Master Systèmes Dynamiques et Signaux

Mémoire de master

Contribution à la prise en charge de patients atteints de la maladie de Huntington : approche basée sur des outils numériques innovants

---

Auteur :
M. Jean-Baptiste Martineau

Jury :
Pr. L. Autrique
Pr. B. Castanier
Pr. A. Heurtier
Dr. L. Perez
Dr. P. Richard

Président :
Pr. L. Hardouin

Version du
1er juillet 2021

Remerciements

Je souhaite remercier les professeurs Anne Heurtier et Paul Richard pour leur aide, leur suivi, leur écoute et leur bienveillance lors de mon stage et durant toute l'année.

Je remercie aussi toutes les autres personnes membres du projet CoMoN avec qui j'ai pu collaborer tout au long de mon stage, à l'occasion de réunions, échanges et réflexions, et plus spécialement Coline Chartier et Julien Godard pour leur travail et leur collaboration.

Je remercie aussi mes parents qui ont fait tout leur possible pour me rendre le télétravail aussi aisé que possible pendant cette période si particulière de crise sanitaire.

Table des matières

Table des figures

List of acronyms

CoMoN	*Cognition, Motricité, Numérique*
SAM	*Santé, Activité physique, Métiers de l'Ingénierie et des Sciences humaines / Sarthe Anjou Mayenne*
MH	*Maladie de Huntington*
ND	*Neurodégénérative*

Introduction

L'objectif de mon stage était de développer un outil innovant pour la prise en charge des symptômes de patients atteints de la maladie de Huntington.

La maladie de Huntington est une maladie héréditaire incurable qui est associée à la dégénérescence des neurones d'une partie du cerveau, partie impliquée dans la gestion des fonctions motrices, cognitives et comportementales. En l'absence de traitement médical efficace, l'exercice physique et la stimulation cognitive restent les meilleurs axes pour la prise en charge des symptômes associés à la maladie de Huntington.

Le projet CoMoN a pour objectif d'étudier la faisabilité et les effets de l'utilisation conjointe de stimulation physique et cognitive sur l'évolution des symptômes de patients atteints de la maladie de Huntington.
L'outil utilisé pour réaliser la stimulation physique et cognitive des patients est une application créée spécialement pour le projet, afin de prendre en compte tous les besoins des patients.

Ce rapport présente et rend compte des objectifs et contraintes respectés pendant le développement de l'application. Il comporte aussi une présentation technique de l'application dans laquelle les choix réalisés au cours du développement sont explicités.
Enfin, la dernière partie traite de la nécessité de créer un outil nouveau et l'impossibilité de s'appuyer sur des outils déjà existants ; les évolutions prévus à court et moyen terme y sont aussi abordées.

Chapitre 1

Objectifs à atteindre et exercices associés

1.1 Introduction

Dans le cadre de mon stage passé à travailler sur le projet CoMoN, j'ai dû développer une application répondant à divers besoins et objectifs fixés avec les autres membres du projet.

Cette application a pour but d'être un outil permettant d'aider la prise en charge des symptômes de patients atteints de la maladie de Huntington. A l'heure actuelle, aucun traitement ne permet de lutter efficacement contre la maladie et/ou de stopper sa progression. Dans ce projet, comme évoqué précédemment, l'objectif est d'utiliser l'exercice physique et la stimulation cognitive, indépendamment et de manière conjointe, pour ralentir la progression de ces symptômes.

L'application développée se devait d'apporter des nouveautés et des améliorations par rapport aux applications déjà existantes, créées dans le cadre de la recherche menées avec des patients atteints de maladies neurodégénératives.

1.2 Objectifs, contraintes et besoins

1.2.1 Objectifs

Avant le début du stage, des objectifs, des besoins ainsi que des contraintes avaient déjà pu être fixés.

Ces objectifs principaux étaient :

- la prise en charge des différents exercices de stimulation cognitive et physique
- la sélection du niveau de la difficulté de ces exercices
et la personnalisation des séances avec un choix des exercices.

De nouveaux objectifs sont aussi vite apparus au début de mon stage. En effet Il a fallu opérer des choix sur le support à privilégier pour développer la première version de l'application, choisir les exercices à ajouter et déterminer une séance type.

1.2.2 Contraintes

Les contraintes liées à l'application ont aussi été très importantes au cours de son développement. La majorité de ces contraintes était liée à la condition générale (physique, cognitive, comportementale) des patients amenés à utiliser l'outil.

Parmi ces contraintes, on peut notamment citer l'ergonomie : l'application doit être facilement utilisable par des utilisateurs peu familiers avec l'informatique et souffrant en plus de troubles cognitifs. Les patients atteints de la maladie de Huntington peuvent être âgés de moins de 30 ans jusqu'à plus de 80 ans, ce qui ne simplifie pas la tâche.
Il faut aussi que les patients qui utiliseront l'application dans le cadre du programme d'étude soient assidus : pour cela, il faut créer une forme de motivation, comme rappelé dans la littérature scientifique. [1] [3]
Enfin, il faut que les exercices proposés soient suffisamment nombreux pour offrir une assez grande variété et avec différents niveaux de difficulté pour éviter un apprentissage dû à la répétition. Éviter l'apprentissage des exercices par les patients permet d'obtenir des résultats plus réalistes dans le cadre d'une étude, comme dans le projet CoMoN.

1.2.3 Besoins

Les besoins, dans ce projet, ont principalement été définis par les autres membres du projet CoMoN.
Ces besoins regroupent :
- la récupération d'informations sur la qualité de la réalisation des exercices cognitifs
- l'intégration d'un outil permettant d'utiliser la reconnaissance vocale
et un choix protégé de la difficulté des exercices cognitifs.
Il était important de respecter ces besoins pour le bon fonctionnement du projet.

