Exercise 1

Classes: week 1 (27th September)

Introduction

In this Course, you will implement an interactive Modern OpenGL based application.

To help your development, you should use the provided Visual Studio 2019 **AVT_Template.zip** template for Windows OS.

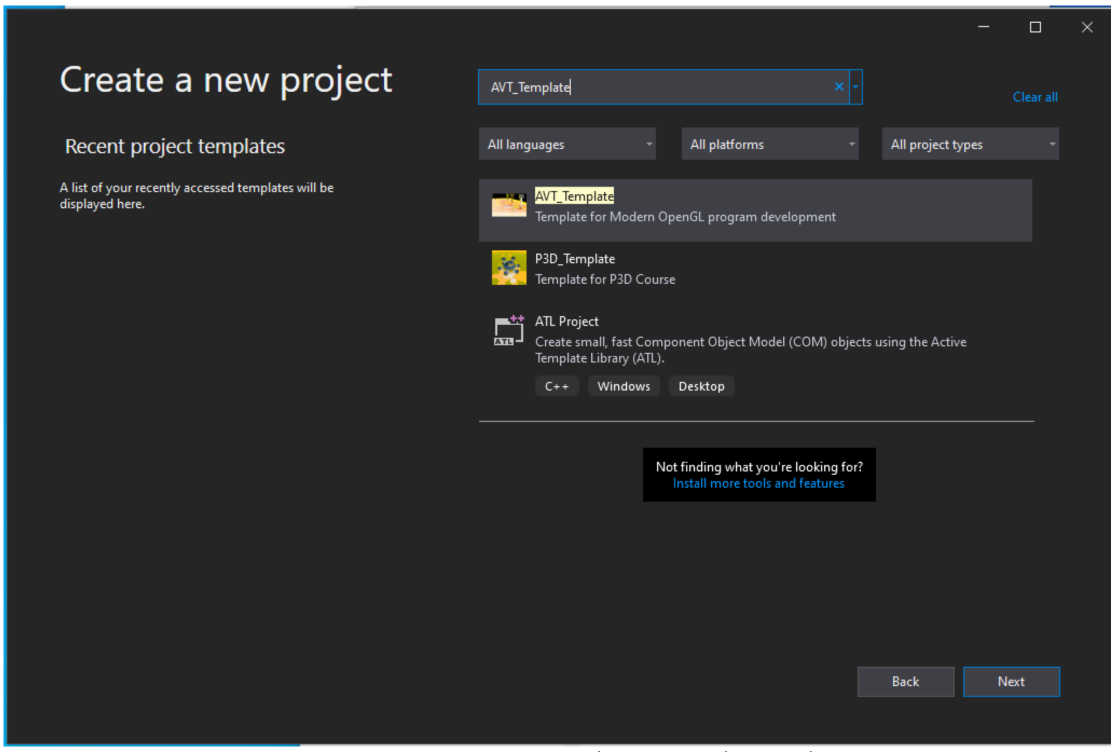
IMPORTANT: Download and install the free Visual Studio 2019 Community edition. The free version provides all that you need.

Update your graphics drivers to their latest versions.

Installing the template for application development

Your first task will be to install the AVT template. Download the template AVT_Template.zip and copy it to: "C:\Users\MyUsername\Documents\Visual Studio 2019\Templates\ProjectTemplates". If Templates and ProjectTemplates directories don't exist, you should create them.

Now you can create your project. Open the Visual Studio and select select File => New => Project from the menu. Select the "AVT_Template" template (Figure 1).



Create a new project

AVT_Template

Recent project templates

A list of your recently accessed templates will be displayed here.

AVT_Template
Template for Modern OpenGL program development

P3D_Template
Template for P3D Course

ATL Project
Create small, fast Component Object Model (COM) objects using the Active Template Library (ATL).

Not finding what you're looking for?
Install more tools and features

Figure 1 - Create a new project using the AVT_Template template.

Then click "Next" and call the project something like "MyAVTapp". After clicking "Create" it will create the new project.

Compiling and running

Select "x64" (this is very important) on the platform, and then right-click on the new "MyAVTapp" project in the Solution Explorer and select Build. With the first build it will set up Dependencies, which includes the **freeglut**, **glew**, **DevIL** and **Assimp** packages, within the project. The Dependencies files are in a self-extracting archive. It will pop up a window asking you where to put it (Figure 2). Simply click the Extract button.

