



# miniRT

Mon premier RayTracer avec la minilibX

*Résumé: Ce projet est une introduction au monde incroyable du Raytracing.  
Une fois terminé, vous pourrez générer des images simples et vous ne serez plus jamais effrayé d'implémenter des formules mathématiques.*

# Table des matières

I	Introduction	2
II	Règles communes	3
III	Partie obligatoire	4
IV	Bonus part	9
V	Exemples	11

# Chapitre I

## Introduction

Afin de générer des images en 3 dimensions sur un ordinateur, deux approches existent : La **Rasterization**, qui est utilisée par la quasi totalité des moteurs graphiques pour son efficacité, et le **RayTracing**.

Le raytracing est apparu pour la première fois en 1968 (amélioré depuis) et est bien plus cher pour l'ordinateur que la **razterization**, et par conséquent n'est pas aussi bien adapté à la modélisation en temps réel. Cependant, il produit des images ayant un degré de réalisme plus élevé.



FIGURE I.1 – Ces images sont modélisées avec la technique de raytracing. Impressionnant non ?

Avant que vous n'arriviez à créer des graphiques d'une telle qualité, vous devez maîtriser les bases : le **miniRT** est un projet en C, humble mais **fonctionnel**.

L'objectif principal est de vous prouver que vous pouvez implémenter n'importe quelle formule mathématique ou physique sans pour autant être un mathématicien.

Nous allons nous contenter d'implémenter uniquement les formules mathématiques les plus simples.

# Chapitre II

## Règles communes

- Votre projet doit être codé à la Norme. Si vous avez des fichiers ou fonctions bonus, celles-ci seront incluses dans la vérification de la norme et vous aurez 0 au projet en cas de faute de norme.
- Vos fonctions doivent pas s'arrêter de manière inattendue (segmentation fault, bus error, double free, etc) mis à part dans le cas d'un comportement indéfini. Si cela arrive, votre projet sera considéré non fonctionnel et vous aurez 0 au projet.
- Toute mémoire allouée sur la heap doit être libéré lorsque c'est nécessaire. Aucun leak ne sera toléré.
- Si le projet le demande, vous devez rendre un Makefile qui compilera vos sources pour créer la sortie demandée, en utilisant les flags `-Wall`, `-Wextra` et `-Werror`. Votre Makefile ne doit pas relink.
- Si le projet demande un Makefile, votre Makefile doit au minimum contenir les règles `$(NAME)`, `all`, `clean`, `fclean` et `re`.
- Pour rendre des bonus, vous devez inclure une règle **bonus** à votre Makefile qui ajoutera les divers headers, librairies ou fonctions qui ne sont pas autorisées dans la partie principale du projet. Les bonus doivent être dans une fichier `_bonus.{c/h}`. L'évaluation de la partie obligatoire et de la partie bonus sont faites séparément.
- Si le projet autorise votre `libft`, vous devez copier ses sources et son Makefile associé dans un dossier `libft` contenu à la racine. Le Makefile de votre projet doit compiler la librairie à l'aide de son Makefile, puis compiler le projet.
- Nous vous recommandons de créer des programmes de test pour votre projet, bien que ce travail **ne sera pas rendu ni noté**. Cela vous donnera une chance de tester facilement votre travail ainsi que celui de vos pairs.
- Vous devez rendre votre travail sur le git qui vous est assigné. Seul le travail déposé sur git sera évalué. Si Deepthought doit corriger votre travail, cela sera fait à la fin des peer-evaluations. Si une erreur se produit pendant l'évaluation Deepthought, celle-ci s'arrête.

# Chapitre III

## Partie obligatoire

Nom du programme	miniRT
Fichiers de rendu	Tous les fichiers nécessaires
Makefile	all, clean, fclean, re, bonus
Arguments	Une scène dans le format *.rt
Fonctions externes autorisées	<ul style="list-style-type: none"><li>• open, read, write, malloc, free, perror, strerror, exit</li><li>• Les fonctions de la lib maths (-lm man man 3 math)</li><li>• Les fonctions de la MinilibX</li></ul>
Libft autorisée	Yes
Description	L'objectif de votre programme est de modéliser des images en respectant le protocole de raytracing. Ces images doivent représenter une scène, vue d'un angle défini et contenant des objets géométriques simples, chacun avec son système d'éclairage.

