MOUTON Clara – PHILIPPOT Marine – CLAIRAIS Aurélien
Groupe 1

PROJET D'INFORMATIQUE

ANNEE 2012 – 2013

EBLOUISSEMENT DES CONDUCTEURS DÛ A LA TOUR INCITY

Intervenant : Raphaël LABAYRADE

Introduction

La tour Incity sera la tour la plus haute de Lyon. Alors que la tour LCL s'élève à 165m et la tour Oxygène à 115m, la tour Incity atteindra, elle, la hauteur de 170m sans antenne et 200m en comptant l'antenne. Cette tour sera dans quartier de la Part-Dieu à la croisée du Cours Lafayette et de la Rue Garibaldi. Sa surface sera majoritairement composée de verre réfléchissant.

En conséquence, on peut se demander si la réflexion du ciel sur la tour n'est pas un danger pour les automobilistes parcourant les trois rues importantes que sont : le Cours Lafayette, la Rue Garibaldi, Rue de Bonnel. Afin d'évaluer les risques, le projet d'informatique nous propose de simuler la réflexion du ciel sur la tour Incity en prenant appui sur les scripts établit en TD.

SOMMAIRE

I. ANALYSE DU PROBLEME ET DES DIFFICULTES

Le projet propose comme objectif de quantifier l'impact de la tour Incity à venir sur la sécurité routière, en matière d'éblouissement des conducteurs. Ce problème est très complexe. Tout d'abord quant au choix de la représentation de l' « impact ». En effet, comment pouvons-nous nous représenter l'éblouissement d'un conducteur ? Nous pouvons nous identifier à lui et construire une image « vue du conducteur » ou plusieurs images afin de monter une vidéo et de suivre l'éblouissement du conducteur tout au long de sa progression sur l'une des rues aux environs de la tour. L'impact peut être aussi représenté par une cartographie dans laquelle les postions dans l'espace ou le conducteur est ébloui seraient représentées. Quoiqu'il en soit, le bon déroulement de la simulation nécessite la modélisation d'un système optique (comme une caméra).

La deuxième difficulté est de représenter la scène. Afin d'avoir la représentation la plus réaliste possible, on doit représenter la scène en 3 dimensions. Là aussi la complexité de ce problème fait que nous avons dû le simplifier grandement. La vraie question ici est : Comment modéliser la tour et les bâtiments environnants le plus efficacement possible ? On pourrait modéliser les surfaces en rejoignant simplement les sommets, ou choisir une géométrie plus avantageuse.

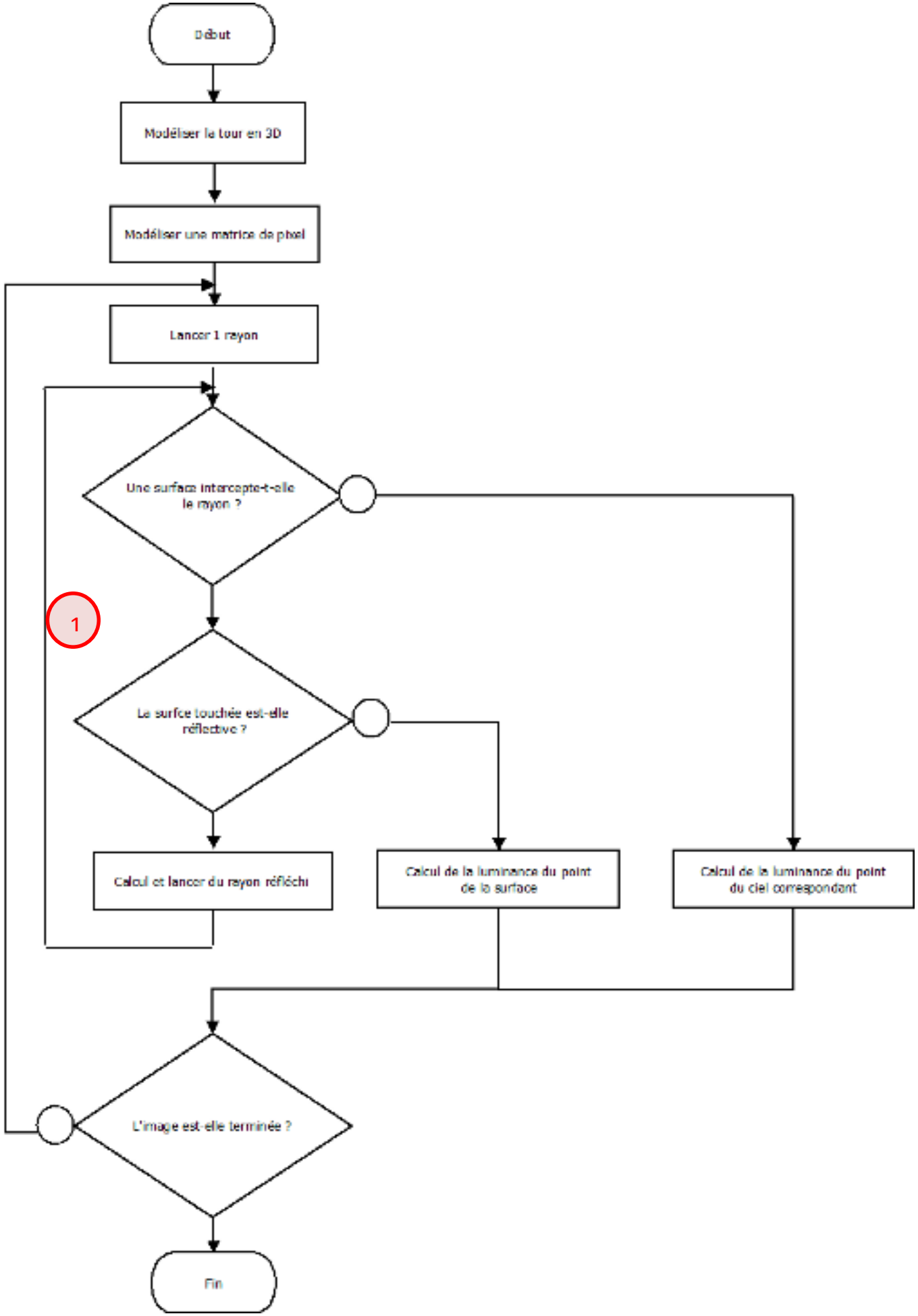
L'éblouissement du conducteur fait intervenir le thème de la luminosité des points de l'image que son œil lui donne. Pour cela, nous pouvions prévoir que ce que nous avions fait lors du TD allait intervenir. Une des difficultés est de quantifier l'éblouissement, par exemple par un indice d'éblouissement.

De là, des objectifs ont été fixés, séparés en objectifs principaux et objectifs secondaires. Les objectifs principaux sont la représentation de la tour en 3 dimensions et la simulation de la réflexion du ciel sur la tour. Les objectifs secondaires développés sont la prise en compte de surfaces non réflexives notamment pour la mise en place d'autres bâtiments dans la scène, la simulation des inter-réflexions entre les différentes surfaces réflexives mise en jeu, le calcul d'un indice d'éblouissement, la cartographie des risques, la simulation des ombres portées et la mise en place de couleurs réalistes dans l'image.

