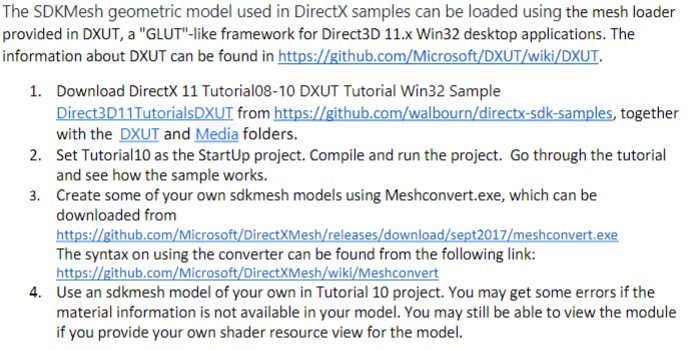
700106 / 700120 Lab Book

Week 7 – Lab G

**Exercise 1. Load SDK Mesh Model using DXUT**

**Question:**



**Solution:**

1. As mentioned in the question follow the steps. Firstly, configure the project in the local and run it.

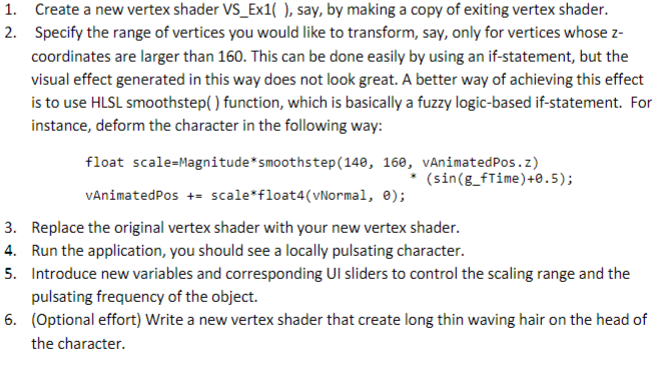
**Sample output:**

**Reflection:** In this assignment, I got to know about the DXTU and loading the mesh model and adding the controls to it, like rotation, full screen, increase/decreasing the puffiness of the model which is the second screen shot shown above.

**Exercise 2: Locally controlled pulsating character.**

**Question:**



**Solution:**

1. Firstly, will be using the same code from the previous assignment and will be adding the pulsating control to the screen, which is given below.

Add the time and frequency in the constant buffer,

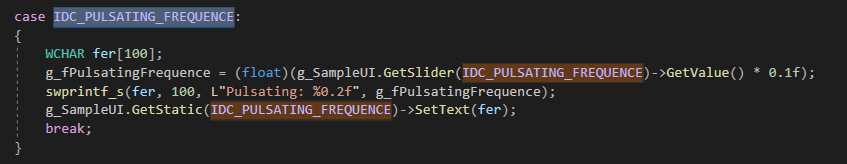


Then, introduce one variable to take the frequency input and pass to the shader.

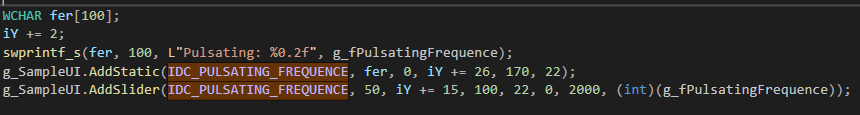


And add the UI control Id  


Add one more case to handle the GUI events, basically will take the frequency input.

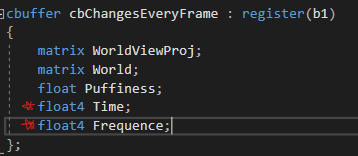


To appear on this pulsating on the screen add this piece of code in the InitApp() method.

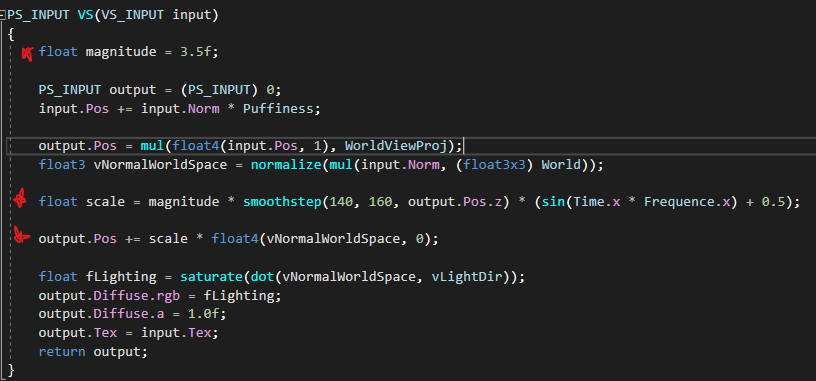


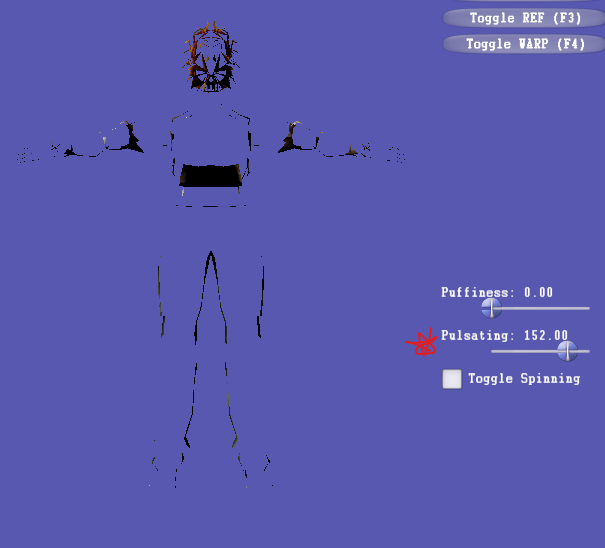
Now the corresponding changes will be made in the shaders.

Firstly, add the frequency and time variable in the constant buffer, as given below.



Modify the vertex shader as marked lines below.

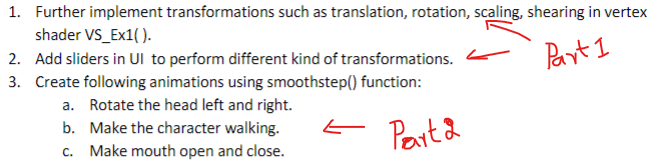


**Sample output:** it will pulsate along the z-axis as shown below****

**Reflection**: As in the previous exercise, here we are doing the modification in the GUI and added the input pulsating scale in the screen and it is varying from 0-200, as we increase the scale it will move very faster. Additionally, used the smoothstep() which Returns a smooth Hermite interpolation between 0 and 1 which is between the ranges that we provide in the smoothstep() method.

