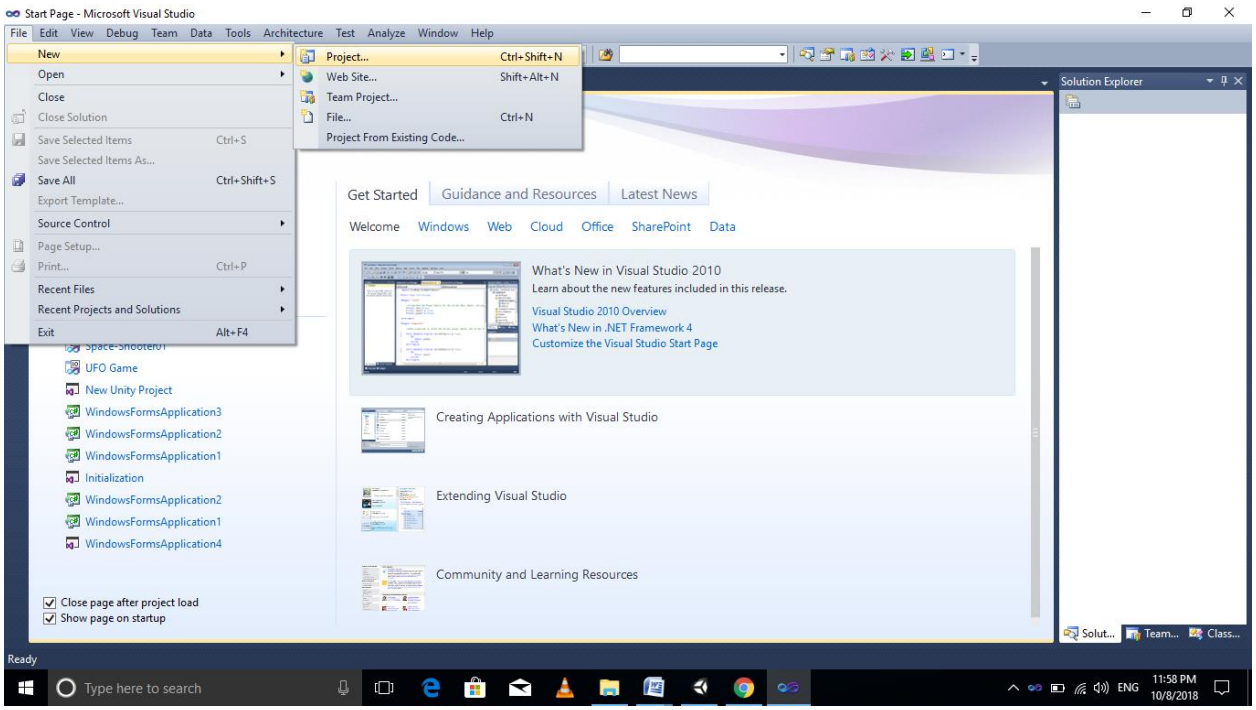
https://devblogs.microsoft.com/cppblog/directx-game-development-with-c-in-visual-studio/

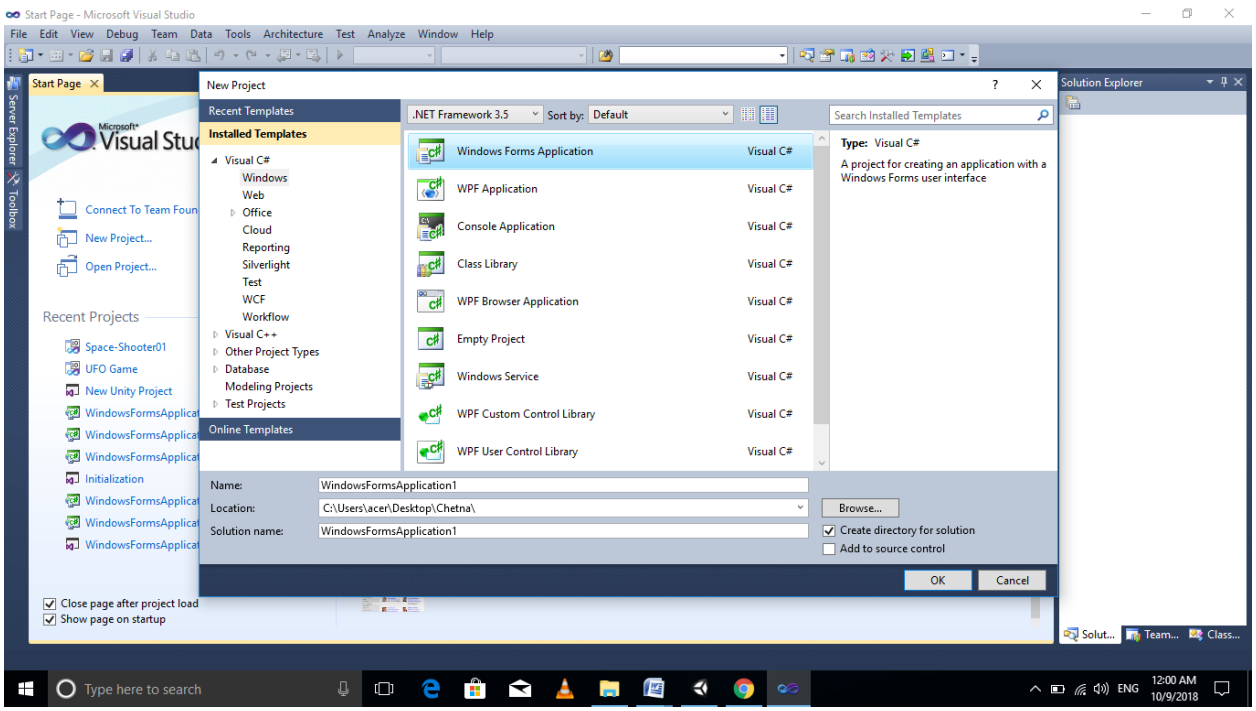
Start Visual Studio 2017

Click on Create Project

Under Installed templates select Visual C# and Select Windows Forms Application from the right panel

