Projet RMI

Fenard Dorian
Klingler Emma
Drouvot Melody
Rochedreux Hugo

Réponses aux questions :

1. Tester le programme en modifiant ses paramètres

Le programme permet de modifier plusieurs paramètres, dont la taille de l'image. Plus la résolution de l'image à calculer est grande, plus le temps de calcul augmente.

2. Observer le temps d'exécution en fonction de la taille de l'image calculée.

Plus la résolution est grande, plus le temps d'exécution augmente. Avec de petites images, c'est quasiment instantané.

Résolution Image	Temps de calcul
256*256	450ms
512*512	1349ms
1024*1024	4649ms
2048*2048	18731ms

Processus fixe : Classe serveur, qui gère les clients.

Processus mobile : Les clients qui servent de calculateurs.

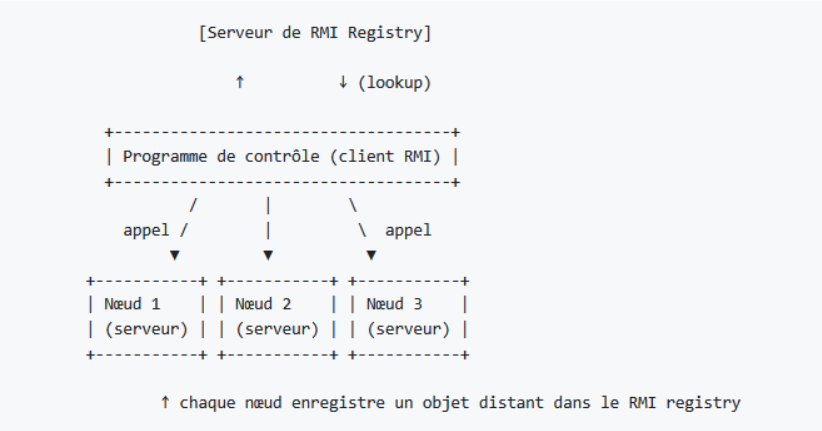
Les types de données échanger : Des morceaux de l'image de la scène qui est calculées. Il y a aussi la liste des clients qui est transférée du serveur vers la classe main.

3. Si on veut que les calculs se fassent en parallèle, que faut-il faire ?

Pour réaliser des calculs en parallèle, il faut utiliser des threads. Il faut faire attention à bien découper les données pour ne pas avoir de conflit en mémoire. Chaque thread travaille sur une partie de l'image dans notre cas.

Travail réalisé

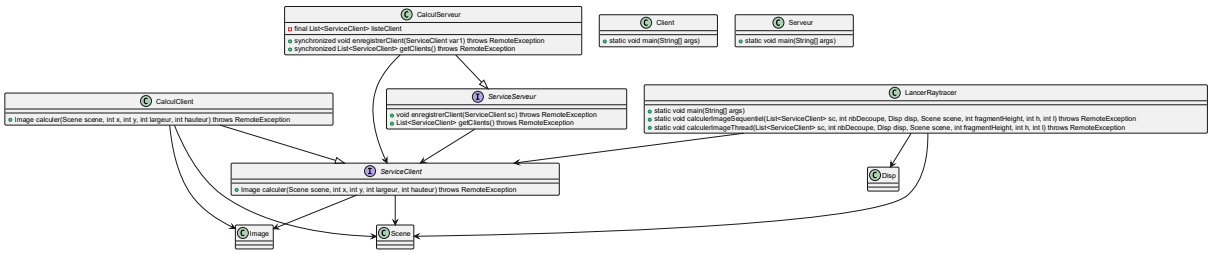
Schéma de l'architecture :



Composant	Équivalent en RMI
Serveur de nœuds	Objet RMI ou composant qui fournit une **liste de nœuds**
Nœuds de calcul	Objets RMI qui implémentent l'interface `Calculateur`
Programme principal	Client RMI qui fait `lookup` sur les nœuds et les appelle

Pour faire des calcules en parallèles il suffit d'utiliser des Thread !

Le diagramme de classe est disponible en plus gros format sur le dépôt git.



Le serveur est lancé en premier, il créer le registre sur le port par défaut (1099) et exporte CalculServeur sous le nom « Calculateur ». Une fois cela fait, les différents Clients sont lancés. Chaque Client récupère le registre, le processus enregistré sous le nom « Calculateur » puis créer un CalculClient et l'enregistre auprès du Calculateur.