**Exercise 3. Per-vertex local skinning transformations**

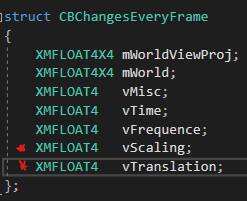
**Question:**



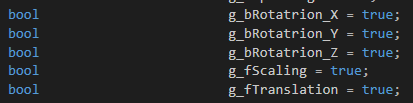
**Part - 1**

**Solution for the transformation:**

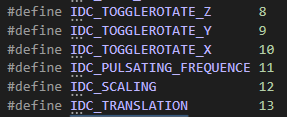
**1**. Firstly, will be adding the two variables in the constant buffer to pass the value.



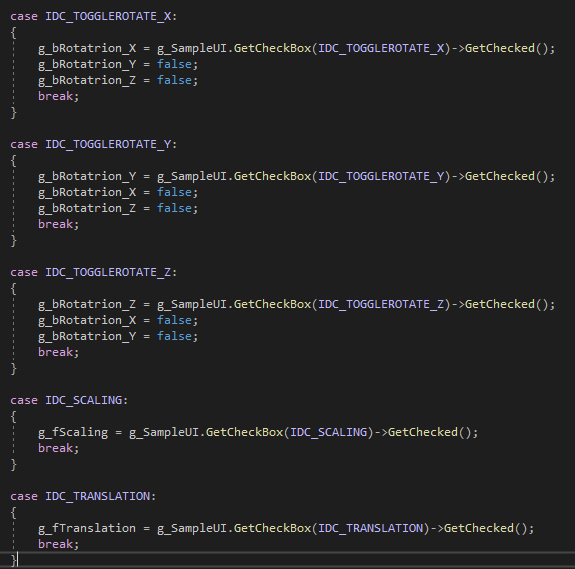
Will be adding global variables to pass the corresponding value.



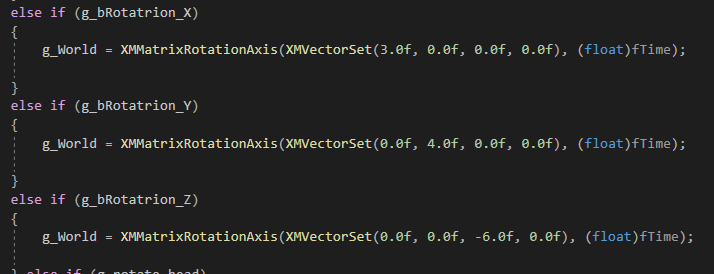
Then, add the UI control Ids.



After that add the cases for each id.



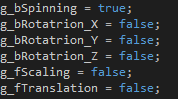
Then, for rotation add the below code for handling the GUI events.



And pass the value to the shader.



In the InitApp() method, add the below code to show the transformation control to appear on the screen.

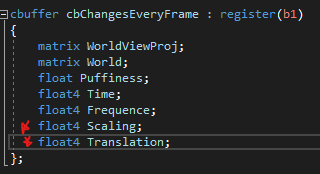
First make all the variables as false. 

Then, add the below code.

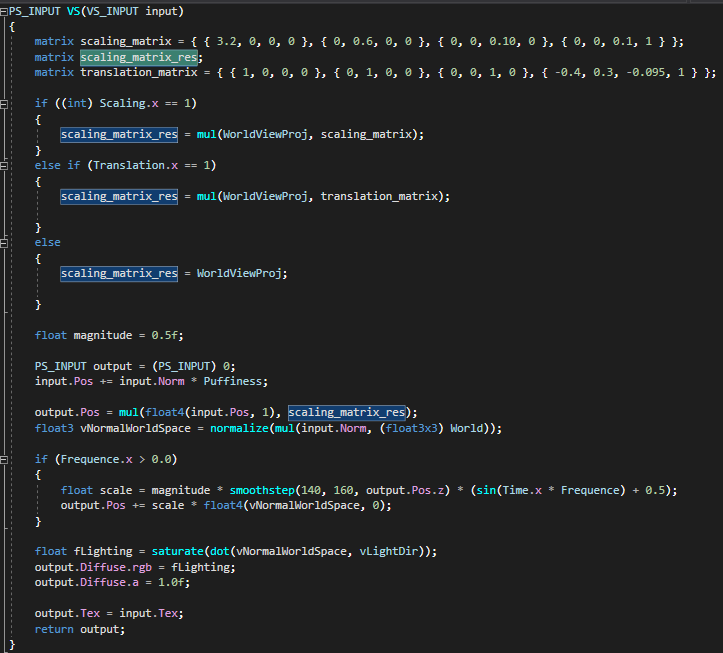


Now, will be making the corresponding changes in the shaders.

Firstly, declare all the variables in the constant buffer, as given below.



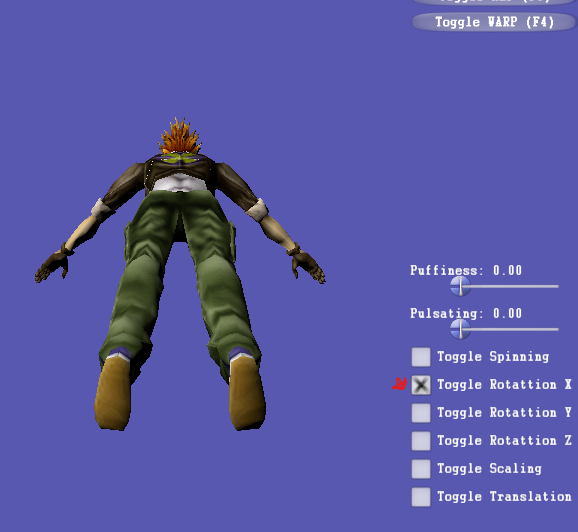
Then, modify the vertex shader.



Here, In above code I have declared the scaling and translation matrix inside the vertex shader and it can be passed from the CPU side as well.

**Sample output for transformation:**

Controls:  
 

Rotation along the X-axis:   
 

Rotation along the Y-axis:  
 

