



ÉCOLE NATIONALE SUPÉRIEURE D'ÉLECTRONIQUE,  
INFORMATIQUE, TÉLÉCOMMUNICATIONS, MATHÉMATIQUES ET  
MÉCANIQUE DE BORDEAUX

DEUXIÈME ANNÉE INFORMATIQUE  
INITIATION À LA RECHERCHE EN LABORATOIRE  
RAPPORT DE STAGE

---

# Graphogame

---

*Auteur :*

TOMAS Léo

*Autres stagiaires :*

EL ALLAM OUSSAMA

HUGUES ADÈLE

*Maître de stage :*

ANDRE JEAN-MARC

# Table des matières

<b>1 Résumé</b>	<b>2</b>
<b>2 Introduction</b>	<b>3</b>
2.1 Présentation du laboratoire . . . . .	3
2.2 Contexte du stage . . . . .	3
<b>3 Réalisation personnelle</b>	<b>7</b>
3.1 Choix des technologies . . . . .	7
3.2 Présentation de l'application . . . . .	8
3.2.1 Organisation générale . . . . .	8
3.2.2 Contraintes traitées imposées par les besoins client et technologies . . . . .	9
3.3 La base de données . . . . .	10
3.4 Interface thérapeute . . . . .	11
3.5 Interface patient . . . . .	11
3.5.1 Exercices de pression . . . . .	12
3.5.2 Exercices de flexion . . . . .	12
<b>4 Continuité du projet et améliorations</b>	<b>14</b>
4.1 À développer en priorité . . . . .	14
4.1.1 Jeu de flexion et autres jeux . . . . .	14
4.1.2 Panel de difficultés de jeu . . . . .	15
4.1.3 Notifications . . . . .	15
4.1.4 Enregistrement des données . . . . .	15
4.2 Application tierce de représentation des données patient . . . . .	15
4.3 Versions de l'application . . . . .	17
4.3.1 Jeux 3D et multijoueurs . . . . .	17
4.3.2 Mode libre : jeu à sauvegarde et multiexercices . . . . .	17
4.3.3 Évaluer l'évolution du patient : utilisation d'une intelligence artificielle . . . . .	17
<b>5 Conclusion</b>	<b>18</b>
5.1 Bilan Personnel . . . . .	18
5.2 Bibliographie . . . . .	18
<b>6 Annexes</b>	<b>19</b>
6.1 Ajouter un jeu C++ à l'application . . . . .	19
6.2 Traiter la difficulté de jeu . . . . .	20
6.3 Données des capteurs en base de données . . . . .	20
6.4 Fonctions de notification patient . . . . .	21
6.5 Exporter votre premier fichier.csv . . . . .	21
6.6 Migrations de la base de données . . . . .	22
6.7 Liens utiles . . . . .	23

# Chapitre 1

## Résumé

La dysgraphie est définie comme une atteinte de la qualité d'écriture sans que cette déficience puisse être expliquée par un déficit neurologique ou intellectuel. Cette difficulté est indépendante des capacités à lire et n'est pas liée à un trouble psychologique.

Les principaux signes de la dysgraphie sont une mauvaise organisation de la page, la maladresse du tracé et des erreurs de formes et de proportion dans le traçage des lettres. Le mouvement est heurté, saccadé, manquant de fluidité, le trait est irrégulier, les liaisons entre les lettres sont souvent absentes ou laborieuses. Les lettres sont mal proportionnées, trop larges ou trop hautes ou au contraire, atrophiées et déformées. Les perturbations de l'écriture vont de la simple erreur de substitution de lettres jusqu'à l'incapacité totale d'écrire.

Le projet *Graphogame* consiste à développer un appareil de rééducation communicant avec un logiciel informatique, et permettant aux patients de graphothérapeute victimes de dysgraphie d'effectuer les exercices de rééducation de façon ludique.

En effet, l'initiative du projet *GraphoGame* a été lancée en 2017 par *Samia HUMEAU*, graphothérapeute dont les patients sont majoritairement des enfants et adolescents. Elle souhaitait développer un système lui permettant de prendre plusieurs patients en même temps, en n'en laissant tout d'abord qu'un puis plusieurs d'entre eux en autonomie avec le *Graphogame*, et ainsi fournir des séances plus régulières à moindres coût afin d'apporter aux enfants une thérapie de qualité.

De plus, les enfants n'effectuent que très peu les exercices recommandés à faire à la maison (manque de temps.. et surtout d'envie!). Le *Graphogame* devait donc également être facilement transportable et leur procurer l'envie d'effectuer ces exercices tout en fournissant au thérapeute un moyen de vérifier que ceux-ci ont été effectués. L'idée retenue, germée avant le début du stage, était un jeu vidéo sur tablette.

À l'aide de deux étudiants, l'un en filière électronique à l'ENSEIRB s'occupant de la conception de l'appareil de rééducation, et l'autre à l'école de cognitique ENSC dont le rôle était à la fois de valider la cohérence pratique de l'avancé du projet et d'effectuer des séries de mesures utiles à l'interprétation des données recueillies par l'appareil, mon rôle a été de développer une application *Android* proposant un panel de jeux vidéo et une programmation de ceux-ci par le thérapeute dans un emploi du temps.

# Chapitre 2

## Introduction

### 2.1 Présentation du laboratoire

Le laboratoire de l'Intégration du Matériaux au Système (IMS, CNRS UMR5218) a été créé le 1er janvier 2007, par la fusion de trois unités de recherche bordelaises, (IXL, PIOM, LAPS) avec une stratégie scientifique commune de développement principalement centrée dans le domaine des Sciences et de l'Ingénierie des Systèmes, à la convergence des Sciences et Technologies de l'Information et de la Communication (STIC), et des Sciences pour l'Ingénieur (SPI). Le laboratoire est rattaché à trois tutelles, le CNRS, l'Université de Bordeaux et Bordeaux Aquitaine INP. Au CNRS, l'UMR5218 est rattachée en principal à l'Institut des Sciences de l'Ingénierie et des Systèmes (INSIS) et en secondaire à l'Institut des Sciences de l'Information et de leurs Interactions (INS2I).

L'IMS offre un positionnement scientifique original dans le domaine des Sciences et de l'Ingénierie des Systèmes, à la convergence des Sciences et Technologies de l'Information et de la Communication (STIC), et des Sciences pour l'Ingénieur (SPI). Le laboratoire développe un ensemble cohérent de travaux de recherche allant du développement de technologies alternatives à la filière silicium pour l'élaboration de dispositifs électroniques innovants jusqu'à l'ingénierie des systèmes hétérogènes. Plus spécifiquement, des travaux de recherche amonts ou finalisés sont mis en œuvre à travers les priorités scientifiques suivantes :

- modélisation et mise en forme de matériaux pour l'élaboration de composants et microsystèmes.
- modélisation, conception, intégration et analyse de fiabilité des composants, circuits et assemblages.
- identification, commande, diagnostic, traitement du signal et des images.
- conduite des processus complexes et hétérogènes, ingénierie humaine et interactions avec le domaine du « vivant ».

Les champs d'application sont nombreux et généralement abordés à travers une approche pluridisciplinaire par projet à tous les niveaux, laboratoire, régional, national ou international. Plus d'une centaine de projets ou contrats sont gérés simultanément à l'intérieur du laboratoire, et plus spécifiquement ciblés dans les domaines du transport, des télécommunications, du vivant, de la connaissance, de l'environnement et des sources d'énergie.

### 2.2 Contexte du stage

La dysgraphie est définie comme une atteinte de la qualité d'écriture sans que cette déficience puisse être expliquée par un déficit neurologique ou intellectuel. Cette difficulté est indépendante des capacités à lire et n'est pas liée à un trouble psychologique.

Le projet *Graphogame* consiste à développer un appareil de rééducation communicant avec un logiciel informatique, et permettant aux patients de graphothérapeute victimes de dysgraphie d'effectuer les exercices de

rééducation de façon ludique. L'initiative du projet *GraphoGame* a été lancée en 2017 par *Samia HUMEAU*, graphothérapeute dont les patients sont majoritairement des enfants et adolescents.

Elle souhaitait développer un système lui permettant de prendre plusieurs patients en même temps, en n'en laissant tout d'abord qu'un puis plusieurs d'entre eux en autonomie avec le *Graphogame*, et ainsi fournir des séances plus régulières à moindres coût afin d'apporter aux enfants une thérapie de qualité.

De plus, les enfants n'effectuent que très peu les exercices recommandés à faire à la maison (manque de temps.. et surtout d'envie!). Le *Graphogame* devait donc également être facilement transportable et leur procurer l'envie d'effectuer ces exercices tout en fournissant au thérapeute un moyen de vérifier que ceux-ci ont été effectués. L'idée retenue, germée avant le début du stage, était un jeu vidéo sur tablette.

À l'aide de deux étudiants, l'un en filière électronique à l'ENSEIRB s'occupant de la conception de l'appareil de rééducation, et l'autre à l'école de cognitique ENSC dont le rôle était à la fois de valider la cohérence pratique de l'avancé du projet et d'effectuer des séries de mesures utiles à l'interprétation des données recueillies par l'appareil, mon rôle a été de développer une application *Android* proposant un panel de jeux vidéo et une programmation de ceux-ci par le thérapeute dans un emploi du temps.

