Sessió 6
Shaders

Durant la sessió d'avui continuarem treballant amb els shaders.

EffectEdit

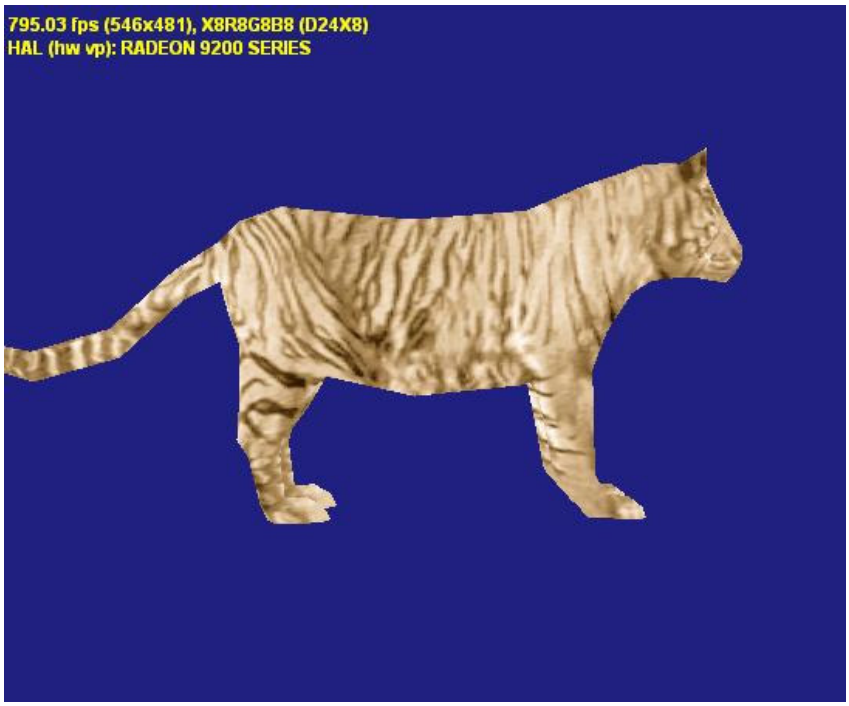
Tornarem a utilitzar l'eina EffectEdit per implementar el codi del shaders d'avui. Els shaders que implementarem:

- *texturedBW.fx*

Amb aquest shader podrem visualitzar les malles en blanc i negre.

- *texturedSepia.fx*

Amb aquest shader podrem visualitzar les malles amb color sepia.



- *TintedTextured.fx*

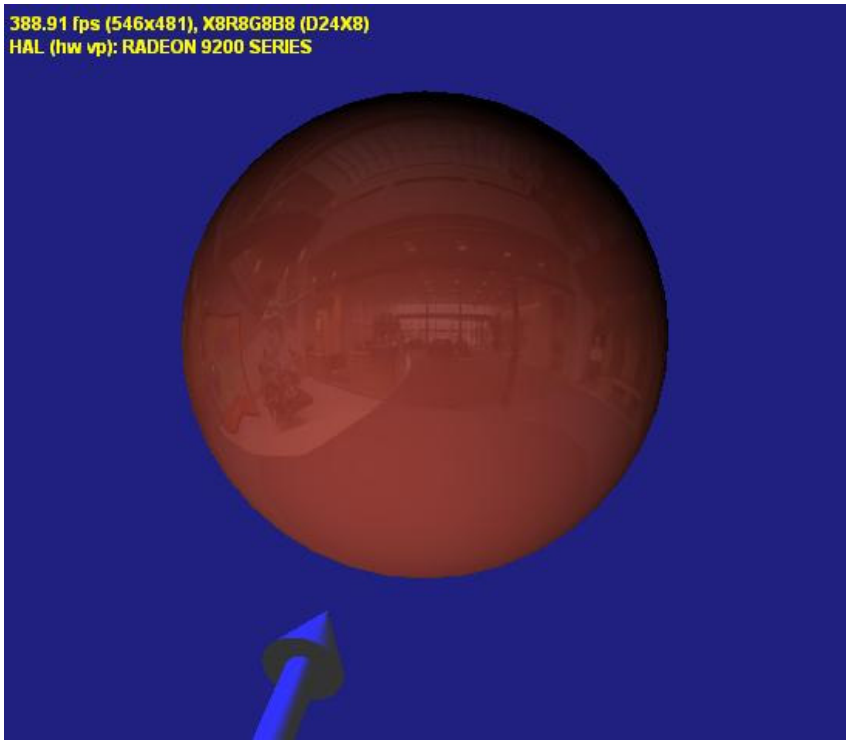
Amb aquest shader podrem tintar la malla dels nostres models amb un color per defecte.

- *RenderPlane.fx*

Amb aquest shader podrem visualitzar en un pla XZ, XY o YZ les malles.

