The Amazing DX Workshop

workshop presenter

rohit gonsalves – computer graphics. AR VR expert

experience

2004-till date – 14 years of experiences in programming games on consoles and mobiles. Virtual reality for broadcasting, augmented reality using tracking mechanisms. Hardware specification designer, rendering engine architect. Enthusiast IoT developer

Engines – unigine, unreal, unity

industry – graphics, AR-VR-MR, Broadcast, Realtime graphics

TYBSc GP workshop
DX11 and Unity

august

08

10:00

intro

Introduction about the workshop and agenda explanation.

10:30

directx11

Step by step six practical of DirectX 11 with api explanation

13:00

lunch

lunch break

14:00

Continue

Continue with practical's

16:00

syllabus

referencing syllabus once again

17:00

conclude

Q/A and conclusion

pralvr

directX 11 practicals

01 initialize dx device

Setup DirectX 11, Window Framework and Initialize Direct3D11.

02 draw plane

buffers, vertices and shaders to render a colored plane o

03 Texturing the plane

Use textures to draw onto the planes to give realistic rendering

04 diffuse lighting

programmable diffuse lightning in HLSL

05 specular lighting

programmable specular lightning in HLSL

06 loading models

load any kind of model exported from other editors using direct3D

Windows 10 SDK requirements

Supported operating systems

Windows 10 App Development (UWP)

- Windows 10 version 1507 or higher: Home, Professional, Education, and Enterprise (LTSB and S are not supported)
- Windows Server 2012 R2 (Command line only)
- Windows Server 2016 (Command Line only)

Win32 Development

- Windows 10 version 1507 or higher
- Windows Server 2016: Standard and Datacenter
- Windows 8.1
- Windows Server 2012 R2
- Windows 7 SP1

(Not all tools are supported on earlier operating systems)

Hardware requirements

- 1.6 GHz or faster processor
- 1 GB of RAM
- 4 GB of available hard disk space

GPU you need
for your assignments

Hybrid GPUs

All newer computers with 4th generation onwards have processor graphics for intel processors. If you have 5th and above generation processors then Processor Graphics are damn good for your work on these assignments.

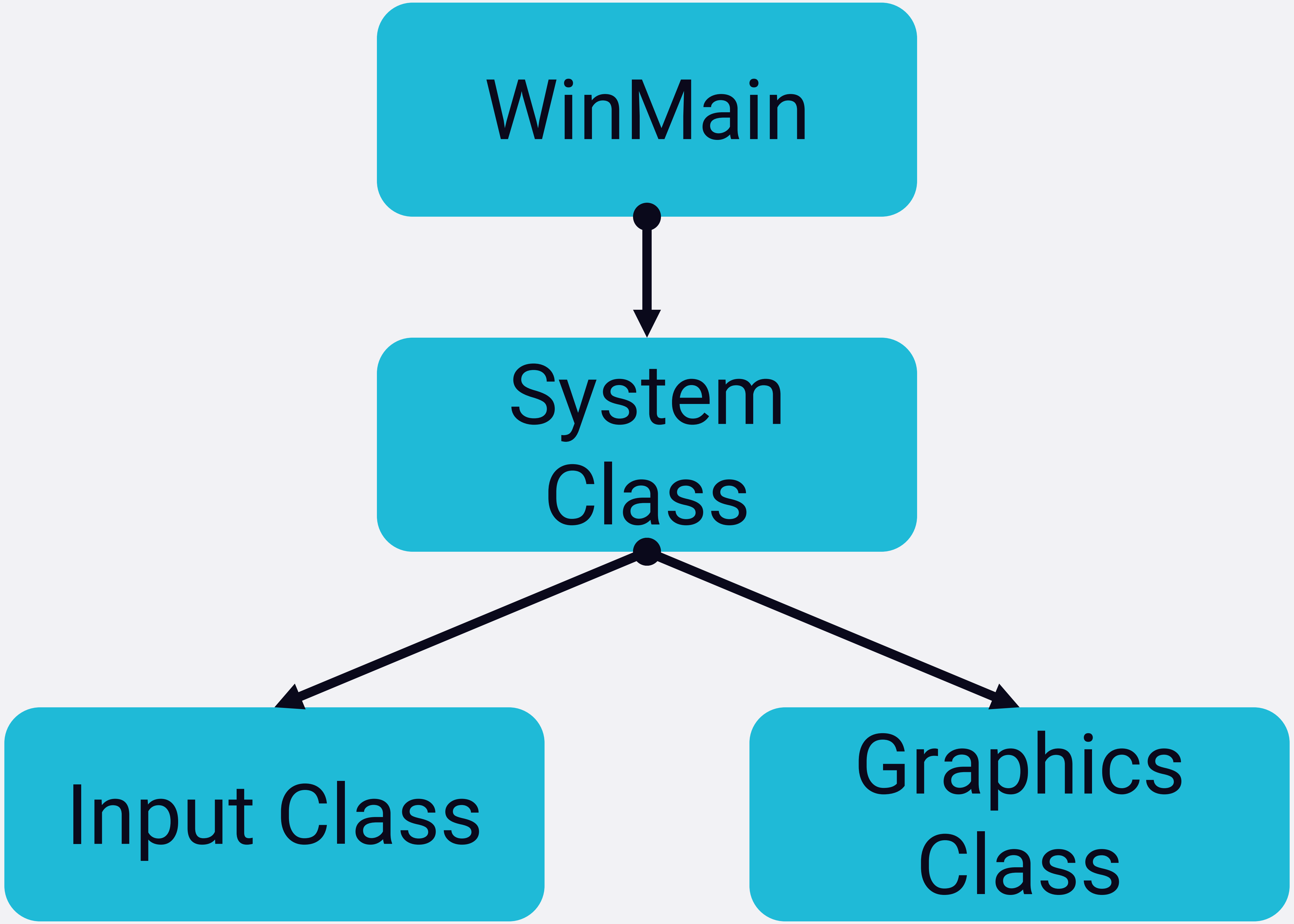
Discreet GPUs

- At least 2 GB one
- There are many options
- But to prepare projects at least one good GPU like ATI 580 or Nvidia 1060 is needed.

