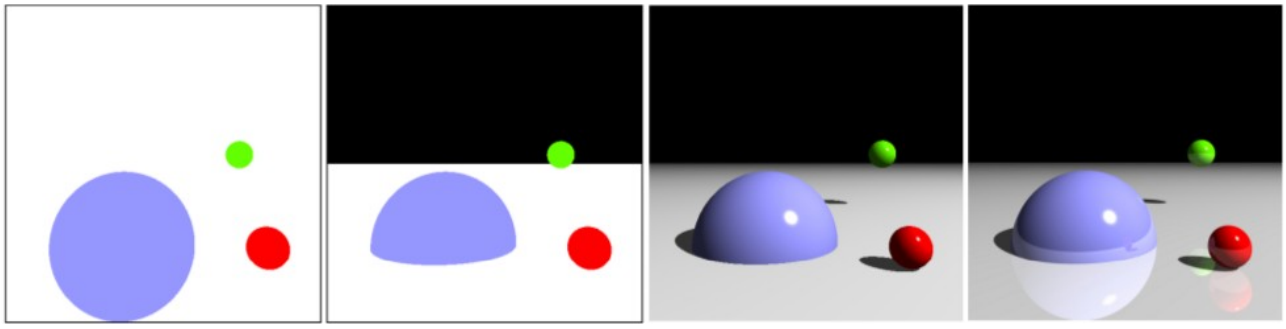
Emery Guillaume
Ziza Simon

21/10/2016

Ray Tracing



Intersection avec les primitives 3D

Intersection avec un plan

Trouver l'intersection entre une droite et un plan revient à résoudre le système d'équation suivant :

$$\begin{cases} < x - x_p , n_p > = 0 \\ x = x_s + t\,u \end{cases}$$

Avec x_p un point quelconque du plan, n_p la normale au plan, x_s un point de la droite et u vecteur directeur de celle-ci.

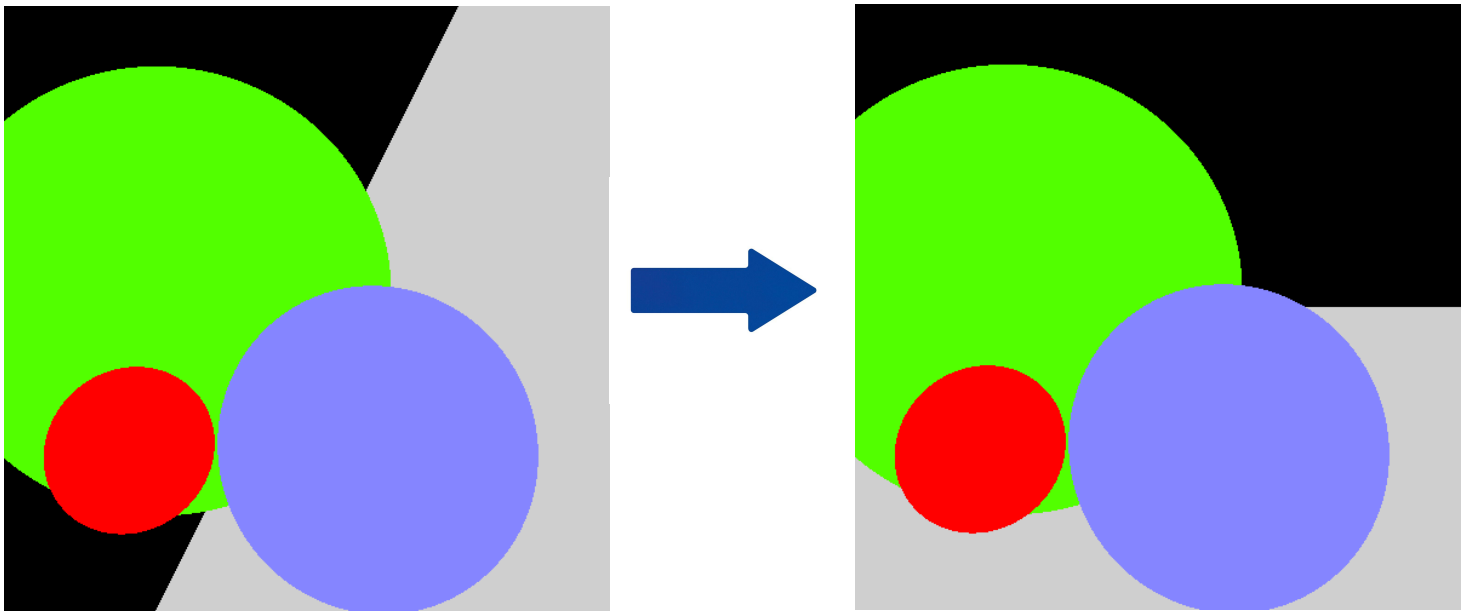
$$\Leftrightarrow < x_s + t\,u - x_p , n_p > = 0$$