Les contraintes sont les suivantes :

- Votre projet **doit** utiliser la **minilibX**. Soit la version disponible sur votre OS, ou depuis les sources. Si vous travaillez depuis les sources, vous devez appliquer les mêmes règles que pour la libft.
- La gestion des fenêtres doit être propre : pas de soucis lorsqu'on passe une autre fenêtre dessus, lorsqu'on la minimise, etc.
- Vous devez implémenter au moins 5 objets géométriques simples : plan, sphère, cylindre, carré et triangle.
- Si applicable, toutes les intersections possibles et l'intérieur de l'objet doivent être gérés correctement.

- Votre programme doit être capable de changer la taille de la propriété unique d'un objet : diamètre pour une sphère, taille du côté d'un carré, hauteur et longueur d'un cylindre.
- Votre programme doit pouvoir appliquer des transformations aux objets, lumières et caméra : translation et rotation (excepté pour les sphères, triangles et lumières qui ne peuvent pas être tournées).
- Gestion de la lumière : luminosité, ombres, lumière d'ambiance (les objets ne sont jamais complètement dans le noir). Les lumières colorées les multi-spots doivent être gérés correctement.
- Au cas où Deepthought aurait des yeux un jour pour évaluer votre projet, et afin de faire de superbes fond d'écrans...  
Votre programme doit pouvoir sauver l'image modélisée au format `bmp` quand le second argument est "`-save`".
- Si il n'y a pas de second argument, le programme doit afficher l'image dans une fenêtre en respectant les règles suivantes :
  - Appuyer sur la touche `ESC` doit fermer la fenêtre et quitter le programme proprement.
  - Cliquer sur la croix rouge de la fenêtre doit fermer la fenêtre et quitter le programme correctement.
  - Si la taille déclarée dans la scène est plus grande que le display, la taille de la fenêtre doit être celle du display actuel.
  - Si il y a plus d'une caméra, vous devez pouvoir alterner entre les caméras à l'aide d'un keybind de votre choix.
  - L'utilisation d'`images` de la `minilibX` est fortement recommandé.
- Votre programme doit prendre en premier argument une description de scène avec un fichier `.rt`.
  - Ce fichier contiendra la taille de l'image/de la fenêtre, ce qui implique que votre `miniRT` doit pouvoir utiliser n'importe quelle taille  $> 0$ .
  - Chaque type d'élément est séparé par une ou plusieurs ligne(s) vide(s).
  - Les éléments peuvent être mis dans n'importe quel ordre dans le fichier.
  - Les éléments qui commencent par une lettre majuscule ne peuvent être déclarés qu'une seule fois dans la scène.

- La première information de chaque élément est son type (un ou deux caractère(s)), suivi de toutes les informations spécifiques de l'élément.

- Resolution :

```
R 1920 1080
```

- identifiant : **R**
  - x render size
  - y render size

- Lumière ambiante :

```
A 0.2 255,255,255
```

- identifiant : **A**
  - Lumière ambiante - ratio dans le range [0.0,1.0] : **0.2**
  - Couleurs R,G,B dans le range [0-255] : **255, 255, 255**

- Camera :

```
c -50.0,0,20 0,0,1 70
```

- identifiant : **c**
  - coordonnées x,y,z du point de vue : **0.0,0.0,20.6**
  - Vecteur d'orientation 3d dans le range [-1,1] pour chaque axe x,y,z **0.0,0.0,1.0**
  - FOV : Champ de vision horizontal en degrés dans le range [0,180]

