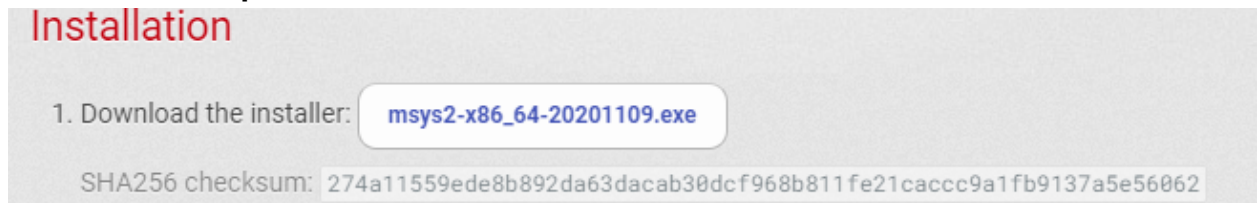


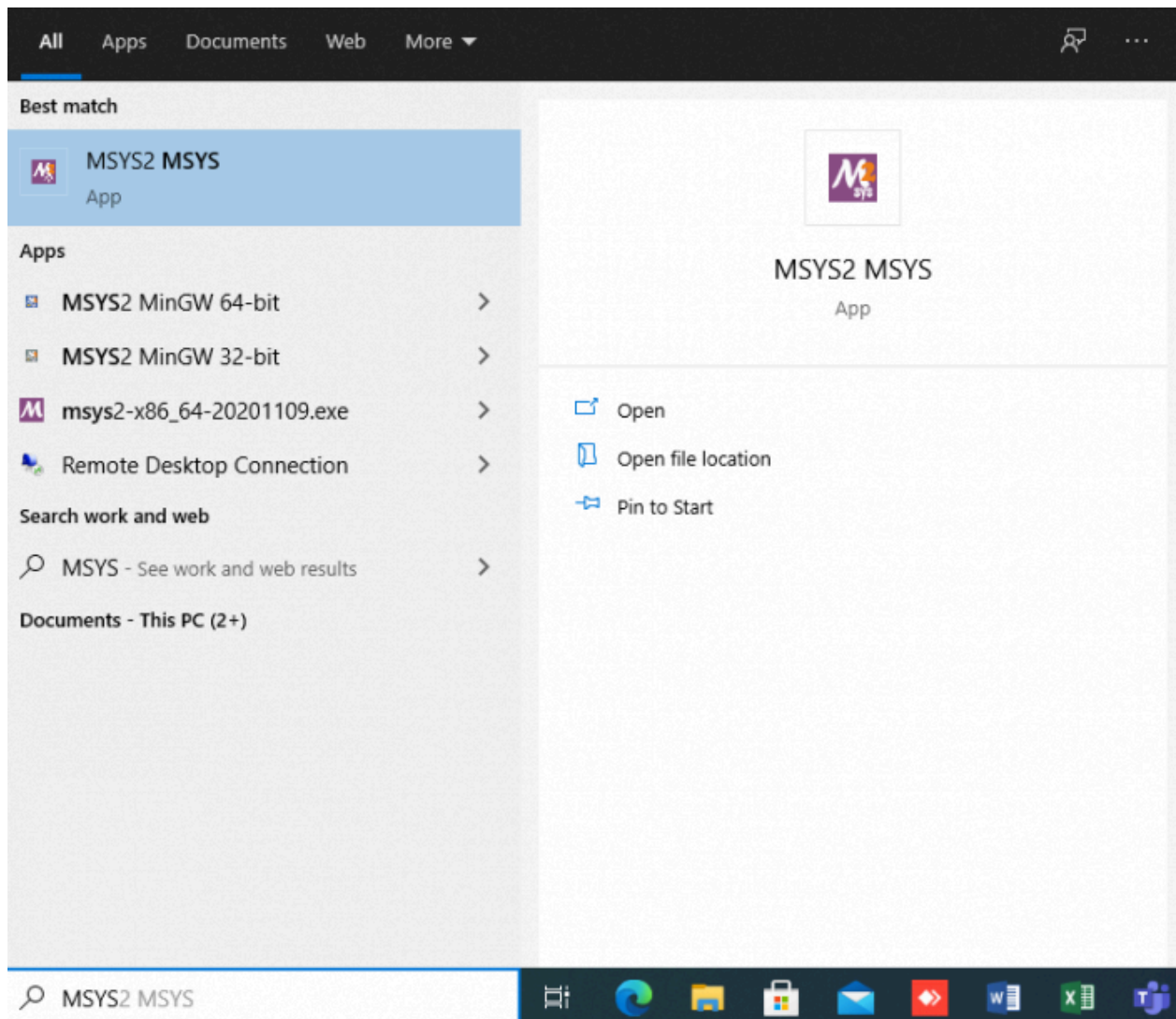
Windows

1. Install and Setup MSYS

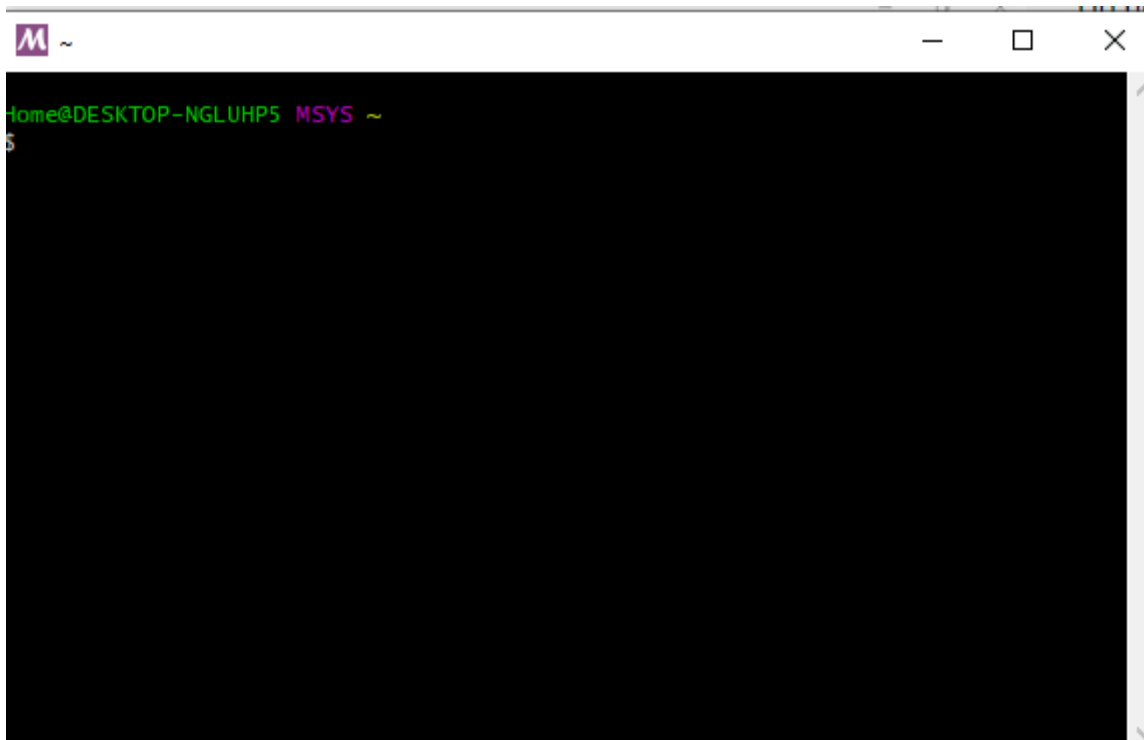


Head over to this link <https://www.msys2.org/> and download the msys2 installer. From there you can download the complete program.

2. Open MSYS2



Search for MSYS2 in the search box and open it. You will get a Command Prompt-like UI as:



3. Install Pacman in the MSYS2

Hit the below command in your brand new MSYS2 CLI

```
$ pacman -Syu
```

4. Install MinGW package via CLI

After Pacman is installed you will be able to use Pacman manager to install the MinGW package easily with the command below

```
$ pacman -S mingw-w64-x86_64-toolchain
```

5. Install Freeglut

Hit the below command to install freeglut

```
$ pacman -S mingw-w64-x86_64-freeglut
```

6. Install Glew

Now install glew with the following command

```
$ pacman -S mingw-w64-x86_64-glew
```

7. Update environment variable