$$\Leftrightarrow < t\,u , n_p > + < x_s - x_p , n_p > = 0$$

$$\Leftrightarrow t < u , n_p > = - < x_s - x_p , n_p >$$

$$\Leftrightarrow t = \frac{< x_p - x_s , n_p >}{< u , n_p >}$$

Au départ, nous avons une image avec trois sphères et un plan non horizontal, on modifie la classe *plane* pour placer le plan dans la position voulue :



Intersection avec une sphère

$$\begin{cases} \|x - x_0\|^2 = r^2 \\ x = x_s + tu \end{cases}$$

Avec x_0 centre de la sphère de rayon r, x_s un point de la droite et u vecteur directeur de celle-ci.

$$\Leftrightarrow \|x_s + tu - x_0\|^2 = r^2$$

$$\Leftrightarrow \langle x_s - x_0 + tu, x_s - x_0 + tu \rangle = r^2$$

$$\Leftrightarrow \langle tu, x_s - x_0 + tu \rangle + \langle x_s - x_0, x_s - x_0 + tu \rangle = r^2$$

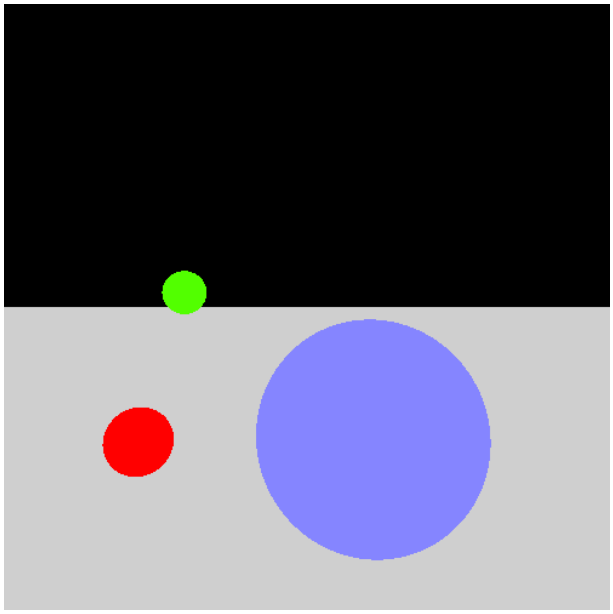
$$\Leftrightarrow \langle tu, x_s - x_0 \rangle + \langle tu, tu \rangle + \langle x_s - x_0, x_s - x_0 \rangle + \langle x_s - x_0, tu \rangle = r^2$$

$$\Leftrightarrow t^2 + 2\langle x_s - x_0, u \rangle t + \|x_s - x_0\|^2 - r^2 = 0$$

$$\Leftrightarrow \begin{aligned} t_1 &= -\langle x_s - x_0, u \rangle + \sqrt{\langle x_s - x_0, u \rangle^2 - \|x_s - x_0\|^2 + r^2} \\ t_2 &= -\langle x_s - x_0, u \rangle - \sqrt{\langle x_s - x_0, u \rangle^2 - \|x_s - x_0\|^2 + r^2} \end{aligned}$$

Si les deux solutions sont positives, on prend la plus petite. Si elles sont de signes opposés on prend la positive, et si les deux sont négatives, on ne renvoie rien.
Le but est d'avoir la plus petite distance positive.

