BORGOBELLO Boris
FEIGLER Thomas

Projet Photon-Mapping
Documentation utilisateur

Sommaire

1. Résumé des capacités

Ce logiciel vous permet de générer des images réalistes avec comme point de vue une caméra conique (comme la vision d'un œil) ou planaire.
Cette caméra regarde une scène, c'est-à-dire un ensemble de lumières et d'objets, qui vont être interprétés pour générer le rendu final à l'aide des méthodes de raytracing et de photon-mapping.

Le fichier d'entrée est donc une description de la scène écrite au format YAML (voir section 3), et génère une image en sortie au format .TGA.

Organigramme récapitulatif

2. Installation

Il y a deux solutions pour installer ce logiciel.

2.1 Exécution immédiate

Le projet a déjà été compilé pour les plateformes suivantes et peut donc être exécuté directement.

Plate-forme	Nom de l'exécutable
Linux 64 bits (Debian)	photon_mapping_deb_64
Linux 64 bits (Fedora)	photon_mapping_fed_64

Dans les exécutables sont liés statiquement toutes les bibliothèques nécessaires au fonctionnement du photon mapper.

2.2 Compilation

Dans le cas où aucun fichier déjà compilé ne pourrait satisfaire la plate-forme utilisée, les fichiers sources sont disponibles et requièrent l'utilisation de librairies libres, disponibles notamment sur internet.

2.2.1 Récupération des bibliothèques

Les bibliothèques suivantes sont requises :

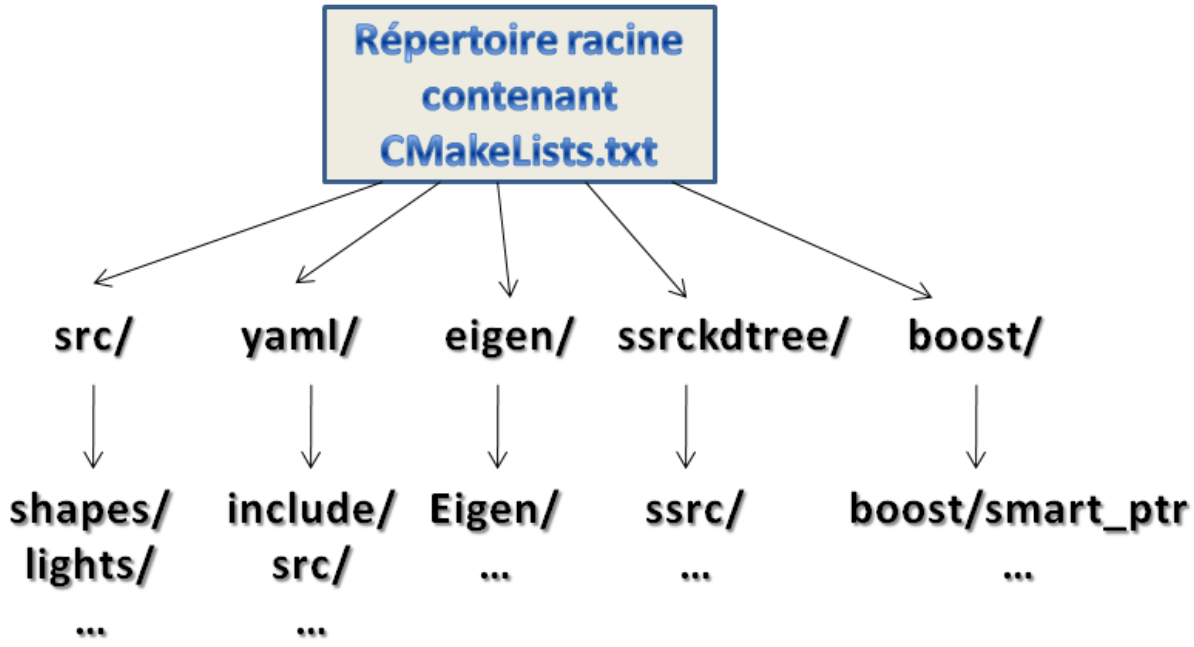
Nom de la librairie	Type	Emplacement
Eigen 2	bibliothèque d'algèbre linéaire	http://eigen.tuxfamily.org/index.php