Assignment 1.1

Create a framework for all the six practicals.

Main
Systemclass
inputclass


WinMain
System Class
Input Class
Graphics Class

Creating window

A window *class* defines a set of behaviors that several windows might have in common

Window Messages

A GUI application must respond to events from the user and from the operating system

Window Procedure

The **DispatchMessage** function calls the window procedure of the window that is the target of the message.

Painting and Closing

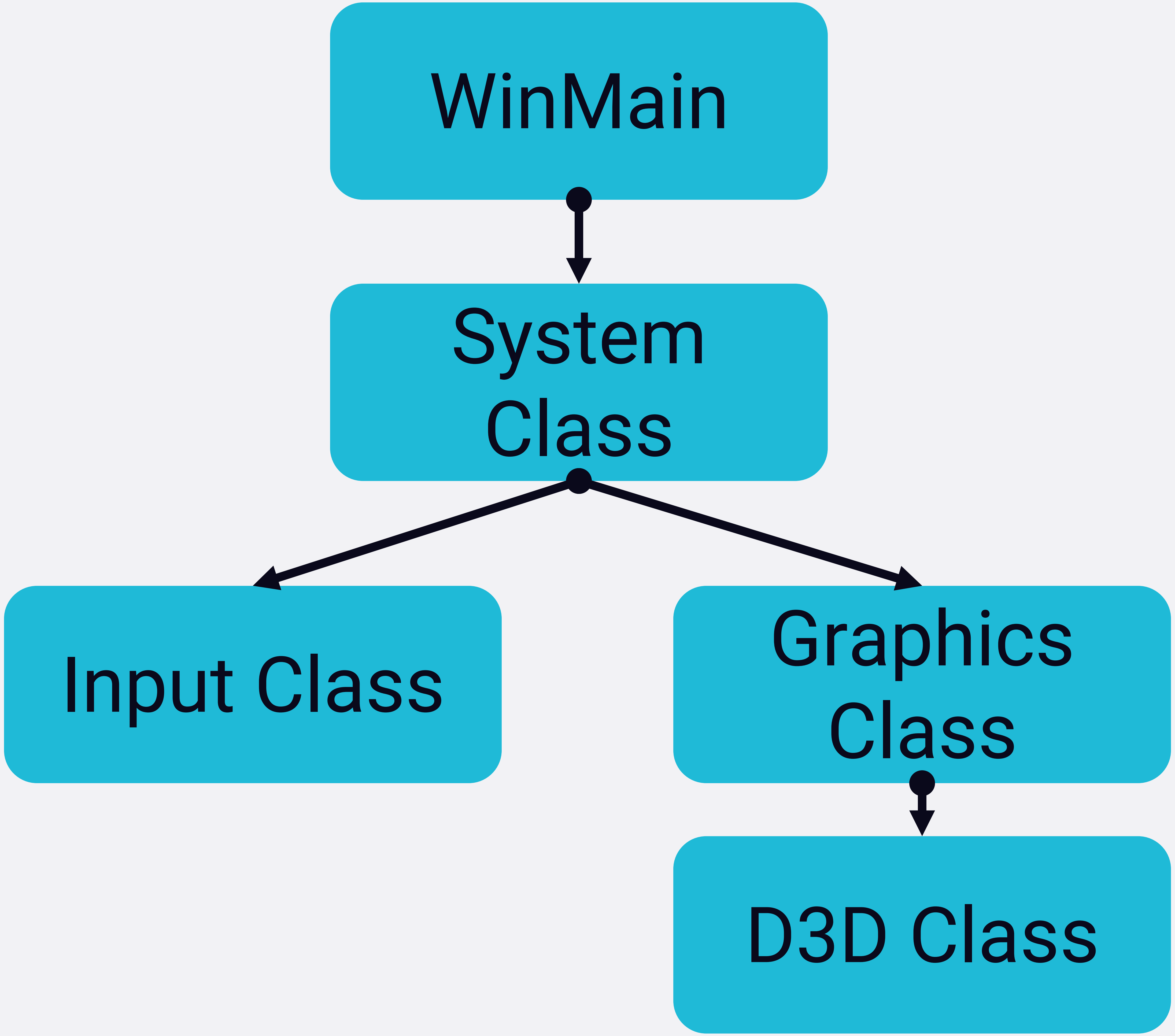
DirectX will paint the Window. ESC. will terminate the application.

```
// Register the window class.
   const wchar_t CLASS_NAME[]  = L"Sample Window Class";

   WNDCLASS wc = { };

   wc.lpfnWndProc   = WindowProc;
   wc.hInstance     = hInstance;
   wc.lpszClassName = CLASS_NAME;

   RegisterClass(&wc);

   // Create the window.

   HWND hwnd = CreateWindowEx(
               0,                              // Optional window styles.
               CLASS_NAME,                     // Window class
               L"Learn to Program Windows",    // Window text
               WS_OVERLAPPEDWINDOW,            // Window style

               // Size and position
               CW_USEDEFAULT, CW_USEDEFAULT, CW_USEDEFAULT, CW_USEDEFAULT,

               NULL,       // Parent window
               NULL,       // Menu
               hInstance,  // Instance handle
               NULL        // Additional application data
      );

   if (hwnd == NULL)
   {
      return 0;
   }

   ShowWindow(hwnd, nCmdShow);
```

Assignment 1.2

Initialize DX Device and paint the window.

Main
System Class
Input Class
Graphics Class
D3D Class


WinMain
System Class
Input Class
Graphics Class
D3D Class

DXGI

Microsoft DirectX Graphics Infrastructure (DXGI) recognizes that some parts of graphics evolve more slowly than others.

Device and Context

Device for creating Resources and Context for all rendering commands.

