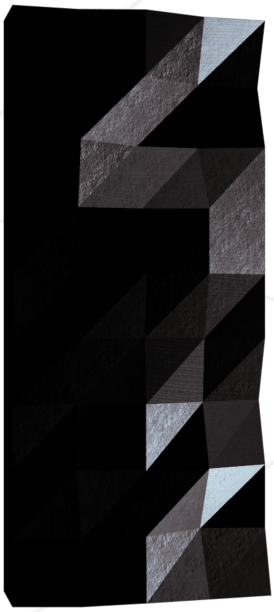

Mini-projet d'Infographie

RTv1

42 staff staff@42.fr

Résumé: *Ce mini-projet est une première étape dans la réalisation d'un programme de Raytracing, en vue de calculer entièrement des images de synthèse.*

Table des matières

Chapitre I

Préambule

"Il semble qu'en Europe, la recherche soit d'un bon niveau. Mais les faiblesses viennent des applications concrètes, une étape qui est en elle même une source d'innovation. Cela je pense vient d'un manque de sociétés prêtes à risquer pour entreprendre. Ce que nous appelons le développement c'est rarement les grandes entreprises qui le font, c'est plus les petites ou les moyennes entreprises. Alors ce qu'il faut c'est beaucoup de petites entreprises, avec des étudiants doués, des capitaux à risque, plus efficaces entre les mains du secteur privé, et aussi des champions que l'on prenne pour modèle, en disant 'l'innovation c'est ça'*. Mais il y a quelque chose de plus subtil : c'est le facteur culturel. En Europe, l'échec c'est très grave. Si en sortant de l'université vous loupez votre coup, cela vous suit toute votre vie. Alors qu'en Amérique, à Silicon Valley, on passe son temps à échouer. Quand on se casse la figure, on se relève et on recommence. Ce qu'il faut pour que l'industrie informatique se développe en Europe et en France, c'est une solide industrie du logiciel. Parce que le logiciel c'est le pétrole des années 80 et 90, de cette révolution informatique. Il faut des centaines de mini entreprises du logiciel, et la France pourrait dominer l'Europe dans le logiciel. Elle a les étudiants les plus brillants, une bonne maîtrise de la technologie, ce que nous devons faire c'est encourager les jeunes à créer des sociétés de logiciel. Nous, nous ne voulons pas mettre la main dessus, le gouvernement ne doit pas non plus tenter de le faire. Elles doivent appartenir à ceux qui prennent des risques."*

Steve Jobs, 1984, interview accordée à Antenne 2

Chapitre II

Sujet

II.1 Suite du préambule

Pour une fois, le sujet est en rapport avec le préambule. Pour réussir, vous devez échouer. C'est pour cela que vous allez commencer par une sorte de proof-of-concept du principe du Raytracing. C'est l'objet du RTv1 : vous familiariser avec le lancer de rayon, les élements géometriques à manipuler, la description de la scene... . Vous allez donc réussir, ou échouer, ce projet, dans le but de le faire par la suite plus gros, plus grand, plus bo, avec tout plein d'options (ça sera le RT tout court - ne commencez donc pas le RT sans avoir fait le RTv1, c'est pas raisonnable).

Voyez la vidéo avec la démo pour comprendre ce que l'on a au départ, et ce que doit faire votre programme. Les ressources sur le net sont plutôt importantes en matière d'explication du Raytracing. Les approches sont parfois variées, trouvez celle qui vous convient. Le RTv1 reste une version simple, voyez ce qui est demandé pour ne pas vous perdre dans des méandres des très nombreuses fonctionnalités que peut contenir un tel programme et qu'il vous faut de toute façon conserver pour le RT.

II.2 Le RTv1

Votre objectif est donc d'être capable, à l'aide de votre programme, de générer des images de synthèse selon la méthode du Ray-Tracing.

Ces images de synthèse représentent chacune une scène, vue d'une position et d'un angle spécifiques, définie par des objets géométriques simples, et disposant d'un système d'éclairage.