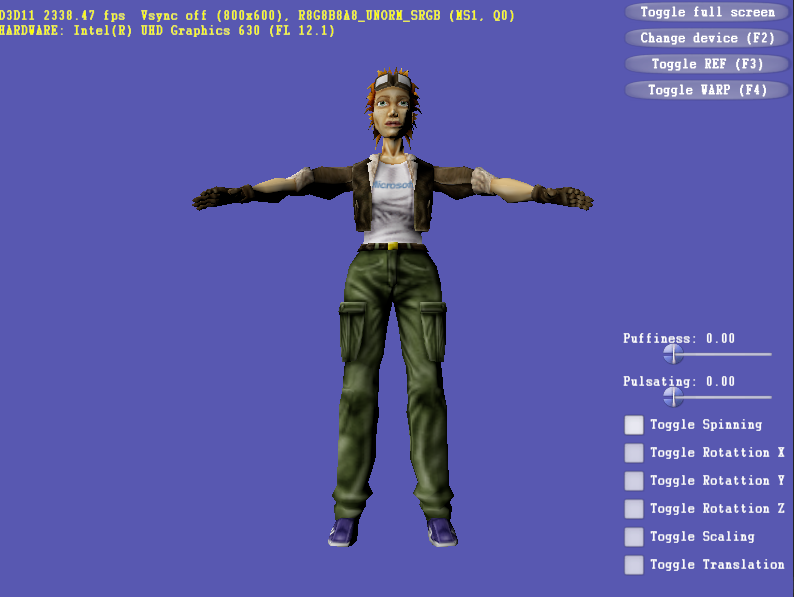
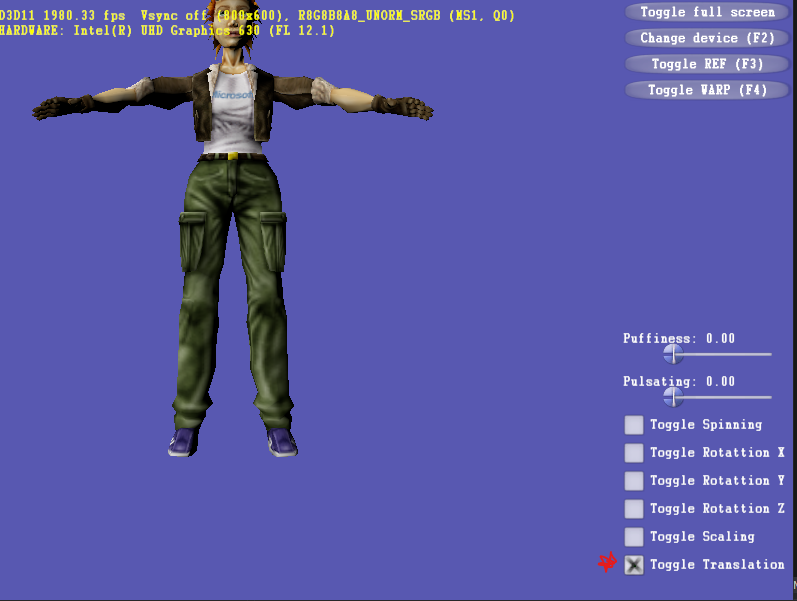
Rotation along the Z-axis:

Scaling along X-axis:

****

With and without Translation:

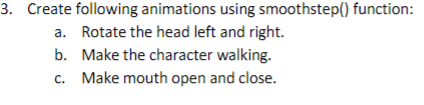
**** ****

**Reflection for transformation:**

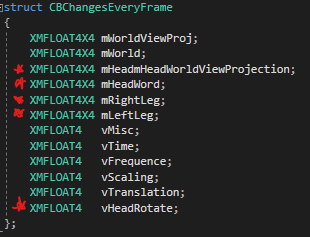
In this assignment got to know about applying the transformation to the model using the UI inputs and passing the values to the shader.

**Part - 2**

**Solution for the:**

****

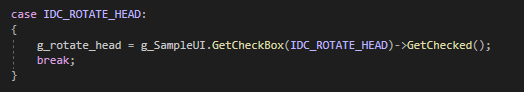
Firstly, modify the constant buffer, by adding the below marked variables.

****

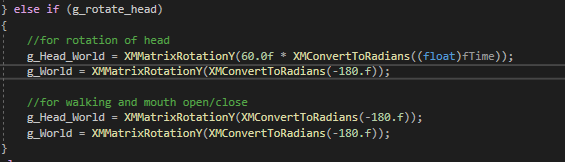
Add the UI control id, will be using the same id for head rotation, walking and open and close mouth.

****

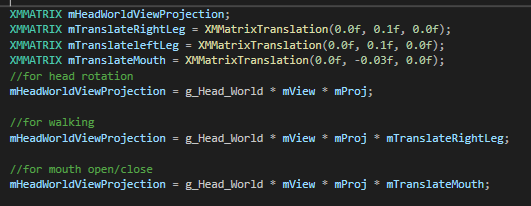
To handle the scene add the below case.



And to handle the events use the below code and comment it according to the uses and comment.

****

Will be using different world, in order to achieve the task and use the below mentioned code according to the comments.

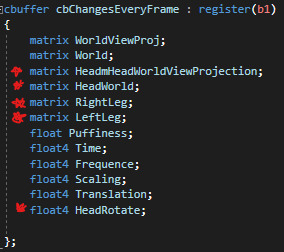
****

****

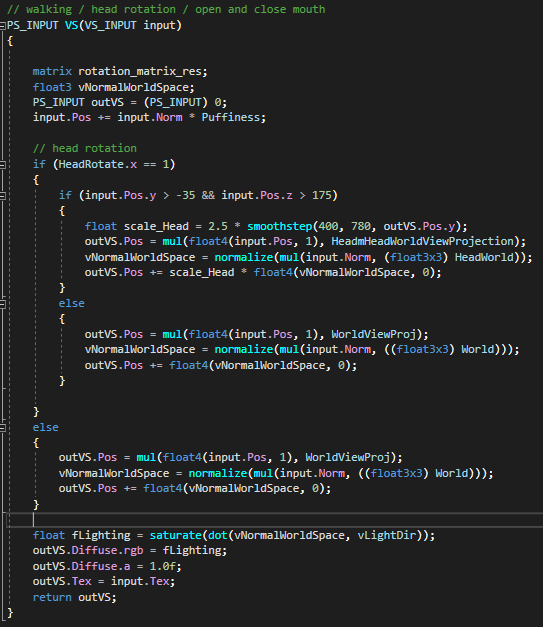
Now add the head rotate toggle to the screen.

****

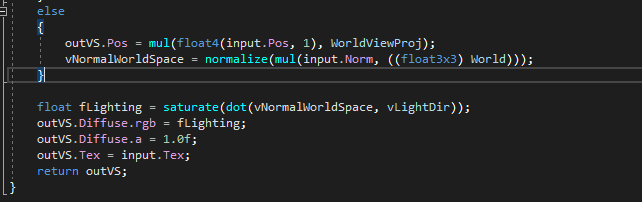
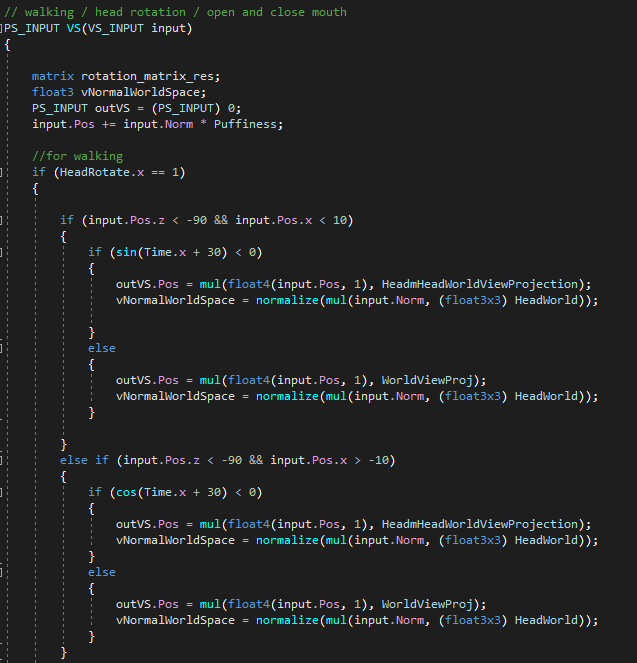
In the shader file modify accordingly, add the below parameters for head rotation, walking, and mouth open/close effects.

