1 - Calcul vectoriel

Objectif

Nous allons maintenant commencer le travail sur le Raytracer en lui même !

Pour le moment, il n'est pas encore question de générer des images, mais d'implémenter quelques briques de base dont nous aurons besoin par la suite.

Spécifications

La plupart des opérations du lanceur de rayons se font sur des éléments de dimension 3.

Opérations à réaliser

Notez bien les symboles utilisés pour chaque opération, il seront réemployés tout au long du projet.

Attention :

- à la différence entre le produit scalaire . et le produit vectoriel $\times$
- le produit vectoriel n'est pas commutatif ($a \times b \neq b \times a$)

Opération	Formule
Addition	$(x_1, y_1, z_1) + (x_2, y_2, z_2) = (x_1 + x_2, y_1 + y_2, z_1 + z_2)$
Soustraction	$(x_1, y_1, z_1) - (x_2, y_2, z_2) = (x_1 - x_2, y_1 - y_2, z_1 - z_2)$
Multiplication par un scalaire	$d * (x_1, y_1, z_1) = (x_1, y_1, x_1) * d = (d * x_1, d * y_1, d * z_1)$
Produit scalaire	$(x_1, y_1, z_1).(x_2, y_2, z_2) = x_1 * x_2 + y_1 * y_2 + z_1 * z_2$
Produit vectoriel	$(x_1, y_1, z_1) \times (x_2, y_2, z_2) = (y_1 * z_2 - z_1 * y_2, z_1 * x_2 - x_1 * z_2, x_1 * y_2 - y_1 * x_2)$
Produit de Schur	$(x_1, y_1, z_1) * (x_2, y_2, z_2) = (x_1 * x_2, y_1 * y_2, z_1 * z_2)$
Longueur	$\|\|(x_1, y_1, z_1)\|\| = \sqrt{x_1 * x_1 + y_1 * y_1 + z_1 * z_1}$
Normalisation	$norm((x_1, y_1, z_1)) = (\frac{1}{\|\|(x_1, y_1, z_1)\|\|}) * (x_1, y_1, z_1)$

Opérations entre types différents

- la **soustraction de deux points** donne un vecteur
- l'**addition d'un point et d'un vecteur** donne un point.

Pertinence des opérations

D'un point de vue conception, on essayera de prendre en compte que toutes les opérations n'ont pas forcément de sens dans tous les contextes. Le tableau suivant récapitule les opérations possibles par type de données manipulées.

Opération	Point	Vecteur (directions, normales)	Couleur
Addition		x	x
Soustraction	x	x	
Multiplication par un scalaire	x	x	x
Produit scalaire		x	
Produit vectoriel		x	
Produit de Schur			x
Longueur		x	
Normalisation		x	

Couleurs

- Une couleur est composée de trois composantes rouge, vert et bleu
  - Chaque composante sera une valeur flottante comprise entre 0 et 1
  - Si elle dépasse 1 il faudra avoir 1 comme maximum

Aide

0 - Création du projet

Créez un projet Maven ou a minima un package différent (de ImageComparator) pour le Ray Tracer.

- Prévoyez d'ajouter régulièrement des tests unitaires et d'intégration
- Vérifiez à tout moment que votre code est lisible, maintenable, que les variables, classes et méthodes sont bien nommées
- Ajoutez régulièrement de la documentation

1 - Classes à créér

- Vous pouvez créer une première classe `AbstractVec3` qui servira de base aux autres classes. Vous y mettrez les opérations en 3 dimensions pertinentes pour toutes les autres.
  - Les composantes seront des `double`
- Vous aurez besoin de 3 classes
  - `Vector` : représente un vecteur (vous en aurez besoin pour les directions et les normales)
  - `Point` : représente un point
  - `Color` : attention, n'utilisez pas de classe proposée par une librairie externe

2 - Méthodes à créer

- Chaque opération citée au dessus aura sa méthode dans les classes pertinentes
- Méthodes `equals`
- Un constructeur par défaut pour les couleurs, qui donnera du noir (tout à 0)

3 - Comparaison avec des double

Attention, les comparaisons avec les `double` sont peu précises.

Voici comment gérer les comparaisons avec des nombres flottants : https://www.javaspring.net/blog/comparing-doubles-java/

4 - Tests unitaires

A ce stade, difficile de savoir si vos opérations marchent correctement. Elles sont pourtant essentielles pour la bonne réussite du projet. Faites un jeu de tests unitaires à l'aide **JUnit** pour les valider et être sûrs qu'elles aient le fonctionnement attendu.

N'hésitez pas à maintenir ces tests et en ajouter au fil de l'évolution du projet.

5 - Conversion de couleurs

Pour passer de couleurs sous forme flottante 0-1 à RGB (lors de l'écriture d'images par exemple), ajoutez la méthode suivante à la classe `Color` :

```
public int toRGB(){
    int red = (int) Math.round(r * 255);
    int green = (int) Math.round(g * 255);
    int blue = (int) Math.round(b * 255);

    return (
        ((red & 0xff) << 16)
        + ((green & 0xff) << 8)
        + (blue & 0xff));
}
```