Swap Chains

a swap chain that encapsulates two or more buffers that are used for rendering and display

RenderTargets

A Buffer that is used to rasterize the pixel data usually a texture.

```
std::vector <IDXGIAdapter*> EnumerateAdapters(void)
{
   IDXGIAdapter * pAdapter;
   std::vector <IDXGIAdapter*> vAdapters;
   IDXGIFactory* pFactory = NULL;


   // Create a DXGIFactory object.
   if(FAILED(CreateDXGIFactory(__uuidof(IDXGIFactory) ,(void**)&pFactory)))
   {
      return vAdapters;
   }


   for ( UINT i = 0;
        pFactory->EnumAdapters(i, &pAdapter) != DXGI_ERROR_NOT_FOUND;
        ++i )
   {
      vAdapters.push_back(pAdapter);
   }


   if(pFactory)
   {
      pFactory->Release();
   }

   return vAdapters;

}
```

Feature Levels

Direct3D 11 devices support a fixed set of feature levels that are defined in the **D3D_FEATURE_LEVEL** enumeration

```
typedef enum D3D_FEATURE_LEVEL
{
        D3D_FEATURE_LEVEL_9_1   ,
        D3D_FEATURE_LEVEL_9_2   ,
        D3D_FEATURE_LEVEL_9_3   ,
        D3D_FEATURE_LEVEL_10_0  ,
        D3D_FEATURE_LEVEL_10_1  ,
        D3D_FEATURE_LEVEL_11_0  ,
        D3D_FEATURE_LEVEL_11_1  ,
        D3D_FEATURE_LEVEL_12_0  ,
        D3D_FEATURE_LEVEL_12_1
} ;
```

```
const D3D_FEATURE_LEVEL lvl[] =
{
        D3D_FEATURE_LEVEL_11_1, D3D_FEATURE_LEVEL_11_0,
        D3D_FEATURE_LEVEL_10_1, D3D_FEATURE_LEVEL_10_0,
        D3D_FEATURE_LEVEL_9_3, D3D_FEATURE_LEVEL_9_2,
        D3D_FEATURE_LEVEL_9_1
};
UINT createDeviceFlags = 0;
#ifdef _DEBUG
createDeviceFlags |= D3D11_CREATE_DEVICE_DEBUG;
#endif

ID3D11Device* device = nullptr;
HRESULT hr = D3D11CreateDeviceAndSwapChain( nullptr,
D3D_DRIVER_TYPE_HARDWARE, nullptr, createDeviceFlags, lvl, _countof(lvl),
D3D11_SDK_VERSION, &sd, &g_pSwapChain, &g_pd3ddevice,
&FeatureLevelsSupported, &g_pImmediateContext );
if ( hr == E_INVALIDARG )
{
hr = D3D11CreateDeviceAndSwapChain( nullptr,
D3D_DRIVER_TYPE_HARDWARE, nullptr, createDeviceFlags, &lvl[1],
_countof(lvl) - 1, D3D11_SDK_VERSION, &sd, &g_pSwapChain, &g_pd3ddevice,
&FeatureLevelsSupported, &g_pImmediateContext );
}

if (FAILED(hr))
return hr;
```

Swap Chain and BackBuffer

Create Swap Chain as per description.
Create rendertarget to render to.
Setup Viewport

```
DXGI_SWAP_CHAIN_DESC sd;
ZeroMemory( &sd, sizeof( sd ) );
sd.BufferCount = 1;
sd.BufferDesc.Width = 640;
sd.BufferDesc.Height = 480;
sd.BufferDesc.Format =
DXGI_FORMAT_R8G8B8A8_UNORM;
sd.BufferDesc.RefreshRate.Numerator = 60;
sd.BufferDesc.RefreshRate.Denominator = 1;
sd.BufferUsage =
DXGI_USAGE_RENDER_TARGET_OUTPUT;
sd.OutputWindow = g_hWnd;
sd.SampleDesc.Count = 1;
sd.SampleDesc.Quality = 0;
sd.Windowed = TRUE;
```

```
ID3D11Texture2D* pBackBuffer;
// Get a pointer to the back buffer
hr = g_pSwapChain->GetBuffer( 0, __uuidof( ID3D11Texture2D ),
( LPVOID* )&pBackBuffer );

// Create a render-target view
g_pd3dDevice->CreateRenderTargetView( pBackBuffer, NULL,
&g_pRenderTargetView );

==============================================================
// Bind the view
g_pImmediateContext->OMSetRenderTargets( 1, &g_pRenderTargetView, NULL );

// Setup the viewport
   D3D11_VIEWPORT vp;
   vp.Width = 640;
   vp.Height = 480;
   vp.MinDepth = 0.0f;
   vp.MaxDepth = 1.0f;
   vp.TopLeftX = 0;
   vp.TopLeftY = 0;
   g_pImmediateContext->RSSetViewports( 1, &vp );

==============================================================
// Setup the color to clear the buffer to.
color[0] = red;
color[1] = green;
color[2] = blue;
color[3] = alpha;

// Clear the back buffer.
m_deviceContext->ClearRenderTargetView(m_renderTargetView, color);
```

change your adapter and color

01

Enum Adapter

Check the adapter and try to enumerate them all.

02

Change BG color

Change the background Color.

03

EndScene()