1.3 Exercices et séance type

Pour ce projet, un nombre d'exercices physiques, cognitifs et mixtes ont du être choisis pour la première version de l'application. Ces exercices ont été choisis conjointement avec les deux autres étudiants présents sur le projet : les exercices cognitifs avec Coline et les exercices physiques avec Julien.

1.3.1 Exercices physiques

Pour les exercices physiques, les critères de sélection ont été déterminés pour tester plusieurs aspects de l'application. Nous avons donc choisi 5 types d'exercices physiques avec 3 niveaux de difficultés pour chacun.

Ces exercices sont :
- des abductions de hanche / kicks latéraux (30° niveau facile, 50° niveau moyen et kick latéral niveau difficile)
- du step, avec des pas de plus en plus compliqués
- du squat (lever de chaise, quart de squat et demi squat)
- des pompes (pompes contre le mur, pompes sur les genoux et position de base)

et de la marche sur place à différents rythmes (90 bpm, 110 bpm et une alternance marche/sprint).
Les exercices, choisis avec les spécialistes de l'exercice physique, prévus pour être utilisés dans le projet n'ont pas encore tous été ajoutés à cette version de l'application.

1.3.2 Exercices cognitifs

Les exercices cognitifs prévus pour être ajoutés à l'application ont été choisis par les professionnels de santé. Une partie de ces exercices ont été implémentés, principalement ceux permettant de stimuler la mémoire de travail. Ces exercices présentent, comme les exercices physiques, différents niveaux de difficultés.

Ces exercices sont :
- exercice de mots miroir : mots épelé à l'envers (exemple - "lit" épelé "t, i, l") ; la difficulté vient de la longueur des mots épelés
- exercice d'addition de chiffres : présentation d'une suite de chiffres à additionner ; la difficulté vient du nombre de chiffres à additionner
- exercice d'ordre alphabétique : ranger les mots présentés dans l'ordre alphabétique ; la difficulté vient du nombre de mots à classer
- exercice de type go/no go : réaction à un signal visuel (ici changement de couleur de l'élément affiché à l'écran) ; la difficulté peut être modulée par la vitesse d'apparition du

signal
- exercice de mots détachés : mots découpés en syllabes épelées à l'envers (exemple - "policier" épelé "cier, li, po") ; la difficulté vient ici du nombre de syllabes du mot
et un exercice d'énonciation de lettres : un mot est épelé dans le désordre, et la catégorie à laquelle il appartient est donné au début de l'exercice (exemple - catégorie "animal", lettres épelées "lpuo", mot "loup") ; la taille des mots permet de moduler la difficulté.

Les exercices ajoutés à l'application ont été choisis car les outils nécessaire à leur implémentation étaient disponibles gratuitement et utilisables sans pré-requis supplémentaires.

1.3.3 Exercices de stimulation mixte

Enfin, les exercices mêlant activité physique et stimulation cognitive n'ont pas encore été ajoutés pleinement à l'application. Ces exercices nécessitent un moyen de gérer les interactions du patients autrement qu'avec un système d'interaction classique de type clavier/souris. Les pistes pour ce système d'interaction se sont orientées vers l'utilisation de caméra 3D. Le capteur retenu est une caméra Orbecc Astra Pro (150€).

1.3.4 Séance type

Une séance type est organisées en 5 temps distincts ; 3 de ces 5 éléments constitutifs de la séance peuvent aussi être amenées à se mélanger. Une séance est prévue pour durer environ 1h.

Le premier élément de la séance est un échauffement. Il dure 10 min et fait appel à des exercices physiques et cognitifs.
Le deuxième élément est un groupement d'exercices physiques. Ils sont divisés en 3 catégories et permettent de travailler l'équilibre, la marche ou le renforcement musculaire. Chacun de ces groupes d'exercices dure entre 12 min 30 et 13 min 30.
Le troisième élément regroupe les exercices cognitifs, qui ont pour but de stimuler la mémoire de travail, la flexibilité et l'inhibition. Le nombre d'exercices à réaliser par session peut dépendre du temps d'exercice physique souhaité.
Le quatrième temps de la séance concerne les exercices en double stimulation. Chacun de ces exercices est prévu pour durer environ 4 min 30.
Enfin, le dernier bloc de la séance permet de s'étirer et de se relaxer pour permettre au corps de récupérer dans les meilleures conditions. Cette fin de séance est constituée d'étirements passifs pendant 5 min, suivis par 5 min d'exercices de respiration.

Toutes les durées d'exercices présentés ci-dessus intègrent temps d'activité et temps d'exercice.

Le nombre d'utilisations hebdomadaires prévu pendant la durée de l'étude a pour le moment été estimé à 3.

Chapitre 2

Présentation technique de l'application

2.1 Introduction

L'application, comme évoqué dans le rapport bibliographique intermédiaire, a été réalisée avec Unity3D. Les scripts sont donc écrits en C# et les données obtenues pendant la réalisation des exercices sont stockées dans un fichier au format `CSV`.

2.2 Organisation de l'environnement

Le premier aspect ajouté à l'application, est l'environnement. Il a donc fallu déterminer à quoi il devait ressembler. Il a ainsi été défini que plusieurs environnements seraient nécessaires à l'application. Le premier de ceux-ci est un environnement recréant la pièce de vie d'une maison au milieu des collines. Une salle de sport est aussi prévue, ainsi qu'un environnement un peu plus interactif et fantastique avec un petit scénario évoluant au cours de la séance.