II. PRESENTATION DES SOLUTIONS RETENUES

A. Algorithme général du projet

Voici le développement schématique du projet :


Début
Modéliser la tour en 3D
Modéliser une matrice de pixel
Lancer 1 rayon
Une surface intercepte-t-elle le rayon ?
1
La surface touchée est-elle réflective ?
Calcul et lancer du rayon réfléchi
Calcul de la luminance du point de la surface
Calcul de la luminance du point du ciel correspondant
L'image est-elle terminée ?
Fin

B. Représenter la tour Incity en 3D

Nous allons débuter par les objectifs principaux qui sont au nombre de deux. Le premier est celui traité dans cette partie, à savoir la représentation en 3D de la tour Incity sans prendre en compte les étages. Le deuxième étant de simuler la réflexion spéculaire du ciel sur la tour vitrée.

L'objectif de cette première partie est de modéliser en 3 dimensions la tour Incity. Bien sûr, nous ne pouvons pas modéliser très fidèlement la tour Incity dans toute sa complexité. En conséquence, il fallait simplifier ce problème en le discrétisant. Du coup, la tour Incity a été divisée en un ensemble fini de surfaces planes. Il nous faut donc prendre en considération tous les sommets de la tour Incity et notamment les informations de hauteur de la tour ont été assez utiles.

Hauteur du dernier étage	154m
Hauteur sans l'antenne	170m
Hauteur avec l'antenne	200m

Ensuite, pour des raisons pratiques à la bonne continuité du projet, il nous fallait des surfaces discrétisées planes. Ainsi, même s'il est vrai que la majorité des côtés de la tour sont sous forme de quadrilatères, les sommets ont été entrés à la main et les chances pour qu'une telle parametrisation corresponde exactement au cas parfait de 4 points coplanaires sont infiniment minces. D'où comme dans l'espace, trois points sont forcément coplanaires, nous avons discrétisé les surfaces en triangles et donc même si les points collent seulement à 99% au réel, cela n'entravera pas le calcul des luminances.

En pratique, nous avons rentré les 8 points de la tour Incity dans une matrice *Mat_Points* où chaque ligne correspond aux coordonnées d'un point de la tour. Les points de la tour ainsi indexés nous ont permis de rentrer aussi à la main les triangles composant la tour de la manière suivante dans une matrice *Mat_Triangles*, par ligne :

Index Premier Point	Index Deuxième Point	Index Troisième Point

L'ordre des points est important, les points ont été entrés de manière à avoir la normale au triangle sortante du bâtiment. Pour tracer les triangles, et donc modéliser la tour concrètement en 3D, nous avons créé la fonction *Tracer_Triangles* dépendant donc de *Mat_Points et Mat_Triangles.* Le fonctionnement de cette fonction est assez simple, il suffit de joindre tous les points constitutifs de tous les triangles grâce à la fonction prédéfinie dans MATLAB, *plot3.*

(Voir figure 1.1)

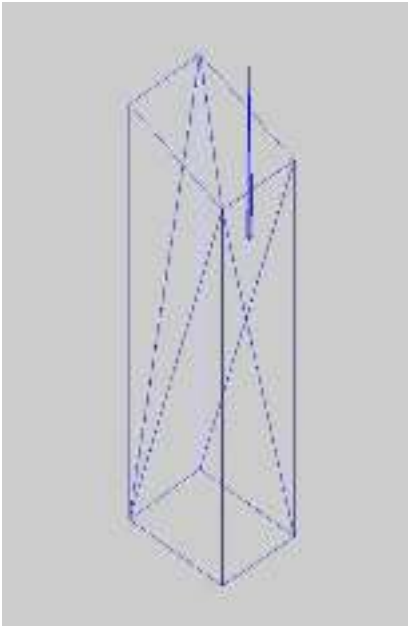


Figure 1.1 : Représentation de la tour en 3D sans les normales

La dernière partie de la modélisation de la tour était de tracer les normales sortantes à tous les triangles de la tour. Ceci a été exécuté dans la fonction *Tracer_Normales* qui en plus de tracer les normales va aussi les rentrés dans une matrice *Mat_Normales* contenant la liste des normales de la scène indexées selon les triangles auquel elles correspondent. (La ligne *i* représente les coordonnées de la normale au triangle n°*i*). Afin de les tracer et qu'elles se voient sur le graphe 3D, les normales ont été normalisée et multiplier par un coefficient arbitraire *30*. Pour les tracer, nous avons utilisé la fonction *quiver3* qui permet de tracer des vecteurs en 3D en prenant en compte l'origine du vecteur et les coordonnées de celui-ci. Pour l'origine des normales lors du tracé, nous avons choisi le centre de gravité du triangle correspondant. Le résultat final de cette première partie est en figure 1.2.

