TD 5 (sur machine) : préparation TP shader

1 Exercice 1.1 :

1. Récupérez sur plubel le fichier starterKitShaderSDLShader
2. Testez la compilation.
3. Ajoutez le code pour déterminer votre version d'openGL (AFAIRE 1)

2 Exercice 1.2 :

1. Faites un vector contenant les sommets d'un cube (entre les sommets (-.5,-.5,-.5) et (.5, .5 ,.5)) en utilisant la structure Sommet (AFAIRE 2).
2. Faites un vector d'index représentant le cube avec des faces triangulaires (AFAIRE 3).
3. Créez 1 VBO, un IBO et un VAO contenant les descriptions des données (AFAIRE 4).
4. Affichez les données du VAO (AFAIRE 5).
5. Créez un vertex shader et un fragment shader (AFAIRE 6)
6. Modifiez le fragment shader pour afficher la couleur rouge pour chaque fragment

3 Exercice 1.3

1. Modifiez le vertex shader pour faire suivre les positions (x,y,z) des sommets au fragment shader
2. Modifiez le fragment shader pour afficher chaque fragment de la couleur correspondant à ces coordonnées (x,y,z) (i.e. rouge=x, vert = y , bleu = z).

4 Exercice 2.1 : 1 buffer pour tous les attributs

1. Ajoutez dans la structure Sommet la couleur (r,g,b)
2. Dans le vector des sommets, affectez des couleurs différentes à chaque sommet.
3. Vérifiez que le VBO peut recevoir toutes ces nouvelles données.
4. Modifiez le VAO pour envoyer à la carte graphique l'attribut couleur.
5. Modifiez le vertex shader pour récupérer et faire suivre au fragment shader la couleur de chaque sommet.
6. Modifiez le fragment shader pour afficher la couleur du fragment (interpolation des couleurs des sommets).

5 Exercice 2.2 :

1. Dans le pg openGL, à l'aide des touches R et r (resp. V,v, et B,b), augmentez et diminuez la valeur d'une variable de type Glfloat entre 0 et 1 varRouge (respec. VarVert, varBlue).
2. Transférez les valeurs de ces 3 variables aux programmes shaders.
3. Récupérez ces valeurs dans le fragment shader et l'utilisez pour modifier la valeur de la couleur affectée à chaque fragment.

6 Exercice 3.1 : GLM

1. A l'aide de la bibliothèque GLM, créez les matrices Perspective, View et model.
2. Transmettez ces matrices aux programmes shaders, et appliquer les transformations à chaque sommet.
3. Utilisez la gestion du cliquer-glisser implémenté dans le programme sarterkit pour modifier
    1. version 1 : la matrice model pour faire tourner le cube sur lui-même.

2. version 2 : la matrice View pour que la caméra tourne autour du cube.

7 Exercice 3.2 : texture procédurale

1. Ajoutez des coordonnées de texture aux sommets pour les faces quadrangulaire de manière a ce que chaque face soit projeté dans l'espace uv sur le carré allant de (-2,-2) à (2,2)
2. Modifiez le fragment shader pour afficher l'ensemble de Mandelbrot sur chaque face.
3. Faire une variante pour afficher un ensemble de Julia défini à partir d'une valeur complexe (C=cx + i cy) passé en paramètre au shader et dont les valeurs sont modulées par les touches clavier (x,X et y,Y).