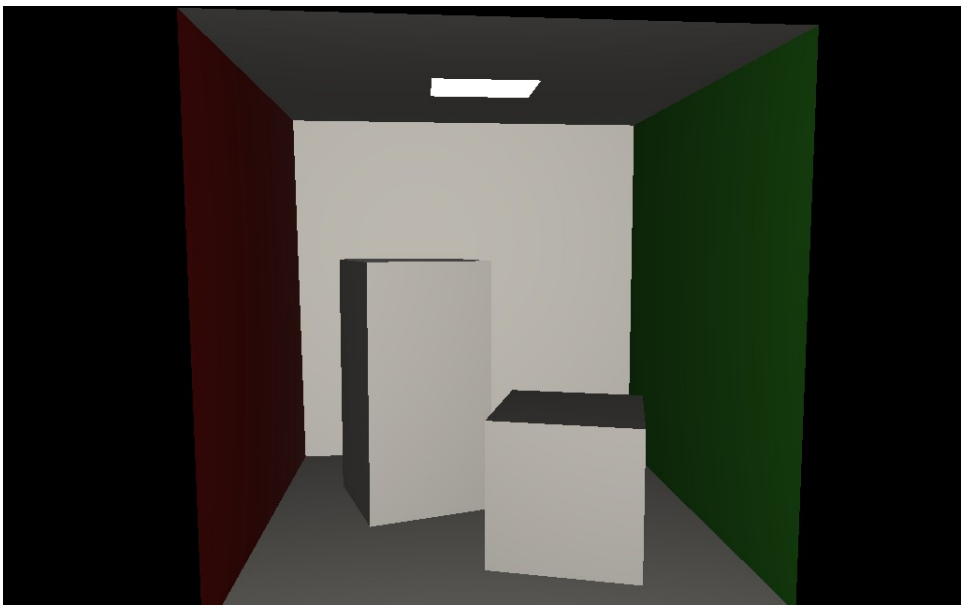
TP3 - Lancer de rayons et intégration numérique

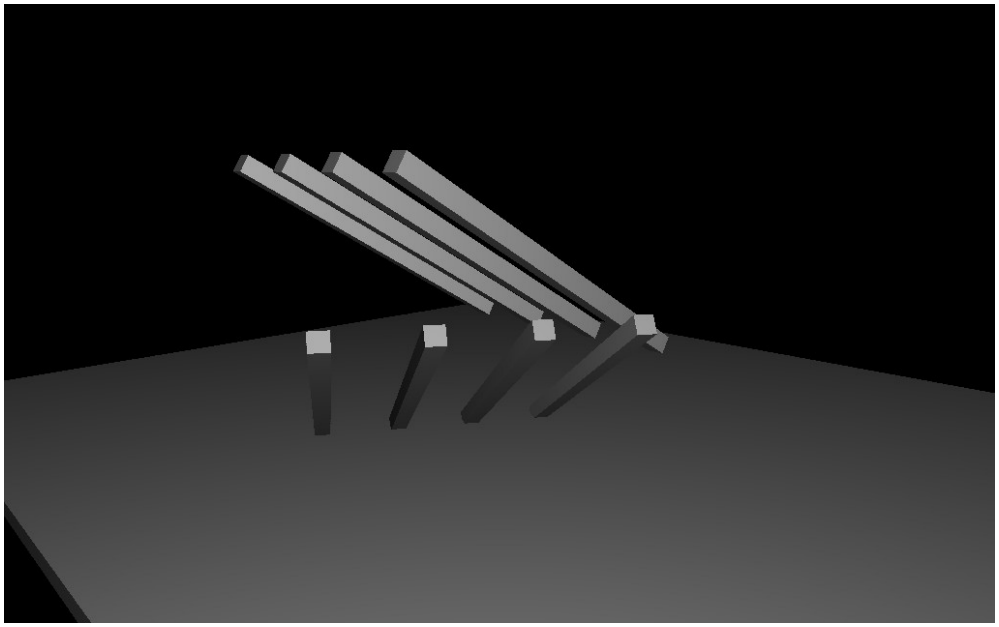
HUYNH Cong Lap - 11419778 - Travail seul

Partie 1 : Visualisation - Partie 1.cbp



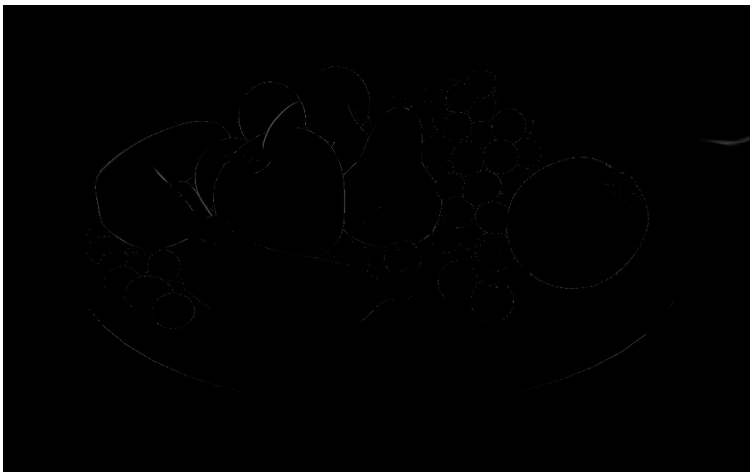
Cornell.obj + orbiter.txt

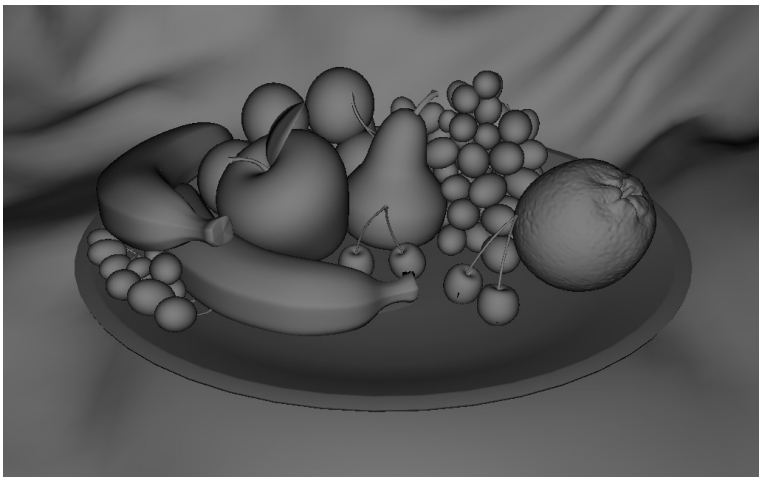
I implement basic Phong lighting: Diffuse + Ambient (Not yet global illumination) + Specular



Shadow.obj + shadow.txt (orbiter)

Fruit Scene with Phong Lighting:



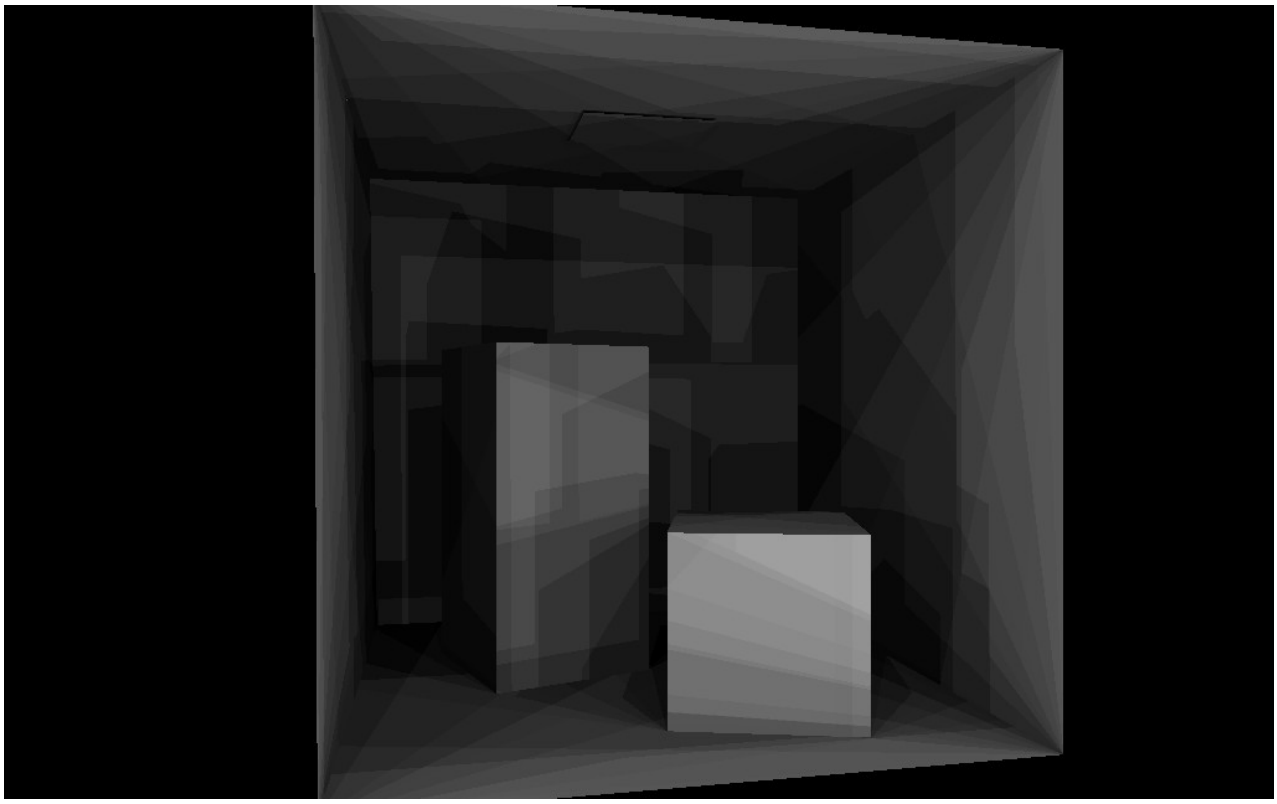


Specular + Diffuse

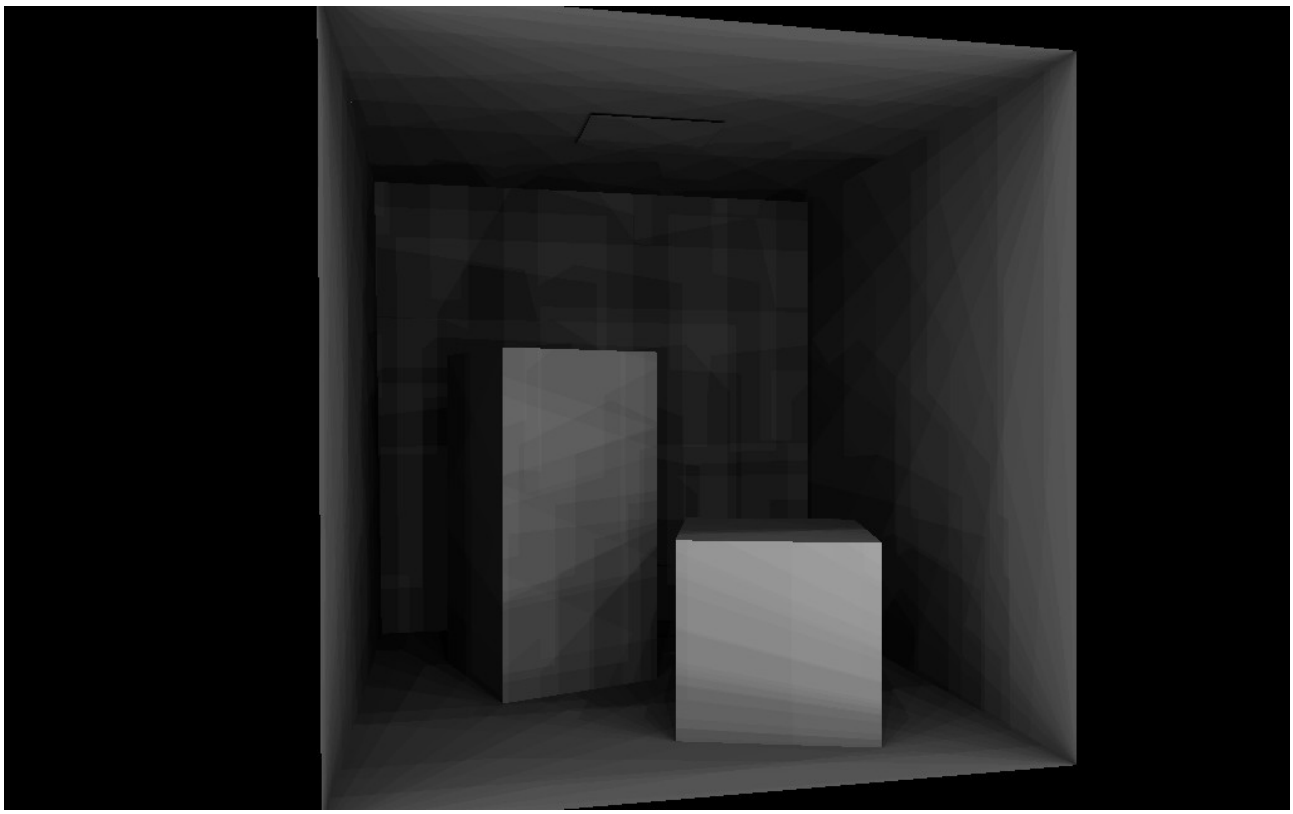
Partie 2 : Éclairage ambiant - Partie 2.cbp

