Recette - Manuel

Global Illumination with Radiance Regression Functions

Etudiants :
Yannick BERNARD
Pierre GUERINEAU
Kevin MENIEL
Romain MOUTRILLE
Matthias ROVES

Encadrant :
Mathias PAULIN

28 février 2017

Table des matières

1 Préparation

1.1 Cloner le dépôt GitHub

```
git clone https://github.com/Zethzer/GIRadianceRegressionFunction.git --recursive
```

1.2 Compilation

Pour compiler le logiciel, il faut compiler les différents modules indépendamment via CMake. Il y a un *CMakeLists.txt* disponible pour chacun :

— PBRT

— LatinHypercube

— Learning module

— Engine/AtlasEngineGI

2 Extraction

2.1 Hypercube latin

Le programme permettant l'échantillonage des positions de caméra et de sources lumineuse par hypercube latin est appelé de la manière suivante :

```
./LatinHypercube <file> <nbCameraSample> <nbLightSample>
```

— `file` : chemin vers le fichier scène au format *.pbrt* v3

— `nbCameraSample` : nombre d'échantillons de positions de la caméra

— `nbLightSample` : nombre d'échantillons de positions des pointlights

On a en sortie les fichiers scènes `.pbrt` v3 correspondant à toutes les configurations caméra/pointlight échantillonnées par l'hypercube latin. Ceux-ci sont créés dans le répertoire courant. Le nom de ces fichiers est construit ainsi :

`scene_<`i_c `*` n_c `+` i_l`>.pbrt` avec i_c et i_l les indices de caméra et de lumière courants et n_c le nombre d'échantillons de caméra.

2.2 Extraction des données d'apprentissage

L'exécution de PBRT sur un fichier scène peut se faire simplement avec l'appel suivant :

```
./pbrt <file>
```

— `file` : chemin vers le fichier scène au format *.pbrt* v3

Les différentes options possibles de cet appel sont disponibles dans la documentation de PBRT, utiliser l'option `-h` permet de les lister, elles ne seront pas détaillées ici. L'appel de cette fonction devant être effectué pour chaque fichier de l'échantillonnage par hypercube latin, cette tâche fastidieuse a été facilitée par la création d'un script *bash*.

Le fichier *.pbrt* à utiliser pour obtenir un échantillonnage de Fibonacci doit utiliser l'intégrateur `"CO_path"` et la caméra `"fibonacci"`. La taille de l'image déterminera le nombre de samples autour de la sphère. Le nom du fichier de sortie doit porter l'extension `".data"` pour obtenir les données d'apprentissage au format ASCII, elles se présenteront sous cette forme :

```
<camera_position.x> <camera_position.y> <camera_position.z>
<light_position.x> <light_position.y> <light_position.z>
<frag_pos.x> <frag_pos.y> <frag_pos.z> <normal.x> <normal.y> <normal.z> <R> <G> <B> *N
```

— `camera_position` représente les coordonnées de la caméra
— `light_position` représente les coordonnées de la pointlight
— `frag_position` représente la position du fragment dans la scène 3D
— `normal` représente la normale du fragment
— `R, G, B` représentent l'éclairage indirect du fragment
— N est le nombre de données, donc le nombre de lignes.

3 Script

Le script *script.sh* permet l'automatisation du rendu avec PBRT des scènes générées précédemment. Il permet de lancer l'échantillonnage par hypercube latin sur une scène, puis de lancer des rendus PBRT sur plusieurs fichier.
Le script est appelé de la manière suivante :

```
./script.sh -pbrt <LatinHypercubeExec> <SceneInputFile> <NbCamSample> <NbLightSample>
./script.sh -data <PBRTExec> <lowRange> <highRange>
```

L'option `-pbrt` permet de réaliser l'échantillonage par hypercube latin :
— `LatinHypercubeExec` : chemin vers l'executable LatinHypercube
— `SceneInputFile` : chemin vers le fichier scène au format *.pbrt*
— `NbCamSample` : nombre d'échantillons de positions de la caméra
— `NbLightSample` : nombre d'échantillons de positions des pointlights

L'option `-data` permet l'automatisation du rendu des fichiers pbrt créés :
— `PBRTExec` : chemin vers l'exécutable de PBRT
— `lowRange, highRange` : paramètres permettant d'effectuer les rendus sur un sous-ensemble des données

On a à la fin de l'execution du script autant de fichiers *.data* que de fichiers scènes générés par l'échantillonnage par hypercube latin.