Yaml-cpp 0.2.5	utilisé comme langage de description des scènes	http://code.google.com/p/yaml-cpp/
Boost 1_42_0 (shared_ptr)	bibliothèque de pointeurs intelligents pour la mémoire	http://www.boost.org/
ssrckdtree	bibliothèque de recherche optimisée des k plus proches voisins	http://www.savarese.com/software/libssrckdtree/

2.2.2 Compilation du logiciel

La compilation peut se faire à partir d'un IDE (nous recommandons Code ::blocks).

Un fichier CMakeLists.txt est également fourni pour les autres distributions linux.
Dans ce cas, il est nécessaire de placer correctement les librairies par rapport aux fichiers sources du photon-mapper comme suit :



Dans le cas ou la librairie est déjà installée nativement sur l'ordinateur, il est quand même nécessaire de récupérer et placer toutes les librairies suivant cette hiérarchie.

Les logiciels Cmake et Make doivent être installés pour que l'installation fonctionne.

Instructions et commandes de compilation :

1) Se placer dans le dossier racine via un terminal (dossier contenant CMakeLists.txt)
2) Exécuter la commande **cmake .** (le point est important car il indique le répertoire courant)
3) Editer le fichier CMakeCache.txt et changer la variable « CMAKE_CXX_FLAGS :STRING= » en « CMAKE_CXX_FLAGS :STRING=-std=c++0x » afin d'autoriser la future version du langage C++.
4) Exécuter la commande **make** qui construit l'exécutable final du nom de **photon_mapper**

Erreurs potentielles :

- Si **cmake** n'est pas une commande reconnue, il est nécessaire d'installer le logiciel cmake via les dépôts de la distribution linux utilisée ou via le site officiel : http://www.cmake.org/
- Si **cmake** demande de choisir le générateur utilisé, taper **cmake** propose à l'utilisateur une liste de générateurs potentiels.
  Exécuter alors **cmake . -G''X''** où X est le générateur choisi.
  Si le problème persiste, se référer à la documentation du site officiel.
- Si **make** indique des erreurs de dépendance (fichiers non trouvés), les librairies sont certainement mal placées. Revérifier la hiérarchie des fichiers.
- Si la commande « -std=c++0x » n'est pas reconnue, la version du compilateur C++ utilisé est trop ancienne ou la commande n'est plus nécessaire.

3. Ecriture d'un fichier de description de scène

3.1 Syntaxe YAML

Ce logiciel utilise deux outils fondamentaux :
- Les tableaux traductibles par « à tel élément correspond telle donnée »
- Les listes qui sont une succession d'éléments

Dans cette partie nous appellerons « chaîne » une succession de caractère ASCII, un mot ou un ensemble de mots, et appellerons « donnée » une chaîne, un nombre entier, un nombre à virgule, un tableau ou une liste.

3.1.1 Tableaux

Un tableau fait correspondre une donnée à une chaîne. Il est écrit différemment si la donnée est un tableau ou une liste (composé).

```
chaîne : donnée
chaîne :
        {données}
```

Dans le second cas, l'écriture s'accompagne d'un saut à la ligne et d'une indentation de **deux espaces** par rapport au début de la chaîne. Dans un but de clarté, les espaces sont des indentations dans ce manuel.

Exemple composé :

```
chaîne :
        chaîne :
                chaîne : donnée
```

3.1.2 Listes

La liste s'écrit également de deux façons :

La représentation typique d'un vecteur à trois dimensions :

[donnée1, donnée2, donnée3]

La liste d'objets :

-donnée1
-donnée2
-donnée3

Exemple composé de tableaux et de listes :

chaîne1 :
    -donnée1
    -chaîne2 :
        - donnée2
    - chaîne3 : [donnée3, donnée4, donnée5]
chaîne4 : donnée6

3.2 Ecriture orientée pour le photon-mapper

Les fichiers utiles pour le photon-mapper prennent par défaut l'extension .sdf, afin de ne pas les confondre avec des fichiers textes .txt

Ces fichiers se composent au maximum de 6 sections écrites en majuscules correspondants au tableau principal.

