UNIVERSITE DU QUEBEC A CHICOUTIMI
6GEN715 – INFOGRAPHIE

Travail 1 – Introduction à WebGL

Dates de remise des programmes de la section 5:

Pour le groupe du lundi : 24 septembre 2017
Pour le groupe du jeudi : 27 septembre 2017

1- OBJECTIF

- Familiariser l'étudiant aux fonctions graphiques de librairie WebGL.

2- MODALITÉ PARTICULIÈRE

Ce travail doit être réalisé individuellement.

3- INFORMATIONS UTILES

Vous pouvez obtenir la description de chaque méthode de WebGL ainsi que des paramètres requis par chacune d'elles en consultant les liens suivants:

- http://msdn.microsoft.com/en-us/library/ie/dn621085(v=vs.85).aspx (Methods)
- https://www.khronos.org/files/webgl/webgl-reference-card-1_0.pdf

4- FAMILIARISATION

Note:
Il est suggéré d'utiliser le navigateur Chrome pour tester et déboguer vos programmes.

a) Le lien suivant vous permet d'obtenir tous les exemples montrés durant le cours (fichiers HTML et Javascript).

- Liste de tous les exemples présentés dans le cours d'infographie

Il est à noter que ce lien est également disponible au tout début de la page des notes de cours.

b) Téléchargez le fichier ZIP suivant :

- Exemples-travail1.zip

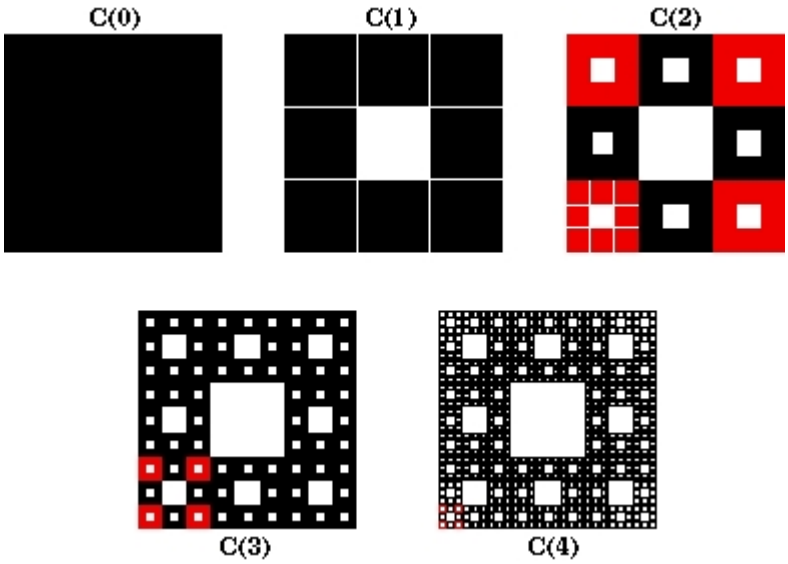
c) Faites l'extraction de tous les dossiers et fichiers contenant dans cette archive ZIP.
d) Dans le dossier « Exemples-travail1 », créez un dossier nommé « Travaux ». Ce dossier devrait être au même niveau que le dossier « Common » provenant de l'archive ZIP.
e) Copiez les fichiers « triangle.html » et « triangle.js » (situés dans le dossier « Class ») dans le nouveau dossier « Travaux ».
f) Modifiez les fichiers « triangles.html » et « triangles.js » de telle sorte qu'ils permettent d'afficher deux triangles de couleur rouge.
    - Pour modifier les fichiers HTML et Javascript, vous pouvez utiliser les logiciels Notepad++ ou Atom.
    - Pour visualiser le résultat, il suffit de double-cliquez sur le fichier HTML une fois vos modifications réalisées.
    - Pour déboguer vos fichiers Javascript, vous pouvez accéder à l'outil WebGL Inspector en cliquant-droit dans la zone du canvas et en sélectionnant « Inspecter l'élément ». N'oubliez pas de désactiver la « cache » sous l'onglet « Network » du débogueur.
g) Une fois le travail réalisé, comparez votre programme Javascript et votre fichier HTML à ceux présents dans le dossier « Class ».
    - « 2trianglesa.html » et « 2trianglesa.js »
    - « 2trianglesb.html » et « 2trianglesb.js »
    - « 2trianglesc.html » et « 2trianglesc.js »
h) Sauvegardez une copie de vos fichiers.
i) Modifiez vos fichiers afin de pouvoir afficher un carré de couleur rouge.
    - Utilisez tout d'abord des triangles pour afficher le carré
    - Ensuite, utilisez un « éventail » (TRIANGLE_FAN) pour afficher le carré. Vous devrez modifier la liste de vos sommets en conséquence.
    - Finalement, utiliser une « bande » (TRIANGLE_STRIP) pour afficher le carré. Dans ce cas également, vous devrez modifier la liste de vos sommets.
j) Vous pouvez comparer votre fichier Javascript (pour l'éventail) avec le fichier « square.js » présent dans le dossier « Class ».
k) À partir d'une copie des fichiers permettant d'afficher deux triangles rouges, modifiez ces derniers afin de pouvoir afficher un triangle rouge et un triangle vert.