Add the path to your Mingw-w64 `bin` folder to the Windows `PATH` environment variable by using the following steps:

- In the Windows search bar, type 'settings' to open your Windows Settings.
- Search for Edit environment variables for your account.
- Choose the `Path` variable in your User variables and then select Edit.
- Select New and add the Mingw-w64 destination folder path to the system path. The exact path depends on which version of Mingw-w64 you have installed and where you installed it. If you used the settings above to install Mingw-w64, then add this to the path: `C:\msys64\mingw64\bin`.
- Select OK to save the updated PATH. You will need to reopen any console windows for the new PATH location to be available.

- Now give the command below to compile the code.

```
$ g++ main.cpp -o demo.exe -lfreeglut -lglew32 -lopengl32 -lglu32
```

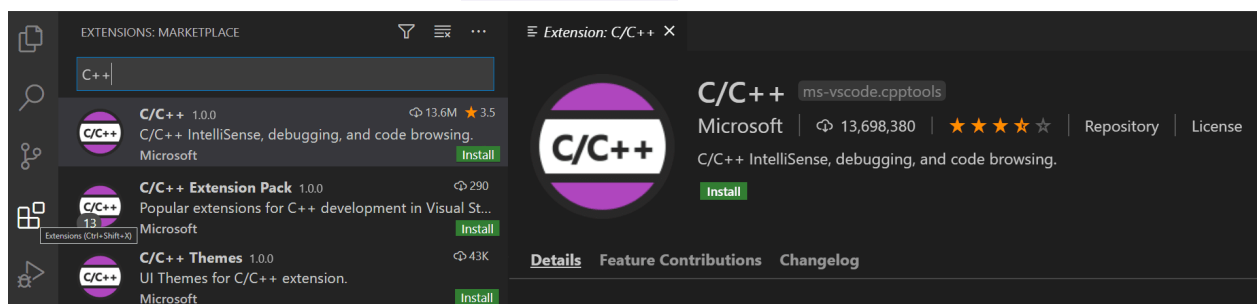
- Now run the OpenGL program with the following command.

```
$ start demo.exe
```