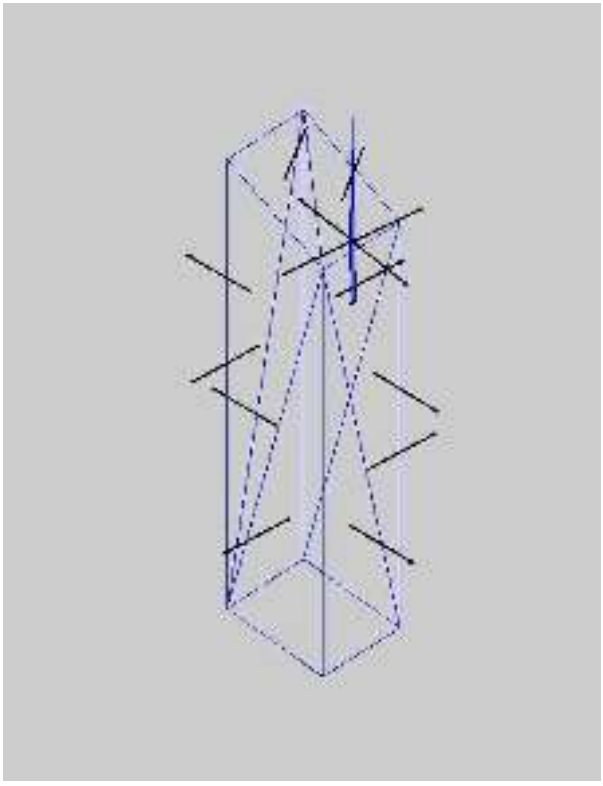


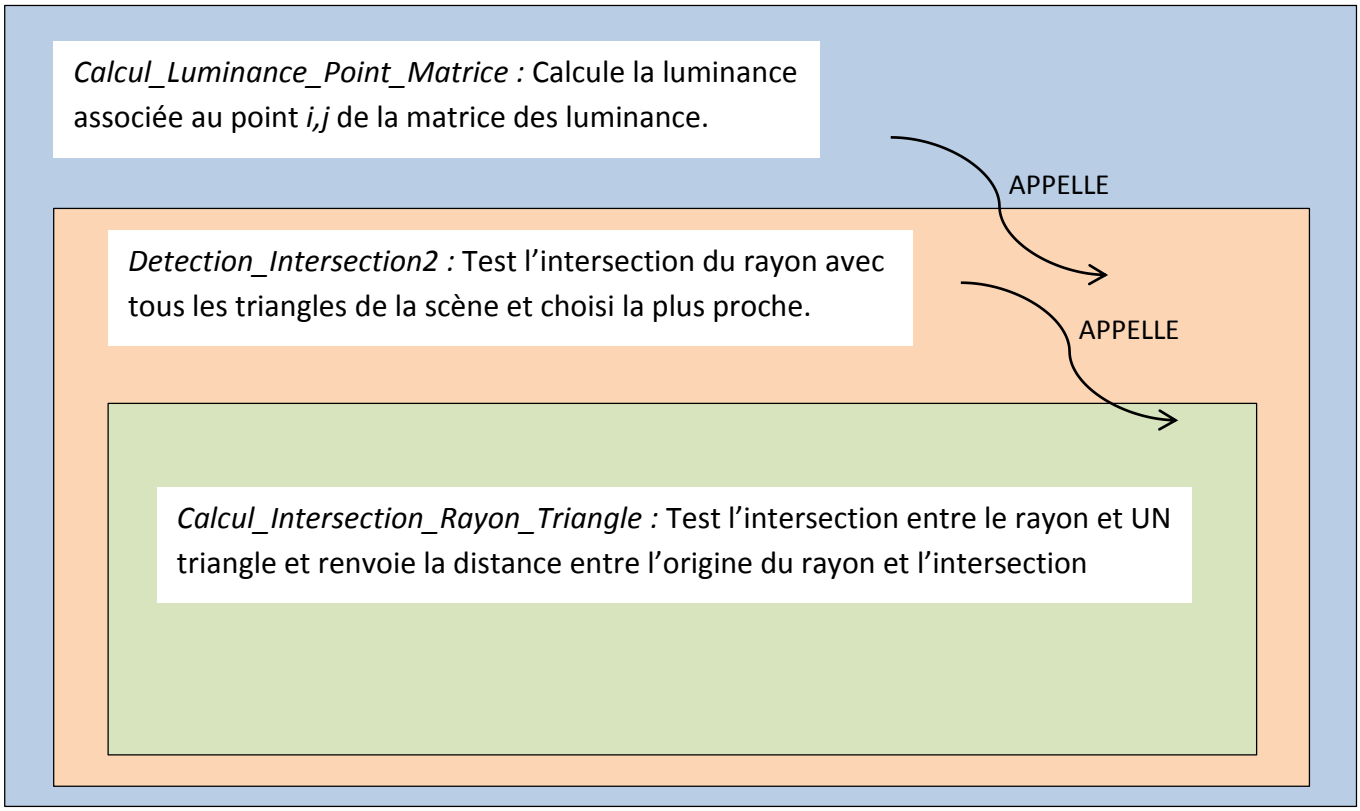
Figure 1.2 : Représentation de la tour en 3D Avec les normales

C. Simuler la réflexion spéculaire du ciel sur la tour vitrée

L'objectif de cette partie est de créer une image représentant la tour Incity et le ciel se réfléchissant sur celle-ci. Tout ceci s'est fait en plusieurs étapes.

La première étape fut de créer l'image physiquement. C'est l'intérêt de l'ensemble de scripts et fonctions « camera » déposé sur le Bureau Virtuel. Ce *package* permet de créer l'image comme si nous la regardions à travers un appareil photo numérique. On y définit le nombre de pixels, la focale, l'angle sous lequel on voit la scène et la position de l'observateur. En contrepartie, ce *package* nous donne une matrice de taille *dimX par dimY* que nous pouvons remplir avec les valeurs de luminance que nous trouvons grâce au lancer de rayon.

Ensuite, cette dernière matrice nous donne une série de *dimX*dimY* points dans l'espace par lesquels nous allons faire des lancer de rayons. Un lancer de rayon c'est le fait de projeter dans l'espace un rayon issue du centre optique et passant par le centre du pixel *i,j* considéré. Ce procédé nous donne le vecteur directeur du rayon ainsi défini. Maintenant que nous avons notre rayon, il ne nous reste plus qu'à savoir où celui-ci nous mène. Pour cela, nous avons crée une fonction *Calcul_Luminance_Point_Matrice* qui associe au point *i,j* de la matrice des Luminance la valeur de la luminance du rayon passant par le centre du pixel associé au point *i,j*. Cette dernière fonction appelle aussi plusieurs fonctions.



Maintenant que l'on possède notre intersection, ou pas, deux cas s'offre à nous. Dans le premier cas, les fonctions n'ont pas détecté d'intersection et alors le rayon est libre d'aller jusqu'au ciel. Dans ce cas-là, nous récupérons l'azimut et la hauteur du vecteur directeur du rayon grâce à la fonction *Conversion_Cart_Sph* et on associe au point *i,j* de la matrice des luminances, la luminance du point du ciel correspondant à cet azimut et cette hauteur. Dans le deuxième cas, il y a intersection. Dans ce cas, on construit le vecteur directeur du rayon réfléchi grâce à la formule donnée. On récupère l'azimut et la hauteur du vecteur directeur du rayon réfléchi grâce à la fonction *Conversion_Cart_Sph* et on associe au point *i,j* de la matrice des luminances, la luminance du ciel correspondant à cet azimut et cette hauteur. Les valeurs de luminances sont données par la fonction *Calcul_Luminance_Point_CALCUL* qui était définie dans le TD.

Maintenant que nous avons notre matrice des luminances absolues, il faut la mettre en image. Il faut savoir que dans MATLAB, une image peut être défini comme une matrice de taille *dimX,dimY,3* où les trois couches correspondent aux couches de rouge, de vert, et de bleu (RGB). Le problème est que les valeurs des codes couleurs RGB sont des scalaires entre 0 et 1. Il fallait donc diviser toute la matrice des luminances par le maximum de cette matrice. En associant à chaque couche les valeurs de la matrice des luminances normée, par la fonction *image* on obtient une image en nuance de gris comme sur le rendu de la figure 2.2.

