUT, SDL etc.
- GLFW:
  - special (external) library written in C
  - can create an OpenGL context
  - can define window parameters
  - can handle user input

Create a Window in OpenGL

- Functions in GLFW:
- glfwInit(); - initializes the GLFW context
- glfwCreateWindow (width, height, name for the window)
- glfwMakeContextCurrent (window) takes our window as a parameter and specifies that it is the current context that we use
  - Context = an “object” that holds all of OpenGL
- glfwTerminate(); - closes the GLFW context

Rendering a graphical app

- Time period of rendering = continuous (one or more tasks that run continuously, until being closed by the user or an event)
- Real-time applications (games or what we are doing at the lab) have this series of steps:
  - Polling events (window resize, input etc.)
  - Estimating execution time
  - Processing detected events
  - Processing the current frame
  - (Optional) Swapping frame buffers
  - Passing to the next frame

More info: Double buffering

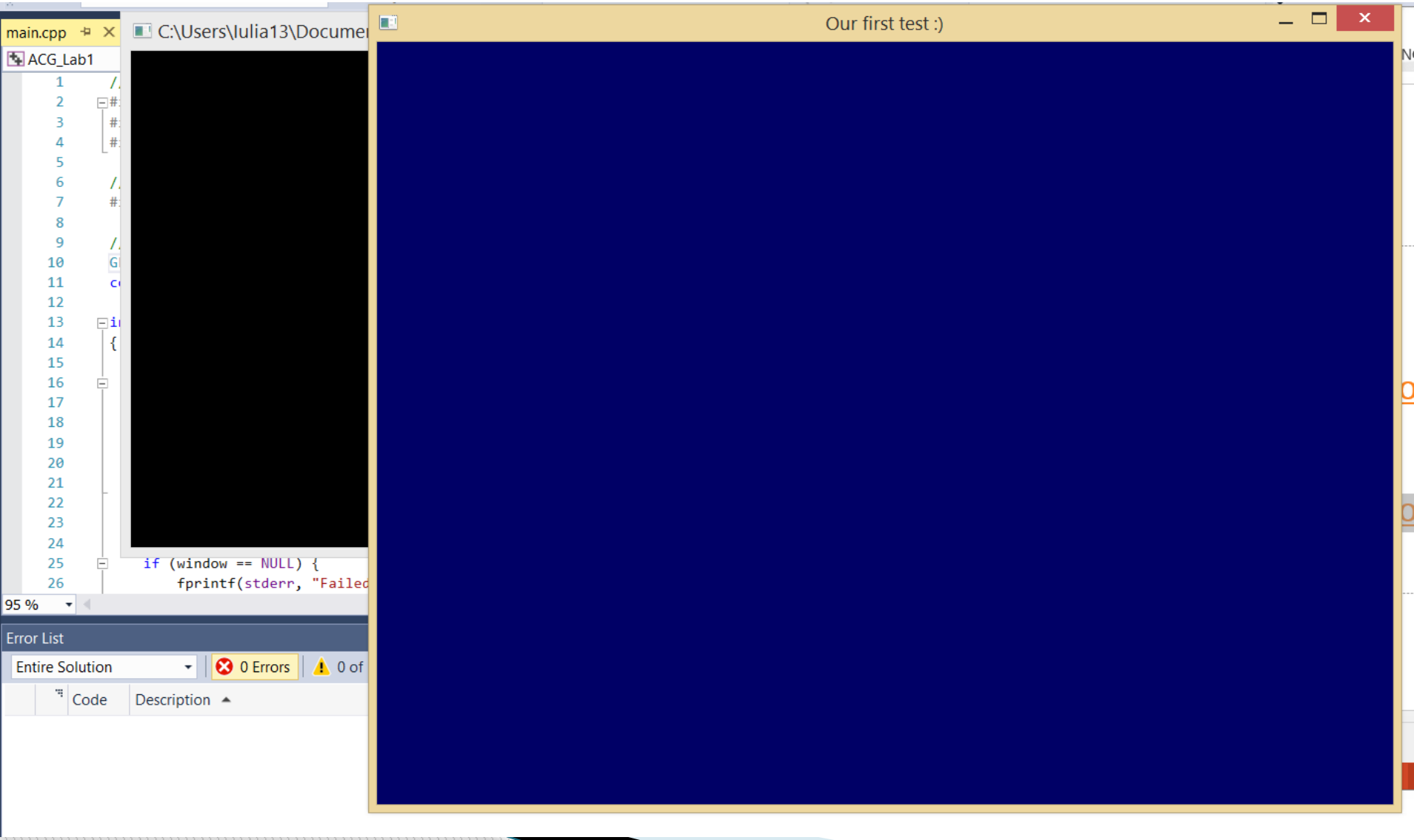
- Single buffering - sending the image OpenGL has generated to the graphics card and then to the monitor; unfortunately, we won't get the result in real time (we have the refresh rate of our monitors) => changing one frame with another causes flickering
- Double buffering - OpenGL uses a 'hidden' render; it can write to it while the previous one is being shown on the screen (we draw a frame on one buffer – the visible one – and the next frame on a second buffer, not visible; when we finished displaying the first frame, we swap buffers etc.)
  - Buffer = a region of a physical memory storage used to temporarily store data while it is being moved from one place to another