Plusieurs types d'appareils ont été imaginés.

L'idée naturelle est une sorte de gant, ou même d'un exosquelette revêtant la main du patient. L'idée du gant a été préférée étant donné le caractère effrayant que peut susciter un exosquelette chez des enfants atteints d'autisme. Les exercices ciblés afin de développer la pince tri-digitale nécessaire à l'écriture (pouce, index et majeur tenant le stylo) sont de deux natures.

- le mouvement de « force » entre le pouce en opposition avec le majeur. Le travail aussi bien de la force que de la fréquence sont recherchés sur ce type d'exercice. À noter qu'après quelques temps, un nouveau besoin client s'est fait sentir : la détection de l'engagement de l'index sur cette opposition, de manière à former la pince tri-digitale spécifique à l'écriture.
- le mouvement de flexion/extension du pouce, de l'index et du majeur. Le but étant de développer leur motricité fine : la précision est donc recherchée. Il fallait également prendre en compte le besoin de mettre en oeuvre, ou de laisser la possibilité d'inclure, un système de résistance à ces mouvements.

Pour le premier type d'exercice, deux capteurs de force seraient fixés aux endroits appropriés respectivement sur le pouce et le majeur.



FIGURE 2.1 – Capteur de force

En effet, un seul capteur ne suffirait pas puisqu'il s'agit de détecter l'opposition entre le pouce et le majeur à un endroit particulier sur les deux doigts.



FIGURE 2.2 – Opposition correctement effectué



FIGURE 2.3 – Opposition mal travaillée

Ces capteurs envoient un nombre en fonction de la force appliquée sur le capteur. Cette valeur a pu être convertie en *Newton* à l'aide d'un diagramme sur le site vendeur. Sa conversion était nécessaire afin de la quantifier, et ainsi d'évaluer un mouvement de jeu adapté à tout patient, quels que soient les critères de sexe, d'âge et de musculature.

Sur l'opposition nécessaire à la pince tri-digitale de la figure 2.2 vient s'ajouter l'index.



FIGURE 2.4 – Pince tri-digitale : opposition pouce/majeur + index

Un capteur tactile thermique serait fixé au niveau de celui-ci afin de détecter cet engagement.



FIGURE 2.5 – Capteur thermique tactile

Le patient travaillerait alors la fréquence de l'engagement de l'index dans la pince tri-digitale.

Un autre système pour ce type d'exercice consisterait à non pas essayer de superposer l'appareil de rééducation à la main, mais de considérer cet appareil comme une réelle manette de jeu. Celle-ci serait sous forme de stylo tri-face, permettant logiquement de simuler au plus près l'exercice d'écriture. Voici un exemple de maquette.



FIGURE 2.6 – Maquette tri-face d'une manette de jeu

Il ne s'agit pas vraiment là d'un stylo, mais simplement d'un exemple de maquette d'une manette de jeu ne revêtant pas la main. Il y aurait un capteur sur chaque face (2 de pressions, un tactile) et la pince tri-digitale serait effectuée (voir figure 2.4). Un problème nous est vite apparu avec ce type d'appareil, il s'agit de la zone de contact des doigts : rien n'empêche à un mouvement mal effectué (voir figure 2.3) d'activer un des 3 capteurs, spécialement pour le pouce car l'amplitude des deux autres doigts étant limité dans ce mouvement de pince tri-digitale.

L'option du gant a finalement été retenu.

Pour le deuxième type d'exercice, le gant ou l'exosquelette est inévitable. Des capteurs de flexions seraient fixés au niveau du pouce, de l'index et du majeur afin de transmettre des données sous forme de nombre encodant l'angle de courbure des doigts. Voici les capteurs retenus.



FIGURE 2.7 – Capteur de flexion

Ceux-ci récupèrent l'angle orienté de courbure, en degré et au centième près. Une telle précision nous a paru nécessaire car les patients travailleront la précision dans ce type d'exercice.

Il faut également souligner que les amplitudes maximales de flexion/extension concernant les 3 doigts varient d'un patient à l'autre, et que les mouvements de jeux devaient également être adaptés suivant ce critère.

En ce qui concerne la possibilité de résistance sur ces mouvements, deux options s'offraient à nous : électronique ou mécanique. C'est cette dernière qui fut retenue afin de fournir rapidement un premier prototype, même si le logiciel informatique devait laisser la possibilité de communication avec l'appareil de rééducation pour régler électriquement une résistance présente sur celui-ci. Les résistances mécaniques sont sous forme de bande élastique d'élasticité variable que l'on fixerait sur chacun des 3 doigts pour un prototype de gant. À noter qu'il y aurait un gant pour chaque niveau de résistance mécanique, ou seulement un gant avec possibilité de réglage électronique.

L'ensemble des capteurs seraient reliés à une carte *Arduino* diffusant par bluetooth ou wifi les valeurs des capteurs à intervalle régulier. Les données des capteurs seraient diffusées successivement, périodiquement et munis d'un préfixe permettant d'identifier le capteur et ainsi d'indiquer à l'application l'interprétation qu'elle doit en faire.

*Samia Humeau* souhaitait également avoir la possibilité de visualiser par des graphiques l'activité des patients afin de surveiller leurs avancées et moduler la thérapie en fonction. Une application supplémentaire serait présente sur son ordinateur, permettant de tracer une liste de représentation des données recueillies sur l'appareil. Elle souhaitait également respecter l'anonymat des patients et ainsi le secret médical, en accord avec la déontologie et l'éthique que nécessite son travail.

L'application *Android* devait alors répondre aux besoins induits par les contraintes exposées dans cette introduction.

# Chapitre 3

## Réalisation personnelle

Mon travail a été de mettre en oeuvre l'application *Android*, de récupérer les données émises par la carte *Arduino* et les faire traiter par l'application.

### 3.1 Choix des technologies

Une première version du *Graphogame* avait été fournie. L'application avait été développée à l'aide de *Scratch*, outils de programmation destiné aux enfants et apprentis programmeurs. Cependant, un problème surgit rapidement : les besoins clients augmentèrent et la version maximale possible de l'application *Android* fût rapidement atteinte, n'offrant aucune possibilité de développement futur et d'améliorations. Je devais donc choisir un ensemble de technologies permettant une reprise du projet, et répondant à des besoins continuellement croissants.

Une décision que je pris fût de séparer l'application *Android* des jeux vidéos, de manière à laisser la possibilité d'ajouter des jeux futurs sans toucher à l'application dont le rôle principal est de programmer ces derniers dans un emploi du temps.

En ce qui concerne l'application, elle fût développée sous *Android studio*. C'est un environnement de développement *Java/Kotlin* pour développer des applications mobiles *Android*. Il s'agit de l'*IDE* recommandé par *Android* et offre donc une possibilité de développement correspondant à une application *Android* quelconque. Il est basé sur *IntelliJ IDEA* et utilise le moteur de production *Gradle* permettant de joindre des projets *C++* à l'application sous forme de module. Les projets *C++* doivent utiliser la *Java Native Interface (JNI)* pour pouvoir être intégré à un projet *Android studio*. La *JNI* est une bibliothèque logicielle d'interfaçage, intégrée nativement au kit de développement *Java (JDK)*, qui permet au code *Java* s'exécutant à l'intérieur de la machine virtuelle *Java (JVM)* d'appeler et d'être appelé par des applications natives (c'est-à-dire des programmes spécifiques au matériel et au système d'exploitation de la plate-forme concernée), ou avec des bibliothèques logicielles basées sur d'autres langages (*C*, *C++*, assembleur, etc).

L'intégration de projets *C++* dans *Android Studio* est donc en théorie possible. Un certain nombre d'interfaces de programmation pour jeux vidéo sont disponibles en *C++* dont la *Simple Directmedia Layer (SDL)* et la *Simple and Fast Multimedia Library* qui a pour but de proposer une alternative orientée objet à la *SDL*. Je me suis dirigé vers la *SFML*, jugeant l'approche objet plus compréhensible et rapide à reprendre.

Des liens sont à disposition dans l'annexe « Liens utiles » afin de faciliter la prise en main des technologies pour une reprise future.

## 3.2 Présentation de l'application

### 3.2.1 Organisation générale

Nous allons voir ici comment concrètement se servir de l'application et la suite d'étape correspondant au déroulement d'une utilisation de celle-ci.

En cliquant sur l'icône de l'application, la page d'accueil est affichée. Le thérapeute doit tout d'abord programmer des activités à faire pour le patient. Pour cela, un accès à l'interface thérapeute est nécessaire. En cliquant sur le bouton approprié, un champ demandant un mot de passe est affiché. Le mot de passe est par défaut initialisé en base de données à 'admin'.