Give the name to the project as “DXPract1” and click OK





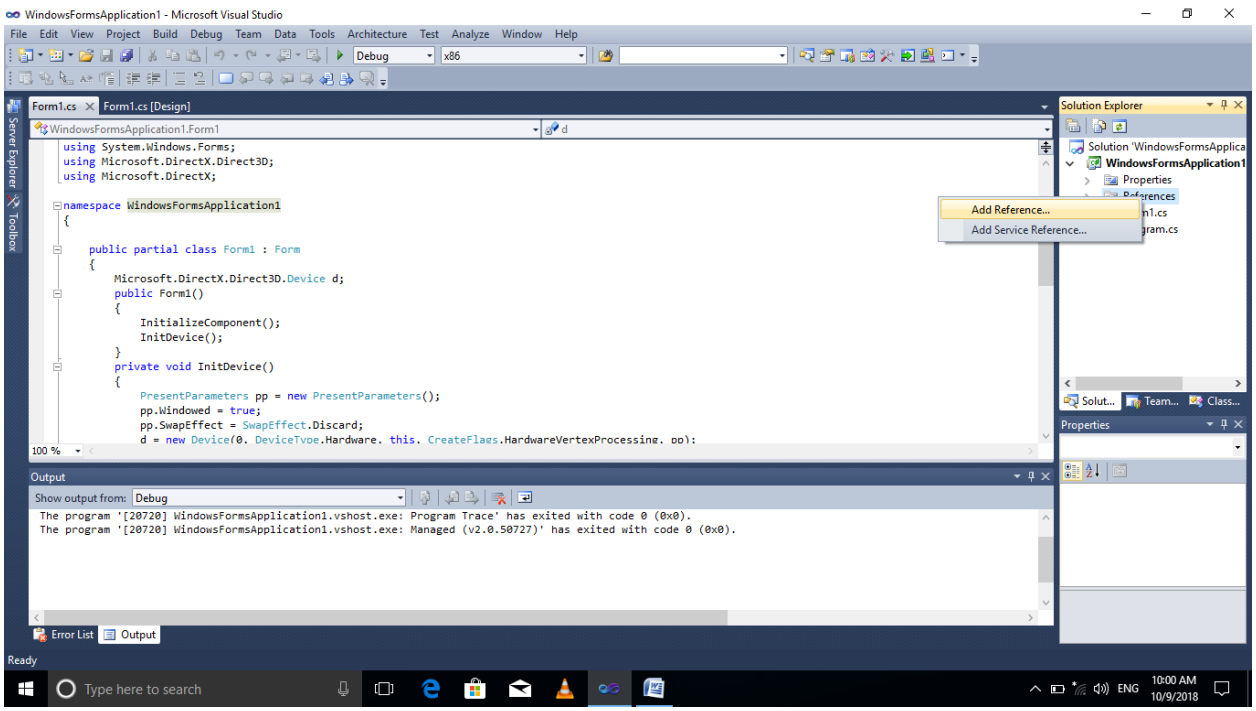
Right click on the Form and select View Code or Press F7

Now in the Solution Explorer window on the right side, right click on References and select Add Reference

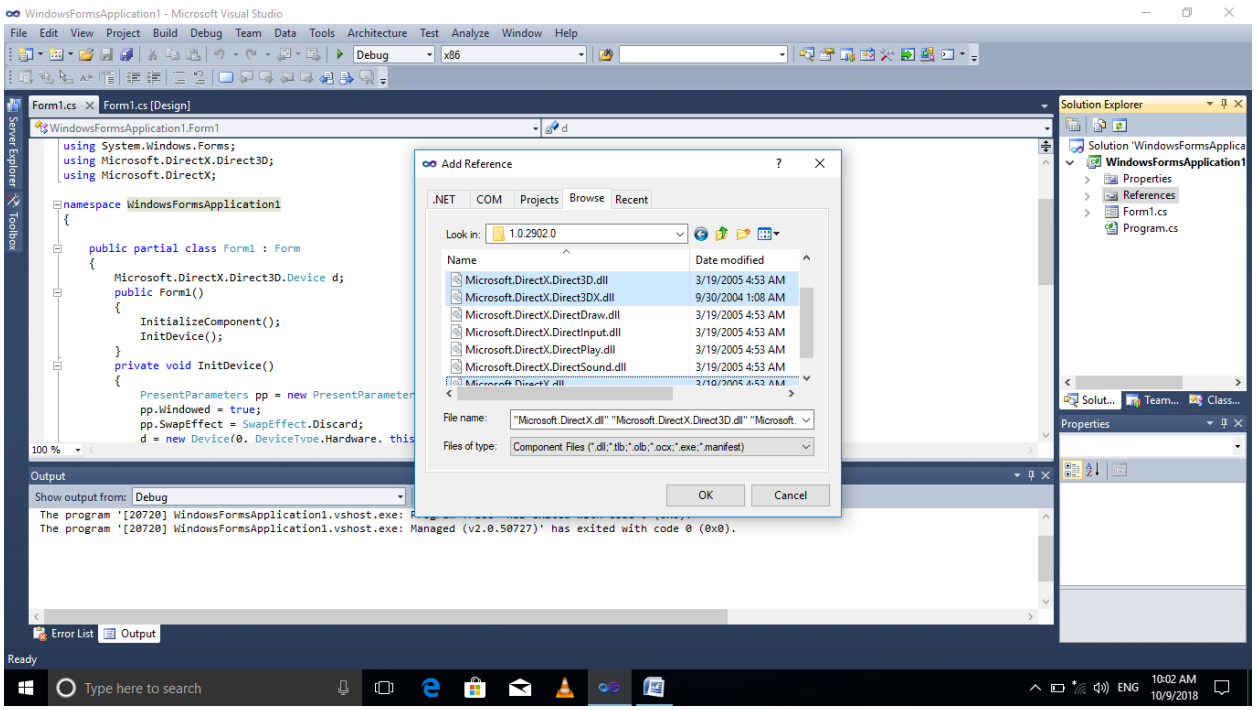
Click on Browse and go to C Drive→Windows→Microsoft .Net→DirectX for Managed Code→1.0.2902.0

