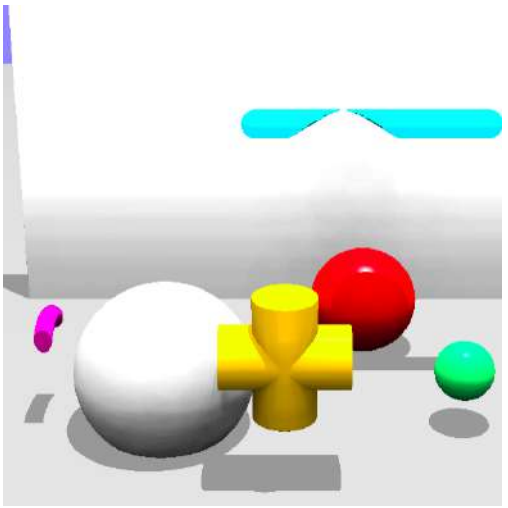
Informatique Graphique : compte rendu

Chouati Linda p1805862 && Faussurier Marc p1707031

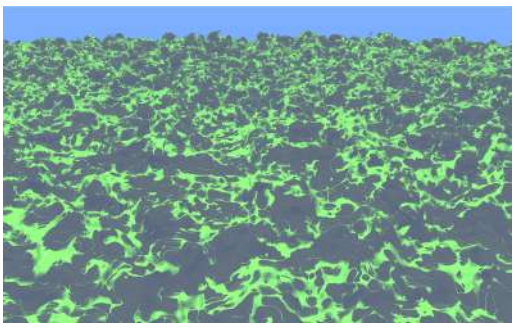
1. Modélisation et modélisation avancée



En bleu clair : la différence entre une capsule et une ellipsoid.

En jaune : l'union de deux cylindres. Utilisation d'une primitive de rotation pour faire une croix.

En violet : l'intersection d'un torus dans une boîte.
https://www.shadertoy.com/view/4ctyWs

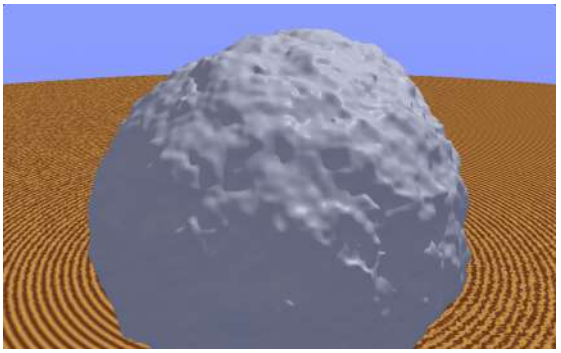


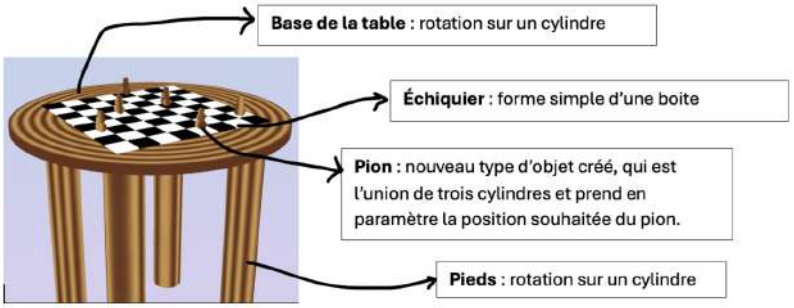
Il s'agit juste d'une primitive simple, un plan, auquel on a rajouté du bruit dans le but de modéliser une montagne.

https://www.shadertoy.com/view/XXSBW1

Une fois le bruit mis en place, cela nous a permis de faire la texture bois, en utilisant une fonction de turbulence pour imiter les veinures du bois. Ici, on a deux primitives simples dont un plan et une sphère.

https://www.shadertoy.com/view/M3BBRy



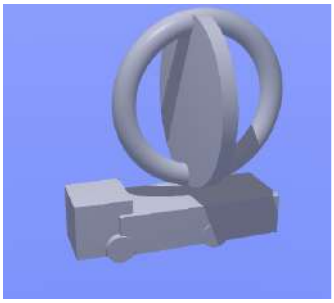


Pour rendre l'image plus réaliste, on a appliqué des textures. Ainsi, la table est en bois et l'échiquier est en damier avec une alternance de cases noires et blanches.

https://www.shadertoy.com/view/M3SBWh

Total des primitives : 1 boîte, 1 cylindre pour le plateau, 4 cylindres pour les pieds, et 6 tours (chaque tour comportant 3 cylindres) = 24 primitives.

