DM n° 3 — Plaquage de textures en lancer de rayons

Antoine de Roquemaurel (G1.1)

1 Avant-propos

L'archive que vous avez obtenu sur Moodle contient plusieurs fichiers organisés comme suit :

`src/` Contient uniquement les fichiers modifiés, c'est-à-dire `main.cpp`, `texturefetch.cpp` et `color.hpp`.

`report_AntoinedeRoquemaurel.pdf` Le présent rapport que vous êtes en train de lire

Comme vous avez pu le voir, j'ai choisi de modifier la classe `Color`. Ces modifications n'étaient pas indispensables, mais permettait d'améliorer la lecture du code. Ainsi j'ai ajouté 3 méthodes :

- Opérateur `+=`, permettant d'éviter de faire `color = color + other`, le `+=` est plus rapide et facile à lire.
- De la même manière, l'opérateur `*=`.
- Méthode `reset()` qui remet toutes les composantes à `0.f`.

2 Opérateur `getTextel()`

Une fois les textures intégrés, on peut observer un effet « d'escalier » sur le quadrillage de l'image, principalement au premier plan, comme le montre la figure 4.

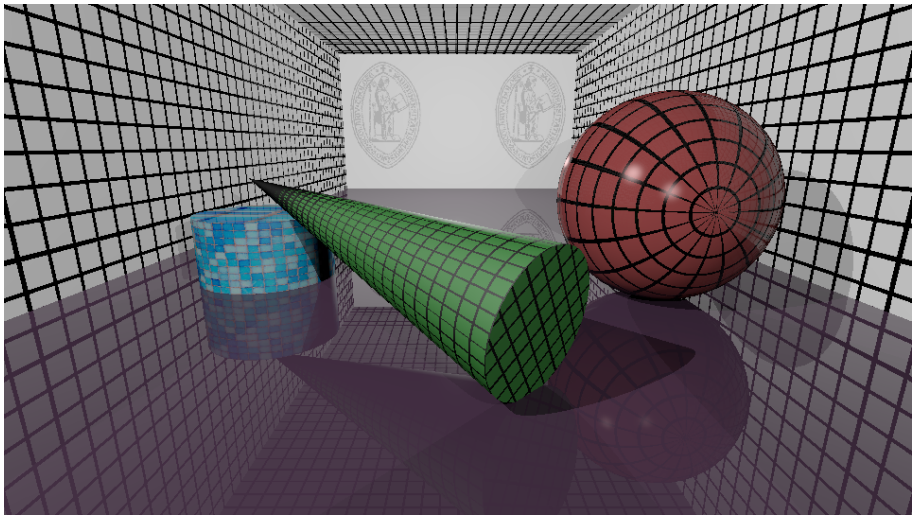

Figure 1 – Image obtenue après le développement de `getTextel()`

Cela vient du fait que nous essayons d'intégrer une texture 2D sur des figures 3D. Afin de bien représenter une profondeur sur nos textures 2D, nous les avons déformés afin que sa taille en premier plan soit plus grande que la taille en arrière plan. C'est cette transformation qui créer un effet de *crênelage* sur le devant de la scène.

3 Opérateur d'interpolation linéaire pour l'accès à un élément de texture

Afin d'améliorer la qualité de notre image, nous avons effectué une interpolation linéaire des pixels voisins. Le but de ce calcul est de prendre en compte les voisins pour afficher notre pixel, ainsi cela va créer un effet « d'adoucissement » dans les zones à fort contraste : le quadrillage par exemple.

C'est ainsi que notre quadrillage va maintenant avoir un effet de « flou » cachant le crénelage précédent.