****

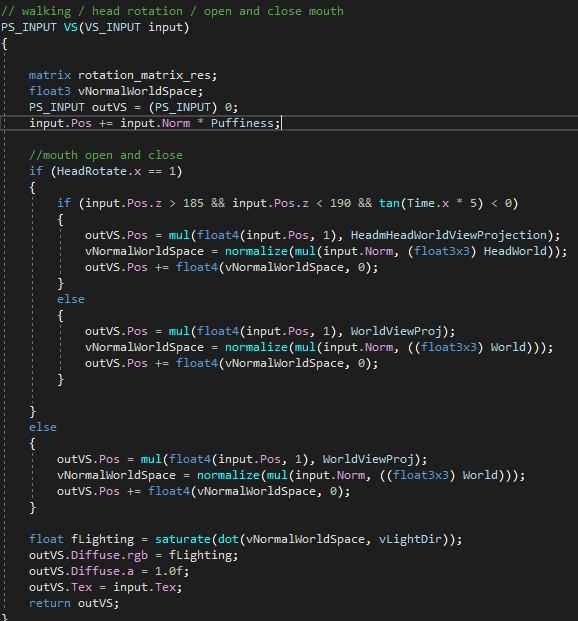
Vertex shader for head rotation.

****

Vertex shader for model walking.

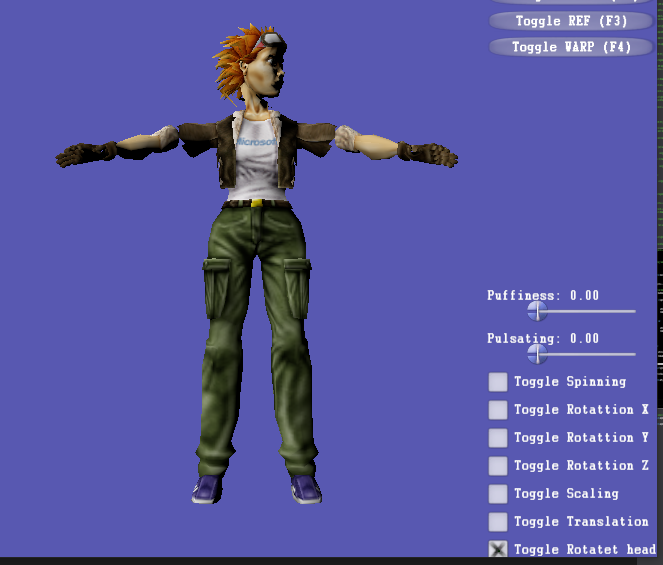


Vertex shader for Mouth Open/Close.

****

**Sample output for Head rotation:**

****

****

**Sample output for walking:**

****

****

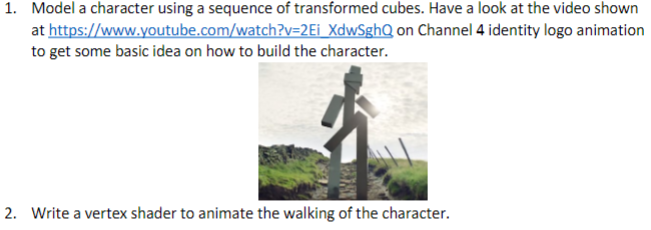
**Sample output for mouth open and close:**

****

****

**Exercise 4. Character animation using a cube model**

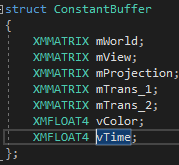
**Question:**



**Solution:**

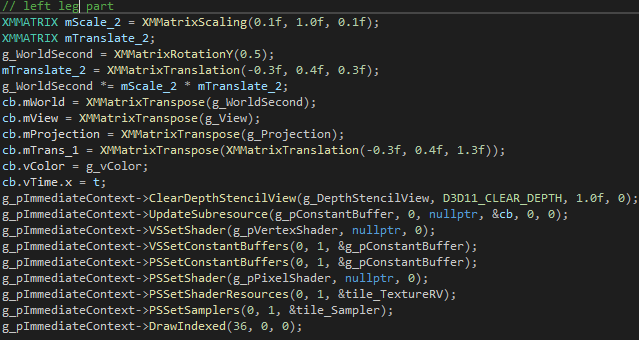
**1**. In this will be using tutorial 4, for the cube character and will be drawing multiple cubes. Additionally, will be using the stone texture for the cube.

Firstly, I have altered the constant buffer for transformation.

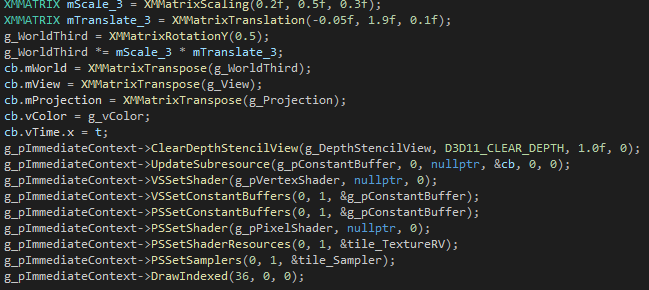


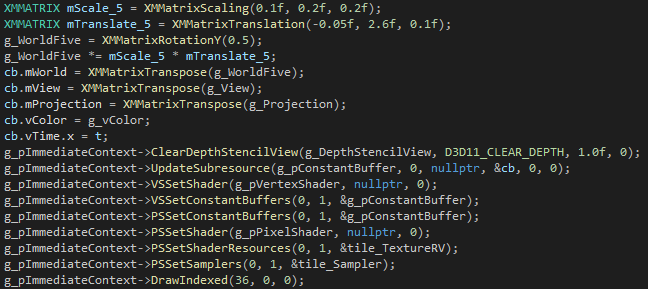
In the Render() method will be drawing multiple cubes using DrawIndexed() function.