- Lumière :

```
l -40.0,50.0,0.0 0.6 10,0,255
```

- identifiant : **l**
  - coordonnées x,y,z du point Lumière : **0.0,0.0,20.6**
  - ratio de luminosité dans le range [0.0,1.0] : **0.6**
  - Couleurs R,G,B dans le range [0-255] : **10, 0, 255**

- Sphere :

```
sp 0.0,0.0,20.6 12.6 10,0,255
```

- identifiant : **sp**
  - coordonnées x,y,z du point sphere : **0.0,0.0,20.6**
  - the sphere diameter : **12.6**
  - Couleurs R,G,B dans le range [0-255] : **10, 0, 255**

— Plane :

```
pl  0.0,0.0,-10.0  0.0,1.0,0.0  0,0,225
```

— identifiant : **pl**

— coordonnées x,y,z du point f0.0,0.0

— Vecteur d'orientation 3d dans le range [-1,1] pour chaque axe x,y,z :  
**0.0,0.0,1.0**

— Couleurs R,G,B dans le range [0-255] : **0, 0, 255**

— Carré :

```
sq  0.0,0.0,20.6   1.0,0.0,0.0  12.6  255,0,255
```

— identifiant : **sq**

— coordonnées du point x,y,z : **0.0,0.0,20.6**

— Vecteur d'orientation 3d dans le range [-1,1] pour chaque axe x,y,z :  
**1.0,0.0,0.0**

— Hauteur : **12.6**

— Couleurs R,G,B dans le range [0-255] : **255, 0, 255**

— Cylindre :

```
cy  50.0,0.0,20.6   0.0,0.0,1.0  10,0,255   14.2  21.42
```

— identifiant : **cy**

— coordonnées x,y,z du point f50.0,.6

— Vecteur d'orientation 3d dans le range [-1,1] pour chaque axe x,y,z :  
**0.0,0.0,1.0**

— diamètre du cylindre : **14.2**

— hauteur du cylindre : **21.42**

— Couleurs R,G,B dans le range [0,255] : **10, 0, 255**

— Triangle :

```
tr  10.0,20.0,10.0  10.0,10.0,20.0  20.0,10.0,10.0  0,0,255
```

— identifiant : **tr**

— coordonnées x,y,z du premier point **10.0,20.0,10.0**

— coordonnées x,y,z du second point **10.0,10.0,20.0**

— coordonnées x,y,z du troisième point **20.0,10.0,10.0**

— Couleurs R,G,B dans le range [0,255] : **0, 255, 255**

- exemple minimaliste de fichier .rt :

```
R 1920 1080
A 0.2
c -50,0,20    0,0,0    70
l -40,0,30      0.7
pl 0,0,0        0,1,0,0
sp 0,0,20
sq 0,100,40    0,0,1,0    30
cy 50.0,0.0,20.6 0,0,1.0   14.2  21.42  10,0,255
tr 10,20,10     10,10,20   20,10,10  0,0,255
```

- Si vous rencontrez un quelconque problème de configuration dans le fichier, votre programme doit se fermer correctement et renvoyer 'Error n" suivi d'un message explicite de votre choix.
- Pour la soutenance, il est idéal de fournir son propre set de scènes avec un focus sur chacun des éléments afin de la faciliter.

# Chapitre IV

## Bonus part

Évidemment, la technique RT gère bien plus de choses, telles que la réflexion, la transparence, la réfraction, ainsi que des objets plus complexes, les ombres légères, la luminosité globale...

Mais pour ce `miniRT`, nous souhaitons rester simples.

Voici donc une liste de bonus simples à implémenter. Si vous souhaitez en faire plus, nous vous recommandons de coder votre propre RT plus tard dans votre vie de développeur, bien entendu après que celui-ci soit terminé et rendu.



FIGURE IV.1 – a spot, a space skybox and a shiny earth-textured sphere with bump-mapping



Les bonus ne seront évalués que si la partie obligatoire est PARFAITE. Complète, sans faille, même lors de mauvais usage ou autre. Si vous n'avez pas TOUS les points de la partie obligatoire, votre partie bonus sera entièrement ignorée.