2.2.1 Environnement ajouté : Intérieur de maison

A ce jour, un seul environnement a été ajouté à la version actuelle de l'application, la pièce de vie de maison. Pour cet environnement, il a fallu chercher et trouver des packs de modèles 3D gratuits de maison et de mobilier, et les placer dans la scène. Un terrain avec des collines a aussi été créé pour ajouter un environnement extérieur. Une `skybox` assez simple complète le tout pour une meilleure immersion.

FIGURE 2.1 – Intérieur de maison

2.2.2 Environnements prévus

Comme dit précédemment, plusieurs environnements pourront être ajoutés : un environnement de salle de sport, un environnement fantastique, ou tout autre environnement qui peut être bénéfique à l'immersion des utilisateurs. Ces environnements seront organisés selon les même principes que ceux appliqués à l'intérieur de maison. Cependant, l'environnement avec un thème plus fantastique devra présenter, par rapport aux autres, plusieurs points de vue où le personnage figurant le patient pourra être placé, afin de permettre une évolution au cours de la séance.

2.3 Personnages et animations

L'objectif des environnements, présentés dans la section précédente, est d'accueillir un personnage qui permettra de diriger la séance. Ce personnage, pour pouvoir guider l'utilisateur, doit être capable d'effectuer différents mouvements pour présenter les exercices physiques à réaliser. Les différents personnages utilisés dans le projet ainsi que les animations associées à ces personnages proviennent de la librairie gratuite de Mixamo.

2.3.1 Personnages

Il faudra que le personnage présent dans la scène puisse être choisi par l'utilisateur, dans un menu d'options ou au début de la séance. La diversité des personnages doit donc être suffisante pour permettre à un maximum d'utilisateurs de s'identifier à au moins un de ces personnages. Cela permet une meilleure immersion et ainsi une meilleure expérience de l'utilisation de l'outil et peut-être une meilleure implication du patient.

Le `GameObject` correspondant au personnage actif dans la scène est censé pouvoir être déplacé dans l'environnement souhaité à une position souhaitée, tout en conservant ses propriétés à tout moment.
Pour obtenir ce comportement, le `GameObject` personnage contient tous les éléments utiles pour le bon fonctionnement d'une séance. Ces éléments sont les différentes interfaces et GameObjects utiles aux exercices physiques, cognitifs et conjoints.

FIGURE 2.2 – Organisation du GameObject personnage

Le personnage embarque aussi tous les scripts nécessaires à la gestion de la séance, ainsi qu'un `Animator` pour utiliser les animations.

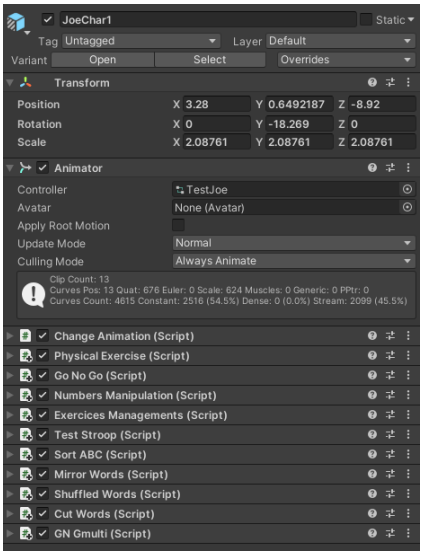

FIGURE 2.3 – Organisation des scripts du personnage

2.3.2 Animations

Le personnage, en plus de son modèle 3D, a aussi besoin d'un jeu d'animations variées correspondant aux exercices physiques présentés. Ces animations, comme évoqué plus tôt, proviennent de la librairies Mixamo, ce qui implique que les spécificités de certains mouvements ne sont pas disponibles. Pour palier ce manque, une autre alternative a été utilisée : ce sont des clips vidéos présentant les exercices à réaliser.
Les animations sont organisées avec un `Animator` qui est un composant Unity qui permet de gérer les transitions entre ces différentes animations. L'`Animator` utilisé pour les animations du personnage, dans cette application, permet de passer d'une transition à l'autre de manière fluide.

Les animations utilisées dans les exercices actuellement présents sont :
- L'animation `Idle` en position d'attente du personnage
- L'animation `Walking` pour les exercices de marche
- La série d'animations `Push up` pour l'exercice de pompes au sol
- L'animation `Squating` pour l'exercice de squat

D'autres animations sont aussi présentes, même si elle ne sont pas utilisées dans la version actuelle de l'application.