2. Volumes englobants



Total des primitives : 8 primitives

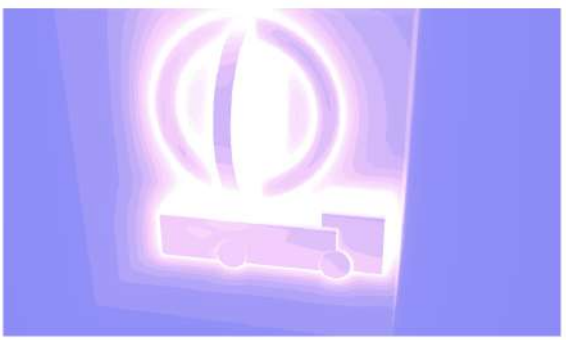
Total de nœuds dans l'arbre : 8 primitives + 7 unions = 15 nœuds

Coût total estimé : 3 (remorque=>boîte) + 3 (cabine=>boîte) + 28 (roues=>cylindre) + 7 (pièce=>cylindre) + 11 (anneau=tore) = 52

On a rajouté un coût arbitraire pour les primitives en tenant quand même compte de la complexité de la primitive. Ainsi, on constate que le coût est beaucoup moins important sur l'image avec un objet englobant. Ce qui entraîne forcément des temps de calcul plus rapides.

https://www.shadertoy.com/view/M3jyDz **https://www.shadertoy.com/view/lXScWh**

Cout du lancer du rayon directement effectué sur le camion



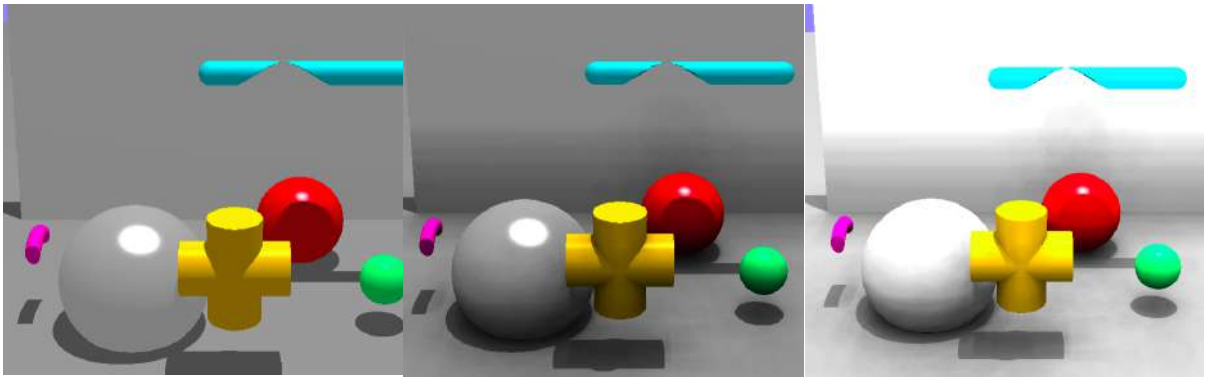
Cout du lancer du rayon avec un objet englobant

3. Éclairement

```
struct Material {
    vec3 color;          // Couleur du matériau
    float ambient;       // Coefficient de réflexion ambiante
    float diffuse;       // Coefficient de réflexion diffuse
    float specular;      // Coefficient de réflexion spéculaire
    float reflectivity; // Facteur de réflexion (0 = mat, 1 = miroir)
};
```

Notre structure pour les matériaux

Occlusion ambiante - https://www.shadertoy.com/view/4ctyWs

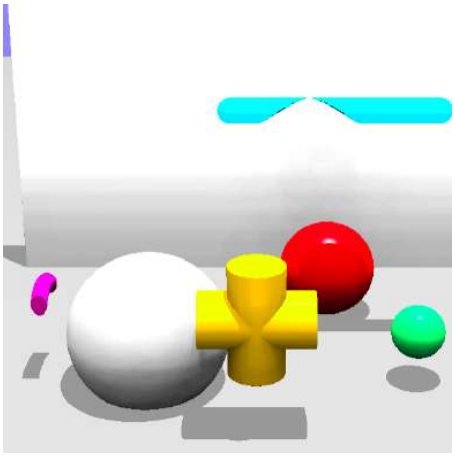


AO désactivée *AO activée avec facteur 0.5* *AO activée avec facteur 1*

L'occlusion ambiante est néscéssaire pour voir la profondeur du cylindre jaune par exemple.

