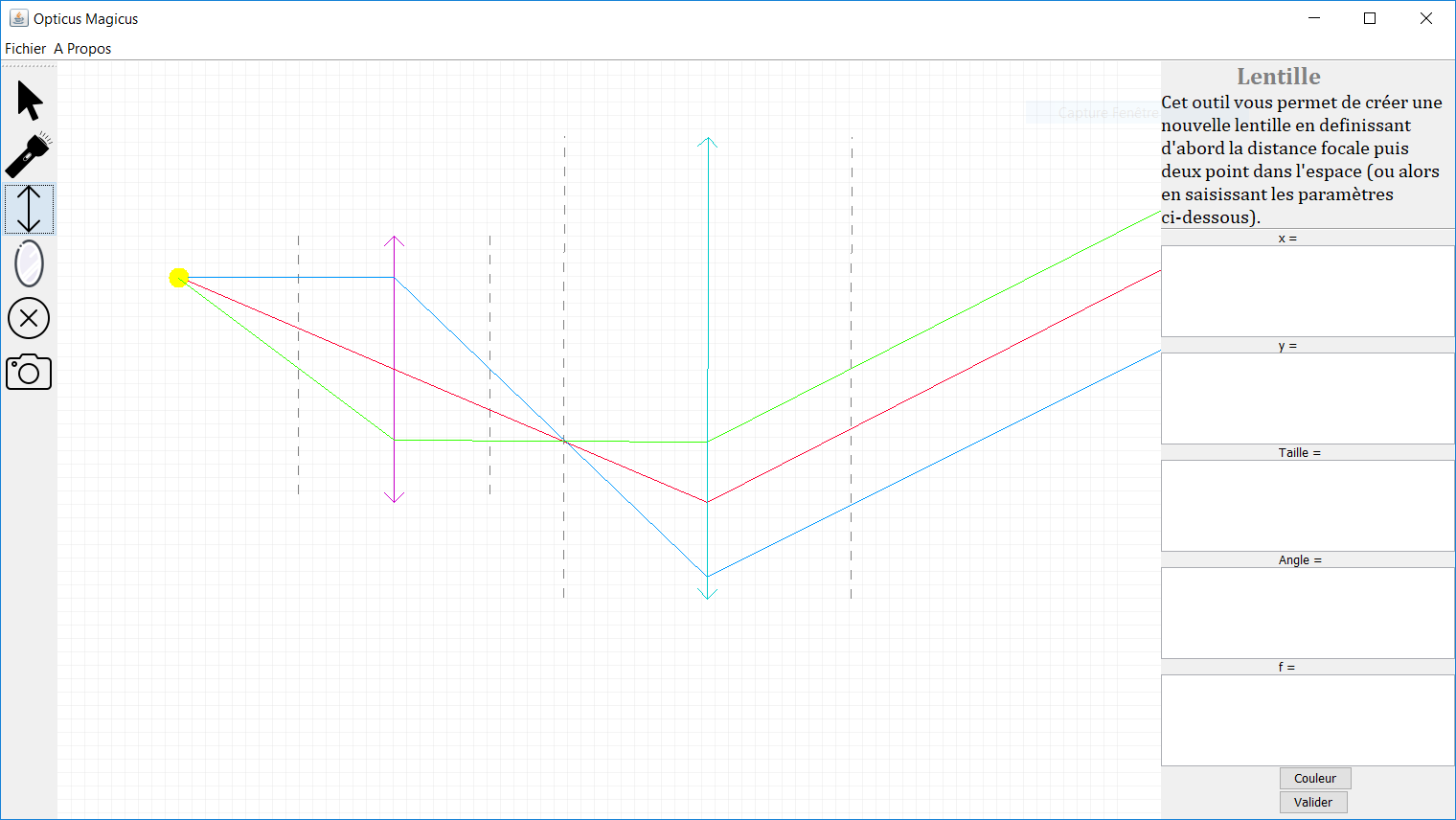
# Projet d’informatique de 2ème année : Opticus Magicus