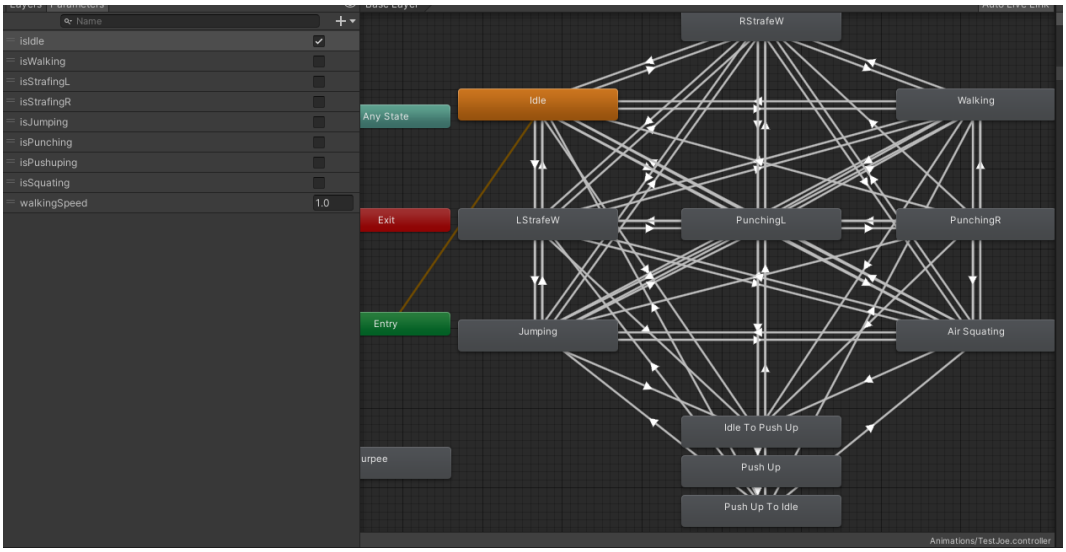

FIGURE 2.4 – Animator du personnage

Les différentes transitions entre les animations sont gérées par plusieurs variables booléennes accessibles à l'intérieur de scripts. Une valeur décimale est aussi présente pour régler la vitesse de l'animation de marche.

2.4 Interface

Un autre élément important de l'application concerne les interfaces. L'application est destinée à un public avec des membres possiblement peu familiers de l'informatique, avec, en plus, des troubles cognitifs. Il faut donc que les différentes interfaces soient compréhensibles facilement, lisibles et simples à utiliser.

2.4.1 Interface de menu

La première interface à laquelle les utilisateurs sont confrontés est le menu. Ce menu dispose seulement de 3 boutons : un bouton `Commencer`, un bouton `Paramètres` et un bouton `Quitter`. Le bouton `Quitter` permet donc de quitter l'application, et le bouton `Paramètres` permet d'accéder au menu de sélection d'un nouveau personnage ainsi qu'aux autres éléments personnalisables de l'application. Enfin, le bouton `Commencer` permet d'accéder à un second menu utilisable pour sélectionner le niveau de difficulté physique de la séance du jour. Ce menu de sélection de la difficulté ne contient que 3 boutons de difficulté : `Facile`, `Normal`, `Difficile`, ainsi qu'un bouton de retour au menu précédent et un bouton pour commencer la séance. Cette interface est prévue pour rester la plus simple d'utilisation et la plus épurée possible. Les seuls éléments qui seront rajoutés seront des éléments apportant un confort d'utilisation global à l'application.

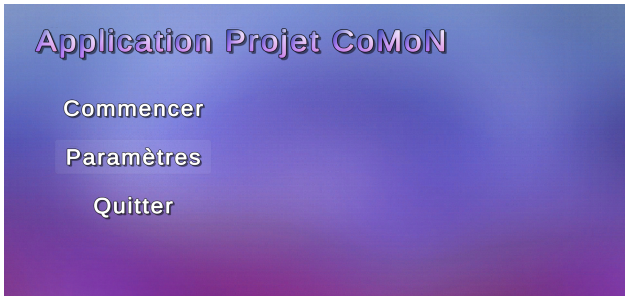

FIGURE 2.5 – Interface de Menu

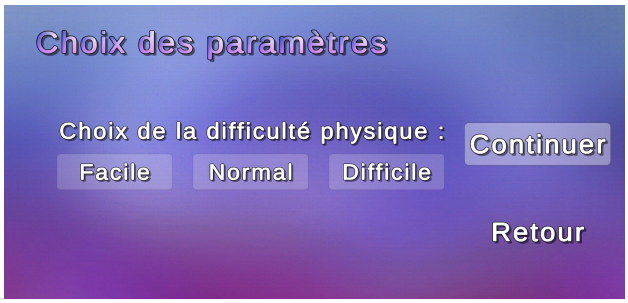

FIGURE 2.6 – Interface de Menu (Commencer)

2.4.2 Interface d'exercice

La deuxième interface importante est celle utilisée dans la scène d'exercice. Cette interface ne contient qu'un seul bouton et il permet de revenir au menu présenté dans le paragraphe ci-dessus.
Les éléments de cette interface sont principalement des zones de texte ou des zones pour afficher des images. Ces zones de texte sont utilisées pour afficher les instructions et les informations au cours des différents exercices.
Les zones d'images sont, elles, utilisées un peu différemment pendant la séance. La première zone est simplement utilisée pour l'exercice de stimulation cognitive "Go/No Go" (son utilisation sera décrite dans la partie script Go/No Go). Et la seconde est une image de type `RawImage` pour permettre d'afficher des vidéos d'instruction de certains des exercices physiques. La lecture des vidéos est possible grâce à un `VideoPlayer` qui est un composant qui permet d'effectuer le rendu d'un clip vidéo sur une texture, qui est ensuite applicable sur un matériau ou un objet de type `RawImage`.

FIGURE 2.7 – Interface d'exercice

2.4.3 Interface de développement

Enfin, une troisième interface est aussi présente, elle n'est cependant disponible que dans la version de développement de l'application. Cette interface permet juste d'afficher plusieurs boutons pour tester les différents exercices individuellement. Cette interface peut être affichée ou désactivée en appuyant sur le bouton `Échap`. Il est aussi possible de naviguer entre les différentes pages de boutons d'exercices disponibles avec les flèches directionnelles.

Figure 2.8 – Interface de développement

2.5 Scripts

Tous les éléments de l'application présentés dans les paragraphes précédents sont gérés et utilisés par des scripts. Ces scripts sont presque tous rattachés au personnage actif dans la scène. Les seuls qui ne le sont pas, sont ceux permettant de gérer l'interface de menu et l'interface de développement. D'autres scripts ne sont pas non plus rattachés au personnages mais ils ne sont présents sur aucun objet de la scène. Ils ne sont utilisés que dans d'autres scripts afin de gérer le stockage et l'écriture de données.
Conserver tous les scripts au même endroit permet de simplifier leur supervision et leur gestion par un autre script. Ce format permet aussi d'éviter les problèmes de gestion en cas de multiples instances d'un même personnage dans la scène.