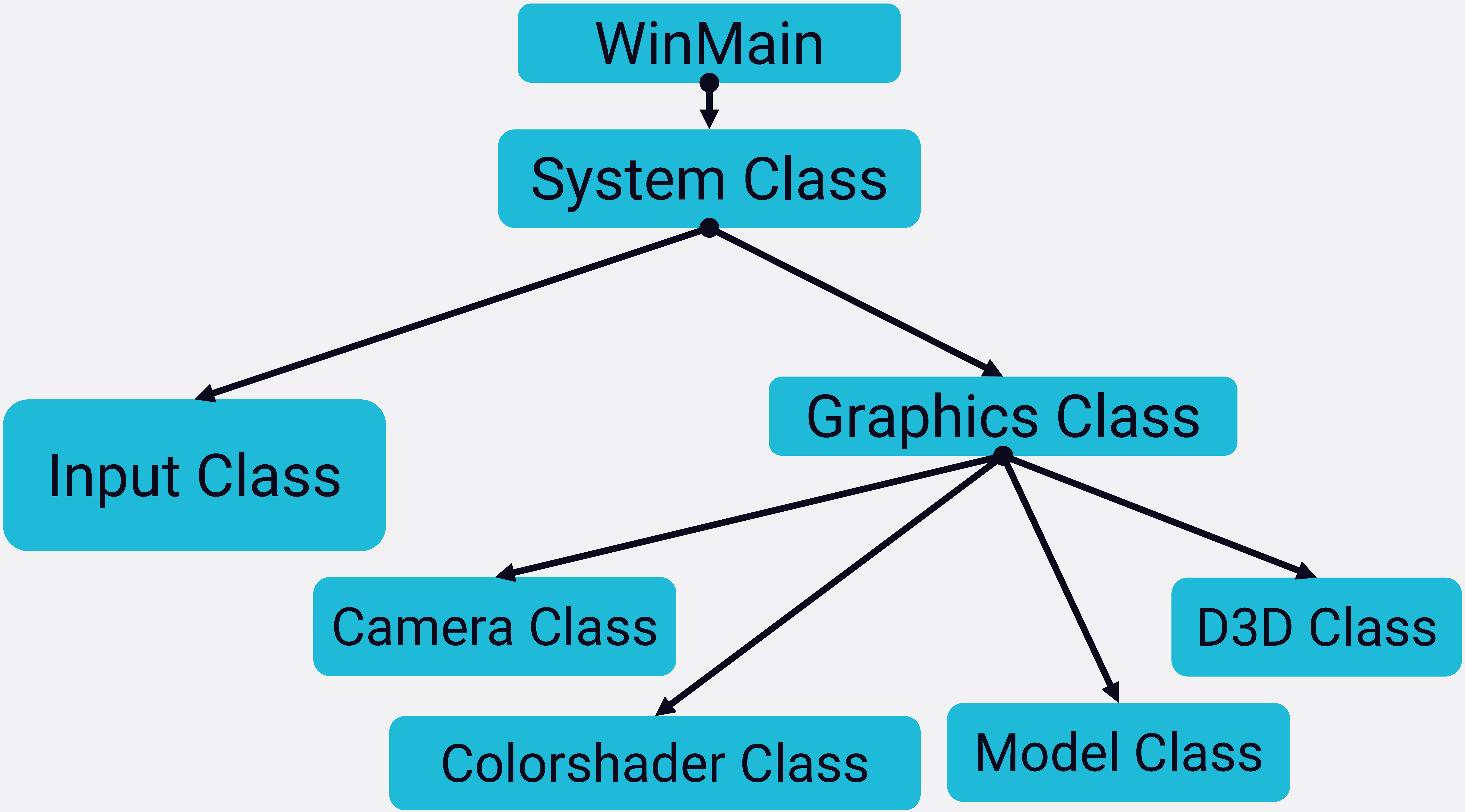
Presenting is important phase mapping.

Anything From Crowd?

Assignment 2

Draw a triangle and render it.

Main
System Class
Input Class
Graphics Class
D3D Class
Model Class
Colorshader Class
Camera Class


WinMain
System Class
Input Class
Graphics Class
Camera Class
D3D Class
Colorshader Class
Model Class

Vertex Buffers

Collection of vertices one needs to render.

Index Buffers

To reduce the memory footprint indices are used.

Buffer Descriptions

All Buffer Resources are described using Buffer Descriptions

SubResource Data

Buffer can hold sub resource data as defined in the Buffer Descriptor Type

```
struct VertexType
{
        XMFLOAT3 position;
        XMFLOAT4 color;
};
==========================================================
vertices = new VertexType[m_vertexCount];
if(!vertices)
        return false;

// Load the vertex array with data.
vertices[0].position = XMFLOAT3(-1.0f, -1.0f, 0.0f); // Bottom left.
vertices[0].color = XMFLOAT4(0.0f, 1.0f, 0.0f, 1.0f);
vertices[1].position = XMFLOAT3(0.0f, 1.0f, 0.0f);  // Top middle.
vertices[1].color = XMFLOAT4(0.0f, 1.0f, 0.0f, 1.0f);
vertices[2].position = XMFLOAT3(1.0f, -1.0f, 0.0f); // Bottom right.
vertices[2].color = XMFLOAT4(0.0f, 1.0f, 0.0f, 1.0f);
==========================================================
// Set up the description of the static vertex buffer.
vertexBufferDesc.Usage = D3D11_USAGE_DEFAULT;
vertexBufferDesc.ByteWidth = sizeof(VertexType) * m_vertexCount;
vertexBufferDesc.BindFlags = D3D11_BIND_VERTEX_BUFFER;
vertexBufferDesc.CPUAccessFlags = 0;
vertexBufferDesc.MiscFlags = 0;
vertexBufferDesc.StructureByteStride = 0;

// Give the subresource structure a pointer to the vertex data.
 vertexData.pSysMem = vertices;
 vertexData.SysMemPitch = 0;
 vertexData.SysMemSlicePitch = 0;

// Now create the vertex buffer.
result = device->CreateBuffer(&vertexBufferDesc, &vertexData, &m_vertexBuffer);
```

Vertex Shaders

HLSL program to transform vertices from world to projected space.

Pixel Shader

A small HLSL program for Rasterization operation.

Input Layout

A description of a single element for the input-assembler stage.