    Deux options sont possibles : 1) Utiliser une variable « uniforme » pour modifier la couleur. 2) Utiliser deux « fragment shaders ». Les fichiers « 2trianglescouleurdifferente.html » et 2trianglescouleurdifferente.js » présents dans le dossier « Class » montrent comment on peut réaliser l'option 2.

5- TRAVAIL À REALISER

A. Double-cliquez sur le fichier gasket2.html localisé dans le dossier « Class ». Le patron (« gasket ») de Sierpinski devrait être affiché.
B. Étudiez les « shaders » du fichier gasket2.html ainsi que le programme Javascript qui y est associé.
C. En s'inspirant de l'algorithme mis en œuvre dans gasket2.html, développez un programme permettant d'afficher le "Sierpinski Carpet" en **deux dimensions** (2D).

   Ce motif est une fractale qui s'obtient en subdivisant récursivement un carré de la manière montrée à la figure de la page suivante.

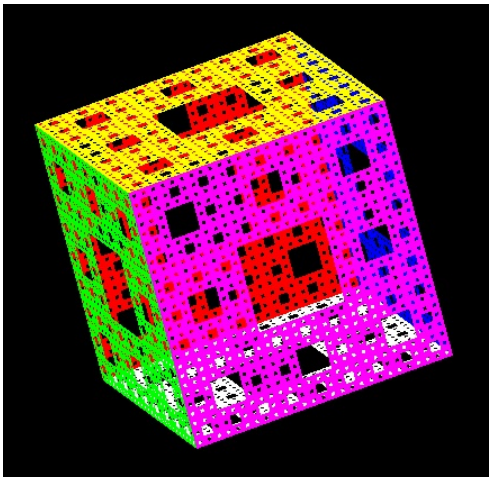


D'autres fractales du même genre peuvent être créées relativement facilement. Le lien suivant en présente quelques autres.

http://ecademy.agnesscott.edu/~lriddle/ifs/carpet/carpet.htm

Par exemple, cliquez sur « Koch Snowflake » présent dans la barre supérieure pour voir comment on peut créer un flocon de neige.

D. Modifiez légèrement votre programme afin d'afficher le « tapis » (« carpet ») en **trois dimensions** (fixez la profondeur Z de vos sommets à zéro).
E. Construisez par la suite un cube dont chacune des faces présente un "Sierpinski Carpet". Le résultat devrait ressembler à la figure suivante :



Veuillez noter que pour bien visualiser le résultat, il vous faudra donner une couleur différente à chaque face. De plus, veuillez remplacer le « vertex shader » que vous avez utilisé jusqu'à présent par le suivant :

```
<script id="vertex-shader" type="x-shader/x-vertex">
attribute vec4 vPosition;
uniform mat4 projection;
void
main()
{
   vec3 theta = vec3(30.0, 20.0, 0.0);

   vec3 angles = radians( theta );
   vec3 c = cos( angles );
   vec3 s = sin( angles );

   // 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

   gl_Position =  projection * rx * ry * rz* vPosition;

}
</script>
```

Insérez également les lignes de code suivantes dans votre fichier Javascript (tout juste avant de dessiner chacune des faces) :

```
var pMatrix = ortho(-1.0, 1.0, -1.0, 1.0, -1.0, 1.0);
var projectionLoc = gl.getUniformLocation(program, "projection");
gl.uniformMatrix4fv(projectionLoc, false, flatten(pMatrix));
```

Pour votre culture personnelle (ceci n'est pas un travail à réaliser) :

a) *Si on poussait l'algorithme un peu plus loin en créant des cubes de manière récursive, on pourrait obtenir l'éponge de Menger (Menger's sponge).*

b) *La création d'un terrain possédant du relief peut s'obtenir facilement à l'aide de fractales. Dans ce cas, on subdivise un grand carré en quatre carrés égaux. On déplace ensuite les sommets à l'aide de nombres aléatoires (en utilisant la méthode Math.random() du langage Javascript. On applique récursivement cette subdivision sur les quatre « carrés » obtenus en réduisant progressivement les déplacements. À la fin, les petits « carrés » sont divisés en deux triangles qui sont ensuite colorés.*

   On peut appliquer cette technique à des triangles pour obtenir récursivement des flancs de montagnes.

   Exemple : https://en.wikipedia.org/wiki/Fractal_landscape

6- **RAPPORT**

Créez un fichier d'archive (ZIP) contenant tous les fichiers requis pour visualiser les images demandés aux étapes C, D et E de la section 5.

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 ("téléversement")