2.5.1 Exercices physiques

PhysicalExercice

Les exercices physiques sont tous gérés par un seul script. Ce script s'appelle `PhysicalExercise` et permet de lancer toutes les séquences d'exercice physique, que les instructions soient sous forme d'animation du personnage ou de vidéo de consigne.

Le fonctionnement de ce script est relativement simple, chaque évènement possible est programmé à l'aide d'une coroutine, une fonction plus simple à interrompre en cas de

besoin et plus pratique à utiliser. Ensuite, chaque groupe d'évènements correspondant à un exercice est ajouté sous la forme d'une liste de tableaux d'entiers de deux cases. Chaque tableau d'entiers correspond à un évènement différent, le premier entier correspondant au numéro de l'évènement et le second à un paramètre propre à chaque exercice. Ce deuxième entier peut, pour l'instant, correspondre à une durée ou au numéro du `VideoClip` à jouer. Cette séquence est créée au début de l'exécution de l'exercice et un pointeur est initialisé sur 0 pour suivre l'évènement de la séquence en cours d'exécution. Au début d'un exercice, une fonction `Sequence` permet de lancer la coroutine correspondant à l'évènement attendu. Au début de chaque coroutine le pointeur est incrémenté puis le contenu de la coroutine est effectué, pendant la durée spécifiée dans la séquence. A la fin de chaque coroutine, la fonction `Sequence` est appelée et lance la coroutine suivante. Ce script contient aussi deux fonctions appelées au début pour `StartSequence` et à la fin de chaque séquence pour `StopSequence`. Ces deux fonctions permettent de remettre toutes les variables utilisées pour observer l'état du système à leurs valeurs initiales et de stopper proprement toute coroutine potentiellement en cours. Ce système de script facilite l'ajout rapide de nouveaux exercices ou la modification de ceux déjà existants.

Ce script peut interagir avec différents éléments de l'application, le `VideoPlayer` présenté plus tôt, un script appelé `ChangeAnimation` et le script `ExercisesManagement`.

ChangeAnimation

Le script `ChangeAnimation` est un script servant d'interface entre l'`Animator` et les autres scripts de la scène. Le script permet de s'assurer qu'une seule des valeurs booléennes utilisées pour contrôler les animations est vraie à tout moment de l'exécution de l'application. La fonction contient le même nombre de variables booléennes que l'`Animator` et grâce à des conditions permet de vérifier si des valeurs sont vraies. Si oui, la variable correspondante de l'`Animator` est activée, si non elle est désactivée ; si aucune n'est vraie, l'animation `Idle` d'attente est jouée.

2.5.2 Exercices cognitifs

Pour les exercices cognitifs, chacun possède son propre script. En effet, ces exercices possèdent beaucoup plus de spécificités et ont été plus complexes à créer. Cependant, la majorité des scripts de ces exercices est construite selon le même modèle. Les scripts des exercices utilisant les systèmes de synthèse et de reconnaissance vocale sont construits de manière à utiliser la synthèse vocale pendant que la reconnaissance vocale est active.

Base des scripts MirrorWords - CutWords - SortABC - ShuffledWords - NumberManipulation

Ces scripts correspondent tous à un exercice cognitif, leurs noms n'étant pas forcément très explicites, voici les correspondances script/exercice.
- Mots Miroirs / `MirrorWords`
- Mots détachés / `CutWords`
- Ordre alphabétique / `SortABC`
- Énonciation de lettres / `ShuffledWords`
- Addition de chiffres / `NumberManipulation`

Chaque script possède donc un nombre de variables adaptées à leur objectif. Ces variables peuvent prendre la forme de liste de mots, de bornes pour générer un nombre aléatoire, ...
Ils possèdent aussi toutes les variables et autres références nécessaires pour mettre en place les systèmes vocaux.

Lorsque un de ces exercices commence, la fonction `StartEx` est appelée et initialise toutes les variables utilisées et prépare les éléments qui seront épelés puis reconnus. Cette fonction lance une coroutine qui s'occupe alors d'épeler les mots ou nombres à manipuler pour l'exercice. La synthèse vocale est permise grâce aux fonctionnalités fournies par le système d'exploitation Microsoft Windows.
Une fois terminée, cette coroutine en appelle une seconde pour la reconnaissance vocale. Cette coroutine utilise un objet Unity de type `PhraseRecognizer` pour mettre en place la reconnaissance vocale. Le fonctionnement de cet outil est un peu différent de ceux que l'on peut rencontrer dans la vie quotidienne. En effet, contrairement aux assistants vocaux, ce système attend un mot ou une liste de mots spécifiques et est incapable d'enregistrer une phrase entière afin d'ensuite la "traiter". Cette limitation bloque notamment l'ajout de certains exercices cognitifs.
Pendant cette coroutine de reconnaissance vocale, un petit système détermine le temps mis par l'utilisateur pour résoudre l'exercice. Une fois ce dernier résolu, le temps enregistré est écrit dans un document au format `CSV`. Après l'enregistrement de ces données, la fonction `StopEx` est appelée pour arrêter l'écoute et la synthèse vocale et remettre à leurs valeurs de base les différentes variables utilisées.
Les seules différences entre tous ces scripts se trouvent au niveau des fonctions de mise en forme des différentes données d'exercice.