1. Setup with Visual Studio Code

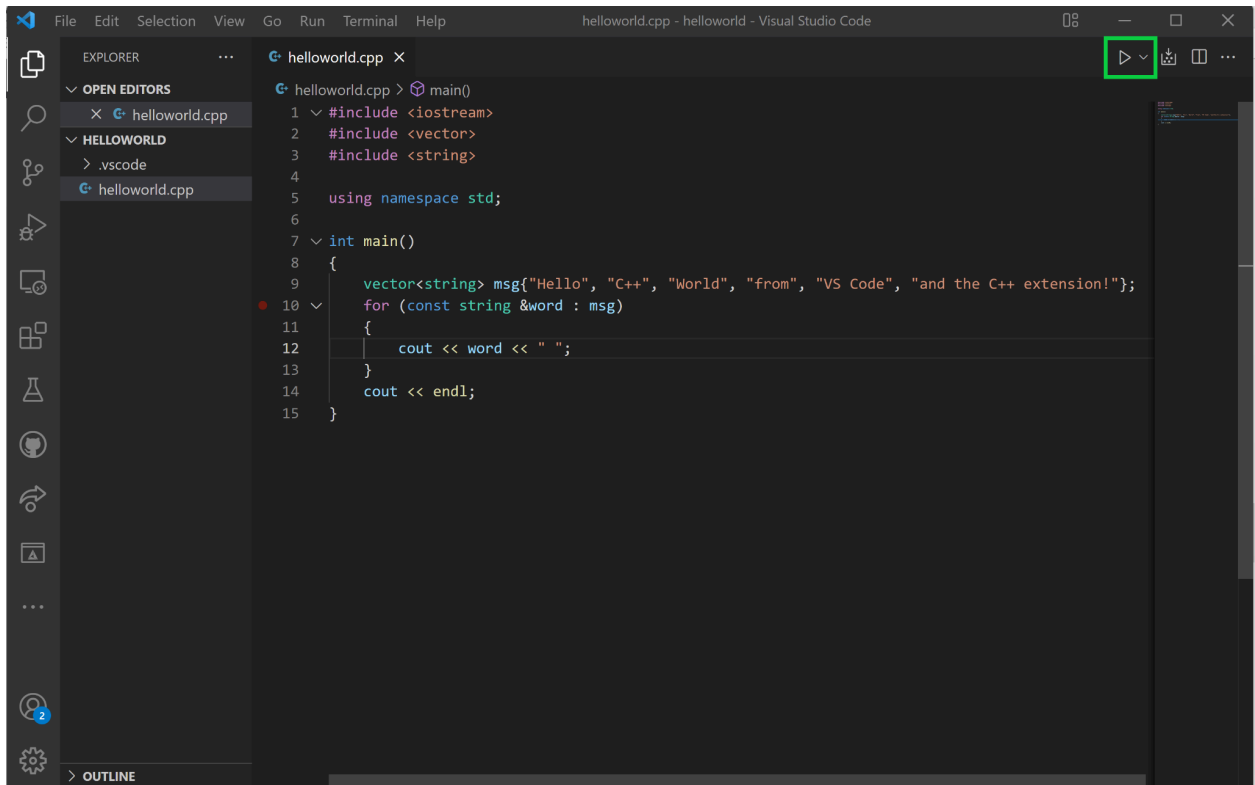
1.1 Prerequisites

- Install [Visual Studio Code](#).
- Install the [C/C++ extension for VS Code](#). You can install the C/C++ extension by searching for 'c++' in the Extensions view (**Ctrl+Shift+X**).

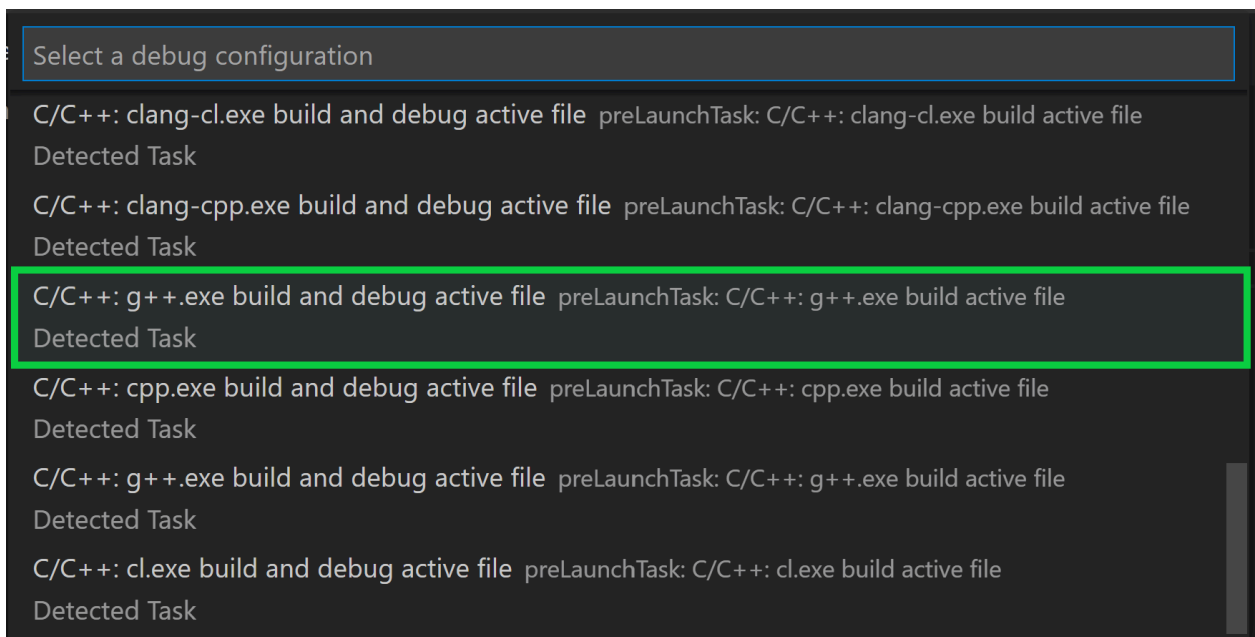


1.2 Running the Demo

- Download the following file and unzip it:
<https://drive.google.com/file/d/1nnXkTwAaOncFi0sanhdhPDxeu0FaLnKk/view?usp=sharing>
- Open the unzipped folder `demo` in vs code.
- In vs code, open `main.cpp` so that it is the active file.
- Press the play button in the top right corner of the editor.



5. Choose C/C++: g++.exe build and debug active file from the list of detected compilers on your system.



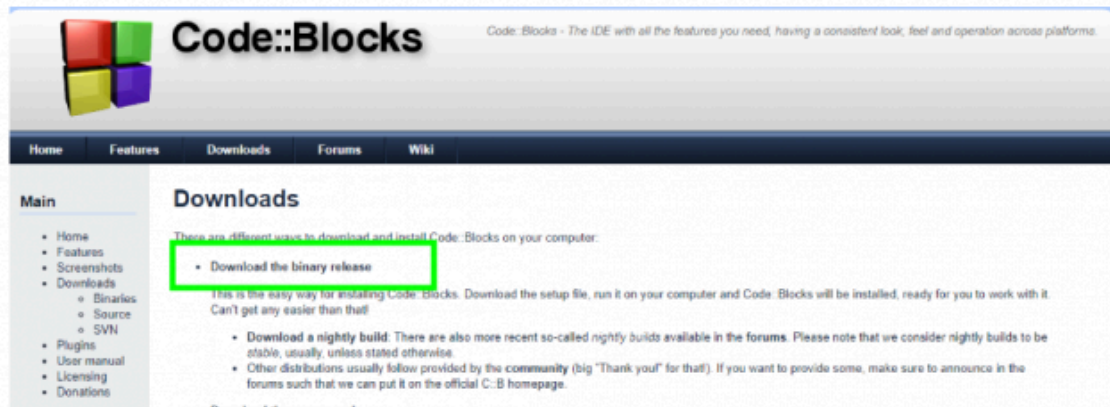
2. Setup with Code Blocks

2.1. Download Code Blocks

Here you have to be super careful on how to download the specific version of code blocks.