Select Microsoft.DirectX.Direct3D.dll , Microsoft.DirectX.Direct3DX.dll and Microsoft.DirectX.dll

Again Click on Browse and go to C Drive→Windows→Microsoft .Net→Framework→ v2.0.50727

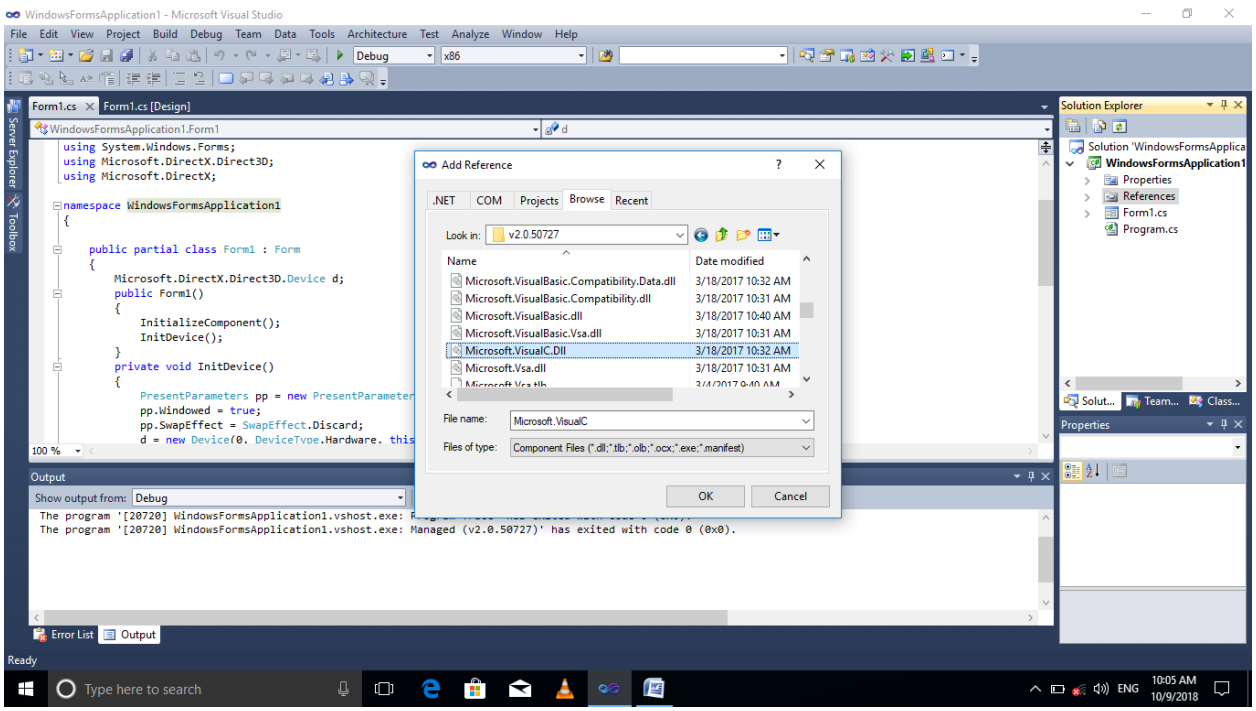


Browse→C Drive→Windows→Microsoft .Net→DirectX for Managed Code→1.0.2902.0



Add one more Reference

Browse→C Drive→Windows→Microsoft .Net→Framework→ v2.0.50727



Select Microsoft.VisualC.Dll

Now go to Design window and in Properties window of the Form set the following properties:-

Size: 800,600

Text: Direct X Practical 1

Now click on Events button in the Properties window, search for Paint event. Double click on it. It will open its code in the code window and call the Render() method in it

You have to specify that the program is a windows program so go to Project-> Project Properties. Put the output type as Windows Application.

