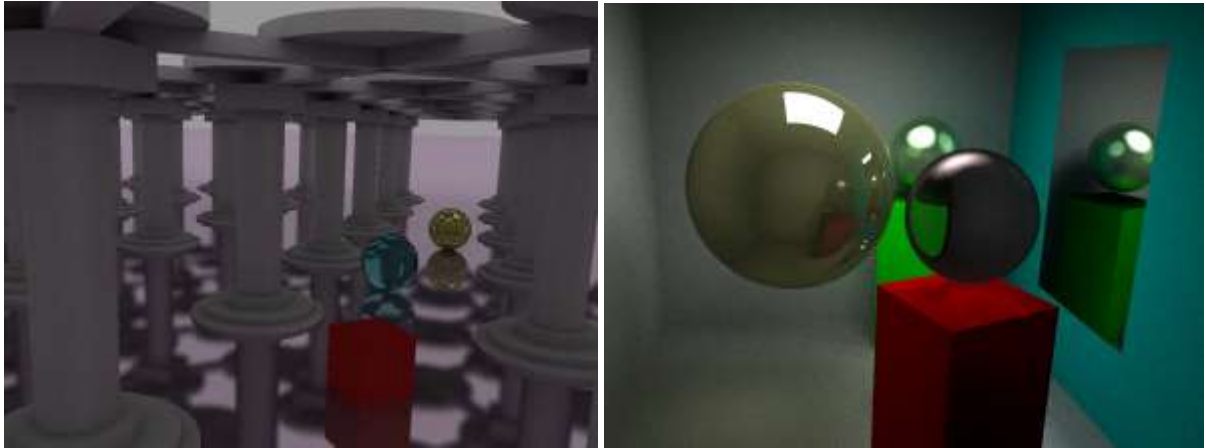


Projet Montecarlo Path-tracing



But : programmer l'algorithme du path-tracing

Le but du projet est de développer la fonction `vec3 random_path(in vec3 D, in vec3 O)` dans un fichier shader (montecarlo.frag, ...). Celle-ci doit renvoyer l'énergie lumineuse collectée par le chemin initiée par le rayon (O,D)

Les programmes fournis *drawsampling & montecarlo*

DrawSampling

Permet d'afficher un échantillonnage, il suffit d'écrire le code de la fonction `vec3 random_ray(in vec3 N)` dans un fichier .frag en se basant sur l'exemple `hsphere_wrong_sampling.frag`. Le paramètre N définit la normale à l'échantillon

Les touches O et P permettent de changer de shader (cf ligne 20 du .cpp)

Montecarlo

Le programme lance des rayons à travers les pixels de l'écran depuis la caméra. Les images sont stockées et cumulées (moyennées) en fonction de l'interface. Vous n'avez donc qu'à coder dans le shader la fonction de parcours du rayon qui renvoie la couleur collectée.

Touches

- A Z E R T Y U I : changer de scène. Les scènes sont créées dans les fonctions `scene_xxxx`.
- O P : shader précédent, suivant (les noms de fichiers sont à la ligne 27 du fichier .cpp)

Vous pouvez modifier les scènes. Les paramètres de la classe Material sont : la couleur RGBA, A étant l'opacité (1 opaque 0 transparent, avec 2 paramètres optionnels, la brillance (shininess) et la rugosité (roughness). Une lumière est générée par la fonction `Material :light(C,I)` C pour la couleur et I pour l'intensité de la lumière.

Pour générer de la géométrie, on peut utiliser les méthodes suivantes (de l'objet `bvh_gpu_scene`):

- `add_sphere`
- `add_cube`
- `add_oriented_Quad`
- `add_cylinder`

Les paramètres sont une matrice de transformation et un *Material*

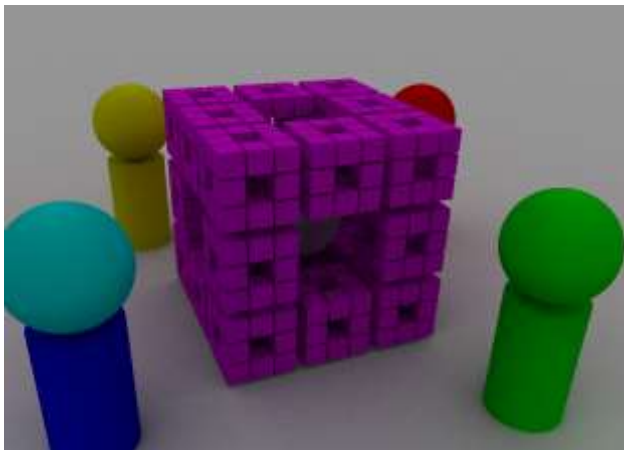
Dans le menu vous pouvez choisir:

- Subsampling: choisir la taille de l'image calculée
- Nombre de rebond du chemin (passé en uniforme au shader)
- Nombre de chemins lancés en mode « interactif »

Si vous cochez lock l'interface se bloque et les images s'accumulent (et se moyennent). Si vous cochez freeze, l'image est alors figée et les calculs sont stoppés

Attention : pour ne pas limiter les performances la synchronisation verticale doit être désactivée ou laissée au choix de l'application.

Phase 1 : Montecarlo purement diffu



Pour cette première version, le rayon on rebondit au « hasard » dans l'hémisphère centré autour de la normale, en accumulant une atténuation (due à la couleur et à l'angle du rayon par rapport à la normale au point d'impact). Si on finit par toucher (se limiter à N rebonds) une lumière on calcule l'énergie finale, sinon on renvoie 0 (ou 0,0,0). Il est possible de définir un ciel procédural qui permettra à l'algorithme de converger beaucoup plus vite que lorsque l'on renvoie du noir.

