

PHS4700
Physique pour les applications multimédia
Automne 2015

PAGE COUVERTURE **OBLIGATOIRE** POUR TOUS LES DEVOIRS

Numéro de devoir : 04

Numéro de l'équipe : 14



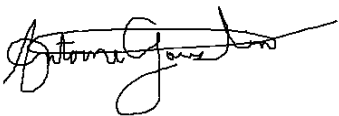
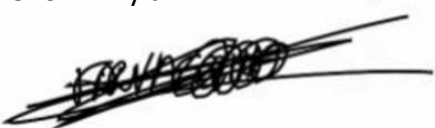
Nom: Rose	Prénom : Alexandre	matricule: 1580973
Signature :		
Nom: Mainville	Prénom : David	matricule: 1636075
Signature :		
Nom: Gosselin	Prénom : Antoine	matricule: 1588443
Signature :		
Nom: Farvacque	Prénom : Dylan	matricule: 1684271
Signature :		



Table des matières

I – Description du problème	2
II – Équations importantes.....	3
Équations d’intersection entre une surface et un rayon	3
Équations de réflexion et de réfraction à l’interface de deux milieux	3
Équations de détermination de position de l’image virtuelle	4
III – Méthode de traçage des rayons	5
IV – Description du logiciel	6
V – Résultats obtenus	7
VI – Analyse des résultats obtenus	11
VII - Discussions sur le devoir.....	13



I – Description du problème

Pour ce présent devoir, notre tâche consiste à simuler les réflexions et réfractions de rayons lumineux dans un fluide. Plus précisément, nous devons représenter l'image, perçue par un observateur immergé dans un fluide, d'une boîte métallique colorée qui est contenue dans un bloc transparent.

Nous nous intéresserons à quatre scénarios avec des indices de réfraction et des positions d'observateur distincts :

1. Observateur situé en $(-10, -10, 15)$, indice de réfraction du milieu de 1 et indice de réfraction du bloc transparent de 1,5.
2. Observateur situé en $(13, 10, 25)$, indice de réfraction du milieu de 1 et indice de réfraction du bloc transparent de 1,5.
3. Observateur situé en $(-10, -10, 15)$, indice de réfraction du milieu de 1,33 et indice de réfraction du bloc transparent de 1,1.
4. Observateur situé en $(13, 10, 25)$, indice de réfraction du milieu de 1,33 et indice de réfraction du bloc transparent de 1,1.

Dans le présent rapport, un bref rappel des équations nécessaires à la simulation sera fait. Ensuite, notre simulation développée sur MATLAB sera présentée. Par la suite, nous présenterons nos résultats et en ferons une analyse détaillée. Enfin, nous conclurons par une discussion sur les problèmes que nous avons dus surmonter au cours du devoir en ce qui a trait à la programmation et aux simulations.



II – Équations importantes

Équations d'intersection entre une surface et un rayon

Afin de déterminer la position du point d'intersection entre une des surfaces, que ce soit une surface du bloc de métal ou du bloc transparent et un rayon lumineux, on utilise la méthode qui permet de trouver l'intersection entre un plan et une droite. Ainsi notre plan et notre droite sont représentés par une équation paramétrique. On résout le système pour trouver le paramètre t qui nous permet de trouver le point d'intersection. Ensuite, on vérifie si ce point est situé entre les limites de notre surface.

Équations de réflexion et de réfraction à l'interface de deux milieux

Pour la réflexion, nous avons utilisé la première loi de Snell-Descartes qui dit que le sinus de l'angle d'incidence est égal au sinus de l'angle de réflexion. La formule pour avoir le vecteur unitaire réfléchi est :

$$\vec{u}_r = \vec{u}_i - 2\vec{l}(\vec{u}_i \cdot \vec{l})$$

Avec \vec{l} le vecteur normal unitaire sortant de la surface.