Right Leg:  
****

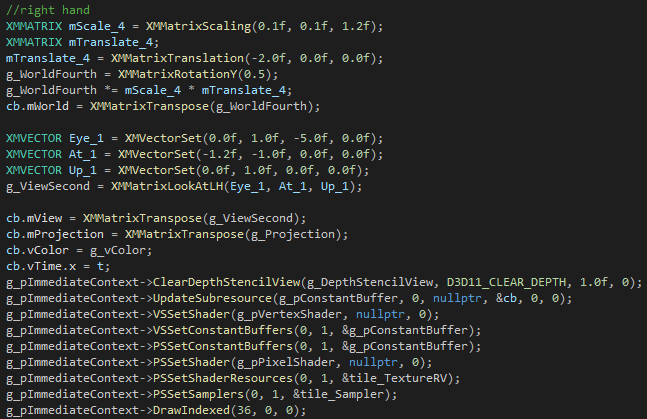
Left Leg:  


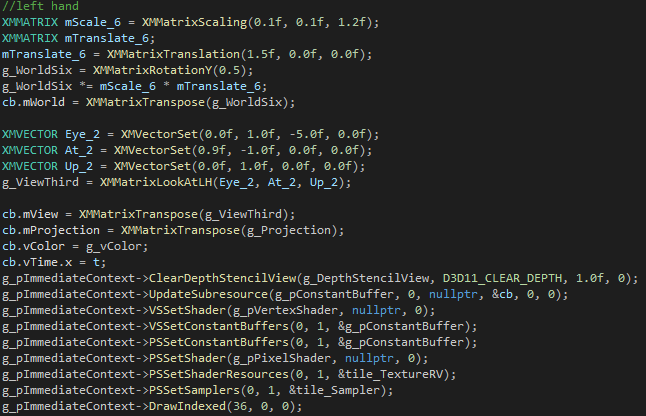
Body:



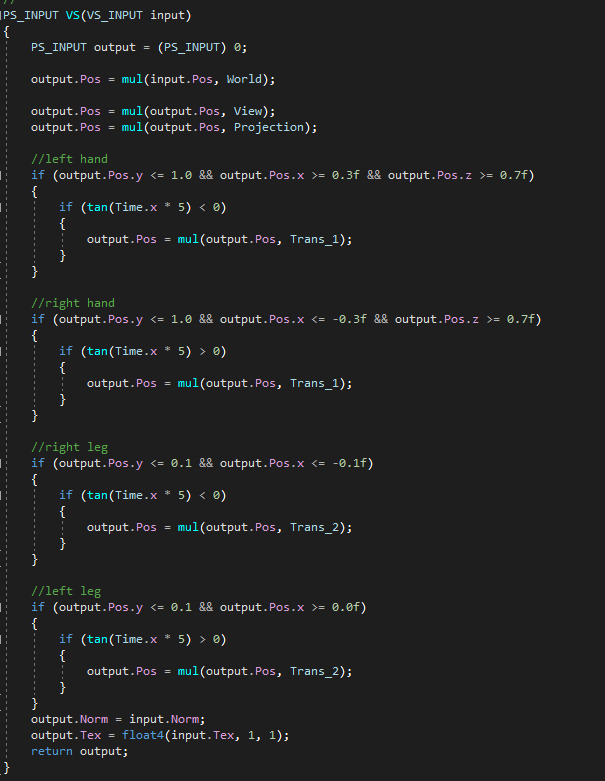
Head:  
 

Right Arm:

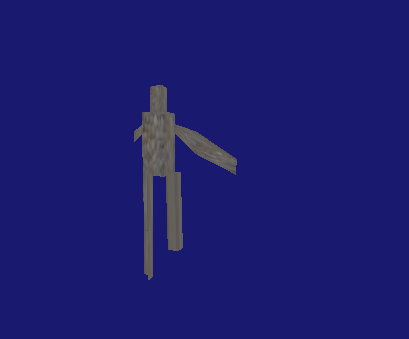
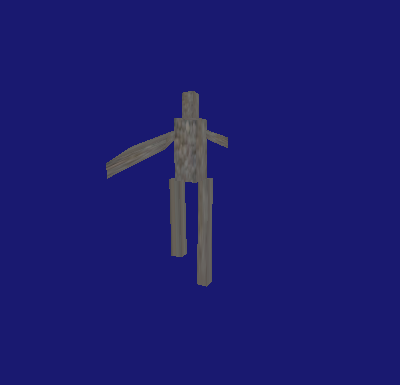


Left Arm:  
****

Similarly, we need to modify the vertex shader, as given below.



**Sample output static:  
**

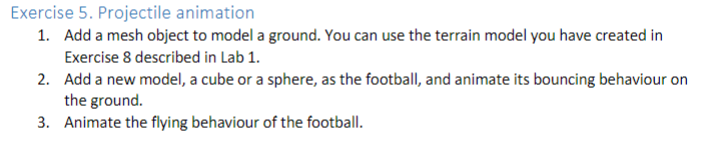
**Sample output walking:**  


**Reflection:**

Here, in this assignment got to know about applying the transformation to the cubes and make it like walking. The same effect can be done in the CPU side using the same conditions which have written in the vertex shader, but it is better to write it in the shader side(GPU side).

**Exercise 5. Projectile animation**

**Question:**

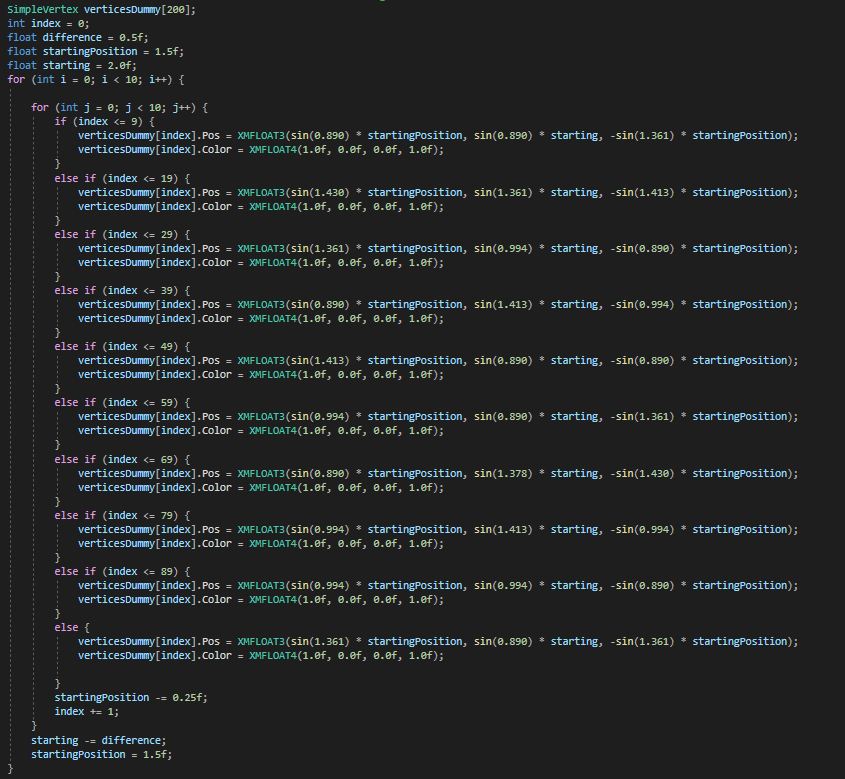
****

**Solution:**

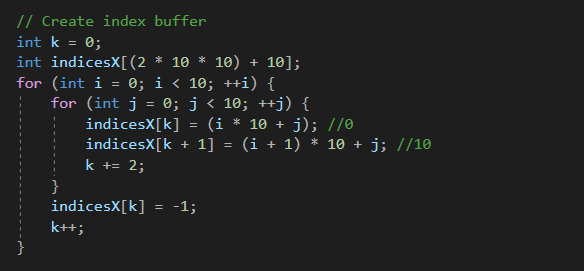
**Drawing the terrain mesh:**

**1**. In this will be using tutorial 4 lab 01 exercise 08 in order to make the ground.

Below is the vertices and indices for the terrain model.

