MASTER ELECTRONIC DESIGN

Homework 1(T2_1)

Miguel Tlapa Juárez

06/04/2014

This document describes the system architecture and design about the body controller module, it's have block diagram and flowchart to describe software and hardware architecture.

Revision History

Date	Revision Number	Author/Editor	Modifications
June2014	0.1	Miguel Tlapa	Created file

Disclaimers

OBJECTIVES:

Modify the Winmain.CPP including the next changes:

I add some variables like that:

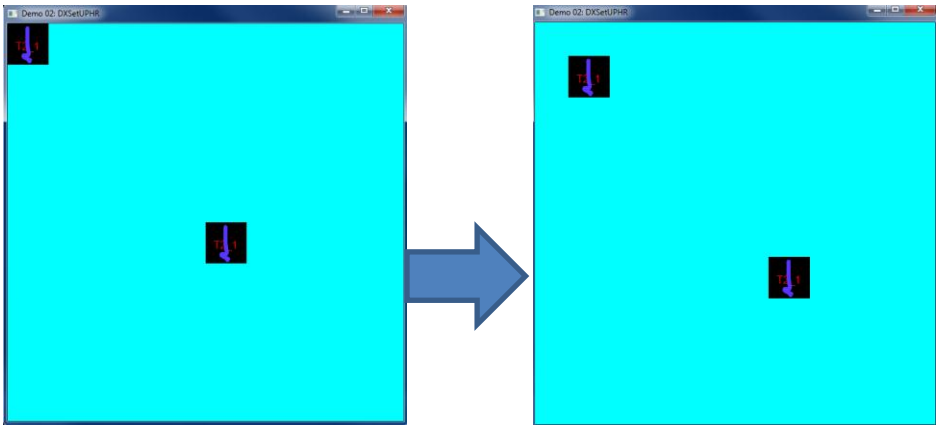
```
class TestApp : public DXApp
{
public:
    TestApp(HINSTANCE hInstance);
    ~TestApp();

    bool Init() override;
    void Update(float dt) override;
    void Render(float dt) override;

private:
    std::unique_ptr<DirectX::SpriteBatch> spriteBatch;
    ID3D11ShaderResourceView *m_pTexture;
    Sprite* sprite;
    int change_color = 0;       // Variable that permit to control the background color
    int change_full_screen = 0; // Variable that use to control the full screen
    int show_hide_sprite = 0;   // Variable that use to show and hide the sprite
    float pos_x = 30;           // Variable that use to define the min value of pos_x
    float pos_y = 30;           // Variable that use to define the min value of pos_Y
};
```

A) **In the Method TestApp:Update**

Ask if there is a key pressing:

keyA = The center of the sprite will move in some position inner of the main window.

```
if (GetAsyncKeyState('A'))
{

    m_Viewport[0].TopLeftX = 50;                    // Defino la Posicion Top LEFTX del Viewport [0]
    m_Viewport[0].TopLeftY = 50;                    // Defino la Posicion Top LEFTY del Viewport [0]
    //m_Viewport[0].Width = static_cast<float> (300);  // Defino el tamano del Ancho del Viewport [0]
    //m_Viewport[0].Height = static_cast<float> (300); // Defino el tamano del Alto del Viewport [0]


    m_Viewport[1].TopLeftX = 350;
    m_Viewport[1].TopLeftY = 350;
    //m_Viewport[1].Width = static_cast<float> (300);
    //m_Viewport[1].Height = static_cast<float> (300);


}
```



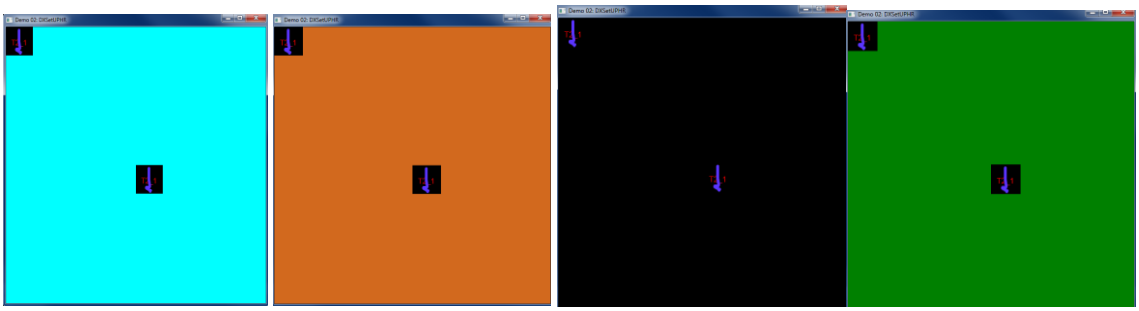
Key B = The background color changes (Cyan, Chocolate,Black, Green)

First I increment the variable change_color in the Method TestApp:Update

```
if (GetAsyncKeyState('B'))
{


    //change_color = ~change_color;
    change_color++;

    if (change_color == 4)
    {
        change_color = 0;
    }
```