- LOADING
- CAMERAS
- OBJECTS
- LIGHTS
- TEXTURES
- SCENE

3.2.1 Description des bibliothèques

Bibliothèques

Les sections CAMERAS,OBJECTS, LIGHTS et TEXTURES vont contenir respectivement des caméras, des objets, des lumières et des textures mais ce n'est pas ces sections qui décident de quels objets/lumières/caméra/textures seront utilisés pour le rendu de la scène.
Ces sections doivent être vues comme des bibliothèques dans lesquels il est possible de choisir d'utiliser certaines ressources.

Section LOADING

C'est la source de la récursivité de ces fichiers : tous les fichiers contenus dans la section LOADING seront analysés également et viendront étoffer la bibliothèque. Ces fichiers peuvent eux-mêmes avoir une section LOADING et ainsi venir encore agrandir cette bibliothèque.

C'est d'abord dans un souci de clarté et de confort pour l'utilisateur qu'ont été mises en place ces solutions.
En effet, chaque fichier n'a pas besoin de contenir toutes les sections. On peut ainsi faire un fichier d'objets, un autre de lumières, et un dernier de camera, ou encore un objet par fichier.

Section SCENE

La section SCENE ne compte que pour le fichier racine, celui fourni en entrée par l'utilisateur. Il doit obligatoirement être présent car c'est lui qui définit à la fois quels sont les éléments qui seront utilisés dans la scène par rapport à ceux disponibles dans les bibliothèques, mais aussi les différents paramètres de rendu de la scène (résolution de l'image de sortie, anti-aliasing…).

La lecture des fichiers se fait de la manière suivante :

- Vérification de la syntaxe YAML (erreur déclarée à la ligne et colonne près)
- Vérification des objets, des lumières et des caméras de la librairie. Tout attribut manquant ou incorrect sera détecté.
- Le programme affiche un rapport d'erreurs, d'avertissements et d'informations contenant la totalité des détails trouvés durant le parcours entier des fichiers.
- Si tous les éléments décrits dans les fichiers sont corrects, le logiciel analyse la scène et annonce également les erreurs.

3.2.2 Analyse de la scène

La scène contient deux types d'informations :

- Les informations générales et directives relatives à l'image de sortie ou au photon-mapper
- Une liste de noms d'objets, une liste de noms de lumières et le nom d'une caméra de la bibliothèque

Les premières informations possèdent des paramètres par défaut, un oubli sera donc annoncé mais non critique. En revanche, les listes et la caméra sont des éléments fondamentaux de la scène, leur absence implique l'abandon du rendu.
Toute erreur d'existence sera notifiée et le rendu sera abandonné. Les textures n'étant que le support des objets, elles seront vérifiées en même temps que l'existence de l'objet qui la porte.

Précautions :

- Si deux éléments de la bibliothèque possèdent le même nom, ceci n'est pas considéré comme une erreur. Elle ne sera donc pas annoncée et le premier élément trouvé sera celui utilisé. Cela donne toujours plus de flexibilité à l'utilisateur.
- Une lumière de type RadiantVolume utilise un objet. Cet objet doit nécessairement figurer également dans la section objets (erreur détectée et annoncée)

Syntaxe finale d'écriture des sections OBJECTS/LIGHTS/CAMERAS/TEXTURES :

SECTION:
    -nom_element1:
        attribut1: valeur
        attribut2: [valeur1, valeur2, valeur3]
        attribut3: valeur
    -nom_element2:
…