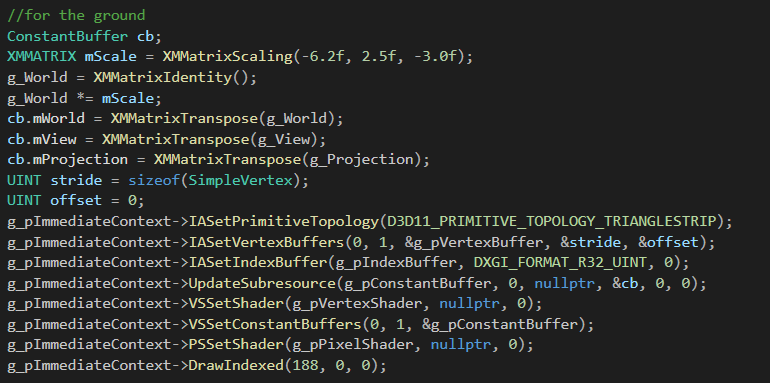


And the Indices.



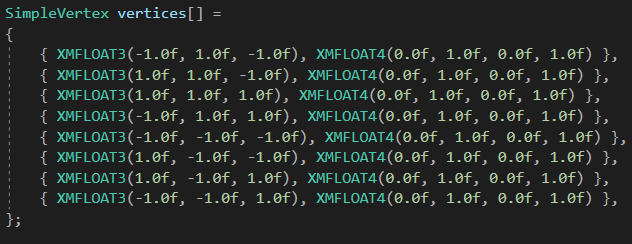
Will be binding both indices and vertices to the D3D11\_BUFFER\_DESC.

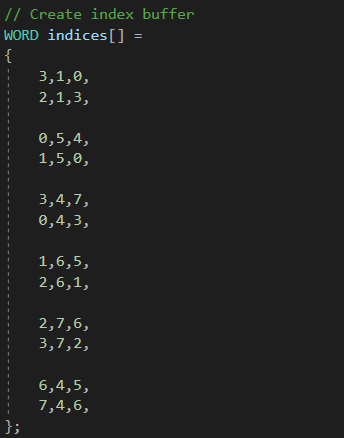
Then, will be drawing the ground in the render() method and I have applied the scaling for the terrain mesh, as given below.

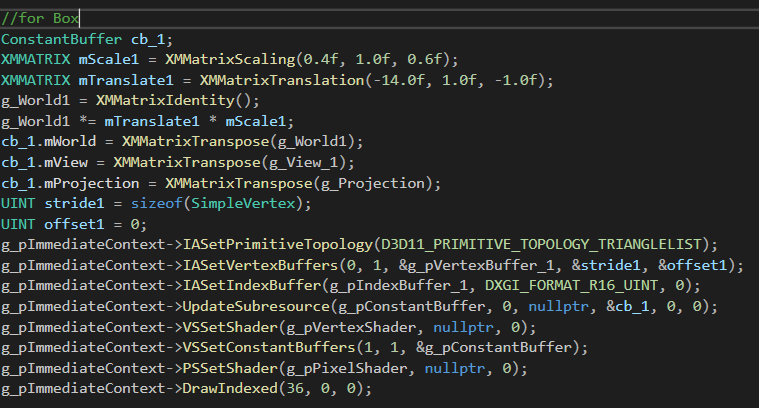


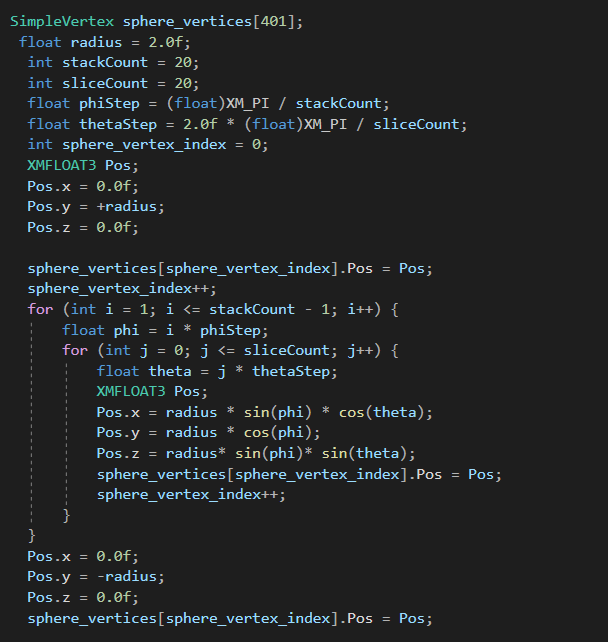
**Drawing the Box:**

Will be using normal cube. Below is the vertices and indices for cube.



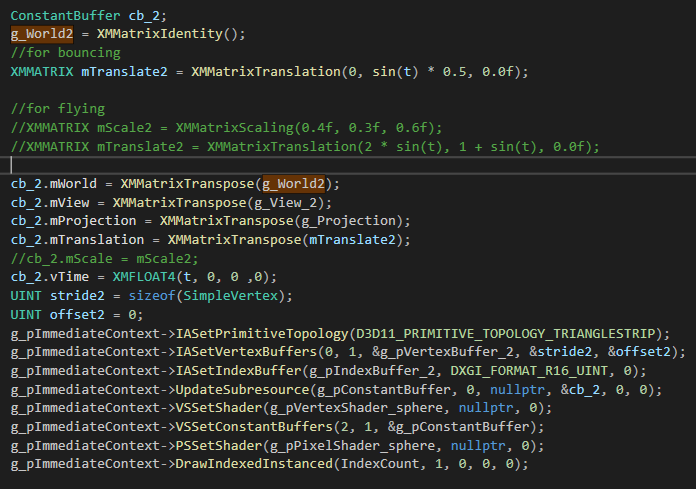


In order to draw the cube I have applied translation and scaling.  


