Projet WebGL DAWIN 2014

Le but du projet est de réaliser une application affichant un système solaire à l'aide de WebGL. Le projet est à réaliser par groupe de 2 ou 3. Le projet est à rendre par mail à alexandre.perrot@labri.fr sous la forme d'une archive <noms du groupe>.tar.gz contenant le ficher html du projet et un dossier avec les ressources necessaires (not. les textures). La date limite de rendu est le dimanche 21 décembre 23h59 GMT+1.

Consignes

Votre système solaire sera composé au minimum, d'un **soleil**, de la **Terre** et de la **Lune**, mais il est fortement recommandé d'ajouter d'autres planètes. Vos planètes seront de préférence sphériques (voir ci-après pour la technique), mais au minimum cubiques. Vous implémenterez les mouvements de **rotation** et de **révolution**. Les orbites seront **circulaires** et matérialisées par un **cercle de couleur**. Vos planètes seront texturées en utilisant les textures disponibles ici : `http://planetpixelemporium.com/planets.html`.

Tout le système sera animé et interactif. Il doit être possible de zoomer et de tourner autour du système.

Dessiner une sphère

La technique proposée ici met en pratique plusieurs éléments vus durant l'UE. Il s'agit de dessiner une sphère à partir d'un cube. Chaque face du cube doit être constituée non pas d'un seul carré, mais d'une grille. Cette grille est ensuite déformée grâce au vertex shader pour donner une sphère. Plus de détails ici : `http://stackoverflow.com/questions/7687148/drawing-sphere-in-opengl-without-using-glusphere`.

La première étape est donc de dessiner un cube dont chaque face est une grille, centré sur l'origine. Les coordonnées de la sphère sont calculées dans le vertex shader avec cette ligne :

```
vec3 position = normalize(aVertexPosition);
```

Le résultat peut être multiplié par le rayon de la sphère pour obtenir des tailles différentes.

Texture coordinates Avec cette technique, les coordonnées de textures n'ont pas besoin d'être passées en tant qu'attribut, elles peuvent être calculées dans le fragment shader à partir de la position. Pour cela, on utilise un varying initialisé dans le vertex shader avec la position normalisée, appelé ici vTex. Les coordonnées de texture sont ensuite obtenues avec :

```
vec2 texCoords = vec2(-(atan(vTex.z, vTex.x) / PI + 1.0) * 0.5,
                                  -(asin(vTex.y) / PI + 0.5));
```