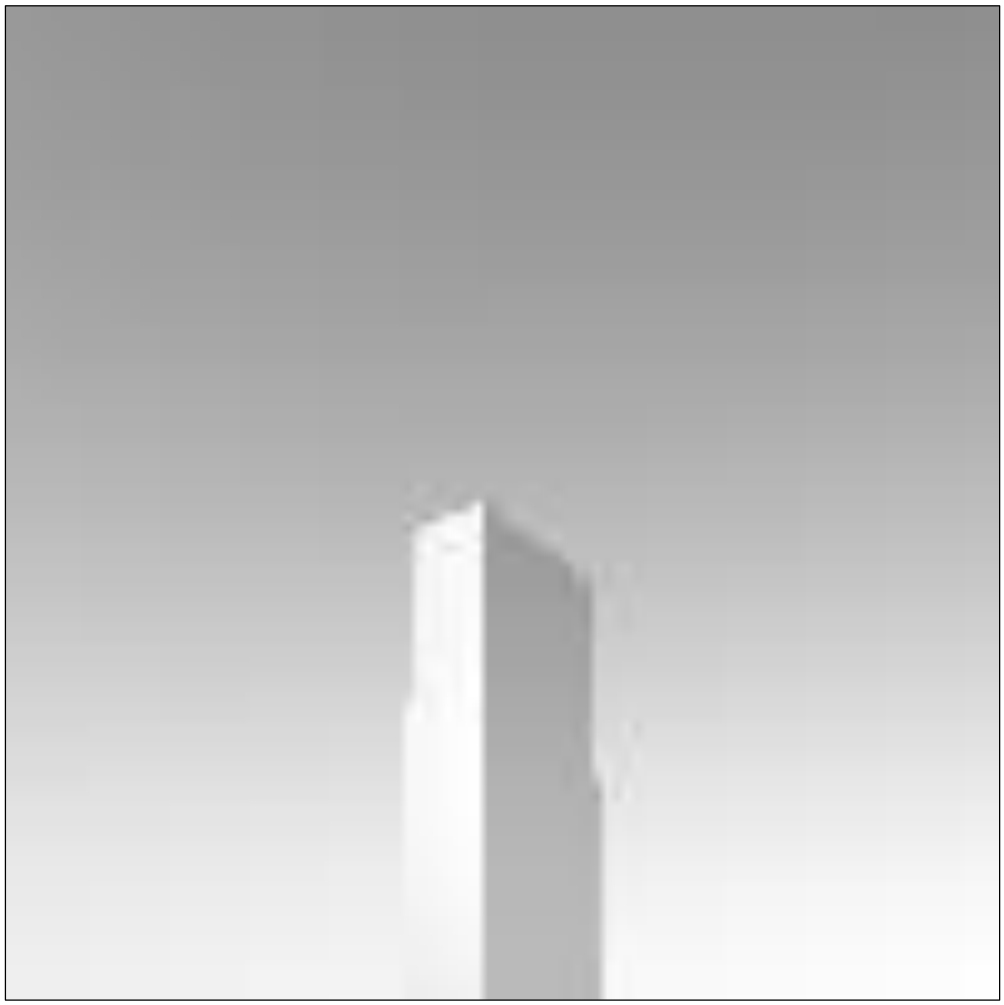


Figure 2.2 : Rendu Final de la tour Incity avec le ciel se réfléchissant

D. Prise en compte des surfaces diffusantes

Lorsque le rayon projeté atteint une surface diffusante, il n'y a aucune raison de créer le rayon réfléchi.

Dans un premier temps, nous avons créé une 4ème colonne dans *Mat_Triangles* représentant le type de surface dont le triangle est issu. On a adopté le code suivant : 1 pour une surface réflective et 2 pour une surface diffusante. Mais aussi, une 5ème colonne représentant un coefficient A entre 0 et 1 tel que la formule de la luminance du point de la surface soit égale au cosinus de l'angle entre le rayon solaire et la normale à la surface multiplié par ce coefficient A.

Cette valeur de luminance remplace le calcul de luminance d'un point du ciel dans ce cas précis.

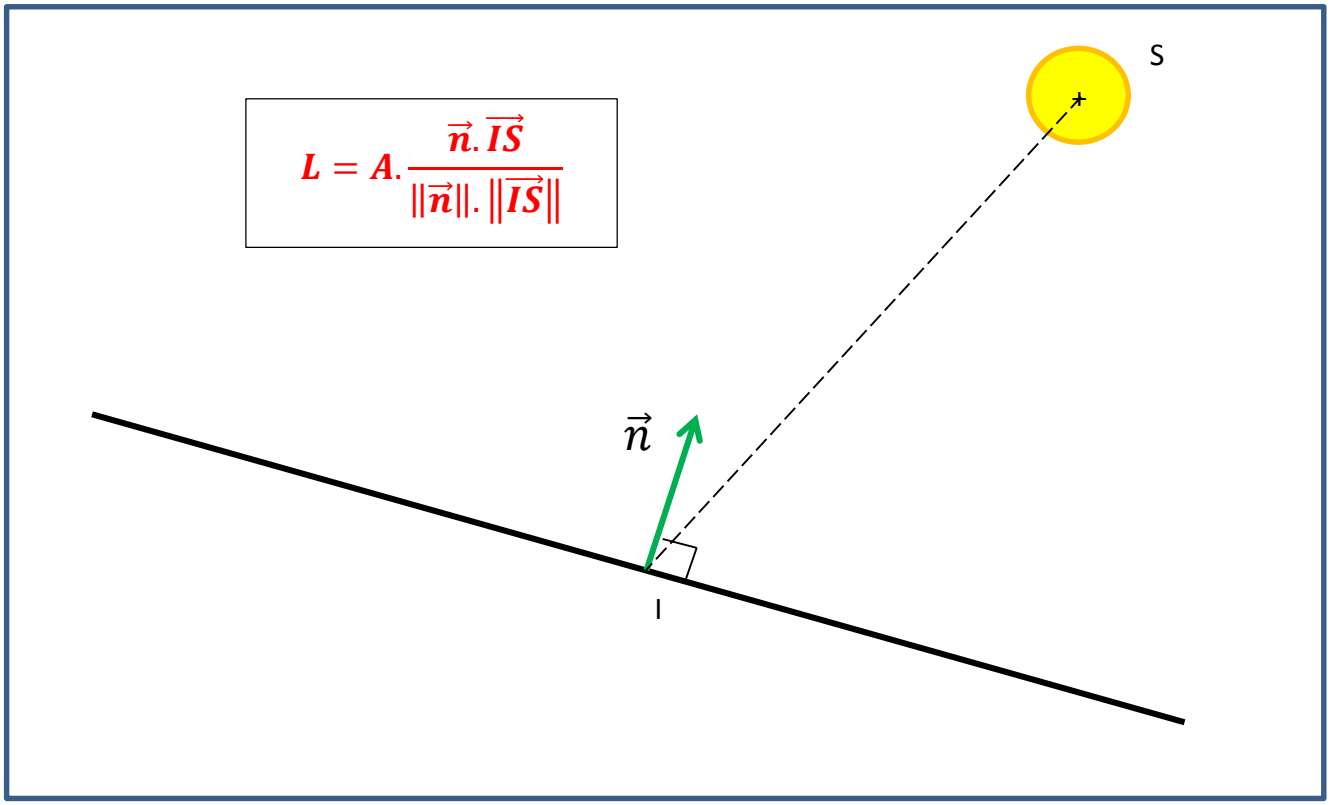


Figure 2.3 : Schéma de calcul de la luminance d'un point d'une surface diffusante.

