

DÉPARTEMENT INFORMATIQUE - IUT 2 GRENOBLE



Année Universitaire 2022-2023

MÉMOIRE DE STAGE

**DÉVELOPPEMENT, AMÉLIORATION ET IMPLÉMENTATION DE
JEUX SÉRIEUX ET FONCTIONNALITÉS DANS UN LOGICIEL POUR
DES PERSONNES EN SITUATION DE HANDICAP**

Laboratoire d'Informatique de Grenoble (LIG)



Présenté par

Noha Boutemeur

Jury

IUT : M. Giuffrida

IUT : M. Corset

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.

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.

A, Grenoble

Le, 01/06/2023



Remerciements

Je remercie Jérôme Goulian de m'avoir conseillé de faire mon stage dans cette entreprise. Je tiens à exprimer ma reconnaissance à Didier Schwab, mon responsable de stage, de m'avoir permis de faire mon stage au LIG. Je remercie également Jordan Arrigo qui est ingénieur d'étude et qui m'a beaucoup aidé durant mon stage. Je tiens également à remercier Tanguy Giuffrida, mon tuteur de stage, de son accompagnement tout au long du stage. Pour finir, je tiens à remercier Paul Sode, Yanis Girardin et Aubin Mouras qui sont tous les trois stagiaires avec moi pour l'entraide générale.

Résumé long

Dans le cadre de mon stage, j'ai été intégré dans l'équipe GETALP. Les recherches de ce groupe sont centrées sur le développement de la communication et du traitement de l'information multilingue. Un des projets de cette équipe, nommé [InterAActionBox](#)*, a pour but de favoriser l'apprentissage et la communication des personnes en situation de handicap cognitif. Ce projet propose plusieurs logiciels de [Communication Alternative et Augmentée \(CAA\)](#)* dans lequel [GazePlay](#)* figure. Il s'agit d'un logiciel proposant près de 60 jeux sérieux jouable avec un [oculomètre](#)* à destination de personnes en situation de handicap. Mon stage consistait à développer, améliorer ou implémenter des jeux sérieux ou fonctionnalités.

[GazePlay](#) comprend trois objectifs spécifiques à atteindre. Tout d'abord, faciliter l'interaction oculaire en permettant aux utilisateurs de contrôler les mini-jeux à l'aide de leur regard. Ensuite, offrir une variété de jeux adaptés aux besoins et capacités des personnes en situation de handicap cognitif et qu'ils soient à la fois divertissants et éducatifs. Pour finir, intégrer des fonctionnalités de [CAA](#)* comme des pictogrammes, des symboles ou des phrases préenregistrées pour faciliter l'expression et la compréhension.

Étant une application visant à être utilisée par des enfants, le design de cette dernière est adapté en arborant un thème très coloré, l'essentiel du visuel est très enfantin. Par ailleurs, le fait que la plupart des éléments de même type comme les jeux du catalogue sont regroupés au même endroit et permet donc une utilisation beaucoup plus compréhensible et rapide.

Pour tout ce qui est relatif au partage de fichier pour le code ou pouvoir récupérer le projet, nous utilisons Github. Concernant la communication, nous utilisons plutôt Discord. Le langage de programmation de GazePlay est Java/JavaFX.

J'ai toujours été relativement en avance sur le planning prévisionnel. Initialement, je devais consacrer 4 semaines à la création et amélioration d'un jeu sérieux. Cependant, ayant fini le jeu beaucoup plus tôt, je me suis mis à en créer d'autres.

Au total, j'ai créé trois jeux sérieux. Le premier est un jeu coopératif se nommant "Sprint to the finish" (ou "Sprint jusqu'à l'arrivée") et consiste à jouer un chat qui doit aller à sa gamelle tout en esquivant des chiens le poursuivant. Le deuxième joueur doit aider le chat en fixant du regard des interrupteurs qui permettent d'ouvrir des portes ou en fixant les chiens pour les immobiliser.

Le deuxième jeu est un jeu de survie qui consiste à esquiver des robots qui se déplacent aléatoirement et qui peuvent tirer sur le joueur. Naturellement, le nom de ce jeu est "Survive against robots" (ou "Survivre contre des robots") et le but est de survivre le plus longtemps pour avoir le meilleur score possible. Plusieurs bonus temporaires sont disponibles pour aider le joueur comme par exemple pouvoir ralentir les robots, devenir temporairement invincible ou bien tirer plus rapidement.

Pour finir, le troisième jeu est un jeu musical s'inspirant énormément du jeu de société Simon. Le principe est de reproduire les touches que la borne fait le tout produisant des notes de musique. Le nombre de touches avant de gagner la partie varie selon le niveau de difficulté allant de 10 pour le plus facile à 32 pour le plus dur.

Table des matières

I. Introduction.....	7
II. Analyse des besoins.....	8
II.1 Objectif.....	8
II.2 Étude de l'existant.....	8
III. Interface graphique.....	9
III.1 Présentation de l'interface de l'application.....	9
III.2 Analyse ergonomique.....	10
IV. Réalisation.....	12
IV.1 Gestion de projet.....	12
IV.2 Phases d'implémentation.....	12
Création de nouveaux jeux sérieux.....	12
Création de nouvelles fonctionnalités.....	17
Correction de bugs.....	18
Difficultés.....	18
Conclusion.....	20
Glossaire.....	21
Annexes.....	24
Sitographie.....	27

Table des figures

Figure 1 : Page de connexion de GazePlay	9
Figure 2 : Page principale de GazePlay	9
Figure 3 : Page des paramètres de GazePlay	10
Figure 4 : Image du catalogue de jeu	11
Figure 5 : Image des boutons de catégorie	11
Figure 6 : Image des trois boutons	11
Figure 7 : Image du jeu Sprint To The Finish - level 16	13
Figure 8 : Image du robot qui ne tire pas	14
Figure 9 : Image du robot qui tire	14
Figure 10 : Bonus ralentissement	15
Figure 11 : Bonus fréquence de tir	15
Figure 12 : Bonus d'invincibilité	15
Figure 13 : Image du jeu Survive Against Robots	15
Figure 14 : Borne du jeu Simon	16
Figure 15 : Sélection des niveaux du jeu Rush Hour	17
Figure 16 : Image du jeu Bubble	17

I. Introduction¹

Le stage s'est produit au Laboratoire d'Informatique de Grenoble (LIG), un laboratoire de recherche créé le 1er janvier 2007, dirigé aujourd'hui par Noël DE PALMA, Sihem AMER YAHIA, Patrick REIGNIER et Pascale POULET. D'après son livret de 2021 [1], le LIG regroupe environ 450 chercheurs, enseignants-chercheurs, doctorants et personnels en soutien à la recherche. Il est de plus en partenariat avec l'Université Grenoble-Alpes (UGA), l'INP (Institut Polytechnique) de Grenoble, le Centre National de la Recherche Scientifique (CNRS) et l'Institut National de Recherche en Informatique et en Automatique (INRIA). Le laboratoire est étendu sur trois sites : un sur le campus, au sein du bâtiment IMAG (Institut d'informatique et mathématiques appliquées de Grenoble), un sur le campus Minatec et un autre à Montbonnot. Le LIG se veut un acteur actif à la recherche et au développement des différents domaines de l'informatique. Il comporte au total 22 équipes portant différentes recherches et projets selon cinq axes thématiques de recherche, qui sont les suivants : Génie des logiciels et des systèmes d'informations ; Méthodes formelles, modèles et langages ; Système intelligents pour les données, les connaissances et les humains ; Systèmes interactifs et cognitifs ; Systèmes répartis, calcul parallèle et réseaux. Sur l'organigramme disponible en [annexe \(1\)](#), nous pouvons voir les liens entre les différents acteurs du LIG.

Dans le cadre de mon stage, j'ai été intégré dans le Groupe d'Étude pour la Traduction Automatique et le Traitement Automatisé des Langues et de la Parole (GETALP) menant des recherches sur les Systèmes intelligents pour les données, les connaissances et les humains. Ses recherches sont centrées sur le développement de la communication et du traitement de l'information multilingue.