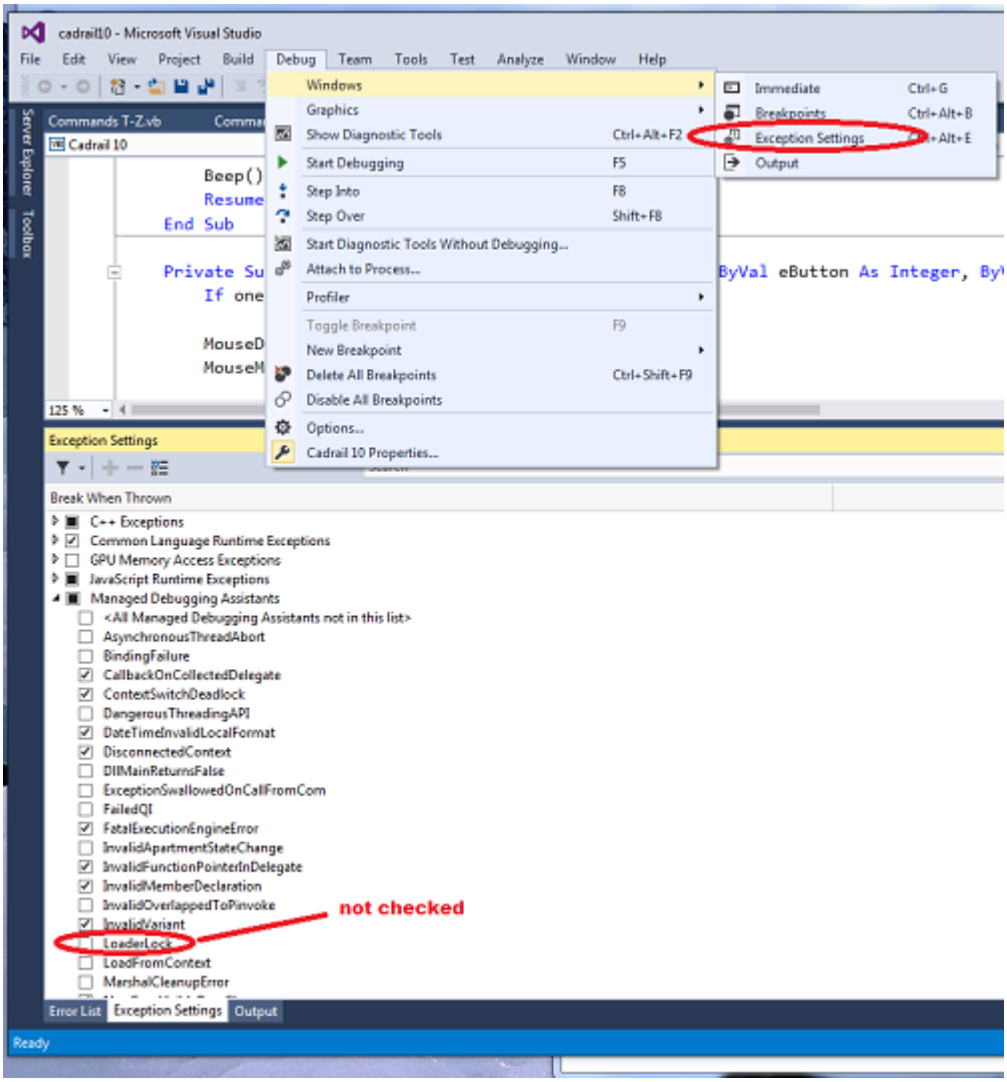
go to Project-> Project Properties, select Debug, Check “Enable native code debugging”

changing the 'configuration' node on my **app.config** from this:
go to your project folder C:\Users\Admin\source\repos\DXpract1\DXpract1 DXpract1 and open App.config file in visual studio and make changes as follows

```
<startup>
  <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.0"/>
</startup>
```
to this:

```
<startup useLegacyV2RuntimeActivationPolicy="true">
  <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.0"/>
</startup>
```

Go to Debug->Windows->Exception Settings, under managed debugging assistant, uncheck “LoaderLock”



Now type the code in the code window

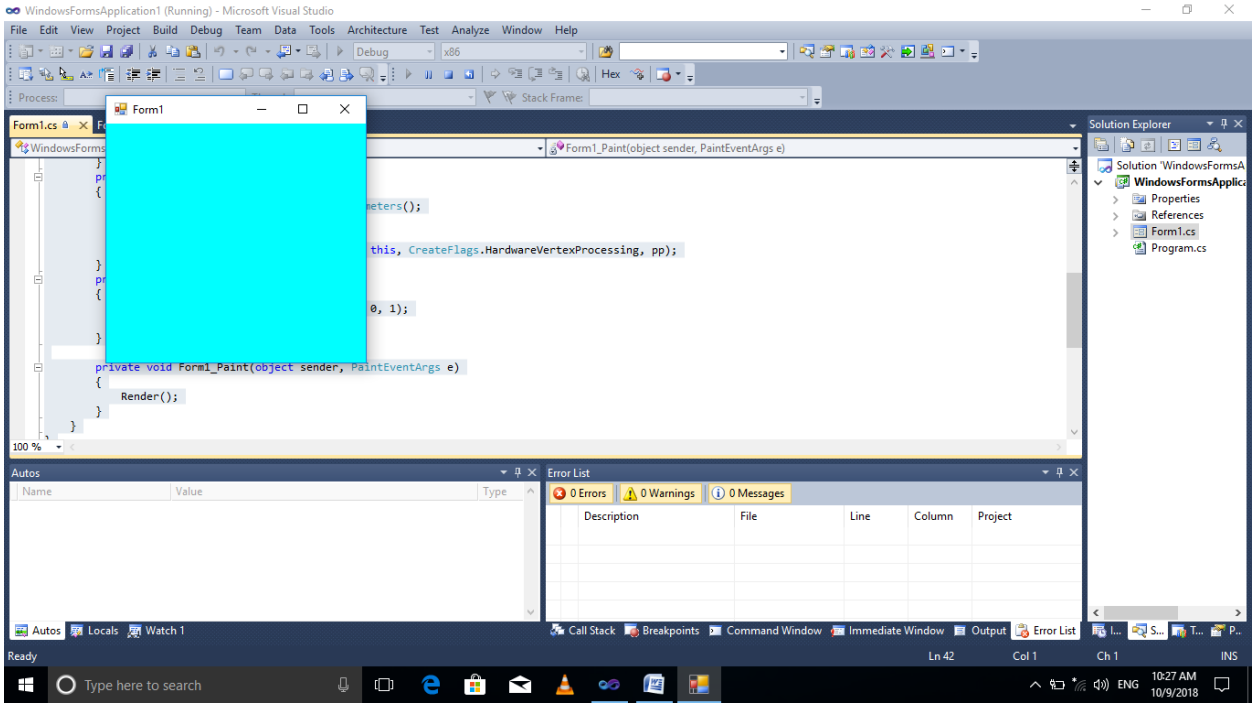
Practical No. 1

Aim: Windows Framework and Initialize Direct3D device.

Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using Microsoft.DirectX.Direct3D;
using Microsoft.DirectX;

namespace WindowsFormsApplication1
{

    public partial class Form1 : Form
    {
        Microsoft.DirectX.Direct3D.Device d;
        public Form1()
        {
            InitializeComponent();
            InitDevice();
        }
        private void InitDevice()
        {

PresentParameters pp = new PresentParameters();
            pp.Windowed = true;     // Specify that it will be in a window
            pp.SwapEffect = SwapEffect.Discard;
        // After the current screen is drawn, it will be automatically deleted from memory
            d = new Device(0, DeviceType.Hardware, this,
CreateFlags.HardwareVertexProcessing, pp);
        }
        private void Render()
        {
            d.Clear(ClearFlags.Target, Color.Aqua, 0, 1);
            d.Present();
        }

        private void Form1_Paint(object sender, PaintEventArgs e)
        {
            Render();
        }
    }
}
```

Output:


WindowsFormsApplication1 (Running) - Microsoft Visual Studio
Form1
Form1.cs
Form1_Paint(object sender, PaintEventArgs e)
private void Form1_Paint(object sender, PaintEventArgs e)
{
Render();
}
Solution Explorer
Properties
References
Form1.cs
Program.cs
Autos
Name
Value
Type
Error List
0 Errors
0 Warnings
0 Messages
Description
File
Line
Column
Project
Call Stack
Breakpoints
Command Window
Immediate Window
Output
Error List
Ready
Ln 42
Col 1
Ch 1
INS
Type here to search
ENG
10:27 AM
10/9/2018

Practical No. 2

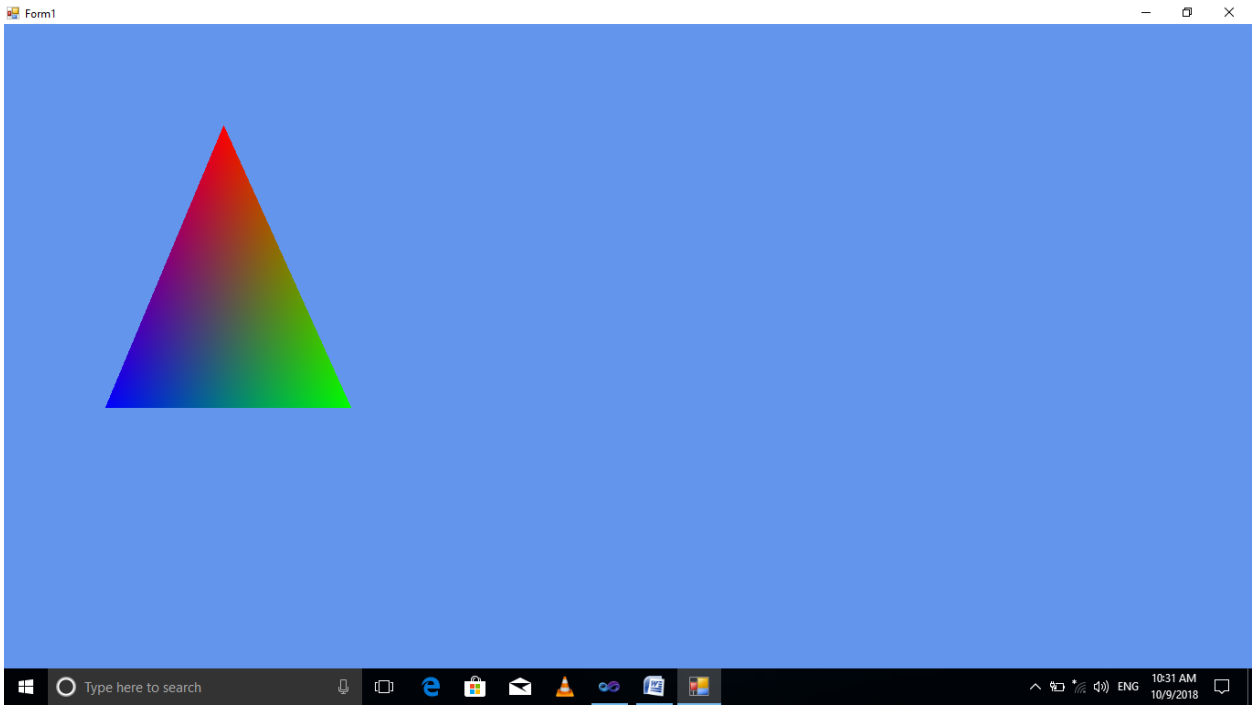
Aim: Drawing Triangle using Direct3D

Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using Microsoft.DirectX.Direct3D;
using Microsoft.DirectX;

namespace WindowsFormsApplication2
{
    public partial class Form1 : Form
    {
        Microsoft.DirectX.Direct3D.Device d;
        public Form1()
        {
            InitializeComponent();
            InitDevice();
        }
        private void InitDevice()
        {
            PresentParameters pp = new PresentParameters();
            pp.Windowed = true;
            pp.SwapEffect = SwapEffect.Discard;
            d = new Device(0, DeviceType.Hardware, this,
CreateFlags.HardwareVertexProcessing, pp);
        }
        private void Render()
        {
            CustomVertex.TransformedColored[] v = new CustomVertex.TransformedColored[3];

            v[0].Position = new Vector4(240, 110, 0, 1);
            v[0].Color = System.Drawing.Color.FromArgb(255, 0, 0).ToArgb();

            v[1].Position = new Vector4(380, 420, 0, 1);
            v[1].Color = System.Drawing.Color.FromArgb(0, 255, 0).ToArgb();

            v[2].Position = new Vector4(110, 420, 0, 1);
            v[2].Color = System.Drawing.Color.FromArgb(0, 0, 255).ToArgb();

            d.Clear(ClearFlags.Target, Color.CornflowerBlue, 0, 1);
            d.BeginScene();
            d.VertexFormat = CustomVertex.TransformedColored.Format;
            d.DrawUserPrimitives(PrimitiveType.TriangleList, 1, v);
            d.EndScene();
            d.Present();
        }
```

```
        private void Form1_Paint(object sender, PaintEventArgs e)
        {
            Render();
        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }
    }

}
```