E. Simuler les inter-réflexions entre les différentes surfaces réflectives de la scène.

Lors d'une réflexion spéculaire, au lieu de calculer l'azimut et le zénith du rayon réfléchi, on lance ce dernier à partir du point d'intersection trouvé qui devient le nouveau centre optique pour CE calcul. Ensuite, on agit comme pour les rayons initiaux. On réitère le procédé autant de fois qu'il faut pour que le rayon touche une surface diffusante ou aucune surface.

A chaque itération le coefficient entre 0 et 1 diminuant la luminance du point de l'image considéré est mis à jour de la manière suivante : $C_n = coefficientdelasurface.C_{n-1}$.

(Cette étape est donnée dans l'algorigramme par le 1)

Afin de vérifier que ce principe fonctionne, nous avons testé cela avec l'utilisateur entre deux miroirs fictifs géants comme atteste la figure 2.4.

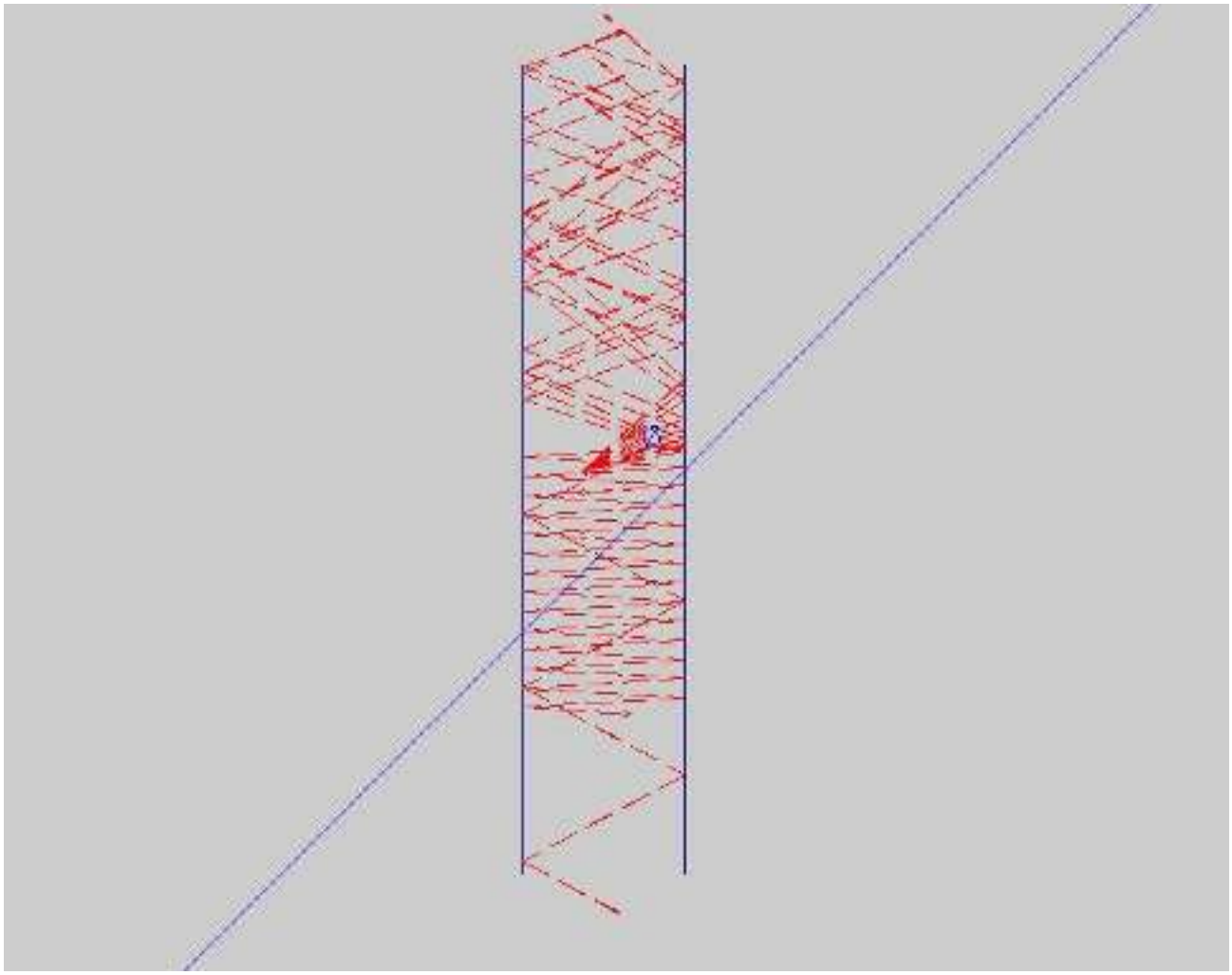


Figure 2.4 : Illustration des Inter-Réflexions

F. Simuler les ombres portées

L'ombre portée intervient lorsque le rayon issu du centre optique atteint une surface diffusante. Comme il n'y a pas de rayon réfléchi on calcul donc directement la luminance du point d'intersection. Cependant, il est intéressant de voir si ce point est exposé au soleil, ou si une autre surface lui obstrue le soleil. Dans ce dernier cas, le point est « dans l'ombre ».

En pratique, il faut détecter si le rayon issu du soleil et atteignant le point I, intersection de la surface et du rayon issu de l'œil, atteint une autre surface. S'il y a intersection avec une autre surface, il faut voir si celle-ci se trouve devant le point I, en quel cas le point I est dans l'ombre, ou derrière le point I, dans ce cas, le point I est exposé. La figure 2.4 explique ce procédé.