FIGURE 3.1 – Page d'accueil

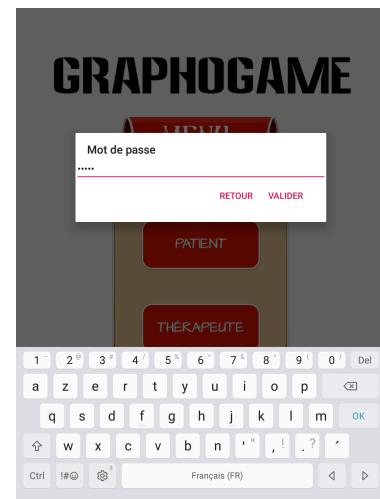


FIGURE 3.2 – Accès à l'interface thérapeute

Il s'affiche alors l'interface permettant principalement au thérapeute de programmer des activités. Pour se faire, en cliquant sur les boutons appropriés, un patient doit être créé puis une activité. Un récapitulatif de ces dernières est affiché dans l'interface thérapeute, dans l'ordre de leur horaire.



FIGURE 3.3 – Interface thérapeute

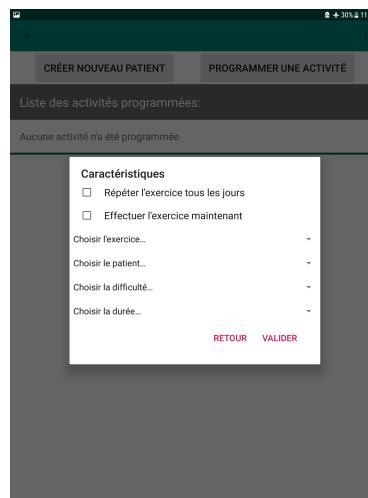


FIGURE 3.4 – Formulaire de création d'activité

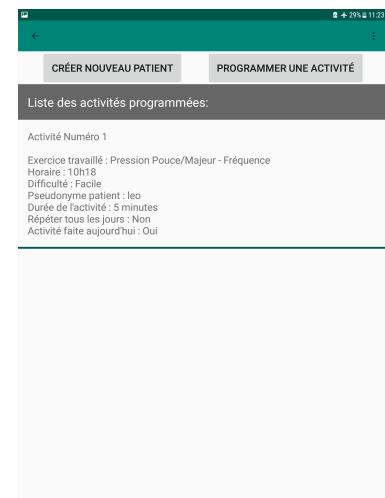


FIGURE 3.5 – Récapitulatif de l'activité

Si l'option « Effectuer l'exercice maintenant » n'est pas été cochée, une horloge s'affiche alors permettant de sélectionner un horaire de programmation d'activité, correspondant au moment auquel le patient pourra commencer l'activité dans la journée. Le thérapeute peut alors retourner à la page d'accueil en cliquant sur la flèche de retour en haut à gauche de l'écran, et laisser la tablette au patient.

Celui-ci va à son tour cliquer sur le bouton redirigeant vers l'interface lui étant dédiée. Une demande d'activation bluetooth est affichée s'il ne l'est pas déjà. Une liste des appareils auparavant apparus est exhibée. Si l'appareil de rééducation n'apparaît pas, l'utilisateur doit se rendre dans les paramètres bluetooth de la tablette afin de former une paire. En le sélectionnant dans l'application, la première activité disponible lance le jeu lui étant associé.

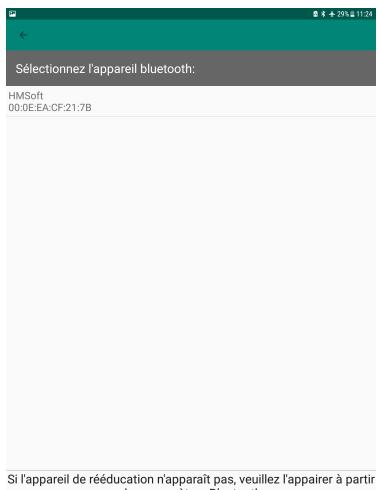


FIGURE 3.6 – Interface patient

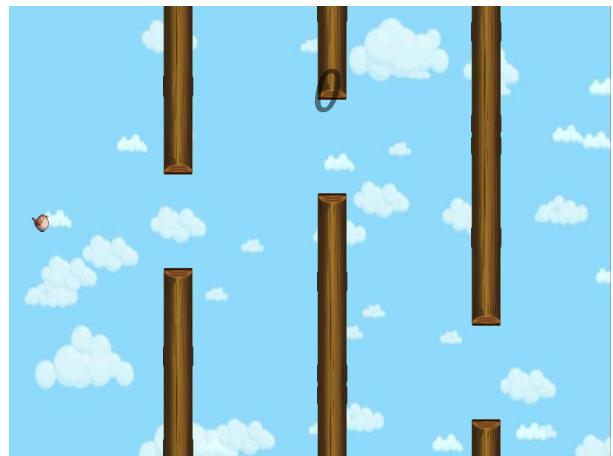


FIGURE 3.7 – Jeu de la première activité programmée chronologiquement disponible

L'utilisateur n'a plus qu'à se munir de l'appareil de rééducation et suivre les instructions de jeu indiquées. Lorsque la première activité est terminée (en accord avec la durée sélectionnée lors de sa création), la suivante est lancée automatiquement.

Il faut préciser que tous les cas d'utilisation non conventionnelles ne sont pas exhibés ici, mais ont été traités afin de prévenir toute anomalie. Typiquement, si l'utilisateur ferme le bluetooth après la demande d'activation dans l'interface patient, une demande est renouvelée si un appareil bluetooth est sélectionné, etc.

### 3.2.2 Contraintes traitées imposées par les besoins client et technologies

La solution de diffusion bluetooth a été retenue afin de garantir un service de rééducation « à la maison » dans toute condition. En effet, aucun service extérieur n'est alors nécessaire (pas de wifi), mais uniquement le matériel *Graphogame*. La carte *Arduino* émet les données sur une socket bluetooth.

Une précision est nécessaire. Une contrainte a ici été imposée par les technologies choisies. Après quelques recherches, il m'est apparu que les sockets bluetooth en C++ ne sont pas opérationnelles sur tout appareil *Android*. Le projet devait être adapté à tout appareil le but final étant que celui-ci soit étendu et disponible pour tout graphothérapeute, les jeux ne pouvaient alors pas directement récupérer les données des capteurs. Celles-ci sont en fait récupérées par l'application qui tourne en tâche de fond sur une socket bluetooth, puis envoyées au jeu lancé (correspondant à un autre processus) sur une socket *TCP* en localhost, qui sont enfin récupérées sur cette dernière socket puis traitées dans un thread isolé par le jeu.

À noter que même si les données des capteurs avaient étaient directement récupérées par le jeu, une communication jeu/application était de toute façon nécessaire afin, par exemple, de récupérer les données depuis l'application pour les sauvegarder (dans le but de tracer des graphiques et visualiser l'avancée des patients, voir

chapitre 2). Cette option m'a donc semblait satisfaire le plus de besoins, particulièrement car elle n'introduisait aucun retard utilisateur visible de réactivité.

Cependant, si pour une quelconque raison future la récupération des données par les jeux devenait nécessaire, une concession serait à choisir : discriminer certains appareils quant à leur compatibilité avec le projet, ou bien imposait une carte Arduino par diffusion wifi et par conséquence directe la nécessité d'un service wifi. Ou bien changer les technologies !

Dans une boucle, la carte *Arduino* exécute une séquence périodique d'envois de la donnée de chaque capteur, sachant que chaque donnée est envoyé à intervalle régulier (de l'ordre de la dizaine de millisecondes) afin de garantir une réactivité de jeu transparente à l'utilisateur.

Comme indiqué dans l'introduction, une quantification des données est nécessaire. Pour chaque jeu relatif à la force et l'amplitude de flexion/extension, une étape d'initialisation de seuil est donc nécessaire avant le démarrage du jeu. Une remarque peut être faite ici : on pourrait imaginer des paramètres d'amplitude, force etc directement enregistrés pour chaque patient et évaluer dans un mode spécial sous le regard du thérapeute. Les seuils de jeu seraient alors choisis selon le maximum du seuil mesuré pour cette activité (dans le cas d'un progrès non encore mis à jour dans le profil du patient) et le seuil défini pour le patient, ce qui permettrait de limiter les « tricheries ».

Le thérapeute devait avoir la possibilité de contrôler la succession et l'horaire des exercices. Ceux-ci sont donc effectués suivant une liste triée selon leur horaire, ou leur instant de création si les horaires sont les mêmes.

En ce qui concerne le secret médical, le nom des patients ne devaient pas être visible depuis l'interface thérapeute. Il suffit donc au thérapeute de choisir un pseudonyme pour le patient, ce qui mettrait automatiquement à jour l'application tierce de traçage de graphique (cf « Application tierce de représentation des données patient » du chapitre 4), qui serait équipée d'une correspondance pseudonyme/patient.

### 3.3 La base de données

Une base de données était nécessaire afin de rendre certaines données persistantes dans l'application *Graphogame*. La SDK *Android Studio* utilise le moteur de base de données *SQLite*. Voici le schéma entités-associations modélisant les données du problème.

