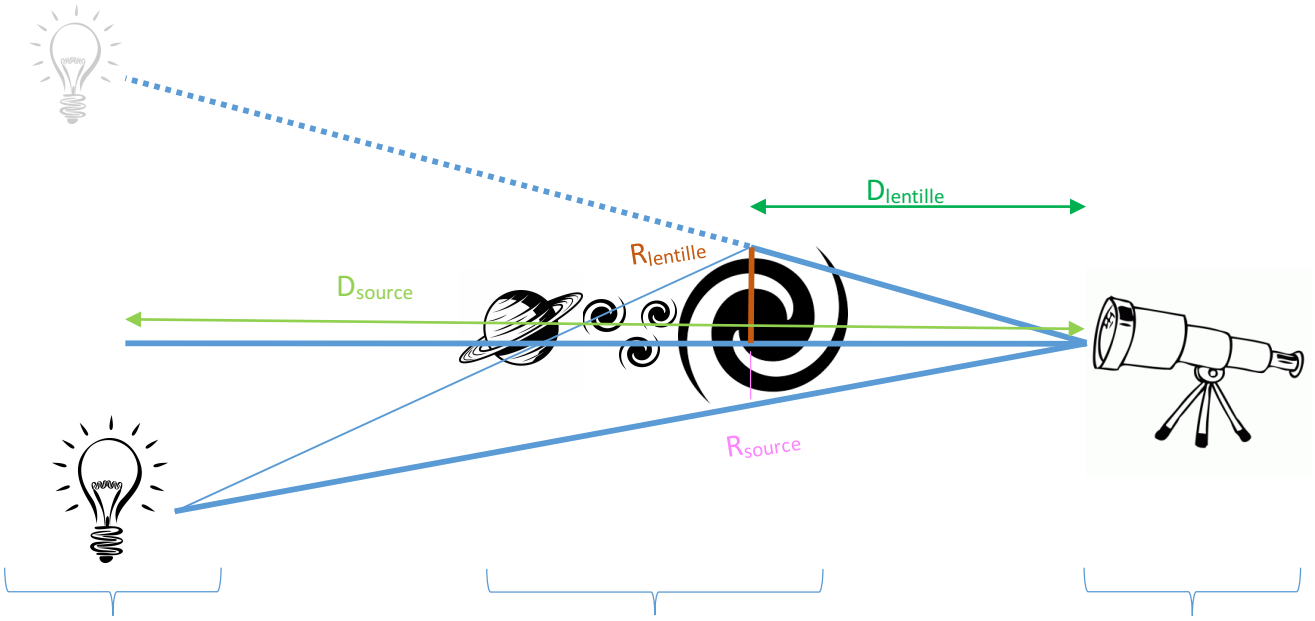
PROJET D'INFO C++ : LENTILLE GRAVITATIONNELLE

INTRODUCTION

La physique des particules et la cosmologie impliquent bien souvent des concepts compliqués dont la compréhension est très subtile. Mais parfois, les simulations numériques permettent d'en rendre certains légèrement plus simples. Discutons par exemple des lentilles, perçues généralement comme des pièces de verre courbées que l'on trouve dans les microscopes ou les télescopes, permettant d'agrandir ou de préciser un objet. De façon similaire, la lentille gravitationnelle permet de voir des objets « cachés » derrière d'autres. Les objets massifs déformant l'espace-temps, la lumière qui elle n'a pas de masse « suit » ces distorsions : si l'espace est courbé, la lumière se courbe également avec lui. C'est Einstein le premier qui a prédit en 1912 le phénomène de lentille gravitationnelle. Les ingrédients de l'observation d'un tel phénomène sont :



Une source

Un objet déflecteur (lentille) : planète, galaxie, amas de galaxie

Un observateur (nous)

ENJEUX ET ARTICULATION DU PROJET

L'objectif principal de ce projet est de réaliser une animation graphique d'un phénomène de lentille gravitationnelle. On s'intéressera notamment au visuel d'une source pendant le passage d'une lentille à proximité de la ligne de visée observateur-source.
L'utilisation de la programmation orientée objet est indispensable dans ce projet. J'ai pu relever 4 étapes essentielles à l'élaboration de ce projet et aux premiers résultats obtenus :

- En premier lieu, l'idée est de créer plusieurs classes, dont les seront utilisés lors de l'exécution d'un programme principal test. Ces différentes classes, qui décrivent les astres mis en jeu, et les méthodes associées seront décrites dans le point suivant. On se servira en particulier d'un programme C fournit, qui permet de représenter, à l'aide de 0 et de 1, le visuel d'une source en fonction de la position, du rayon, de la masse ainsi que la distance à l'observateur de l'objet déflecteur.

- Ensuite, nous procédons à la conversion en image. En effet, cette alternance de 0 et de 1 ne permet pas de réaliser une animation. Nous utiliserons alors une classe donnée dans le projet C++ « Transfert d'image », pour conserver l'enjeu de la programmation orientée objet. Cette classe permet de transformer à partir d'un tableau, une image au format bmp. Nous obtenons alors non pas une alternance de 0 et de 1 mais bel et bien une image qui sera exploitable pour la réalisation d'une animation ou gif.
- Enfin, la réalisation de l'animation se fera en convertissant l'ensemble des images.bmp en une image.gif, sur laquelle nous pourrons observer l'évolution du visuel, en fonction de divers paramètres.