Pour la réfraction, nous avons utilisé la seconde loi de Snell-Descartes qui dit que le sinus de l'angle de réfraction S_t est donné par

$$s_t = \sin \theta_t = \left(\frac{n_i}{n_t} s_i \right) = \left(\frac{n_i}{n_t} \vec{u}_i \cdot \vec{k} \right)$$

Avec \vec{k} le vecteur normal unitaire du plan d'incidence.

Pour trouver l'angle critique nous avons utilisé

$$\theta_{max} = \left| \arcsin \frac{n_2}{n_1} \right|$$



Équations de détermination de position de l'image virtuelle

Enfin, pour déterminer la position de l'image virtuelle, on doit d'abord déterminer le vecteur unitaire donnant la direction du rayon observé. Ceci est donné par l'équation suivante où \vec{r}_0 est la position de l'observateur et \vec{r}_1 la position de l'intersection entre le rayon et une des faces du bloc transparent.

$$\vec{u} = \frac{\vec{r}_1 - \vec{r}_0}{|\vec{r}_1 - \vec{r}_0|}$$

Ensuite, on doit déterminer la distance parcourue par le rayon avant qu'il ne touche une des faces du bloc de métal. À chaque collision, on ajoute la distance parcourue depuis son dernier point de collision. Ainsi, cette distance totale est donnée par la somme de toutes les distances.

$$d = \sum_{i=1}^n |\vec{r}_i - \vec{r}_{i-1}|$$

Enfin, la position telle que vue par l'observateur d'un point est donnée par l'équation suivante :

$$\vec{r}_p = \vec{r}_0 + d\vec{u}$$

III – Méthode de traçage des rayons

La méthode utilisée pour tracer les rayons consiste à subdiviser les surfaces du bloc transparent en petits carrés de 0.1 cm x 0.1 cm. Ensuite, nous avons déterminé la position du point de chacun de ces carrés. Nous nous sommes construits une structure contenant chacun des vecteurs ayant comme point initial la position de l'observateur et comme point final la position précédemment trouvée. Ensuite, nous avons effectué la simulation telle que décrite dans la section suivante avec les équations susmentionnées. Pour notre simulation, nous ne simulons que les rayons vers les faces visibles. À cet égard, l'observateur, dépendamment de sa position, pourrait voir entre une et trois faces. Les faces les plus petites sont composées de 70 x 70 carrés de 0.1 cm x 0.1 cm pour un total de 4900 rayons. Les autres faces sont, quant à elles, composées de 150 x 70 carrés de 0.1 cm x 0.1 cm pour un total de 10500 rayons. Ainsi, selon le nombre de faces vues par l'observateur le nombre de rayons simulés variera.

IV – Description du logiciel

Les simulations sont entièrement réalisées à l'aide du logiciel MATLAB. Tout d'abord on initialise nos objets, soit l'observateur, le bloc transparent et le bloc de couleur. On définit chacune des propriétés disponibles dans l'énoncé pour ces trois objets.

Ensuite, on divise chacune des faces visibles du bloc transparent en petit carrés de 0.1 cm par 0.1 cm. Pour déterminer si une face est visible par l'observateur, il suffit que le produit scalaire entre la normale (sortante) et le vecteur directeur du rayon soit supérieur à zéro. On itère ensuite à travers chacun des points trouvés et l'on crée pour chacun de ces points une droite allant de l'observateur au bloc transparent.

Ensuite, on effectue les simulations sur les différents vecteurs que nous avons généré précédemment. Pour chaque vecteur, on vérifie s'il y a collision avec un des plans du bloc transparent, normalement tous les rayons générés ont une collision. On détermine ensuite si le rayon rebondit ou s'il traverse la surface du bloc. S'il rebondit on le rejette, sinon on calcule la diffraction puis on recommence l'opération, mais à l'intérieur du bloc transparent.

On regarde donc si le rayon entre en collision avec le bloc de couleur. Si oui, on garde le rayon initial, la distance parcourue et on note la couleur de la surface. Autrement, on calcule la réflexion du rayon à l'intérieur du bloc transparent ou la partie réfléchie lors de la réfraction sur la surface du bloc, le cas échéant. Finalement, on recommence cette série d'opérations jusqu'à ce qu'il y ait une collision avec le bloc de couleur ou jusqu'à un maximum de 100 itérations.