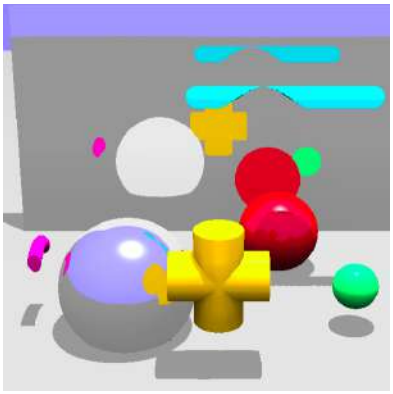
Notre occlusion ambiante est calculée à chaque cycle, elle pourrait être précalculée.

Sur les deux figures ou l'AO est activée; on remarque que les ombres au sol ne sont pas nettes. La fonction `ShadeWithAO2` corrige ce problème en modulant l'occlusion ambiante selon l'intensité de la lumière directe sur chaque point.



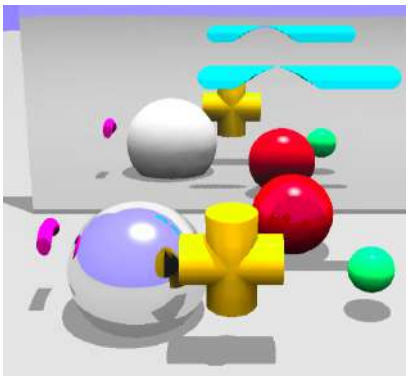
AO activée avec facteur 1 et ombres au sol corrigées.

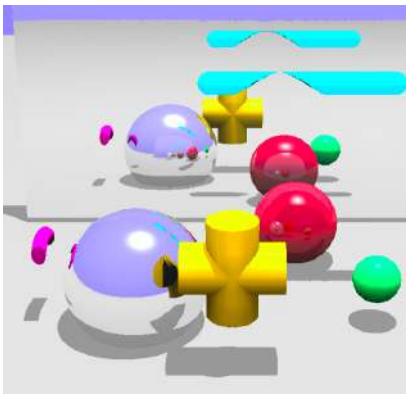
Réflexion - https://www.shadertoy.com/view/4ctyWs



La première version de notre réflexion (`ShadeWithAO2AndReflection`) fonctionne mais on y remarque deux problèmes : l'ombrage est absent sur l'image réfléchie et il n'y a pas plusieurs niveaux de réflexion.

La fonction `ShadeWithAO2AndReflectionWithShadows` corrige cela en appelant la fonction d'ombrage sur l'image réfléchie.



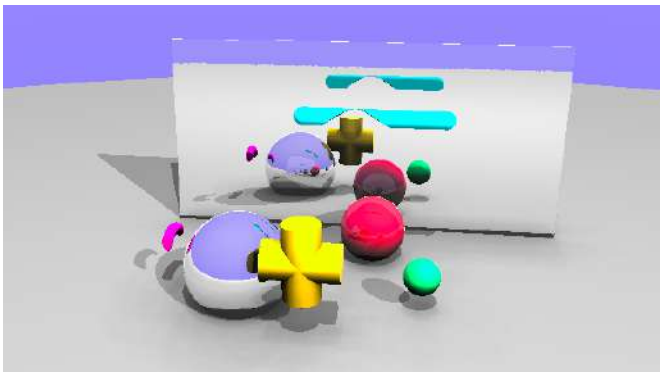


La fonction `ShadeWithAO2AndNestedReflection` est la version de notre réflexion ajoutant plusieurs niveaux de réflexion; paramétrable via une constante `maxReflections` (ici 5). Le langage GLSL ne supportant évidemment pas la récursion à cause de la nature des GPU, une approche itérative a donc dû être explorée.

Notre soleil se déplace de manière réaliste. Il s'agit d'une source de lumière ponctuelle suivant un cercle.
https://www.shadertoy.com/view/4ctyWs

Éclairage avec deux sources de lumière: dont une directionnelle.

https://www.shadertoy.com/view/4fdcDl



```
struct Light {
    vec3 position;      // Position for point lights
    vec3 direction;     // Direction for directional lights
    vec3 color;         // Light color
    float intensity;    // Light intensity
    bool isDirectional; // Whether the light is directional
};

const float sunSpeed = .40; // Speed factor
const int numLights = 2; // Nombre de lumières
Light lights[numLights];

void initLights() {
    lights[0] = Light(vec3(0.0, 5.0, 5.0), vec3(-1.0, -1.0, -1.0), vec3(1.0, 1.0, 1.0), .2, true);
    // Sun-like moving light source
    vec3 sunPos = vec3(50.0 * sin(iTime * sunSpeed), 10.0, 100.0 * cos(iTime * sunSpeed));
    lights[1] = Light(sunPos, vec3(0.0), vec3(1.0, 1.0, 1.0), 1.0, false); // Set as a moving point
}
```

Initialisation de plusieurs lumières. Ici une directionnelle et une ponctuelle.