Spirale de Fibonacci VS Directions aléatoires

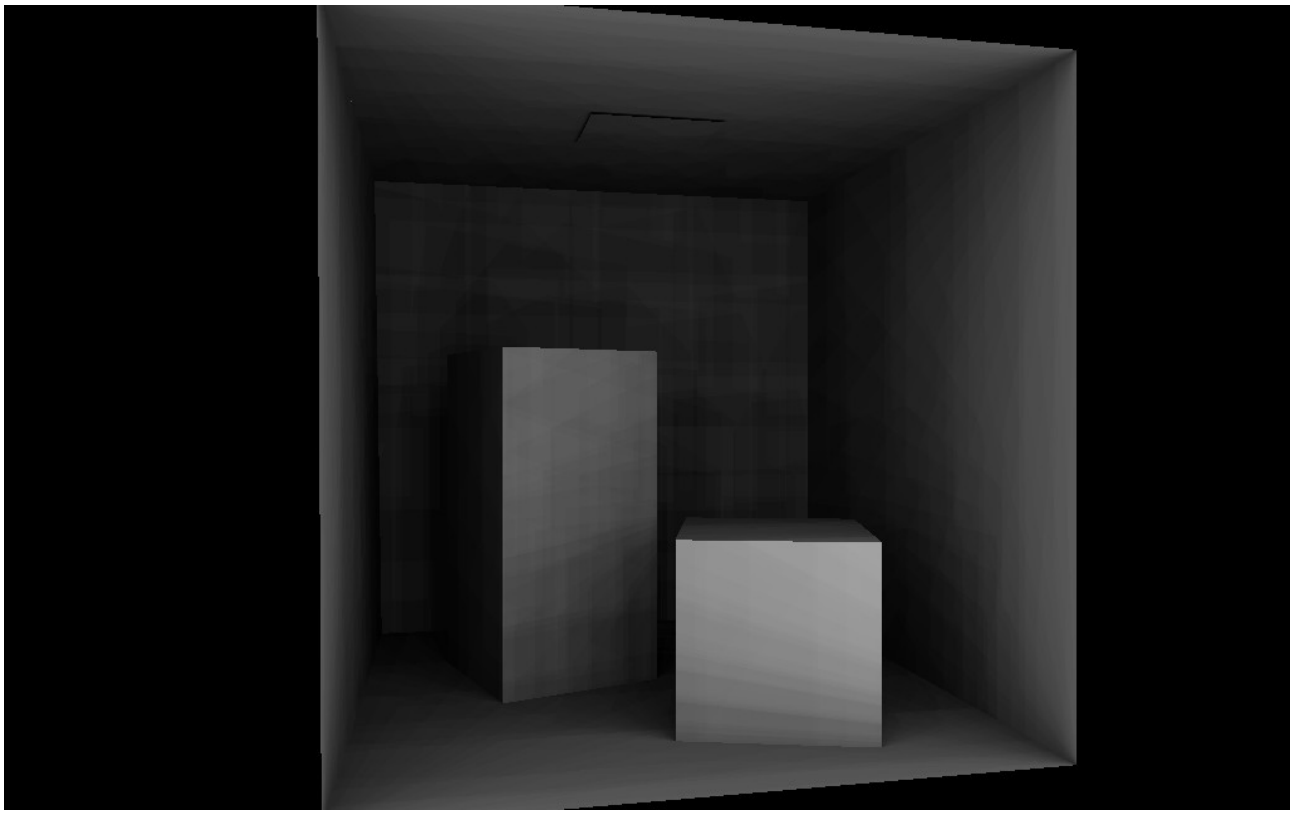
Spirale de Fibonacci perturbée



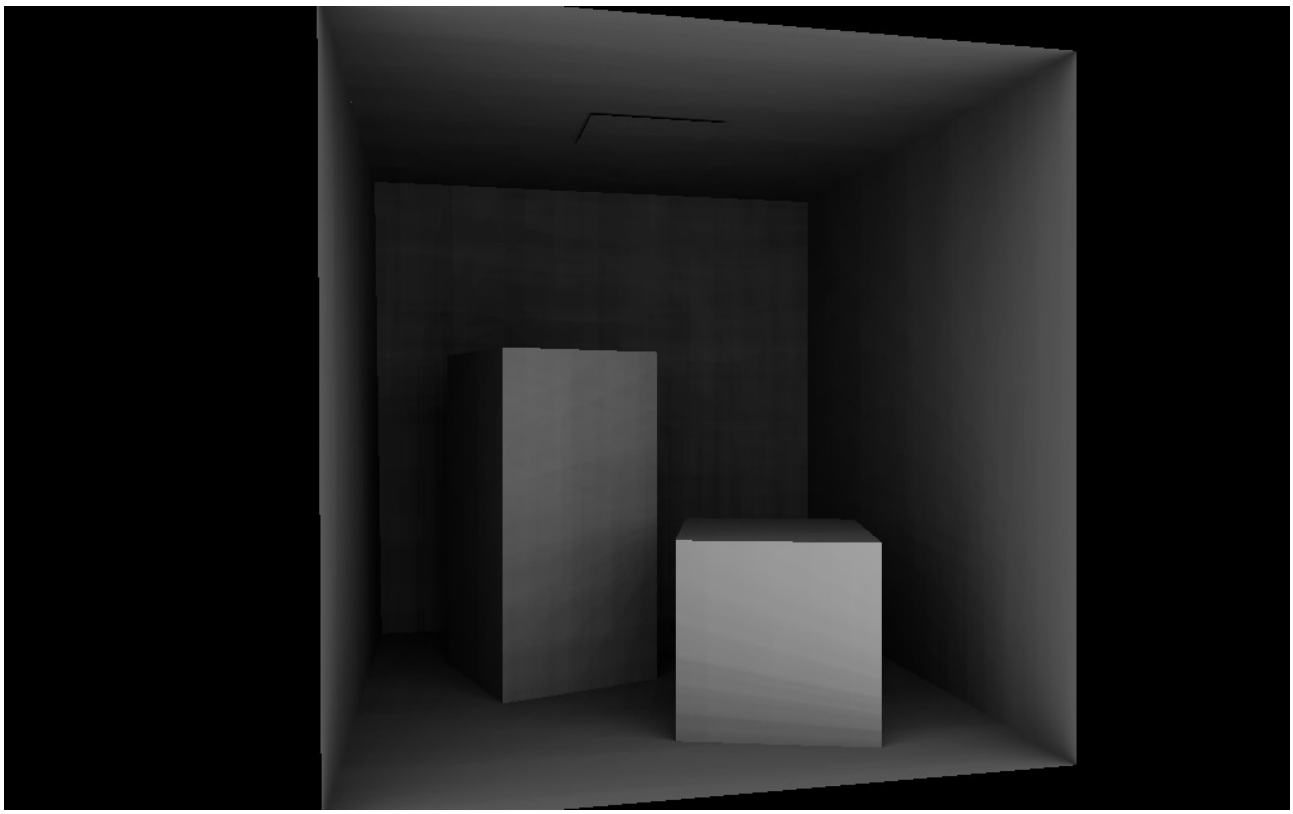
32 directions