Un des projets de cette équipe, nommé [InterAACTIONBox](#)*, a pour but de favoriser l'apprentissage et la communication des personnes en situation de handicap cognitif. Plus particulièrement, il intègre un ordinateur ou une tablette permettant des interactions oculaires et tactiles, un système d'exploitation qui gère [InterAACTIONBox](#)* et plusieurs logiciels de [Communication Alternative et Augmentée \(CAA\)](#)*. La [CAA](#)* représente l'ensemble des techniques et outils permettant aux personnes ayant des difficultés pour communiquer de s'exprimer ou de comprendre les autres.

Le sujet de mon stage portait sur l'un de ces logiciels appelé [GazePlay](#)*. Il s'agit d'un logiciel libre et gratuit qui rassemble plusieurs jeux sérieux jouables grâce à un [oculomètre](#)*. Un [oculomètre](#)* (Eye-tracker) est disponible en [annexe \(2\)](#) et est un appareil permettant de mesurer les mouvements des yeux. La dernière version compte près de 60 jeux. Les objectifs du stage ont été définis par un planning prévisionnel disponible en [annexe \(3\)](#). Dans ce planning, la première semaine faisait office d'introduction dans le stage. Elle consistait à m'initier à ce projet notamment en me familiarisant avec le code et en implémentant quelques petites nouvelles fonctionnalités. Une fois cette introduction faite, la deuxième partie du stage consistait à créer des nouveaux jeux sérieux de zéro et d'améliorer d'autres jeux existants.

Dans un premier temps, ce rapport présentera une analyse des besoins portants sur le projet [InterAACTIONBox](#)* et plus particulièrement sur [GazePlay](#)*. Ce rapport portera aussi sur l'interface graphique du logiciel. Pour finir, le rapport traitera des différentes réalisations que j'ai pu produire dans ce projet allant de corrections de bug et d'ajouts de nouvelles fonctionnalités jusqu'à la création de nouveaux jeux.

¹ La présentation de l'entreprise a été rédigé avec l'aide de Paul et Yanis

II. Analyse des besoins

II.1 Objectif

Les objectifs principaux du projet [InterAActionBox](#)* visent à atteindre les fonctionnalités essentielles requises pour favoriser l'apprentissage et la communication des personnes en situation de handicap cognitif le tout en proposant des outils le plus libre. L'un de ces projets est un logiciel, [GazePlay](#)*, qui utilise un oculomètre pour permettre aux utilisateurs de jouer à des mini-jeux interactifs.

Le développement de [GazePlay](#)* comprenait trois objectifs spécifique à atteindre :

1. Faciliter l'interaction oculaire : Le logiciel doit prendre en charge les interactions oculaires, permettant aux utilisateurs de contrôler les mini-jeux à l'aide de leur regard.
2. Offrir une variété de jeux adaptés : Le logiciel doit proposer une sélection de mini-jeux adaptés aux capacités et aux besoins des personnes en situation de handicap cognitif. Ces jeux doivent être à la fois divertissants et éducatifs, afin de favoriser l'apprentissage et la stimulation cognitive.
3. Intégrer des fonctionnalités de [CAA](#)* : Le logiciel doit permettre aux utilisateurs de communiquer en utilisant des techniques et des outils de [CAA](#)*. Cela peut inclure des fonctionnalités telles que des pictogrammes, des symboles ou des phrases préenregistrées pour faciliter l'expression et la compréhension.

Mon objectif durant ce stage est d'améliorer ce logiciel en proposant des nouvelles fonctionnalités ou des nouveaux jeux.

II.2 Étude de l'existant

D'après un rapport de l'UNICEF [2] publié le 10 novembre 2021, le nombre d'enfants handicapés dans le monde est estimé à près de 240 millions soit près de 1 sur 10. Dans ce cadre-là, [InterAActionBox](#)* propose plusieurs logiciels en commençant par [GazePlay](#)*. Il y a également [AugCom](#)*, un site internet permettant la création de grilles de communication alliant images et mots et dont un exemple est disponible en [annexe \(4\)](#). Pour finir, [InterAActionScene](#)* est également un site internet qui permet à l'utilisateur de créer des "scènes" ou liste d'images, de leur attribuer des images et d'y placer des dessins ou des [hotspots](#)* qui sont des zones que l'on peut placer et qui produisent des sons choisis par l'utilisateur lors de leur clique.

Le langage de programmation de [GazePlay](#)* est Java et on utilise [JavaFX](#)* qui est un framework pour Java qui permet d'intégrer des graphiques vectoriels, des animations, du son et des vidéos assez facilement. Un [framework](#)* fournit une bibliothèque de fonctionnalités spécifiques. De plus, la version de java utilisée est [Azul](#)*, qui est une distribution gratuite, mise à jour fréquemment et dont le support commercial est autorisé en plus d'avoir une licence permissive. [L'IDE](#)* que l'on utilise est IntelliJ Idea Ultimate version. Un [environnement de développement intégré \(IDE\)](#)* [6] est une application logicielle qui aide les programmeurs à développer efficacement le code logiciel. On utilise précisément la version Ultimate parce que [Spring](#)* [4], qui est un [framework](#)* d'application pour la plateforme Java, n'est pas supporté dans la version Community.

III. Interface graphique

III.1 Présentation de l'interface de l'application

L'application [GazePlay*](#), destinée aux enfants, présente un design adapté qui se caractérise par des couleurs vives et l'utilisation de multiples nuances. Elle est composée de trois pages principales, en dehors des jeux et des fenêtres contextuelles (pop-ups). Lorsque l'application est lancée, l'utilisateur est d'abord dirigé vers une page où il peut créer un compte utilisateur, se connecter avec un profil existant ou utiliser un profil par défaut, comme illustré dans la Figure 1.



Figure 1 : Page de connexion de GazePlay

Après connexion, la page principale nous est proposée, listant les jeux du catalogue. Comme on peut le voir à la Figure 2 :

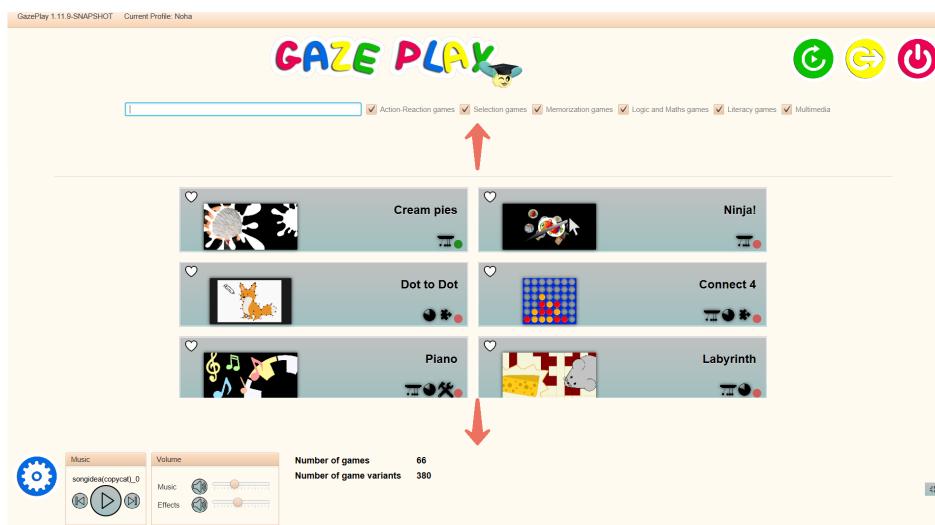


Figure 2 : Page principale de GazePlay

Depuis cette interface, en cliquant sur le bouton tout en bas à gauche, nous avons accès aux paramètres de l'application, voir Figure 3 :

