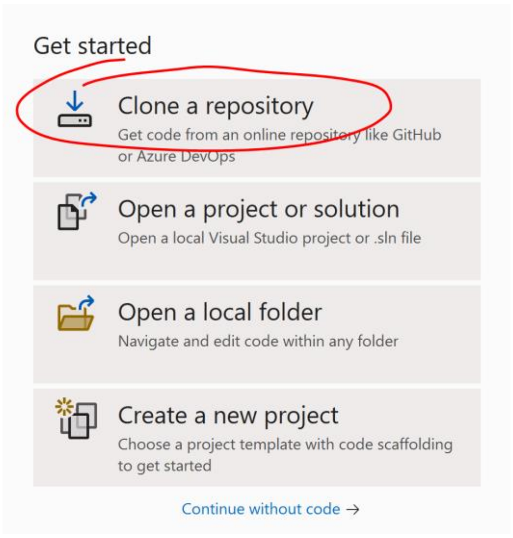
Objectives

1. Understand how to create direct3D project
2. Understand the structure of a simple DirectX program
3. Understand how to create simple geometric objects as triangle meshes
4. Understand how to specify rasterization state
5. Understand how to specify geometric primitives

Install Direct3D SDK tutorials

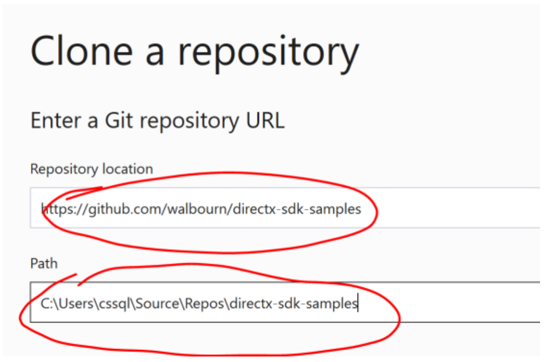
These tutorials will be used as the starting template for doing the lab exercises specified below. Please note that the solutions to the exercises **will not be provided**. The main reason behind the consideration is that this will encourage you to engage more actively with lab demonstrators and to do research to Direct3D API details.

- Start Visual Studio 2022
- Create a project by cloning Github repository:


Get started
Clone a repository
Get code from an online repository like GitHub or Azure DevOps
Open a project or solution
Open a local Visual Studio project or .sln file
Open a local folder
Navigate and edit code within any folder
Create a new project
Choose a project template with code scaffolding to get started
Continue without code →

- In the pop-up boxes, copy the following link in the Repository location box:

  https://github.com/walbourn/directx-sdk-samples



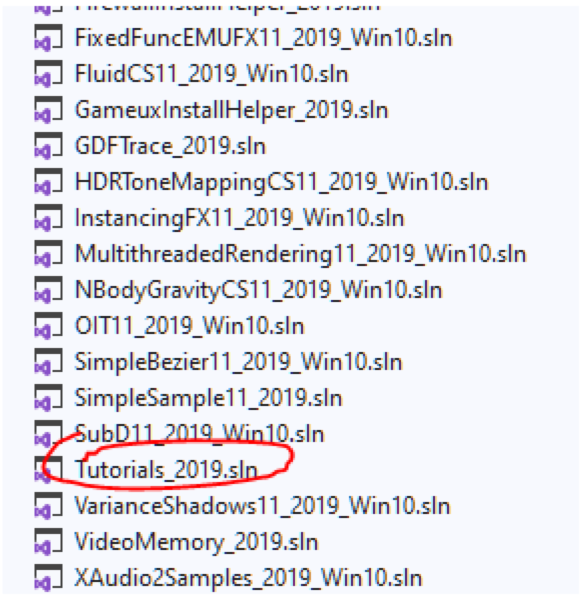
Clone a repository

Enter a Git repository URL

Repository location: https://github.com/walbourn/directx-sdk-samples

Path: C:\Users\cssql\Source\Repos\directx-sdk-samples

- In the path box, specify where you would like to save the cloned project
- Then, click the Clone button to clone the project.
- Once the repository has been cloned, you should see a list of Visual Studio Solutions.
- Select Solution Tutorials 2019 by double clicking Tutorial_2019.sln:



FixedFuncEMUFX11_2019_Win10.sln
FluidCS11_2019_Win10.sln
GameuxInstallHelper_2019.sln
GDFTrace_2019.sln
HDRToneMappingCS11_2019_Win10.sln
InstancingFX11_2019_Win10.sln
MultithreadedRendering11_2019_Win10.sln
NBodyGravityCS11_2019_Win10.sln
OIT11_2019_Win10.sln
SimpleBezier11_2019_Win10.sln
SimpleSample11_2019.sln
SubD11_2019_Win10.sln
Tutorials_2019.sln
VarianceShadows11_2019_Win10.sln
VideoMemory_2019.sln
XAudio2Samples_2019_Win10.sln

- You should now have a list of Tutorials shown in the solution explorer. You can compile and run each project and verify if the repository has been properly cloned.