MirrorWords

Ce script possède différentes listes de mots correspondant chacune à une difficulté. Au début de la séance, les mots sont mélangés pour pouvoir en proposer des différents à chaque séance.

FIGURE 2.9 – Mots miroir : mot reconnu

CutWords

Ce script contient une liste de mots découpés en syllabes qui sont eux aussi mélangés à chaque séance.

SortABC

Pour ce script, une liste de mot est mélangée à chaque lancement de l'exercice et ensuite un certain nombre de mots, nombre déterminé grâce à une variable du script, sont utilisés pour l'exercice.

NumberManipulation

A chaque début de cet exercice, plusieurs nombres sont tirés aléatoirement entre deux bornes (définies par des variables du script) puis stockés dans une `list` d'entiers. Une variable permet aussi de stocker le total de l'addition des nombres tirés.

Go/No Go

Le script permettant de réaliser l'exercice de type "Go/No Go" est un des scripts différents de ceux présentés au-dessus.
Ce script utilise un élément de l'interface liée au personnage, une image. L'objectif de cet exercice est d'interagir avec le système, dans le cas présent à l'aide de la barre d'espace, au moment où l'image change de couleur (bleue au lieu de noire). Au début de l'exercice, la fonction `StartGNG` est appelée pour préparer toutes les variables nécessaires à l'exercice, afficher l'image, puis lancer la coroutine `GNG`. Cette coroutine s'occupe d'effectuer, à intervalle de temps défini, un tirage pour déterminer si la couleur de l'image doit être changée ou non. Si ce tirage est vrai ou si un certain nombre de tirages a été atteint, la fonction `ChangeColor` est appelée. Cette fonction, comme son nom l'indique, s'occupe de changer la couleur de l'image, ici elle deviendra bleue.
Lorsque l'image change de couleur, un système de chronométrage est activé et permet de mesurer le temps de réaction de l'utilisateur. Cette durée est enregistrée si la personne réussit à interagir à temps avec le système, sinon l'application enregistre que le temps pour réagir a été dépassé et que l'utilisateur a échoué. Une fois le temps disponible écoulé, l'image redevient noire et l'application recommence les tirages.
Cette action est répétée un nombre défini de fois (déterminé par une variable du script). Une fois le nombre de répétitions de l'exercice atteint, la fonction `StopGNG` est appelée pour remettre les variables dans leur état initial et cacher l'image.
En cas d'interaction avec le système en dehors du temps prévu à cet effet, l'application enregistre aussi l'erreur du patient.

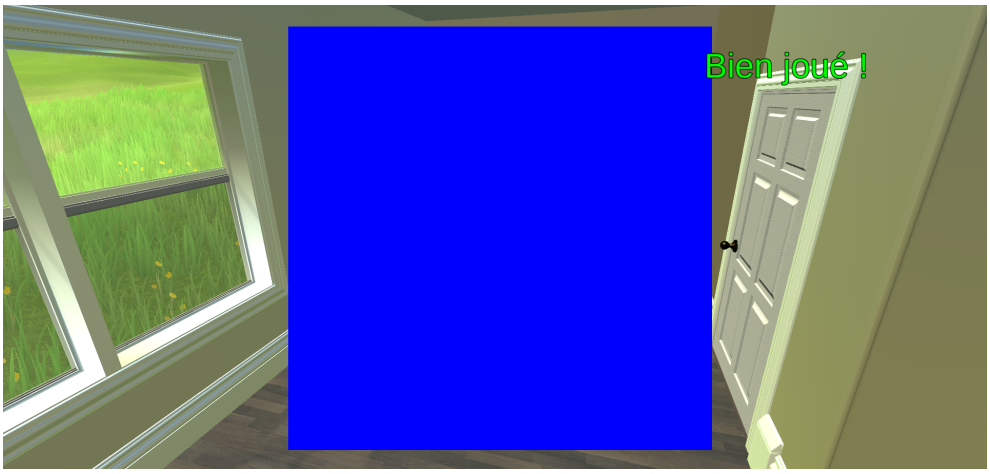

FIGURE 2.10 – Go/No Go : réussite

2.5.3 Gestion des exercices

ExercisesManagement

Tous les exercices sont supervisés par un script appelé `ExercisesManagement`. Cet exercice possède les fonctions utilisées pour lancer un exercice. Passer par un script autre que celui d'origine permet de s'assurer qu'un seul est en cours d'exécution à chaque fois. Cette fonctionnalité est assurée par une variable booléenne activée par les différents scripts d'exercices. Cette variable est vérifiée avant de lancer un exercice.

Ce script contient aussi trois coroutines proposant trois scénarios déterminés de manière arbitraire pour pouvoir tester le fonctionnement de l'application. Les trois coroutines correspondent aux 3 niveaux de difficultés présents dans l'interface du Menu.

2.5.4 Gestion des données

L'application utilise deux systèmes de données distincts, un pour enregistrer les performances du patient pendant l'utilisation de l'application et l'autre pour enregistrer ses paramètres.

CSVWriter

Le script `CSVWriter` permet de créer un fichier au format `CSV` avec pour nom la date du jour de la séance. Le fichier est créé au début de la séance et son existence est vérifiée au début de chaque exercice pour prévenir d'éventuels problèmes.
La fonction `WriteData` permet d'écrire des données dans le fichier dans le format `CSV`. Utiliser ce format permet de fournir des fichiers directement lisibles par les professionnels de santé grâce à des logiciels comme Microsoft Excel.
Ce script permet aussi de gérer le chemin d'accès du fichier selon l'environnement dans lequel se trouve l'application, ce qui rend le script fonctionnel même sous `iOS` ou `Android`.
Ce script est une classe statique, qui permet d'appeler aisément ses fonctions à l'intérieur de n'importe quel autre script.