Liste de bonus :

- **Normal disruption** : par exemple utiliser `sine` pour avoir un effet de vague.
- **Color disruption** : par exemple, un plateau d'échecs
- **Color disruption** : un effet arc-en-ciel.
- Un capuchon sur les cylindres
- Lumière parrallèle : la lumière suit une direction précise
- Élément composé : Cube (6 carrés)
- Élément composé : Pyramide (4 triangles, un carré)
- Un autre objet de second degré : Cone, hyperbole, paraboloïde...
- Filtre de couleurs : Sepia, filtre R/G/B...
- Anti-aliasing
- Stéréoscopie simple (comme des lunettes vert/rouge)
- Rendu multithreadé
- Texture sur les sphères (uv mapping)
- Une superbe skybox
- Gérer les reliefs de texture (bump map)
- Interactions clavier (translation/rotation) pour la caméra
- Interactions clavier (translation/rotation) pour les objets
- Faire tourner la caméra à l'aide de la souris



Vous êtes autorisés à utiliser d'autres fonctions pour compléter la partie bonus tant que leur utilisation est justifiée. Vous pouvez également modifier le fichier de scène et ses règles pour les bonus.  
Soyez malins !



Vous n'avez besoin de valider que 14 bonus pour valider tous les points, donc choisissez bien et ne perdez pas de temps

# Chapitre V

## Exemples

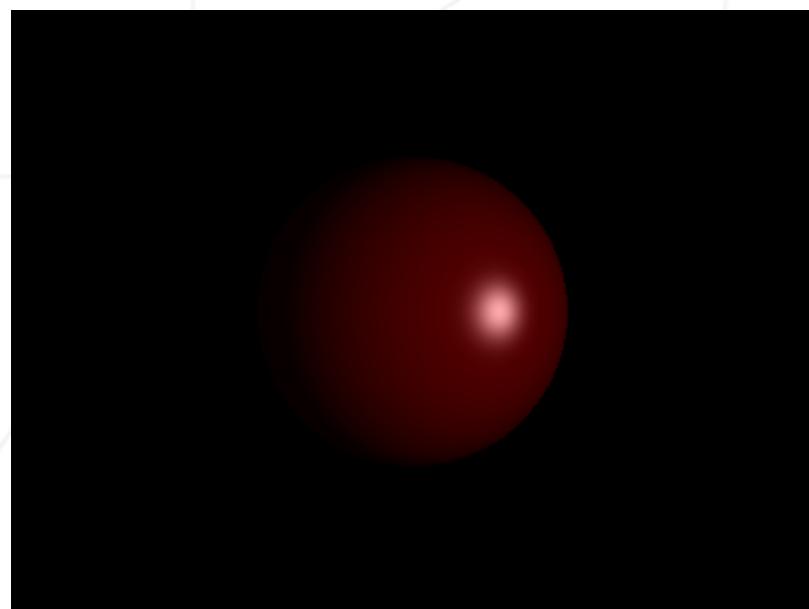


FIGURE V.1 – Une sphère, un spot, un reflet (optionnel)