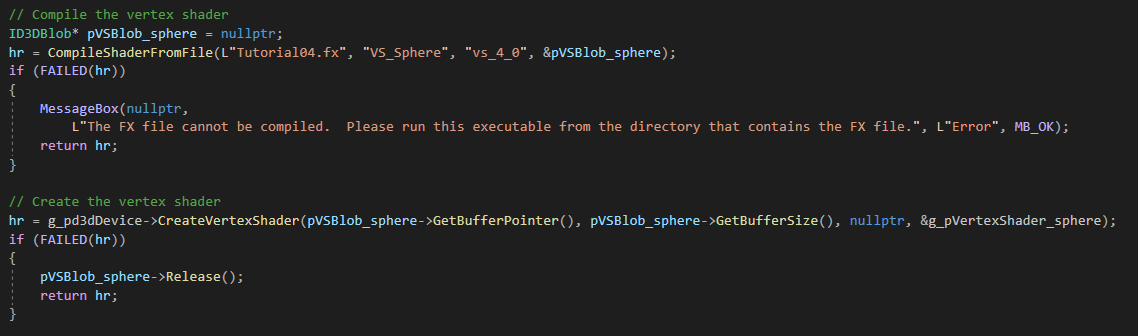
**Drawing the Sphere:** Vertices for the sphere.  


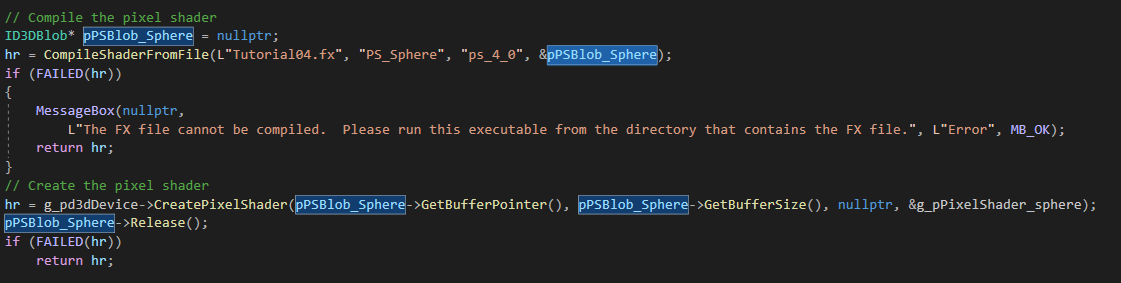
Indices for the sphere.



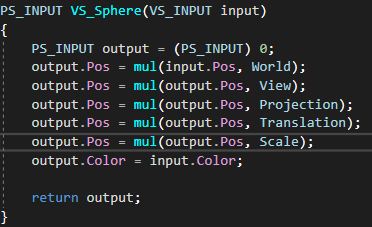
For drawing the sphere, and will be applying the translation and scaling to achieve the flying and bouncing effect.(will be commenting the translation and scaling in order to do corresponding effects and given in the below scree shot)  
   


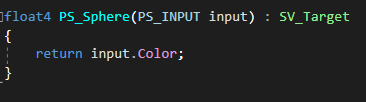
I have created the new vertex shader and pixel shader for applying the translation and scaling, which are given below.

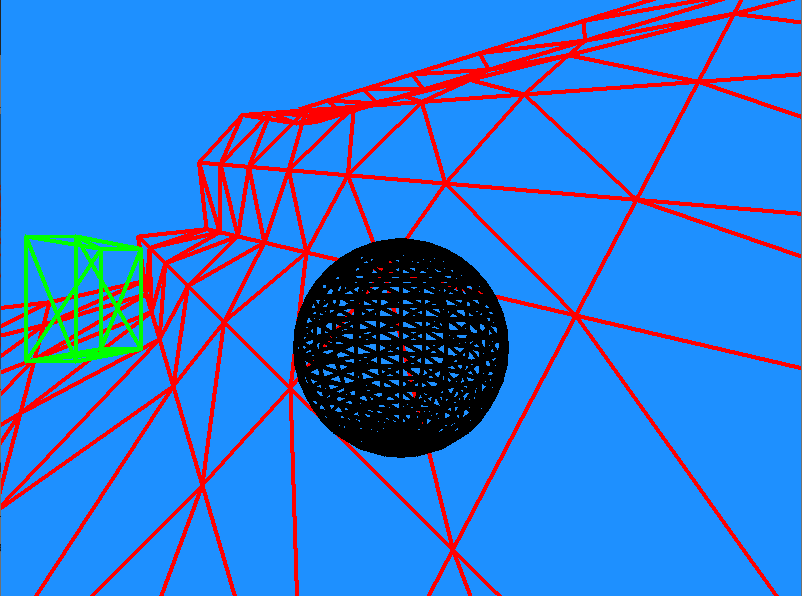
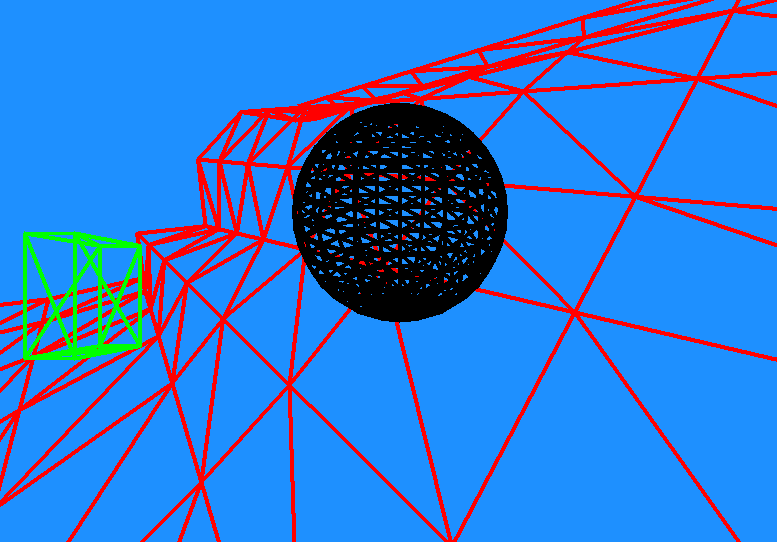


