



Année universitaire 2022-2023

MEMOIRE DE STAGE

Développement de la version Linux d'un logiciel pour des personnes en situation de handicap



Présenté par

Rachel Peretti

Jury

IUT : M. Martin

IUT : Mme Rosset

Société : M. Schwab

Déclaration de respect des droits d'auteurs

Par la présente, je déclare être le seul auteur de ce rapport et assure qu'aucune autre ressource que celles indiquées n'ont été utilisées pour la réalisation de ce travail. Tout emprunt (citation ou référence) littéral ou non à des documents publiés ou inédits est référencé comme tel et tout usage à un outil doté d'IA a été mentionné et sera de ma responsabilité.

Je suis informé qu'en cas de flagrant délit de fraude, les sanctions prévues dans le règlement des études en cas de fraude aux examens par application du décret 92-657 du 13 juillet 1992 peuvent s'appliquer. Elles seront décidées par la commission disciplinaire de l'UGA.

À Saint-Martin-d'Hères, le 18 juin 2024

Signature

A handwritten signature in black ink, appearing to read "R. A." followed by a surname.

Remerciements

Je tiens à remercier M. Didier Schwab et M. Jordan Arrigo pour l'aide précieuse et les conseils apportés au cours de ce stage, ainsi que pour cette opportunité d'apprendre. Je remercie également M. Jérôme Goulian pour la recommandation du stage ainsi que la visite des lieux — enfin, je remercie M. Philippe Martin pour ses opinions sur le rapport de stage et pour sa touche d'humour décontractante !

Table des matières

1	Introduction	6
1.1	Présentation de l'entreprise	6
1.1.1	Organisme d'accueil	6
1.2	Présentation du stage	6
1.3	Cadre général de travail	7
2	Étude de l'existant	7
2.1	Composition du projet GazePlay	7
2.2	Java, JavaFX et Gradle	8
2.3	Git	8
2.4	Tobii	8
3	Réalisations	8
3.1	Portage de l'application sur Linux	8
3.2	Génération de releases	9
3.3	Création d'un nouveau jeu	9
3.3.1	Structure globale	10
3.3.2	Interface graphique	10
	Glossaire	12
A	Planning du stage	13
B	Classe TrainColor : avant modifications	14
C	Classe TrainColor : après modifications	15
D	Méthodes launch() et dispose()	16
E	Exemple de statistiques	17
F	Diagramme de classes	18
	Résumé	18
	Abstract	20

Table des figures

1	Exemple d'utilisation de l'eye-tracker.	6
2	Un Tobii 5.	8
3	Le processus de création d'un exécutable.	9
4	L'inspiration proposée par le contributeur.	9
5	L'interface graphique.	11

Résumé long

Le LIG (Laboratoire d’Informatique de Grenoble) accueille 22 équipes de chercheurs pour 5 axes de recherche, et GETALP (Groupe d’Étude en Traduction Automatique/Traitement Automatisé des Langues et de la Parole) en fait partie, son domaine étant le langage. M. Didier Schwab, membre de l’équipe et du projet InterAACtionBox visant à rendre la Communication Alternative et Augmentée accessible à tous, est le fondateur du projet GazePlay, en collaboration avec l’AFSR (Association Française du Syndrome de Rett). GazePlay est une application de mini-jeux ludiques et éducatifs, conçue pour les personnes en situation de handicap, et se jouant à l’aide d’un Eye-tracker*. Les résultats des parties des utilisateurs peuvent ensuite être étudiés en neuropsychologie par leurs soignants, pour jauger leurs aptitudes. Le langage de développement choisi est Java, ainsi que son équivalent pour les interfaces graphiques JavaFX, et Gradle est utilisé pour effectuer des tests et générer des Release pour distribuer une version stable de l’application à télécharger. Un dépôt GitHub contient le code du projet et est ouvert à tout contributeur — c’est ici que les contributions de code sont soumises à la vérification et aux tests des responsables du projet. Pour tester les jeux avec un eye-tracker, ceux de la marque Tobii sont recommandés et mis à disposition pour le développement.

L’objectif de ce stage est de porter l’application de Windows à Linux, puis de créer un nouveau jeu à intégrer dans l’application. La première étape a été de tester tous les jeux pour voir leur niveau de fonctionnalité, puis de repasser sur ceux qui ne se lancent pas. Un problème a été retrouvé de multiples fois, concernant les liens de fichiers et la manière dont ils sont gérés sur Linux : la casse est plus stricte, et beaucoup d’images et d’éléments nécessaires aux jeux n’étaient plus retrouvés. La modification des lignes de code concernées pour qu’elles soient exactes a suffi à régler le problème. L’ajout de la compatibilité de l’eye-tracker au menu principal ne s’est pas déroulée comme prévu, les choix étant sélectionnés sans aucune action de l’utilisateur. La source du problème s’est avérée complexe, et n’a pas pu être résolue — mais le dernier souci, des caractères en trop dans le fichier des traductions, a été causé par une erreur de formattage. La modification du fichier avec un autre éditeur a fonctionné.

