1. Justifie bien que les diffèrent effets sont visibles et notre fichier de test sur les vecteurs
2. Une fonctionnalité que nous souhaitions ajouter est de gérer la génération d’image de maniéré ‘propre ‘ au lieu de tout mettre dans le main. Nous avons donc créé une classe ObjectScenario qui permet de regrouper l’ajout des objets, le calcul de l’image et son enregistrement. Cette classe pourra par exemple être utilisé pour génère plusieurs images à la suite et de rendre le code plus lisible. Ça principale utilité cependant est de grandement faciliter l’utilisation du code. Quelqu’un souhaitant s’interfacer sur notre code n’aura potentiellement que cette classe à utiliser pour générer des images.

Dans notre ‘simple’ cas pour générer plusieurs images, nous créons une classe ‘ScenarioPersoX’ héritant de ObjectScenario. Nous redéfinissons son constructeur afin d’ajouter tous les objets que nous souhaitons à l’instanciation de cette classe dérivée. Nous appelons ensuite la méthode toImage qui permet de calculer l’image liée au scenario et de l’enregistrer.

Le calcul d’un scenario personnalisé est de la forme :

ScenarioPerso(int largeur, int hauteur).toImage(char[] nom, bool afficherTexte, int profondeur);

Une autre fonctionnalité qui nous semble importante en comparant les images que nous avons générés et celle que nous pouvons obtenir est l’implémentation d’une lumière qui ‘adoucisse’ les ombres. Il suffit pour réaliser cela que la lumière vienne de plusieurs ‘points’ situés relativement proche les uns des autres.

Nous avons donc choisi de créer une classe « Lumiere\_Smooth » héritant de « Lumiere\_Ponctuelle ». Nous ajouté 2 attribut dans cette nouvelle classe : un flottant représentant la « largeur » de la lumière et un entier proportionnel à la qualité du rendu.

En pratique cette Lumiere\_Smooth se comporte comme un « cube » de lumière ponctuelle. L’ entier nbLums représente le nombre de lumière par face du cube. Au total 3\*\*nbLums lumière ponctuelle sont donc générée lors du calcul. En pratique, nbLums à 3 ou 4 produit en rendu suffisamment réaliste. Augmenter nbLums est très couteux en temps de calcul, il faut donc faire attention à ce paramètre.

Il aurait été possible de faire cette classe en disposant les lumières en sphère autour du centre pour un rendu plus réalise, ou en carré afin de diminuer le temps de calcul pour un résultat relativement similaire.