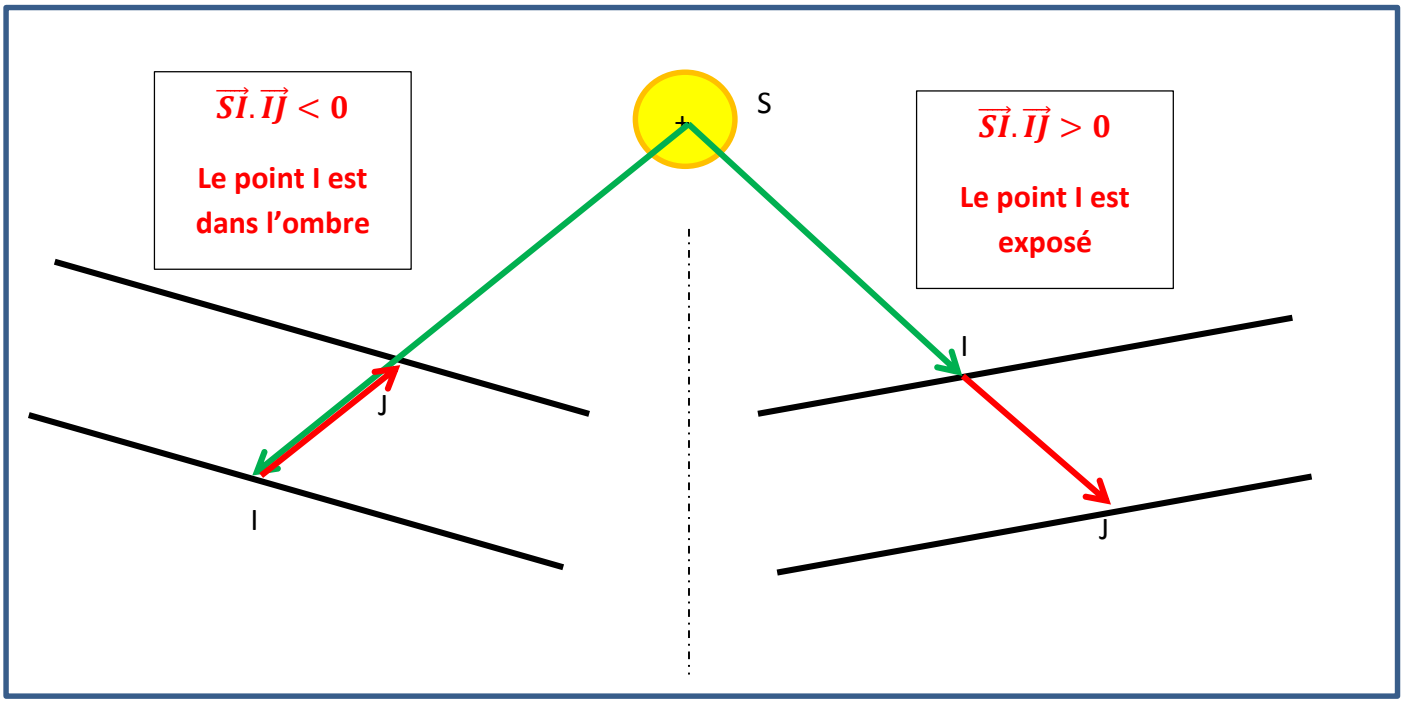


Figure 2.5 : Principe de l'ombre portée

Pour vérifier cette partie, nous avons créé une image mettant en scène des ombres portées.

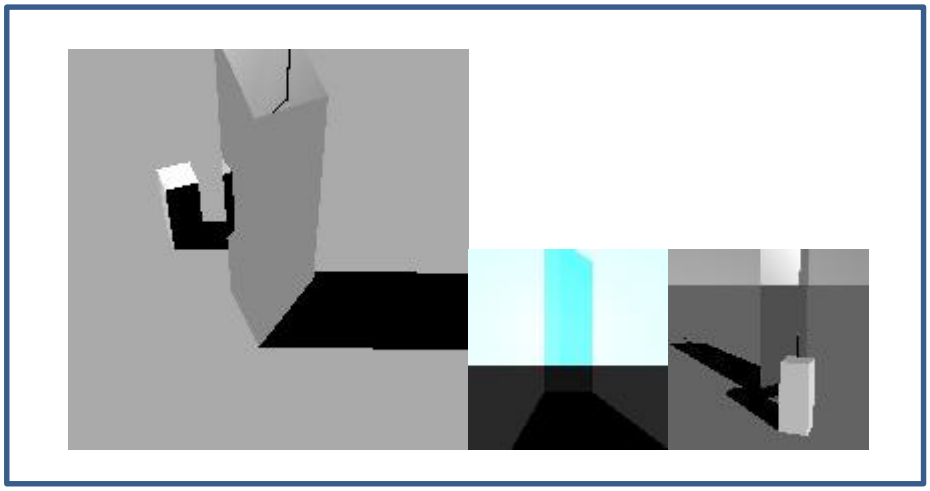


Figure 2.6 : Assortiment d'images mettant en scène l'ombre portée

G. Colorer l'image de manière réaliste

Afin d'avoir des couleurs réalistes, il a fallu différencier les points de l'image représentant le ciel et les points représentant les surfaces bétonnées. Ainsi, la fonction *Calcul_Luminance_Point_Matrice* renvoie désormais en plus de la luminance du point, le chiffre 1 si le pixel considéré correspond à un point du ciel et 2 sinon.

Ensuite, il a fallu créer 2 échelles de couleurs différentes. Ces deux échelles sont représentées par les matrices *Couleurs_du_CIEL* et *Couleurs_du_BETON* enregistrées dans le dossier du code. Elles ont été créées de la manière suivante :

**On crée un vecteur allant de 0 à 1 par pas de 0.1*
**On remplit les colonnes correspondant aux valeurs de luminances pour lesquelles on impose la couleur*
**On linéarise les trois dernières lignes (R – G – B) : la fonction Linearisation a été créée pour l'occasion.*

On obtient une matrice de la forme suivante, par colonne :

Valeur normée de Luminance
R
G
B

Figure 2.7 : Echelle de Couleurs du Ciel Type 1

Figure 2.8 : Echelle de Couleurs du Béton Type 2

Ensuite, il faut assigner aux différentes couches de la matrice *IMAGE* la valeur en rouge, en vert et en bleu correspondant à la luminance normée du coefficient *i,j* de la matrice *Matrice_Luminance_Normée.* Pour vérifier, il fallait créer une image dans laquelle on observe une partie du ciel et une partie du sol.

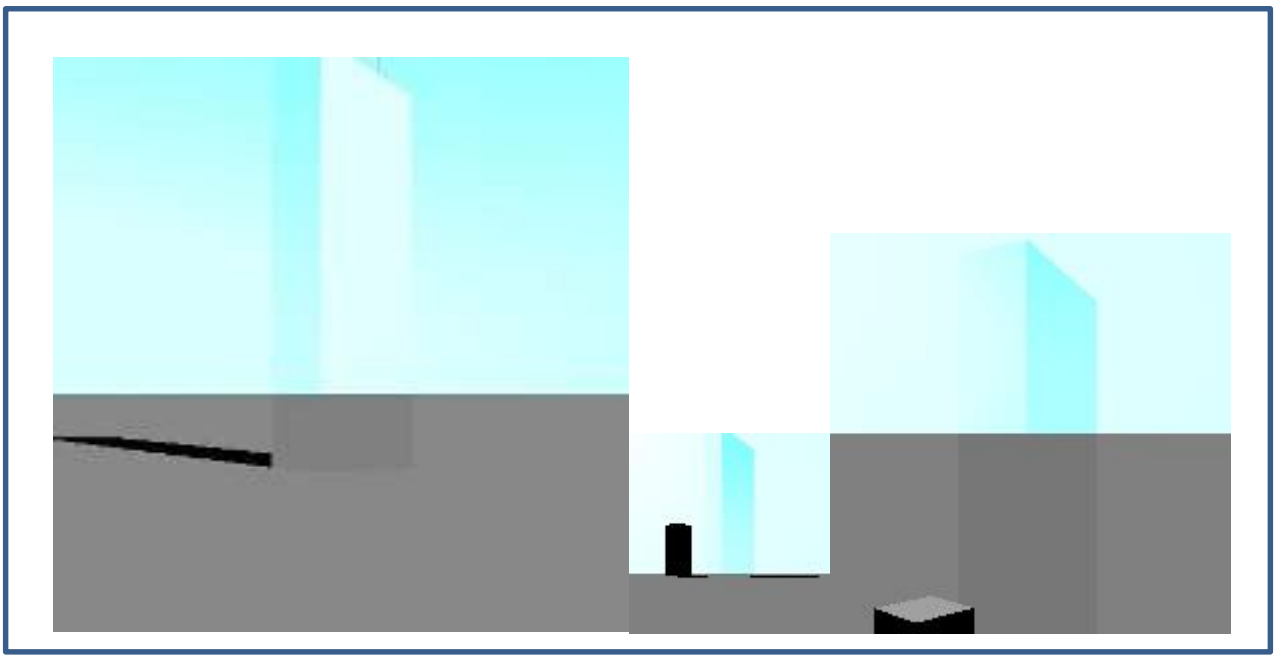


Figure 2.9 : Images mettant en scène des couleurs réalistes.