Une fois la simulation terminée, on reconstitue l'image virtuelle à l'aide des rayons initiaux et de la distance parcourue, puis on l'affiche dans un graphique.

Pour réduire le temps requis par la simulation, nous avons utilisé un thread pool de Matlab (parpool). Pour tirer profit des processeurs multi-cœurs actuels, nous initialisons un cluster local avec 6 fils d'exécution concurrents. Nous divisons ensuite le travail également de manière proportionnelle au nombre de fil d'exécution. Cette méthode n'est pas la plus efficace,



mais elle est facile à gérer, elle minimise le nombre de synchronisation en dupliquant les ressources communes et finalement elle bien plus performante que la méthode linéaire conventionnelle. Nous avons observé un gain en performance variant entre 2 et 3 fois plus rapide dépendamment des simulations.

V – Résultats obtenus

Dans les quatre simulations qui suivent, la position de l'observateur est représentée par un point noir.

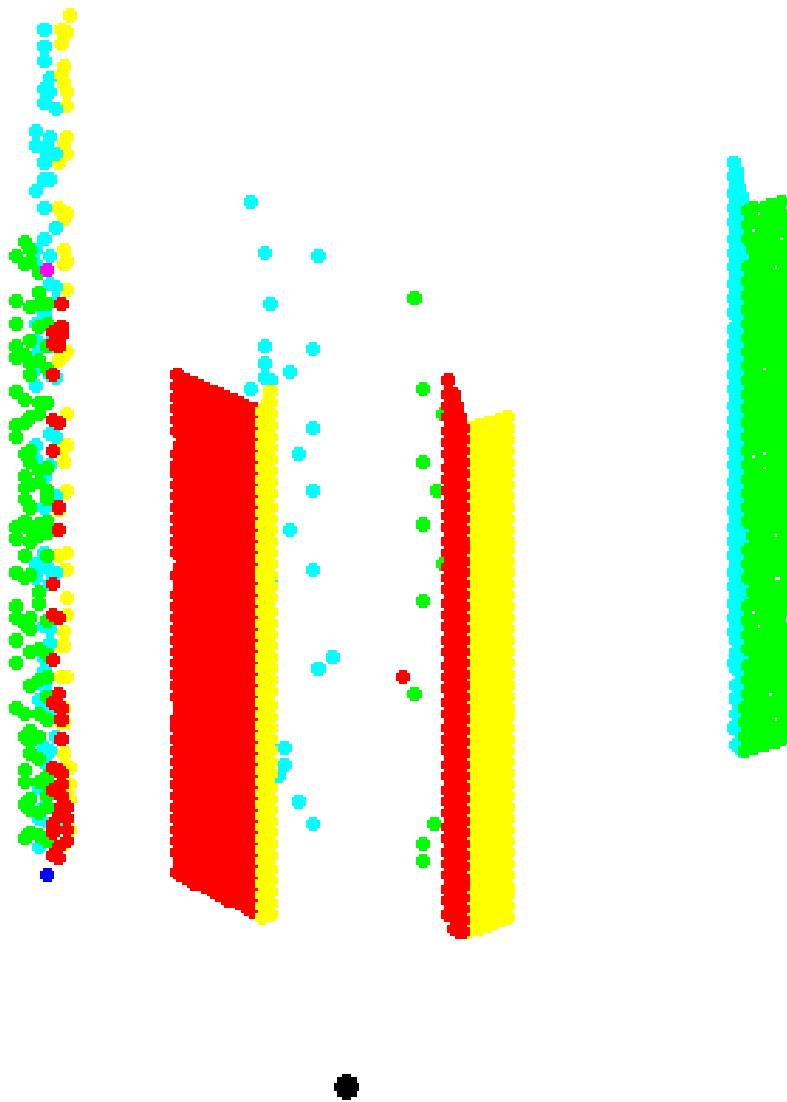


Figure 1 Simulation #1

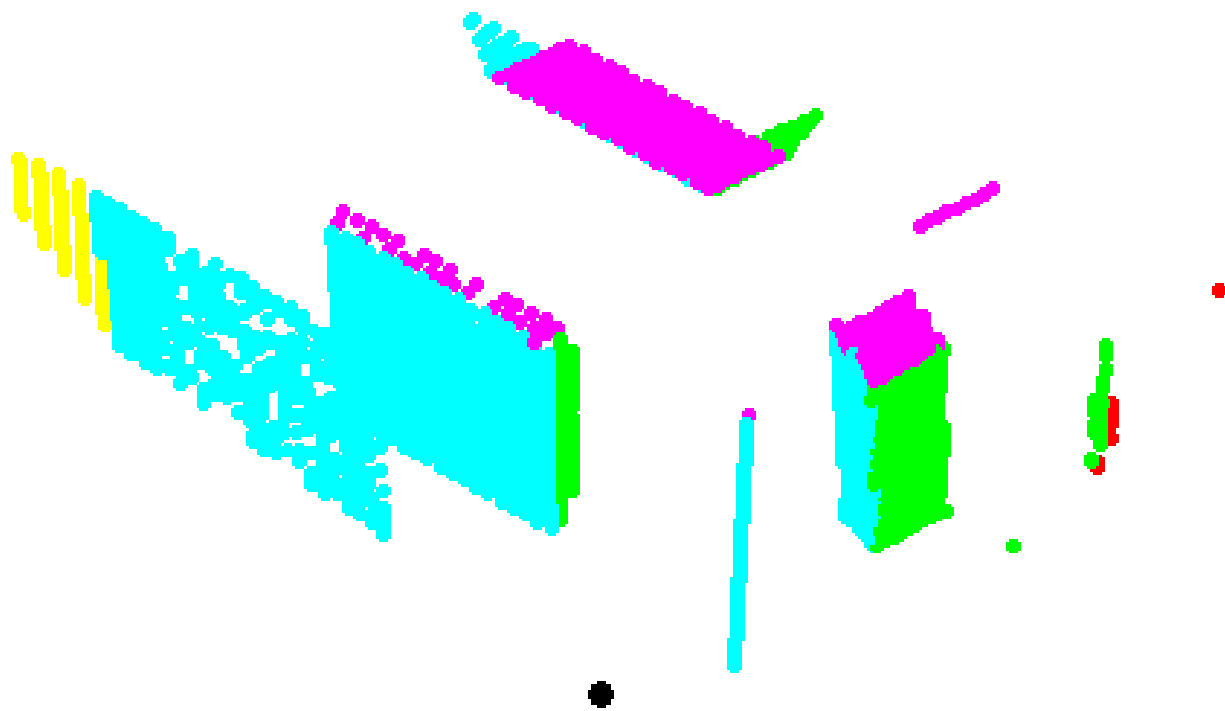


Figure 2 Simulation #2

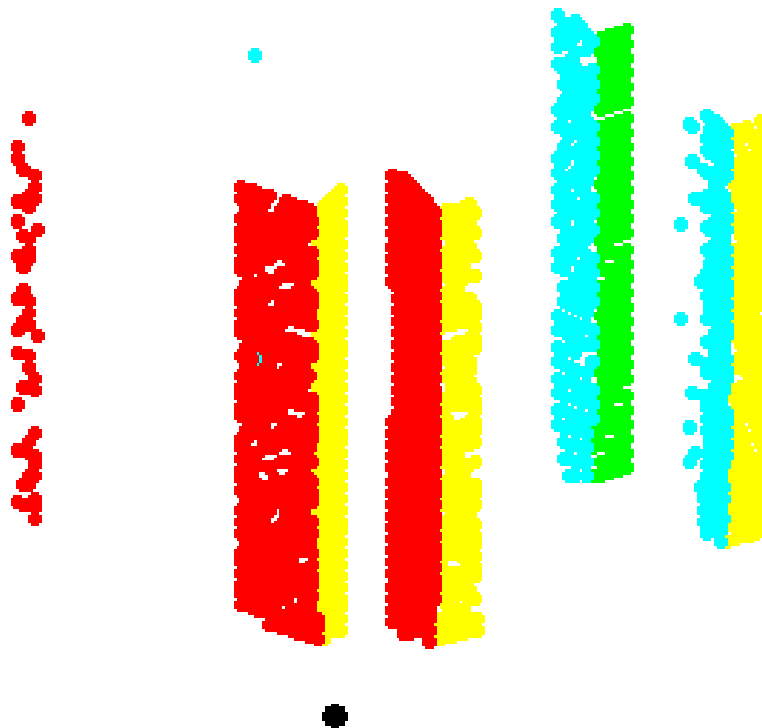


Figure 3 Simulation #3

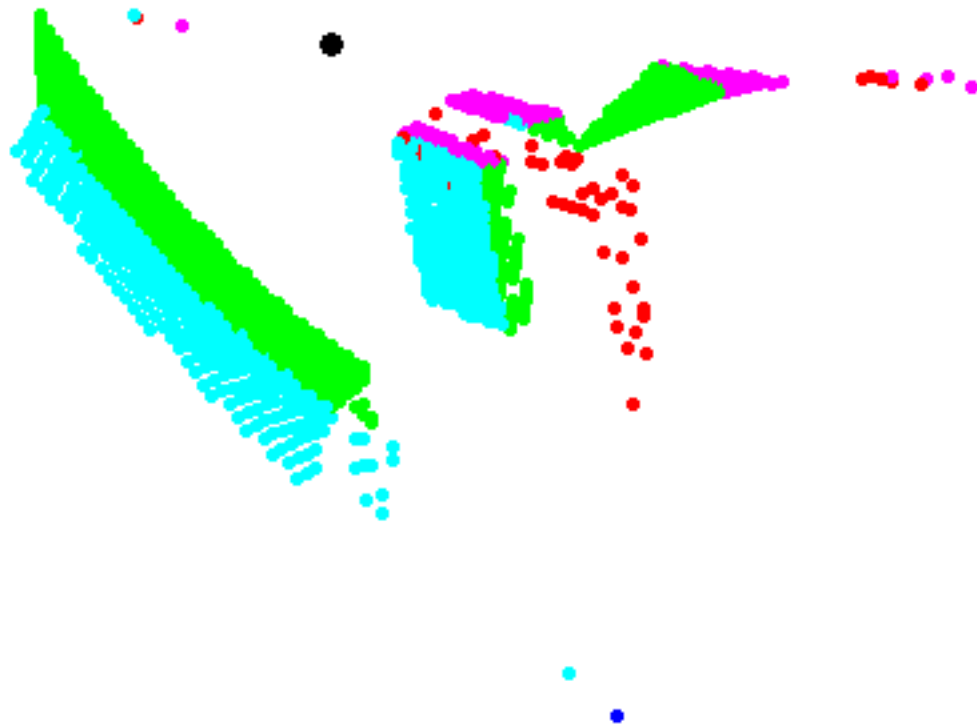


Figure 4 Simulation #4



VI – Analyse des résultats obtenus

Dans la première simulation, nous pouvons voir que nous avons obtenu une image assez nette. En effet, nous observons deux mêmes morceaux du bloc coloré. Ceux-ci sont issus de très peu de réflexions et/ou réfractions. Cela explique pourquoi ils sont aussi nets. Par la suite, on peut voir à droite de l'image une autre partie du bloc coloré. De par sa netteté et sa distance par rapport à l'observateur, cette partie-là de l'image est issue d'un plus grand nombre de réflexions et de réfractions. C'est aussi pour cela que l'on ne voit pas les mêmes faces du bloc coloré et donc pas les mêmes couleurs. De plus, à droite de l'image on voit qu'il devait aussi y avoir une partie du bloc coloré visible, en fait on peut même en déduire que cette partie du bloc devait être la même que celle à droite de l'image. Enfin, nous observons aussi qu'il y a des points qui sont présents, mais de manière non uniforme. Ce sont des rayons issus d'un très grand nombre de réflexions et de réfractions. Si un rayon est totalement isolé, on le considère comme erroné.