Create a Window in OpenGL

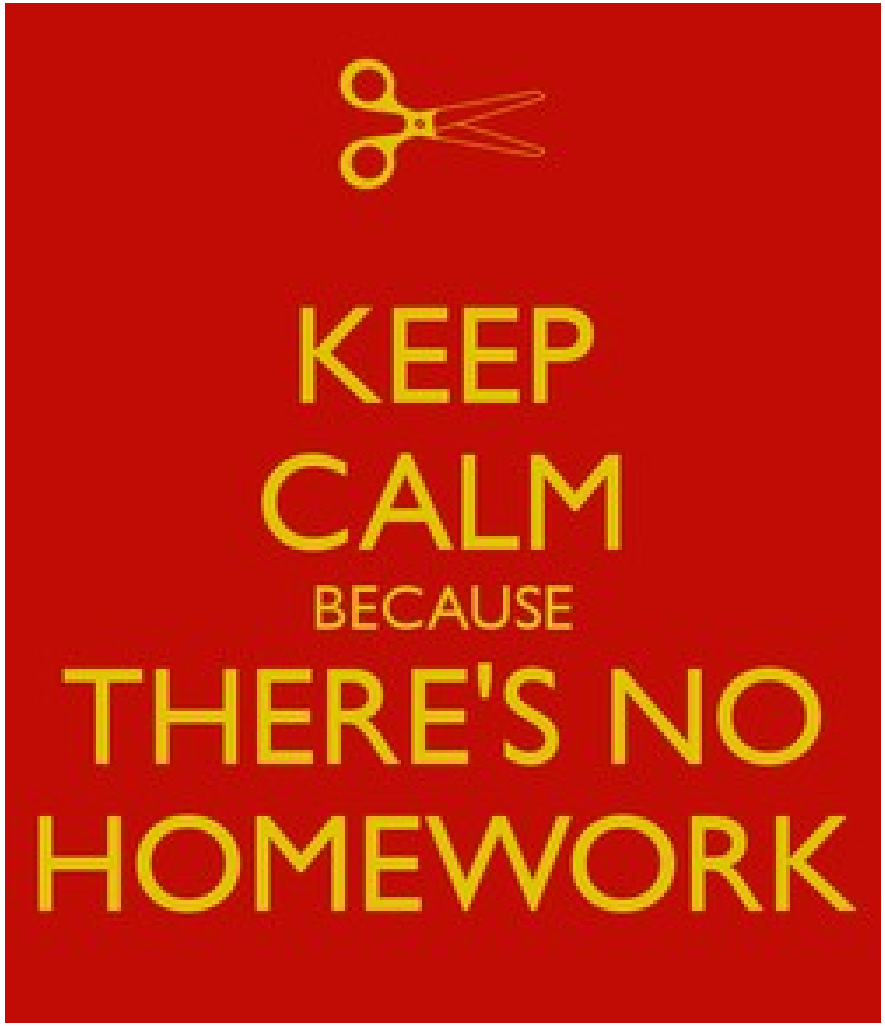
- We want the application to keep drawing images and handle user input until the program has been explicitly told to stop
- We will use a render loop that keeps running until we tell GLFW to stop

```
while(!glfwWindowShouldClose(window))
{
        glfwSwapBuffers(window);
        glfwPollEvents(); // what is this?
}
```

Result:


main.cpp
ACG_Lab1
C:\Users\Iulia13\Docume
if (window == NULL) {
fprintf(stderr, "Failed
95 %
Error List
Entire Solution
0 Errors
0 of
Code
Description
Our first test :)

Homework



…or is it?

- Install Visual Studio on your personal computers
- Create a new project and configure an OpenGL context; then draw your first window
- You can use the example provided on Moodle to test the OpenGL functionalities presented in this lab

- Obs:
  - Open the ACG_Lab1.sln file, NOT the main (so that all dependencies are automatically linked)
  - Run it in DEBUG mode, not in Release.

Tutorials and other useful links

- Configure VS and OpenGL (Windows): https://www.youtube.com/watch?v=k9LDF016_1A
- !!!! If you never used VS, it is highly recommended to follow this tutorial: https://docs.microsoft.com/en-us/visualstudio/ide/getting-started-with-cpp-in-visual-studio?view=vs-2015#BKMK_CreateApp

- Configure OpenGL on Mac (obviously without VS): https://www.youtube.com/watch?v=Tz0dq2krCW8
  (or Mac M1different paths: https://www.youtube.com/watch?v=MHlbNbWlrIM)
- OpenGL official docs: https://www.opengl.org/sdk/docs/man/
- Great OpenGL tutorials: https://learnopengl.com/
  http://www.opengl-tutorial.org/
  https://open.gl/
- About buffering: https://en.wikipedia.org/wiki/Multiple_buffering
- Video tutorial SIGGRAPH: https://www.youtube.com/watch?v=6-9XFm7XAT8