Constant Buffers

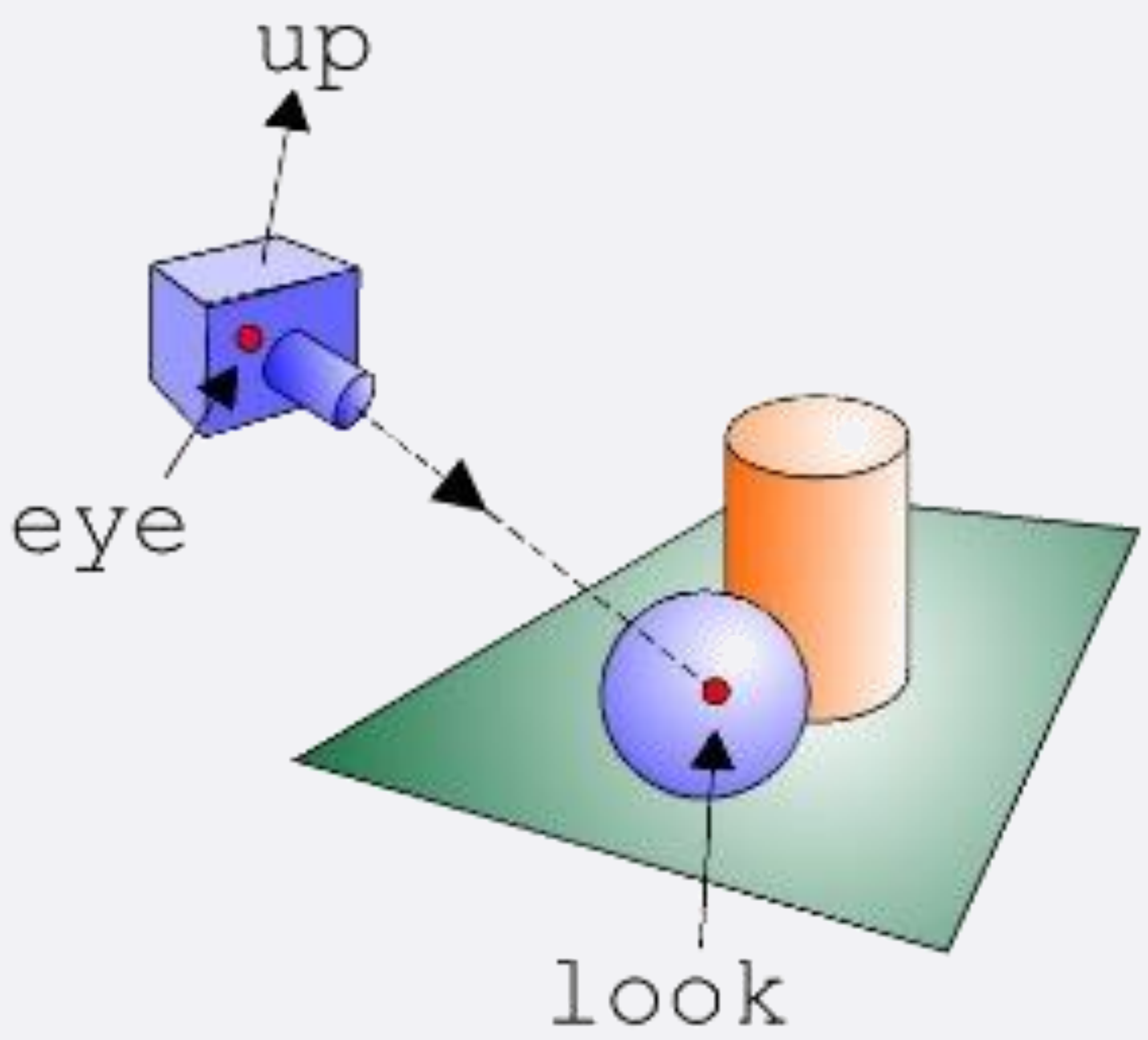
The buffers whose memory footprints remain constant in the Video Memory.

```
cbuffer MatrixBuffer
{
			matrix worldMatrix;
			matrix viewMatrix;
			matrix projectionMatrix;
};

struct VertexInputType
{
   float4 position : POSITION;
   float4 color : COLOR;
};

struct PixelInputType
{
   float4 position : SV_POSITION;
   float4 color : COLOR;
};

PixelInputType ColorVertexShader(VertexInputType input)
{
   PixelInputType output;


			// Change the position vector to be 4 units for proper matrix calculations.
   input.position.w = 1.0f;

			// Calculate the position of the vertex against the world, view, and projection matrices.
   output.position = mul(input.position, worldMatrix);
   output.position = mul(output.position, viewMatrix);
   output.position = mul(output.position, projectionMatrix);

			// Store the input color for the pixel shader to use.
   output.color = input.color;

   return output;
}
==========================================================================
struct PixelInputType
{
   float4 position : SV_POSITION;
   float4 color : COLOR;
};

float4 ColorPixelShader(PixelInputType input) : SV_TARGET
{
   return input.color;
}
```

Preparing your Camera

```
lookAt(vec3 eye, vec3 look, vec3 up)
```

- Typical maths library function
- Returns `mat4`
- Sets camera position
- Point at target
- <u>Careful</u> with “up” unit vector
- Not ideal for full 3d rotation



Changing MSAA, colors of vertices and changing transformations

01 Change MSAA

How DirectX 11 Handles Anti Aliasing.

02 Change Vertices color

How Rendering Pipeline renders pixels.