3.3 Catalogue des objets, lumières, caméras et textures

3.3.1 Résumé des bibliothèques

Objets :
- Sphere
- Plane
- Triangle
- Parallelepiped

Lumières :
- GlobalLighting (illumination globale)
- Punctual (source ponctuelle de lumière)
- Hemispherical (source ponctuelle de lumière restreinte à un hémisphère d'émission)
- RadiantVolume (volume émettant de la lumière)

Caméras :
- Conic (identique à l'œil)
- Planar

Textures :
- Color (colorée)
- Checkers (échiquier)
- Bitmap (image externe au format .TGA)

3.3.2 Description détaillée des bibliothèques

Dans la suite seront utilisés les mots suivants :
- Double : nombre à virgule flottante de précision double
- Vecteur : liste de trois Double

Objets

Sphere

Nom du paramètre	Descriptif	Valeur du paramètre
type	type de l'objet	Sphere
texture	nom de la texture utilisée	nom_texture
absorb	probabilité d'absorption	double entre 0 et 1

reflect	probabilité de réflexion	double entre 0 et 1
refract	probabilité de réfraction	double entre 0 et 1
indice	indice du matériau	double
center	centre de la sphère	vecteur
radius	rayon de la sphère	double

Remarque : la somme des probabilités absorb, reflect et refract doit être inférieure ou égale à 1.

Plane

Nom du paramètre	**Descriptif**	**Valeur du paramètre**
type	type de l'objet	Plane
texture	nom de la texture utilisée	nom_texture
absorb	probabilité d'absorption	double entre 0 et 1
reflect	probabilité de réflexion	double entre 0 et 1
transparency	probabilité de laisser passer	double entre 0 et 1
point	un point quelconque du plan	vecteur
normal	la normale au plan	vecteur

Remarque : la somme des probabilités absorb, reflect et transparency doit être inférieure ou égale à 1.

Triangle

Nom du paramètre	**Descriptif**	**Valeur du paramètre**
type	type de l'objet	Triangle
texture	nom de la texture utilisée	nom_texture
absorb	probabilité d'absorption	double entre 0 et 1
reflect	probabilité de réflexion	double entre 0 et 1
transparency	probabilité de laisser passer	double entre 0 et 1
point1	premier point du triangle	vecteur
point2	second point du triangle	vecteur
point3	troisième point du triangle	vecteur
for_volume	fait partie d'un volume	booléen

Remarque : la somme des probabilités absorb, reflect et transparency doit être inférieure ou égale à 1.

Parallelepiped

Nom du paramètre	**Descriptif**	**Valeur du paramètre**
type	type de l'objet	Parallelepiped
texture	nom de la texture utilisée	nom_texture
absorb	probabilité d'absorption	double entre 0 et 1
reflect	probabilité de réflexion	double entre 0 et 1
refract	probabilité de réfraction	double entre 0 et 1
indice	indice du matériau	double
corner	un coin du parallélépipède	vecteur
x_vector	premier vecteur	vecteur
y_vector	second vecteur	vecteur

z_vector	troisième vecteur	vecteur

Remarques :

- la somme des probabilités absorb, reflect et refract/transparency doit être inférieure ou égale à 1.
- les vecteurs portent l'orientation des arêtes
- la taille des arêtes du parallélépipède est donnée par la norme de chacun des vecteurs.

Sources lumineuses

Punctual source

Nom du paramètre	Descriptif	Valeur du paramètre
type	type de lumière	Punctual
color	couleur de la source	vecteur
power	puissance de la source	double
origine	position de la source	vecteur

Hemispherical source

Nom du paramètre	Descriptif	Valeur du paramètre
type	type de lumière	Hemispherical
color	couleur de la source	vecteur
power	puissance de la source	double
origine	position de la source	vecteur
direction	hemisphere choice	vecteur

Global Lighting (illumination globale)

Nom du paramètre	Descriptif	Valeur du paramètre
type	type de lumière	GlobalLighting
color	couleur de la source	vecteur
power	puissance de la source	double

Radiant Volume (volume émettant de la lumière)

Nom du paramètre	Descriptif	Valeur du paramètre
type	type de lumière	RadiantVolume
color	couleur de la source	vecteur
power	puissance de la source	double
volume	nom du volume utilisé	nom_volume

Remarques :

- Le volume utilisé est un parallélépipède ou une sphère ayant été définie dans une section OBJECTS d'un des fichiers parcouru.
- Cet objet doit également avoir été ajouté dans la scène (section SCENE).