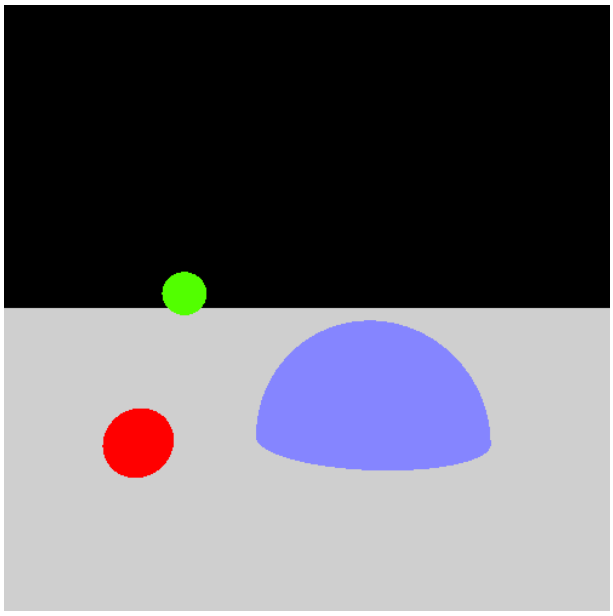
De la même manière que pour le plan, on modifie la classe *sphere* pour replacer nos trois disques:



Lancer de rayons

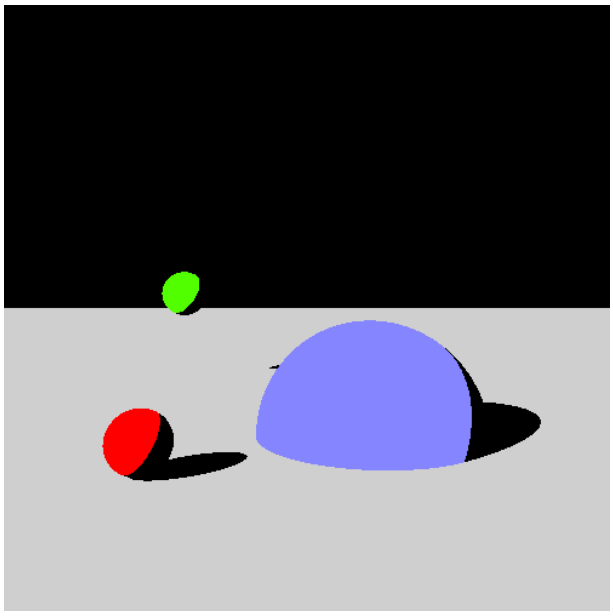
Recherche de l'intersection la plus proche

On recherche les intersections les plus proches de la caméra pour que les sphères ne se superposent pas au plan. On obtient un effet de découpage des sphères par le plan :