La création d’un nouveau jeu se fait à partir d’une idée proposée par un contributeur sur la page GitHub du projet, et si approuvée par un des responsables, le développement du jeu débute. Ici, l’idée repose sur un groupe de formes comportant une forme différente, et l’utilisateur doit la retrouver, de par sa taille ou de sa couleur. Si elle est trouvée, c’est gagné — sinon, une croix s’affiche sur la forme. Le jeu en lui-même est composé d’une classe OddShapeGame comportant le contenu du jeu, OddShapeGameLauncher pour démarrer des parties du jeu, OddShapeStats pour générer les statistiques qui seront analysées (mouvement des yeux, temps de jeu...), OddShapeGameSpecSource pour définir le titre, les catégories et la miniature du jeu dans le menu principal, et OddShapes contient les formes capables d’être dessinées (Rectangle, Carré, Triangle et Cercle). L’interface graphique est conçue pour être jouée à l’eye-tracker, les formes sont donc de grande taille et de couleurs vives pour être sélectionnées plus facilement.

Ce stage m’a appris l’importance du développement d’une application créée hors de l’IUT et dans un contexte particulier, et je suis heureuse d’avoir pu mettre à l’appui les compétences que j’ai ap- pris pendant mes cours — notamment l’IHM (Interface Homme-Machine) pour la phase graphique de mon jeu, ou encore le travail en équipe à l’aide de GitHub et des réunions hebdomadaires organisées par mes tuteurs. Ce projet me conforte dans mes capacités de développement et me motive pour les objectifs à venir.

1 Introduction

1.1 Présentation de l'entreprise

1.1.1 Organisme d'accueil

Le LIG (Laboratoire d'Informatique de Grenoble) regroupe des équipes de chercheurs, avec 22 équipes pour 5 axes de recherche.[1]

L'équipe GETALP (Groupe d'Étude en Traduction Automatique/Traitement Automatisé des Langues et de la Parole), créée en 2007, en fait partie et étudie le traitement du langage écrit/oral, avec plusieurs sous-axes de recherche : la traduction et transmission automatique de la parole, le traitement et l'analyse d'interactions, ou encore le processus automatique de clarification du sens. GETALP collabore avec d'autres équipes du LIG, mais aussi de nombreux laboratoires grenoblois ou encore à l'international avec un important réseau de contacts, avec par exemple des ambassadeurs en Malaisie ou aux États-Unis.[2]

1.2 Présentation du stage

Le stage est fondé sur le développement de GazePlay, une application du projet InterAACtion-Box en partenariat avec l'AFSR (Association Française du Syndrome de Rett) visant à rendre la communication alternative plus accessible aux personnes en situation de handicap. [3] GazePlay, conçue par M. Didier Schwab, est une application de bureau Open-source, c'est à dire au code librement accessible, réunissant une collection d'à ce jour 75 mini-jeux avec une particularité : ils se jouent principalement à l'aide d'un Eye-tracker, appareil effectuant le mouvement du curseur en s'appuyant sur la position des yeux.¹ L'application est distribuée à un but éducatif et ludique, mais le module GazePlay Eval permet également une évaluation des aptitudes de l'utilisateur, pouvant par la suite être analysée en neuropsychologie.

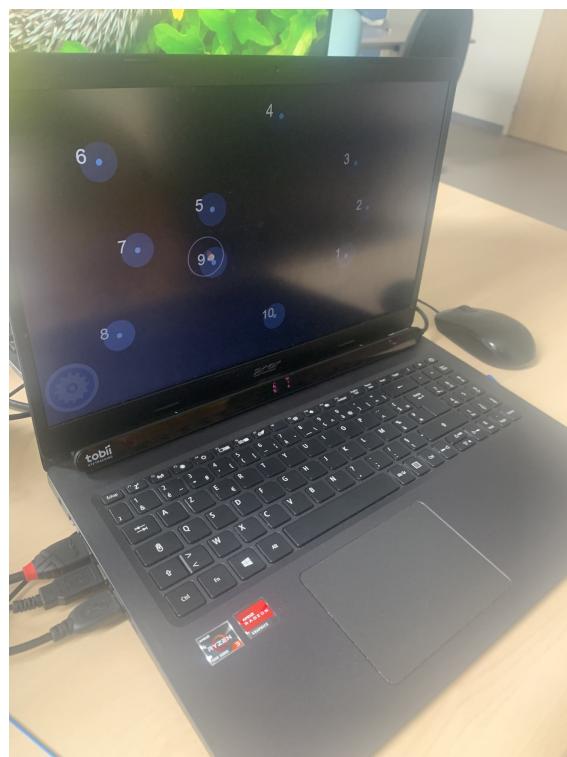


FIGURE 1 – Exemple d'utilisation de l'eye-tracker.