DataSaver

Le script `DataSaver` est un script qui permet de stocker des informations mises en formes grâce à une classe sérialisable. Utiliser une classe de ce type permet de la convertir en token `Json` pour ensuite la stocker en tableau d'octet dans un fichier extérieur à l'application. L'avantage de cette méthode est de pouvoir stocker un type varié de données utiles à l'application. Il suffit de créer une classe sérialisable regroupant les informations à enregistrer.

Seance du 16/06	DifficulteP 0	DifficulteC 1
GoNoGo	Temps Reaction	Reussite
2	temps ecoule	2
3	temps ecoule	2
4	temps ecoule	2
5	temps ecoule	2
5	rate	0
5	0,3161229	1
6	rate	0

FIGURE 2.11 – Exemple de fichier CSV produit

Tous les scripts présentés ont été prévus pour être facilement compréhensibles, modifiables ou dérivables. Cette application est amenée à évoluer et à être modifiée par d'autres personnes, il était donc nécessaire que le temps demandé pour comprendre l'organisation de ses différents composants soit le plus réduit possible.

Chapitre 3

Comparaison et évolutions de l'application

3.1 Comparaison

Cette application a été créée à partir d'un projet Unity vierge. Tout a été créé pour répondre aux besoins du projet.
Comme présenté dans le rapport bibliographique intermédiaire, il n'existait pas encore de solution technique proposant toutes les fonctionnalités requises pour réaliser l'objectif du projet CoMoN.

La seule solution technique spécialement développée pour des patients atteints de la maladie de Huntington (MH), dans le cadre d'exercice physique, est un DVD [2]. Cet outil n'est pas adapté au projet, cependant certains aspects en ont été conservés dans la version actuelle de l'application. Les consignes pour les exercices physiques sont en partie présentées grâce à des vidéos similaires à celles que l'on peut retrouver dans ce genre de support.

Les autres solutions utilisée dans le cas d'autres maladies neurodégénératives (ND) ne proposaient pas non plus toutes les fonctionnalités demandées pour le projet. La solution pour la stimulation cognitive utilisant deux approches complémentaires [3] met en avant l'intérêt de la stimulation cognitive informatisée, mais montre aussi l'intérêt de l'apprentissage de l'outil comme moyen de stimulation. L'application possède des menus simples et intuitifs, l'objectif est que le patient puisse l'utiliser en autonomie et s'approprie l'environnement au fur et à mesure des séances. Cette indépendance du patient peut participer à la motivation chez les utilisateurs.

L'autre solution présentée dans le rapport intermédiaire, pour les maladies neurodégénératives autre que la MH, évoque l'utilisation d'un "serious exergame" [1]. Un exergame est

un jeu vidéo d'entraînement physique. Cette solution est beaucoup plus intéressante pour le projet CoMoN, car un exergame, par sa nature de jeu vidéo, peut être créée de manière à offrir une expérience cumulant stimulation physique et cognitive. Il a été démontré que cette solution créait un engagement plutôt fort chez les patients ayant participé aux tests. Cependant, il est important de noter que les patients atteints de maladies ND ont joué moins longtemps et obtenu des performances inférieures aux patients en bonne santé. Les patients atteints de ND ont aussi présentés des niveaux d'activité physique inférieurs à ceux de patients sains.

L'application développée s'apparente à un exergame et a été développée en s'inspirant de ce qui existe dans d'autres applications de ce type. Le format classique d'un exergame n'était cependant pas entièrement adaptable au projet, l'application étant destinée à des patients atteints de troubles cognitifs, il ne fallait pas que l'environnement présente trop d'éléments parasites dans les décors ou soit cognitivement fatiguant. Il fallait aussi que le type d'exercice demandé soit adapté et ne soit pas trop intense.
Pour la stimulation physique la plus intense, que ce soit pour les exercices de pure activité physique ou pour les exercices conjoints, les mouvements du patients ne sont pas utilisés pour interagir directement avec la simulation, contrairement à un exergame classique. Ce choix évite de suraugmenter la charge cognitive du patient mais permet d'allonger la durée de la séance.

3.2 Retours utilisateurs

Sur les dernières semaines de stage, des tests avec différentes personnes ont pu être réalisés. Huit personnes ont participé aux tests, sept étudiants de 18 à 24 ans et une femme de 68 ans atteinte de la maladie de Huntington, présentant des symptômes légers.

Les retours obtenus des étudiants concernent principalement l'aspect esthétique de l'application. Le retour le plus récurrent concerne la synthèse vocale : la voix utilisée n'est pas très compréhensible pour certains sons. Ce défaut peut être corrigé en utilisant un autre système de synthèse vocal.
Plusieurs personnes ont signalé que certaines animations leur semblaient trop rigides ou que les transitions manquaient de fluidité. Pour améliorer ce point, il faudrait avoir accès à un plus grand nombre d'animations et retravailler les transitions.
Des retours ont aussi été faits sur l'aspect de l'interface, les couleurs du menu, la forme et le relief des boutons, ainsi que la lisibilité de certains textes. Ces points nécessitent juste d'être retravaillés pour la prochaine version de l'application.
Quelques autres points ont été évoqués : l'idée de pouvoir changer l'image de l'exercice Go/no Go, modifier le moment de la journée ou changer l'extérieur de l'environnement de l'espace de vie de la maison.
Il est agréable de noter que tous ont apprécié la facilité d'utilisation.