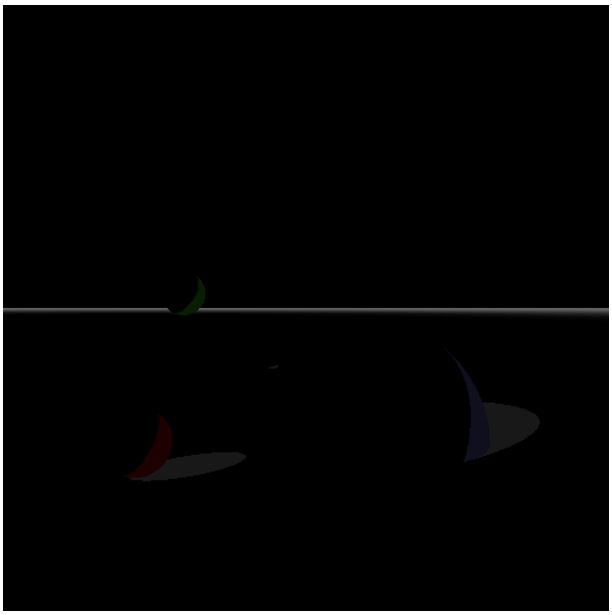
Ombrage

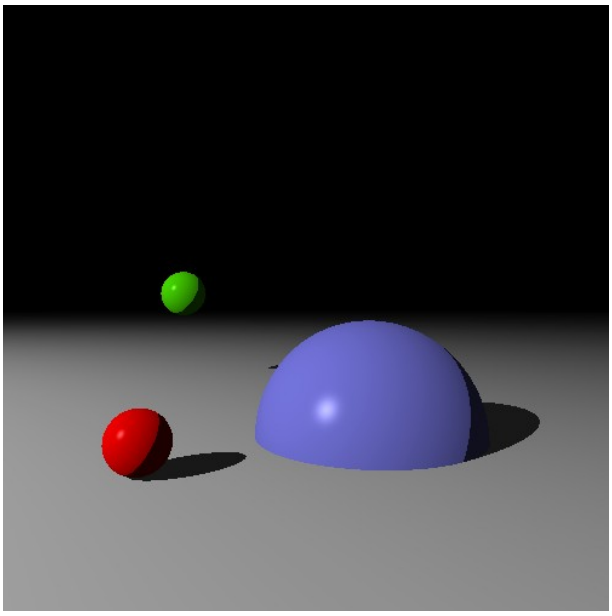
On ajoute maintenant les ombres générées par les sphères sur le plan. Pour cela, on vérifie pour chaque pixel s'il existe un objet (ou un point) entre ce pixel et la source lumineuse.



Illumination

On peut alors ajouter une illumination de Phong à notre image.





Avec de mauvais calculs de normales on obtient l'image de gauche.
Il ne faut pas oublier de diviser la couleur par la distance au carré (entre la source lumineuse et le point d'intersection). Ce calcul donne ainsi un dégradé de luminosité plus on s'éloigne de la source lumineuse.

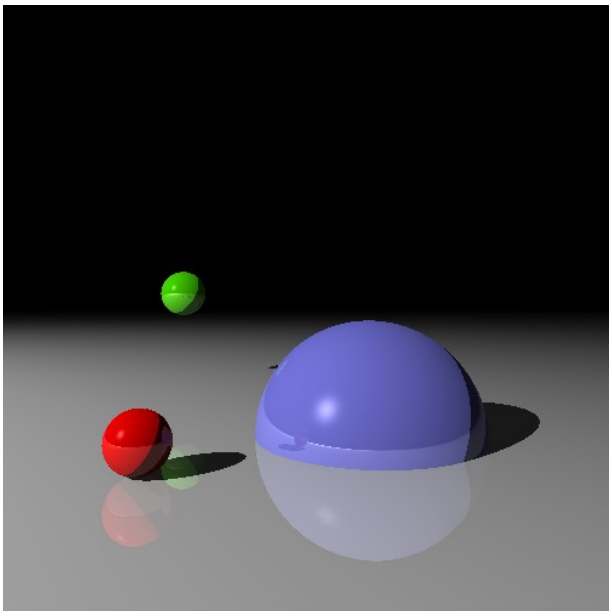
```
c += L.power * L.c * compute_shading(mat.shading(), mat.color_object(), p0, L.p,
   intersection.normal, scene.get_camera().center()) / pow(norm(L.p - p0),2);
```

Il faut ensuite augmenter la puissance lumineuse (*L.power*) qui doit s'adapter aux tailles des objets : plus ils sont grands, plus il faudra augmenter la puissance lumineuse.
Pour obtenir l'éclairage de la photo de droite, nous avons utilisé une puissance lumineuse de 400 (au lieu de un par défaut).