Cameras

Les vector1 et vector2 positionnent le plan à partir duquel est vue la scène.

Planar camera (caméra planaire)

Nom du paramètre	Descriptif	Valeur du paramètre
type	type de caméra	Planar
origine	position dans l'espace	vecteur
vector1	premier vecteur de direction	vecteur
vector2	second vecteur de direction	vecteur
size1	première dimension	double
size2	seconde dimension	double

La caméra planaire est un rectangle de vision dans l'espace, un cadre. Size1 et size2 définissent sa taille dans l'espace suivant, respectivement, les vecteurs vector1 et vector2.

Conic camera (caméra conique)

Nom du paramètre	Descriptif	Valeur du paramètre
type	type de caméra	Conic
origine	position dans l'espace	vecteur
vector1	premier vecteur de direction	vecteur
vector2	second vecteur de direction	vecteur
size1	premier angle d'ouverture	double
size2	second angle d'ouverture	double

La caméra planaire est un rectangle de vision dans l'espace, un cadre. Size1 et size2 sont des valeurs exprimées en radians qui représentent les angles d'ouverture suivant, respectivement, la largeur et la hauteur de l'image.
Par exemple, size1 = pi/2 = 1.57 implique une ouverture angulaire maximale de 90° soit +45° et -45° par rapport à la direction de la caméra (produit vectoriel de vector1 et vector2), suivant la largeur.

Textures

Colored (colorée)

Nom du paramètre	Descriptif	Valeur du paramètre
type	type de texture	Color
color	couleur de la texture	vecteur

Checkers (échiquiers)

Nom du paramètre	Descriptif	Valeur du paramètre
type	type de texture	Checkers
text_x	vecteur d'orientation 1	vecteur
text_y	vecteur d'orientation 2	vecteur
x_dim	largeur des carreaux	double
y_dim	hauteur des carreaux	double
even	première couleur	vecteur
odd	seconde couleur	vecteur

Remarque : Les vecteurs d'orientation précisent les directions utilisées lors du mapping de la texture sur les formes.
Par exemple : un plan suivant les vecteurs **x** et **y** devrait avoir une texture étant mappé sur **x** et **y.**

Bitmap (fichier image)

Nom du paramètre	**Descriptif**	**Valeur du paramètre**
type	type de texture	Bitmap
text_x	vecteur d'orientation 1	vecteur
text_y	vecteur d'orientation 2	vecteur
x_dim	largeur dans l'espace	double
y_dim	hauteur dans l'espace	double
file	fichier au format .TGA	chaîne (chemin)

Remarque : Les vecteurs d'orientation précisent les directions utilisées lors du mapping de la texture sur les formes.
Par exemple : un plan suivant les vecteurs **x** et **y** devrait avoir une texture étant mappé sur **x** et **y.**

Exemple d'utilisation

Exemple de l'écriture de deux sphères dans la section OBJECTS :

```
OBJECTS:
    -SPHERE1:
        type: Sphere
        texture: Rouge
        absorb: 0.5
        reflect: 0.5
        refract: 0
        indice: 1.2
        center: [0, 0.4, -0.7]
        radius: 0.2
    -SPHERE2:
        type: Sphere
        texture: Bleu
        absorb: 0.6
        reflect: 0.4
        refract: 0
        indice: 1.2
        center: [0.6, -0.7, 0.5]
        radius: 0.8
```

Les textures Bleu et Rouge doivent avoir été définies dans la librairie des textures.

3.3.3 Chargement des autres fichiers

Pour la section LOADING, le tiret de liste doit être suivit d'un espace.