03 Transofrmation()

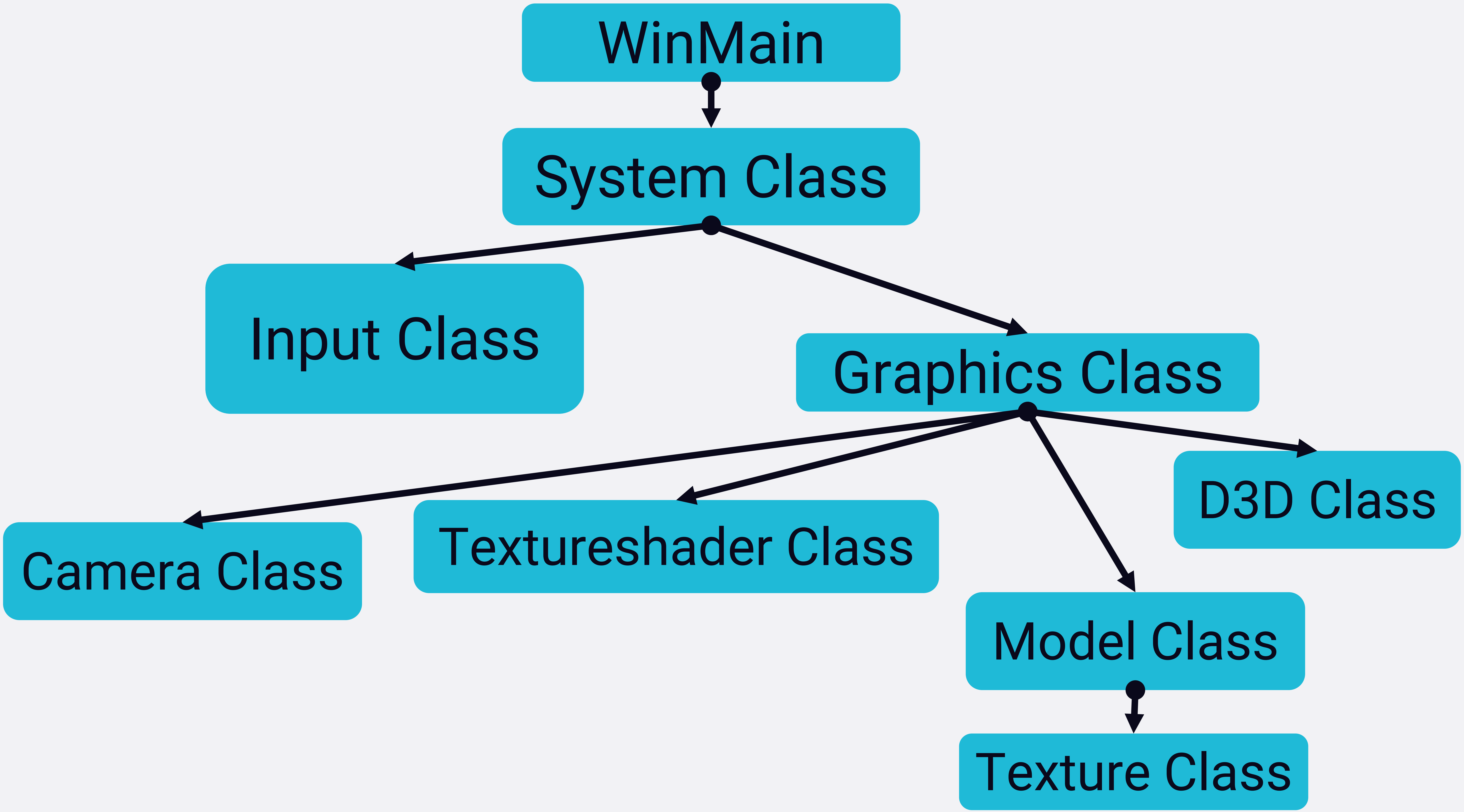
Translation + Rotation


Is Crowd
still Alive?

Assignment 3

Texture the triangle.

Main
System Class
Input Class
Graphics Class
D3D Class
Model Class
Texture Class
Textureshader Class
Camera Class


WinMain
System Class
Input Class
Graphics Class
Camera Class
Textureshader Class
D3D Class
Model Class
Texture Class

Loading Textures

Collect Raw RGBA DAta

Texture Description

Describe the Texture Resource

Sampler State

How you gonna sample your texture

Texture Coords

This is new input Layout Element

```
// Setup the description of the texture.
	textureDesc.Height = height;
	textureDesc.Width = width;
	textureDesc.MipLevels = 0;
	textureDesc.ArraySize = 1;
	textureDesc.Format = DXGI_FORMAT_R8G8B8A8_UNORM;
	textureDesc.SampleDesc.Count = 1;
	textureDesc.SampleDesc.Quality = 0;
	textureDesc.Usage = D3D11_USAGE_DEFAULT;
	textureDesc.BindFlags = D3D11_BIND_SHADER_RESOURCE | D3D11_BIND_RENDER_TARGET;
	textureDesc.CPUAccessFlags = 0;
	textureDesc.MiscFlags = D3D11_RESOURCE_MISC_GENERATE_MIPS;

	// Create the empty texture.
	hResult = device->CreateTexture2D(&textureDesc, NULL, &m_texture);
	if(FAILED(hResult))
		return false;

	// Set the row pitch of the targa image data.
	rowPitch = (width * 4) * sizeof(unsigned char);

	// Copy the targa image data into the texture.
	deviceContext->UpdateSubresource(m_texture, 0, NULL, m_targaData, rowPitch, 0);

	// Setup the shader resource view description.
	srvDesc.Format = textureDesc.Format;
	srvDesc.ViewDimension = D3D11_SRV_DIMENSION_TEXTURE2D;
	srvDesc.Texture2D.MostDetailedMip = 0;
	srvDesc.Texture2D.MipLevels = -1;

	// Create the shader resource view for the texture.
	hResult = device->CreateShaderResourceView(m_texture, &srvDesc, &m_textureView);
	if(FAILED(hResult))
	{
		return false;
	}

	// Generate mipmaps for this texture.
	deviceContext->GenerateMips(m_textureView);
```

Change Filtering
change tex Coords

01 Change Filtering Mode

From Point To Linear To Anisotropic.

02 Change Tex Coords

How to map texture on to the vertices.


Any Questions?

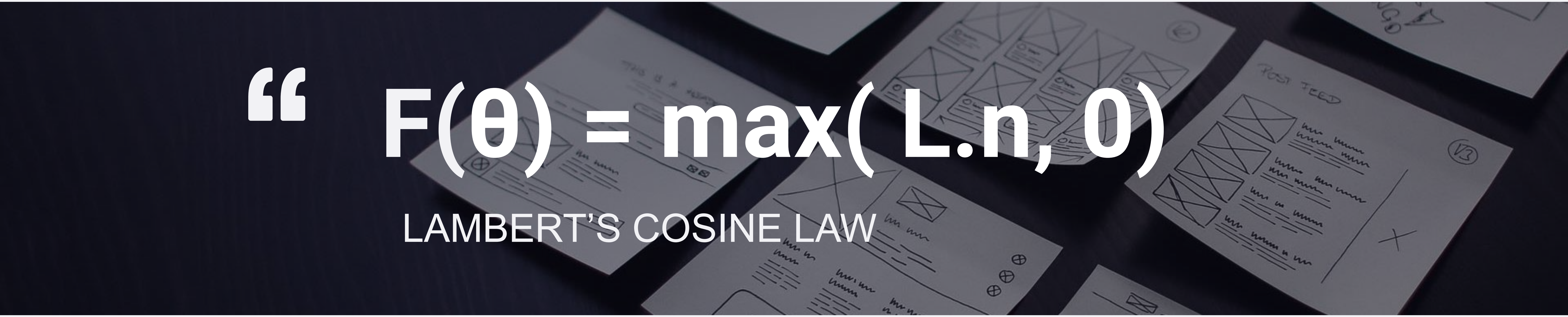
Assignment 4

Diffuse Directional Light.