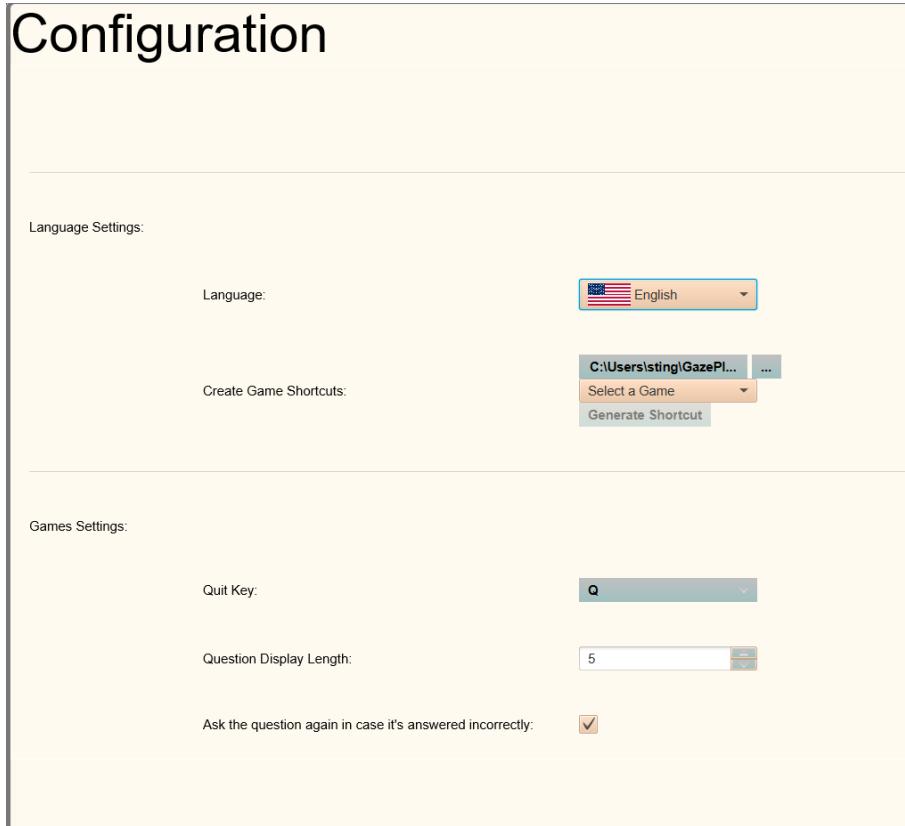


Figure 3 : Page des paramètres de GazePlay

III.2 Analyse ergonomique

Pour faire une analyse ergonomique de l'application, nous allons nous baser sur les [critères ergonomiques de Bastien et Scapin*](#) [3].

Les chercheurs en psychologie ergonomique et en ergonomie cognitive, Christian Bastien et Dominique Scapin, se sont focalisés sur l'expérience utilisateur et les Interfaces Humain-Machine ([IHM](#))*. [L'IHM](#)* représente les outils et moyens mis en œuvre pour qu'une personne puisse communiquer ou contrôler une ou plusieurs machines.

Dans leur article pionnier publié en 1993, Bastien et Scapin ont introduit un modèle ergonomique visant à évaluer l'utilisabilité des [IHM](#)*. Ce modèle repose sur huit dimensions clés : guidage, charge de travail, contrôle explicite, adaptabilité, gestion des erreurs, homogénéité, signifiance des codes et compatibilité. De plus, certaines dimensions clés possèdent des sous-critères. Selon ces chercheurs, ces dimensions fournissent une structure pour appréhender l'utilisabilité des [IHM](#)* et établissent des critères précis permettant de s'assurer que l'interface répond aux besoins et aux capacités des utilisateurs. Leur approche est aujourd'hui largement reconnue comme une référence majeure en ergonomie des [IHM](#)* et dans l'évaluation de l'utilisabilité, car elle propose un cadre rigoureux et exhaustif pour analyser et décomposer les facteurs humains impliqués dans la conception d'interfaces utilisables.

Nous allons nous pencher sur les critères ergonomiques les plus importants pour cette application qui sont respectés.

En premier lieu, nous allons parler du guidage de l'application. Le guidage* représente l'ensemble de moyens mis à disposition pour informer, orienter ou encore conduire l'utilisateur lors de ses interactions avec l'ordinateur. En raison du public ciblé qui sont des personnes en situation de handicap plus particulièrement des enfants. Malgré que les personnes utilisant cette application soient généralement des proches aux personnes en situation de handicap et se chargent de lancer le jeu. Il faut que l'application soit facile d'utilisation et que les utilisateurs ne soient pas perdus en l'utilisant.

Parmi les moyens de guidage mis en place pour guider l'utilisateur, figure le sous-critère de groupement/distinction*. Ceci consiste à regrouper visuellement des informations de même type. Nous pouvons voir cela à divers endroits comme par exemple le catalogue de jeux situé en plein milieu de la fenêtre, voir Figure 4 :

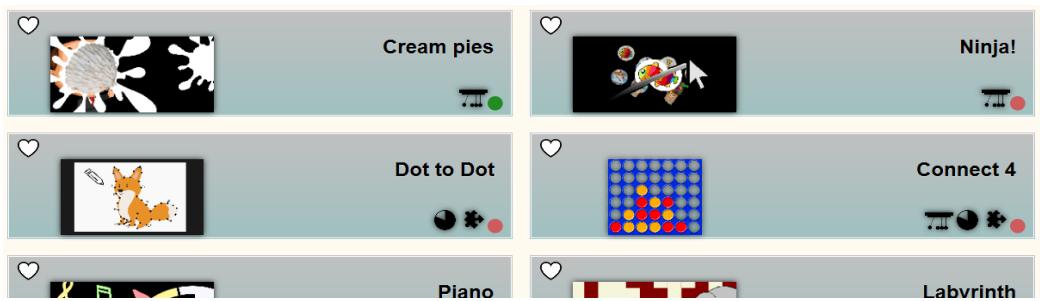


Figure 4 : Image du catalogue de jeu

C'est également présent sur les boutons permettant d'afficher les jeux disposant uniquement d'une certaine catégorie, voir Figure 5 :

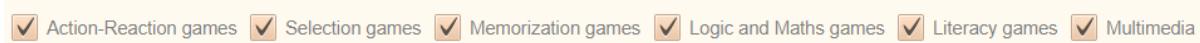


Figure 5 : Image des boutons de catégorie

Cependant, il y a un point à améliorer concernant l'interface graphique. Par exemple, d'un point de vue du contrôle explicite, qui consiste à proposer à l'utilisateur de valider certaines actions importantes ou difficilement réversibles. Sur la Figure 6 ci-dessous, il y a trois boutons. De gauche à droite, un bouton Replay pour reproduire les actions faites dans un jeu. Déconnexion pour pouvoir se déconnecter et Quitter pour fermer l'application. Lorsqu'on appuie sur l'un de ces boutons, par exemple sur le bouton pour quitter, cela quitte l'application immédiatement. De plus, grâce à un système d'info-bulles, on peut aussi savoir à quoi servent les boutons en passant la souris dessus. Pour respecter le contrôle explicite, il faudrait avoir un popup permettant de confirmer si on veut bel et bien quitter l'application ou pas.



Figure 6 : Image des trois boutons

IV. Réalisation

IV.1 Gestion de projet

Pour communiquer avec Jordan qui est mon responsable technique et qui se charge de me donner des tâches, nous utilisons [Discord](#)*. Il s'agit d'une application de messagerie et d'appel gratuit. En raison du fait qu'il travaille majoritairement en distanciel, nous sommes amenés essentiellement à communiquer presque entièrement via [Discord](#). En ce qui concerne le partage de fichiers comme par exemple récupérer le projet existant, nous utilisons [Github](#)*. [Github](#) [5] est un service permettant de partager son code informatique ce qui permet de travailler de façon collaborative sur un projet.

Au tout début de mon stage, nous avons préparé un planning à tenir qui répertorie diverses tâches à réaliser à travers les semaines. Généralement, Jordan me donne une tâche à réaliser, une fois que je l'ai finie, je fais une [Pull Request \(PR\)](#)* sur [Github](#)*. Une [PR](#) [7] ou demandes de tirage en français est une demande que l'on fait sur Github pour valider les modifications faites et l'ajouter au projet de base. Lorsque je fais une [PR](#)* Jordan me donne ensuite une nouvelle tâche à réaliser. Si jamais la [PR](#)* est refusée à cause de bug ou parce qu'on a eu de nouvelles idées à ajouter, je corrige le problème jusqu'à ce qu'elle soit acceptée ou je rajoute la nouvelle idée. Afin de détecter les bugs, j'utilise [Spotbug](#)* qui est un programme qui analyse le code Java en vérifiant s'il est contenu dans une liste prédéfinie de patterns ou de règles de codage. Pour avoir un suivi de ce planning, Jordan organise généralement un jour dans la semaine pour faire le point de l'avancement avec tous les autres stagiaires présents dans ma salle.