64 directions

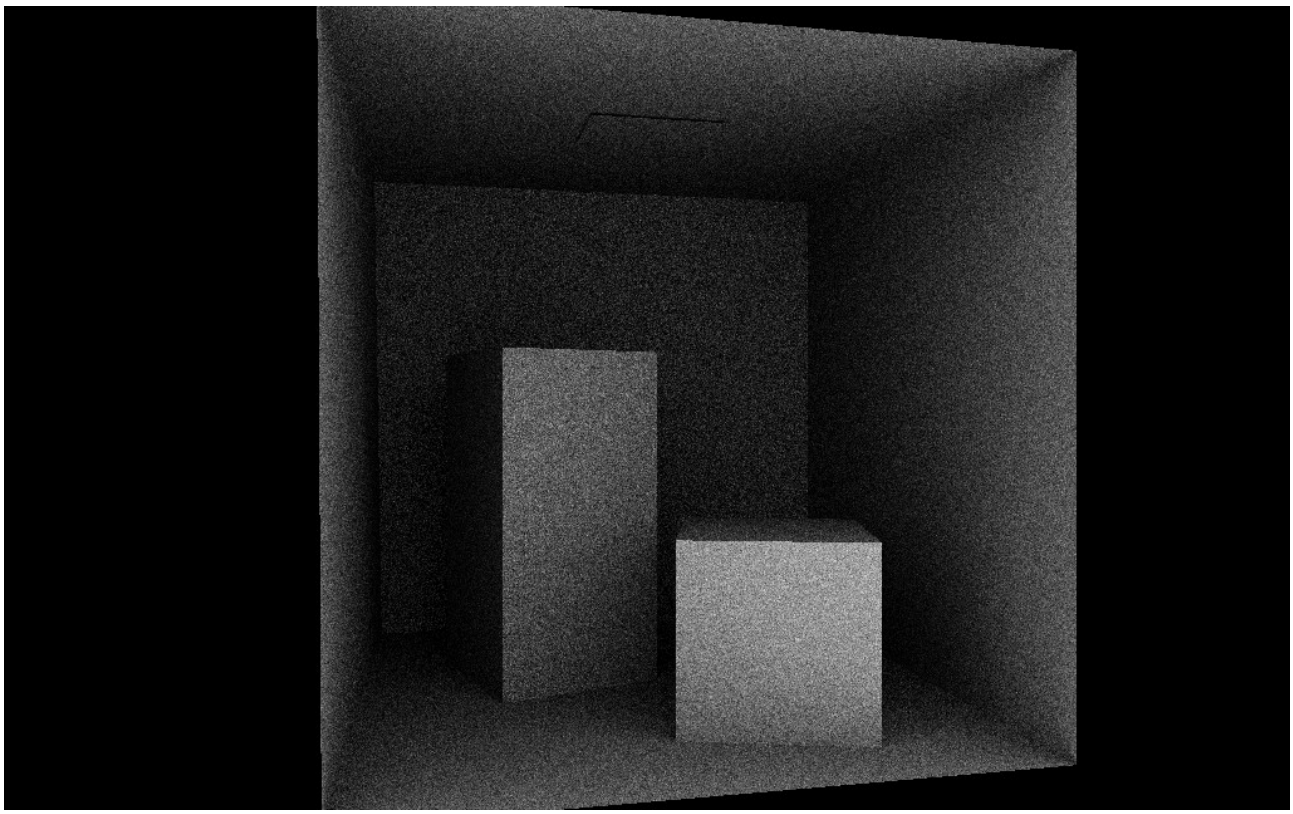


128 directions

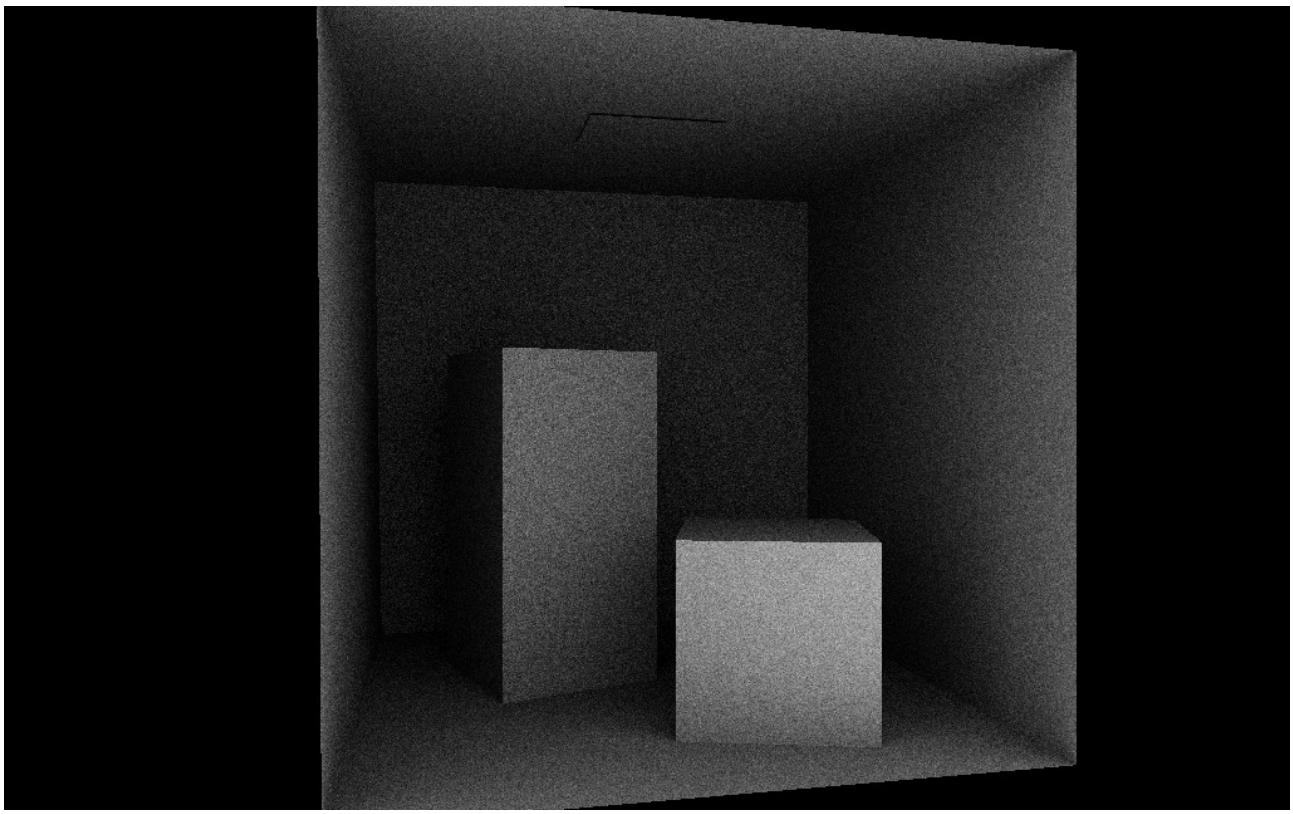


256 directions

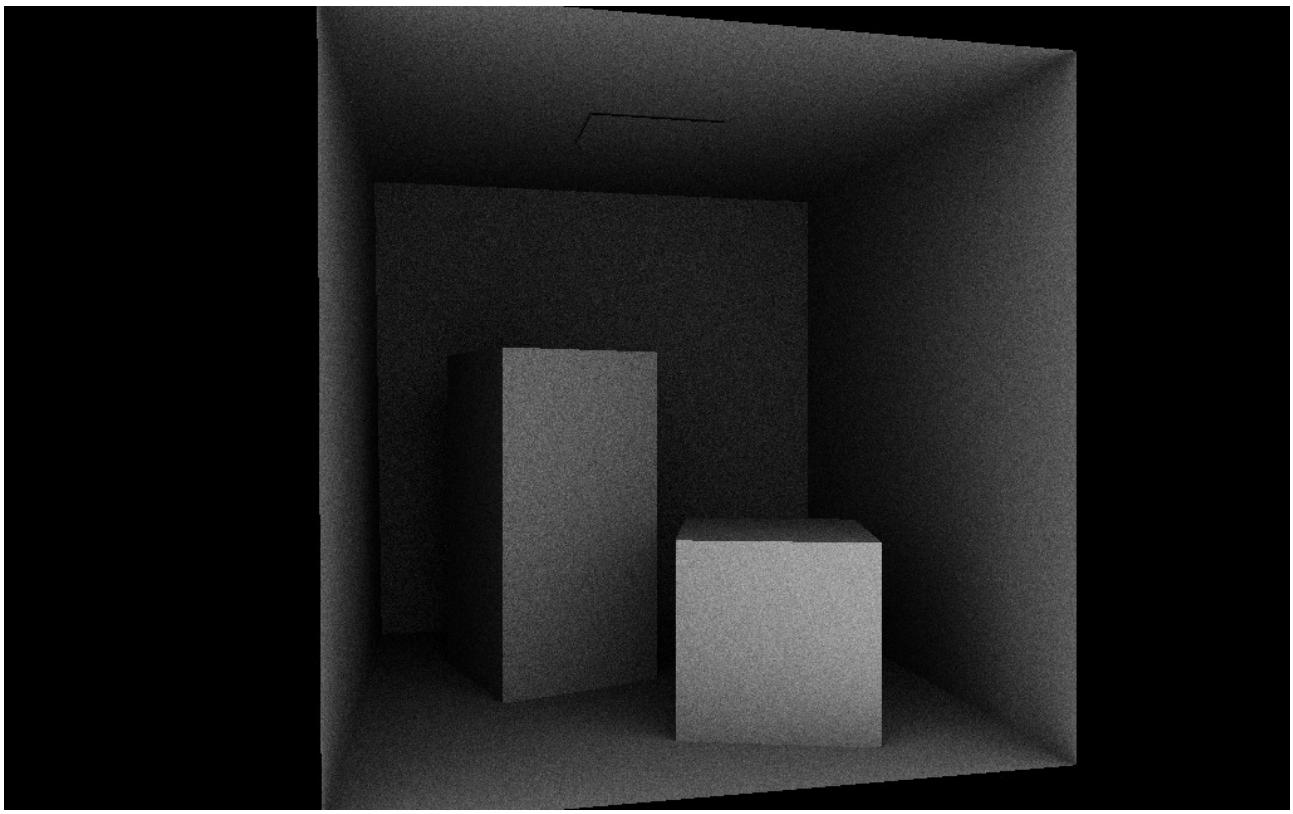