Since we have already set up MinGW package we don't need code block with mingw setup so install simple code block without mingw-setup as shown below :

Head over to this link <http://www.codeblocks.org/downloads> and download the binary release of code blocks as shown below:



Windows XP / Vista / 7 / 8.x / 10:

File	Date	Download from
codeblocks-20.03-setup.exe	29 Mar 2020	FossHUB or Sourceforge.net
codeblocks-20.03-setup-nonadmin.exe	29 Mar 2020	FossHUB or Sourceforge.net
codeblocks-20.03-nosetup.zip	29 Mar 2020	FossHUB or Sourceforge.net
codeblocks-20.03mingw-setup.exe	29 Mar 2020	FossHUB or Sourceforge.net
codeblocks-20.03mingw-nosetup.zip	29 Mar 2020	FossHUB or Sourceforge.net
codeblocks-20.03-32bit-setup.exe	02 Apr 2020	FossHUB or Sourceforge.net
codeblocks-20.03-32bit-setup-nonadmin.exe	02 Apr 2020	FossHUB or Sourceforge.net
codeblocks-20.03-32bit-nosetup.zip	02 Apr 2020	FossHUB or Sourceforge.net
codeblocks-20.03mingw-32bit-setup.exe	02 Apr 2020	FossHUB or Sourceforge.net
codeblocks-20.03mingw-32bit-nosetup.zip	02 Apr 2020	FossHUB or Sourceforge.net

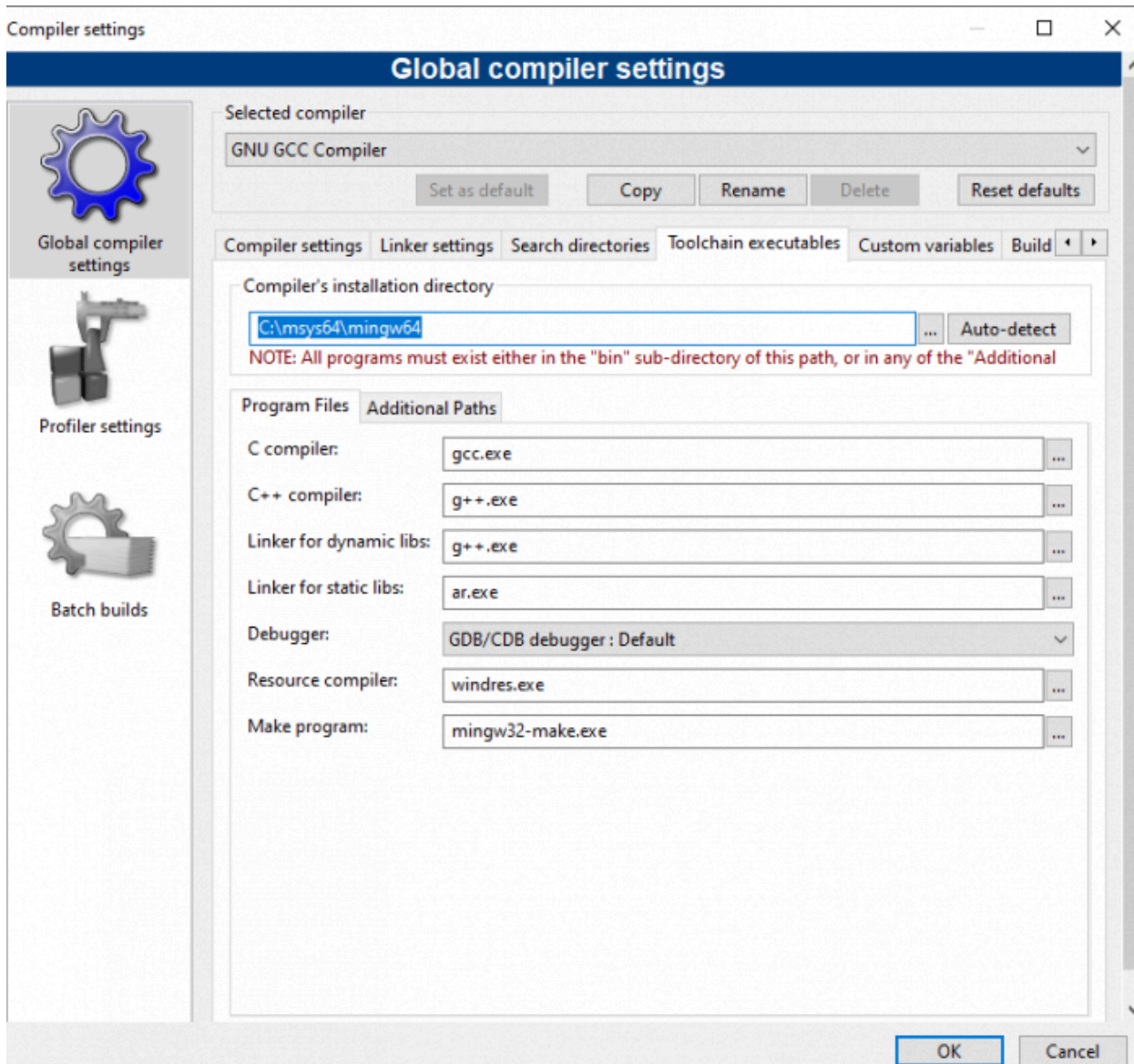
2.2. Set up the compiler in Code Blocks

After you are done with the installation of the code blocks, we have to do some compiler configuration in it.