Concernant mes heures de travail, je dispose de 7h de travail quotidiennes à réaliser et je peux les organiser comme je le souhaite. J'ai toujours été relativement en avance sur le planning prévisionnel. J'ai rapidement fini la création de mon premier jeu et j'en ai profité pour en faire directement un deuxième. Un planning effectif de mon stage est disponible en [annexe \(5\)](#).

IV.2 Phases d'implémentation

Création de nouveaux jeux sérieux

Pour le premier jeu que j'ai créé, je me suis dit qu'il serait intéressant de proposer un jeu coopératif en simultané. Généralement, les enfants en situation de handicap qui utilisent cette application ont un proche qui les aide à utiliser le logiciel. Qui plus est, [GazePlay](#)* ne possède pas encore ce type de jeu, et c'est dans cette optique que j'ai eu l'idée de jeu suivante :

Le jeu consiste à contrôler un chat avec un clavier qui doit aller à sa gamelle. Celui-ci se fait poursuivre par des chiens qui peuvent être immobilisés en les fixant via le regard grâce à l'[oculomètre](#)*. Au fil des différents niveaux, la difficulté augmente avec l'apparition de nouvelles mécaniques, comme par exemple des murs ou encore des interrupteurs à activer pour ouvrir un passage. Ces interrupteurs s'activent à l'aide de l'[oculomètre](#)*. Pour que le but

du jeu soit le plus compréhensible possible pour tous, le nom du jeu est “Sprint To The Finish” (ou “Sprint jusqu’à l’arrivée” dans sa version française).

Le premier jour où j’ai travaillé sur ce jeu, j’ai eu du mal à l’afficher en tant que jeu sélectionnable et jouable depuis le menu déroulant contenant la liste des jeux. Après plusieurs manipulations et un peu d’aide, j’ai réussi et mieux compris comment faire pour mes futurs jeux à implémenter.

Suite à cela, j’ai enfin pu commencer la création du jeu. Mon but était de créer chacune des différentes classes* et fonctionnalités nécessaires pour qu’après, je n’ai plus qu’à me concentrer sur la création des niveaux. C’est à dire de décrire la structure de mes objets*, comment il est constitué et comment il fonctionne. Ces objets* sont issus de l’instanciation des classes* et possèdent des attributs. Elles contiennent également des méthodes* pour pouvoir définir le comportement de mes objets*

N’ayant pas d’expérience avec JavaFX* pour la création de jeu, je me suis énormément renseigné en regardant le code des autres jeux déjà présent dans GazePlay, en me documentant sur internet ou encore en suivant des tutoriels.

Pour les chiens, j’ai simplement fait en sorte qu’ils se déplacent continuellement en direction du chat. Si un obstacle se situe entre les deux, le chien va se coller à cet obstacle tout en essayant de suivre le chat.

Concernant les interrupteurs, le fonctionnement est relativement simple. Il suffit de le fixer avec le regard afin de déclencher un événement. Celui-ci lorsqu’il est déclenché permet d’enlever ou d’ajouter des murs.

Puis dans une optique de dynamiser l’expérience de jeu, j’ai aussi ajouté des murs qui se déplacent d’un point à un autre. Si jamais le chat touche le mur, le niveau recommence et on peut également immobiliser ces murs si on les regarde.

Après avoir effectué des tests fonctionnels pour être sûr que ces fonctionnalités ne posaient pas de problème lié à des bugs et n’impactent pas des jeux déjà existant, j’ai créé 17 niveaux. J’en ai profité pour ajouter des images (libres de droits ou originales) pour éviter de jouer avec des rectangles. Voici une image du jeu qui correspond à un niveau qui contient tous les éléments cités plus haut, voir Figure 7 :

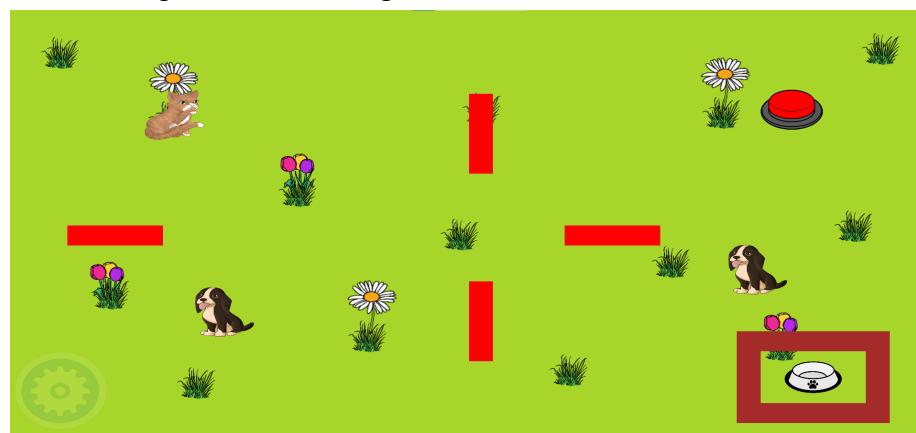


Figure 7 : Image du jeu Sprint To The Finish - level 16

Par la suite, j'ai créé une variante de ce jeu, consistant à inverser la façon de jouer. Cette fois-ci, on déplace le chat avec les yeux et on peut immobiliser les chiens/murs ou actionner les interrupteurs avec le clavier. Il faut appuyer sur A pour immobiliser le chien le plus proche, Z pour activer l'interrupteur et E pour immobiliser le mur. Dans le code que j'ai écrit, le chat, les chiens, les interrupteurs et les murs sont des objets*. Tous les objets* du niveau sont dans une liste propre à leur type d'objet*. Ils sont également contenus dans une liste que j'ai appelé "obstacle" dont je me sers pour diverses choses comme par exemple gérer les différents cas en cas de collisions entre les différents objets. Je me sers également des listes spécifiques à ces objets pour récupérer quel objet est le plus proche grâce à un parcours de ces listes et un petit calcul mathématique.

Pendant toute la durée de la création de ce jeu, j'ai été amené à plusieurs reprises à réadapter la difficulté de mon jeu. En effet, ces jeux doivent être adaptés pour des personnes en situation de handicap. Il m'est souvent arrivé de rendre le jeu trop compliqué, par exemple, les chiens se déplaçaient plus vite ce qui rendait le jeu plus difficile parce que les personnes en situation de handicap possèdent un temps de réaction inférieur à la normale. Après avoir fait des ajustements, ajouté des commentaires et de la documentation, j'ai fait une PR.

Le deuxième jeu que j'ai créé consiste à survivre contre des robots qui se déplacent aléatoirement. Le personnage jouable tire automatiquement des projectiles dans la direction où il se déplace pour pouvoir détruire les robots. Parmi les robots qui apparaissent, certains ont la capacité de tirer. Toujours dans l'optique que le nom du jeu soit clair et compréhensible, nous avons nommé celui-ci "Survive Against Robots" (ou "Survivre contre les Robots" dans sa version française). Il y aura deux façons de jouer à ce jeu, l'une avec les flèches directionnelles du clavier et l'autre avec le regard.

Ayant déjà eu de l'expérience grâce au premier jeu, ce jeu a pris beaucoup moins de temps que le jeu coopératif à développer. J'ai beaucoup utilisé du code que j'avais déjà fait. Par exemple, le système que j'utilise pour déplacer le joueur est le même dans ce jeu que celui du jeu "Sprint To The Finish". Je me suis donc concentré en priorité sur les nouvelles fonctionnalités à implémenter, en commençant par les tirs automatiques du joueur. Après avoir ajouté les tirs, j'ai ajouté les robots. Ils apparaissent aléatoirement dans le jeu et selon le niveau de difficulté du jeu, il peut y avoir des robots qui tirent automatiquement sur le joueur. Pour aider les joueurs à différencier lesquels tirent et lesquels ne tirent pas, les robots qui ne tirent pas ont des yeux oranges et ceux qui tirent ont des yeux rouges, comme le montre les Figures 8 et 9 :

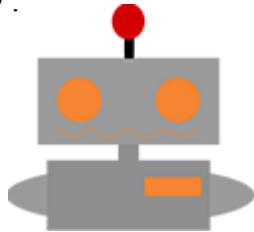


Figure 8 : Image du robot qui ne tire pas

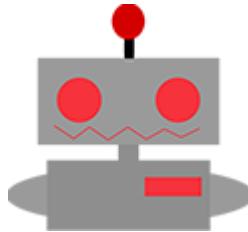


Figure 9 : Image du robot qui tire

J'ai également ajouté un texte sur l'écran qui permet de savoir le score du joueur. Ce score augmente de deux façons. Soit en survivant chaque seconde, le score s'incrémentera de 1.