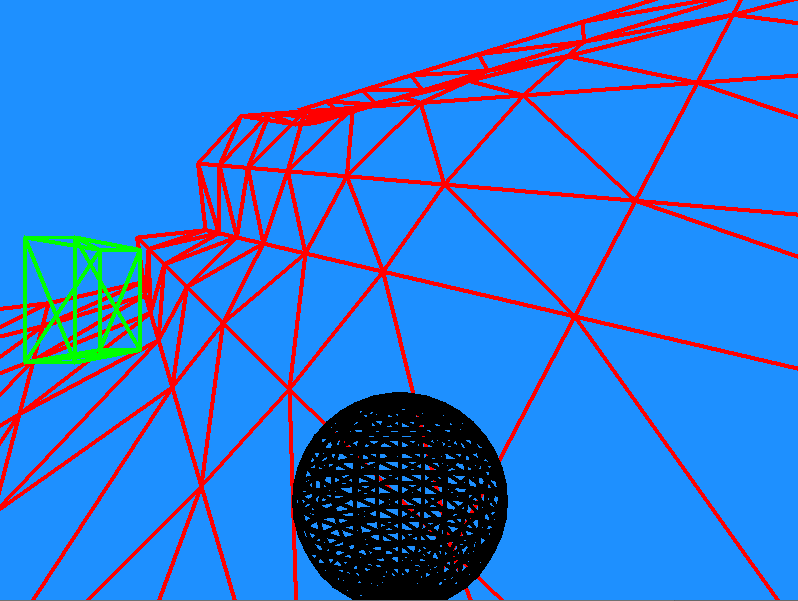


In the shader file I have added the new vertex and pixel shader methods,

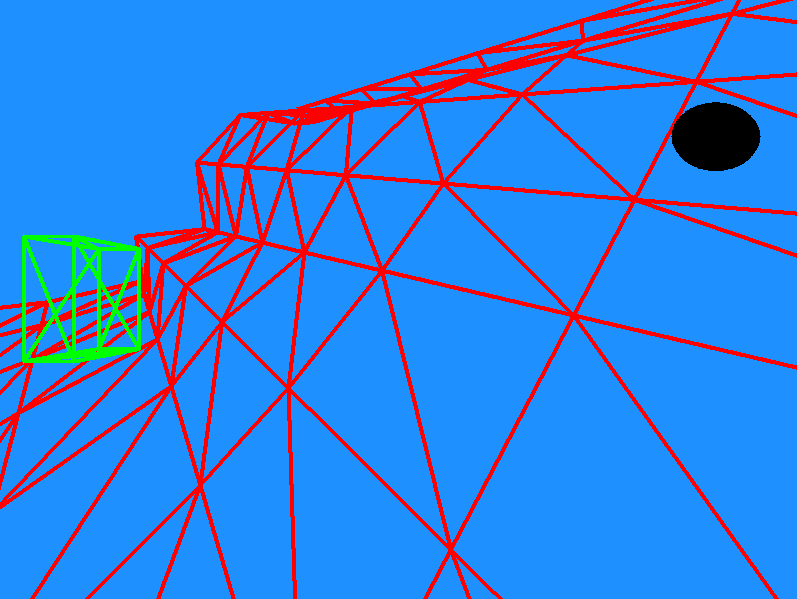
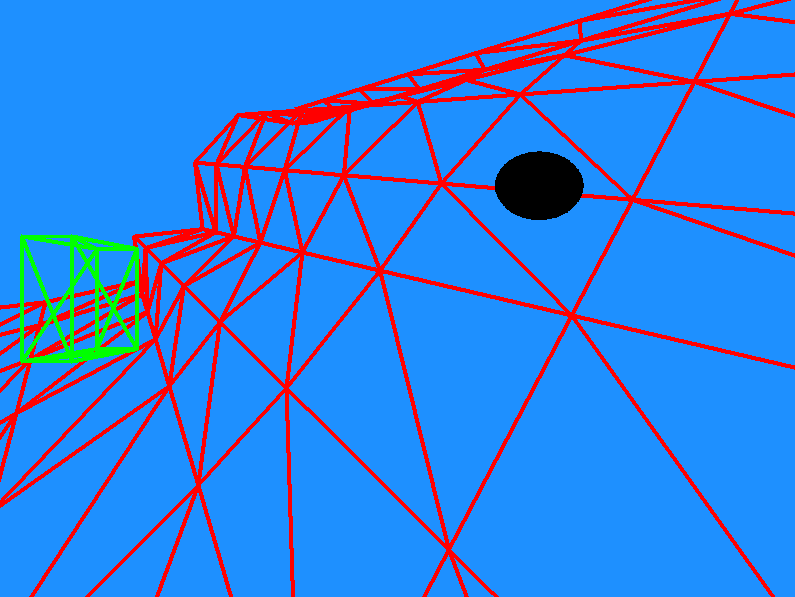


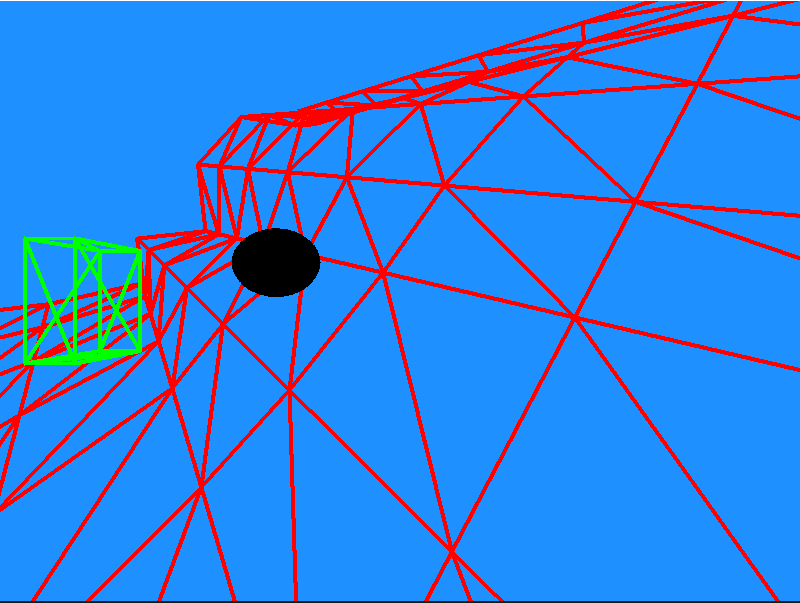
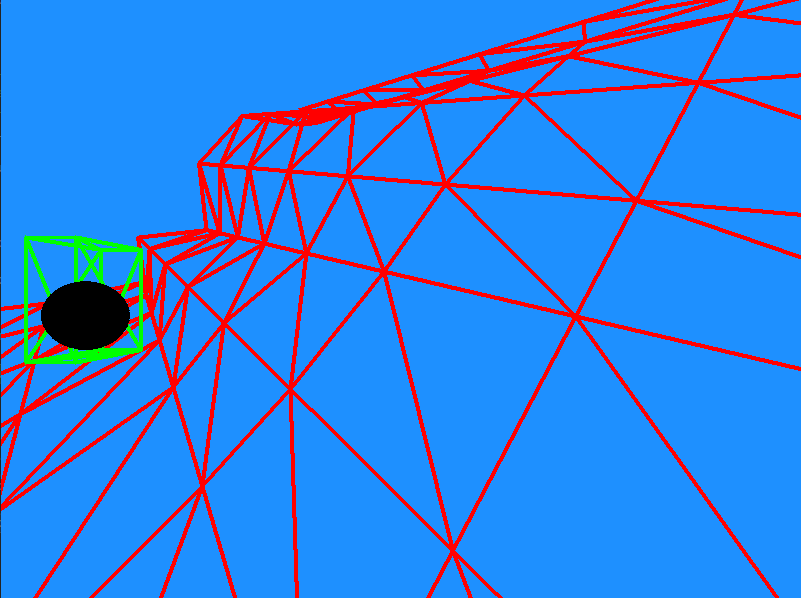


**Sample output Bouncing:  
**

****

**Sample output for flying:**

****

****

**Reflection:**

Here, in this assignment loaded the multiple items and applied translation for an object using the new vertex shader. Most challenging part was to draw a sphere.