Click on **Settings > Compiler > Global Compiler Settings**

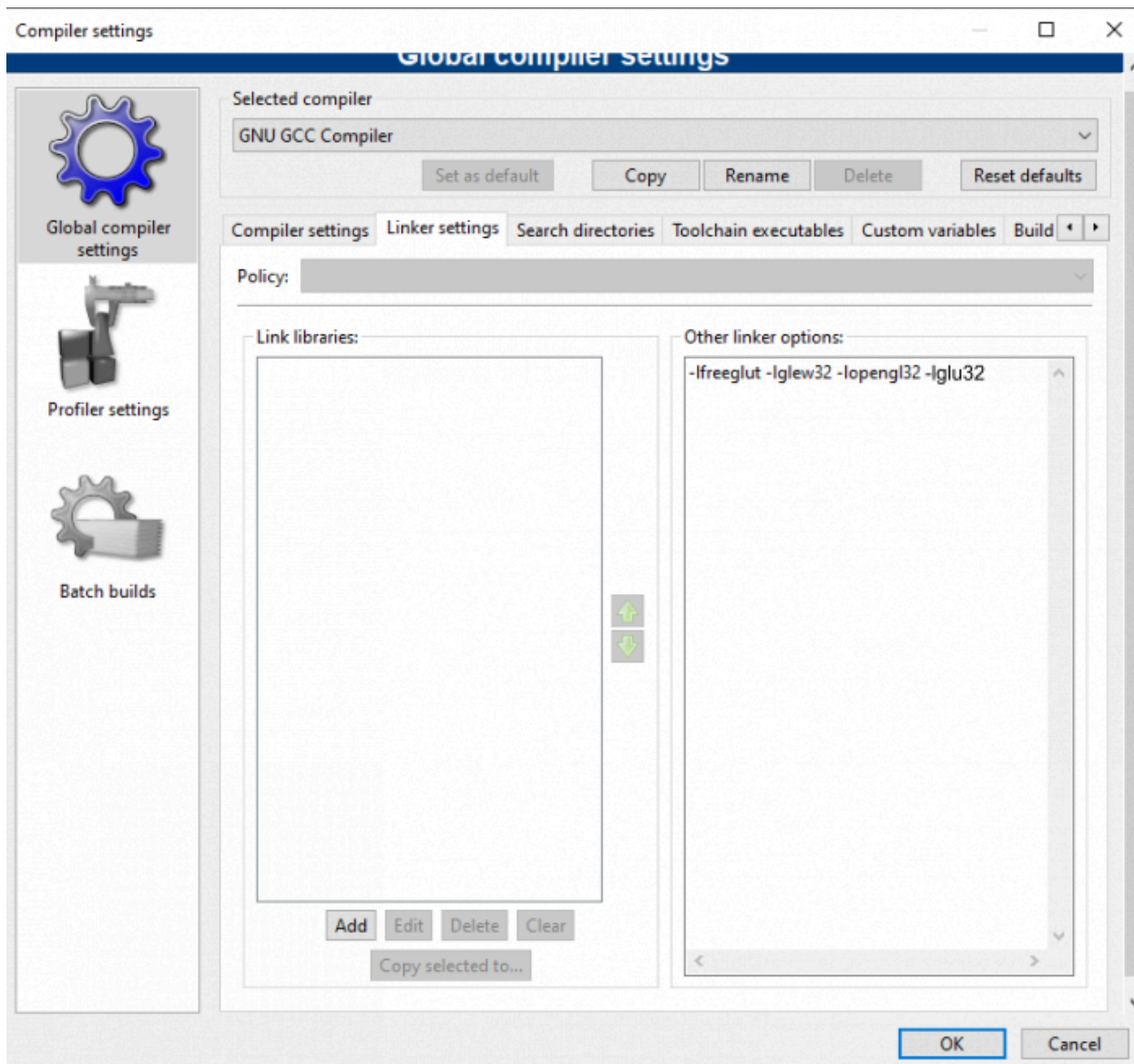
Under **Toolchain Executables** Tab there is **Compiler's Installation Directory**

You have to replace that with the mingw64 which resides inside the msys64 in the Local Disk C which is **C:\msys64\mingw64** as shown in the below figure:



After setting up the path in **Toolchain executables** now Click on the **Linker Settings** Tab. In the Right Portion, there are Other linker options where you have to add the following command.
`-lfreetglut -lglew32 -lopengl32 -lglu32`

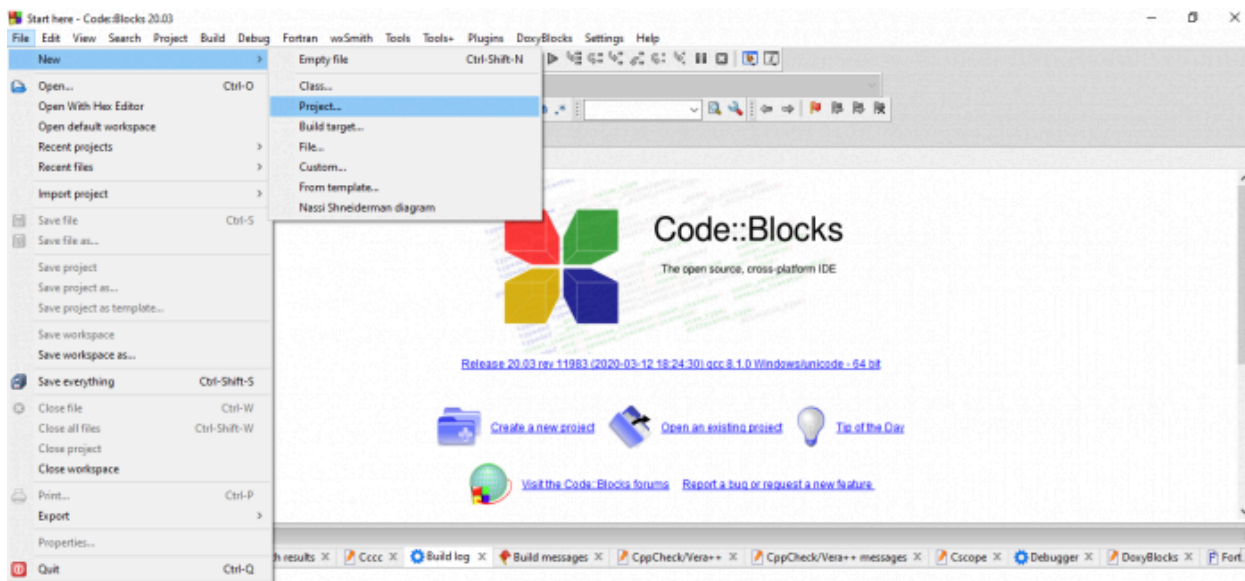
After adding it, the tab looks like as :