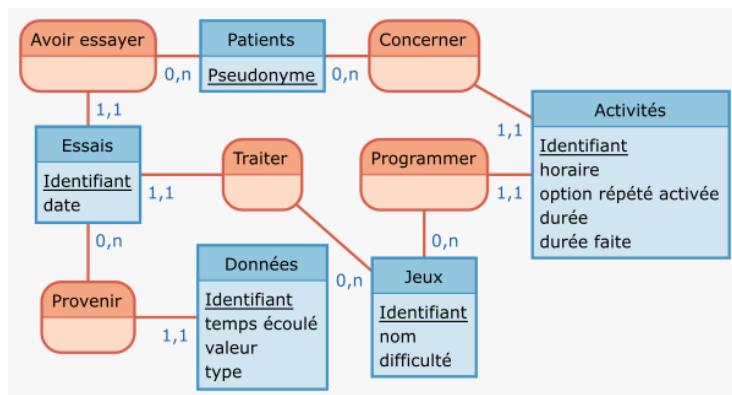


FIGURE 3.8 – Schéma entités-associations

Tout d'abord, précisons l'attribut « temps écoulé » de la table « Données ». Il s'agit de la durée (en millisecondes) écoulée depuis le début de l'essai auquel il est relié.

Une instance de la table « Essais » est un essai d'activité, donc d'une instance de la table « Activités ». Les deux tables sont intrinsèquement liées, et on peut remarquer qu'elles ont des associations avec leurs cardinalités en communs. Il est légitime de se demander pourquoi nous ne les avons pas fait hériter d'une entité commune ou même pourquoi est ce qu'il n'y a aucune association entre elles.

En fait, il s'agit d'un choix pratique. Les essais d'activités sont inscrits durablement dans la base de données alors que les activités n'y sont présentes que le temps de leur durée de vie. En prenant en compte ce fait, les représentations d'héritage *SQLite* nécessiterait plus d'espace pour pas grand chose (une table en plus et donc des clés étrangères dans les deux classes filles), car très peu d'activités sont présentes en base de données quel que soit l'instant considéré. De plus, si une association les reliait, la clé étrangère présente dans la table « Essais » pourrait pointer vers une instance de la table « Activités » n'existant plus, d'où le problème.

Ensuite, le schéma entités-associations ne représente pas la base de données actuelle mais bien une modélisation des données du problème.

En réalité une table supplémentaire existe, la table « Thérapeute », munie de deux attributs : « email » et « mot de passe ». Cependant, il n'y aura qu'une unique instance dans l'application. La table n'a donc pas de clé primaire et ne serait reliée à aucune association dans le schéma si elle y était représentée. Il s'agit en fait simplement d'une donnée unique qu'il fallait rendre persistante, je l'ai donc jugée inutile à représenter sur le schéma.

La table « Activités » possède également deux attributs supplémentaires. Le premier « fait ? » indique si l'activité a été fait au jour actuel, qui peut être déduit du signe de la soustraction de « durée faite » à « durée » ; l'attribut « durée faite » ayant été introduit très tardivement, « fait ? » a été gardé pour un soucis de temps (beaucoup de modification de code à effectuer) et très peu de résultat (le nombre d'activité en base de données étant très limité en réalité, le gain en espace mémoire est ridicule). Le deuxième attribut est nécessaire pour la fonction lançant la notification liée à l'activité, il s'agit d'un choix d'implémentation non requis pour la modélisation.

En ce qui concerne les migrations, voir l'annexe « Migrations de la base de données ».

## 3.4 Interface thérapeute

Il s'agit de l'interface de programmation des activités à faire par les patients.

Les fonctionnalités principales ont déjà été présentées dans « Organisation générale ». Cependant, faisons quelques remarques. Tout d'abord, en cliquant sur un résumé d'activité (figure 3.5), une fenêtre s'affiche proposant de la supprimer. Ensuite, en haut à droite de l'interface nous pouvons voir l'icône standard d'un menu sur la barre d'action, qui s'affiche lorsque l'icône est cliqué. Des options s'affichent alors, permettant de modifier le mot de passe et l'email de notification thérapeute, et également d'exporter les données de capteur en base de données sous la forme d'un fichier.csv visualisable par n'importe quel tableur. L'option de téléchargement vers *Drive* est recommandée car a été testée. Voir l'annexe « Exporter votre premier fichier.csv » pour introduire des données en base de données et le tester vous-même.

## 3.5 Interface patient

Il faut préciser que si aucune activité n'a été programmée, le « mode libre » (vide pour l'instant) dans lequel le patient pourrait choisir un jeu et une difficulté est lancé, afin d'effectuer des exercices supplémentaires s'il le souhaite (d'où le but de rendre les jeux le plus attractif possible!).

Il faut également préciser qu'à propos des résistances mécaniques évoquées en introduction, le code du thread tournant en arrière fond des jeux et traitant la socket bluetooth contient également une fonction permettant d'écrire sur la socket, qui ne sert pas encore et servirait à indiquer à l'appareil *Graphogame* la résistance désirée/adéquate.

### 3.5.1 Exercices de pression

Nous avons vu, dans l'introduction, deux types d'exercice de pression (force et fréquence). Ceux-ci sont effectués sur le même jeu. Lorsqu'il est lancé, des instructions sont affichées en fonction de l'exercice. Le patient suit les instructions puis peut commencer à jouer. Tant que le mouvement de jeu n'a pas été effectué, le personnage reste immobile.

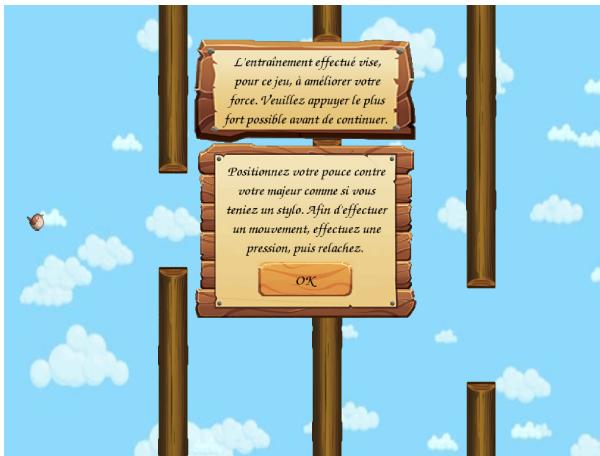


FIGURE 3.9 – Règles du jeu de force

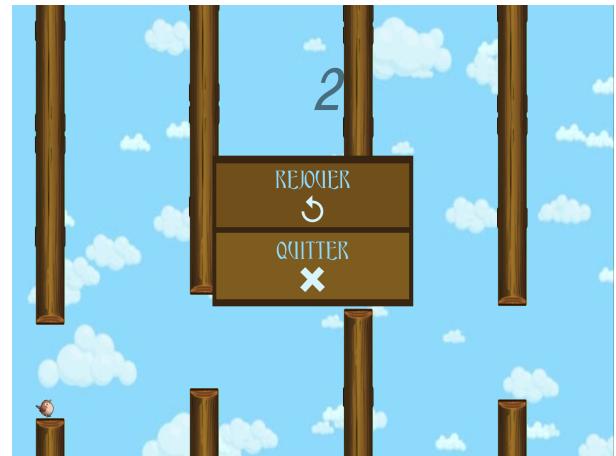


FIGURE 3.10 – Menu de jeu

Lorsqu'un mouvement est effectué (simple pour l'exercice de fréquence ou pression à 80% de la force mesurée pour l'exercice de force, voir figure 3.9) en dehors du menu ou des règles de jeu, le personnage effectue un saut, puis retombe par la gravitation. Les obstacles défilent alors vers la gauche à vitesse constante. L'ouverture des obstacles est générée aléatoirement, mais sans être trop espacée de celle qui la précède et la suit, afin d'éviter les cas extrêmes (ouverture tout en bas puis tout en haut). Le score correspondant au nombre d'obstacles passé est affiché en continu. Des chants d'oiseaux en musique de fond peuvent être entendu, et un son de score est émis à chaque obstacle passé.

La partie est perdue si le personnage passe au-dessous de l'écran ou touche un obstacle, et le menu s'affiche alors. Si la durée totale de l'activité est passée, un menu ne permettant que de quitter l'application est affiché.

La durée de jeu effective est calculée et ajoutée à la « durée faite » en base de données (voir « La base de données »). De cette manière, si un patient doit effectuer 10 minutes ce jour mais doit partir et ferme l'application après 5 minutes, sa progression est sauvegardée. La durée écoulée pour l'essai commence à partir du moment où le premier mouvement effectué, jusqu'à ce que le menu soit affiché. Ensuite, cette durée est envoyée à l'application qui met à jour la base de données en tâche de fond. Ainsi, un patient ne peut pas tricher et « faire l'activité » sans véritablement jouer.

### 3.5.2 Exercices de flexion

Le jeu de pression n'a pas pu être ajouté à l'application à temps. Sur les liens *Github* du projet, un accès à une version *Ubuntu* est fournie afin de visualiser le résultat que l'on souhaite obtenir.

Voici à quoi ressemble le jeu de flexion sur tablette.