FIGURE V.2 – A cylindre, un spot

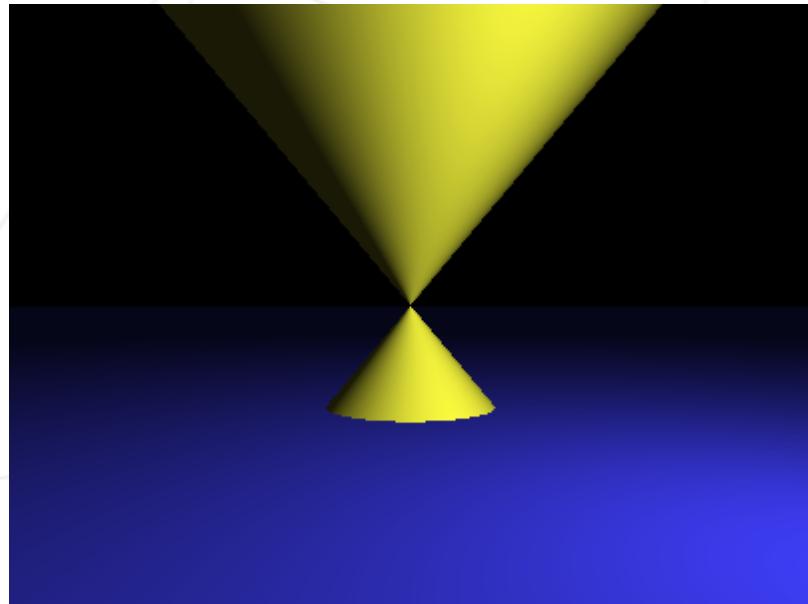


FIGURE V.3 – Un cone (optionnel), un plan, un spot

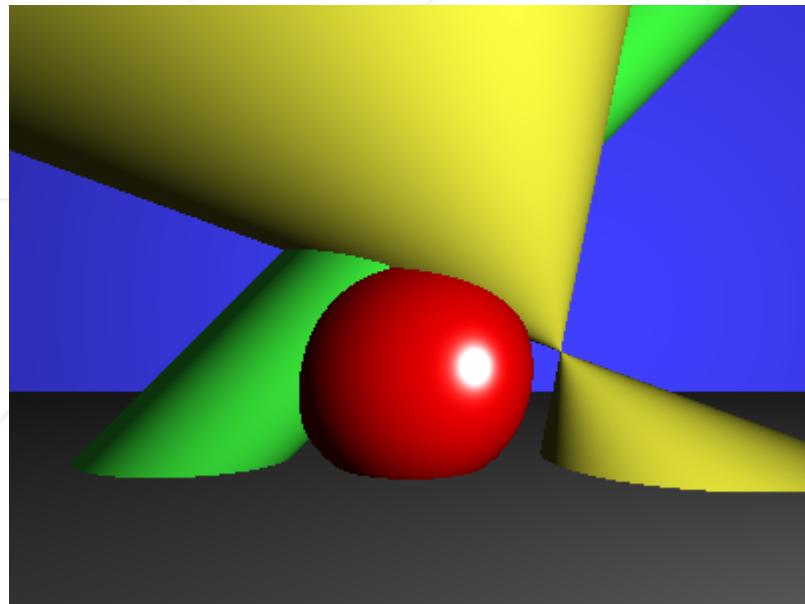


FIGURE V.4 – Un peu de tout, incluant 2 plans