Exemple :

LOADING:
- cornel_box_2.sdf
- cornel_box_3.sdf

Il n'est pas obligatoire que les fichiers d'entrées aient l'extension .sdf. Ceci est juste un moyen de ne pas confondre ces fichiers de description avec de simples fichiers texte.

3.3.4 Paramètres de scène

La syntaxe de la section scène est légèrement différente, il n'y a pas de tiret entre la chaîne et la donnée.

Paramètres généraux de rendu

Paramètre	**Descriptif**	**Valeur**	**Valeur par défaut**
resolution	résolution de sortie de la chaîne	liste de deux entiers	[800,600]
supersampling	activation de l'anti-aliasing	booléen	false
nb_photon_MAX	nombre de photons lancés par le photon-mapper	entier	10000
nb_photon_to_find	nombre de photons à trouver autour d'un point lors du calcul de radiance	entier	100
photon_depth	nombre maximum de réflexion/réfraction pour un même photon durant le photon-mapping	entier	20
raytracer_depth	nombre maximum de divisions d'un rayon (en rayons réfracté/réfléchi) issue de la caméra	entier	3

Utilisation des bibliothèques

Paramètre	**Valeur**
camera	une seule caméra
lights	une liste de nom de lumières présentes dans la bibliothèque (décrites dans les fichiers)
objects	une liste de nom d'objets présents dans la bibliothèque (décrits dans les fichiers)

Exemple de scène :

SCENE:
resolution: [600, 600]
supersampling: true
nb_photon_MAX: 40000
nb_photon_to_find: 10
photon_depth: 10
raytracer_depth: 400
camera: La_camera
objects: [SPHERE1, SPHERE2, PLAN1]
lights: [Ponctuelle1]

Les sous-sections **camera**, **objects** et **lights** sont obligatoires.
Objects et **lights** doivent toujours être suivis d'une liste même si la liste est composée d'un élément.

4. Paramètres généraux et utilisation du photon-mapper

Une fois que la scène a correctement été décrite, et que le logiciel ne trouve plus aucune erreur, le photon-mapping et le raytracer peuvent rendre la scène.

Les arguments passés au programme déterminent alors le fonctionnement du photon-mapper

Commande	**Description et valeur de X**	**Exemple**
--out=X	Nom du fichier image de sortie	--out=resultat.tga
--photonmap=X	Indique que l'on souhaite également créer une image X de la carte des photons	--photonmap=map.tga
--display	Active l'affichage automatique des rendus lorsque ceux-ci sont terminés (nécessite l'application ImageMagick)	(commande sans second membre)
libre	Fichier d'entrée au format YAML	cornel_box.sdf

Le chemin du fichier d'entrée est obligatoire, les autres paramètres sont facultatifs.
Par défaut,

- le fichier de sortie est result.tga
- la carte des photons n'est pas rendue
- aucun affichage automatique

Remarque : Les chemins des fichiers d'entrée-sorties peuvent être fournis en absolu ou en relatif. Le fichier d'entrée doit donc être bien placé (donc existant).

5. Exemple complet d'une scène

Une scène écrite en YAML accompagne directement les fichiers sources.