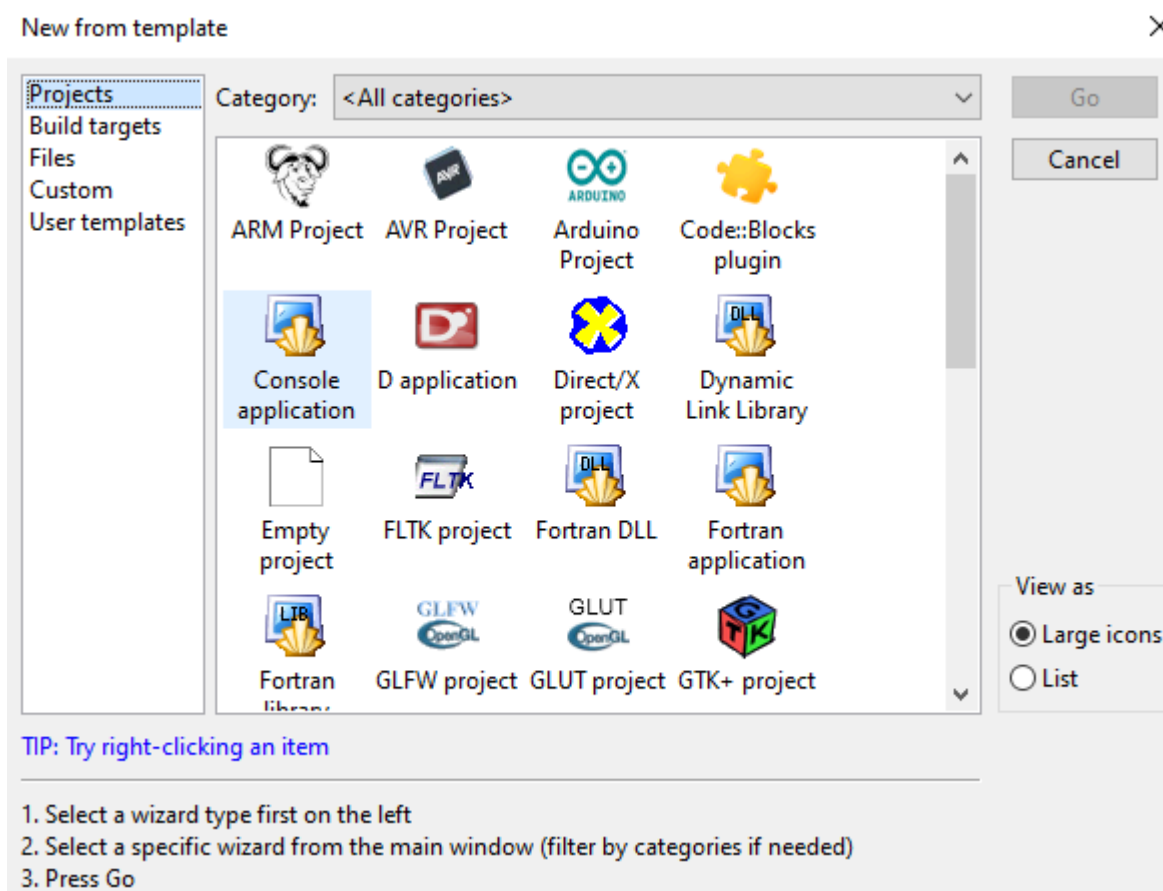
All done. Now you can create a new project for creating your OpenGL program. But before that we have to get some ground ideas to start a project for an OpenGL program in code blocks.