Output:



Practical No. 3

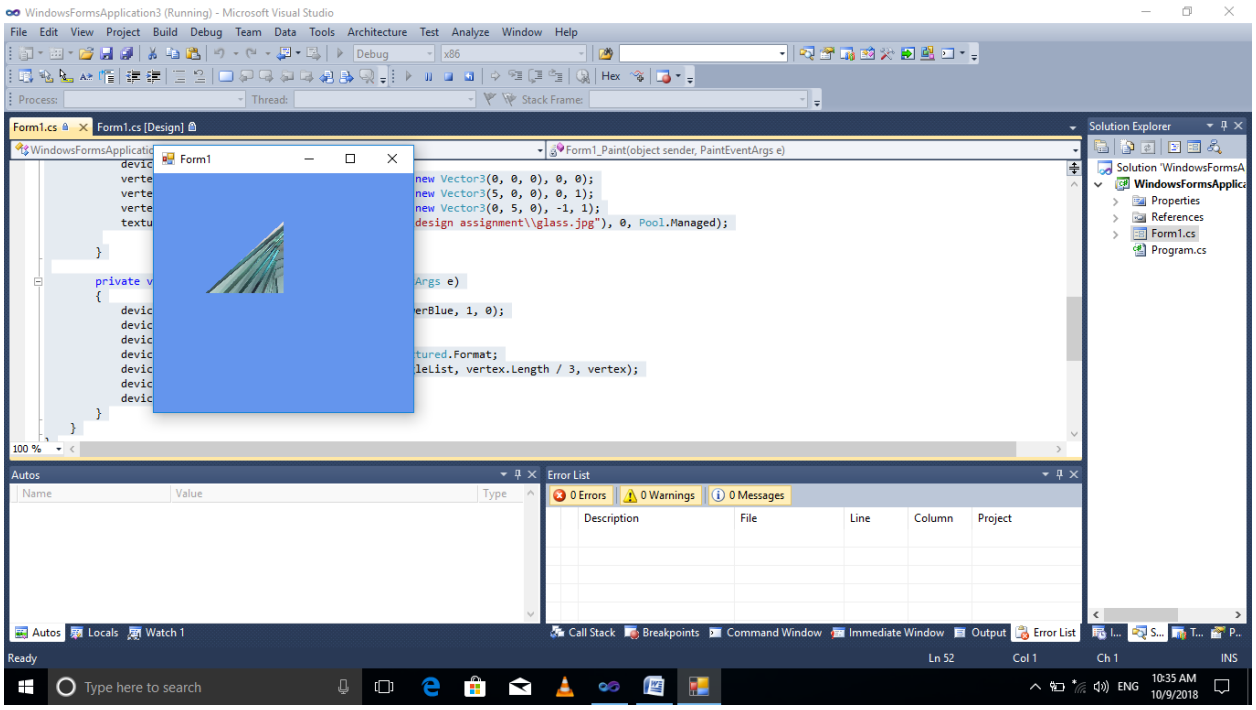
Aim : Texture and triangle using Direct3D

Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using Microsoft.DirectX;
using Microsoft.DirectX.Direct3D;

namespace WindowsFormsApplication3
{
    public partial class Form1 : Form
    {
        private Microsoft.DirectX.Direct3D.Device device;
        private CustomVertex.PositionTextured[] vertex = new
CustomVertex.PositionTextured[3];
        private Texture texture;
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            PresentParameters pp = new PresentParameters();
            pp.Windowed = true;
            pp.SwapEffect = SwapEffect.Discard;
            device = new Device(0, DeviceType.Hardware, this,
CreateFlags.HardwareVertexProcessing, pp);
            device.Transform.Projection = Matrix.PerspectiveFovLH(3.14f / 4,
device.Viewport.Width / device.Viewport.Height, 1f, 1000f);
            device.Transform.View = Matrix.LookAtLH(new Vector3(0, 0, 20), new Vector3(),
new Vector3(0, 1, 0));
            device.RenderState.Lighting = false;
            vertex[0] = new CustomVertex.PositionTextured(new Vector3(0, 0, 0), 0, 0);
            vertex[1] = new CustomVertex.PositionTextured(new Vector3(5, 0, 0), 0, 1);
            vertex[2] = new CustomVertex.PositionTextured(new Vector3(0, 5, 0), -1, 1);
            texture = new Texture(device, new Bitmap("E:\\glass.jpg"), 0, Pool.Managed);

        }

        private void Form1_Paint(object sender, PaintEventArgs e)
        {
            device.Clear(ClearFlags.Target, Color.CornflowerBlue, 1, 0);
            device.BeginScene();
            device.SetTexture(0, texture);
            device.VertexFormat = CustomVertex.PositionTextured.Format;
            device.DrawUserPrimitives(PrimitiveType.TriangleList, vertex.Length / 3,
vertex);
```

