

# IRT : un ray tracer interactif



par Matthieu Brucher (<http://matthieu-brucher.developpez.com/>) (Blog)

Date de publication : 16/05/2008

Dernière mise à jour :

Comment créer un raytracer ? Comment le rendre utilisable en temps réel pour une application 3D ? Telles sont les questions auxquelles cette série de tutoriels tentera de répondre.

I - Objectifs

II - Fonctionnement d'un raytracer

III - Comment réaliser un tel projet ?

## I - Objectifs

Tout d'abord, un ray tracer est un outil permettant de recréer une scène. On peut s'imaginer cet algorithme comme étant une caméra observant la scène, et pour connaître le contenu de chaque pixel de celle-ci, un rayon est envoyé vers la scène observée. Selon les objets touchés, les rayons se propagent sur d'autres objets ou touchent des lumières et les informations des couleurs sont envoyés à nouveau à la caméra.

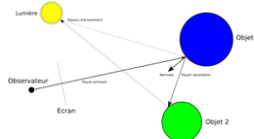
Un ray tracer est somme toute facile à créer. Les formules mathématiques sont relativement simples pour une version donnant des résultats très acceptables. En revanche, il est nettement plus difficile de créer un ray tracer interactif, et c'est ce que je vous propose de réaliser dans cette série de tutoriels.

Interactif ou temps réel signifie que le dessin d'une image doit être rapide, très rapide. Cela signifie aussi qu'il ne sera pas possible d'utiliser toutes les techniques réalistes. En revanche, celles-ci seront présentées au maximum avec leur impact sur les performances.

Après avoir commencé par créer un ray tracer pas du tout réaliste, son optimisation sera tout de suite envisagée. Une fois qu'une image réaliste sera créée, les techniques améliorant ce rendu seront proposées. Enfin, les diverses techniques d'optimisation d'un ray tracer seront exposées.

## II - Fonctionnement d'un raytracer

Dans la vision humaine, les rayons lumineux parcourent le monde à partir de la lumière, frappant les objets puis reflétés ou transmis jusqu'à arriver à l'oeil (ou caméra, appareil photo, ...). On parle parfois de *backwards ray tracing* si une scène était dessinée ainsi (dans certain cas, il désigne le raytracing lui-même). Le principe du ray tracing est de partir de l'arrivée et de remonter à la source, toujours grâce aux principes de la physique optique.



### Fonctionnement du lancer de rayons

Contrairement au fonctionnement réel d'un oeil ou d'une caméra (c'est pourquoi on parle de caméra *virtuelle*), le ray tracing consiste à afficher les rayons passant par l'écran et qui partent vers l'observateur.

Dans le cas présenté ici, un de ces rayons frappe l'objet 1 selon un certain angle par rapport à la normale. L'objet 1 est éclairé par la lumière et donc la couleur du rayon va dépendre de cette lumière, de l'objet et de l'angle entre la normale à l'objet et la lumière.

Le rayon frappant l'objet est aussi reflété et touche l'objet 2 selon un autre angle. Tout comme l'objet 1, il est éclairé par la lumière angle, ce qui donne une certaine couleur au rayon reflété, fonction de la lumière et de l'angle d'incidence. La couleur de ce rayon reflété est mélangé à la couleur du rayon frappant l'objet 1 et ce mélange de couleurs est la couleur qui sera donné au pixel sur l'écran.

Le mélange des couleurs est fonction de plusieurs paramètres :

- la couleur de l'objet
- la couleur de la lumière
- la matière de l'objet (un objet mat ou brillant donnera une couleur due à la lumière différente)
- la réflectivité de l'objet (quelle est la quantité de lumière reflétée par l'objet ?)
- la transmissibilité de l'objet (quelle est la quantité de lumière qui est transmise dans l'objet ?)

Un ray tracer simple (primaire) ne va tenir compte que de la couleur de l'objet, puis en ajoutant la lumière ambiante (les rayons provenant des différentes lumières de la scène qui frappent l'objet), un premier résultat réaliste se profile (avec des zones d'ombre qui peuvent apparaître). En ajoutant les rayons reflétés (parfois appelés rayons secondaires), le résultat devient très réaliste (dans le cas proposé, la lumière sur l'objet 2 pourrait se refléter sur un objet 3 puis 4, ... le nombre de reflets est complètement paramétrisable). D'autres techniques peuvent être utilisées pour améliorer ce réalisme, ce qui sera expliqué dans cette série de tutoriels.

### III - Comment réaliser un tel projet ?

Ce projet est réalisé en C++ et en Python. Le C++ sera utilisé pour la partie critique au niveau du temps d'exécution. Python sera utilisé pour réaliser des tests, des mesures et pour afficher des images. La raison de ce choix est qu'il est plus simple de créer une scène avec Python qu'en C++. De plus, cela sera une très bonne application de **ce tutoriel**.

Le code est proposé en licence LGPL sur [launchpad.net](http://launchpad.net). A chaque tutoriel correspondra une révision qui sera donnée à la fin de celui-ci.

Bonne lecture.