La fonction d'échantillonnage de l'hémisphère est très importante. Pour commencer coder la et tester la dans **DrawSampling**. Ceci vous permettra de visualiser la direction des rayons et de vérifier le coté uniformément aléatoire. Le pseudo code de la version correct est donné dans l'exemple fourni !

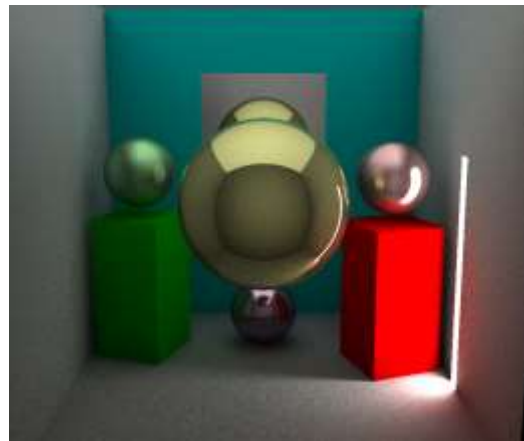
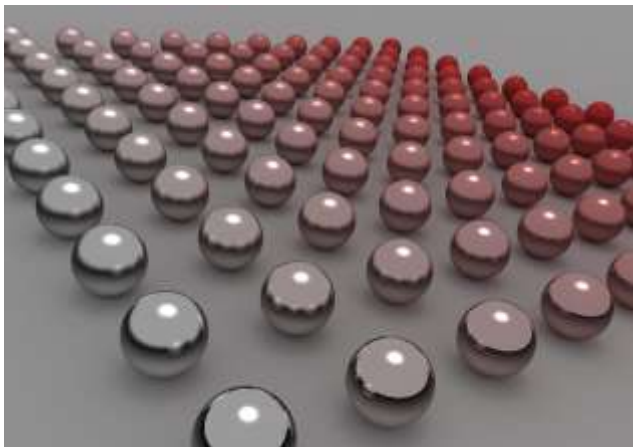
Pour tout le projet, en plus du GLSL, vous pourrez utiliser les fonctions fournies vues en TP :

- `void traverse_all_bvh(in vec3 O, in vec3 D);`
- `bool hit();`
- `void intersection_info(out vec3 N, out vec3 P);`
- `vec4 mat = intersection_mat_info();`
- `vec4 col = intersection_color_info();`

Et on aura besoin de nombres aléatoires :

- `float random_float();` // renvoie un nombre aleatoire dans [0,1]
- `vec2 random_vec2();` // 2 nombres aléatoires
- `vec2 random_vec3();` // 3 ...
- `vec2 random_vec4();` // 4

Phase 2 : Matériaux spéculaires

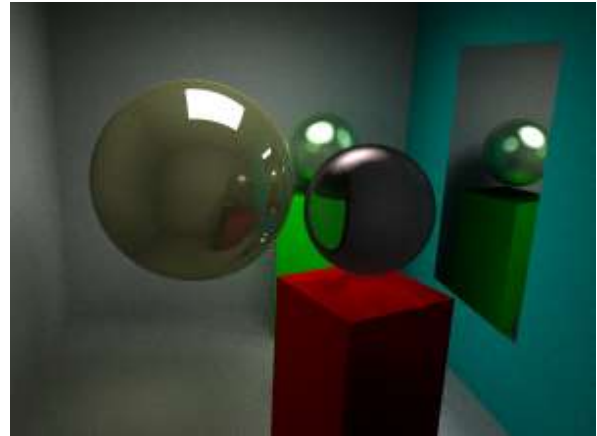
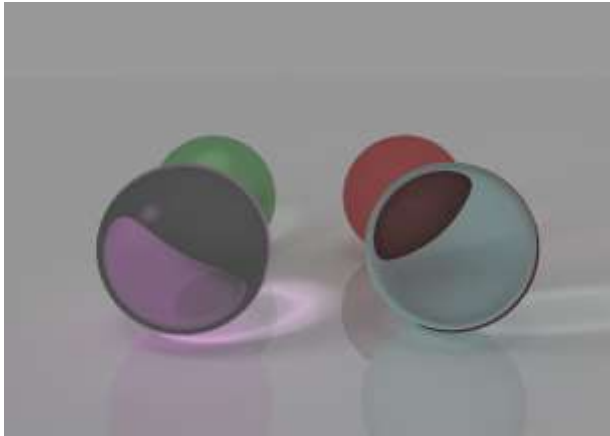


Modifier la fonction `random_path` pour utiliser les paramètres de matériau : `.r` pour la brillance (metal vs plastic) et `.g` pour la rugosité afin de tirer des rayons dans une direction particulières. Modifier votre échantillonnage de l'hémisphère pour prendre en compte la rugosité. Faire en sorte que lors de la convergence le bon pourcentage de rayon ait été lancé dans la bonne direction.

Vous pouvez commencer par utiliser en plus du `diffu` uniquement un matériau totalement spéculaire (miroir) qui est le cas le plus simple pour ensuite introduire des matériaux intermédiaires plus complexes.

Phase 3 : Matériaux transparents

Faire de même pour les matériaux transparents.



Pour les phases 2 et 3 vous pourrez vous baser sur la correction du TP