# UNIVERSITÉ DU QUÉBEC À CHICOUTIMI 6GEN719 – INFOGRAPHIE

**Travail 1 – Introduction à WebGL**

---

**Date de remise des programmes de la section 5** (ci-dessous): 22 septembre 2025

---

## 1- OBJECTIF

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

## 2- MODALITÉ PARTICULIÈRE

Ce travail doit être réalisé individuellement.

***Il est cependant possible de commencer vos programmes en utilisant un des programmes donnés en exemple dans le cours. Le logiciel MOSS (qui permet la détection de plagiat) permet d'ignorer toute ressemblance à des programmes fournis en exemple, car il s'agit d'une pratique fréquente dans l'apprentissage d'un langage.***

## 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:

- https://developer.mozilla.org/fr/docs/Web/API/WebGLRenderingContext

  Note importante: Le « WebGLRenderingContext » mentionné à plusieurs endroits dans le document réfère simplement à la variable « gl. » utilisée dans les programmes de démonstration vus dans le cours.

- https://www.khronos.org/files/webgl20-reference-guide.pdf

## 4- FAMILIARISATION

**Notes importantes :**

⇨ **Le travail de cette section ne peut être réalisé qu'après vous avoir familiarisé avec les paramètres *TRIANGLE_FAN* et *TRIANGLE_STRIP* du chapitre 2.**

⇨ **Il est recommandé d'utiliser Visual Studio Code pour prévisualiser et déboguer votre programme.**

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

   Pour télécharger tous ces exemples, cliquez ici.

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 Visual Studio Code (avec l'extension Live Preview).
   - Si vous n'utilisez pas l'extensions Live Preview avec VS Code, pour visualiser le résultat, il suffit de double-cliquer sur le fichier HTML une fois vos modifications réalisées.
   - Pour déboguer vos fichiers Javascript, vous pouvez accéder à l'outil de débogage 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 aux fichiers suivants, présents dans le dossier « Class ». Ces derniers montrent différentes manières de créer les deux triangles. Il est important de bien comprendre chacune des méthodes.
   - « 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 deux triangles pour dessiner le carré.
   - Ensuite, utilisez un « éventail » (TRIANGLE_FAN) pour afficher le carré. Vous devrez modifier la liste de vos sommets en conséquence. Vous

pouvez ensuite comparer votre fichier Javascript avec le fichier « square.js » présent dans le dossier « Class ».

- Finalement, utilisez une « bande » (TRIANGLE_STRIP) pour afficher le carré. Dans ce cas également, vous devrez modifier la liste de vos sommets.

j) On peut dessiner les deux triangles de telle sorte qu'ils aient une couleur différente. Deux méthodes sont possibles pour y arriver.

1- La première consiste à utiliser deux « fragment shaders ». Étudiez attentivement les fichiers « 2trianglescouleurdifferente.html »et « 2trianglescouleurdifferente.js » (présents dans le dossier « Class ») pour comprendre comment il est possible de gérer plusieurs« shaders ».

2- La seconde méthode consiste à utiliser une variable dite « uniform ». Nous décrirons en détail ce genre de variable en classe très bientôt. Vous pouvez cependant étudier les fichiers « 2triangles-using-uniform-qualifier.html » et « 2triangles-using-uniform-qualifier.js » pour voir comment on peut modifier ce type de variable.

(suite à la page suivante)

## 5- TRAVAIL À REMETTRE

**Note importante :**

- ⇨ **Le travail de cette section ne peut être réalisé qu'après avoir vous avoir familiarisé avec le chapitre 2.**
- ⇨ **Si vous tentez d'effectuer ce travail avant de vous avoir familiarisé avec le chapitre 2, vous éprouverez très certainement des difficultés et perdrez du temps inutilement.**

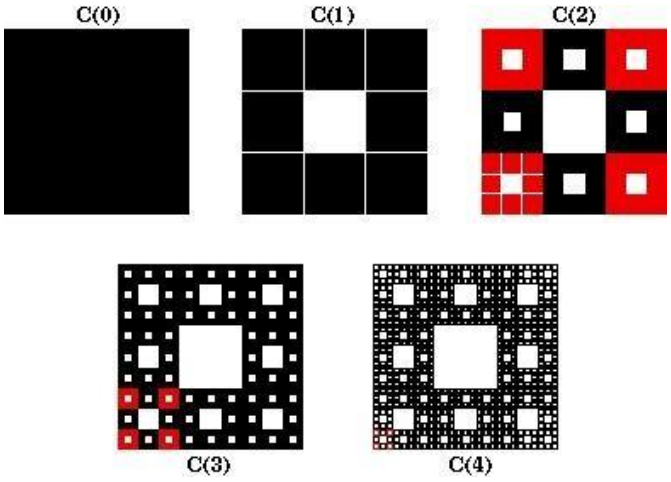
*Rappel : Comme cela a été mentionné lors de la présentation du plan de cours, le plagiat entre étudiants (même partiel) entraînera les conséquences prévues par l'institution. Le logiciel MOSS sera employé pour identifier les sections de code présentant des similitudes. Ce dernier comparera votre programme à celui de tous les autres étudiants,* ***incluant ceux des années précédentes****.*

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). Veuillez utiliser des triangles (gl.TRIANGLES) pour dessiner vos carrés.

**Suggestions :**

- **Vous pouvez utiliser les fichiers « *gasket2.html* » et « *gasket2.js* » comme points de départ pour votre programme.**
- **La fonction *mix( )* vous permettra de calculer facilement les points sur les arêtes.**
- **Pour les quatre points du carré intérieur, il est suggéré d'utiliser les diagonales du grand carré extérieur.**
- **Ne pas utiliser « d'éventails » (TRIANGLE_FAN) ou de « bandes » (TRIANGLE_STRIP).**

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



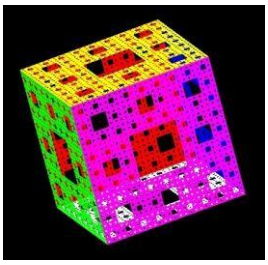

D'autres fractales du même genre peuvent être créées relativement facilement. Le lien suivant en présente quelques autres.

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 celui de la page suivante (Attention lors de l'opération « copier/coller » … vérifiez bien que toutes les lignes ont été correctement copiées…).

Notez que le fonctionnement de ce « shader » sera expliqué dans les prochaines semaines. Ce « shader » tournera votre cube.

```
<script id="vertex-shader" type="x-shader/x-vertex">#version 300 es
in 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 );
    // Rappel : ces matrices sont en ordre colonne-major (les colonnes sont
stockées les unes après les autres)
    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));
```

**L'exemple suivant montre le dessin d'un cube utilisant le « shader » précédent et les trois lignes de code Javascript :** cube-shader-rotation-travail1.html

⇨ Cliquez ici pour obtenir les fichiers de cet exemple

***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 aussi appliquer cette technique à des triangles pour obtenir récursivement des flancs de montagnes.*

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

---

## 6- DOCUMENTS À REMETTRE

Créez un fichier d'archive (ZIP, 7Z ou RAR) contenant tous les fichiers requis pour visualiser les images demandées aux étapes C et E de la section 5. N'oubliez pas d'inclure votre nom au début de votre fichier *Javascript*.

**Veuillez inclure le dossier « Common » dans votre fichier d'archive de telle sorte que l'extraction du contenu de votre fichier ZIP permette 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 ».**

Transmettez le fichier résultant sur le site Moodle du cours.

REMISE DES TRAVAUX (Moodle)