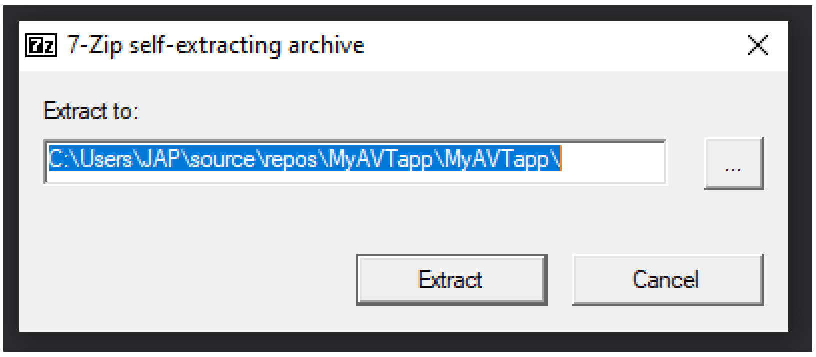
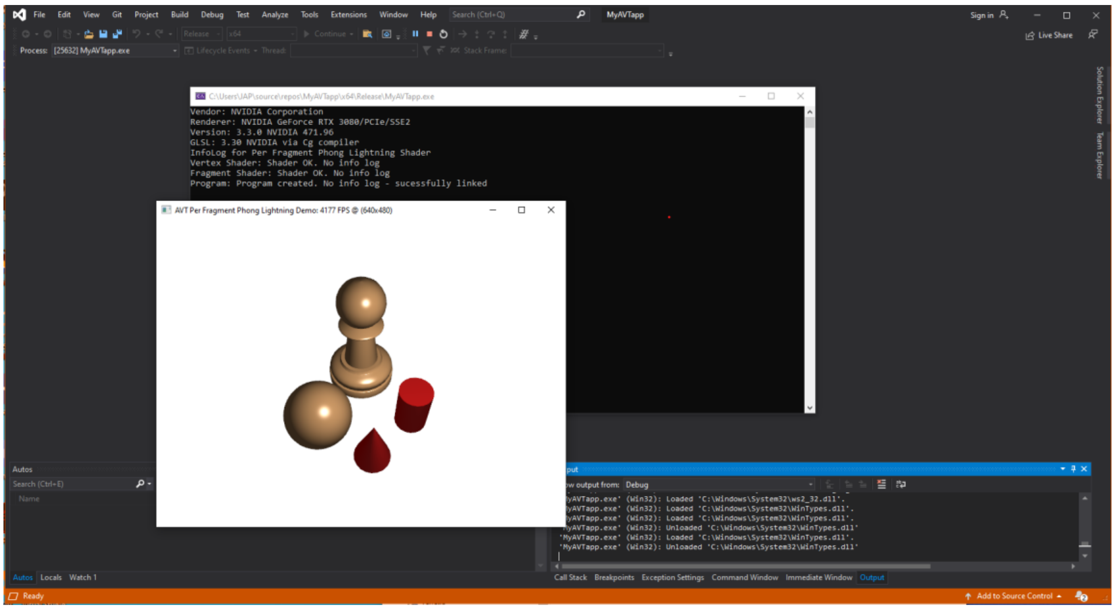

7-Zip self-extracting archive
Extract to:
C:\Users\JAP\source\repos\MyAVTapp\MyAVTapp\
...
Extract
Cancel

Figure 2: Click Extract to unpack the freeglut, glew and DevIL packages.

Then, select Debug => Start Debugging (or Release=> Start Debugging for faster performance). Then, it will be displayed in your monitor an OpenGL window with four revolution solids (Figure 3). You can press the mouse left button to move the camera as well as to press the right button to zoom. The last feature can also be accomplished by using the mouse wheel.