1.3 Cadre général de travail

Les missions du stage sont données par Jordan Arrigo, contributeur du projet GazePlay. Son rôle est également d'étudier le code donné en proposition sur le GitHub de l'application, et de le tester pour être sûr qu'il n'y a aucun problème. Des réunions hebdomadaires sont organisées avec lui et les autres stagiaires pour faire le point sur le travail de la semaine. Un planning ^A a également été prévu pour le déroulé du stage.

2 Étude de l'existant

2.1 Composition du projet GazePlay

Gazeplay est divisé en plusieurs modules, chacun jouant son rôle dans le fonctionnement de l'application.

`gazeplay-commons` contient les classes utilisées dans toute l'application : par exemple le traducteur, changeant les titres et les interfaces en fonction de la langue du système à partir d'un fichier tableur, ou les classes gérant l'intégration de l'eye-tracker.

`gazeplay-core` contient les classes permettant de démarrer le jeu, ainsi que celles des profils utilisateurs et des paramètres. Chaque profil est entièrement personnalisable, avec un nom et une photo de profil, et possède un système de favoris pour que les utilisateurs puissent retrouver les jeux auxquels ils veulent jouer facilement, tout en haut de la liste des jeux. On peut aussi y voir les classes pour la traduction des textes de l'application.

`gazeplay-dist` contient des scripts écrits en Bash pour la génération de distributions / Releases* Linux, et des scripts écrits en Batch pour Windows. Si exécutés, ces fichiers installent le nécessaire pour pouvoir lancer GazePlay sur l'appareil.

`gazeplay-games` est composé entièrement des jeux de l'application. Un dossier "games" contient tous les jeux en sous-dossiers, et un autre dossier "resources" contient les images et sons nécessaires au fonctionnement des jeux.

`gazeplay-games-commons` joue le même rôle que `gazeplay-commons`, mais pour les jeux : les classes les plus utilisées y sont répertoriées, comme celle gérant l'animation utilisée lors de la victoire d'un utilisateur ou celle implémentant la notion de cycle de vie d'un jeu — c'est à dire les méthodes de début et de fin.

`gazeplay-melordi` et `gazeplay-picto-pick` sont des projets n'ayant pas un grand nombre d'informations à leur sujet — mais qui se basent respectivement sur les activités musicales et les pictogrammes.

`gazeplay-side-projects` possède également peu d'informations, mais on peut y trouver des dossiers suggérant une utilisation potentielle de cartes [Arduino](#) : des cartes électroniques souvent utilisées dans des projets de robotique et permettant une certaine flexibilité, pouvant être intégrées dans tous types de projets.

Sur les modules du projet, ceux faisant le sujet du stage sont `gazeplay-core` et `gazeplay-games`. Tout code mentionné en fait partie.

2.2 Java, JavaFX et Gradle

L’application est développée sous Java, et JavaFX est le moteur d’interface graphique choisi. Gradle est utilisé ici pour générer des tests, des constructions de code, mais son utilisation principale est pour les Release du jeu, versions stables prêtes à être utilisées.

2.3 Git

GitHub est utilisé par les contributeurs pour partager du code. Une branche "developMain" contient le code destiné aux releases, et les développeurs créent leurs propres branches sur le projet pour rajouter des fonctionnalités. Elles sont ensuite intégrées dans la branche principale à l’aide de Pull requests, demandes de contribution de code étudiées par les responsables du projet puis éventuellement fusionnées.

2.4 Tobii

Des Eye-trackers* Tobii sont utilisés par les développeurs afin de vérifier la compatibilité des jeux, et il est recommandé aux utilisateurs de prendre les mêmes. Les éditions 4C et 5² sont mises à disposition pour le stage.



FIGURE 2 – Un Tobii 5.

Des pilotes sont également à installer pour permettre à l’ordinateur de convertir l’activité des yeux de l’utilisateur aux mouvements du curseur. Le logiciel utilisé, Tobii Eye Tracking Core[4], n’est pas compatible avec ma distribution de Linux — je l’ai donc uniquement utilisé sur Windows, la souris faisant substitut de l’Eye-tracker durant le développement.

3 Réalisations

3.1 Portage de l’application sur Linux

L’application étant créée pour et sur des plateformes Windows, elle ne marche pas sur Linux si des modifications ne sont pas appliquées. Pour identifier les changements à réaliser, j’ai effectué un test individuel de chaque jeu. Le portage n’a pas été simple et j’ai rencontré de nombreux obstacles.