Directions aléatoires



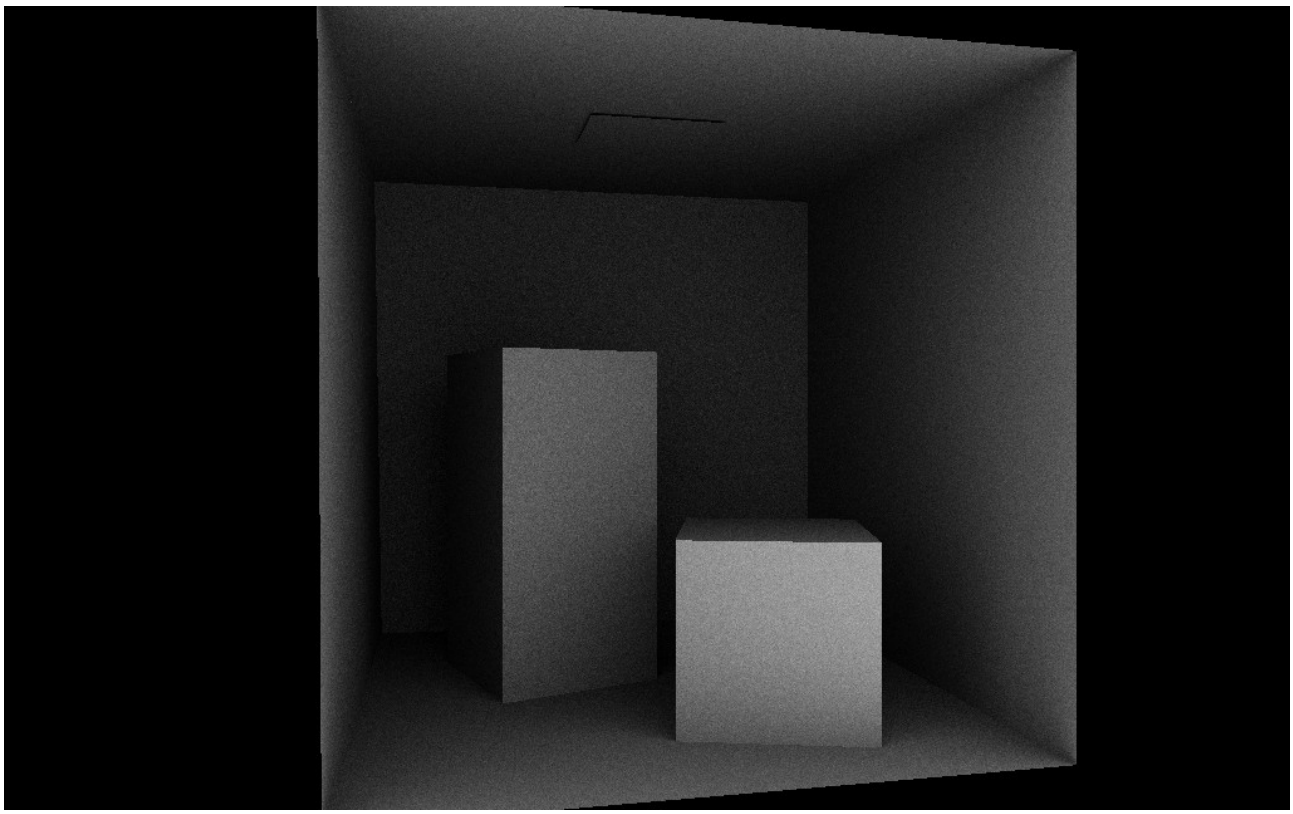
32 directions



64 directions



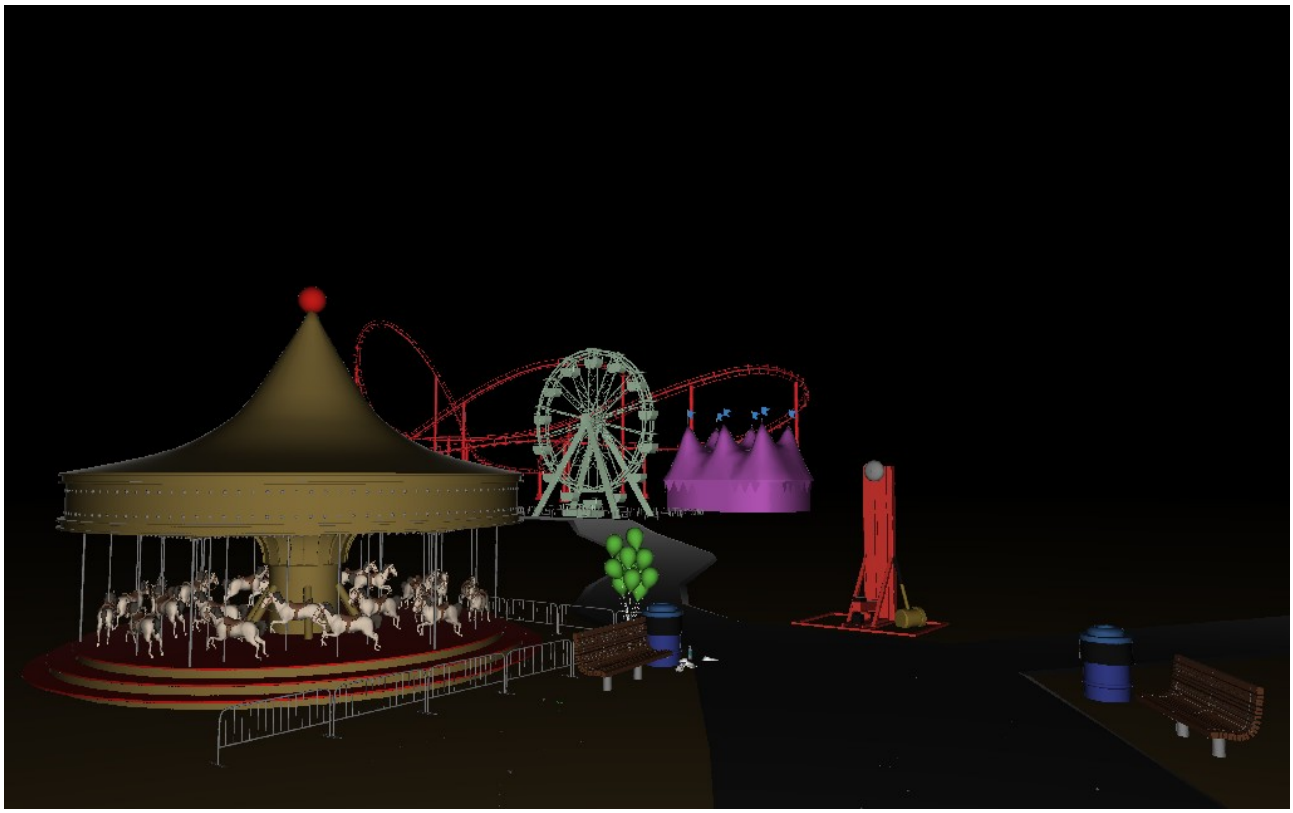
128 directions



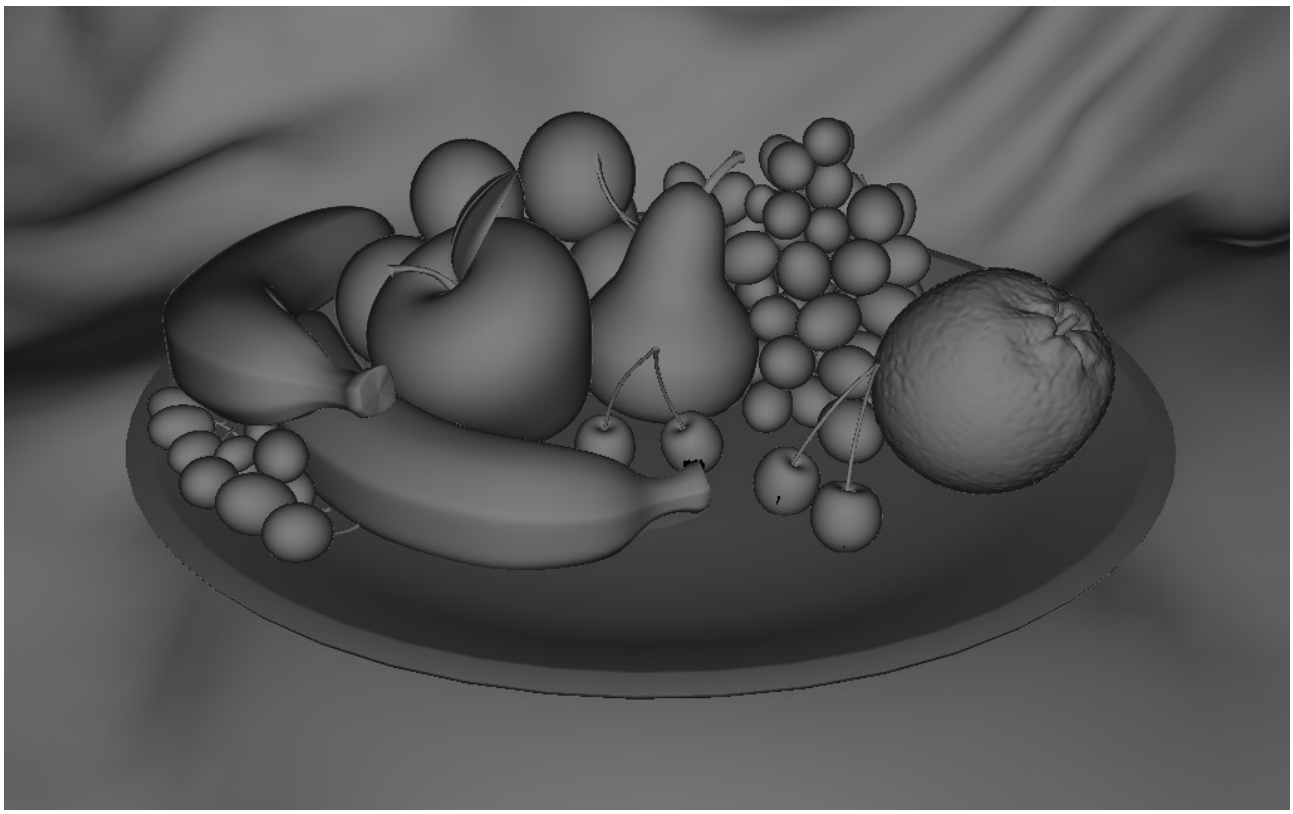
256 directions

Partie 3 : Structures accélératrices - Partie 3.cbp

BVH et algorithme de parcours pour calculer **la diffusion** d'une scène "ouverte" – Parcours ordonné



