UNIVERSITE DU QUEBEC A CHICOUTIMI
6GEN715 – INFOGRAPHIE

Travail 2 – partie 1
Programmation GLSL et interactions avec l'usager

Dates de remise de l'anneau de l'étape f:

Pour le groupe du lundi : 1 octobre 2017
Pour le groupe du jeudi : 4 octobre 2017

1- OBJECTIF

- Familiariser l'étudiant au développement de « shaders » dans le langage GLSL.
- Familiariser l'étudiant aux mécanismes permettant des interactions avec l'usager dans le langage Javascript.

2- MODALITÉ PARTICULIÈRE

Ce travail doit être réalisé individuellement.

3- TRAVAIL À RÉALISER

a) Téléchargez les fichiers suivants :

- cube.html
- cube.js

b) Étudiez le code des deux « shaders » ainsi que le fichier *cube.js*. Notamment :
   - Observez la présence de la variable « uniform » **theta** dans le « vertex shader ».
   - Observez les lignes suivantes dans le fichier *cube.js*.
     i. thetaLoc = gl.getUniformLocation(program, "**theta**");
     ii. gl.uniform3fv(thetaLoc, theta);

   Pourriez-vous expliquer ce que réalisent les deux lignes précédentes?

c) Renommez les fichiers téléchargés en remplaçant « cube » par « anneau ». Vous devrez évidemment apporter une modification au contenu du fichier *anneau.html* de telle sorte qu'il utilise le fichier *anneau.js* plutôt que *cube.js*.

d) Modifiez le fichier *anneau.js* de telle manière qu'il soit possible de modifier la taille (selon les trois axes) et la position du cube (également selon les trois axes). Vous

aurez besoin d'un « vertex shader » qui permet de modifier les sommets avant qu'ils subissent une rotation. Pour simplifier votre tâche, je vous propose d'utiliser le code suivant :

```
<script id="vertex-shader" type="x-shader/x-vertex">

attribute  vec4 vPosition;
attribute  vec4 vColor;
uniform vec3 theta;
uniform  vec4 vScale;
uniform  vec4 vDisplacement;
varying vec4 fColor;

void main()
{
   // Compute the sines and cosines of theta for each of
   //   the three axes in one computation.
   vec3 angles = radians( theta );
   vec3 c = cos( angles );
   vec3 s = sin( angles );
   vec4 newPosition, rotated_position;

  // Appliquer vos modifications sur la variable vPosition
   newPosition.x = vScale.x * vPosition.x + vDisplacement.x;
   newPosition.y= vScale.y * vPosition.y + vDisplacement.y;
   newPosition.z= vScale.z * vPosition.z + vDisplacement.z;
   newPosition.w= vPosition.w;

   // Remember: these matrices are column-major
   //           (columns are stored one after another)

   mat4 rx = mat4( 1.0,  0.0,  0.0, 0.0,    //column 1
                   0.0,  c.x,  s.x, 0.0,    //column 2
                   0.0, -s.x,  c.x, 0.0,    //column 3
                   0.0,  0.0,  0.0, 1.0 );  //column 4

   mat4 ry = mat4( c.y, 0.0, -s.y, 0.0,    //column 1
                   0.0, 1.0,  0.0, 0.0,    //column 2
                   s.y, 0.0,  c.y, 0.0,    //column 3
                   0.0, 0.0,  0.0, 1.0 );  //column 4

   mat4 rz = mat4( c.z,  s.z, 0.0, 0.0,   //column 1
                  -s.z,  c.z, 0.0, 0.0,   //column 2
                   0.0,  0.0, 1.0, 0.0,   //column 3
                   0.0,  0.0, 0.0, 1.0 ); //column 4

   rotated_position =  rz * ry * rx * newPosition;

   // Now we need to invert the Z coordinates since Web
   // browsers map the clip coordinates to the NDC (normalized
   // device coordinates) expecting the Z axis to be inverted
   // (as do perspective projection matrices).

   rotated_position.z = -rotated_position.z;

   fColor = vColor;

   gl_Position = rotated_position;
}
</script>
```

Étudiez le code du « shader » de la page précédente (regardez, en particulier, les quatre lignes sous le commentaire « *// Appliquer vos modifications sur la variable vPosition* »).

Modifiez le fichier *anneau.js* afin de pouvoir contrôler la taille et la position du cube (vous devrez passer des valeurs aux variables *vScale* et *vDisplacement* du nouveau « vertex shader »).

e) Modifiez ensuite le fichier *anneau.js* de manière à afficher deux cubes **en n'utilisant que les coordonnées d'un seul cube** comme le montre l'animation suivante : animation 1 (youtube)

Vous devrez créer ces deux cubes (l'un est déformé, l'autre non) en n'utilisant **que les 8 sommets du programme Javascript original**. Chaque cube (déformé ou non) doit être créé en exécutant la fonction « drawArrays» avec ces mêmes sommets. Vous devrez modifier les variables *vScale* et *vDisplacement* du « vertex shader » avant chaque appel à la fonction « drawArrays».

f) Modifiez finalement le fichier *anneau.js* de manière à afficher un anneau comme celui montré dans l'animation suivante : animation 2 (youtube)

Tout comme au point (e), vous devrez créer cet anneau en n'utilisant que les 8 sommets du programme Javascript original.

4- RAPPORT

S.V.P. Nommez votre fichier ZIP en indiquant qu'il s'agit de la partie 1
(Exemple : travail2partie1-Audet-Daniel.zip.pgp)

Le fichier d'archive (ZIP) doit contenir tous les fichiers requis pour visualiser **l'anneau demandé à l'étape f**.

Veuillez inclure le dossier « Common » dans votre fichier d'archive de telle sorte que l'extraction du contenu de votre fichier ZIP permettent de visualiser tous les fichiers HTML en double-cliquant sur ceux-ci (vos fichiers HTML doivent donc contenir un chemin correct pour qu'un navigateur puisse retrouver les fichiers Javascript présents dans le dossier « Common »).

Cryptez le fichier d'archive ZIP et transmettez le fichier résultant (PGP) via l'interface prévue à cette fin.

Procédure de cryptage à utiliser pour remettre les travaux

REMISE DES TRAVAUX ("upload")