Exercise 0.

Select Tutorial04 as the start-up project. Open “Tutorial04.cpp” and familiar yourself with the program by walking through the code. Compile and run, you should see a coloured cube in rotation. If you do NOT want to see the cube in rotation, simply go to the line

```
g_World = XMMatrixRotationY( t );
```

and change the variable `t` in `XMMatrixRotationY( t ) with a fixed number, say, 0.5f.`

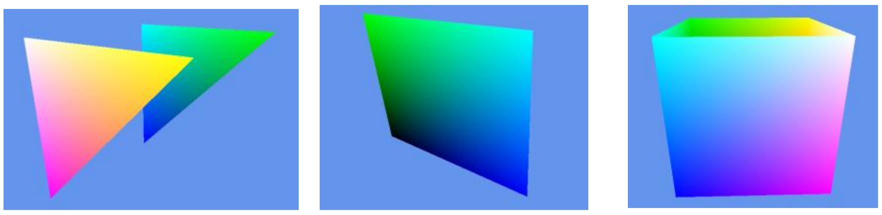
With current eye position, you are unable to see the top face of the cube. You can move the eye position slightly higher than the one specified in the program to make the top face of the cube visible. To change the eye position, go to the line

```
XMVECTOR Eye = XMVectorSet( 0.0f, 1.0f, -5.0f, 0.0f );
```

and change the current eye position (`0.0f, 1.0f, -5.0f`) to, say, (`0.0f, 2.5f, -5.0f`).

Exercise 1.

Modify the vertex list in `indices[]` or modify the parameters in the DrawIndexed( ) to draw: (1) two triangles; (2) one face of the cube; (3) the four walls of the cube

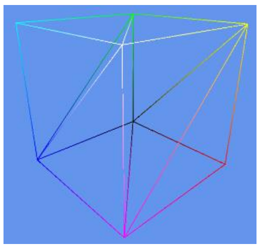


You may notice that only one face of the triangle and square face being drawn. This is because by default, the back face of the cube is culled out. You can specify the cull mode to be none by filling the rasterization state table:

```
ID3D11RasterizerState* m_rasterState=0;

D3D11_RASTERIZER_DESC rasterDesc;
rasterDesc.CullMode = D3D11_CULL_NONE;
rasterDesc.FillMode = D3D11_FILL_SOLID;
rasterDesc.ScissorEnable = false;
rasterDesc.DepthBias = 0;
rasterDesc.DepthBiasClamp = 0.0f;
rasterDesc.DepthClipEnable = true;
rasterDesc.MultisampleEnable = false;
rasterDesc.SlopeScaledDepthBias = 0.0f;


hr = g_pd3dDevice->CreateRasterizerState(&rasterDesc, &m_rasterSate);

g_pImmediateContext->RSSetState(m_rasterSate);
```

Exercise 2.

Draw the cube as a wireframe. This can be done by modify the fill mode in the rasterizer description table.



Exercise 3

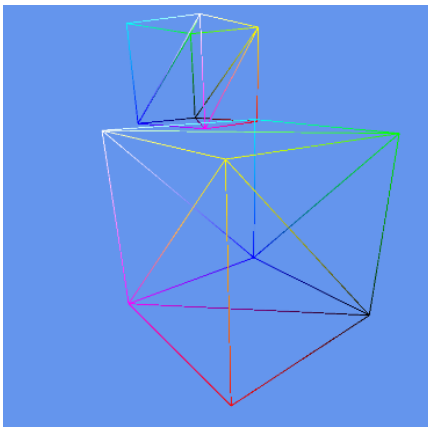
Modify the parameter in `IASetPrimitiveTopology( )` and `indices[]` to draw:

1. A list of points corresponding to the cube's eight vertices.
2. The 12 edges of the cube (not as a wireframe triangle mesh).

Exercise 4

Draw two wireframe cubes. There are different ways of achieving this. One simple way is to pass a different world matrix to the vertex shader. For example,

```
g_World *= XMMatrixTranslation(-2.8f, 0.0f, 0.0f);
cb.mWorld = XMMatrixTranspose(g_World);

g_pImmediateContext->UpdateSubresource(g_pConstantBuffer, 0, nullptr, &cb, 0, 0);
g_pImmediateContext ->DrawIndexed( … … );
```



Exercise 5

Draw the cube as triangle strips by setting primitive topology as D3D11_PRIMITIVE_TOPOLOGY_TRIANGLESTRIP.