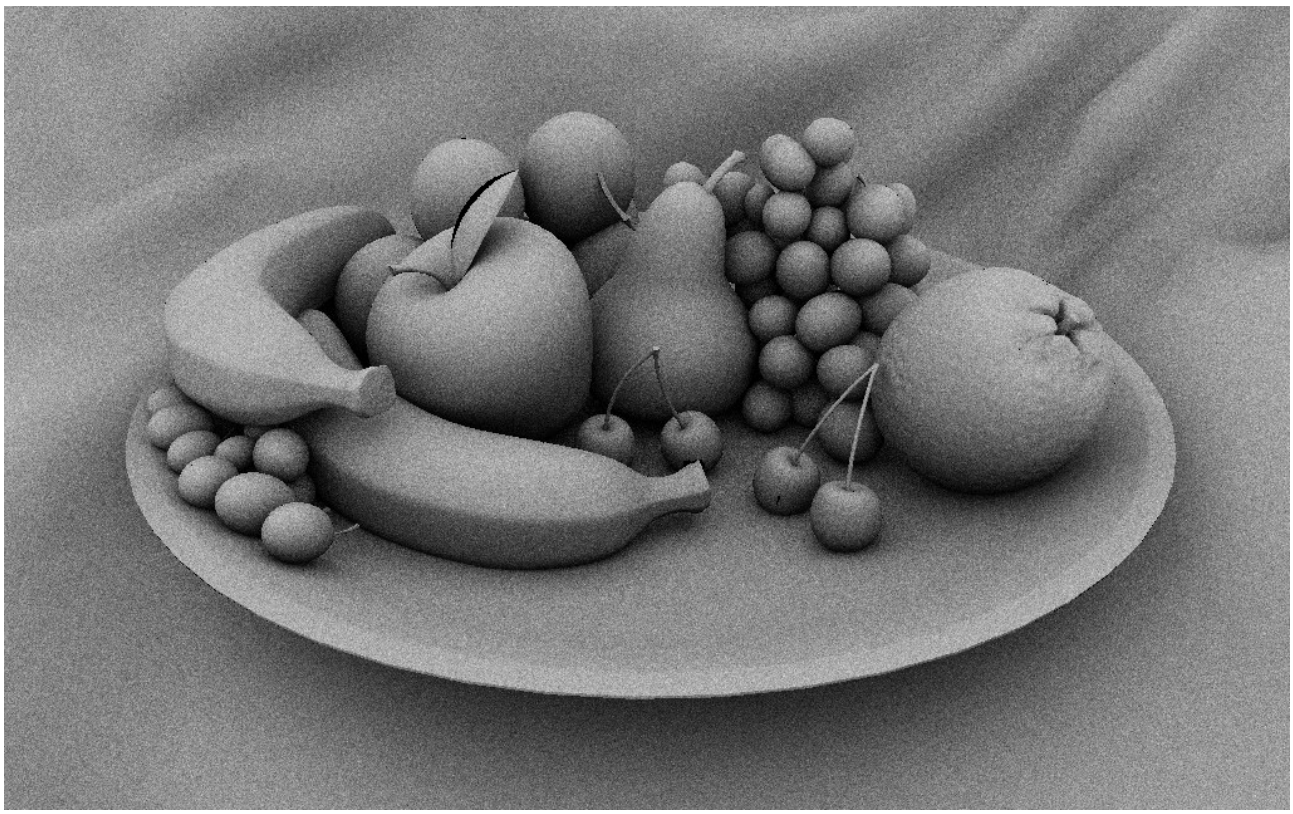
TheCarnival.obj - Triangles: 449 858 - SAH cost: 21,7 - Nodes: 889 715



Fruit_v2.obj - Triangles: 207 226 - SAH cost: 40,48 - Nodes: 414451

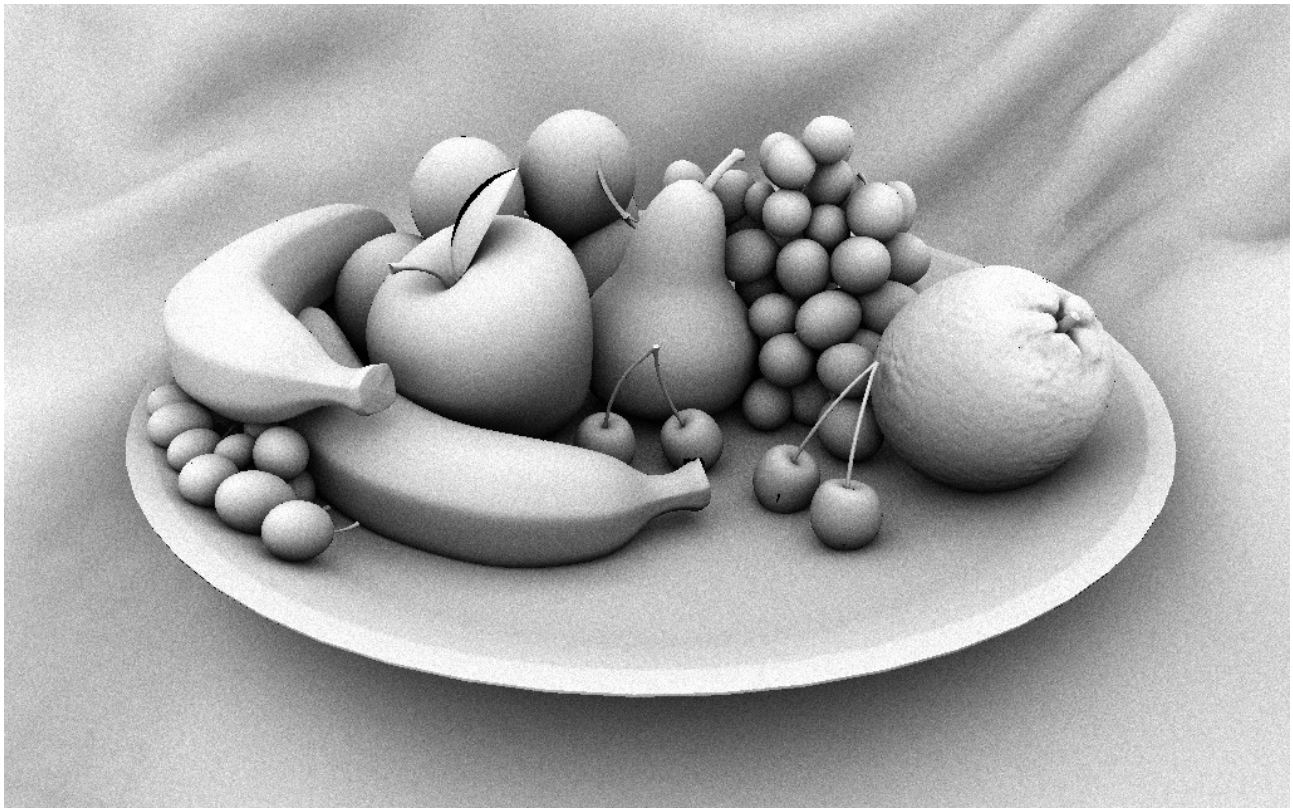
Lighting_Challenge_24_theCabin.obj - Triangles: 422 735 - SAH cost: 29.9 - Nodes: 845 469

BVH et algorithme de parcours pour calculer **l'éclairage ambiant** d'une scène "ouverte" – Parcours ordonné