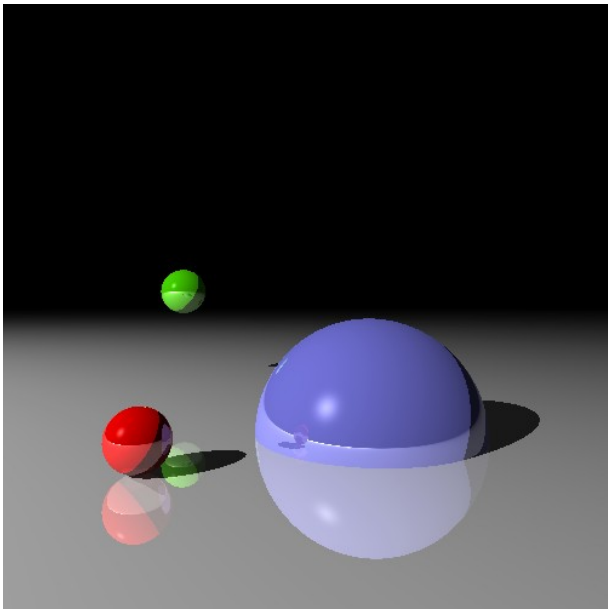
Réflexion

On peut ajouter un effet de réflexion en lançant un second rayon symétrique au premier par rapport à la normale au point d'intersection. Pour lancer ce deuxième rayon, il faudra faire attention à deux choses :
- placer l'origine du nouveau rayon sur le point d'intersection.
- appliquer un léger décalage au rayon (méthode *offset*) afin d'éviter que le rayon soit bloqué dans la sphère (s'il y a des erreurs d'approximation, l'origine se trouve à l'intérieur de la sphère, il va alors rebondir à l'intérieur de celle-ci).

On peut répéter cette action autant que nécessaire, mais pour notre image, deux ou trois itérations suffisent. Le facteur de réflexion permet d'augmenter ou de diminuer cet effet.



Facteur = 0.2

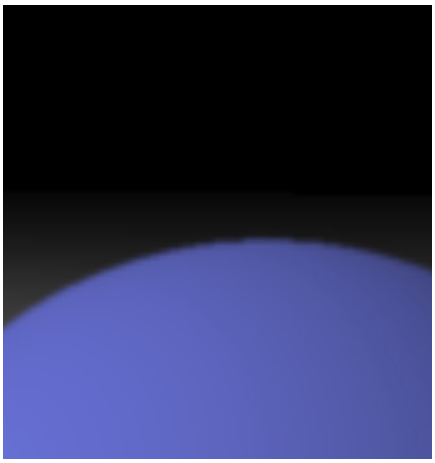


Facteur = 0.4

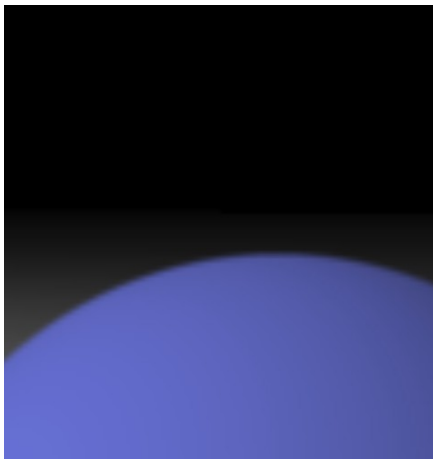
Antialiasing

On remarque, en zoomant, qu'un crénelage est visible sur les contours des sphères. Pour remédier à cela, on calcule la couleur différemment. Au lieu de prendre directement la couleur obtenue après lancer du rayon, on effectue un moyenne pondérée avec les couleurs autours de ce point. L'augmentation du nombre d'échantillons améliore le rendu du contour.