Un premier problème est que Linux ne gère pas les liens de fichiers de la même manière que Windows, car plus strict sur la casse, et les fichiers ne sont alors pas trouvés — il manque donc des images ou des sons, et les jeux ne fonctionnent plus. Dans ce cas il faut parfois modifier des classes entières, et le jeu TrainSwitches en est un exemple, avec la classe TrainColor^B : l’utilisation des valeurs se fait en les incorporant dans des liens de fichier, où le même problème de gestion de liens se retrouve. J’ai ajouté quelques lignes de code pour régler le problème.^C

Un autre problème avec Linux est arrivé lors d'une de mes missions : lors de l'intégration de l'Eye-tracker* au menu principal, la sélection de jeux se faisait toute seule, sans aucun mouvement de la part de l'utilisateur. J'ai découvert le problème, mais après de longues vérifications, n'ai pas trouvé sa source.

Enfin, lors d'une de mes modifications au fichier tableau de langues du jeu pour ajouter des traductions manquantes, il y a eu une erreur de formatage, rajoutant des caractères en trop et rendant le fichier illisible pour le code. J'ai donc modifié le fichier à la main dans l'éditeur de code plutôt qu'avec un logiciel dédié.

3.2 Génération de releases

Lors du développement de GazePlay, il est important de générer des Releases* régulièrement afin de pouvoir vérifier l'absence de problèmes sur une réelle installation du jeu, et non en développement. Pour cela, une action Gradle est utilisée pour créer des Releases* pour Windows, Linux et MacOS.

Pour Windows, lors de la génération Gradle, [InnoSetup](#) est utilisé : un logiciel libre donnant à son utilisateur la possibilité de créer des installateurs. Ici, j'ai simplement utilisé les actions Gradle mises à ma disposition : je n'ai pas configuré l'action ni InnoSetup.

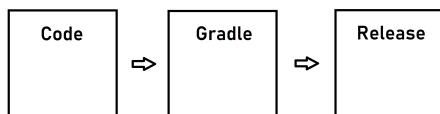


FIGURE 3 – Le processus de création d'un exécutable.

Le code passe dans l'action Gradle, qui génère un exécutable d'Innosetup. L'installateur, ayant récupéré auprès de l'utilisateur le dossier où installer le jeu, va mettre les fichiers aux emplacements nécessaires et générer un fichier exécutable de Gazeplay.³

3.3 Crédit à un contributeur

Lors de la création d'un jeu à ajouter à Gazeplay, une idée venant d'une Issue GitHub, une suggestion d'un utilisateur, est généralement reprise. Ici, le jeu que j'ai choisi a été soumis par un contributeur, et où l'utilisateur doit trouver la forme différente des autres, nommé OddShape.⁴ Le jeu propose à l'utilisateur de s'entraîner sur le Pattern Matching, et de tester sa rapidité.

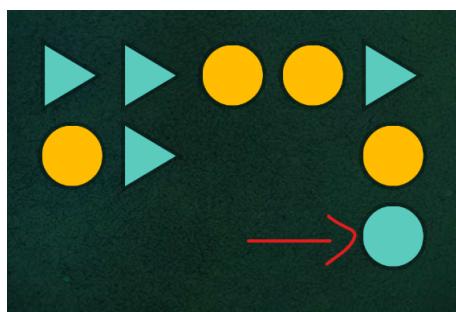


FIGURE 4 – L'inspiration proposée par le contributeur.

3.3.1 Structure globale

Chaque jeu est composé d'un nom, d'une miniature, de catégories, et de différentes classes :

OddShapes est un enum possédant en valeur toutes les formes qu'il est possible de créer : RECTANGLE, CERCLE, TRIANGLE, et CARRE.

OddShapeGame représente le jeu en lui-même, c'est à dire que j'ai codé le contenu du jeu à l'intérieur. Généralement, un jeu a besoin d'implémenter l'interface GameLifeCycle, donc un cycle de vie, et d'avoir obligatoirement deux méthodes launch() et dispose() : respectivement le début et la fin du jeu. Toutes les méthodes à utiliser lors de l'utilisation du jeu sont appelées dans launch(), et celles qui "nettoient" le jeu dans dispose().^D

Mon jeu est construit en utilisant des objets Shape, donc des formes, qui sont natives à JavaFX. Chaque forme possède un contour, une couleur, et peut se décliner en plusieurs sous-catégories : celles utilisées dans le jeu sont Circle, Rectangle et Polygon.

Pour créer une partie, plusieurs fonctions sont appelées. selectShape() choisit une forme parmi une liste de formes valides à l'aide de l'enum OddShapes, createShape() crée un objet Shape basé sur cette forme, puis createShapeGroup() compose un groupe avec des copies de cette forme, en utilisant duplicateShape(). Une forme différente est également créée dans une fonction createBadShape(), en réutilisant la même liste qu'auparavant, mais sans la forme déjà utilisée.