Main
System Class
Input Class
Graphics Class
D3D Class
Model Class
Texture Class
Light Class
Lightshader Class
Camera Class

Diffuse directional lighting
Lamberts Law

Light that strikes a surface point head-on is more intense than light that just glances a surface point; Consider a small shaft of incoming light with cross-sectional area dA. So the idea is to come up with a function that returns different intensities based on the alignment of the vertex normal and the light vector. (Observe that the light vector is the vector from the surface to the light source; that is, it is aimed in the opposite direction the light rays travel.) The function should return maximum intensity when the vertex normal and light vector are perfectly aligned (i.e., the angle θ between them is 0°), and it should smoothly diminish in intensity as the angle between the vertex normal and light vector increases. If θ > 90°, then the light strikes the back of a surface and so we set the intensity to zero. Lambert's Cosine Law gives the function we seek, which is given by



“ $F(\theta) = \max(L.n, 0)$

LAMBERT'S COSINE LAW

```
struct VertexType
{
	XMFLOAT3 position;
	XMFLOAT2 texture;
	XMFLOAT3 normal;
};
```

```
// Sample the pixel color from the texture using the sampler at this texture coordinate location.
textureColor = shaderTexture.Sample(SampleType, input.tex);

// Invert the light direction for calculations.
lightDir = -lightDirection;

// Calculate the amount of light on this pixel.
lightIntensity = saturate(dot(input.normal, lightDir));

// Determine the final amount of diffuse color based on the diffuse color combined with the light intensity.
color = saturate(diffuseColor * lightIntensity);

// Multiply the texture pixel and the final diffuse color to get the final pixel color result.
color = color * textureColor;
```

Change Diffuse Color
Change Direction

01

Change Diffuse Color

You may change the look.

02

Change Direction

This will change the intensities.

03

Create moon shade

Varying normal of sphere can have the real effect..

Assignment 5

Specular Directional Light.

