TP Noté

Le Nalinec Tibère  
Pedretti Zack

Toussaint Jarod

Préambule :

Pour résoudre ce problème, nous pouvons utiliser la topologie en étoile.

Le service central centralise le calcul global. Il répartie les petits calculs vers les clients et récupère les résultats renvoyés par les clients.

Lorsqu’un nouveau client se connecte, le service central lui envoie un bout du calcul.

Partie « Le tracé de rayon » :

Question 2 :

|  |  |
| --- | --- |
| Taille de l’image | Temps de calcul (ms) |
| 16x16 | 35 |
| 32x32 | 68 |
| 64x64 | 151 |
| 128x128 | 383 |
| 256x256 | 761 |
| 512x512 | 1914 |
| 1024x1024 | 6072 |
| 20 |  |

Question 3 :

Vous pouvez retrouver ci-dessous les lignes qu’on a modifié afin de pouvoir afficher deux parties de l’image :

On calcule d’abord deux images différentes avec scene.compute() et on les affiche avec disp.setImage().

int x0 = 0;

int y0 = 0;

int x1 = largeur/2;

int y1 = hauteur/2;

int l = largeur/2;

int h = hauteur/2;

//Image en bas à gauche

Image image = scene.compute(x0, y0, l, h);

//Image en haut à droite

Image image2 = scene.compute(x1, y1, l, h);

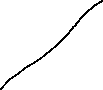
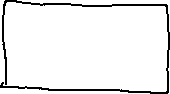
// Affichage de l'image calculée

disp.setImage(image, x0, y0);

disp.setImage(image2, x1, y1);

Partie « Accélérons les choses » :

Schéma :



Service RayTracing

Service RayTracing

Service RayTracing

1. Le processus fixe serait le service central (noté SC sur le schéma) et les processus éphémères seraient les services RayTracing que les clients envoient au service central.
2. Une fois les services RayTracing envoyé au service central, celui-ci leur attribue des « pixels » à calculer sous la forme d’un point de coordonnées x y et la largeur et la hauteur du pixel.

Si l’on veut que les calculs se fassent en parallèle on peut décomposer l’image principale en « pixel » de même largeur et hauteur, par exemple, une image qui sera décomposer en 6x6 pixels, et ensuite le serveur central aura une boucle for qui attribue des pixels pour chaque service RayTracing.  
  
Pour ce faire, on pourrait, par exemple, avoir un int serviceCourant définit de base à 0 dans la boucle for qui selon la valeur de serviceCourant attribue le pixel courant à tel ou tel service (grâce à tabService[serviceCourant] si le tableau de service RayTracing dans le service central s’appelle comme ça), après une attribution, on ferait +1 à cette valeur tant qu’elle n’est pas supérieur au nombre de service (tabService.length).

On peut également mettre un try catch dans le for afin de ne pas arrêter le programme si un service RayTracing est inactif : si le programme catch une exception, on enlèvera le service obsolète du tableau et on ferait -1 à la valeur de i afin de pouvoir attribuer le pixel qui n’a pas été fait au nouveau service RayTracing qui aurait le numéro serviceCourant dans le tableau après suppression.