Lors de la création de la forme "mère", une couleur aléatoire est choisie avec selectColor() pour définir la couleur du groupe. La forme différente prend également une couleur de la liste, et peut donc être de la même couleur que le groupe de formes, rajoutant de la difficulté au jeu.

Toutes ces formes sont conservées en mémoire puis affichées. Lorsque la bonne forme est trouvée, l'animation de victoire se joue. En revanche, si le joueur a faux, une croix s'affiche.

OddShapeGameLauncher permet de lancer la partie lorsque le jeu est sélectionné dans le menu principal. Elle contient quatre méthodes : createNewStats(), createSavedStats(), createNewGame(), et replayGame().

OddShapeGameSpecSource implémente l'interface GameSpecSource — elle ne possède qu'une méthode getGameSpec(), qui construit l'identité du jeu en lui donnant un nom, une miniature et des catégories pour la page d'accueil.

Enfin, OddShapeStats génère des statistiques ^E sur l'utilisation de la souris et de l'Eye-tracker durant le jeu : elles sont affichées à la fin de la partie, et ensuite étudiées par des soignants pour voir où l'utilisateur a regardé et les actions effectuées.

Un diagramme de classes^F est disponible en annexe.

3.3.2 Interface graphique

Lorsque j'ai conçu l'interface graphique du jeu, il a été nécessaire de prendre en compte les utilisateurs cibles : ici les personnes en situation de handicap. Les éléments visuels doivent donc être compatibles avec l'utilisation d'un eyetracker, que cela soit de par leur taille, leur position, ou leur couleur. Dans le cas de mon jeu, les formes sont grandes et colorées pour être sélectionnables facilement. Seuls les paramètres du jeu ne respectent pas ces critères, pour éviter que le joueur ne sorte du jeu ou fasse des mauvaises manipulations — un accompagnateur est généralement présent pour les aider.

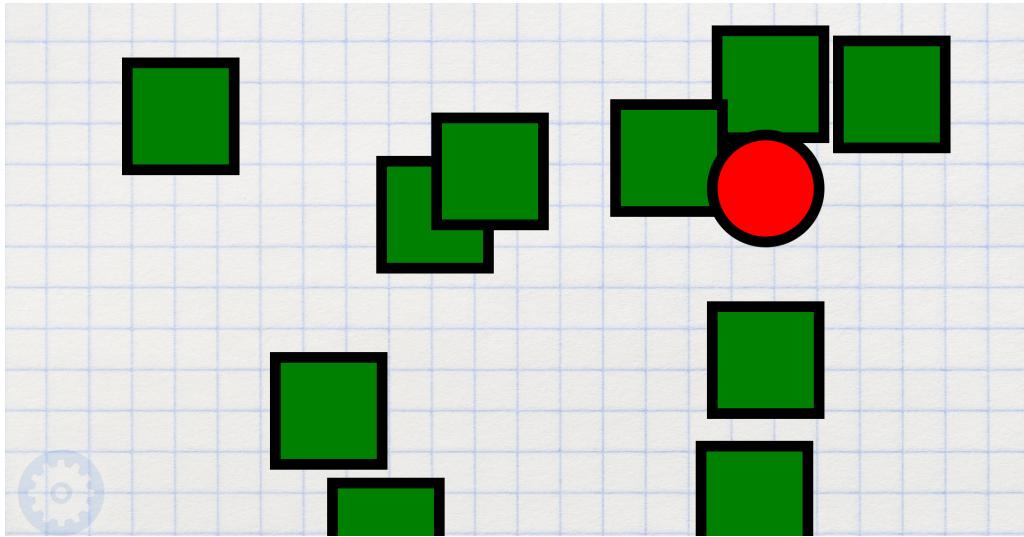


FIGURE 5 – L'interface graphique.

Conclusion

Je suis satisfaite du travail effectué sur ce projet et de ce que j'ai pu y apporter, et fière des efforts que j'y ai mis — à la hauteur d'environ 400 lignes de code. A l'heure de l'écriture de ce rapport, il ne me reste qu'à finaliser le jeu OddShape en cours et le peaufiner avant de le soumettre à mon tuteur.

Bilan

Le projet GazePlay m'a permis d'apprendre beaucoup de choses sur ce qu'est le développement d'une application "sur le terrain" — notamment avec la gestion de projet avec Git et l'importance de contrôler et tester chaque morceau de code avant de le publier en production, mais aussi le fait d'avoir des objectifs précis et une attente spécifique du travail à rendre à ses supérieurs.

Des compétences acquises à l'IUT ont également pu être mises à l'appui, telles que l'IHM (Interface Homme-Machine) sur l'ergonomie de l'apparence graphique de mon jeu et l'importance que cela a sur le confort des utilisateurs, ou le travail en équipe à l'aide d'outils collaboratifs comme GitHub. Le stage m'a permis de développer ces connaissances et d'être confiante en mes capacités de développement dans un contexte non-éducatif.