Main

System Class

Input Class

Graphics Class

D3D Class

Model Class

Texture Class

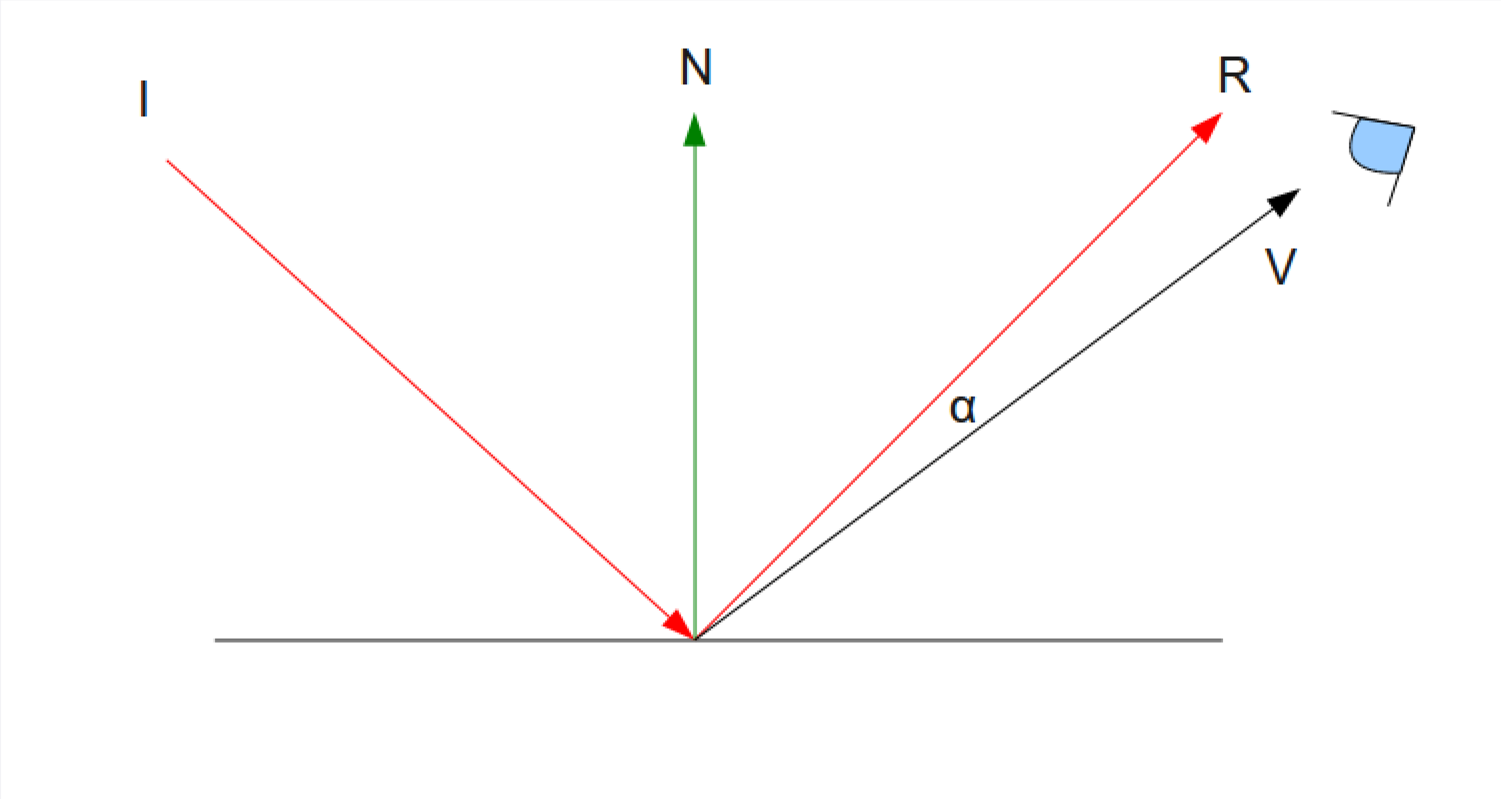
Light Class

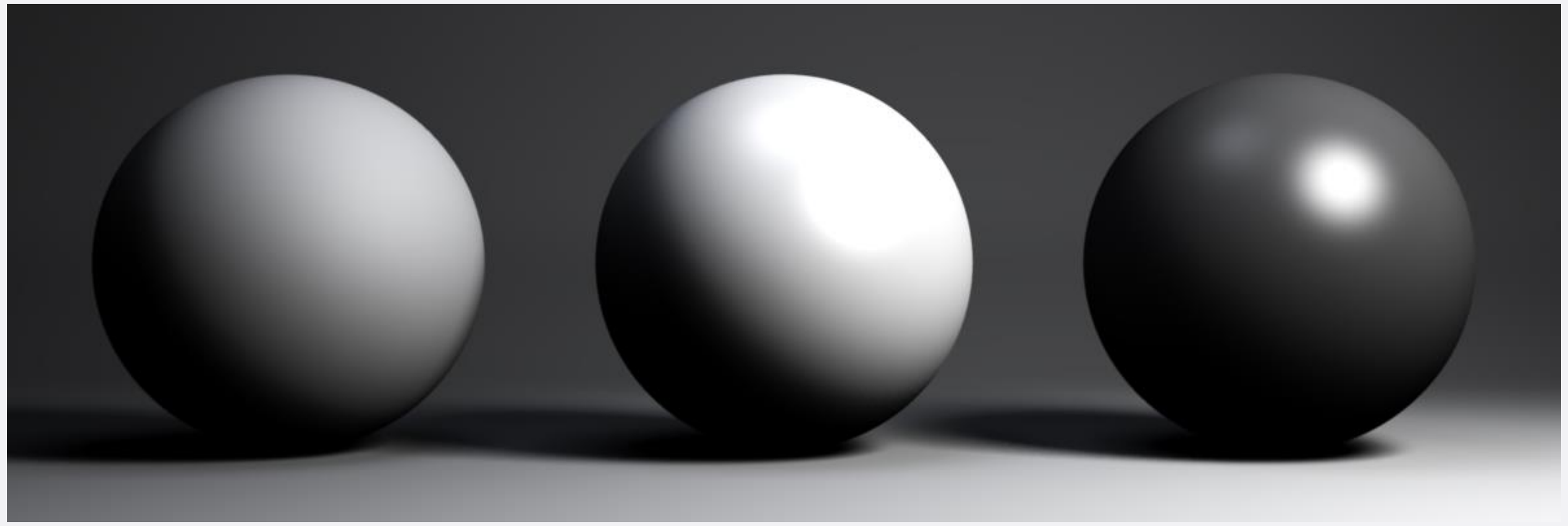
Lightshader Class

Camera Class

Specular lighting
Reflectance from viewpoint

When light strikes a smooth surface, the light rays reflect sharply in a general direction through a cone of reflectance; this is called a specular reflection. In contrast to diffuse light, specular light might not travel into the eye because it reflects in a specific direction; the specular lighting calculation is viewpoint dependent. This means that as the eye moves about the scene, the amount of specular light it receives will change.





Change specular power
Change Depth buffer

01

Change specular power

Specular power increases the intensity and the cone.

02

Change Depth

For this tutorial if depth buffer is not implemented we lose the depth.

Assignment 6

Data From Model Files

OBJ

FBX

3DS

COLLADA

Vertex Count: 36

Data:

-1.0	1.0	-1.0	0.0	0.0	0.0	0.0	-1.0
1.0	1.0	-1.0	1.0	0.0	0.0	0.0	-1.0
-1.0	-1.0	-1.0	0.0	1.0	0.0	0.0	-1.0
-1.0	-1.0	-1.0	0.0	1.0	0.0	0.0	-1.0
1.0	1.0	-1.0	1.0	0.0	0.0	0.0	-1.0
1.0	-1.0	-1.0	1.0	1.0	0.0	0.0	-1.0
1.0	1.0	-1.0	0.0	0.0	1.0	0.0	0.0
1.0	1.0	1.0	1.0	0.0	1.0	0.0	0.0
1.0	-1.0	-1.0	0.0	1.0	1.0	0.0	0.0
1.0	-1.0	-1.0	0.0	1.0	1.0	0.0	0.0
1.0	1.0	1.0	1.0	0.0	1.0	0.0	0.0
1.0	-1.0	1.0	1.0	1.0	1.0	0.0	0.0
1.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0
-1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0
1.0	-1.0	1.0	0.0	1.0	0.0	0.0	1.0
1.0	-1.0	1.0	0.0	1.0	0.0	0.0	1.0
-1.0	1.0	1.0	1.0	0.0	0.0	0.0	1.0
-1.0	-1.0	1.0	1.0	1.0	0.0	0.0	1.0
-1.0	1.0	1.0	0.0	0.0	-1.0	0.0	0.0
-1.0	1.0	-1.0	1.0	0.0	-1.0	0.0	0.0
-1.0	-1.0	1.0	0.0	1.0	-1.0	0.0	0.0
-1.0	-1.0	1.0	0.0	1.0	-1.0	0.0	0.0
-1.0	1.0	-1.0	1.0	0.0	-1.0	0.0	0.0
-1.0	-1.0	-1.0	1.0	1.0	-1.0	0.0	0.0
-1.0	1.0	1.0	0.0	0.0	0.0	1.0	0.0
1.0	1.0	1.0	1.0	0.0	0.0	1.0	0.0
-1.0	1.0	-1.0	0.0	1.0	0.0	1.0	0.0
-1.0	1.0	-1.0	0.0	1.0	0.0	1.0	0.0
1.0	1.0	1.0	1.0	0.0	0.0	1.0	0.0
1.0	1.0	-1.0	1.0	1.0	0.0	1.0	0.0
-1.0	-1.0	-1.0	0.0	0.0	0.0	-1.0	0.0
1.0	-1.0	-1.0	1.0	0.0	0.0	-1.0	0.0
-1.0	-1.0	1.0	0.0	1.0	0.0	-1.0	0.0
-1.0	-1.0	1.0	0.0	1.0	0.0	-1.0	0.0
1.0	-1.0	-1.0	1.0	0.0	0.0	-1.0	0.0
1.0	-1.0	1.0	1.0	1.0	0.0	-1.0	0.0


Too Much for the day.

thank you.