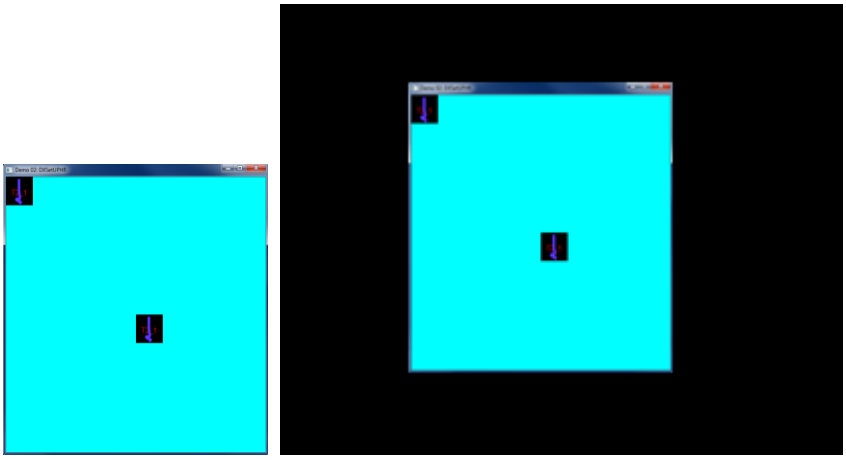
After I changed the color in the Method TestApp:Render

```
void TestApp::Render(float dt)
{

    if (change_color == 0)
    {
        m_pImmediateContext->ClearRenderTargetView(m_pRenderTargetView, DirectX::Colors::Cyan);// No cambiar su posicion

    }

    if (change_color == 1)
    {
        m_pImmediateContext->ClearRenderTargetView(m_pRenderTargetView, DirectX::Colors::Chocolate);// No cambiar su posicion
    }

    if (change_color == 2)
    {
        m_pImmediateContext->ClearRenderTargetView(m_pRenderTargetView, DirectX::Colors::Black);// No cambiar su posicion
    }

    if (change_color == 3)
    {
        m_pImmediateContext->ClearRenderTargetView(m_pRenderTargetView, DirectX::Colors::Green);// No cambiar su posicion

    }
```



Key C = Switch between FullScreen and Windowed

```
if (GetAsyncKeyState('C'))
{

    if (change_full_screen == 0)
    {
        m_pSwapChain->SetFullscreenState(TRUE, NULL);
        change_full_screen = ~change_full_screen;
    }

    else
    {
        m_pSwapChain->SetFullscreenState(FALSE, NULL);
        change_full_screen = ~change_full_screen;
    }



}
```



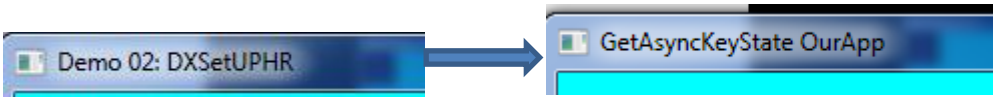
Key D = Print the Feature Level as the Title of the Window

In included the next libraries

```
#include "DXApp.h"
#include "Sprite.h"
#include <string.h>
#include <sstream>
```

```
class TestApp : public DXApp
{
public:
    TestApp(HINSTANCE hInstance);
    ~TestApp();

    bool Init() override;
    void Update(float dt) override;
    void Render(float dt) override;

private:
    std::unique_ptr<DirectX::SpriteBatch> spriteBatch;
    ID3D11ShaderResourceView *m_pTexture;
    Sprite* sprite;
```

```
if (GetAsyncKeyState('D'))
{
    std::ostringstream outs;
    outs << "GetAsyncKeyState OurApp";
    SetWindowText(m_hAppWnd, outs.str().c_str());

}
```



B) Use two Viewport

I add the array in DXApp.h

```
protected:
    //win32 attributes
    HWND        m_hAppWnd;
    HINSTANCE   m_AppInstance;
    UINT        m_ClientWidth;
    UINT        m_ClientHeight;

    std::string m_appTitle;
    DWORD       m_WndStyle;

    //DirectX attributes

    ID3D11Device*           m_pDevice;
    ID3D11DeviceContext*    m_pImmediateContext;
    IDXGISwapChain*         m_pSwapChain;
    ID3D11RenderTargetView* m_pRenderTargetView;
    D3D_DRIVER_TYPE         m_DriverType;
    D3D_FEATURE_LEVEL       m_FeatureLevel;
    //D3D11_VIEWPORT            m_Viewport;
    D3D11_VIEWPORT          m_Viewport[2]; // DXApp.h
```