Cette expérience professionnelle me motive à continuer dans cette voie, afin d'aider à développer des applications qui peuvent servir à des utilisateurs en ayant besoin.

Glossaire

Communication Alternative et Augmentée Moyen alternatif pour les personnes ayant besoin de communiquer autrement qu'avec la parole, par exemple un programme permettant à l'utilisateur de choisir des mots qui seront ensuite dits à voix haute par l'appareil.. 5

Eye-tracker Appareil utilisant la position des yeux pour diriger un curseur d'ordinateur.. 5, 6, 8, 9, 10

Getter Fonction permettant de récupérer la valeur d'un attribut.. 15

Issue Une fonctionnalité de GitHub permettant de faire savoir aux développeurs si ils ont trouvé des problèmes ou des suggestions pour leur programme.. 9

Open-source Programme au code librement accessible sur Internet, et où les contributions sont les bienvenues.. 6

Pattern Matching Traduit littéralement par "correspondance de modèles", est le fait de savoir reconnaître des motifs répétitifs.. 9

Pull request Une fonctionnalité de GitHub permettant de contribuer à un programme en proposant de rajouter du code.. 8

Release Version stable, non en développement, d'une application.. 5, 7, 8, 9

Annexes

A Planning du stage

Planning de travail prévu

(détaillez par période de 1 à 2 semaine(s) en précisant les dates)

Période	Travail à réaliser
15/04-19/04	Initiation au projet GazePlay (lecture du code et découverte de l'existant)
22/04-03/05	Listing des bugs existant + fonctionnalités manquantes entre la version Linux et Windows
05/05-24/05	Correction et ajout de fonctionnalité pour la version Linux
27/05-01/06	Préparation du rapport préliminaire
03/06-07/06	Préparation du mémoire de stage
10/06-14/06	Amélioration de la version Linux
17/06-21/06	Soutenance de Stage + mise à jour du guide utilisateur de Gazeplay (Linux)
24/06-05/07	Résolution d'"issues" + Correction de bug
15/04-05/07	Révision de "Pull Request" Github jusqu'à validation (Merge)

B Classe TrainColor : avant modifications

Le constructeur de la classe est vide et son contenu n'est pas appliqué à un attribut color.

```
1 package net.gazeplay.games.trainSwitches;
2
3 public enum TrainColors {
4     BLACK("black"),
5     BROWN("brown"),
6     DARKBLUE("darkBlue"),
7     DARKGREEN("darkGreen"),
8     GREEN("green"),
9     GREY("grey"),
10    LIGHTBLUE("lightBlue"),
11    ORANGE("orange"),
12    PINK("pink"),
13    PURPLE("purple"),
14    RED("red"),
15    WHITE("white"),
16    YELLOW("yellow");
17
18    TrainColors(final String color){
19        }
20
21 }
```

C Classe TrainColor : après modifications

La classe contient à présent un constructeur minimal, un attribut color ainsi qu'un Getter* pour récupérer la valeur d'une couleur.

```
1 package net.gazeplay.games.trainSwitches;  
2  
3 public enum TrainColors { 29 usages ± HugolinBouhineau +1  
4     BLACK("black"),  
5     BROWN("brown"), 1 usage  
6     DARKBLUE("darkBlue"), 1 usage  
7     DARKGREEN("darkGreen"), 1 usage  
8     GREEN("green"),  
9     GREY("grey"), 1 usage  
10    LIGHTBLUE("lightBlue"), 3 usages  
11    ORANGE("orange"),  
12    PINK("pink"), 2 usages  
13    PURPLE("purple"), 2 usages  
14    RED("red"),  
15    WHITE("white"),  
16    YELLOW("yellow");  
17  
18    private final String color; 2 usages  
19  
20    > TrainColors(final String color) { this.color = color; }  
21  
22    > public String getColor() { return color; }  
23  
24    }  
25  
26  
27 }
```