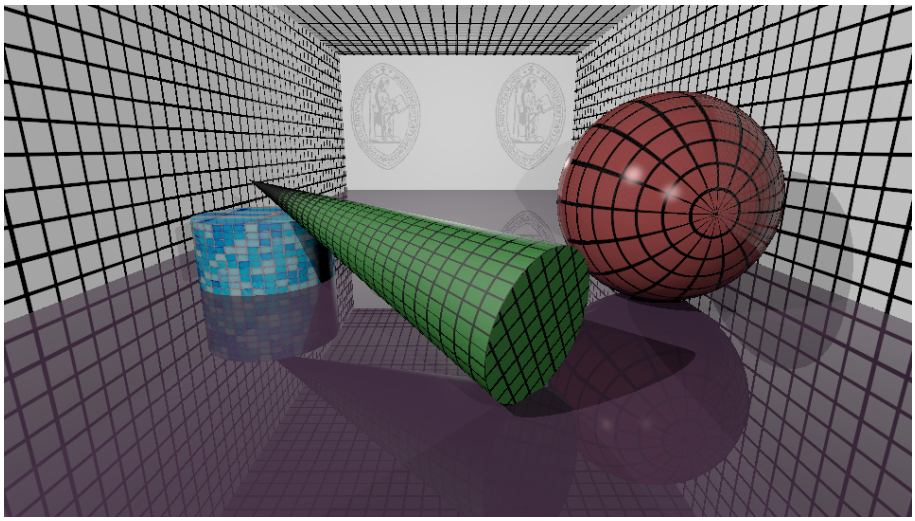

Figure 2 – Image obtenue après le développement de `interpolateTexture()`

4 Opérateur d'intégration pour le filtrage de texture

Une fois cette fonction développée, le temps d'exécution à grimpé en flèche comme le montre le listing 1.

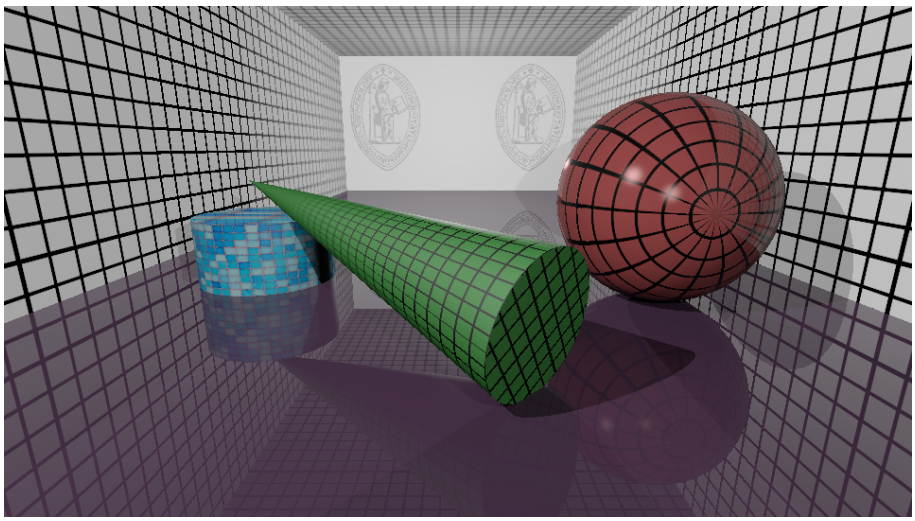

Figure 3 – Image obtenue après le développement de `integrateTexture()`

```
1 aroquemaurel@Luffy ~/projets/cpp/raytracing/bin <master> time ./main
```

```
./main  13,19s user 0,04s system 98% cpu 13,459 total
```

Listing 1 – Temps d'execution avec `interpolateTexture`

5 Opérateur de pré-filtrage de texture

Après le développement de cette fonciton, le temps d'exécution a été divisé par plus de 4 !

```
aroquemaurel@Luffy ~/projets/cpp/raytracing/bin <master> time ./main
./main  3,09s user 0,03s system 94% cpu 3,300 total
```

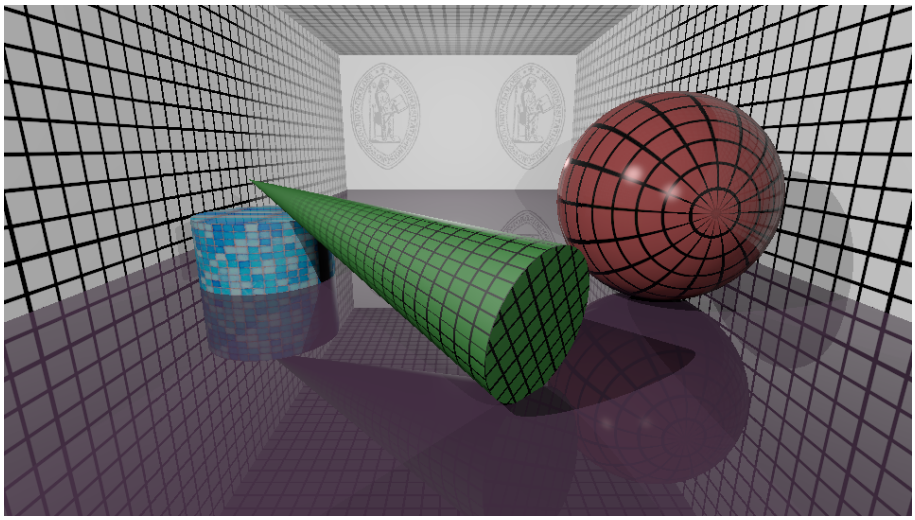

Figure 4 – Image obtenue après le développement de `prefilterTexture()`

6 Analyse et estimation des limites des techniques de filtrage implantées