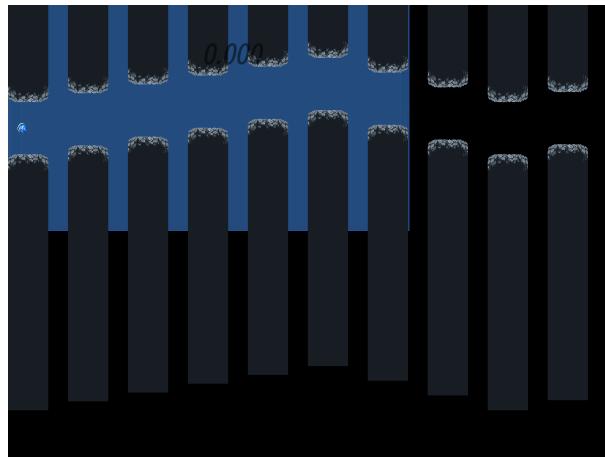


FIGURE 3.11 – Jeu de flexion sur l'application *Graphogame*

Nous pouvons remarquer que le jeu nécessite d'être adapté à l'écran de la tablette, cependant ce n'est pas la seule modification à apporter, voir « Jeu de flexion et autres jeux » du chapitre 4 pour une présentation complète. En effet, le capteur de flexion n'ayant pas pu être fonctionnel avant la fin du stage, il m'était impossible de réellement le mettre en place sur tablette, même si l'adaptation graphique aurait pas exemple pu être effectuée. Cependant, l'intégralité du code à lui associer étant très similaire au premier (par exemple sur la transmission des données), je me suis concentré à fournir un premier jeu complètement fonctionnel et sans bug (cas utilisateurs testés).

Et voilà la version ordinateur.

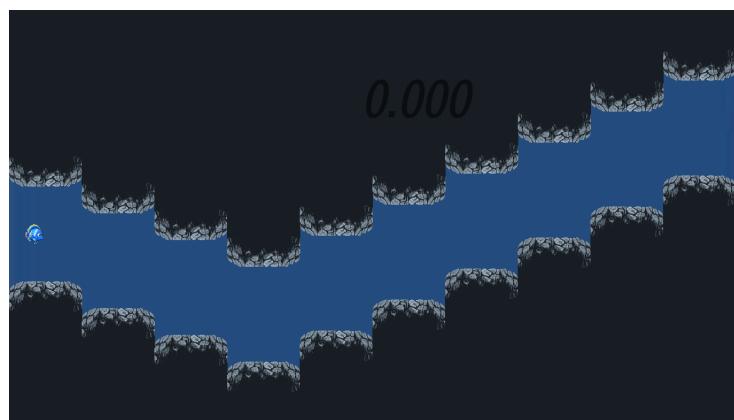


FIGURE 3.12 – Jeu de flexion sur ordinateur

Le score de jeu est alors la durée écoulée depuis le début de l'essai, et un son est produit toutes les 10 secondes. Il nous faut remarquer qu'il faut appuyer sur espace pour démarrer le jeu puis les flèches haute et basse pour effectuer le mouvement associé.

Sur la version *Graphogame*, un mouvement sera tel que la position tout en haut de l'écran correspondra à l'extension maximale et inversement pour la flexion, ou presque pour ne pas toucher l'obstacle car si l'on veut faire travailler l'amplitude du mouvement il faudrait pouvoir aller à l'amplitude maximale (à moins de créer un autre jeu car le but visé pour celui-ci était initialement de travailler la précision).

# Chapitre 4

# Continuité du projet et améliorations

C'est dans cette partie que je présente comment continuer le projet et par où commencer. Si vous reprenez le projet, ce chapitre est spécialement pour vous.

## 4.1 À développer en priorité

Étant donné mon départ prématué du stage, dû à l'option de 3<sup>me</sup> année, il m'a manqué deux semaines afin finir le développement de certains aspects du projet que je vais présenter ici.

### 4.1.1 Jeu de flexion et autres jeux

Comme indiqué dans la section « Exercices de flexion » du chapitre 3, le jeu d'exercice de flexion n'a pas pu être adapté à temps à l'application.

Une des premières choses à faire est d'adapter la taille des images à tout écran. Le jeu a initialement été adapté pour la résolution de mon ordinateur, sans l'avoir été aux autres résolutions. Pour se faire, il suffit de multiplier toutes images chargées par un coefficient, correspondant à la taille de la fenêtre *SFML* (résolution plein écran de l'appareil) divisé par la résolution de mon ordinateur (1366 x 768).

Ensuite, il faut ajouter les menus de jeux, l'affichage des règles, la communication application/jeu (dans l'application, il s'agit du code dans l'activité *java* implémentant le travail d'arrière fond effectué ; dans le jeu, il s'agit du thread récupérant les données. À reprendre en quasi-totalité sur le jeu d'exercice de pression implémenté.

La position du personnage de jeu serait alors implementée selon l'angle de flexion/extension effectué avec le doigt, l'extension et la flexion maximales correspondant respectivement à la position vertical maximale et minimale. Une phase d'initialisation récupérant ces deux données serait nécessaire, sinon des zones de jeu seraient inaccessibles pour certains patients et ils perdraient fatallement dans certains cas. Pour se faire, reprendre la phase d'initialisation pour le jeu de force et l'adapter au capteur de flexion.

Néanmoins augmentez significativement la vitesse de jeu par rapport à la version *Ubuntu*, de manière à rendre le jeu ludique et non-trivial, le mouvement de flexion/extension pouvant s'effectuer beaucoup plus rapidement que ce qui a été codé pour le mouvement haut/bas (même pour les patients, et de toute façon la faire varier en fonction de la difficulté de jeu).

Il faut également remarquer que le premier jeu est assez pratique pour l'exercice de force, mais l'exercice de fréquence pourrait être mieux penser. Typiquement, pour faire sauter le personnage, il serait mieux de mon point de vue de forcer l'utilisateur à appuyer un certain nombre de fois très rapidement (sans seuil de force) sur le capteur plutôt que de déclencher le saut dès qu'une pression est ressentie, et en remettant le compteur de fréquence à 0 après un certain laps de temps sans action (une seconde ? Des essais s'imposeraient si l'idée est

retenue).

Je souligne également ici qu'un jeu pour le capteur tactile est également nécessaire lorsque celui-ci sera intégré sur un appareil *Graphogame*, ou cela pourrait être une action de jeu à intégrer en plus de l'opposition pouce/majeur (voir « Mode libre : jeu à sauvegarde et multiexercices »).

#### 4.1.2 Panel de difficultés de jeu

Un panel de difficulté a déjà été mis en place de façon non-fonctionnelle : au niveau de la base de donnée, de l'interface thérapeute, la difficulté de jeu est envoyée par l'application et est même récupérée par le jeu. Il suffit simplement maintenant de la faire traiter par l'application, c'est-à-dire en fonction de sa valeur modifier la vitesse de jeu, l'ouverture des obstacles, etc. Voir l'annexe « Traiter la difficulté de jeu ».

#### 4.1.3 Notifications

##### Patient

Comme toute application, il serait commode que celle-ci envoie des notifications sur l'appareil, par exemple lorsqu'un horaire d'une activité est arrivé.

Des fonctions complètes permettant de lancer une notification et de la supprimer ont déjà été implémentées, et avaient même été initialement utilisées afin de se faire. Néanmoins, avec les dernières avancées du stage, la notification actuelle n'était pas bien supprimée et une nouvelle notification n'était pas non plus démarrée dans tous les cas. J'ai alors préféré enlever toutes les occurrences de démarrage/suppression des notifications afin de ne pas perdre toute personne s'appropriant le projet, et lui laisser le soin de placer ces occurrences au bons endroits. Voir « Fonctions de notification patient » pour les détails.

L'idée était de démarrer une notification lorsque l'horaire d'une activité est dépassée, puis de répéter cette notification toute les 30 minutes jusqu'à ce que l'activité soit faite. Lorsque l'activité est faite, une nouvelle notification est lancée si l'horaire d'une autre activité est dépassée est ainsi de suite. Cette vérification d'horaire ainsi que le lancement de notification ne se feraient évidemment que lorsque l'application n'est pas en premier plan. Enfin, chaque jour, vers minuit, les activités faites et non supprimées (devant donc être répétée tous les jours) doivent être remises à « non faite » en base de données, et en même temps la notification actuelle doit être supprimée - à l'aide d'un *service Android Studio*.

##### Thérapeute

Le thérapeute devait être notifié lorsqu'une activité a été faite par le patient. Pour se faire, il suffit d'envoyer un email de notification à l'adresse mail enregistrée dans la base de données via l'interface thérapeute. À noter que cette solution, qui a été retenue, nécessite internet.

#### 4.1.4 Enregistrement des données

Les données de capteurs doivent être enregistrées afin de permettre au thérapeute, à terme, de visualiser l'avancé des patients. La communication application/jeu étant fonctionnelle, il suffit donc d'accumuler ces données et de les enregistrer en base de données au bon moment. Voir « Données des capteurs en base de données ».