ORGANISATION ET COMPOSITION DU PROGRAMME

Les classes utilisées dans ce projet sont réunies dans une « Super Classe » nommée : GravLens.h. On y trouve 3 classes : la classe Lentille, la classe Source et la classe Image.

- La classe Lentille fournit 4 variables membres : Xlentille, Ylentille, Dlentille, Mlentille, vx, vy et vz qui donnent respectivement les coordonnées de la position de la masse déflectrice, sa distance à l'observateur (nous),sa masse et sa vitesse. A cette classe sont également associées 2 fonctions membres : initialise (qui associe une valeur à chaque attribut) et affiche (qui permet de tester le bon fonctionnement de la classe).
- La classe Source ressemble sensiblement à la classe Lentille. Elle donne 2 variables membres : Rsource et Dsource qui sont respectivement le rayon de la source et sa distance à l'observateur (nous). Sont associées à cette classe deux méthodes : initialise et affiche.
- La classe Image prend en compte trois variables : lens et source, étant elles-mêmes des objets issus des classes Lentille et Source (que l'on a instancié) et un entier N, le nombre d'images que l'on veut créer. Cette classe image sert dans un premier temps à créer un visuel du phénomène grâce à des 0 et des 1 grâce à la fonction membre creerimage (en balayant tous les pixels en étudiant le phénomène d'un point de vue « inverse » : on prend un rayon issu de la source et on remonte à l'observateur). Cette méthode sera ensuite affinée grâce à la classe bmp_io, qui nous permettra d'obtenir une image au format bmp.

Un programme principal test GravLenstest.cpp permet d'obtenir une image, après avoir instancié les objets de type image, qui prennent en argument : Mlentille,Dlentille,Xlentille, Ylentille,Vxlentille,Vylentille,Vzlentille,Rsource,Dsource,Nbre_images. On peut créer plusieurs images à partir desquelles on peut créer un seul et unique gif.
Pour obtenir le gif complet (le but étant d'avoir une animation), on écrira dans le terminal : convert image_0*.bmp nom_gif.gif

Le diagramme de classe est donné en Annexe.

COMPILATION

La première commande consiste à compiler le programme, où le fichier lens contient toutes les données.

```
$  g++ GravLenstest.cpp    GravLens2.cpp    bmp_io.cc    –o lens
```

Pour obtenir toutes les images créées par le programme, on exécute :

```
$  ./lens
```

Enfin, on souhaite créer un gif à partir de toutes ces images, on utilisera la commande :

```
$ convert image_0*.bmp  nom_gif.gif
```

RÉSULTATS

On constate que le visuel d'un phénomène de lentille gravitationnelle dépend de plusieurs paramètres.

-Amas de Abell : C'est un amas situé dans la constellation de la Vierge. Cet amas, très massif, forme une lentille gravitationnelle pour les objets situés derrière lui, notamment une galaxie. On peut tester le programme pour un tel amas.

- Les données qui nous étaient fournies considéraient un objet de type solaire comme lentille ($m=2^e30kg$) à 4kpc, qui met en évidence une géante rouge ($R=3^e11m$) à 8kpc . On obtient de belles images qui reflètent le visuel de la source en fonction du déplacement de la lentille. On notera notamment que l'on observe un anneau de Einstein lorsque la source, l'observateur et la lentille sont parfaitement alignés.

- Nous avons également testé des objets plus petits comme par exemple : un observateur sur Mars, qui observe le Soleil sachant que la planète Terre se situe entre les deux (utilisée comme lentille) : on constate alors que le phénomène n'est que très peu visible et est surtout accentué pour des objets très massifs, comme des galaxies ou amas de galaxies.

PERSPECTIVES

- Il aurait pu être intéressant d'ajouter une classe Vecteur h, qui servirait à décrire par la suite la vitesse de la masse ou de la source. Ainsi, les classes Source et Lentille pourraient utiliser cette classe Vecteur, et nous pourrions éventuellement tester d'autres paramètres qui influencent le visuel d'un phénomène de lentillage gravitationnel, comme notamment la vitesse de l'objet déflecteur mais également celui de la source.

- Il aurait été intéressant aussi de représenter la lentille sur notre animation, ou encore d'avoir pu maîtriser la classe bmp un peu plus tôt pour pouvoir superposer à notre animation une image « réelle » de galaxies ou d'amas.

CONCLUSION

Ce projet m'a permis de redécouvrir, et me familiariser avec le monde de la programmation orientée objet, un monde qui m'était jusque-là assez complexe, même après plusieurs séances de TDs. J'ai su m'approprier ce langage à travers un sujet physique très intéressant, bien que le programme soit encore très perfectible. J'ai notamment réussi, de la façon la plus simple, à créer un programme qui réponde à notre problématique : réaliser une animation graphique d'un phénomène de lentille gravitationnelle. Je vais très probablement poursuivre ce projet, en essayant d'intégrer des notions que je n'ai pas utilisées (comme l'Héritage ou l'encapsulation) et y intégrer les différentes perspectives de travail décrites plus haut.

ANNEXE : DIAGRAMME DE CLASSES