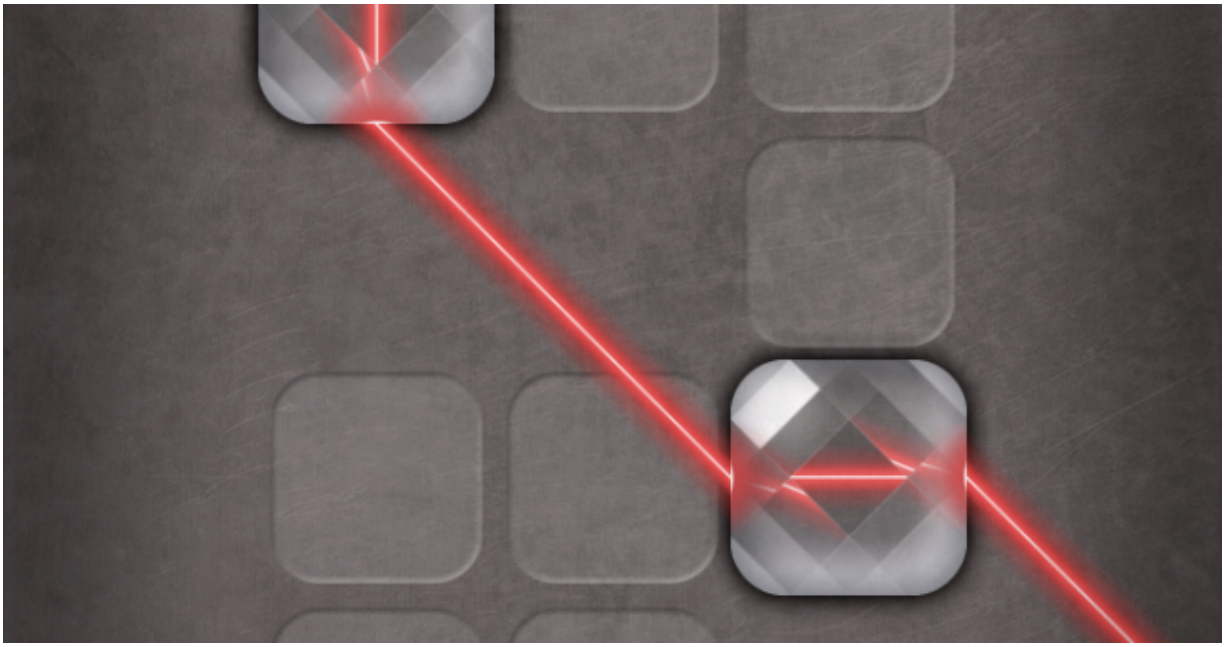
Université de Paris

Yann EDIKEU

Khalifa ABABACAR DIONE

Marie Tiffany EMMOU COFFI

RAPPORT DE PROJET:



Présentation du projet:

Notre projet se base sur le jeu Lazors . Il s'agit d'un jeu assez simple, le but étant de faire réfléchir un laser à l'aide de blocs jusqu'à une cible. Notre jeu comporte 10 niveaux différents. Ces niveaux vous permettront de pouvoir rencontrer les différents blocs disponibles. En effet, ce jeu comporte plusieurs sortes de bloc qui réfléchissent (ou non) le laser à leur manière. Nous en avons implémenté 8.

Organisation:

On a commencé à travailler sur le projet dès la répartition des groupes, ce qui nous a laissé à peu près 3 mois pour finaliser le projet. Toutes les semaines, on s'est vu

en rendez-vous sur Discord ou à la bibliothèque pour montrer ce qu'on avait implémenté de notre côté et planifier ce qui devrait être développé dans le futur.

On a utilisé la plateforme GitLab (gaufre.informatique.univ-paris-diderot.fr) de l'université pour partager nos ajouts successifs, ce qui nous a permis de mieux coordonner notre travail en groupe.

I-DEVELOPPEMENT:

Pour commencer on a fait un cahier de charge pour modéliser et créer toutes les classes. On savait les objets qu'on allait devoir manipuler plus tard.

Une fois qu'on a créé les classes, on a implémenté les grandes fonctions internes(package model), par exemple la fonction qui permet d'orienter le laser si ce dernier est en contact avec un bloc. Ce sont toutes des fonctions qui admettent au moins un argument en entrée pour exécuter un action spécifique. On s'est dit qu'on allait générer plus tard les événements qui permettent d'activer ces fonctions.

Une fois qu'on a implémenté l'ensemble des fonctions internes, on a commencé à développer la vue(package vue) qui est constituée de plusieurs classes et dont l'une renferme une grande partie des constituants graphiques du jeu (la classe VuePlateau).