And Initialize the Viewport in the DXapp.cpp

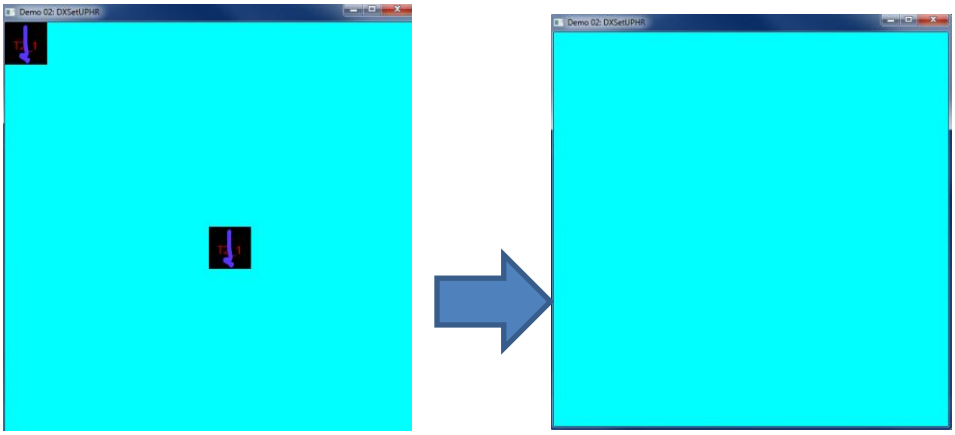
```
#pragma region 3. INITIALIZE THE VIEWPORT
    //viewport creation
    // Original Start
    //m_Viewport.Width = static_cast<float> (m_ClientWidth);
    //m_Viewport.Height = static_cast<float> (m_ClientHeight);
    //m_Viewport.TopLeftX = 0;
    //m_Viewport.TopLeftY = 0;
    //m_Viewport.MinDepth = 0.0f;
    //m_Viewport.MaxDepth = 1.0f;
    // Original EnD
    //Bind Viewport

    m_Viewport[0].Width = static_cast<float> (300);
    m_Viewport[0].Height = static_cast<float> (300);
    m_Viewport[0].TopLeftX = 0;
    m_Viewport[0].TopLeftY = 0;
    m_Viewport[0].MinDepth = 0.0f;
    m_Viewport[0].MaxDepth = 1.0f;

    m_Viewport[1].Width = static_cast<float> (300);
    m_Viewport[1].Height = static_cast<float> (300);
    m_Viewport[1].TopLeftX = 300;
    m_Viewport[1].TopLeftY = 300;
    m_Viewport[1].MinDepth = 0.0f;
    m_Viewport[1].MaxDepth = 1.0f;
```

I updated the viewport in the Winmain:: TestRender()

```
m_pImmediateContext->RSSetViewports(1, &m_Viewport[0]);

if (show_hide_sprite == 0)
    {

    sprite->setPosition(DirectX::SimpleMath::Vector2(pos_x, pos_y));
    spriteBatch->Begin();
    sprite->Draw(spriteBatch.get());
    spriteBatch->End();
    }

m_pImmediateContext->RSSetViewports(1, &m_Viewport[1]);

if (show_hide_sprite == 0)
{
    sprite->setPosition(DirectX::SimpleMath::Vector2(pos_x, pos_y));
    spriteBatch->Begin();
    sprite->Draw(spriteBatch.get());
    spriteBatch->End();
}
```

C) Change the Sprite

Key W = Show / hide the Sprite



I add the next instruction TestApp:Update

```
if (GetAsyncKeyState('W'))
{

    show_hide_sprite = ~show_hide_sprite;


}
```

I asked if the show_hide_sprite = 0 in the TestApp:Render

```
if (show_hide_sprite == 0)
    {

    sprite->setPosition(DirectX::SimpleMath::Vector2(pos_x, pos_y));
    spriteBatch->Begin();
    sprite->Draw(spriteBatch.get());
    spriteBatch->End();
    }

m_pImmediateContext->RSSetViewports(1, &m_Viewport[1]);

if (show_hide_sprite == 0)
{
    sprite->setPosition(DirectX::SimpleMath::Vector2(pos_x, pos_y));
    spriteBatch->Begin();
    sprite->Draw(spriteBatch.get());
    spriteBatch->End();
}

m_pSwapChain->Present(0, 0);// No cambiar su posicion
```

Key R= move the Sprite to the Right
Key L= move the Sprite to the Left
Key S= move the Sprite to the Down
Key U= move the Sprite to the UP

```
if (GetAsyncKeyState('R'))
{
    pos_x = pos_x + 1;

    if (pos_x == 200)
    {
        pos_x = pos_x -1;

    }

}

if (GetAsyncKeyState('L'))
{

    if (pos_x == 30)
    {
        pos_x = pos_x + 1;

    }
    pos_x = pos_x - 1;


}

if (GetAsyncKeyState('S'))
{
    pos_y = pos_y + 1;
    if (pos_y == 200)
    {
        pos_y = pos_y - 1;

    }


}

if (GetAsyncKeyState('U'))
{
    if (pos_y == 30)
    {
        pos_y = pos_y + 1;

    }
    pos_y = pos_y - 1;
```