2.3. Creating a OpenGL program

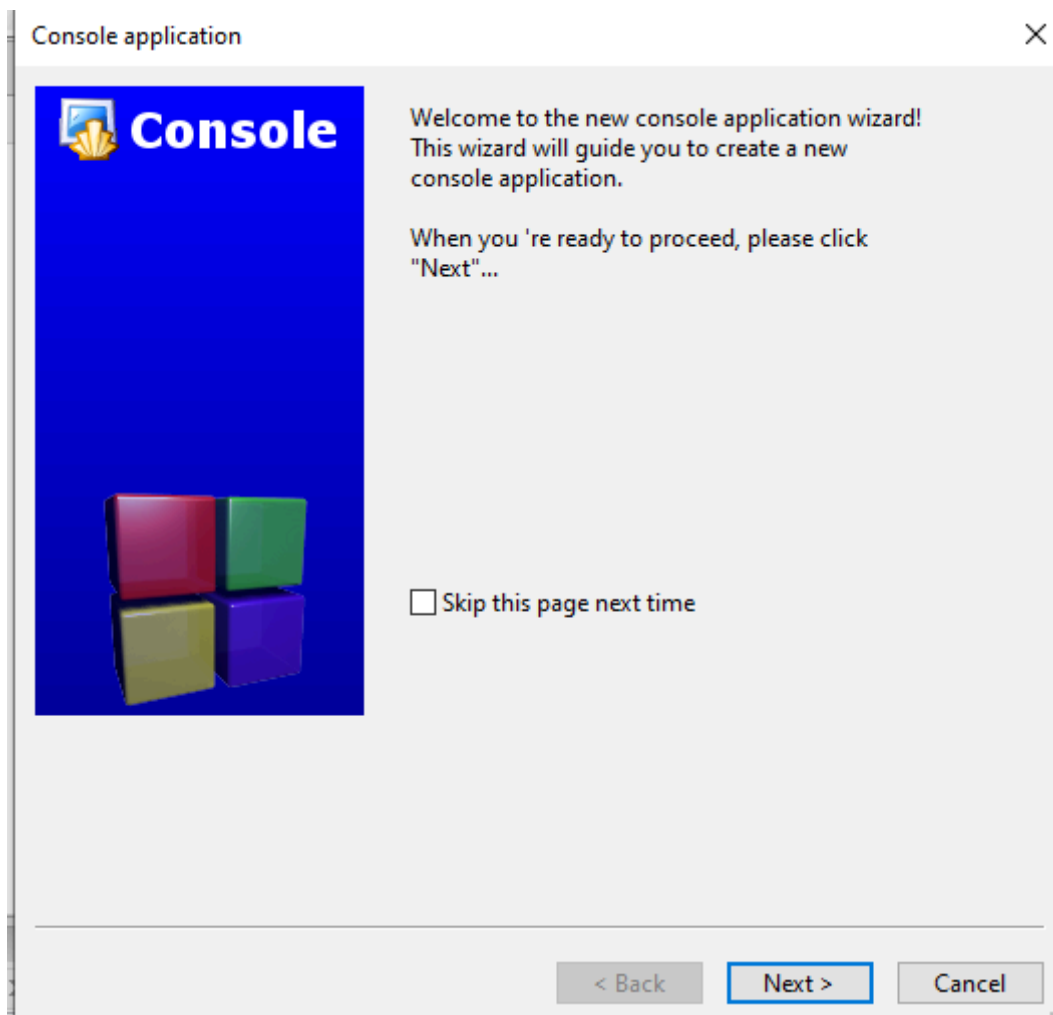
Click on **File > New > Project**



Now a **New From Template Tab** pops out in which you have to select **Console Application** and Click **Go** as illustrated below:

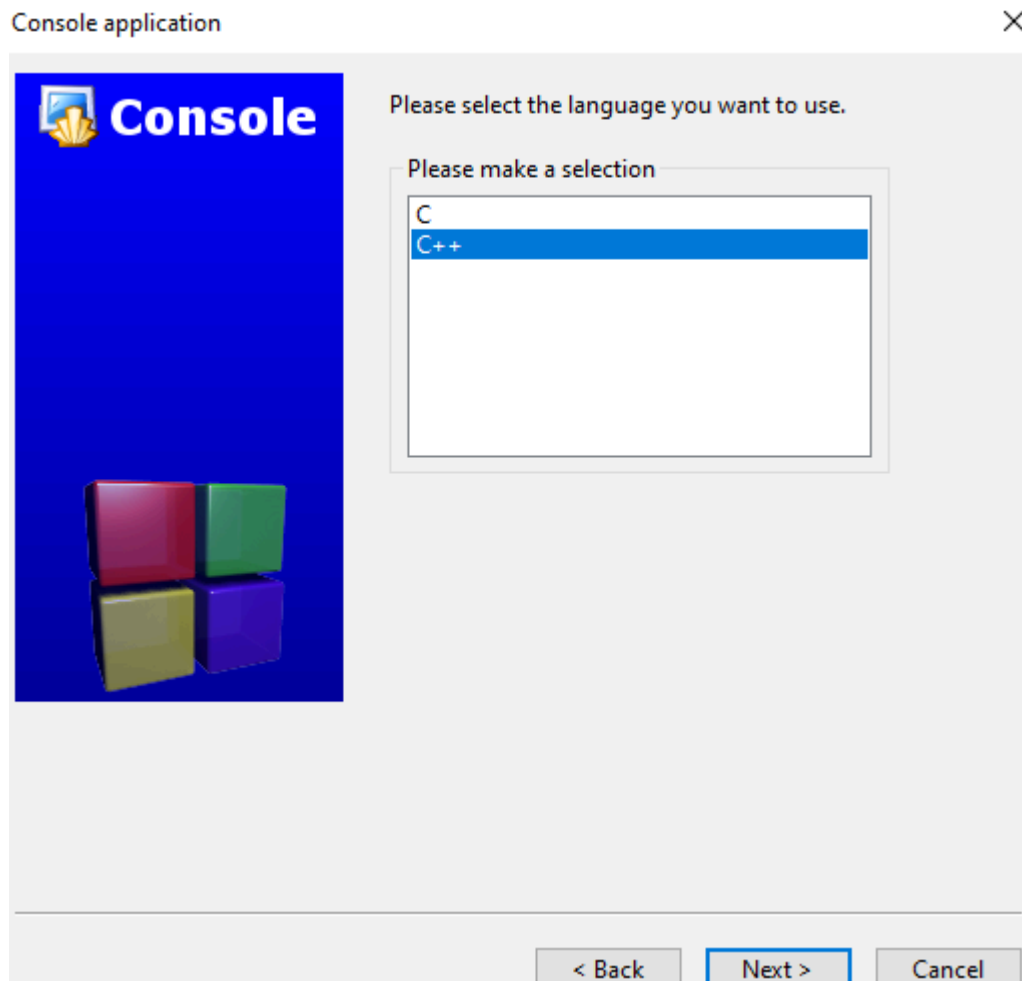


Now a console application tab pops out as



Click on **Next**.

Now there will be an option for C and C++. Choose the language in which you are going to write the program. For now we are choosing C++ language as :



After selecting the language, click **Next**

Now give a project title and click on **Next** as

Console application

Please select the folder where you want the new project to be created as well as its title.

Project title:
sample_opengl

Folder to create project in:
C:\Users\Home\Desktop\OpneGI\


Project filename:
sample_opengl.cbp

Resulting filename:
C:\Users\Home\Desktop\OpneGI\sample_opengl\san

< Back Next > Cancel

Finally click on **Finish**

Console application ✕



Please select the compiler to use and which configurations you want enabled in your project.