La classe VuePlateau contient des fonctions synonymes à celles que l'on trouve dans les classes internes.Ce sont des fonctions qui, après un événement, récupèrent des paramètres, activent une fonction interne puis s'occupent du rendu graphique post-événement.

De manière générale, n'importe quel objet (bloc, laser, cible…) est représenté par une classe non graphique(model), même si elle ne sert qu'à stocker des coordonnées.

On également d'autres classes de l'interface graphique (ButtonMenu, FinDePartie, LevelButton) qui s'occupent de tout ce qui est bouton, événement de bouton, passage d'une fenêtre à une autre mais également l'organisation des niveaux du jeu.

Le rendu final s'exprime sous forme de fenêtre qui contient des ImagesIcon et des buttons. Les ImageIcon et JButton appellent un ActionListener/MouseListener quand ils sont cliqués, qui appellent une fonction graphique. Cette fonction s'occupe de récupérer les paramètres utiles et appelle une fonction non graphique, puis imprime

le rendu. Le programme continue alors jusqu'à ce qu'un autre élément soit cliqué, ce qui recommence la boucle.

II-CLASSES:

Les classes sont contenues dans différents packages spécifiés d'un nom qui renseigne sur son type de contenu. Parmi ces packages nous avons :

Le package model qui contient les classes :

- Bloc { classe abstraite qui contient toutes les méthodes et propriétés d'un bloc }

- BlocReflechissant, BlocAbsorbant, BlocPrismatique, BlocSemiReflechissant, BlocTeleporteur {qui sont toutes des classes qui étendent de Bloc et redéfinissent chacune les méthodes et propriétés de Bloc selon le comportement du laser sur chacune d'elles notamment avec la methode la plus importantes `public int deviationLaser()`}

- Case { classe abstraite qui contient toutes les méthodes et propriétés d'une qu'elle soit visible ou cachée}

- CaseVisible, CaseCachee { comme pour les blocs nous avons aussi différents types de cases. Ces deux classes étendent de Case et contiennent toutes les méthodes et propriétés dont une case doit avoir et selon le type de case(visible ou cachée) }

- Cible { classe concrète qui compose les propriétés d'une cible(position x,y) }

- Laser { classe concrète qui représente un laser. Dans cette classe nous trouvons les composants d'un laser }

- Plateau { la plus grande classe du modèle, elle s'occupe de maintenir le plateau du jeu. C'est la classe qui implémente l'étendue des fonctions du jeu, et elle prend en paramètre la taille plateau (tableau de cases), une Linkedlist

de lasers et un tableau de cibles. Dans cette classe on voit également la méthode : `public void initLaser()(Méthode qui va initialiser le calcul des points des lasers du plateau. Tout en vérifiant ses déviations et la condition de victoire.), public boolean deplacementPossible(int x1, int y1, int x2, int y2)(qui vérifie si le déplacement d'un bloc est possible), public void calculerChemin(Laser l) qui prend en paramètre un laser définie le type de déviation et calcule le chemin de la trajectoire du laser lorsqu'il touche un bloc), public int nouvelAngle(int x, int y, int angle)(méthode qui définie la direction du laser selon le bloc),public int[] caseAVerifier(int x, int y, int angle)(methode spécifique au laser qui lui permet de vérifier les blocs par lesquels passe le laser)... }`

Le package vue c'est le package de l'implémentation graphique. il est composé des classes suivantes :

- ButtonMenu { cette classe implémente les différents boutons qui nous permettent de contrôler la vue }
- LevelButton { cette classe implémente les boutons permettant de choisir les niveaux,lorsque l'on clique dessus le niveau souhaité est chargé }
- FinDePartie { classe concrète graphique qui étend JDialog
- que l'on a implémenté sous forme de fenêtre de JFrame qui interagit avec le jeu et devient visible si toutes les cibles d'un niveau sont atteintes }
- VuePlateau { C'est la plus grande classe de la vue. Cette dernière implémente les fonctions qui permettent de donner une vue graphique du contenu du plateau et ceci avec des méthodes comme : `public void Plateau(Graphics g)(vue graphique case du plateau), public void TraceLaser(Graphics g)(vue graphique du laser, le laser est implémenté sous forme de ligne(drawLine)grâce à une linkedlist de point), public void Cible(Graphics g)(vue graphique d'une cible, les cibles sont des point contenus dans un tableau), public void initbloc()(vue graphique des blocs et des cases, qu'on a implémenté sous forme d'image Icon pour plus d'esthétique) }`

Le package controleur : package qui l'implémentation à la fois l'interface graphique et le model. il est constitué d'une seule classe :

- Controleur { classe qui initialise tous les niveaux du jeu et interagit en même temps avec le model et la vue. Elle met en place l'ensemble des propriétés du plateau de jeu(cases, blocs, lasers , cibles …) }

Le package niveau : package qui contient les différents niveaux du jeu sauvegardés dans des fichiers .ser (serializable) notamment par la méthode `public void sauvegarder(String fileName)` de la classe Plateau.