4 Apprentissage

Le module d'apprentissage est appelé de la manière suivante :

```
./LearningModule <path_to_training_folder>
```

— `path_to_training_folder` : chemin vers le répertoire contenant les fichiers *.data* créés dans le module d'extraction.

Un fichier de configuration des réseaux de neurones *config.xml* doit être placé dans le répertoire, il est de la forme :

```
<Config>
   <Network>
      <Architecture></Architecture>
      <LayersActivationFunctions></LayersActivationFunctions>
      <InputsNames></InputsNames>
      <OutputsNames></OutputsNames>
      <ScalingLayerMethod></ScalingLayerMethod>
      <UnscalingLayerMethod></UnscalingLayerMethod>
   </Network>
   <DataSet>
      <Splitting>
         <SplittingMethod></SplittingMethod>
         <SelectionRatio></SelectionRatio>
         <TestingRatio></TestingRatio>
      </Splitting>
      <Training>
         <ScalingMethod></ScalingMethod>
         <TrainingMethod></TrainingMethod>
         <GradientNormGoal></GradientNormGoal>
      </Training>
   </DataSet>
</Config>
```

- `Architecture` : `<nb_inputs> <nb_perceptrons_layer_1> ... <nb_perceptrons_layer_n> <nb_outputs>`
- `LayersActivationFunctions` : `[Threshold | SymmetricThreshold | Logistic | HyperbolicTangent | Linear] * number of layers - 1`
- `InputsNames` : liste de noms de taille le nombre d'inputs
- `OutputsNames` : liste de noms de taille le nombre d'outputs
- `ScalingLayerMethod` : `[NoScaling | MinimumMaximum | MeanStandardDeviation]`
- `UnscalingLayerMethod` : `[NoUnscaling | MinimumMaximum | MeanStandardDeviation]`
- `SplittingMethod` : `[Random | Sequential]`
- `SelectionRatio` : double , $0 \leq ratio \leq 1$
- `TestingRatio` : double, $0 \leq ratio \leq 1$, $(ratio\_selection + ratio\_test) \leq 1$
  $(ratio\_selection + ratio\_test + ratio\_apprentissage) = 1$
- `ScalingMethod` : `[MinimumMaximum | MeanStandardDeviation]`
- `TrainingMethod` : `[QuasiNewton | LevenbergMarquardt | GradientDescent | ConjugateDescent | NewtonMethod]`
- `GradientNormGoal` : double

La progression de l'apprentissage est inscrite dans un fichier *training.log* créé automatiquement *(s'il n'existe pas)*. Ce fichier contient les noms des fichiers *.data* déjà parcourus. Si l'apprentissage est arrêté avant la fin *(avant le traitement de tous les fichiers .data)*, il faut le reprendre :
Il faut supprimer manuellement le réseau de neurones le plus ancien parmis les deux fichiers *neuralnetworksave1.xml* et *neuralnetworksave2.xml*. Le programme reprendra l'apprentissage du réseau de neurones restant lors de l'exécution.
A la fin de l'exécution, le programme va sauvegarder le réseau de neurones dans un fichier *neuralnetwork.xml*.

Pour recommencer l'apprentissage de zéro, il est nécessaire de supprimer le fichier *training.log* et les deux fichiers *neuralnetworksave1.xml* et *neuralnetworksave2.xml*.