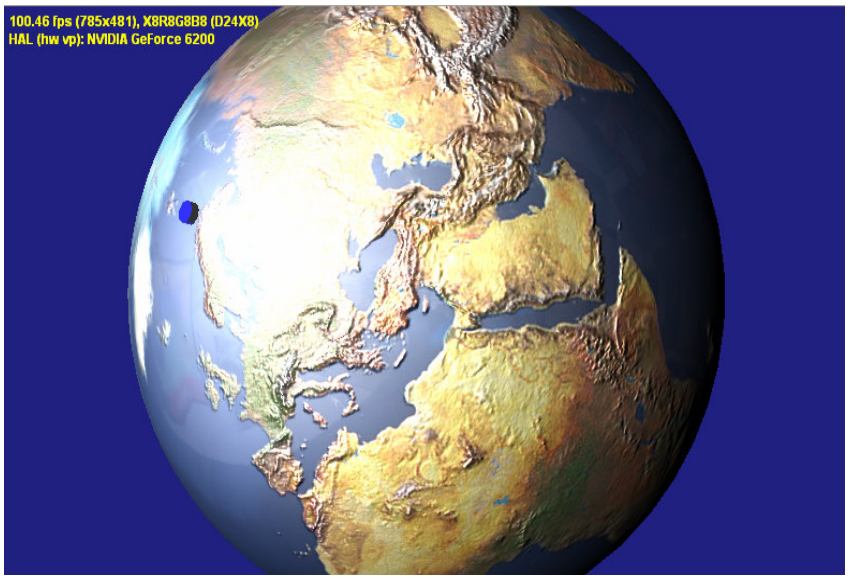
- *environmentCubeMap.fx*

Amb aquest shader podrem visualitzar una malla reflexant una textura de tipus cubemap.



- *normalMap.fx*

Amb aquest shader podrem visualitzar una malla amb efecte bump (de relleu) mitjançant una textura de normals (normalmap).



Render Cal3D per Vèrtex Shader

Un dels problemes que ens vem trobar durant l'assignatura d'Introducció als Videojocs és que quan executàvem l'aplicació en mode debug el frame rate baixava fins a menys de 30fps.

Una forma de millorar el rendiment del nostre videojoc mitjançant shader és realitzar els càlculs del model animat. Per realitzar això, hem de crear un objecte de la classe `CalHardwareModel` que ens permetrà utiltizar-ho per poder passar-li a la technique un model de dades amb la informació dels vèrtexs.

Aquest vèrtex estarà format per una estructura com la següent:

```
struct VERTEX_HW_CAL3D
{
      D3DXVECTOR3    pos;
      float  Weight[4];
      float Indices[4];
      D3DXVECTOR3 normal;
      FLOAT tu,tv;
};
```

Ens trobem amb un vèrtex que pot tenir aplicats quatre ossos (els Indices són els índexs dels ossos que afecten el vèrtex) amb diferents pessos (el Weight són els pessos aplicats en cadascun dels quatre ossos) format per la posició (pos), normal i coordenades de textura (tu,tv).

En la inicialització del vertex buffer omplirem l'objecte de tipus `CalHardwareModel` amb la informació necessària, cridant als mètodes:

- `setVertexBuffer`, amb aquest mètode establirem on és troben les coordenades x, y, z del nostre vèrtex buffer. Li passarem l'stride del vèrtex (el desplaçament per passar d'unes coordenades xyz fins a la següent).
- `setWeightBuffer`, establim els pessos del vèrtex buffer passant-li el primer vèrtex on es troben.
- `setMatrixIndexBuffer`, estableix els índexos dels ossos que s'apliquen sobre un vèrtex.
- `setNormalBuffer`, ara establirem les normals passant-li on comença les normals del vèrtex.
- `setTextureCoordNum`, li diem quantes coordenades de textura tenim en el nostre vèrtex.
- `setTextureCoordBuffer`, li passem on es troba la coordenada de textura del vèrtex dient-li el numero de textura.
- `setIndexBuffer`, establim els índexs de la malla.
- `load`, és necessari pel cal3d, li hem de passar el número d'ossos que té la nostra malla.

Per fer el render de la malla per shader hem creat un nou mètode dins de la classe CCal3DInstanceActor RenderByHardware on com sempre establirem les variables que necessita la nostra technique, establint la matriu de WorldViewProjection (`g_WorldViewProjMatrix`), els ossos del model (`g_Bones`).

Renderitzar els Items amb EnvironmentMap

Per donar un imatge més agradable en el nostre videojoc li aplicarem un shader de EnvironmentMap als items aplicant-li reflexes utilitzant una textura de tipus *cubemap*.

Per fer això utilitzarem un shader inspirat en el shader *environmentCubeMap.fx*.