D Méthodes launch() et dispose()

```
//méthode pour démarrer le jeu
@Override  ↗lynirl
public void launch() {

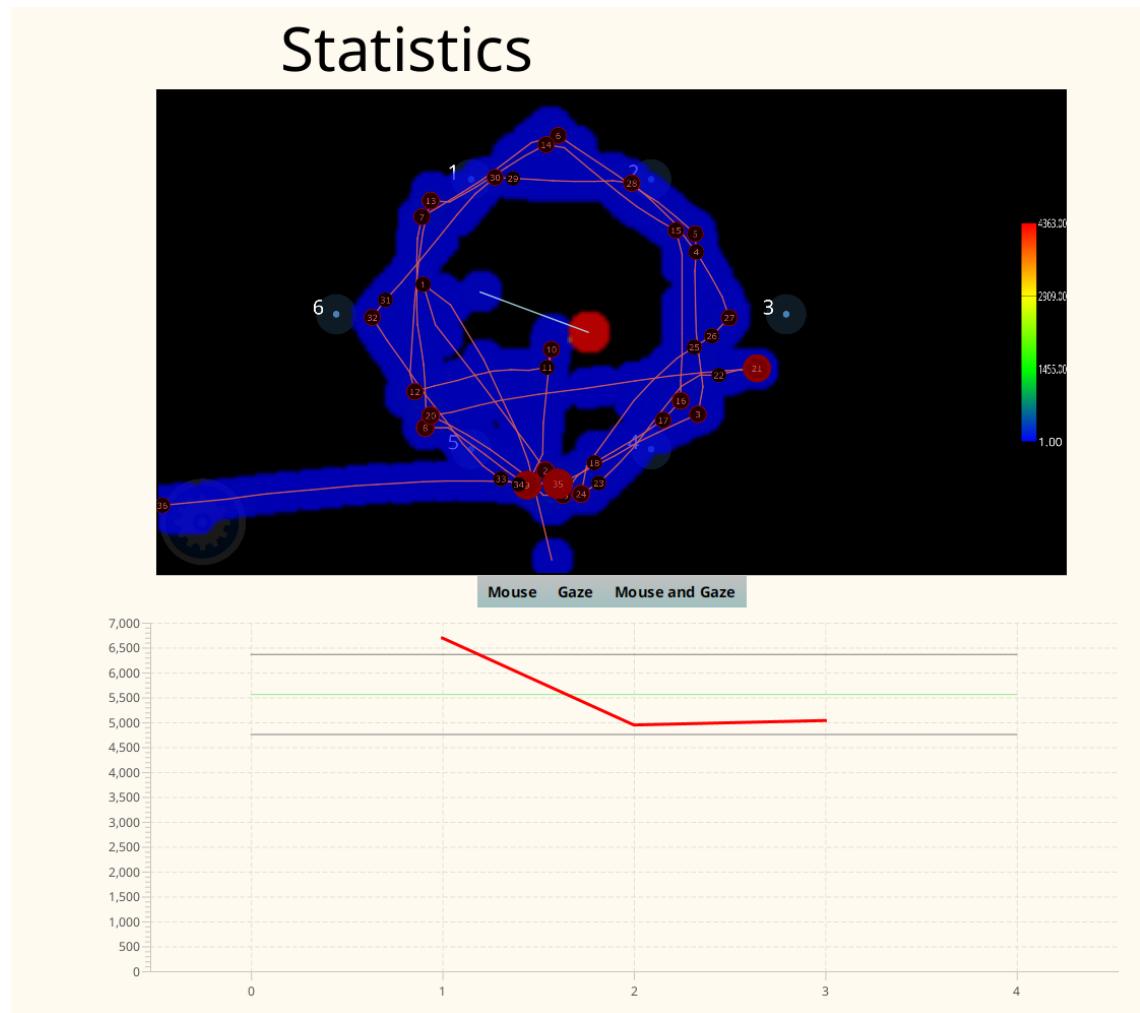
    //afficher background
    createBackground();
    //initialiser les couleurs
    initialiseColors();
    //sélectionner une forme
    baseShape = selectShape();
    //puis faire le groupe de bonnes formes
    goodShapes = createShapeGroup(createShape(baseShape));
    //et générer la mauvaise
    badShape = createBadShape();
    drawShapes();

}

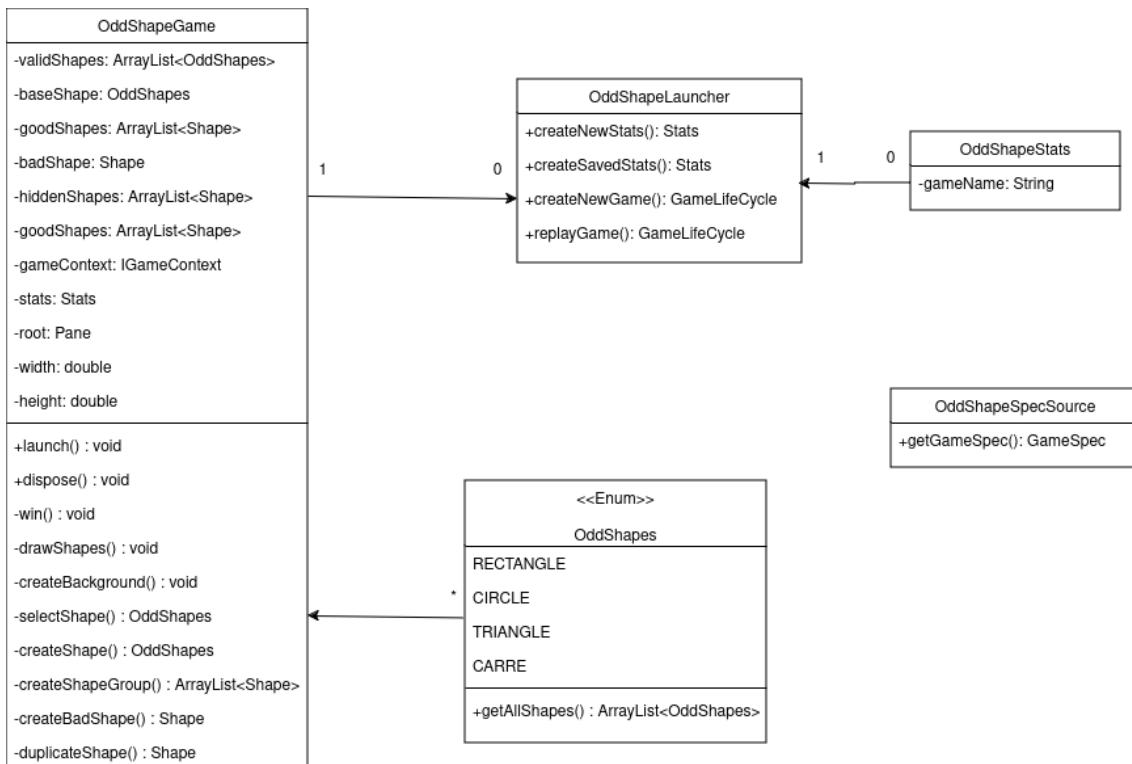
//méthode pour tout nettoyer
@Override  ↗lynirl
public void dispose() {
    //on réinitialise les formes possibles
    validShapes = OddShapes.getAllShapes();
    //et on enlève tout le monde
    root.getChildren().removeAll();
}
```