C:\Users\JAP\source\repos\MyAVTapp\x64\Release\MyAVTapp.exe
AVT Per Fragment Phong Lightning Demo: 4177 FPS @ (640x480)
Release
x64
Show output from: Debug

```
Vendor: NVIDIA Corporation
Renderer: NVIDIA GeForce RTX 3080/PCIe/SSE2
Version: 3.3.0 NVIDIA 471.96
GLSL: 3.30 NVIDIA via Cg compiler
InfoLog for Per Fragment Phong Lightning Shader
Vertex Shader: Shader OK. No info log
Fragment Shader: Shader OK. No info log
Program: Program created. No info log - sucessfully linked
```

```
yAVTapp.exe' (Win32): Loaded 'C:\Windows\System32\ws2_32.dll'.
yAVTapp.exe' (Win32): Loaded 'C:\Windows\System32\WinTypes.dll'.
yAVTapp.exe' (Win32): Loaded 'C:\Windows\System32\WinTypes.dll'.
yAVTapp.exe' (Win32): Unloaded 'C:\Windows\System32\WinTypes.dll'
'MyAVTapp.exe' (Win32): Loaded 'C:\Windows\System32\WinTypes.dll'.
'MyAVTapp.exe' (Win32): Unloaded 'C:\Windows\System32\WinTypes.dll'
```

Figure 3: AVT Per-Fragment Phong Lighting demo

Template code overview

This template implements an interactive OpenGL application to demonstrate the per-fragment Phong shading technique which will be presented and discussed in the theoretical classes. You should analyze the code, in particular the transformations and code to move the camera and objects. To understand the given example, besides the Course bibliography, you are advised to consult the following tutorial in http://www.lighthouse3d.com/tutorials/glsl-tutorial/glsl-core-tutorial-index/

The lightDemo.cpp file contains the OpenGL C++ application. This C++ program together with both vertex and fragment GLSL-coded shaders, stored in *SHADERS* directory, draw four revolution solids with Phong shading in an interactive window. The fragment shader implements the per-fragment Phong lighting technique. The C++ application uses the *freeglut*, a simple open-source library, to communicate with the Windows windowing system and uses another open-source library, named *glew*, to make easier an user to access the OpenGL functions.

freeglut package gives us the bare necessities required for rendering goodies to the screen. It allows you to create an OpenGL context, define window parameters and handle user input, which is plenty enough for your purposes.

Because OpenGL is only really a standard/specification it is up to the driver manufacturer to implement the specification to a driver that the specific graphics card supports. Since there are many different versions of OpenGL drivers, the location of most of its functions is not known at compile-time and needs to be queried at run-time. It is then the task of the developer to retrieve the location of the functions he/she needs and store them in function pointers for later use. Retrieving those locations is OS-specific which can be a complex process. Additionally, OpenGL has many versions and profiles which expose different sets of functions. So, managing function access is cumbersome and window-system dependent. To overcome these difficulties, *glew* library is used in order to hide details and make easier an user to access the OpenGL functions.

The code of the template includes three important C++ libraries, adapted from http://www.lighthouse3d.com/very-simple-libs, to assist your development, namely vsShaderLib.cpp, AVTmathLib.cpp and basic_geometry.cpp. The vsShaderLib.cpp file contains functions to make life simpler when dealing with shaders; AVTmathLib.cpp file includes functions to perform geometric transformations to three types of stacked matrices, namely MODEL, VIEW and PROJECTION; and basic_geometry.cpp file which provides functions to create basic geometry (quad, cube, sphere, cone, cylinder, torus, etc.).

Although not used in the Phong lighting demo, the template also includes the Texture_Loader.cpp file which contains two functions to load both 2D and 3D (cubic) textures. These 2 functions make use of the DevIL package to support several image formats like jpg, png or tga.

A very popular model importing library called *Assimp*, that stands for Open Asset Import Library, is also included in the template. Assimp is able to import dozens of different model file formats (and export to some as well) by loading all the model's data into Assimp's generalized data structures. As soon as Assimp has loaded the model, we can retrieve all the data we need from Assimp's data structures. Because the data structure of Assimp stays the same, regardless of the type of file format we imported, it abstracts us from all the different file formats out there.

You are free to use these features, to change and/or extend them to make your further developments.

IMPORTANT: this code comes with no guarantee, so use it at your own risk.

Tasks to be implemented

First lab class

After installing and running the demo provided by the AVT template you should:

1) Add a cube to the scene.
2) Check the file cube.h and understand why the number of vertices is 24 instead of 8.
3) Change the code in lightDemo.cpp to display 60 FPS. To accomplish that, you should program the callback refresh(int value) which is used by the glutTimerFunc().

Second lab class

1) Modify the function createCube() in the file basic_geometry.cpp by just using glBufferData() function and removing the glBufferSubData().
2) Adapt the GLSL code in the shaders to implement the Gouraud shading technique.