Compiler:
GNU GCC Compiler ▼

☒ Create "Debug" configuration:

"Debug" options

Output dir.:

Objects output dir.:

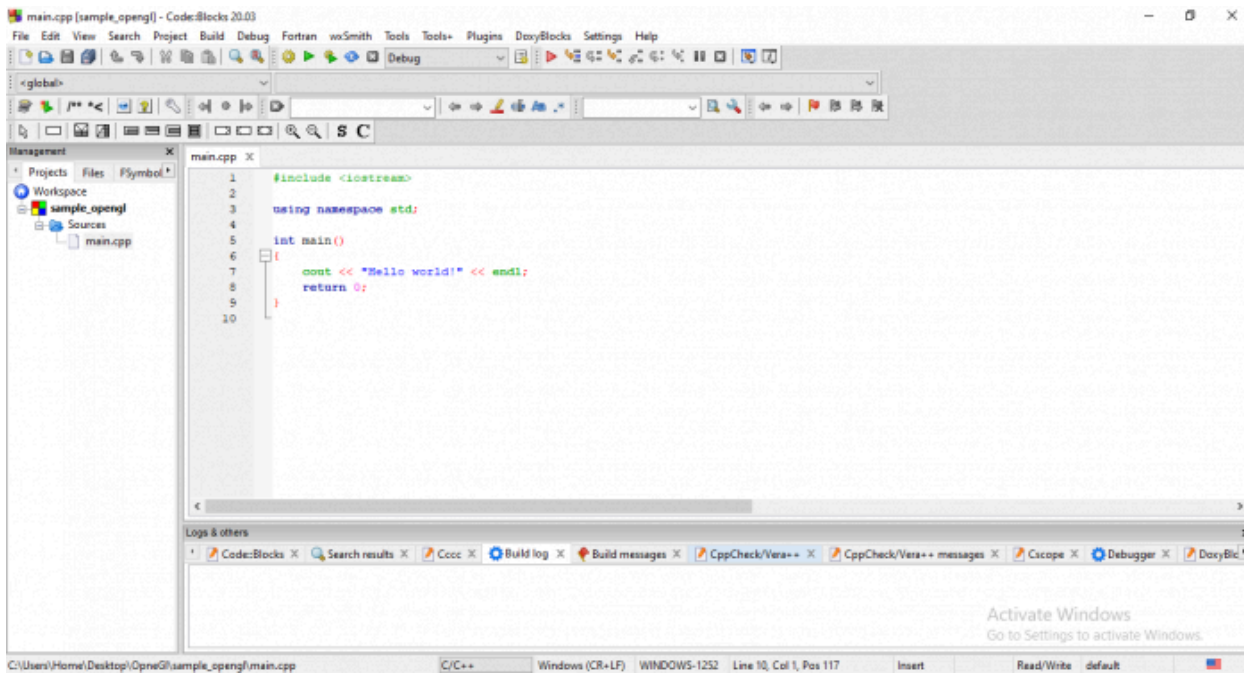
☒ Create "Release" configuration:

"Release" options

Output dir.:

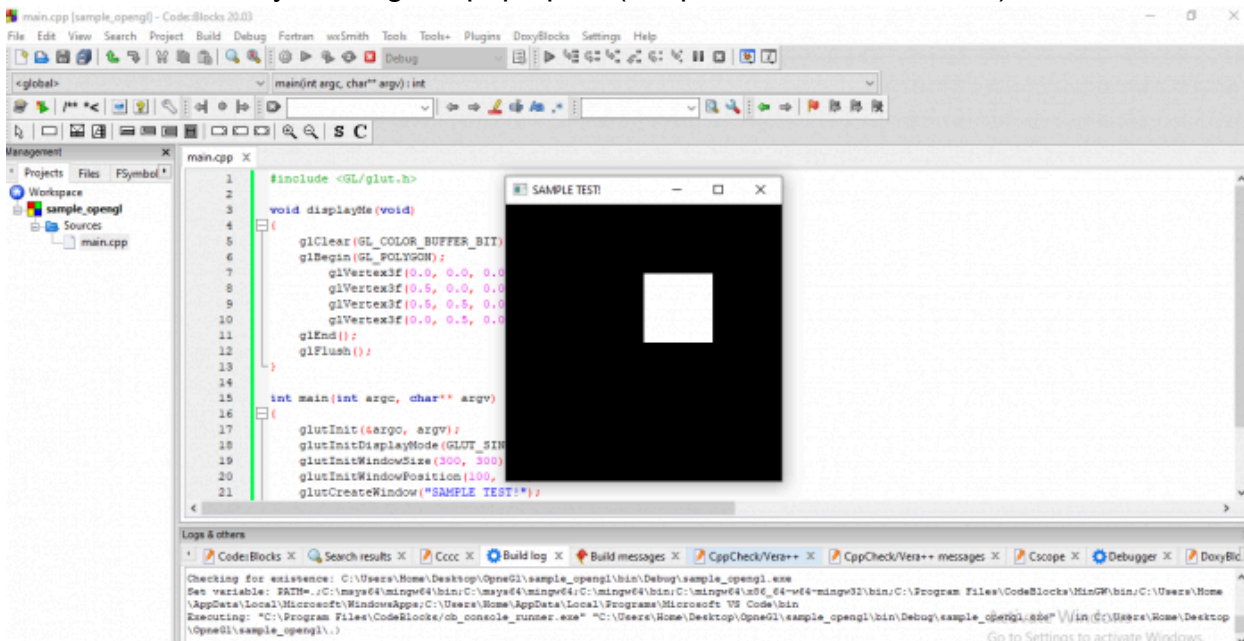
Objects output dir.:

Now our IDE starts and a sample Hello World Program is given under **Sources > main.cpp** file. Build and Run it and check the Setup runs successfully.



Check your OpenGL programs by replacing the sample main.cpp file with the following file:
<https://drive.google.com/file/d/1nnXkTwAaOncFi0sanhdhPDxeu0FaLnKk/view?usp=sharing>

After build and run you will get a pop up as (the picture will be different):



Congrats! All set you are ready to go!

References:

<https://medium.com/swlh/setting-opengl-for-windows-d0b45062caf>

Ubuntu

1. Run the following commands to install OpenGL.

```
$ sudo apt-get update
```

```
$ sudo apt-get install libglu1-mesa-dev freeglut3-dev mesa-common-dev
```

2. Now to test if OpenGL libraries are working fine on our Linux, download the following C++ Program:

<https://drive.google.com/file/d/1nnXkTwAaOncFi0sanhdhPDxeu0FaLnKk/view?usp=sharing>

3. Now give the command below to compile the code.

```
$ g++ main.cpp -o demo -lglut -lGLU -lGL
```

4. Now run the OpenGL program with the following command.

```
$ ./demo
```

References:

<http://www.codebind.com/linux-tutorials/install-opengl-ubuntu-linux/>