Soit en détruisant des robots, le score s'incrémente de 5. Après avoir effectué tous ces ajouts et avoir essayé le jeu, je me suis rendu compte qu'il pouvait être compliqué pour les joueurs de se concentrer à la fois sur le déplacement, et à la fois sur le fait de viser les robots. De ce fait, après discussion, on a convenu que le plus simple était de proposer une variante du jeu avec des tirs automatiques qui visent automatiquement le robot le plus proche du joueur. Cette variante a été développée rapidement. En effet, pour connaître le robot le plus proche du joueur, j'utilise le même code que j'ai écrit dans le jeu "Sprint To The Finish". De plus, pour tirer automatiquement dans la direction de la souris/regard, j'appelle une méthode* qui prend en paramètre une cible qui correspond ici à la souris/regard. J'ai utilisé cette même méthode en spécifiant simplement que la cible n'est plus la souris mais le robot.

En testant le jeu, je réfléchissais à un moyen de le rendre plus dynamique. Dans cette optique, j'ai eu l'idée d'ajouter un système de bonus qui aide le joueur. Il s'agirait de faire en sorte que lorsqu'un robot est détruit, il a un pourcentage de chance de laisser derrière lui un bonus. Il existe trois bonus à l'heure actuelle, un bonus qui ralentit tous les robots. Le deuxième permet d'augmenter la fréquence de tir du joueur et le troisième, de rendre temporairement invincible le joueur. Ces bonus varient entre 6 et 10 secondes et voici à quoi ils ressemblent, voir Figures 10, 11 et 12 :

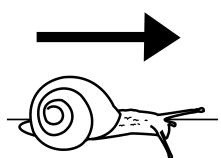


Figure 10 : Bonus ralentissement



Figure 11 : Bonus fréquence de tir



Figure 12 : Bonus d'invincibilité

J'en ai profité pour ajouter en haut à droite les bonus et leur durée pour que le joueur puisse s'adapter en fonction du temps restant. Durant la création du jeu, j'ai souvent fait tester le jeu à plusieurs personnes afin d'avoir des retours et ensuite améliorer les points négatifs. Par exemple, il arrivait des fois que les robots pouvaient apparaître très proche du joueur ce qui avait pour effet de nous faire perdre la partie, puisqu'il est compliqué de réagir rapidement. J'ai donc corrigé cela en ajoutant une zone autour du joueur dans laquelle les robots ne peuvent pas apparaître. Voici une image du jeu en pleine partie, voir Figure 13 :

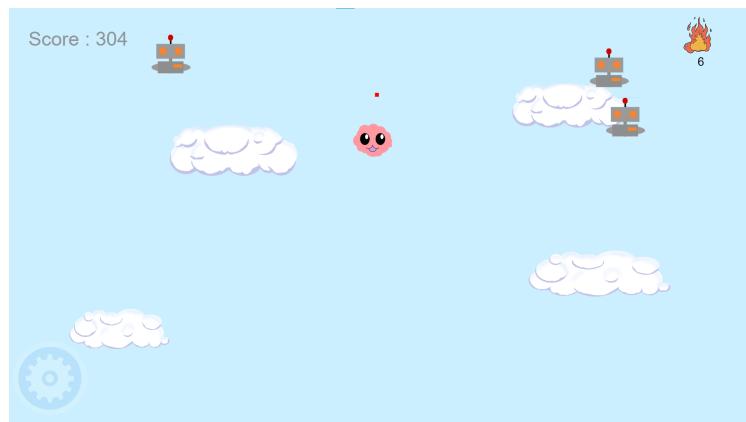


Figure 13 : Image du jeu Survive Against Robots

Après avoir fini les ajustements nécessaires, ainsi que l'ajout ou la mise à jour de la documentation et des commentaires, j'ai fait une [PR*](#).

Le troisième jeu consiste à recréer le jeu de société “Simon”. Ce jeu de société se présente sous la forme d'un plateau circulaire ayant 4 boutons de couleurs différentes. D'après wikipédia [9], dans son mode de base, Simon va éclairer une des quatre couleurs et produire un son associé à cette couleur. Il possède trois modes de jeu :

- Le mode 1 consiste à reproduire une suite de touches créée aléatoirement par le jeu, la plus longue possible. L'édition originale de Simon considère la partie comme gagnée au bout de 32 touches retenues.
- Le mode 2 inverse les rôles : le joueur va composer la suite de touches, et Simon va la reproduire. Cependant, le joueur ne doit pas se tromper en produisant sa propre suite.
- Le mode 3 est un mode multijoueur : le jeu attribue une touche à chaque joueur, et devra la presser quand Simon la lui demandera. Si un joueur ne répond pas assez rapidement quand Simon lui demande, ou répond au mauvais moment, il sera éliminé de la manche.

Pour la création du jeu, la première étape fut la création de la borne Simon. Voici à quoi ressemble la borne que j'ai créé, voir Figure 14 :

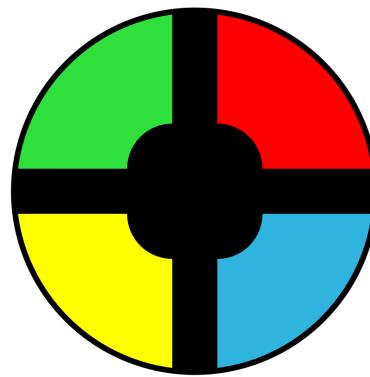


Figure 14 : Borne du jeu Simon

Après avoir produit cette borne, je me suis concentré sur la création du premier et deuxième mode du jeu. Je n'ai pas eu de soucis particulier à implémenter ceux-là, à part pour le format des fichiers audio. En regardant les autres jeux musicaux de l'application, je me suis rendu compte que certains jeux utilisent le format [Musical Instrument Digital Interface \(MIDI\)*\[8\]](#) qui est un type de fichier pour produire du son. En regardant le code de ces autres jeux, je voulais m'en inspirer pour pouvoir utiliser aussi des fichiers au format [MIDI*](#) pour le son des boutons. Néanmoins, je n'arrivais pas à comprendre le code parce qu'il utilisait beaucoup de nouvelles classes comme un Synthétiseur. Après discussion, on en a conclu que c'était beaucoup plus simple d'utiliser des fichiers au format WAV. Une fois que les sons ont été implémentés, il ne me reste plus qu'à implémenter le troisième mode multijoueur (actuellement en cours).

Création de nouvelles fonctionnalités

La plupart des nouvelles fonctionnalités que j'ai implémenté proviennent d'une liste de suggestion disponible sur le [Github*](#) de [GazePlay*](#).

Une des nouvelles fonctionnalités concerne l'ajout d'un menu de sélection de niveau sur l'un des jeux existant qui est le jeu Rush Hour. Ce jeu proposait une trentaine de niveaux mais lorsque l'on cliquait sur le jeu, on commençait tout le temps au niveau 1 sans aucune possibilité de choisir le niveau auquel on souhaite commencer. Cela n'était pas pratique dans le cas où on souhaitait reprendre une progression par exemple. La partie du code sur lequel j'ai travaillé était en grande partie liée à celle qui gérait la création de variantes de jeux. De ce fait, j'ai pu apprendre à créer des variantes de jeux.