Dans la seconde simulation, on distingue qu'à gauche de l'image, la face cyan est réfractée plusieurs fois. On remarque aussi que plus on s'éloigne de l'observateur, plus l'image réfractée perd en précision. De même, la partie en haut de l'image est en fait une portion de la réflexion de la partie de l'image où il y a les faces magenta, cyan et verte. C'est pour cela qu'elles sont inversées l'une par rapport à l'autre. Enfin, comme pour la première simulation, on peut voir la présence de rayons isolés et donc erronés.

Pour la troisième simulation, on note que, similairement à la première simulation, nous obtenons deux images assez nettes du bloc coloré. Ces deux parties de l'image sont obtenues après un petit nombre de réflexions et de réfraction dans le bloc transparent. Par la suite, sur le bord droit de l'image, on observe la face cyan avec soit la face verte ou jaune du bloc coloré. Cela s'explique encore une fois par le fait que notre programme évalue chaque rayon pour un grand nombre de réflexions et de réfractions à l'intérieur du bloc transparent. Enfin, on note la face rouge à la gauche de notre image qui est comparable aux deux premières parties de l'image qui



sont rouge et jaune. On voit que plus on se dirige vers la gauche de l'image, plus la face jaune devient perpendiculaire à l'observateur et celui-ci la discerne de moins en moins.

Enfin, pour la dernière simulation, on voit que, de façon générale, l'image est ondulée et a un point de fuite en bas à droite. Par ailleurs, on remarque que nous avons beaucoup de rayons qui sont isolés, nous pouvons donc les considérer comme erronés. Pour finir, sachant que l'observateur se trouve au-dessus du bloc transparent, on peut déterminer que la partie de l'image qui compte les faces magenta, vert et cyan est la partie qui a subi le moins de déformation due à la réfraction (le rayon frappe le bloc après peu d'itérations). Au contraire, la partie de l'image plus à droite est un peu ondulée, signe qu'il y a eu plus d'itérations pour que ses rayons frappent une des faces du bloc coloré.

VII - Discussions sur le devoir

Lors de ce laboratoire, nous avons rencontré des problèmes dû au fait que Matlab n'est pas un langage de programmation mais plus un langage de calcul. En effet, nous avons créé des classes afin de représenter notre problème sous forme d'objets. Il est par exemple, très compliqué de créer des tableaux 2D d'objets. Bref, bien que Matlab ne soit pas un langage qui facilite l'orientée objet, cela nous a tout de même permis de clarifier et d'améliorer la qualité générale du code.

Nous avons aussi du optimiser notre programme en utilisant plusieurs threads. En effet, cela fut requis car le temps d'exécution du programme était très important pour chaque simulation compte tenu du fait qu'il fallait simuler jusqu'à 100 rebonds entre un rayon et les parois du bloc transparent. Il nous a donc fallu rechercher comment nous pouvions implémenter du multithreading en Matlab.

Enfin, il nous a été compliqué de savoir si nos résultats étaient valides. Effectivement, il est presque qu'impossible d'imaginer l'image résultante des réflexions et réfractions avec des indices de réfractions de milieux. Cependant, il nous était possible de voir si les images étaient cohérentes avec les informations de base. On peut noter que dans les deux premières simulations, l'indice dans la boîte transparente est supérieur à celui de l'environnement de l'observateur. C'est une situation semblable à notre vie quotidienne entre l'air ambiant et l'eau, il nous est donc possible de déduire la déviation des rayons lorsque ceux-ci changent de milieu. Nous avons donc essayé de vérifier de façon générale si les résultats que nous avons obtenus sont logiques.