H. Calculer un indice d'éblouissement

Dans cette partie, nous avons procédé à un simple calcul. L'indice d'éblouissement représente la proportion de pixel possédant une luminance supérieure à 5000 cd/m² dans la matrice des luminances.

La formule est : $\frac{Nombre\ de\ pixels>5000}{dimX*dimY}$

I. Etablir une cartographie des risques

Pour établir une cartographie des risques, on parcourt les 3 rues environnant la tour Incity : la rue Garibaldi, le cours Lafayette, la rue de Bonnel, dans le sens permettant d'être toujours face à la tour. Les positions ont un intervalle de 50m. Pour chaque emplacement, on crée une image en 20x20 (pour que les calculs soit rapides) et on calcule l'indice d'éblouissement. Si l'indice est inférieur à 0.5 on trace un « + » vert à l'emplacement du conducteur, si l'indice est supérieur à 0.5 c'est un « + » rouge. Enfin, on trace avec des « * » bleus les points d'altitude nulles pour représenter les bâtiments.

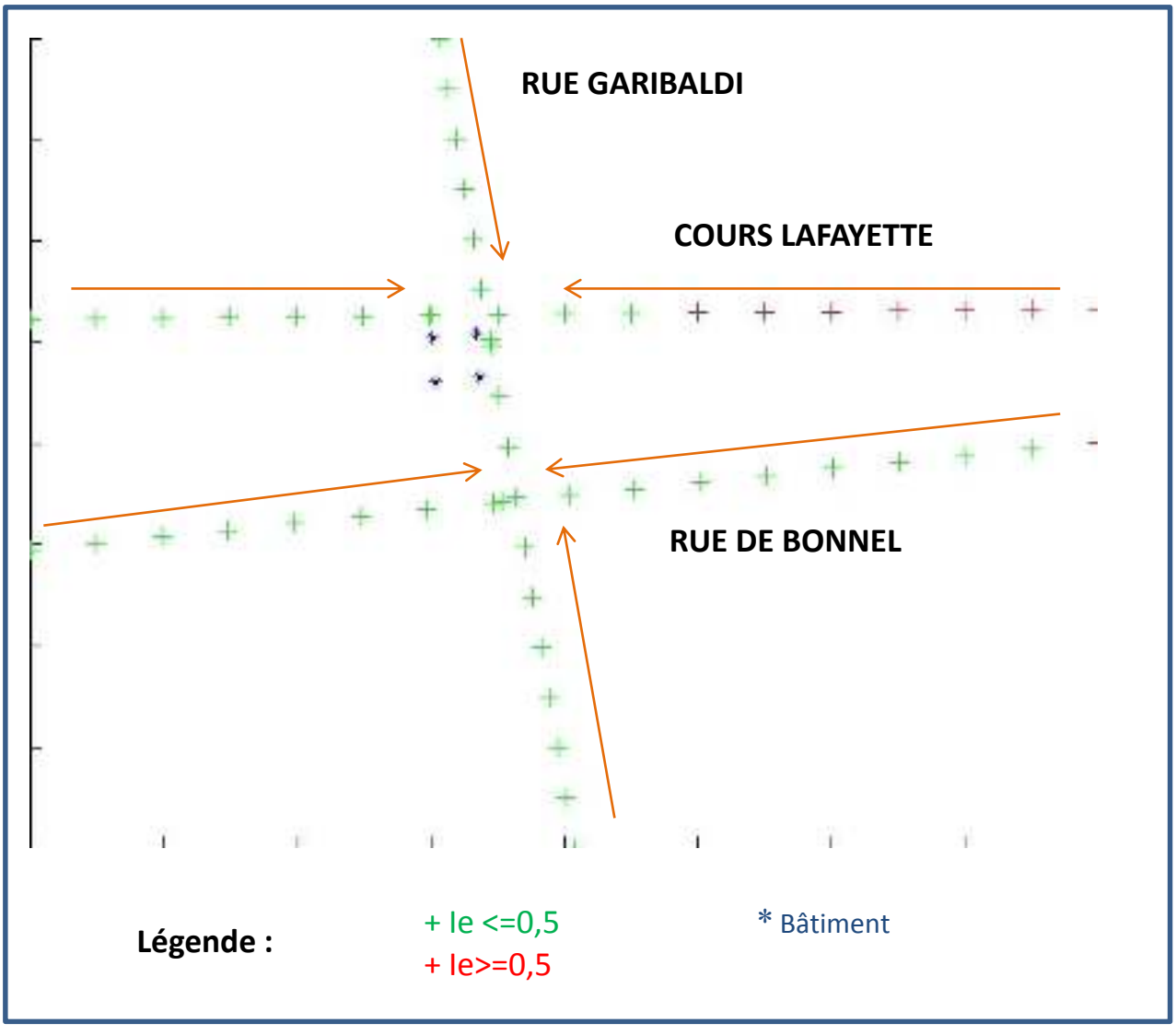


Figure 2.10 : Cartographie des risques le 10/06/2012 à 16h00

III. RESULTATS

Les objectifs ont été globalement réalisés. Comme l'atteste l'image suivante :