Les éléments à réaliser sont les suivants :

- Coder en C à la norme
- Avoir un Makefile normal (tout ce que vous avez l'habitude d'y mettre)
- Implémenter la méthode du lancer de rayon (le raytracing quoi..) pour obtenir une image de synthèse
- Avoir au moins 4 objets géométriques simples comme objets de base (non composés) : plan, shpère, cylindre et cône

- Gestion du réaffichage sans re-calcul (en gros avec la MinilibX on gère l'expose correctement) : si une partie de la fenêtre doit etre redessinée à l'écran, c'est mieux si il n'y a pas tous les calculs à refaire..
- Position et direction quelconque du point de vision, et des objets simples
- Gestion à minima de la lumière : luminosité, ombres

Les bonus possibles :

- Multi-spot
- Brillance

Aucun autre bonus ne sera accepté, cela sera une des options du projet suivant : le RT.

Pour la soutenance, il serait souhaitable que vous ayez un jeu de scènes mettant en avant ce qui est fonctionnel, facilitant ainsi le contrôle des éléments à réaliser. Par exemple :

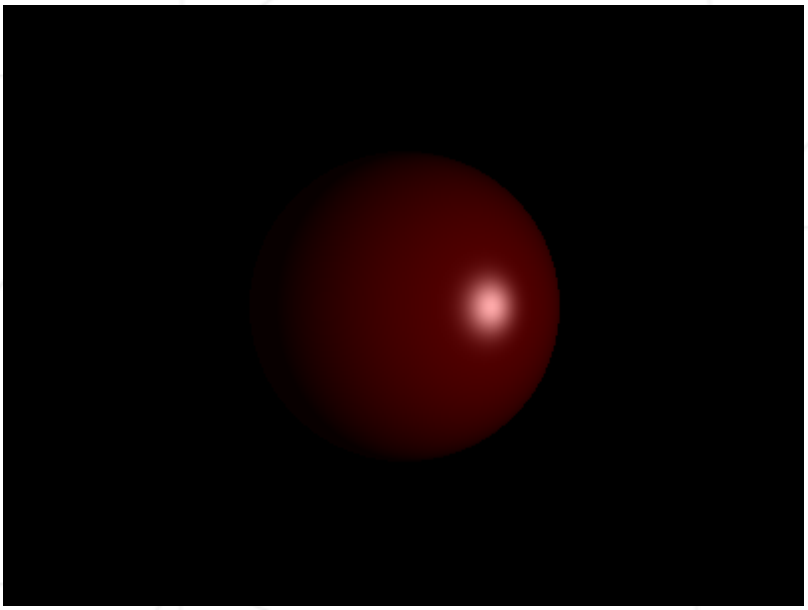

FIGURE II.1 – Une sphère, un spot, de la brillance (option)