Voici le menu de sélection du jeu Rush Hour, voir Figure 15 :

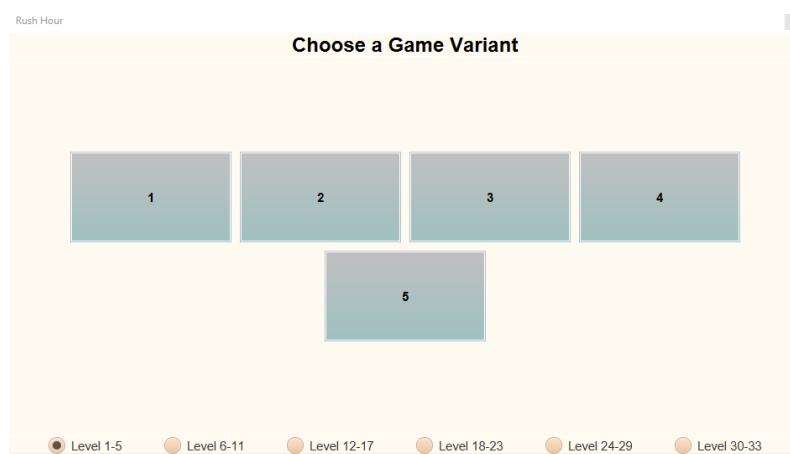


Figure 15 : Sélection des niveaux du jeu Rush Hour

J'ai également amélioré certains jeux comme par exemple le jeu Bubble. Le principe du jeu consiste à faire éclater des bulles qui apparaissent à l'écran avec le regard. Le jeu possède également une image de fond. L'issue sur [Github*](#) était de cacher l'image de fond et de la révéler au fur et à mesure que les bulles s'éclatent. Pour créer cette fonctionnalité, je me suis inspiré d'un autre jeu du nom de Blocs. Il consiste à dévoiler une image en regardant des blocs. J'ai récupéré le système de génération des blocs qui cachent l'image dans ce jeu et je l'ai intégré au jeu Bubble. Voici le résultat en Figure 16 :

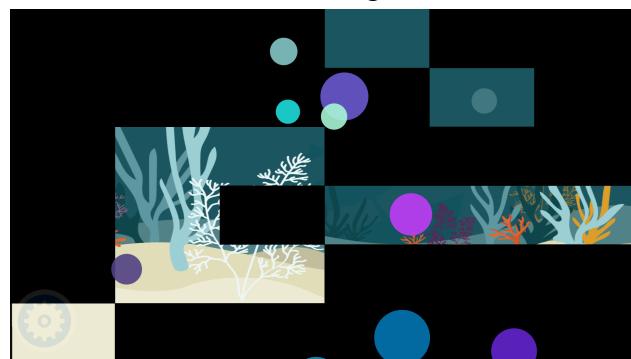


Figure 16 : Image du jeu Bubble

Comme autre amélioration faite sur des jeux existants, une issue de [Github*](#) concernait le jeu Egg. Le but de ce jeu était de regarder un œuf qui éclosait petit à petit. L'idée de l'amélioration est juste de faire en sorte que l'image de l'œuf se rétrécit petit à petit et qu'au moment de l'éclosion, elle reprenne sa taille initiale. J'ai pas eu de difficulté particulière à réaliser cette tâche.

Correction de bugs

Sur le [Github*](#), il y a également une liste qui répertorie les différents bugs présents sur les jeux. Parmi les tâches qu'on m'a attribué, il m'arrivait de devoir corriger des bugs sur des jeux existants. Ce fut le cas avec le jeu Rush Hour. Dans ce jeu, on doit bouger une voiture jaune et la faire sortir du parking. Sur ce même parking, il y a d'autres voitures qui empêchent la voiture jaune d'avancer. Il faut donc également les déplacer d'une certaine façon pour pouvoir libérer le passage. Dans [GazePlay*](#), tout ce qui est relatif à la sélection ou actionner un objet quelconque, on utilise ce qu'on appelle un [ProgressIndicator \(PI\)*](#). Un [PI*](#) est un timer qui se déclenche et permet d'effectuer une action à la fin de celui-ci. Il est très souvent utilisé lorsqu'on regarde un objet et il est représenté sous forme d'un cercle de chargement qui se remplit petit à petit, jusqu'à arriver à 100%. Lorsqu'on regarde une voiture, le [PI*](#) se déclenche donc et nous permet de sélectionner la voiture à déplacer. Normalement, lorsqu'on arrête de regarder une voiture, cela enlève la sélection de cette voiture. Le bug était qu'il fallait alterner entre regarder la voiture et arrêter de la regarder plusieurs fois pour qu'elle ne soit plus sélectionnée. J'ai vite repéré que le timer du [PI*](#) pouvait se déclencher lorsqu'on regardait la voiture et il pouvait se re-déclencher aussi lorsqu'on le regardait à nouveau alors qu'il était déjà sélectionné. Cela avait pour effet d'annuler le fait d'enlever la sélection de la voiture. J'ai donc simplement rajouté une variable booléenne pour contrôler ceci.

Difficultés

Durant mon stage, j'ai été amené à avoir des difficultés que je devais résoudre afin de remplir mes tâches.

L'un des problèmes que j'ai eu qui touchait de façon relative tous les jeux que j'ai créé était d'adapter la difficulté. En effet, surtout au début, je n'arrivais pas suffisamment à me mettre à la place des personnes en situation de handicap. De ce fait, il n'était pas rare que les jeux soient trop compliqués. Mais au fur et à mesure, j'ai mieux compris comment faire et comment rendre les jeux plus facile.

Pour le jeu coopératif, très souvent dans les niveaux que je propose, je place au moins un interrupteur qui emprisonne la gamelle du chat afin d'ajouter de la difficulté, j'ai créé une [méthode*](#) qui permet d'automatiser cela m'évitant ainsi de placer à la main les coordonnées des murs. J'ai eu quelques difficultés à faire en sorte que cela fonctionne en fonction des coordonnées et de la taille. Les murs faisaient bel est bien un carré autour de la gamelle mais légèrement décalé ce qui avait pour effet d'avoir un mur rentrant dans une partie de la gamelle. J'ai réussi au bout de quelques heures de travail ce qui m'a fait gagner énormément de temps pour créer des niveaux.

La principale difficulté que j'ai eu pour ce jeu était d'implémenter les événements liés aux yeux. Avant d'implémenter ces évènements, dans la structure de mon code, j'avais une [classe*](#) se nommant Cat qui était une [classe abstraite*](#) ayant deux autres classes héritant de celle-ci. La particularité d'une [classe abstraite*](#) c'est qu'elle n'est pas instanciable. De ce fait, cela posait des problèmes et empêchait l'utilisation de ses événements. Après avoir revu rapidement la structure de mon code, et avec un peu d'aide, j'ai corrigé le problème.

Concernant le jeu de survie, j'ai eu quelques problèmes liés au tir automatique. Tout d'abord, un tir consiste à faire partir un petit rectangle du joueur dans une direction donnée, mais je n'arrivais pas à faire en sorte qu'il garde l'angle correctement tout au long de la distance qu'il doit parcourir. En faisant des recherches, j'ai réglé assez rapidement le problème.

Pour le jeu Simon, j'ai eu un peu de problème à créer la borne de 4 couleurs parce que je n'arrivais pas à donner la forme escomptée avec des cercles. Le problème était que je n'arrivais pas à cacher d'une façon propre les cercles pour faire des demi-arc de cercles qui ressemblent au jeu de base. C'était jusqu'à ce que je me rende compte qu'avec [JavaFX*](#), il y a une classe "Arc" qui permet littéralement de créer des arcs de cercle. J'ai pu ensuite très rapidement créer la borne en utilisant cette classe.

J'ai également eu des problèmes concernant le menu de sélection des niveaux pour le jeu Rush Hour. En effet, je n'avais pas très bien compris la structure du code et dès le début, j'étais parti sur une mauvaise piste. Cette fonctionnalité est celle qui m'a pris le plus de temps car en arrivant sur le projet, la première chose que j'ai faite était de lire le plus de code mais j'avais du mal à le comprendre en réalité.

Après avoir fait une visio pour voir et expliquer l'ensemble du code, j'ai beaucoup mieux compris comment il fonctionnait et cela m'a été d'une grande aide de façon générale pour mon stage.

Conclusion

Pour conclure, ce stage m'a permis d'approfondir mes connaissances en Java/[JavaFX](#)* et en création de jeux vidéo. Par ailleurs, c'est toujours assez compliqué d'arriver sur un gros projet déjà bien avancé. C'est pour cela qu'il est important d'avoir de l'expérience dans la lecture du code de quelqu'un d'autre et savoir comment l'utiliser. Ce stage a été également très enrichissant sur ce point. J'ai été amené à faire beaucoup de recherches internet durant mon stage pour savoir comment faire certaines fonctionnalités. De même, lorsque je ne comprenais pas certaines choses liées à du code spécifique au projet, j'étais amené à poser énormément de questions à Jordan.