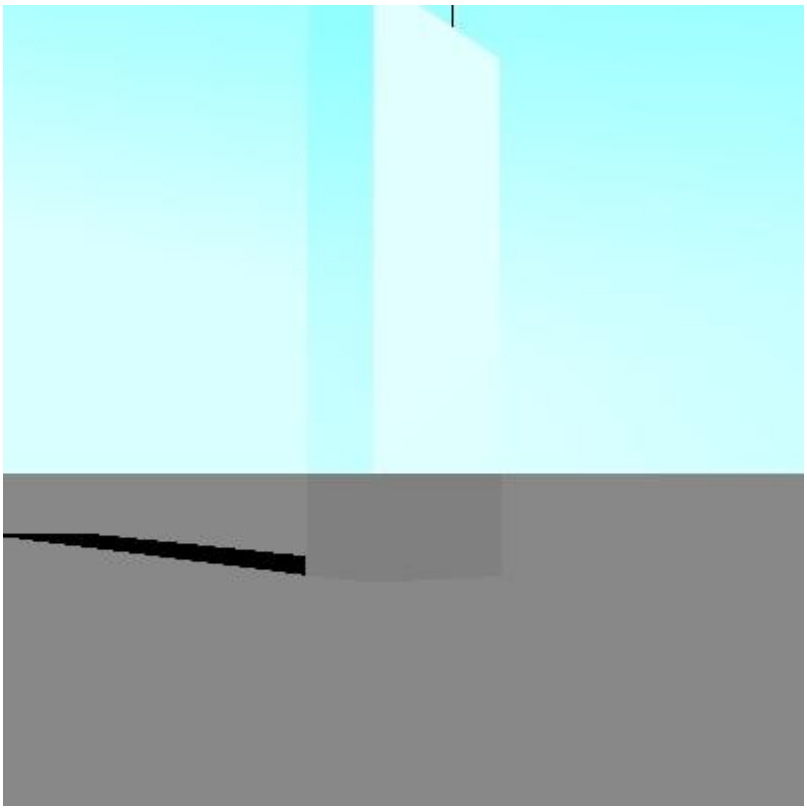
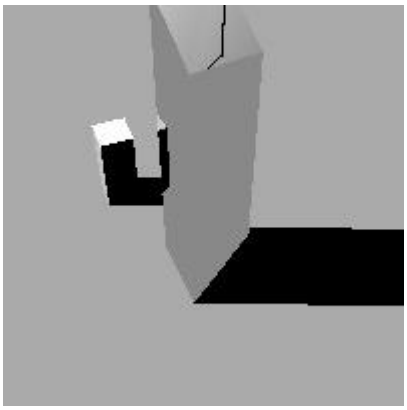


Figure 3.1 : Image globale

Cependant, il reste quelques soucis et « bugs ». En effet, il se peut que dans la modélisation des ombres portées, certaines exceptions apparaissaient. Dans l'image suivante, l'ombre du bâtiment adjacent à la tour ne semble pas apparaître dans son reflet.

Figure 3.2 :
Bug dans la simulation des ombres portées



Certains calculs font apparaître des luminances négatives. Nous n'avons pas trouvé la cause. C'est le cas de l'antenne qui apparaît en noir car dans la matrice des luminances, nous avons remplacé les luminances négatives par des 0.

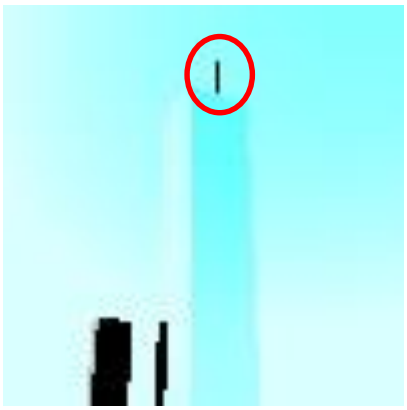


Figure 3.3 :
Bug de luminance sur l'antenne

De plus, l'échelle de couleur est trop claire. En conséquence, vers midi, les images sont quasiment illisibles. La raison est que la luminance du Soleil est très élevée et que la luminance des autres points décroit très rapidement. Il est donc difficile d'établir une échelle correspondant à 100% au réel.

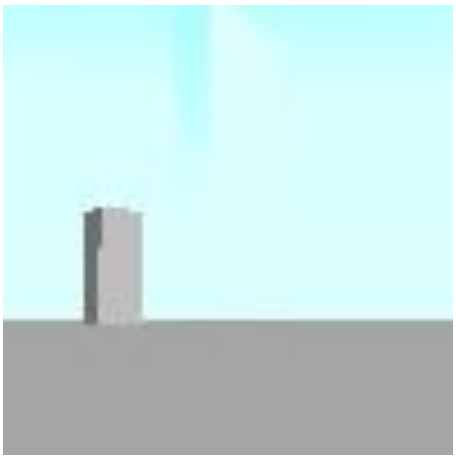


Figure 3.4 :
Image où l'éclairage est trop intense

Pour conclure, même si les objectifs sont presque tous atteints, le projet peut être toujours grandement amélioré tant sur la qualité des rendus que sur la vitesse d'exécution. On pourrait par exemple faire une fonction permettant pour un rayon donné de présélectionner les triangles à tester.

IV. Annexe

Listing de l'ensemble des fichiers remis (cf. fichier *LisezMoi.txt)* sous forme d'arborescence :

* Dossier : Présentation PPT 18122012
    * Présentation du projet informatique.pptx (*Support de présentation orale faite le 18/12/2012*
    * Dossier : images présentation
        * Cartographie des risques exemple.jpg
        * Couleurs_du_BETON.jpg
        * Couleurs_du_CIEL.jpg
        * Diagramme Programme
        * Image 1062012_90_200x200.jpg
        * Image 1062012_645_400x400.jpg
        * IMAGEINTEREFLEXION.jpg
        * IMAGERAYON1.jpg
* Dossier : SYNTHESE 20122012
    * Rapport Projet Informatique.docx
    * Dossier : images synthèse
        * Algorigrammeprojet.bmp
        * Cartographie des risques exemple.jpg
        * Image_1032012_90_100x100.jpg
        * Image_1062012_80_100x100.jpg
        * Image_1062012_90_100x100.jpg
        * Image_1062012_80_200x200.jpg
        * Image_1062012_645_400x400.jpg
        * Image_1062012_845_100x100.jpg
        * Image_1062012_1645_100x100.jpg
        * Image_1062012_1645_200x200.jpg
        * Image_1062012_1845_100x100.jpg
        * IMAGEINTEREFLEXION.jpg
        * Tour ville.jpg
        * Touravecnormales.jpg
        * Toursansnormales.jpg
* Dossier : Scripts
    * Dossier : IMAGES (*Qui se remplit automatiquement)*
    * calcul_position_soleil.m
    * Calcule_Luminance_Point_Matrice.m
    * calcule_nouvelle_position.m
    * calcule_position_centres_pixels.m
    * calcule_position_initiale.m
    * Calculer_intersection_rayon_triangle.m
    * Calculer_Luminance_CALCUL.m

* Conversion_Cart_Sph.m
* Couleurs_du_BETON.jpg
* Couleurs_du_BETON.mat
* Couleurs_du_CIEL.jpg
* Couleurs_du_CIEL.mat
* Detection_Intersection2.m
* Linearisation (*Pour information, n'intervient pas dans le script du projet mais utilisé pour créer les échelles de couleurs)*
* manipule_cameram.m
* parametrise_camera.m
* Pdt_Vectoriel .m (*fonction utilisée à la place de la fonction MATLAB)*
* positionne_camera.m
* Script_Principal_Projet.m
* Tracer_Normales.m
* Tracer_Triangles.m

* LisezMoi.txt

L'ensemble des fichiers sera compressé dans un fichier .zip portant le nom : **MOUTONPHILIPPOTCLAIRAIS.zip** .

Ce dernier dossier sera envoyé par mail à l'adresse : raphael.labayrade@entpe.fr dans le courant du jeudi 20 décembre 2012.

Pour tout renseignement ou en cas d'oubli dans l'intégralité du dossier, nous mettons à disposition nos mails ENTPE.fr :

clara.mouton@entpe.fr
marine.philippot@entpe.fr
clairais.aurelien@entpe.fr