Sans anti-aliasing

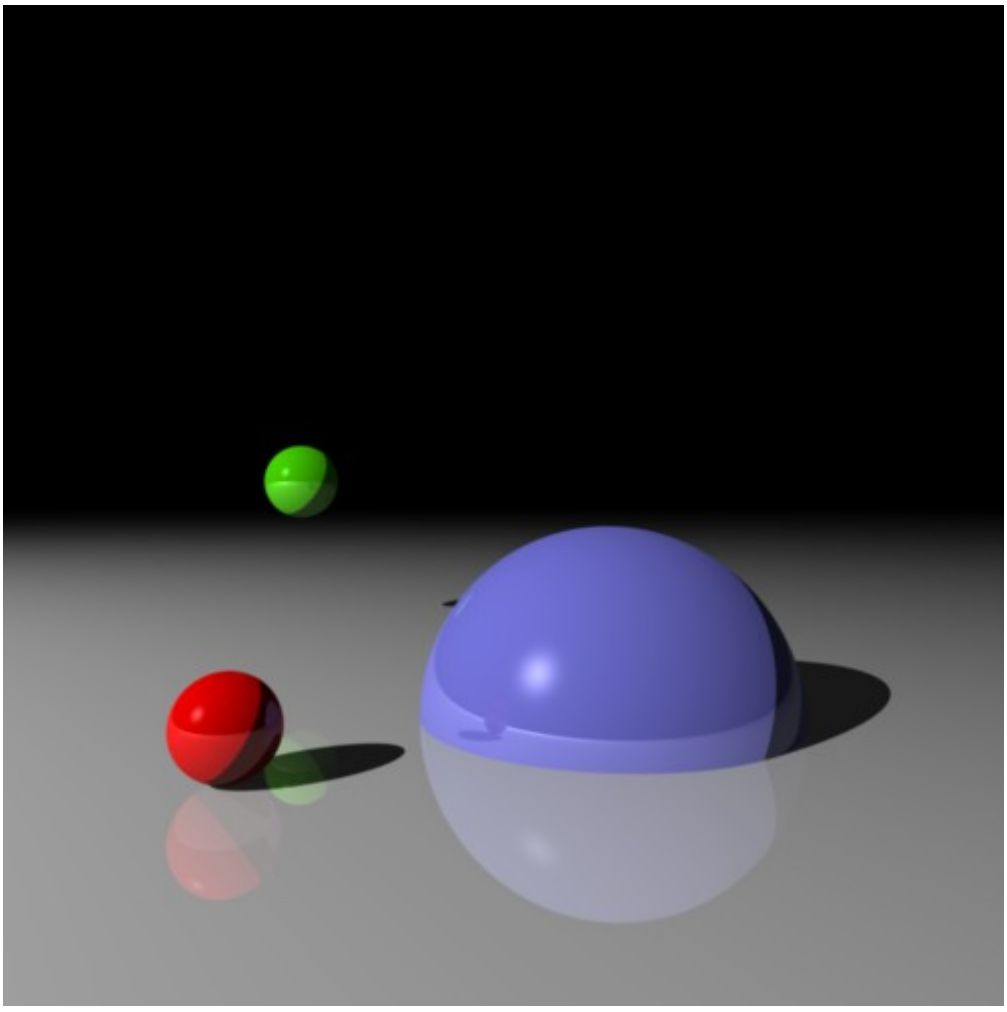


5 échantillons



15 échantillons

On obtient finalement l'image suivante :



Parallélisation :

Après avoir implémenter l'aliasing, le temps d’exécution du programme devient conséquent. Ce dernier traitant les pixels un par un, il est facilement parallélisable (instruction #pragma omp parallel for avant les boucles for à paralléliser). Le temps gagné avec la parallélisation est important. (entre 5 et 6 plus rapide).