De plus, c'est également très enrichissant d'un point de vue professionnel. Côtoyer d'autres stagiaires instaure une certaine sphère de travail très différente du monde universitaire. D'autant plus qu'il arrive que je participe à des séminaires qui ont lieu dans le bâtiment où je travaille et que l'ambiance y est radicalement différente de ce qu'on a l'habitude en tant qu'étudiant.

Par ailleurs, durant mon stage, j'ai très souvent été amené à réfléchir à comment améliorer les jeux que je créais. Comme par exemple, l'ajout de bonus temporaire pour le jeu Survive Against Robots. Il y a certaines fonctionnalités qui seraient intéressantes d'ajouter comme par exemple un éditeur de niveau pour le jeu coopératif ou bien un mode multijoueur pour le jeu de survie contre les robots.

Au-delà de nouvelles fonctionnalités, il y a certaines méthodes de travail que j'aurais pu adopter. À chaque fois que je faisais des recherches sur internet, c'était très souvent lié à un problème. Mais il y a certains points où j'aurais dû faire mes recherches avant de me mettre à coder. Par exemple, dans le jeu coopératif et le jeu de survie, j'utilise une [méthode](#)* pour pouvoir vérifier les collisions par un calcul mathématique des coordonnées. Cependant, je me suis rendu compte que dans la bibliothèque [JavaFX](#)*, il y a déjà des méthodes qui permettent cela. Bien que je n'ai pas perdu énormément de temps à créer cette méthode, je pense qu'il aurait été plus simple de le faire de cette façon.

Ce stage m'a énormément appris sur comment gérer diverses situations, qu'elles soient techniques ou humaines et m'a confirmé mon souhait de continuer ma formation en BUT 3 en alternance.

Glossaire

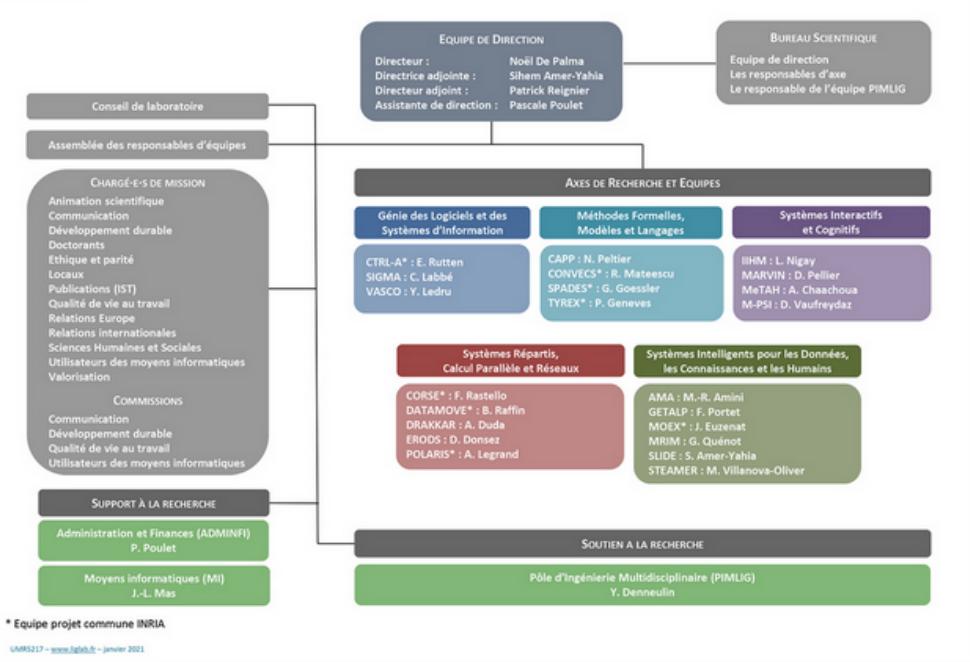
AugCom	Site internet permettant la création de grilles de communication alliant images et mots. Ces grilles de communication aident les personnes en situation de handicap à communiquer en sélectionnant les mots souhaités et en les prononçant oralement.
Azul	Distribution gratuite ayant des mises à jour fréquentes et dont le support commercial est autorisé en plus d'avoir une licence permissive.
Communication Alternative Augmentée (CAA)	Ensemble des techniques et outils permettant aux personnes ayant des difficultés pour communiquer de s'exprimer ou de comprendre les autres.
Classe	Une classe dans un langage de programmation décrit simplement la structure d'un objet, comment il est constitué et comment il fonctionne.
Classe abstraite	Une classe abstraite est une classe qui n'est pas instanciable.
Critères ergonomiques de Bastien et Scapin [3]	Caractéristiques de l'interface qui vont déterminer son utilisabilité comprenant 8 dimensions clés : guidage, charge de travail, contrôle explicite, adaptabilité, gestion des erreurs, homogénéité, signification des codes et compatibilité.
Discord	Application gratuite permettant d'effectuer des appels ou d'envoyer des messages.
Framework	Fournit une bibliothèque de fonctionnalités spécifiques.
GazePlay	Logiciel libre et gratuit qui rassemble plusieurs mini-jeux jouables grâce à un oculomètre (Eye-tracker). La dernière version compte près de 60 jeux.
Github [5]	Service d'hébergement Open-Source, permettant aux programmeurs et aux développeurs de partager le code informatique de leurs projets afin de travailler dessus de façon collaborative.
Groupement/distinction	Sous critère ergonomique du guidage qui consiste à regrouper des informations de même type.
Guidage	Ensemble de moyens mis à disposition pour informer, orienter ou encore conduire l'utilisateur lors de ses interactions avec l'ordinateur.

Hotspots	Zones que l'on peut placer et qui produisent des sons choisis par l'utilisateur à leur clique.
IDE [6]	Un environnement de développement intégré (IDE) est une application logicielle qui aide les programmeurs à développer efficacement le code logiciel.
Interface Homme Machine (IHM)	Outils et moyens mis en œuvre pour que l'Homme puisse communiquer ou contrôler avec une ou plusieurs machines.
InterAActionBar	Projet mené par l'équipe GETALP qui a pour but de favoriser l'apprentissage et la communication des personnes en situation de handicap cognitif.
InterAActionScene	Site internet permettant à l'utilisateur de créer des scènes, de leurs attribuer des images et d'y placer des dessins ou des hotspots.
JavaFX	Framework pour Java qui permet d'intégrer des graphiques vectoriels, des animations, du son, des vidéos assez facilement.
Méthode	Définition d'une partie du fonctionnement d'une classe.
Musical Instrument Digital Interface (MIDI) [8]	Le Musical Instrument Digital Interface ou MIDI est un type de fichier utilisé pour le son.
Objet	Un objet est issu de l'instanciation de la classe et possède des attributs.
Oculomètre	Appareil permettant de mesurer les mouvements des yeux.
ProgressIndicator (PI)	Timer qui se déclenche et qui permet d'effectuer une action à la fin de celui-ci. Il est très souvent utilisé lorsqu'on regarde un objet et il est représenté sous forme d'un cercle de chargement qui se remplit à 100% lorsque c'est terminé.
Pull Request (PR) [7]	Les Pull Request vous permettent d'informer d'autres personnes des modifications que vous avez apportées à une branche dans un dépôt sur GitHub. Une fois qu'une demande d'extraction est ouverte, vous pouvez discuter et revoir les changements potentiels avec vos collaborateurs et ajouter des commentaires de suivi avant que vos changements ne soient fusionnés dans la branche de base.
Spotbug	Programme qui analyse le code Java en vérifiant s'il est contenu dans une liste prédéfinie de patterns ou de règles de codage
Spring [4]	Framework d'application pour la plateforme Java. Il est utilisé pour construire des applications web modernes avec un support pour les architectures microservices, les systèmes cloud, le traitement réactif et les charges de travail sans serveur.