FIGURE II.2 – Un cylindre, un spot

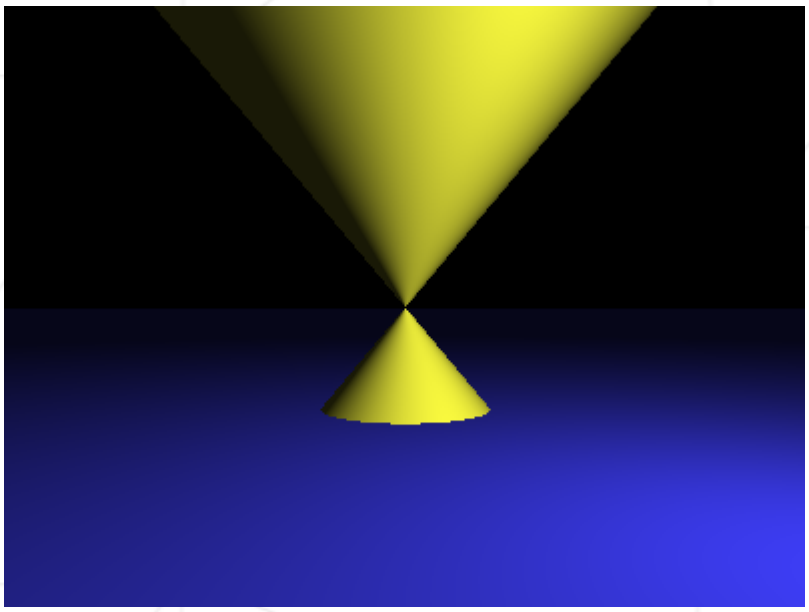

FIGURE II.3 – Un cone, un plan, un spot

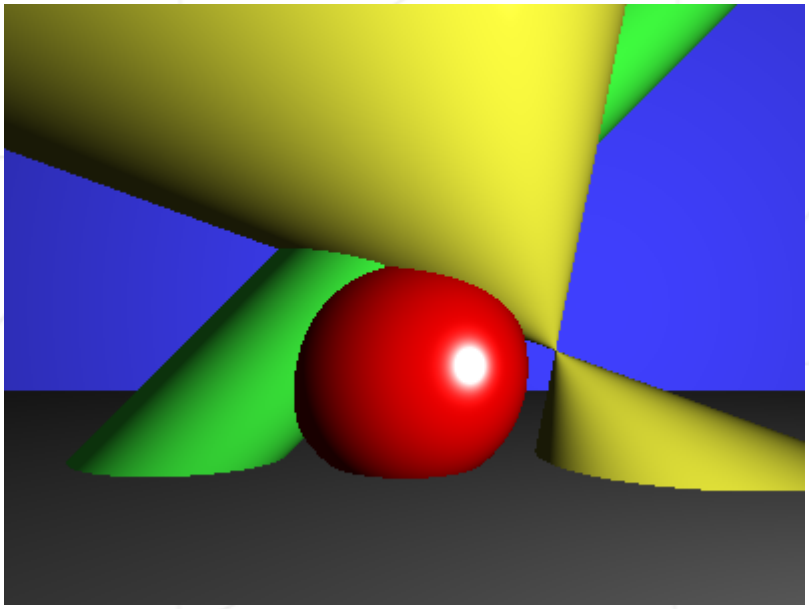

FIGURE II.4 – Un peu de tout, dont 2 plans

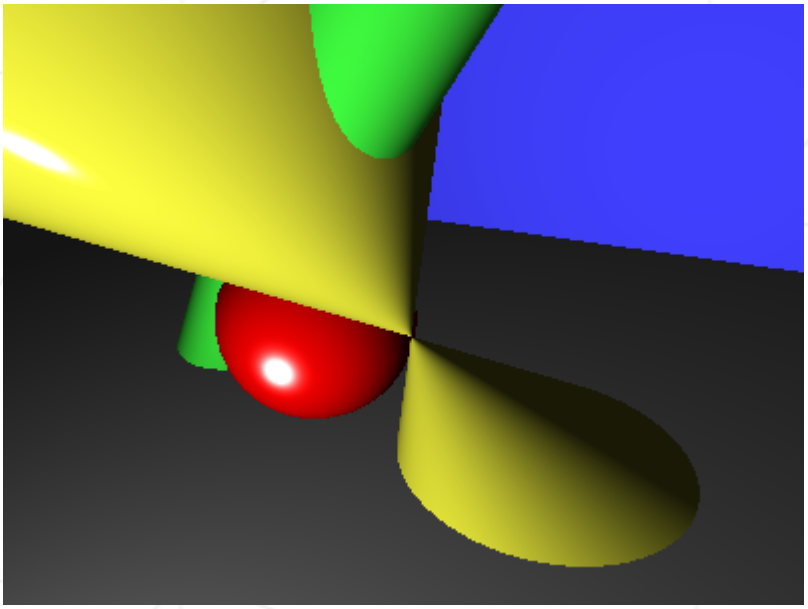

FIGURE II.5 – Même scene, différent point de vue

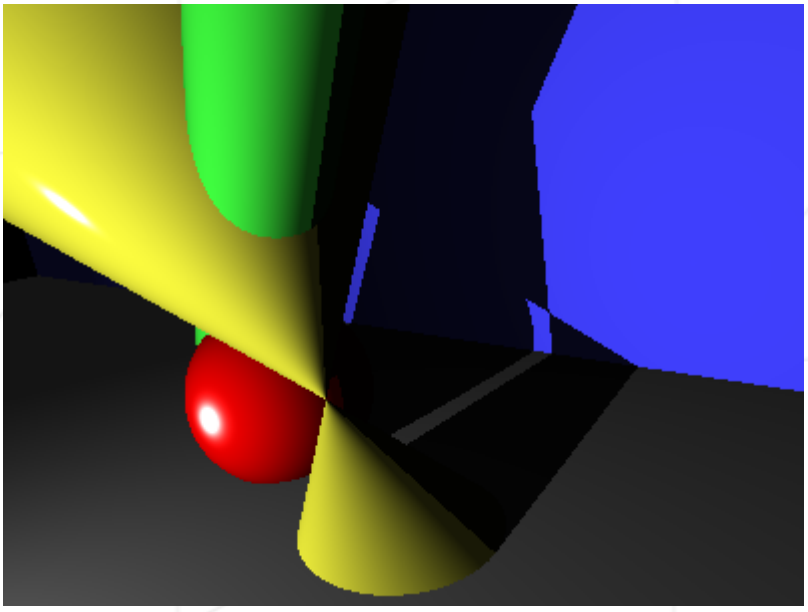

FIGURE II.6 – Cette fois-ci avec des ombres

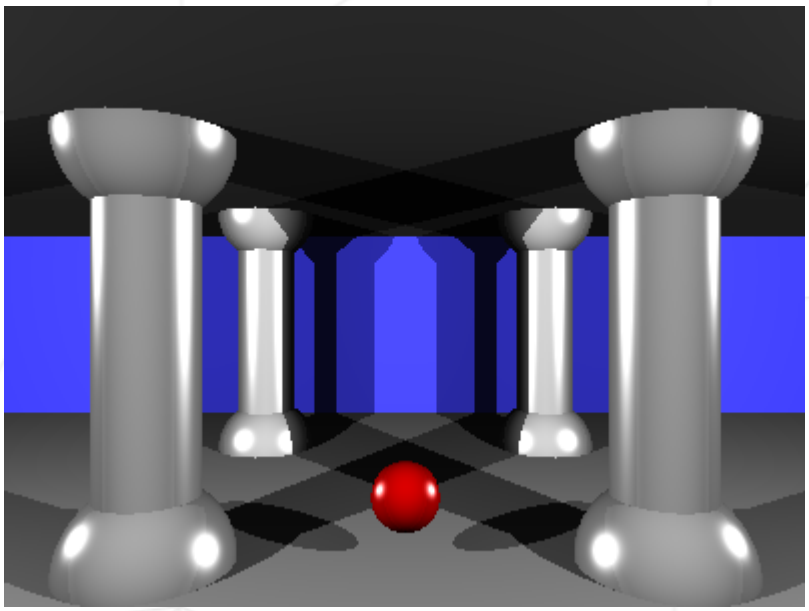

FIGURE II.7 – Et enfin avec plusieurs spots

Chapitre III

Détails

Rendu classique sur le git, Makefile classique, et bien sûr, seul le contenu de votre dépot sera utilisé en soutenance.

Vous devez utiliser en priorité les fonctions qui doivent être présentes (et qui normalement le sont) dans votre libft, en lieu et place de celles de la libc. Vous ne pouvez pas utiliser des librairies non présentes par défaut sur le dump de l'école, excepté votre libft et la minilibX native MacOS que vous rajouterez alors a votre rendu selon les directives de compilation habituelles. Un conseil : restez simple, libft, libm et minilibX suffisent.

Vous pouvez utiliser les types float et/ou double et/ou long double du C.

Bon projet !