5 Rendu

5.1 Chargement de scènes PBRT

Le moteur de rendu temps réel prend en paramètre le fichier scène au format *.pbrt* :

```
./AtlasEngineGI <file>
```
— `file` : chemin vers le fichier scène au format *.pbrt* v3

Le contenu du fichier de scène minimal est, par exemple, comme celui-ci :

```
WorldBegin
   MakeNamedMaterial "Shape1" "string type" [ "matte" ] "rgb Kd" [ 1.0 1.0 1.0 ]
   NamedMaterial "Shape1"
   Shape "trianglemesh" "integer indices" [ 0 2 1 2 0 3 ] "point P" [ 10 0 -10 10 20 -10 10 20 10 10
       0 10 ] "normal N" [ -1 0 0 -1 0 0 -1 0 0 -1 0 0 ] "float uv" [ 0 0 1 0 1 1 0 1 ]
   AttributeBegin
      LightSource "point" "rgb I" [ 17 12 4 ] "point from" [ 0 18 0 ]
   AttributeEnd
WorldEnd
```

Il s'agit ici d'une scène représentant un carré blanc (Shape1) avec une pointlight d'une couleur légèrement jaune à la position (0,18,0).

Notes :

1. Il est primordial de respecter la syntaxe : `"parametre" [ valeur1 valeur2 ]`
2. `integer indices` contient le vecteur d'indices pour dessiner le carré à l'aide triangles.
3. `point P` contient les points nécessaires *(minimum 4 ici)* pour dessiner le carré.
4. `normal N` contient les normales de chaque points.
5. `float uv` contient les uv de chaque points.
6. L'intensité de la lumière est égale au carré de la norme de la couleur (à 3 composantes) calculée lors du chargement de la scène, ie. $i = ||color||_2^2, color = [rgb]^T$
7. Pour cet exemple, il faudra reculer la caméra. Car elle se trouve de base dans le carré (à travers).

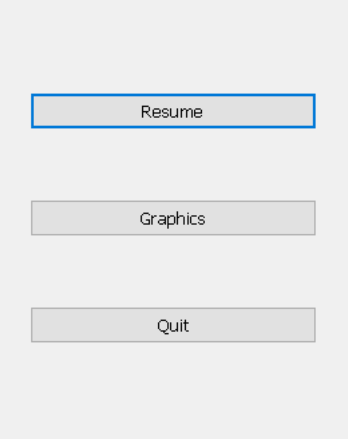
(Pour plus d'informations sur la construction d'un fichier scène .pbrt, consulter documentation de PBRT*)*

5.2 Touches du clavier

Déplacements caméra	
Z	Avancer la caméra
Q	Amener la caméra vers la gauche
S	Reculer la caméra
D	Amener la caméra vers la droite
A	Descendre la caméra
E	Monter la caméra
Déplacements source lumineuse	
I	Descendre la source lumineuse
P	Monter la source lumineuse
O	Reculer la source lumineuse
K	Amener la source lumineuse vers la droite
L	Avancer la source lumineuse
M	Amener la source lumineuse vers la gauche
Autres	
F	Activer/Désactiver le mode plein écran
Echap	Ouvrir le menu

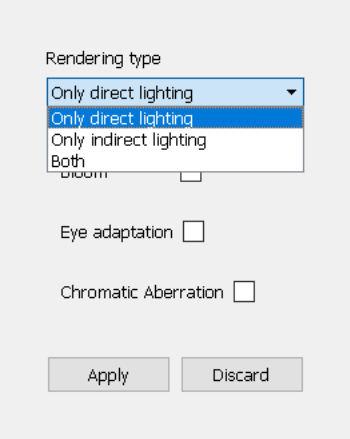
5.3 Menu

Le menu du moteur, accessible avec la touche *Echap* est minimaliste, mais permet d'accéder à différentes options graphiques. Nous pouvons par exemple activer ou désactiver le *bloom*, l'*eye adaptation*, ou encore la *chromatic aberration*. Mais principalement, le menu permet de choisir le mode de rendu, à savoir seulement l'éclairage direct, seulement l'éclairage indirect, ou les deux. Ce choix s'effectue à partir d'une liste déroulante. Le changement d'un paramètre graphique donne un résultat immédiat sur le rendu.



Ouverture du menu

Menu Graphics



Choix du mode de rendu