## Introduction

Afin de mettre en œuvre l’ensemble des compétences acquises en informatique durant nos deux premières années à l’INSA de Lyon, notre objectif a été de concevoir un simulateur optique dénommé « Opticus Magicus ». En effet, l’optique constitue une partie importante des enseignements de physique du Premier Cycle, aussi bien en 1ère qu’en 2ème année. Nous avons donc réalisé un programme permettant de simuler les trajets de faisceaux lumineux en présences de plusieurs objets optiques tels que les lentilles convergentes et divergentes ou encore les miroirs. D’autre part, afin que le programme puisse être utilisé par un public très large, nous nous sommes particulièrement intéressés à la facilité d’utilisation. Afin de pouvoir développer le projet de manière méthodique, il nous a été nécessaire d’élaborer un cahier des charges, qui s’est enrichi au fur et à mesure.

Capture d’écran faite pour la simulation de la construction d’image dans un microscope

## Cahier des charges

Notre programme a été conçu dans un contexte éducatif et de découverte, s’adressant par exemple à des étudiants pouvant modéliser leurs expériences d’optique facilement. Le programme doit donc être intuitif et assez complet pour l’utilisateur. Pour cela, l’interface graphique doit être suffisamment épurée et les fonctionnalités doivent être facilement identifiables.

L’utilisateur doit pouvoir placer différents objets optiques respectant les lois physiques dans un espace plan :

* Une source de lumière pouvant être modélisée par un unique faisceau, dont la position et la direction du faisceau est choisie
* Des lentilles, dont la focale est déterminée par l’utilisateur (positive pour des lentilles convergentes et négative pour des lentilles divergentes)
* Des miroirs parfaits, c’est-à-dire des miroirs qui réfléchissent la totalité de la lumière reçue, mais aussi des miroirs semi-réfléchissants, qui réfléchissent une partie de la lumière et en transmettent une autre

Afin de rendre le programme le plus simple pour l’utilisateur et faciliter la compréhension, ce dernier doit pouvoir choisir la couleur des objets optiques ainsi que celle des faisceaux de lumière. Dans ce même but, la position et la taille des objets optiques doit pouvoir être choisie.

Le programme doit disposer d’une fonctionnalité de sauvegarde de l’état de la construction graphique, et donc d’une fonctionnalité d’importation de fichier pour ouvrir une certaine configuration. De plus, le programme doit permettre de prendre une capture d’écran de la zone de traçage directement à travers un bouton intégré à l’interface graphique.

## Problèmes posés

Les principaux problèmes rencontrés sont :

* La détermination de l’intersection entre les faisceaux et les objets optiques :

Ce problème nous a fait perdre du temps et nous avons modifié notre code de nombreuses fois pour que cela fonctionne normalement

* Le codage de la trajectoire du rayon transmis :

Le problème s’explique par la nécessité d’utiliser un mode de construction graphique, et donc tracer un autre faisceau, et déterminer d’autres points d’intersections

* S’assurer que le rayon transmis est du « bon » côté de l’objet optique :

En effet, cela demande d’être sur du coté sur lequel arrive le faisceau, ce qui a été résolu dans notre cas par l’utilisation du produit scalaire

## Principe de l’algorithme

### Interface graphique

Pour faciliter l’utilisation du programme, il suffit de lancer le run.bat, ce qui va ouvrir directement le programme. Afin de rendre le programme le plus intuitif possible, 5 grandes icônes sont présentes sur la barre d’outils, qui elle-même peut être placée où l’utilisateur le désire.

Afin de placer un objet optique, on a le choix entre saisir la position (x,y) de son centre, sa taille et son angle d’inclinaison par rapport à l’horizontale, ou bien on peut directement placet l’objet avec la souris. Grâce à l’icône « Sélectionner », on peut même facilement repositionner les objets avec un « click and drag ». Les valeurs des tailles, positions et angle s’adaptent alors automatiquement.

Les systèmes optiques étudiés étant généralement composés de nombreuses lentilles ou miroirs, il est proposé à l’utilisateur de choisir la couleur de chaque objet optique grâce à une large palette de couleur en cliquant sur « Couleur ».

Les trois icônes suivantes permettent donc de choisir entre « Source », « Lentille », et « Miroir ». Afin de supprimer un élément, l’utilisateur peut soit sélectionner l’objet et appuyer sur « Supprimer », ou alors appuyer sur l’icône « Supprimer » et ensuite choisir le ou les objets à supprimer.