Fichier : cornel_box.sdf

```
LOADING:
        - cornel_box_2.sdf
TEXTURES:
        -Rouge:
                type: Color
                color: [1, 0, 0]
        -Bleu:
                type: Color
                color: [0, 0, 1]
        -Vert:
                type: Color
                color: [0, 1, 0]
        -Violet:
                type: Color
                color: [1, 0, 1]
        -Jaune:
                type: Color
                color: [1, 1, 0]
        -Blanc:
```

```
        type: Color
        color: [1, 1, 1]
    -Orange:
        type: Color
        color: [1, 0.5, 0]
    -Check1:
        type: Checkers
        text_x: [0, 0, 1]
        text_y: [0, 1, 0]
        x_dim: 0.15
        y_dim: 0.15
        even: [0, 0, 0]
        odd: [1, 1, 1]
    -Check2:
        type: Checkers
        text_x: [0, 0, 1]
        text_y: [0, 1, 0]
        x_dim: 0.15
        y_dim: 0.15
        even: [0, 0, 0]
        odd: [1, 1, 1]
    -Polytech:
        type: Bitmap
        text_x: [0, 0, 1]
        text_y: [0, 1, 0]
        x_dim: 2
        y_dim: 2
        file: polytech.tga
OBJECTS:
    -TRIANGLE:
        type: Triangle
        texture: Blanc
        absorb: 0.8
        reflect: 0.2
        transparency: 0
        point1: [-0.6, -1, -1]
        point2: [0, -1, 1]
        point3: [0, 0.8, 0]
        for_volume: false
    -FACE:
        type: Plane
        texture: Polytech
        absorb: 0.95
        reflect: 0.05
        transparency: 0
        point: [1, -1, -1]
        normal: [-1, 0, 0]
    -LEFT:
        type: Plane
        texture: Rouge
        absorb: 0.3
        reflect: 0.7
```

```
            transparency: 0
            point: [0, 0, -1]
            normal: [0, 0, 1]
        -RIGHT:
            type: Plane
            texture: Bleu
            absorb: 0.3
            reflect: 0.7
            transparency: 0
            point: [0, 0, 1]
            normal: [0, 0, 1]
        -UP:
            type: Plane
            texture: Jaune
            absorb: 1
            reflect: 0
            transparency: 0
            point: [0, 1, 0]
            normal: [0, -1, 0]
        -DOWN:
            type: Plane
            texture: Violet
            absorb: 0.95
            reflect: 0.05
            transparency: 0
            point: [0, -1, 0]
            normal: [0, -1, 0]
SCENE:
        resolution: [100, 100]
        supersampling: false
        nb_photon_MAX: 100000
        nb_photon_to_find: 5
        photon_depth: 40
        raytracer_depth: 4
        camera: Ze_camera
        objects: [UP, DOWN, LEFT, RIGHT, FACE, SPHERE1, SPHERE3, PARA1]
        lights: [Radiant_SPHERE, Glob]
```

Fichier : cornel_box_2.sdf

```
OBJECTS:
        -SPHERE1:
            type: Sphere
            texture: Blanc
            absorb: 1
            reflect: 0
            refract: 0
            indice: 1.2
            center: [0, 1, 0]
            radius: 0.25
        -SPHERE2:
            type: Sphere
```

```
            texture: Check2
            absorb: 0.9
            reflect: 0.1
            refract: 0
            indice: 1.2
            center: [1, -1, 1]
            radius: 0.7957
        -SPHERE3:
            type: Sphere
            texture: Blanc
            absorb: 0.2
            reflect: 0
            refract: 0.8
            indice: 1.55
            center: [-0.5, 0, -0.3]
            radius: 0.3
        -PARA1:
            type: Parallelepiped
            texture: Vert
            absorb: 0.9
            reflect: 0.1
            refract: 0
            indice: 1.55
            corner: [0.2, -1, 0]
            x_vector: [0.25, 0, 0.25]
            y_vector: [0.25, 0, -0.25]
            z_vector: [0, 0.75, 0]
CAMERAS:
        -Ze_camera:
            type: Conic
            origine: [-5, 0, 0]
            vector1: [0, 0, -1]
            vector2: [0, 1, 0]
            size1: 1.2
            size2: 1.2
LIGHTS:
        -Ponctu_ONE:
            type: Punctual
            color: [1, 1, 1]
            power: 0.45
            origine: [0.25, 0.65, 0]
        -Ponctu_DEUX:
            type: Punctual
            color: [1, 1, 1]
            power: 0.45
            origine: [-0.25, 0.65, 0]
        -Radiant_PARA:
            type: RadiantVolume
            color: [0.8, 0.7, 1]
            power: 3
            volume: PARA1
        -Radiant_SPHERE:
```

```
    type: RadiantVolume
    color: [1, 1, 1]
    power: 3
    volume: SPHERE1
-Glob:
    type: GlobalLighting
    color: [1,1,1]
    power: 4
```