E Exemple de statistiques

Les statistiques récupérées. Non affichés sont le temps passé sur le jeu, le temps d'activité sur le jeu, et la déviance du regard par rapport aux éléments du jeu.



F Diagramme de classes



Références

- [1] Laboratoire d'Informatique de Grenoble, [Présentation, Axes de Recherche](#), [\[en ligne\]](#), consulté le 30/05/24
- [2] GETALP, [Accueil](#), [\[en ligne\]](#), consulté le 30/05/24
- [3] InterAACTIONBox, [A propos, Partenaires](#), [\[en ligne\]](#), consulté le 30/05/24
- [4] Tobii, [Downloads](#), [\[en ligne\]](#), consulté le 03/06/24

Résumé

Les personnes en situation de handicap possèdent généralement peu ou aucun moyen de jouer à des jeux vidéos. Le projet GazePlay, dirigé par l'équipe InteraactionBox et en partenariat avec l'AFSR et l'UGA, mène à donner aux enfants n'ayant pas l'usage de leur corps un moyen de continuer à s'amuser, mais aussi d'apprendre de manière ludique. Pour cela, les jeux sont développés pour l'usage d'un eye-tracker, appareil captant l'activité des yeux pour les retranscrire en mouvements de curseur. Il est donc important que les jeux soient compatibles avec cet usage, donc avec des éléments colorés et de taille. Les concepts des jeux restent simples, avec du coloriage, un casse-briques ou encore des opérations comme des additions ou des soustractions, et durant le stage un nouveau jeu a été développé : OddShape, jeu de pattern-matching où il faut trouver la forme différente parmi un groupe. Cependant, pour que l'application soit accessible à tous, il est nécessaire de la rendre disponible sur de multiples plateformes : GazePlay est disponible sur

Windows, mais ne l'est qu'à moitié sur Linux, à cause de problèmes de compatibilité avec l'eye-tracker ou une gestion différente de chemins de fichiers. Il a été nécessaire de modifier des classes afin de garder les jeux fonctionnels, et à présent GazePlay Linux est à jour avec sa contrepartie.

Mots clés : eye-tracker, pattern-matching

Abstract

GazePlay: Developing Educative Games for Disabled Children Playable with Eye Trackers

Rachel Peretti

Abstract: Nowadays, video games aren't developed with disabled children in mind, and it is difficult for them to be entertained, especially if they have a body disability. InterAACtion-Box's GazePlay aims to provide them educative games so that they can still play while developing various skills in math, drawing, etc. The games are specifically developed to be played with an eye-tracker, which transforms eye activity into cursor movement, and then into the application. To make the games more available, the elements of each game are big and colorful, and the concepts aren't too complex — such as dot-to-dot, tower defense, or potion-making games, and even a pet simulator. However, the usage of GazePlay isn't exclusively reserved to children, and an older audience can very well enjoy playing: feedback was received from an elderly man who had recovered a great amount of eye movement thanks to a daily usage of the games, and from a student who simply enjoyed trying out his new eye-tracker with the app. We discuss how to make games compatible for eye-trackers and what it means for their development.

Keywords : Eye-tracker, disability, educative games, tower defense,