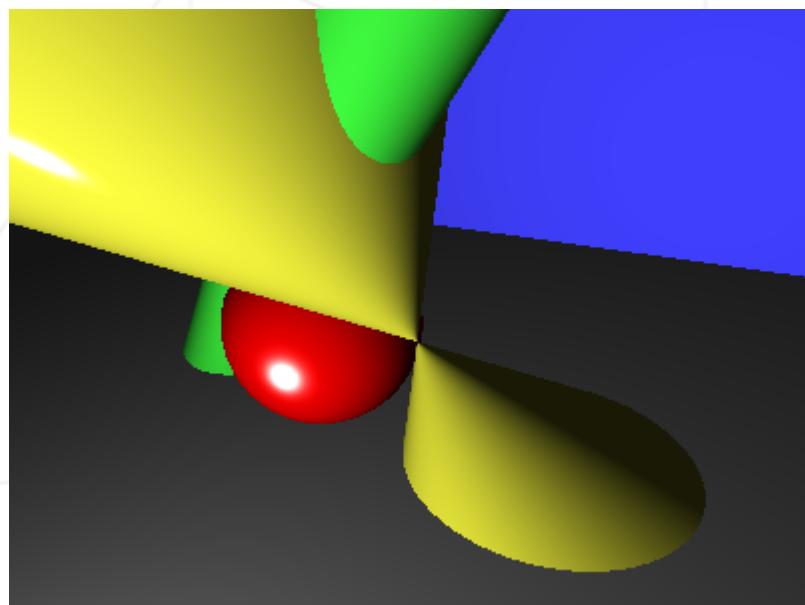


FIGURE V.5 – La même scène avec une caméra différente

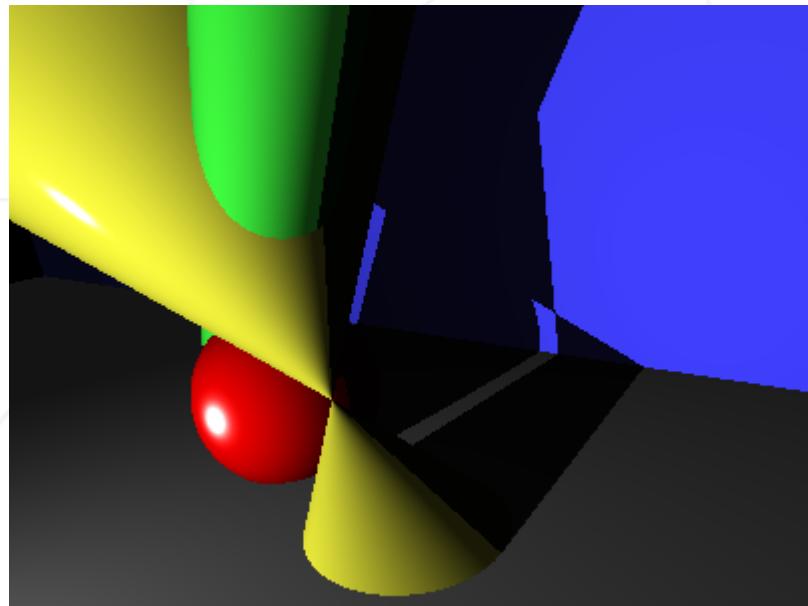


FIGURE V.6 – Les ombres en plus

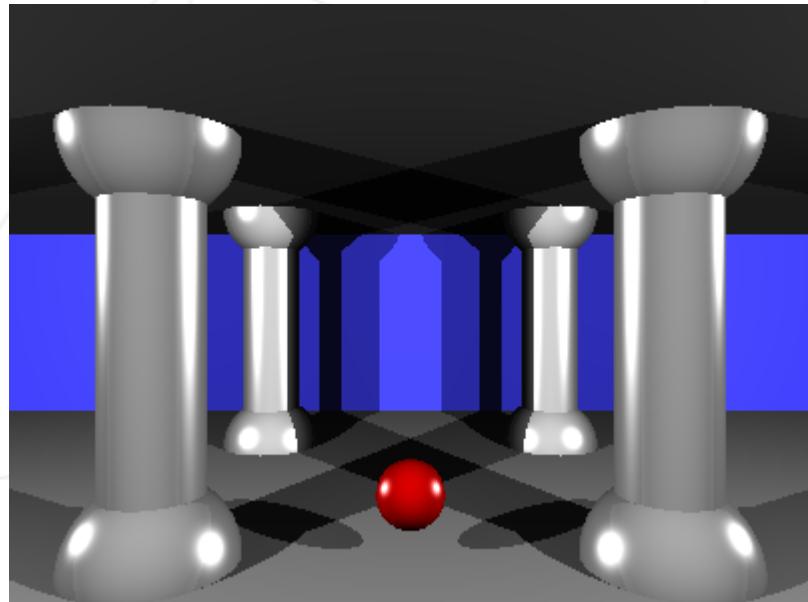


FIGURE V.7 – Plusieurs spots