Timeline AnimationTimer	ou	Classes permettant l'exécution de code à une certaine fréquence pendant X temps.
----------------------------	----	--

Annexes

Annexe 1 : Organigramme du LIG



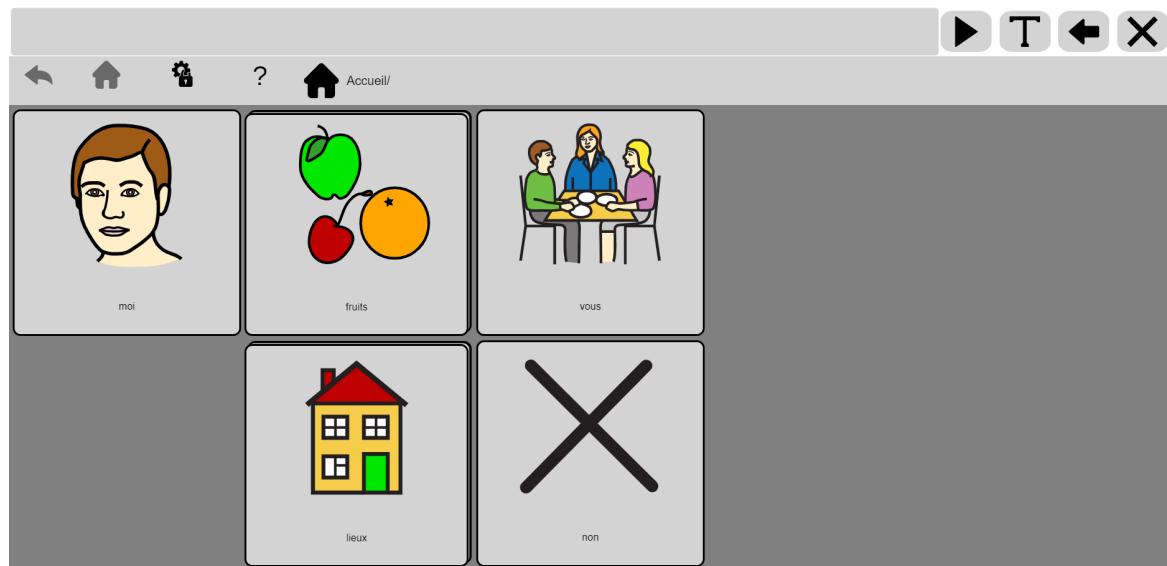
Annexe 2 : Oculomètre (Eye-Tracker) Tobii



Annexe 3 : Planning prévisionnel du stage

Période	Travail à réaliser
17/04-21/04	Initiation au projet GazePlay (lecture du code et implémentation de nouvelles fonctionnalité)
24/04-05/05	Création d'un nouveau jeu coopératif de 0
08/05-19/05	Amélioration du jeu sérieux coopératif (ajout d'options + check si error test)
22/05-26/05	Ajout d'un nouveau mode de jeu pour un jeu sérieux existant
29/05-03/06	Préparation du rapport préliminaire
05/06-09/06	Préparation du mémoire de stage
12/06-16/06	Amélioration d'un jeu existant
19/06-23/06	Soutenance de Stage + création d'un guide pour un joueur tout venant
26/06-07/07	Ajout de nouveau guide + modifications de certains jeux + correction de bug
17/04-07/07	Révision de Pull Request Github jusqu'à validation (Merge)

Annexe 4 : Exemple de grille de communication AugCom



Annexe 5 : Planning effectif

Période	Travail à réaliser
17/04-21/04	Initiation au projet GazePlay (lecture du code et implémentation de nouvelles fonctionnalité)
24/04-05/05	Création d'un nouveau jeu coopératif (Sprint to the Finish) de 0 + variantes du jeu + amélioration du jeu
08/05-19/05	Création d'un nouveau jeu de survie (Survive Against Robots) de 0 + variantes du jeu
22/05-26/05	Réalisation de tâches sur github (issues) + Création d'un nouveau jeu musical (Simon) de 0
29/05-03/06	Préparation du rapport préliminaire
05/06-09/06	Préparation du mémoire de stage
12/06-16/06	Amélioration d'un jeu existant
19/06-23/06	Soutenance de Stage + création d'un guide pour un joueur tout venant
26/06-07/07	Ajout de nouveau guide + modifications de certains jeux + correction de bug
17/04-07/07	Révision de Pull Request Github jusqu'à validation (Merge)

Sitographie

[1] Présentation du LIG. Disponible sur <https://www.liglab.fr/fr/presentation>

(consulté le 28/04/2023) et sur son livret de 2021:

https://www.liglab.fr/sites/lig/files/Mediatheque/livret_small-2021_compressed.pdf

(consulté le 28/04/2023)

[2] Étude de l'UNICEF sur le nombre d'enfants handicapés en 2021 dans le monde :

<https://www.unicef.org/fr/communiqu%C3%A9s-de-presse/le-monde-compte-pres-de-240-millions-d-enfants-handicap%C3%A9s#:~:text=NEW%20YORK%2C%20le%2010%20novembre,un%20lire%20dans%20ce%20rapport>

(consulté le 26/05/2023)

[3] Critères ergonomiques de Bastien et Scapin :

<https://www.usabilis.com/criteres-ergonomiques-bastien-et-schapin/>

[4] Définition de Spring :

<https://www.jetbrains.com/help/idea/spring-support.html>

[5] Définition de Github :

<https://datacientest.com/github-tout-savoir#:~:text=GitHub%20est%20un%20service%20d,travailler%20dessus%20de%20fa%C3%A7on%20collaborative.>

[6] Définition d'IDE :

<https://aws.amazon.com/fr/what-is/ide/#:~:text=qu'AWS%20Cloud9%20%3F-,Qu'est%2Dce%20qu'un%20IDE%20%3F,d%C3%A9velopper%20efficacement%20le%20code%20logiciel>

-

[7] Définition de Pull Request :

<https://docs.github.com/fr/pull-requests/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/about-pull-requests>

[8] Définition de MIDI :

https://fr.wikipedia.org/wiki/Musical_Instrument_Digital_Interface#:~:text=Le%20Musical%20Instrument%20Digital%20Interface,s%C3%A9quenceurs%2C%20et%20logiciels%20de%20musique.

[9] Règle du jeu Simon :

[https://fr.wikipedia.org/wiki/Simon_\(jeu\)](https://fr.wikipedia.org/wiki/Simon_(jeu))

Résumé court

Pendant mon stage au sein de l'équipe GETALP, j'ai travaillé sur le projet InterAACTIONBox, qui vise à améliorer la communication et le traitement de l'information pour les personnes en situation de handicap cognitif. Mon rôle était de développer, améliorer et implémenter des jeux sérieux et des fonctionnalités pour un logiciel appelé GazePlay. GazePlay est un logiciel de Communication Alternative et Augmentée (CAA) qui propose près de 60 jeux sérieux jouables avec un oculomètre. Les objectifs de GazePlay sont de faciliter l'interaction oculaire, d'offrir des jeux adaptés aux besoins des personnes en situation de handicap cognitif et d'intégrer des fonctionnalités de CAA pour faciliter l'expression et la compréhension. J'ai créé trois jeux sérieux pendant mon stage : 1) "Sprint to the finish", un jeu coopératif où un chat doit atteindre sa gamelle tout en évitant des chiens, avec l'aide d'un deuxième joueur ; 2) "Survive against robots", un jeu de survie où le joueur doit esquiver des robots qui tirent aléatoirement ; et 3) un jeu musical inspiré du jeu de société Simon, où le joueur doit reproduire les touches jouées par la machine.

Mots-clefs : Communication Alternative et Augmentée (CAA), développement, GazePlay, GETALP, handicap cognitif, InterAACTIONBox, jeux sérieux, oculomètre

Abstract

GazePlay: Enhancing Communication and Learning for Cognitive Disabled Individuals

Noha Boutemeur.

abstract: GETALP researches multilingual communication and information processing. Their InterAACTIONBox project enhances learning and communication for people with cognitive disabilities. I developed serious games for GazePlay, an AAC software that allows individuals to play nearly 60 games using an eye tracker. The software enables users to control the mini-games using their gaze, enhancing eye interaction capabilities. Three serious games were successfully created within GazePlay. 1) "Sprint to the Finish" is a cooperative game where a cat must reach its food bowl while evading dogs, with assistance from another player. 2) "Survive against Robots" is a survival game where the player must avoid shooting robots, aiming for the highest score. And 3), a musical game inspired by Simon challenges the player to replicate note sequences. The assigned tasks were completed ahead of schedule. GazePlay, programmed in Java/JavaFX, features a user-friendly design with vibrant visuals and grouped elements for improved usability.

Keywords: GazePlay, GETALP, communication, learning, cognitive disabilities, serious games, Augmentative and Alternative Communication (AAC), eye tracker, InterAACTIONBox.