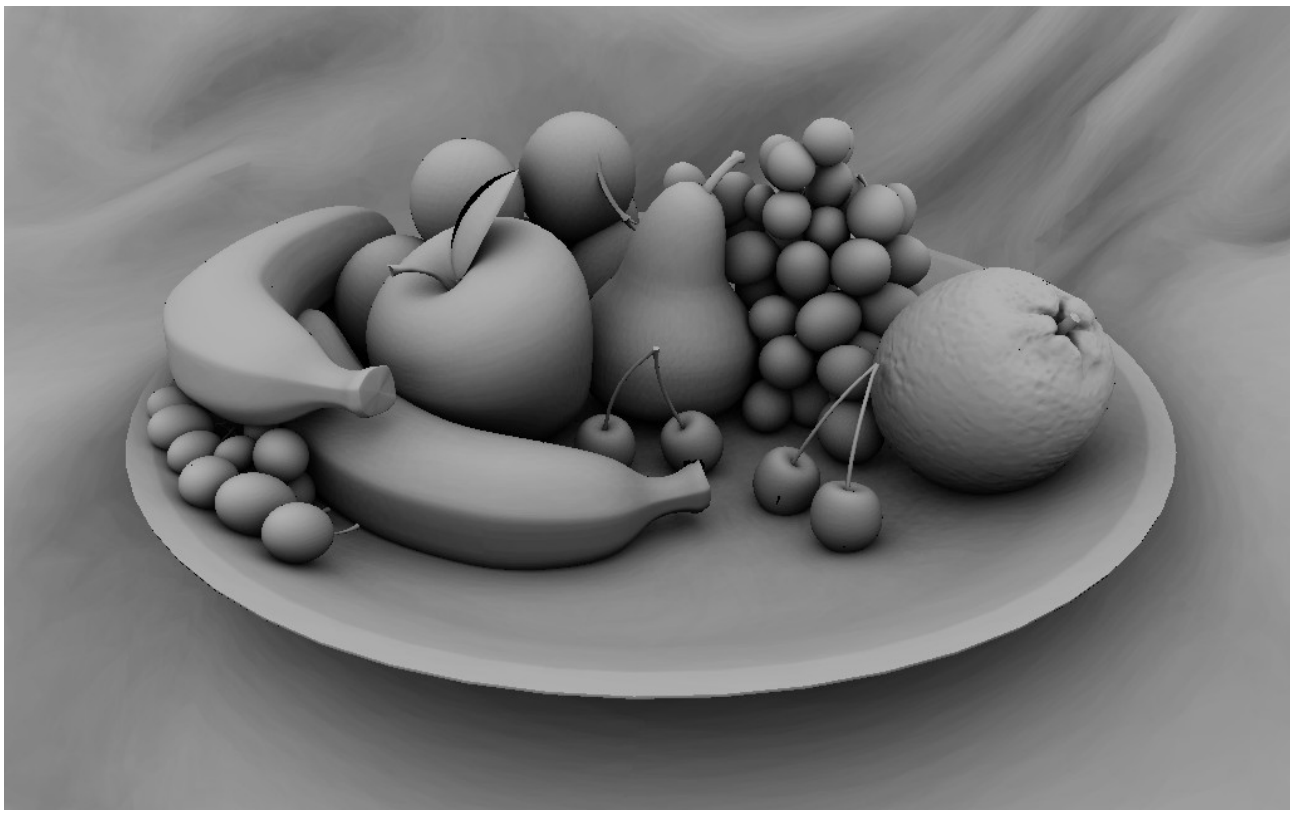


fruit_v2.obj + fruit_v2.txt - **64 directions aléatoires** – Global Ambient

fruit_v2.obj + fruit_v2.txt - **64 directions Spirale de Fibonacci perturbée** – Global Ambient



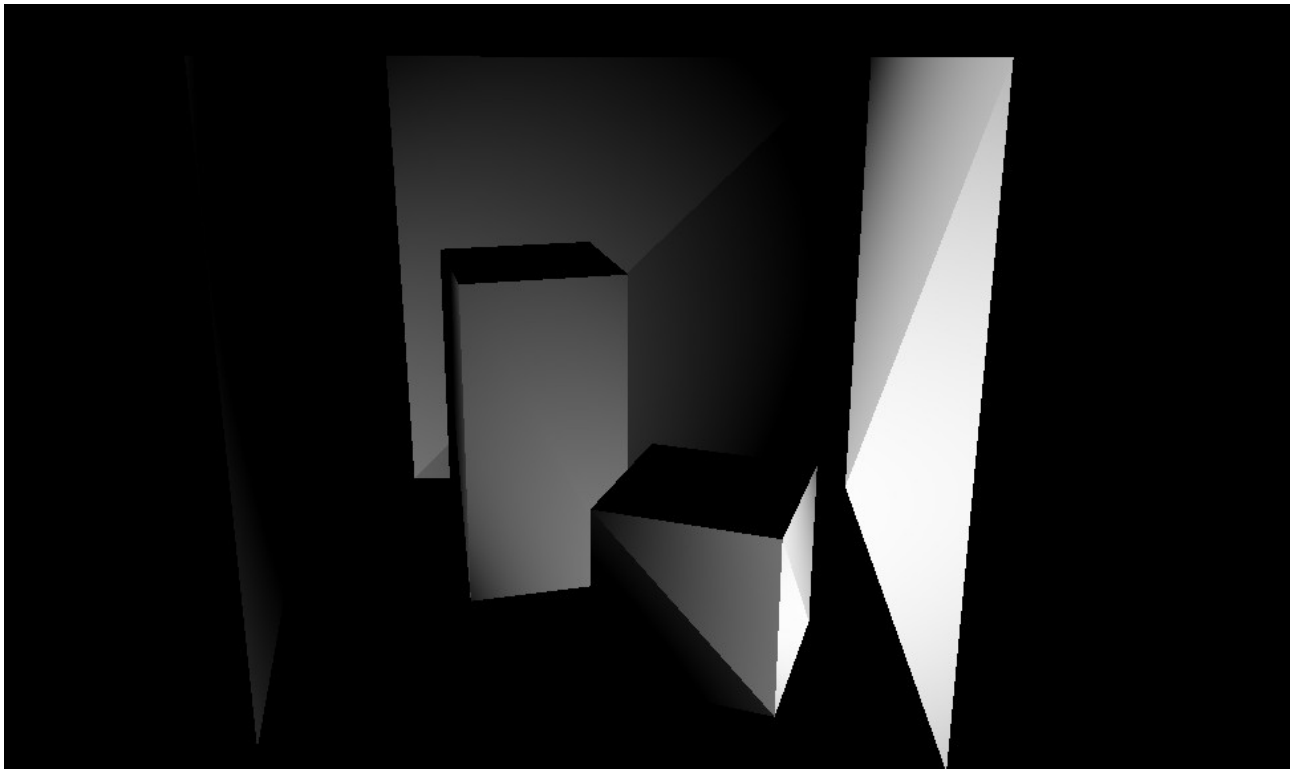
fruit_v2.obj + fruit_v2.txt - **256 directions aléatoires** – Global Ambient



fruit_v2.obj + fruit_v2.txt - **256 directions Spirale de Fibonacci perturbée** – Global Ambient

Partie 4 : Avec des shaders

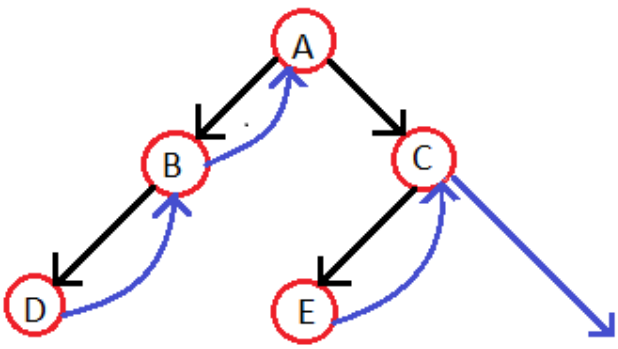
Lighting diffuse for Cornell.obj



Using BVH avec shaders - Not finished yet

Because a shader doesn't support recursion, my solution for this part to traverse nodes in the shaders: *Non recursive Inorder traversal for a Threaded Binary Tree*

- https://en.wikipedia.org/wiki/Threaded_binary_tree



Algorithm

Step-1: For the current node check whether it has a left child which is not there in the visited list. If it has then go to step-2 or else step-3.

Step-2: Put that left child in the list of visited nodes and make it your current node in consideration. Go to step-6

Step-3: Print the node and If node has right child then go to step 4 else go to step 5

Step-4: Make right child as current node.

Step-5:if there is a thread node then make it the current node.

Step-6: if all nodes have been printed then END else go to step 1