Afin de rendre le programme encore plus utile, pour les travaux de groupes par exemple, l’utilisateur doit pouvoir envoyer ou montrer ses constructions. Pour cela, il peut utiliser la dernière icône « Capture d’écran » qui permet d’enregistrer la zone de traçage à l’endroit souhaité sur l’ordinateur, ou bien sauvegarder la construction en cliquant sur « Fichier », puis « Sauvegarder ». Inversement, il peut ouvrir une construction en appuyant sur « Importer/Charger un fichier ».

### Gestion du rayon au contact des objets

Il est assez compliqué de déterminer la trajectoire des rayons, en effet, durant les Premier Cycle, aucune formule n’est donnée, cela est fait uniquement de manière graphique. Ainsi, pour déterminer les comportements, il était d’abord nécessaire de déterminer l’intersection des rayons incidents avec l’objet.

La première partie du rayon est alors tracé, puis pour déterminer la 2ème partie du faisceau, on doit déterminer le point d’intersection entre le plan focal de la lentille (du bon côté !) et un faisceau parallèle au premier faisceau passant par le centre optique de la lentille. On connaît alors 2 points de la 2ème partie du faisceau, et on peut le tracer ensuite. Afin de déterminer ce « bon côté », il est possible d’utiliser le produit scalaire. En effet, en posant que la normale à nos objets optiques est vers la droite, ensuite on peut déterminer le plan focal de la lentille qui nous intéresse.

Pour le miroir, il fallait également déterminer le point d’intersection entre le faisceau et le miroir. Il fallait alors déterminer le symétrique de l’angle d’incidence par rapport à la normale du miroir. Ensuite, aussi bien pour les lentilles que pour les miroirs, il faut considérer la possibilité que le faisceau sortant rencontre à son tour un autre objet optique.

## La structuration des données

Afin de rendre notre code efficace, nous avons créé une classe mère abstraite « ObjetOptique » car les sources, les lentilles et les miroirs partagent tous les 3 des attributs communs comme une position du centre, un angle avec l’horizontale, une taille et une couleur. Ainsi, grâce au principe d’héritage appris cette année, cela permet d’éviter d’avoir plusieurs classes quasiment identiques.

Pour ce qui est de l’interface graphique, notre « FenetrePrincipale » est la classe qui code la JFrame et les 2 panels existants. La classe « BarreOutils » est la classe qui rassemble les 5 boutons utilisés de la barre.

La classe « Geometrie » rassemble différents calculs ou méthodes nécessaires pour le traçage des rayons. La plus importante de nos classes au niveau du code est « ZoneTracage ». En effet, c’est elle qui rassemble les différents calculs qui permettent de déterminer les points d’intersection et les rayons sortant des objets optiques.

## Problèmes connus et pistes d’amélioration

Le programme a pour objectif de visualiser le trajet des faisceaux de lumières à travers différents objets optiques, c’est pourquoi la notion d’équilibre d’intensité des rayons incidents, réfléchis et transmis (pour les miroirs semi-réfléchissants) est négligée dans cette version du programme.

D’autre part, notre programme fonctionne avec des sources représentées par des rayons uniques. On peut donc imaginer des sources ponctuelles émettant de nombreux rayons ou bien des « murs » de lumière, c’est-à-dire plusieurs rayons tous parallèles.

De même, les couleurs que l’on utilise ne servent qu’à éviter d’embrouiller l’utilisateur. Cependant, nous pouvons que notre programme prenne en compte les longueurs d’onde des faisceaux et ainsi adapter leurs comportements. Ainsi, il est possible de créer d’autres objets optiques tels que les prismes.

## Conclusion

Pour conclure, ce projet nous a permis de mener le développement d’un programme à partir de l’élaboration d’un cahier de charges jusqu’à son rendu. Ce projet nous a donc permis de mettre en application les connaissances pluridisciplinaires acquises durant notre formation à l’INSA afin de répondre à un besoin, ce qui constituera une part de notre futur métier d’ingénieur. Plus généralement, ce projet nous a permis d’améliorer nos compétences, aussi bien en informatique que des compétences nécessaires pour un bon ingénieur tels que le travail d’équipe, et une recherche efficace de solution au problème posé.