**Final Project Write Up**

**10 pts - (10% of final project grade)**

In this Write Up, you will summarize and describe the implementation of each of the below topics within the Final Project

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Objective:**

* Demonstrate understanding of topics within the project

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***Grade:***

* *Summary of each topic (.5 pts each)*
  + *For each topic, summarize what you have learned about that topic from class/labs/textbook/doc reading.*
* *Implementation description of each topic (.5 pts each)*
  + *For each topic, describe how you are using that topic in the implementation of your project. Use detail.*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |
| --- | --- | --- |
| **Topic** | **Summary** | **Implementation** |
| Windows Programming | **Windows programming implies creating two functions. WinMain which registers a window object, instantiate a window with the instance created, and start the main message loop. WndProc which handles every messages (events).** | **WndProc is essential in our project as we have to make the camera and the spaceship move depending on the user inputs. In WndProc we call the correct function depending on the event type. For instance, the one we call here is OnKeyDown when a key is pressed.** |
| Shaders - Pixel | **The pixel shader is a function, declared in a shader file using the HLSL language, which returns a color for a specific position. We can use it to apply a texture, a gradient of color, handle lighting…** | **I created several pixel shaders to apply a texture to the planets and the spaceship, apply a color gradient and a light source on the box… When I render an object in the Render function, I use the function PSSetShader on the immediate context to specify which pixel shader to use.** |
| Shaders - Vertex | **The vertex shader is similar to the pixel shader but it is used to apply transformation to every vertex. We use it to apply 3D transformations (translations, rotations, scaling) through matrices.** | **In the Render function I update the constant buffer with a matrix representing the transformations I want to apply to the current object.**  **This is how the planet orbiting animation, the movement of the spaceship… are made.** |
| Vertex Buffers | **Vertex buffers are composed of vertices representing a single 3D object.** | **The only two objects I declare manually are the billboards and the 2D shape which are two “planes” always rendered perpendicular to the camera direction to face it.**  **The spaceship, the planets are loaded through a function that fills a vertex buffer with an .obj file.**  **I rely on the function I made In a previous lab to generate a polygon for the 2D shape.** |
| Constant Buffer | **Constant buffers are used to pass data from the CPU to the GPU (our shaders).** | **Before rendering an object in the Render function, we update the variable of the constant buffer so that the pixel and vertex shaders is aware of which transformations to apply, what the light position is etc…**  **The constant buffer is then responsible of the transformations I apply on the orbiting planet, the moving spaceship, the orbiting box and the box receiving its light…** |
| 3D Transformations | **The three 3D transformations are: translation, rotation and scaling. We use matrices to represent these transformations by applying functions like XMMatrixRotationY, XMMatrixScaling or XMMatrixTranslation.** | **The code below is making the moon orbiting around the planet.**  **It also make sure that it is scale correctly depending on whether j or k has been pressed.** |
| Textures | **Textures are images applied to a 3D object.**  **We use the function D3DX11CreateShaderResourceViewFromFile to create Shader Resource View from an image.**  **The input layout is composed also of the texture coordinates the current vertex has. Therefore, the pixel shader knows which part of the texture image to draw on this position thanks to the sampler.** | **I created a pixel shader to handle texture which is used by the two planets and the spaceship.** |
| Lighting | **To handle lighting, I update the constant buffer with a list of lights positions and colors.**  **In the pixel shader which handles light, I apply every light in the list to the current drawn pixel.** | **I use only this pixel shader for the box with a gradient color which is lit by a box orbiting around it.** |
| Loading Objects | **As said previously, I rely on the the LoadOBJ and Load3DS provided in a previous lab to load 3D models found online.**  **It allows us to add good looking 3D objects to our scene without having to fill the vertex buffer manually.** | **For instance, the spaceship is a 3D model I found online with its texture.** |
| Skybox | **Skyboxes are images applied on a surface (a plane or a sphere) close to the camera to represent a sky or an environment background.** | **For this project I apply a space image to a sphere around the camera.**  **The main difference with others objects is that we have to update the depth stencil state and make the texture visible from the inside, as shown below.**    **We have to make sure that the sphere is big enough or follows the camera so that the user always see the texture.** |