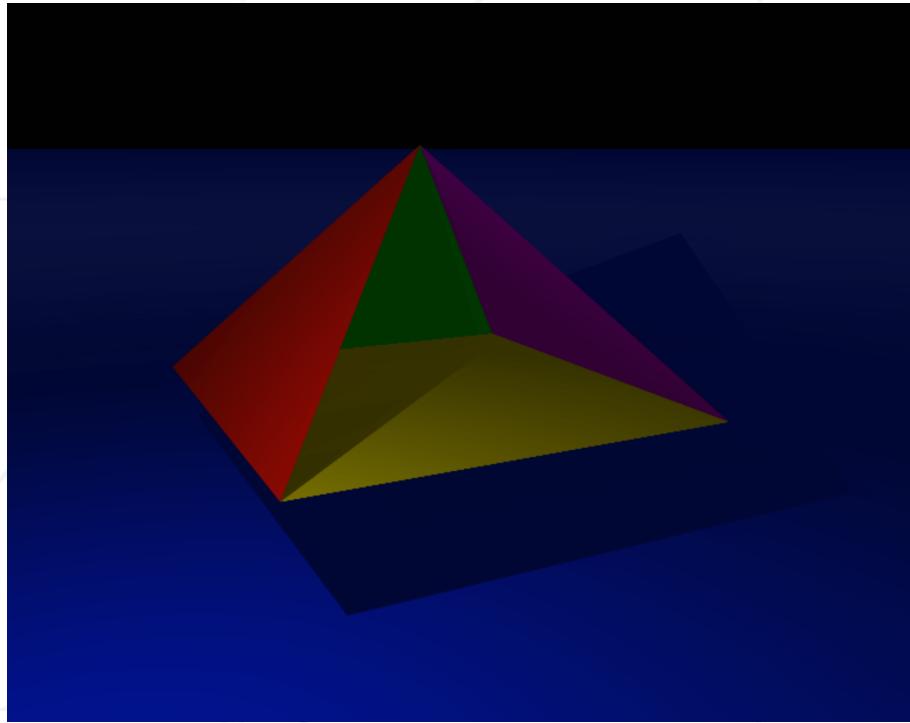


FIGURE V.8 – un plan, 3 triangles, 1 carré, ratio de luminosité

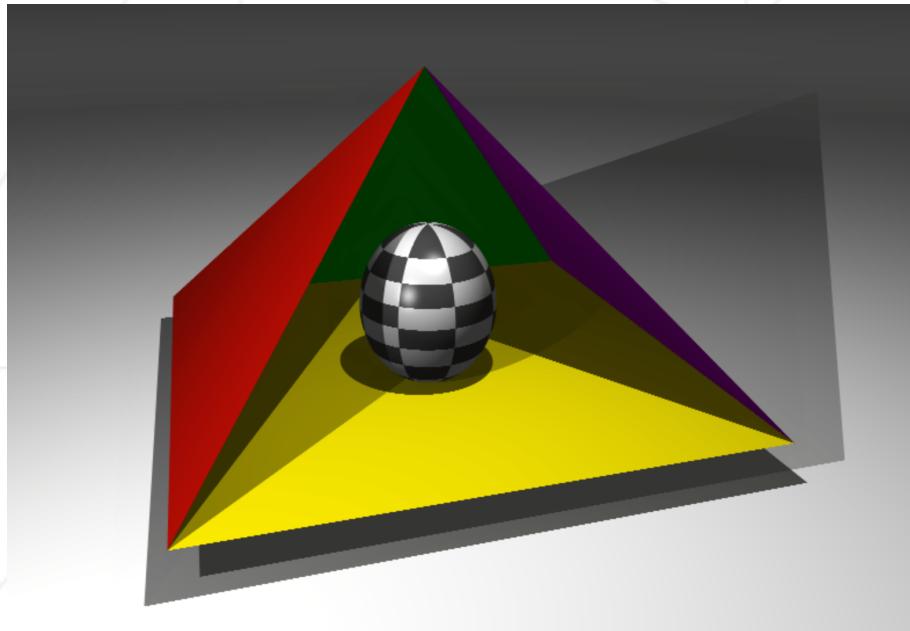


FIGURE V.9 – Et enfin, plusieurs spots et une sphère quadrillée réfléchissante au milieu (optionnel)