BRDF Viewer
Documentation programmeur

BRDF Viewer est un outil permettant de visualiser la BRDF produite par différents modèles analytiques existants dans la littérature. Plusieurs modes de visualisation sont proposés, et l'interface permet de modifier de manière interactive l'angle d'incidence et les paramètres des modèles.

Ajouter un nouveau modèle de BRDF

L'ajout d'un nouveau modèle de BRDF est réalisé en 3 étapes :

1/ Création du nouveau modèle (brdf.hpp/cpp)

2/ Référencement dans le système de visualisation (camera.hpp/cpp)

3/ Gestion de l'interface pour la prise en compte (camera.hpp/cpp)

Étape 1 : Création du nouveau modèle

La gestion des modèles de BRDF est réalisée à l'aide des mécanismes d'héritage C++. Le fichier brdf.hpp comporte une classe mère décrivant les prototypes des méthodes à écrire pour le bon fonctionnement de chaque modèle.

```
//====================================================================
// BRDF mother class
//
// WARNING !!!
// Vo corresponds to an observer-pixel direction
//====================================================================
class brdf {
protected:
   char m_values[512];  // for rView system

public:

   virtual float fr(pVect Vi, pVect Vo, pVect N) = 0;
   virtual int nbParam() = 0;
   virtual void resetParam(int i) = 0;
   virtual void changeParam(int i, float val) = 0;
   virtual char *paramValues() = 0;
   virtual ~brdf() {};
};
```

Tous les modèles de BRDF définis dans le programme doivent hériter de cette classe pour assurer le bon fonctionnement.

La méthode principale est *fr*. A partir des directions d'incidence, de réflexion, et de la normale (paramètres Vi, Vo et N), elle définit la valeur de la BRDF (sr^{-1}).

Note : le paramètre Vo doit être donné dans le sens observation-surface.

Les autres méthodes sont des utilitaires pour gérer l'interface graphique :

- *nbParam* retourne le nombre de paramètres du modèle
- *resetParam* modifie le paramètre *i* et le met à la valeur *0.0f*
- *changeParam* incrémente le paramètre *i* de *val*
- *paramValues* retourne une chaine de caractère décrivant le nom du modèle et les valeurs des paramètres (pour l'affichage en bas à droite sur l'interface graphique)

Pour des exemples de mise en œuvre des modèles, l'utilisateur peut s'inspirer des modèles déjà réalisés dans les fichiers brdf.hpp/cpp.

Étape 2 : Référencement d'un nouveau modèle

Lorsque le modèle est terminé, sa prise en compte dans le programme est réalisée dans la classe camera (fichiers camera.hpp/cpp). Dans un premier temps, il doit être déclaré dans la liste d'attributs de la classe camera, avec les autres modèles.

Exemples :

```
// ------------------------------------------------------------
  brdf *m_Fr;
  brdf *m_fLambert;
  brdf *m_fPhong;
  brdf *m_fCookT;
  brdf *m_fOrenN;
  // ------------------------------------------------------------
```

L'allocation des objets correspondant à chaque modèle est réalisée par le constructeur de la classe camera (au début du fichier *camera.cpp*).

```
  m_fLambert = new rLambert();
  m_fPhong = new rPhong();
  m_fCookT = new rCook();
  m_fOrenN = new rOrenNayar();
```

Cette déclaration permet d'avoir en permanence une instance de la brdf en mémoire.

Étape 3 : Gestion du nouveau modèle dans l'interface

La dernière étape consiste à prendre en compte le nouveau modèle lorsque l'utilisateur demande le changement. Ce processus nécessite uniquement de mettre à jour correctement le pointeur m_Fr (la BRDF active) dans la méthode *void camera::changeModel()*.

A partir de cette étape, le programme doit être fonctionnel et la nouvelle BRDF peut être prise en compte.