Une fois cela fait, le main peut être lancé, il se situe dans la classe LancerRaytracer. Il récupère le registre sur le port par défaut, récupère le processus enregistré sous le nom « Calculateur », une fois récupérer il récupère les clients de celui-ci via la méthode « getClients ». Ensuite il peut se servir des clients pour calculer de deux manières, soit séquentielle soit parallèle. C'est choisi au lancement via un paramètre à mettre à true.

CalculClient implémente l'interface ServiceClient. Celle-ci possède la méthode calculer qui prend en paramètre différent éléments permettant de décrire le morceau de l'image à calculer.

CalculServeur implémente l'interface ServiceServeur. Cette interface possède deux méthodes distinctes. Une pour enregistrer un ServiceClient, en prenant en paramètre ce ServiceClient, et une autre pour récupérer la Liste des ServiceClient.

Code des interfaces :

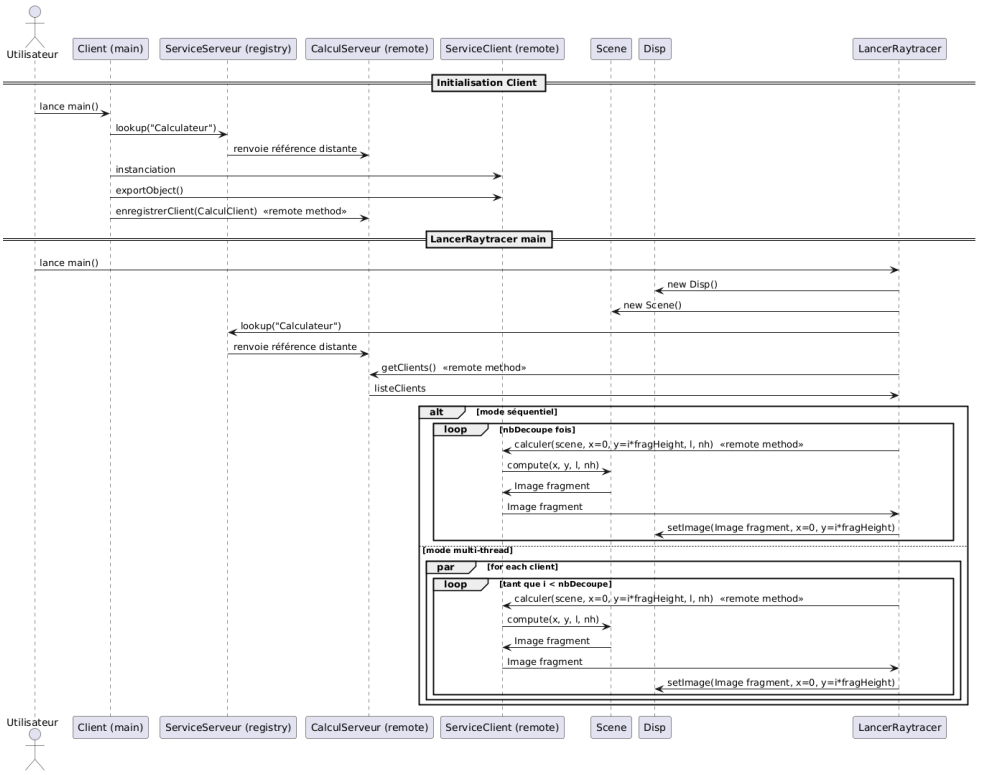
Interface ServiceClient :

```java
import raytracer.Image;
import raytracer.Scene;
import java.rmi.RemoteException;
import java.rmi.Remote;

public interface ServiceClient extends Remote{
    Image calculer(Scene scene, int x, int y, int largeur, int hauteur) throws RemoteException;
}
```

Interface ServiceServeur :

```java
import java.rmi.RemoteException;
import java.rmi.Remote;
import java.util.List;

public interface ServiceServeur extends Remote {
    void enregistrerClient(ServiceClient client) throws RemoteException;
    List<ServiceClient> getClients() throws RemoteException;
}
```

Diagramme de séquence : (Disponible sur le repository git)



La vidéo de présentation est le fichier mp4 nommé « Prog_Répartie.mp4 ».

Comparaison des résultats :

Test réalisé avec 3 clients et les images sont découpés en 100 morceaux.

Résolution	Séquentiel	Parallèle
512*512	1961ms	1413ms
1024*1024	4124ms	5065ms
2048*2048	12518ms	5964ms
4096*4096	46457ms	18472ms

On peut constater qu'avec une résolution assez basse, la différence entre le mode de calcule séquentiel et parallèle est peu conséquent. Cependant sur de grande image comme 4096 * 4096, la différence est flagrante, le calcul séquentiel est plus de 2x plus lent que le calcul parallèle. Ces statistiques varient d'une machine à l'autre.