La classe Main qui contient la fonction main pour le lancement et l'exécution du programme.

(voir DIAGRAMMES DES CLASSES.PDF)

III-Problèmes Rencontrés:

Bien évidemment , ce projet ne s'est pas réalisé sans encombre. Nous avons dû faire face à plusieurs problèmes . Nous avons commencé par adopter une mauvaise méthode : chacun travaillait de son côté sur une différente version du projet.Nous avons rapidement vu que cela nous faisait perdre énormément de temps et avons décidé de nous accorder sur une seule et même version . Nous avons également dû faire face à des changements sur la composition de notre groupe. Au début, nous étions un groupe de 5 puis nous avons dû échanger un des membres de notre groupe. Suite à cet échange il a fallu tout réexpliquer au nouvel arrivant pour qu'il puisse se plonger dans le projet . Cela s'est très bien passé mais finalement le nouveau membre n'a jamais participé au projet.De plus, un autre membre du groupe a subitement décidé (sans nous prévenir au préalable pour qu'on puisse mieux s'organiser) d'arrêter. Nous nous sommes donc finalement retrouvés à 3 pour faire ce projet ce qui nous a beaucoup handicapé.

En dehors de ces problèmes d'organisations nous avons également rencontré des difficultés et blocages au niveau de l'implémentation du jeu. Le laser fut un défi considérable pour nous. Nous ne savions pas comment le représenter.Assez tôt dans le projet, nous avons pus remarquer en jouant au jeu original que le laser ou encore les cibles se trouvait en quelque sorte au milieu des cases du plateau a chaque fois(elles ont une position qui (paire,impaire) ou (impaire,paire)).À partir de ce constat nous avions alors élabore une première tentative d'implémentation qui consistait à convertir les points du laser et les cibles dans une base différente(2 fois plus grande) afin qu'elle se trouve bien au milieu des cases comme souhaité. Malgré toute notre bonne volonté, cette méthode d'implémentation ne marchait tout simplement pas (cela était très certainement dus à des mauvais calculs), et pourtant nous avons persisté sur cette voie pendant au moins la moitié du projet, ce qui nous a fait considérablement perdre du temps.(A noter également que cette idée d'implémentation avait été soumise par les membres qui ont finalement quitté le

projet). Face à cette difficulté qui semblait insurmontable, nous avons décidé de simplifier notre implémentation en supprimant le changement de base, et en considérant que les positions des points du laser ou des cibles se trouvent à des positions deux fois plus grandes que notre plateau de case.

Un autre problème sur lequel nous avons passé pas mal de temps, c'est la représentation de la scène. Nous étions partagé entre peindre l'ensemble de la scène ou utiliser des "widgets". Et comme mentionné un peu plus tôt, on travaillait chacun de notre côté ce qui ne nous a pas aidé à prendre une déci

sion. Finalement notre encadrant nous a suggéré de choisir la première option qui était plus sûr sur le long terme et que tous les jeux étaient implémentés de la sorte.

Après avoir pris ces deux décisions, le reste du projet a put avancer a une vitesse folle en comparaison de ce qui avait ete fait jusqu'à lors. Un dernier problème rencontré a été lors de l'implémentation des bloc semi-réfléchissants et téléporteurs, puisque au départ nous étions parti sur l'idée selon laquelle nous n'aurions pas besoin d'ajouter un nouveau laser pour représenter leur déviation. Au final en tentant de rajouter les "manuellement" dans la liste des points du laser, les déviations des blocs sur son chemin n'était pas calculées, donc finalement nous avons implémenter leur déviation avec l'ajout d'un nouveau laser. Mais là aussi, un nouveau problème a fait surface étant donné que les nouveaux laser ajouter ne se supprimaient pas lors du déplacement des blocs concernés. Pour remédier à ce problème nous avons alors tout simplement initier un entier final qui correspond au nombre de laser de départ.

CONCLUSION:

Malgré de nombreuses péripéties rencontrées tout au long de l'élaboration de notre projet, au final nous avons fini par réussir à répondre au attente de notre cahier des charges. Ce projet a été très intéressant à faire car il nous a permis d'assimiler ce que nous avons appris durant le semestre dernier, il nous a aussi permis d'expérimenter le travail en équipe et de comprendre la complexité du travail en groupe et de son concept, comme le travail en équipe.

Cependant nous avons implémenté qu'une partie des extensions(blocs fixes, bloc plus) que nous avions tenu à faire, ce qui rend notre jeu pas très original au final, et c'est sûrement le seul regrets que nous avons mais sinon dans l'ensemble on a été à la hauteur des attentes.