IMN715

Travail pratique # 4
Photon mapping

But :

Implanter l'affichage de caustiques en utilisant le photon mapping.

Directives :

Le travail va se faire en deux étapes, génération des photons et modification du tracé de rayons pour afficher les caustiques.

Pour la génération des photons, vous devez coder la fonction GenerePhotons, située dans le fichier *GenerePhoton.cpp* et *GenerePhoton.h*. Cette fonction est appelée lorsque vous cliquez sur le bouton « genere photons » dans le menu. Elle prend 2 paramètres, la caméra pour les lumières et la scène. Dans cette fonction, il suffit d'émettre des photons des sources lumineuses et d'emmagasiner les photons dans le photon map qui est implémenté par la classe PhotonMap. Vous avez accès au photon map par les « settings » du tracé de photon, **pFenAff3D->PhotonTracing()->PhotonMapCaustique()** (retourne **PhotonMap*)**. Le photon map est défini dans les fichiers *PhotonMap.cpp*, *PhotonMap.h*, *Photon.cpp* et *Photon.h*. Vous pouvez changer le nombre de photons émit en modifiant NB_PHOTON_CAUSTIQUE qui est dans le fichier *SettingTracePhoton.h*.

Pour l'affichage des caustiques, il faudra modifier votre tracé de rayons pour qu'il puisse ajouter la contribution des caustiques.

Photon map

L'utilisation est simple, la méthode **Store** permet d'insérer les photons dans le *photon map*. La méthode **Balance** est appelée à la fin pour créer le kd-tree balancé. Si **Balance** n'est pas appelée, la recherche (locate) de photons ne fonctionnera pas. La méthode **Locate** permet de faire une recherche dans le *photon map*. Elle va rechercher au maximum les *n* photons les plus proches. Ces photons sont pris dans un rayon maximal par rapport à un point donné. Pour la remise, assurez-vous que le nombre de photons est de 200 et que le rayon maximal est de 0.3. Pour connaître exactement les paramètres à passer à ces méthodes du **PhotonMap** consulter le fichier *PhotonMap.h.* La structure d'un photon se trouve dans le fichier *Photon.h.*

Exemples d'utilisation de « store » et « locate » :

```
PhotonMap* CaustiqueMap = pFenAff3D->PhotonTracing()->PhotonMapCaustique();

couleur puissance_photon; point
p ;
vecteur direction; // direction d'où provient le photon
…
direction.normalise();
CaustiqueMap ->Store( puissance_photon, p, direction );

point PointIntersection;
reel rayon=0.3; int
nphotons, found;
…
reel *dist2 = NULL;  // tableau de distances aux carrées
// (attention c'est bien aux carrées).
                     // Ne pas tenir compte de la position 0 dans le tableau,
                     // celle-ci est utilisée pour des données internes
   const Photon **ph = NULL; // tableau de pointeurs de photons.
// Ne pas tenir compte de la position 0

CaustiqueMap->Locate( PointIntersection, rayon, nphotons, found, &dist2, &ph );
…
vecteur dir_photon = ph[i] ->dir();
couleur energie_photon = ph[i]->energie(); // il est plus exacte de parler de
                                           // puissance
…
delete [] dist2; // libérer l'espace mémoire
delete [] ph;
```

Fichier .vsn : modifications aux lumières.

Pour définir le type de lumière désiré on doit le spécifier dans le fichier .vsn. Pour faire la distinction entre un « spotlight » et une lumière ponctuelle, il y a maintenant un identificateur : SP pour un « spotlight » et PC pour une lumière ponctuelle.

Les paramètres associés à un « spotlight » sont :
type SP position (x y z)
intensité (r g b) intensité ambiante (r g b) puissance de la lumière (r g b) direction (vx vy vz)
angle d'ouverture (degré)

Les paramètres associés à une lumière ponctuelle sont :
type PC position (x y z)
intensité (r g b) intensité ambiante (r g b)
puissance de la lumière (r g b)

On distingue l'intensité de la lumière et la puissance de la lumière. Pour le tracé de photons, utiliser la puissance de la lumière. Vous utiliseriez la puissance de la lumière si vous tenez compte de la distance lors du calcul de l'illumination locale dans le tracé de rayons (ne faite pas cela dans ce TP, utilisez plutôt l'intensité de la lumière comme lors du TP1).

Utilisez la classe **Lumiere** pour ce tp et non la classe **Camera** (comme au TP1) pour manipuler les lumières. La méthode **RayonAléatoire** de la classe **Lumiere** permet de générer un rayon aléatoire pour une source lumineuse. Cette méthode est utilisée pour générer une direction aléatoire à un photon. Notez qu'un photon sera réfléchi tant qu'il rencontrera une surface ayant une composante miroir (mais sa puissance peut diminuer). Pour vérifier qu'une lumière (en particulier, un spotlight) éclaire un point, vous utiliseriez la méthode **eclaire** dans le tracé de rayons. Pour ce TP, ne pas considérer cette méthode.

Remise :

Soumettre par turnin web les fichiers GenerePhoton.cpp et Rayon.cpp. Le TP doit être remis au plus tard vendredi le **28 octobre**. Ne pas oublier d'écrire votre nom et matricule dans le fichier GenerePhoton.cpp et Rayon.cpp.

Note : Si le photon map est vide, on veut tout de même pouvoir faire le tracé de rayons. Et on veut aussi pouvoir faire plusieurs tracés de rayons sans avoir à générer tous les photons de nouveau.