### 4.2 Application tierce de représentation des données patient

Afin de visualiser les avancés des patients par des graphiques représentant la valeur des capteurs de diverses manières (moyenne de force, force maximale, tremblement angulaire, fréquence d'opposition pouce/majeur etc), avec la thérapeute nous avons décidé que cela se ferait sur une application *Windows* installée sur son ordinateur.

Tout d'abord, il s'agira de visualiser les données récupérées par téléchargement direct des fichiers de base de données à partir de l'interface thérapeute, comme évoqué dans la partie « Interface thérapeute ». L'application analysera ces fichiers et permettra d'afficher divers graphiques tels ceux qui suivent, qui représentent des prélevements de capteurs effectués sur les stagiaires (donc supposés « sains »). Tout d'abord pour le capteur de flexion.

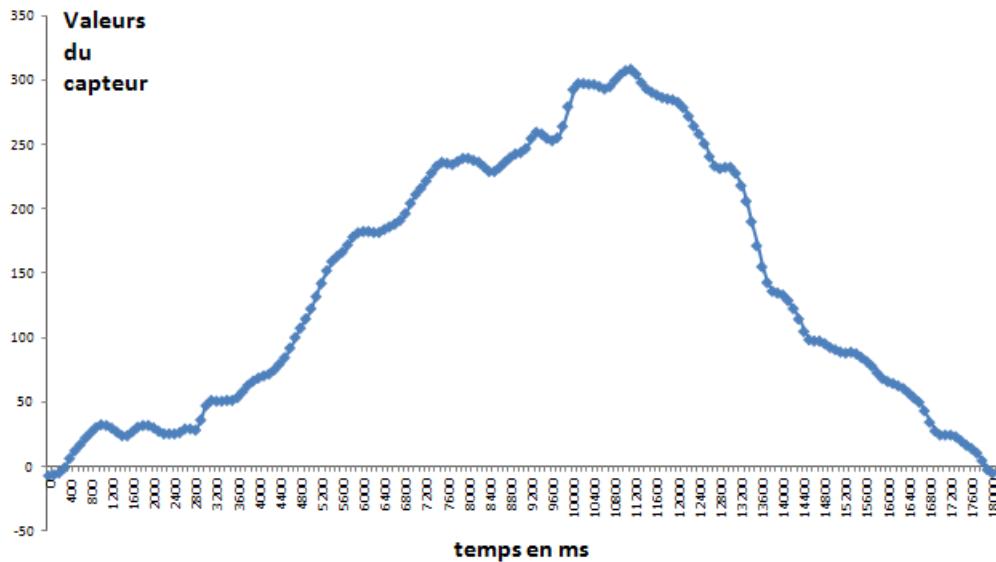


FIGURE 4.1 – Valeur du capteur de flexion en fonction du temps

Le capteur de flexion étant extrêmement précis (au centième de degré près), nous pourrions observer des tremblements angulaires sur des patients. Enfin pour le capteur de pression.

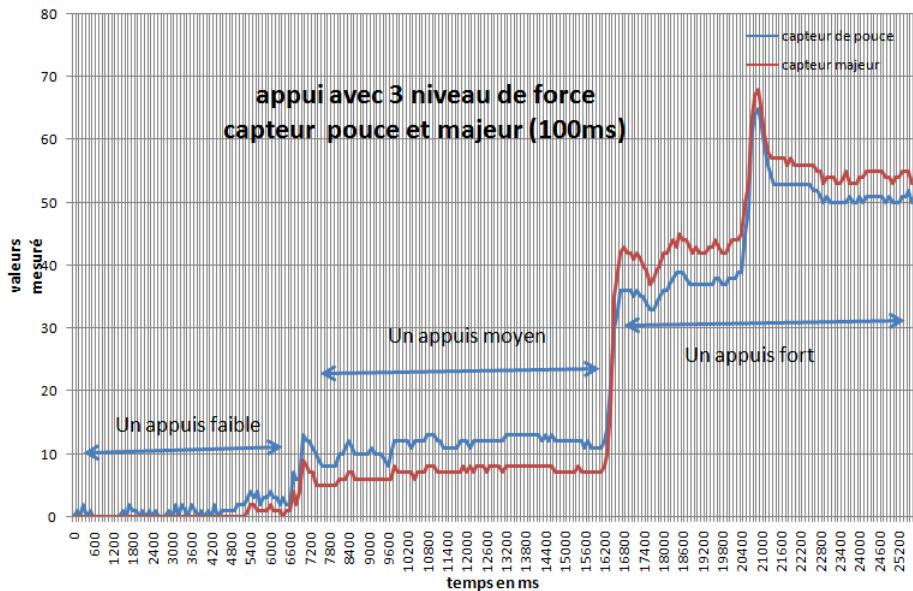


FIGURE 4.2 – Valeur du capteur de pression en fonction du temps à diverses intensités

Sur la figure ci-dessus, il s'agirait d'observer la valeur de force que les valeurs du capteur de pression prélevées représentent. Nous pouvons remarquer également des tremblements visibles pouvant représenter des irrégularités

dans la fonctionnalités de l'opposition pouce/majeur. Les valeurs associées aux pressions effectuées par pouce sont plus importantes que celles perçues sur le majeur lorsqu'elles sont importantes, alors que l'effet inverse est visible pour une pression faible. Nous pouvons attribuer cette différence à l'orientation du vecteur de force (légèrement incliné) exercé par le pouce sur le plan formé par le capteur du majeur (pour un résultat optimal, il faudrait une force perpendiculaire au plan).

L'application devra également être munie, en base de données, d'une correspondance entre le pseudonyme des patients entrés dans l'application *Graphogame* et leur identité.

## 4.3 Versions de l'application

Cette partie présente quelques idées sur ce qui pourrait être développé à la suite de ce qui a été évoqué plus haut. Cependant, il ne s'agit plus de priorités et la décision devra être discutée.

### 4.3.1 Jeux 3D et multijoueurs

Comme indiqué dans la partie « Choix des technologies », la *SFML* inclue un module *OpenGL* et peut donc être utilisé dedans. Le développement de jeu *3D* et de ce fait possible avec ce qui a été mis en place sans aller chercher d'autres bibliothèques *3D* (voir « Liens utiles » pour un exemple en vidéo).

De plus, on pourrait également imaginer un jeu multijoueurs où l'application gèrerait dans chaque thread une socket bluetooth vers un nouvel appareil de rééducation. Dans ce cas, puisque les threads seraient en boucle infinis, il faudrait un cœur par thread (tout thread inclus, pas seulement ceux pour la gestion des données d'un joueur, par exemple pour le processus du jeu le thread principal de la boucle de jeu). Il faudrait alors sûrement restreindre l'application à certains appareils possédant l'architecture adéquat (ou pour la version multijoueurs spécifiquement).

### 4.3.2 Mode libre : jeu à sauvegarde et multiexercices

Le développement du mode libre, qui est pour l'instant une vue vide, pourrait être développé. L'ensemble des jeux seraient disponibles ainsi que le choix de la difficulté. Les données de capteur n'auraient pas besoin d'être enregistrées dans ce mode puisque la thérapeute pourrait programmer des activités pour en récupérer.

Ce mode pourrait justement permettre de nouveaux types de jeux, les *RPG* (Role Playing Game). Plusieurs actions seraient alors disponibles pour l'utilisateur, activées par différents mouvements disponibles sur l'appareil *Graphogame*. L'objectif est évidemment d'amener le patient, de sa propre volonté, aux exercices de rééducation.

### 4.3.3 Évaluer l'évolution du patient : utilisation d'une intelligence artificielle

Un réseau de neurones pourrait également être utilisé afin de décider, lorsqu'un patient retourne chez lui avec le *Graphogame*, les paramètres de jeu afin d'adapter une difficulté globale permettant à l'utilisateur d'effectuer des parties à la fois faisables et compétitives. Ce réseau de neurones pourrait s'entraîner sur les données récupérées sur les patients, à condition de noter les résultats attendus quelque part (résultats en terme de difficulté adéquate). L'intelligence artificielle adapterait la difficulté sur un ensemble de paramètres : par exemple, pour le jeu « Exercices de pression », l'ouverture des obstacles, la vitesse de défilement des obstacles et avec la gravité du personnage (car les deux sont liés à peu de choses près, sinon le personnage ne pourrait pas atteindre des obstacles situés en haut et précédé par un obstacle trop bas), et pourquoi pas la hauteur du saut également pourraient être modifiés.

Afin de prévenir la triche, le réseau de neurones pourrait également prendre en compte les données enregistrées pour le patient dans le mode spécial décrit dans la partie « Contraintes traitées imposées par les besoins client et technologies ».

# Chapitre 5

## Conclusion

Pour conclure le *Graphogame* n'est pas dans une version finale mais est au contraire toujours en plein développement. Les technologies ont justement été choisies dans le but de laisser la possibilité d'un développement futur très important, contrairement à l'ancienne application codée avec *Scratch*.

Des tests utilisateurs ont été effectués afin de tester tous les cas d'utilisation, par exemple lorsque l'appareil de rééducation n'est plus connecté en bluetooth à l'application (hors de portée, appareil éteint etc) en plein partie de jeu. Cependant, les tests unitaires n'ont pas eu le temps d'être développés. À noter qu'*Android Studio* fournit un environnement de test.

