Informatique graphique

L'ensemble des travaux pratique est effectué sous **Shader Toy** www.shadertoy.com en GLSL dont la syntaxe est proche du langage C++. Plusieurs types de données fondamentaux sont définis, comme les vecteurs dans le plan et l'espace *vec2* et *vec3*, ou les matrices *mat3*, et certaines fonctions comme le produit scalaire *dot* ou vectoriel *cross*, ou la longueur d'un vecteur *length*.

Forme : le rendu se fera sous la forme d'un document PDF de deux pages maximum résumant les principaux éléments développés. Il contiendra le nom, prénom, numéro d'étudiant, les pointeurs vers l'adresse web du shader, plusieurs captures d'écran présentant les différents éléments de la scène (éventuellement pris avec différents points de vue), l'organisation de la scène et les statistiques permettant d'évaluer la complexité de la scène : nombre de primitives ou de nœuds dans l'arbre de construction, nombre de textures, temps de génération d'une image ou nombre d'images par seconde…

Modélisation de formes simples (TP1)

L'objectif est de modéliser des formes simples sous la forme de surfaces implicites représentées par des fonctions scalaires dans l'espace.

Concepts : modélisation de formes par fonctions scalaires, transformations, assemblage par opérateurs booléens.

Référence : www.shadertoy.com/view/Wty3z1

Modélisation : créer des formes simples en écrivant des fonctions calculant la distance signée à un objet : implémenter les primitives sphère, boite, tore, cylindre, capsule, plan. Coder les formes polyédriques construites à partir de plan.

Assemblage et placement : écrire les opérateurs union, intersection, différence permettant de combiner les objets ; éventuellement des opérateurs lisses, définir des opérateurs de déformation selon les transformations rotation, translation, et homothétie.

Hiérarchie : créer des fonctions définissant hiérarchiquement des formes complexes, par exemple une colonne comme assemblage de cylindres, boites et tores, un escalier comme assemblage de boites, et un monument combinant colonnes, plateformes et escaliers.

Texture et shading (TP2)

L'objectif est de définir des textures procédurales (damier, bois, roche, sable, métal rouillé) par combinaison de fonctions de bruit multifréquences. Le code de calcul d'un point de la surface a été modifié de manière à retourner non seulement la fonction scalaire, mais également l'indice de la texture.

Concepts : couleur, texture procédurale, fonctions de bruit, turbulence, équations locales d'éclairement.

Référence : www.shadertoy.com/view/tlyXWW

Eric Galin
eric.galin@liris.cnrs.fr

Texture : Définir des fonctions de texture permettant de calculer la couleur en tout point de l'espace : commencer par une texture uniforme, puis implémenter une fonction damier volumique paramétré par la taille du côté. A l'aide des fonctions de bruit et de mouvement Brownien fractionnaire turbulence, écrire une texture marbre avec et sans veines, et bois avec ou sans nœuds.

Eclairement : Prendre en compte les paramètres ambient, diffus et spéculaires du matériau. Compléter le modèle d'éclairement de manière à gérer les matériaux réfléchissants ou partiellement réfléchissants, ainsi que les matériaux transparents, en changeant les procédures de calcul d'intersection et de shading.

Ombres (TP3)

L'objectif est de coder trois modèles d'ombre différents et complémentaires : les ombres (dures) générées par les sources ponctuelles, les ombres douces provenant de sources (linéaires, surfaciques ou volumiques) étendues, et l'occlusion ambiante.

Concepts : ombre, lancer de rayon.

Référence : https://www.shadertoy.com/view/4XBSR1

Ombres dures : Ecrire une fonction *Shadow* retournant 0.0 si le point est dans l'ombre, et 1.0 si le point est éclairé par une lumière ponctuelle.

Ombres douces : Implémenter différents types de sources étendues : *SoftDisc* approximant une ombre douce et retournant une valeur dans [0,1] pour n points échantillonnant la surface d'un disque, et de même *SoftSegment* et *SoftSquare* pour des éclairages correspondants.

Ecrire une fonction *Fibonacci* construisant le i-ème point d'une série de n sur une sphère. A partir de cette fonction, écrire *SoftShadow* approximant une ombre douce et retournant une valeur dans [0,1]. Tester et comparer les ombres obtenues et les performances pour n variant de 1 à 200.

Occlusion ambiante : A partir de la fonction *Fibonacci*, écrire une fonction *Hemisphere* construisant la i-ème direction unitaire d'une série de n points sur une sphère. Ecrire *Occlusion* approximant l'occlusion ambiante et retournant une valeur dans [0,1]. Tester et comparer les ombres obtenues et les performances pour n variant de 1 à 200.

Combiner les modèles d'ombre et le mettre en œuvre sur des modèles géométriques avec des détails permettant de voir leur impact. Comparer les performances et la qualité des ombres obtenues avec le nombre d'échantillons.

Eric Galin
eric.galin@liris.cnrs.fr