```
            device.EndScene();
            device.Present();
        }
    }
}
```

Output:



Practical No. 4

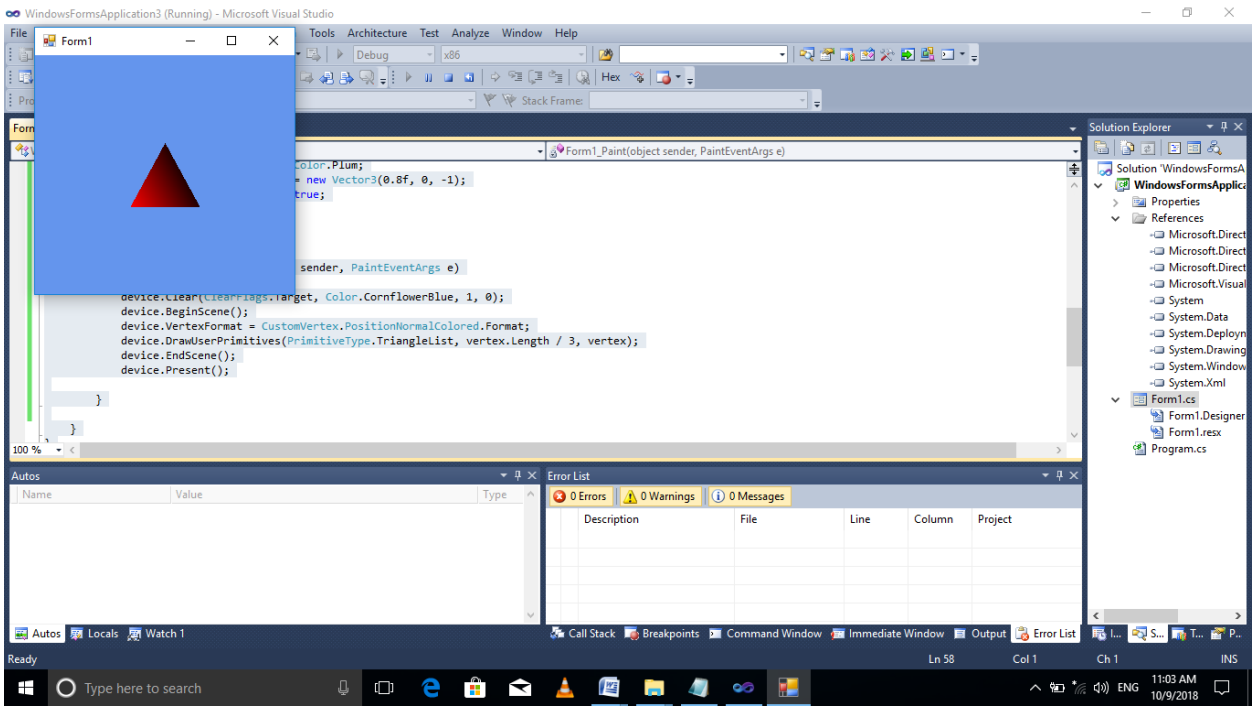
Aim: Programmable diffuse lightning using Direct3D.

Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using Microsoft.DirectX;
using Microsoft.DirectX.Direct3D;


namespace WindowsFormsApplication3
{
    public partial class Form1 : Form
    {
        private Microsoft.DirectX.Direct3D.Device device;
        private CustomVertex.PositionNormalColored[] vertex = new
CustomVertex.PositionNormalColored[3];

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            PresentParameters pp = new PresentParameters();
            pp.Windowed = true;
            pp.SwapEffect = SwapEffect.Discard;
            device = new Device(0, DeviceType.Hardware, this,
CreateFlags.HardwareVertexProcessing, pp);
            device.Transform.Projection = Matrix.PerspectiveFovLH(3.14f / 4,
device.Viewport.Width / device.Viewport.Height, 1f, 1000f);
            device.Transform.View = Matrix.LookAtLH(new Vector3(0, 0, 10), new Vector3(),
new Vector3(0, 1, 0));
            device.RenderState.Lighting = false;
            vertex[0] = new CustomVertex.PositionNormalColored(new Vector3(0, 1, 1), new
Vector3(1, 0, 1), Color.Red.ToArgb());
            vertex[1] = new CustomVertex.PositionNormalColored(new Vector3(-1, -1, 1),
new Vector3(1, 0, 1), Color.Red.ToArgb());
            vertex[2] = new CustomVertex.PositionNormalColored(new Vector3(1, -1, 1), new
Vector3(-1, 0, 1), Color.Red.ToArgb());
            device.RenderState.Lighting = true;
            device.Lights[0].Type = LightType.Directional;
            device.Lights[0].Diffuse = Color.Plum;
            device.Lights[0].Direction = new Vector3(0.8f, 0, -1);
            device.Lights[0].Enabled = true;


        }
```

```
        private void Form1_Paint(object sender, PaintEventArgs e)
        {
            device.Clear(ClearFlags.Target, Color.CornflowerBlue, 1, 0);
            device.BeginScene();
            device.VertexFormat = CustomVertex.PositionNormalColored.Format;
            device.DrawUserPrimitives(PrimitiveType.TriangleList, vertex.Length / 3,
vertex);
            device.EndScene();
            device.Present();

        }

    }
}
```

Output:



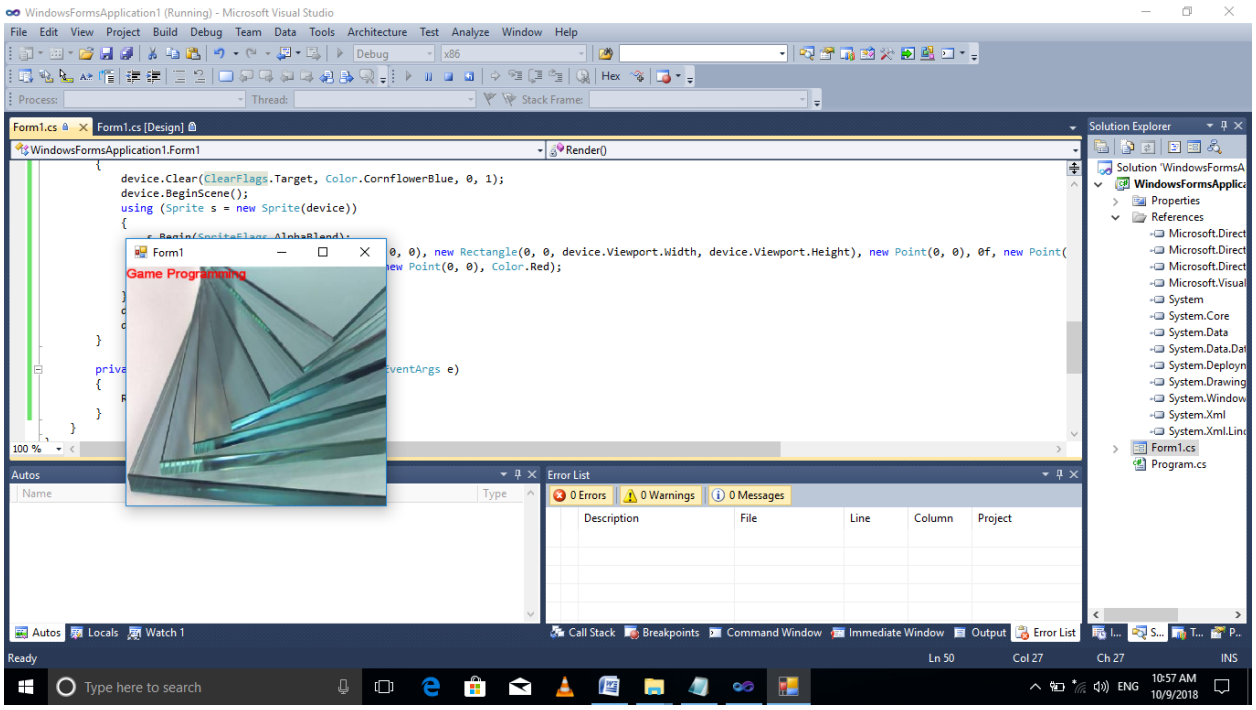
Practical No. 5

Aim: Loading Models into DirectX

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using Microsoft.DirectX;
using Microsoft.DirectX.Direct3D;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        private Microsoft.DirectX.Direct3D.Device device;
        private Microsoft.DirectX.Direct3D.Texture texture;
        private Microsoft.DirectX.Direct3D.Font font;

        public Form1()
        {
            InitializeComponent();
            InitDevice();
            InitFont();
            LoadTexture();
        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }
        private void InitFont()
        {
            System.Drawing.Font f = new System.Drawing.Font("Arial", 16f,
FontStyle.Regular);
            font = new Microsoft.DirectX.Direct3D.Font(device, f);
        }
        private void LoadTexture()
        {
            texture = TextureLoader.FromFile(device, "E:\\glass.jpg", 400, 400, 1, 0,
Format.A8B8G8R8, Pool.Managed, Filter.Point, Filter.Point, Color.Transparent.ToArgb());
        }
        private void InitDevice()
        {
            PresentParameters pp = new PresentParameters();
            pp.Windowed = true;
            pp.SwapEffect = SwapEffect.Discard;
            device = new Device(0, DeviceType.Hardware, this,
CreateFlags.HardwareVertexProcessing, pp);
        }
        private void Render()
        {
            device.Clear(ClearFlags.Target, Color.CornflowerBlue, 0, 1);
```

```
            device.BeginScene();
            using (Sprite s = new Sprite(device))
            {
                s.Begin(SpriteFlags.AlphaBlend);
                s.Draw2D(texture, new Rectangle(0, 0, 0, 0), new Rectangle(0, 0,
device.Viewport.Width, device.Viewport.Height), new Point(0, 0), 0f, new Point(0, 0),
Color.White);
                font.DrawText(s, "Game Programming", new Point(0, 0), Color.Red);
                s.End();
            }
            device.EndScene();
            device.Present();
        }

        private void Form1_Paint(object sender, PaintEventArgs e)
        {
            Render();
        }
    }
}
```

Output:



DirectX Classes and Methods used in above practicals

PresentParameters Class

Describes the presentation parameters.

DeviceWindow	Retrieves or sets the display window.
SwapEffect	Retrieves or sets the swap effect.
Windowed	Boolean value that indicates whether an application is running in a windowed mode.

Device Class

Generate a Scene

public Device(int, DeviceType, Control, CreateFlags, PresentParameters[]);

Device.Clear Method ()

Clears the viewport or a set of rectangles in the viewport to a specified RGBA color, clears the depth buffer, and erases the stencil buffer.

public void Clear(ClearFlags, Color, float, int);

Device.Present Method ()

Presents the display with the contents of the next buffer in the sequence of back buffers owned by the device.

public void Present();

CustomVertex Class

Defines various custom fixed-format vertex types. This class is a container of structures.

CustomVertex.TransformedColored Structure

Describes a custom vertex format structure that contains transformed vertices and color information.

Device.DrawUserPrimitives(PrimitiveType,Int32,Object) Method

Renders data specified by a user memory pointer as a sequence of geometric primitives of the specified type.

Device.VertexFormat Property

Retrieves or sets the supported flexible vertex formats.

Device.BeginScene() Method

Begins a scene.

Device.EndScene() Method

Ends a scene that was started by calling the Device.BeginScene method.

Matrix.PerspectiveFovLH(Single,Single,Single,Single) Method

Builds a left-handed perspective projection matrix based on a field of view.

CustomVertex.PositionTextured Structure

Describes a custom vertex format structure that contains position and one set of texture coordinates.

public PositionTextured(Vector3, float, float);

CustomVertex.PositionNormalColored Structure

Describes a custom vertex format structure that contains position, color, and normal data.

Device.RenderState Property

Retrieves a render-state value for a device.

RenderStateManager.Lighting Property

Enables or disables Microsoft Direct3D lighting.

TextureLoader.FromFile Method ()

Creates a texture from a file.

Sprite.Begin(SpriteFlags) Method (Microsoft.DirectX.Direct3D)

- Draw A Sprite

Sprite.Draw2D Method ()

Adds a sprite to the list of batched sprites. Used for presentation in 2-D space.

Sprite.Draw2D(Texture,Rectangle,Rectangle,Point,Single,Point,Color) Method