L'ensemble du code a fait sujet d'une documentation minutieuse afin de faciliter toute reprise du projet.

Je me permets de préconiser une connaissance d'*Android Studio* (pas spécialement de *Gradle* ni de la *JNI* mais d'une connaissance de l'interface) ainsi que d'une interface de programmation *C++* pour jeux vidéo telle que la *SDL* ou la *SFML*, ou encore mieux *OpenGL*.

### 5.1 Bilan Personnel

J'ai appris l'importance de deux points principaux dans la gestion de projet. Premièrement, après le choix des technologies, s'occuper du commencement de l'emboîtement des différentes étapes et vérifier la cohérence du choix des technologies en fonction du résultat plutôt que de commencer une étape en elle-même. Typiquement, se pencher tout d'abord sur la bonne introduction d'un programme *SFML* dans *Android Studio* afin de changer de technologie si des problèmes se présentent plutôt que commencer par l'implémentation de l'application *Graphogame* ou des jeux *SFML*. Deuxièmement, tempérer les exigences clients afin de ne pas se laisser déborder et donner des réponses现实的. En effet, j'ai dû effectuer à peu de chose près autant de mon temps libre que d'heures rémunérées afin de tenir le calendrier prévu.

Enfin, toutes les technologies choisies étaient nouvelles pour moi et m'ont donc apportées des connaissances (*SFML*, *Android Studio* et même *JNI*, que je n'utilise qu'à travers la *SFML* mais que j'ai pris en main en essayant d'introduire un programme *SFML* dans *Android Studio*). J'ai également pris conscience de la difficulté de certains problèmes en temps raisonnable pour les jeux vidéos tels que la détection de collision *2D* et *3D* pour la paire personnage/obstacle, approximée mais proche de parfaite.

### 5.2 Bibliographie

- <https://www.ims-bordeaux.fr/fr/ims/le-laboratoire-ims>
- [https://fr.wikipedia.org/wiki/Android\\_Studio](https://fr.wikipedia.org/wiki/Android_Studio)
- [https://fr.wikipedia.org/wiki/Scratch\\_%28langage%29](https://fr.wikipedia.org/wiki/Scratch_%28langage%29)
- [https://fr.wikipedia.org/wiki/Java\\_Native\\_Interface](https://fr.wikipedia.org/wiki/Java_Native_Interface)

# Chapitre 6

## Annexes

Tous les répertoires et fichiers cités sont situés dans le répertoire du projet.

### 6.1 Ajouter un jeu C++ à l'application

Je vais tout d'abord vous montrer comme intégrer un jeu *SFML*.

Pour commencer, installez le projet **avec les versions logiciels indiquées** sur la page *GitHub* (voir « Liens utiles »). Nous allons ensuite créer une librairie C++ à partir des fichiers formant le jeu à l'aide d'*Android Studio*, puis la lancer dans une activité Java.

Pour se faire, rendez-vous dans le répertoire *jni* et créez un répertoire contenant les fichiers de votre jeu. De même, ajoutez tout fichier multimédia dans le répertoire *assets*.

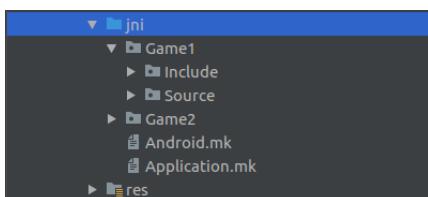


FIGURE 6.1 – Répertoires de *jni* et de jeux

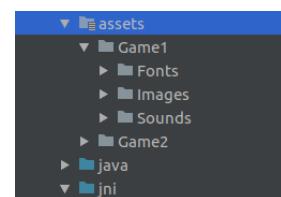


FIGURE 6.2 – Répertoire *assets* et fichiers multimédia

Ouvrez ensuite les fichiers *Android.mk* et *Application.mk* du répertoire *jni*.

```
1 NDK_TOOLCHAIN_VERSION := 4.9
2 APP_PLATFORM := android-24
3 APP_STL := c++_shared
4 APP_ABI := armeabi-v7a
5 APP_MODULES := sfml-activity-d Game1 Game2
6 APP_OPTIM := debug
7 APP_CFLAGS := -g -ggdb -O0
```

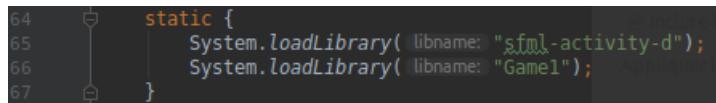
FIGURE 6.3 – Fichier *Application.mk*

```
39 include $(CLEAR_VARS)
40 LOCAL_MODULE := Game2
41 LOCAL_C_INCLUDES := $(LOCAL_PATH)/Game2/Include/
42 LOCAL_SRC_FILES := \
43     Game2/Source/LaunchGame.cpp \
44     Game2/Source/Background.cpp \
45     Game2/Source/Character.cpp \
46     Game2/Source/Fence.cpp \
47     Game2/Source/Game.cpp \
48     Game2/Source/GameSound.cpp \
49     Game2/Source/Input.cpp \
50
51 LOCAL_SHARED_LIBRARIES := sfml-system-d
52 LOCAL_SHARED_LIBRARIES += sfml-window-d
53 LOCAL_SHARED_LIBRARIES += sfml-graphics-d
54 LOCAL_SHARED_LIBRARIES += sfml-audio-d
55 LOCAL_SHARED_LIBRARIES += sfml-network-d
56 LOCAL_SHARED_LIBRARIES += sfml-activity-d
57 LOCAL_SHARED_LIBRARIES += openal
58 LOCAL_WHOLE_STATIC_LIBRARIES := sfml-main-d
59
60 include $(BUILD_SHARED_LIBRARY)
```

FIGURE 6.4 – Fichier *Android.mk*

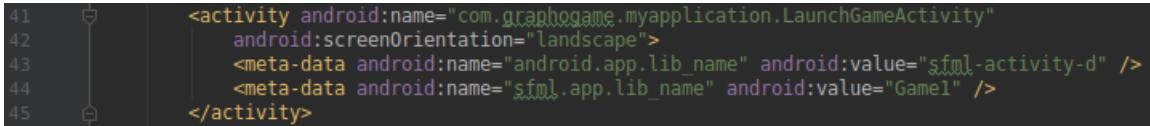
Dans le fichier *Application.mk*, ajouter un module correspondant à votre jeu (ligne 5). Dans *Android.mk*, ajoutez les lignes correspondant à la figure 6.4 en adaptant évidemment au nom de module indiqué précédemment et aux fichiers de votre jeu. *Android Studio* et fin prêt à construire la bibliothèque de jeu, voyons désormais comment lancer votre jeu dans l'application.

Créez une nouvelle activité *Java* qui s'occupera du job effectué tâche de fond. Pour se faire, reprenez le code de *LaunchGameActivity.java* du répertoire *java* et ajoutez les de la figure 6.5, sans oublier de la faire hériter de la classe *NativeActivity* indiquant à *Android Studio* qu'elle peut utiliser du code natif *C++*. Déclarez également la nouvelle activité *java* dans le fichier *AndroidManifest.xml* comme sur la figure ??.



```
64     static {
65         System.loadLibrary( libname: "sfml-activity-d" );
66         System.loadLibrary( libname: "Game1" );
67     }
```

FIGURE 6.5 – Fichier *LaunchGameActivity.java*



```
41 <activity android:name="com.graphogame.myapplication.LaunchGameActivity"
42     android:screenOrientation="landscape">
43     <meta-data android:name="android.app.lib_name" android:value="sfml-activity-d" />
44     <meta-data android:name="sfml.app.lib_name" android:value="Game1" />
45 </activity>
```

FIGURE 6.6 – Fichier *AndroidManifest.xml*

Vous pouvez désormais utiliser l'activité *Java* créée qui lancera votre jeu lorsqu'elle sera elle-même lancée. Plus exactement, ce sera la fonction *main* présente dans un fichier définit dans *Android.mk* (voir figure 6.4) qui sera appelée.

Créer un jeu et l'intégrer dans l'application avec une autre bibliothèque *C++* (par exemple la *SDL*) se ferait avec une démarche similaire, mais peut-être que l'appel à la bibliothèque se ferait à l'aide de *JNI* (rappel : *OpenGL* est inclus dans la *SFML*) - voir « Liens utiles » pour une introduction complète à *JNI*.

## 6.2 Traiter la difficulté de jeu

La difficulté de jeu est récupérée par le jeu, voir la fonction *readBTBata* exécutée par un thread qui est un attribut d'une instance d'*Input*, dans le fichier *Input.cpp* du répertoire *jni/Game1/Source*.

Il suffit maintenant de faire varier la largeur des obstacles, la vitesse de jeu et gravitation du personnage, et tout paramètre de jeu pertinent en fonction de l'attribut *difficulty* de l'instance de *Game*.

