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…

Objectif : coder un lancer de rayon simple sur des objets géométriques, de définir la texture des objets à l'aide de procédures, de mettre en place les équations d'éclairement, de coder des modèles d'éclairement plus complexes prenant en compte la réflexion, et l'occlusion ambiante.

Référence : https://www.shadertoy.com/view/fl2yz3

Modélisation

Primitives : on définira la scène comme une union de primitives simples. Implémenter des formes simples en écrivant un algorithme d'intersection avec un rayon : ellipsoïde, boite (parallèle aux axes du repère monde) cylindre, capsule, tore.

Placement : écrire les opérateurs de transformations affines permettant d'effectuer la rotation, la translation, et l'homothétie d'un rayon, de manière à pouvoir modéliser des primitives orientées de manière quelconque dans l'espace.

Extra : implémenter d'autres formes, par exemple des objets de type fonction de distance signée pour permettre de modéliser des formes complexes, ou des opérations booléennes de différence ou d'intersection entre deux objets simples (ne pas traiter le cas général d'un arbre de construction).

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.

Implémenter une texture produisant des variations de couleur concentriques selon la distance d'un point à un centre, ou radiales selon la distance à un axe.

Extra : A l'aide des fonctions de bruit et de mouvement Brownien fractionnaire turbulence, coder par exemple une texture marbre.

Eric Galin
eric.galin@liris.cnrs.fr

Eclairement

Modèle local : Implémenter le modèle d'éclairement de Phong dans les textures, décrivant une surface non pas seulement par sa couleur mais par un matériau de composantes ambiante, diffuse et spéculaire. Coder une texture damier alternant des cases diffuses et spéculaires.

Ombres : implémenter dans la procédure d'éclairement les ombres pour des sources lumineuses ponctuelles et directionnelles.

Objets réfléchissants : modifier le modèle de matériau de manière à pouvoir définir un matériau mat ou réfléchissant ; modifier en conséquence l'algorithme de rendu pour gérer les rayons réfléchis avec 1 niveau de réflexion.

Occlusion : implémenter le modèle d'éclairement local avec une calcul d'occlusion ambiante.

Extra : Coder les objets transparents et la réfraction, implémenter les réflexions avec n niveaux.

Eric Galin
eric.galin@liris.cnrs.fr