## 6.3 Données des capteurs en base de données

En ce qui concerne l'enregistrement des données en base de données, l'idée est d'enregistrer celles relatives à un essai lorsque celui-ci est fini.

Pour se faire, il serait judicieux de créer l'essai en base de données lorsque le jeu affiche le menu et envoie la durée écoulée pour l'essai à l'application. Il faut donc se reporter au thread *receiveFromGame* du fichier *LaunchGameActivity.java* du répertoire *java* pour créer l'essai - il s'agit du thread recevant la durée et mettant à jour l'activité courante en base de données.

Les données pourraient en même temps être enregistrées en base de données ; cependant, il reste à décider comment les récupérer. Elles pourraient être stockées pour chaque essai par le jeu puis envoyées à l'application

en associant chaque valeur à un « temps écoulé » (colonne de la table « Données », voir « La base de données ») depuis le début de l'essai grâce à l'horloge *partyClock*, attribut de l'instance de *Game*, ou bien depuis l'application mais il reste alors à déterminer cette durée.

## 6.4 Fonctions de notification patient

Les fonctions suivantes dans l'activité *TherapeuticInterface.java* dans le répertoire *java* permettent de lancer et supprimer une notification.

```

766     private void startNotification(Calendar c, int requestCode){
767
768         int notificationSet = settings.getInt("notificationSet", 0);
769         if (notificationSet == 0) {
770             notificationSet = 1;
771             SharedPreferences.Editor editor = settings.edit();
772             editor.putInt("notificationSet", notificationSet);
773             editor.apply();
774             //notificationSet = true;
775
776             AlarmManager alarmManager = (AlarmManager) getSystemService(Context.ALARM_SERVICE);
777             Intent intent = new Intent(TherapeuticInterface.this, NotificationReceiver.class);
778             PendingIntent pendingIntent = PendingIntent.getBroadcast(TherapeuticInterface.this, requestCode, intent, 0);
779
780             //if (c.before(Calendar.getInstance()))
781             //    c.add(Calendar.DATE, 1);
782
783             assert alarmManager != null;
784             alarmManager.setRepeating(AlarmManager.RTC_WAKEUP, c.getTimeInMillis(), 1000 * 60 * 2, pendingIntent);
785         }
786
787     }
788
789     private void cancelNotification(int requestCode) {
790         int notificationSet = 0;
791         SharedPreferences.Editor editor = settings.edit();
792         editor.putInt("notificationSet", notificationSet);
793         editor.apply();
794         AlarmManager alarmManager = (AlarmManager) getSystemService(Context.ALARM_SERVICE);
795         Intent intent = new Intent(TherapeuticInterface.this, NotificationReceiver.class);
796         PendingIntent pendingIntent = PendingIntent.getBroadcast(TherapeuticInterface.this, requestCode, intent, 0);
797         assert alarmManager != null;
798         alarmManager.cancel(pendingIntent);
799     }

```

FIGURE 6.7 – Fonctions de notifications

Il reste maintenant à les appeler aux bons endroits dans le code. À noter que ces fonctions pourraient être regroupées dans une classe *BroadcastReceiver* à part et pourrait être utilisées ainsi par d'autres activités *java* (telles que l'interface patient, afin de supprimer la notification d'une activité faite).

Notons de plus que chaque jour, les notifications doivent être réinitialisées et en même temps les activités ayant été faites comme indiqué dans la partie « Patient » de la section « Notifications ».

## 6.5 Exporter votre premier fichier.csv

Nous allons voir comment essayer l'exportation d'un fichier.csv, car pour l'instant aucune donnée de capteur n'est enregistrées en base de données donc aucune donnée n'est présente dans le fichier.csv téléchargé. Nous allons donc en injecter « à la main ». Pour se faire, désinstallez l'application sur la tablette *Android*. Ensuite, aller dans le fichier *MydbHandler.java* dans le répertoire *java* et décommentez ces lignes de la fonction *onCreate*.

```

169
170     db.execSQL("INSERT INTO " + TABLE_TRY + "(" + KEY_TRY_ID + "," + KEY_DATE
171             + "," + KEY_GAME_REFERENCE + "," + KEY_PATIENT_REFERENCE + ") values(0, '7/24/2019 14:59:35', 2, 'Patient')");
172     db.execSQL("INSERT INTO " + TABLE_DATA + "(" + KEY_DATA_ID + "," + KEY_VALUE
173             + "," + KEY_ELAPSED_TIME + "," + KEY_DATA_TYPE + "," + KEY_TRY_REFERENCE + ") values(0, -100, 6, 'Pression Pouce', 0)");

```

FIGURE 6.8 – Injection d'une donnée en base de données

Réinstallez l'application, l'option est prête à être utilisée et le fichier contient la donnée correspondante. Le téléchargement dans *Drive* a été testé et est recommandé. Vous pouvez également essayer de cocher l'option de suppression des données dans la fenêtre s'affichant à l'issu du téléchargement.

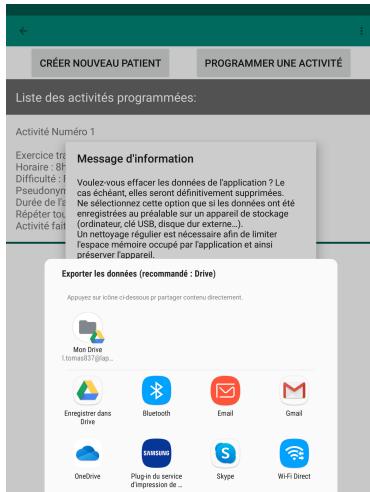


FIGURE 6.9 – Sélection de l'outils de redirection du téléchargement

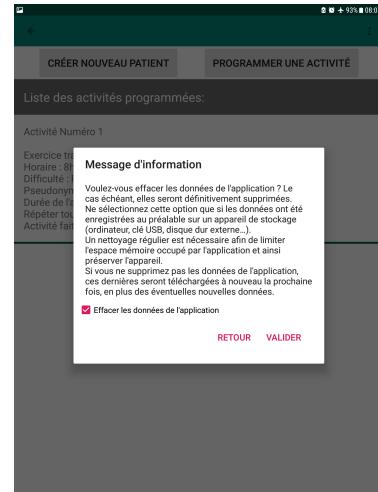


FIGURE 6.10 – Réinitialisation des données en base de données

Téléchargez à nouveau les données, et observer le vide du contenu du fichier indiquant qu'aucune donnée n'est désormais en mémoire.

## 6.6 Migrations de la base de données

Pour l'instant, aucune donnée n'est enregistrée en base de données, donc lorsqu'un changement dans la structure de la base de donnée est effectué, il suffit de désinstaller l'application, appliquer les changements dans la fonction *onCreate* de *MydbHandler.java* dans le répertoire *java* puis réinstaller l'application.

```

179 /**
180 * Fonction appelée lorsque la version de la base de données est incrémentée.
181 * @see #DATABASE_VERSION
182 * Cependant, il s'agit là d'une suppression et d'une refonte complète de la base de données. À ne pas utiliser sans précaution, se reporter au rapport.
183 */
184 @Override
185 public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
186     db.execSQL("DROP TABLE IF EXISTS " + TABLE_ACTIVITY);
187     db.execSQL("DROP TABLE IF EXISTS " + TABLE_TRY);
188     db.execSQL("DROP TABLE IF EXISTS " + TABLE_DATA);
189     db.execSQL("DROP TABLE IF EXISTS " + TABLE_GAME);
190     db.execSQL("DROP TABLE IF EXISTS " + TABLE_THERAPIST);
191     db.execSQL("DROP TABLE IF EXISTS " + TABLE_PATIENT);
192
193     onCreate(db);
194 }

```

FIGURE 6.11 – Fonction actuelle de migration

Cependant, lorsque l'enregistrement des données sera effectif, il ne faudra pas supprimer l'application de la tablette, car cela entraînera la suppression de la base de données et de toutes les données présentes. Le téléchargement du fichier.csv propose déjà de supprimer les données de toute manière pensez-vous, il suffit en fait de télécharger les données présentes avant la désinstallation. Néanmoins, tous les patients seront pas exemple supprimés de l'application. Il est plus raisonnable d'implémenter correctement la fonction *onUpdate* de *MydbHandler.java* dans le répertoire *java* en gérant les différents cas de versions de base de données, en modifiant les tables préexistantes (*ALTER*) et les données déjà présentes (*UPDATE*).

## 6.7 Liens utiles

- <https://github.com/ltomas837/Game2> : lien *GitHub* du projet.
- <https://github.com/ltomas837/Graphogame> : lien *GitHub* vers la version *Ubuntu* du jeu de flexion.
- <https://www.sfml-dev.org/> : site de la *SFML*.
- <https://developer.android.com/studio> : site *Android Studio*.
- <https://www3.ntu.edu.sg/home/ehchua/programming/java/JavaNativeInterface.html> : introduction à *JNI*.
- <https://docs.oracle.com/javase/7/docs/technotes/guides/jni/spec/functions.html> : documentation *JNI*.
- <https://www.youtube.com/watch?v=SktXhGElf7w> : lien vers une démonstration 3D de la *SFML* avec *OpenGL*.