En ce qui concerne les échanges avec la patiente, nous avons eu des retours plus nombreux et plus développés que les précédents. Certains, comme les problèmes liés à la voix, son ton ou son volume, ont été évoqués. La patiente à aussi permis de soulever de nouvelles interrogations et points à discuter avec les autres membres du projet CoMoN.
Tout d'abord, en ce qui concerne la "barrière de l'écran", la question a été posée de savoir si l'écran était forcément adapté à tous les utilisateurs potentiels de l'application.
Un autre point abordé concerne le rythme des exercices. En effet, la patiente testée a encore une activité physique relativement intense et craint que les exercices physiques proposés par l'application ne fassent doublon avec l'activité physique quotidienne. Une question a aussi été posée sur le manière de choisir les exercices physiques à réaliser pendant la séance : si certains exercices proposés par l'application reproduisent des mouvements de l'activité quotidienne, il serait souhaitable de pouvoir les sortir de la séance de la journée. Il reste encore à déterminer si ce genre de fonctionnalité est utile pour le projet, et si oui, quel degré de liberté accorder à l'utilisateur.
Une proposition a été faite, le patient peut-il choisir lui- même ses activités ? Cependant, cette possibilité a déjà été évoquée et écartée lors des réunions de projet, pour éviter que les patients ne réalisent pas toujours les mêmes exercices.
Enfin, le dernier point abordé est la proposition d'une fonctionnalité permettant de mettre l'application en pause en cas de besoin lors de la séance. Cette fonctionnalité peut facilement être implémentée et serait un ajout pertinent à l'application.

3.3 Évolutions

La version actuelle de l'application n'est qu'une première version et est amenée à évoluer. Puisqu'un certain nombre de points à ajouter, améliorer ou modifier a déjà été relevé.

La première fonctionnalité à enrichir est la liste des exercices disponibles. Ceux présents dans la version actuelle ont été choisis car ils étaient implémentables sans avoir à rajouter d'outils payants supplémentaires dans l'application.
Pour pouvoir ajouter la majorité des exercices manquants, il faudrait pouvoir utiliser un capteur 3D pour suivre les mouvements de l'utilisateur et mettre en place une interaction avec le système.

Il faudra aussi penser à ajouter de la musique pour les exercices physiques, il existe déjà un fichier avec les morceaux utilisables.

Il faut aussi ajouter les différents environnements prévus, définir le petit scénario pour l'environnement "fantastique" et en imaginer d'autres. Il reste aussi à sélectionner des personnages adaptés à ces différents environnements.

Certains scripts d'exercices cognitifs auront besoin d'être modifiés et devront proposer plusieurs niveaux de difficulté.

Il serait bon de créer un système pour gérer la difficulté des exercices cognitifs, difficulté qui devra être adaptée dynamiquement en fonction des exercices.

L'amélioration du menu de l'interface `Paramètres` devra être envisagée pour ajouter une option de sélection des scènes et des personnages à ce menu.

Enfin, il ne faudra pas oublier de tenir compte des retours d'expérience des testeurs qui permettront de faire évoluer l'application au fil du temps et des avancées de la recherche médicale.

Conclusion et perspectives

Dans le cadre du projet CoMoN, j'ai eu à mener des recherches qui ont conduit à l'élaboration d'une application souhaitée innovante. Celle-ci est destinée à aider les patients atteints de la maladie de Huntington, à améliorer leur quotidien en participant à la prise en charge de leurs symptômes, afin de retarder la survenue des déficiences physiques et cognitives. Après un inventaire attentif des solutions techniques utilisées pour des pathologies proches, l'application a pu être créée en tenant compte des besoins et des contraintes liée aux besoins spécifiques de chaque patient.

Tout au long de cette période de développement, j'ai pu échanger et discuter de mes avancés avec les autres membres de l'équipe. Cela m'a permis d'appréhender le déroulement d'un tel projet, les recherches à mener pour définir celui-ci, ainsi que les interactions à mettre en place entre les membres pour un fonctionnement optimal.

Le développement de cette application m'a appris à écouter pour comprendre les attentes des différents acteurs, à synthétiser les solutions existantes, puis à élaborer les solutions techniques associées. J'ai aussi appréhendé la gestion de l'organisation à mettre ne place pour respecter le calendrier établi. La perspective de la continuation du projet par d'autres, m'a poussé à organiser et structurer la conception logicielle afin d'en faciliter la prise en main.

Suite à cette phase de développement j'ai aussi pu me familiariser avec le passage de tests en organisant des essais de l'application avec l'aide d'une patiente volontaire et d'étudiants.

Tout le travail mené m'a permis de m'ouvrir au monde de la recherche scientifique et d'en appréhender certains codes.

J'espère que ce travail initial contribuera, avec les apports futurs, à améliorer la qualité de vie des patients atteints de la maladie de Huntington.

Annexes

PhysicalExercise

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using TMPro;
5  using UnityEngine.Video;
6
7  public class PhysicalExercise : MonoBehaviour
8  {
9      [SerializeField] private ChangeAnimation anim;
10     [SerializeField] private bool isRunning;
11     [Range(0, 10)] public int delay = 2;
12
13     List<int[]> seq;
14     private int point;
15
16     public TextMeshProUGUI instructions;
17
18     public VideoPlayer videoPlayer;
19
20     private List<VideoClip> videoLinks;
21     public VideoClip pompeGenou;
22     public VideoClip stepEasy;
23     public VideoClip stepMedium;
24     public VideoClip stepHard;
25     public VideoClip hancheEasy;
26     public VideoClip hancheMedium;
27     public VideoClip kickHard;
28     public VideoClip squatEasy;
29
30     public ExercicesManagements exercicesManagements;
31
32     void Start()
33     {
34         videoLinks = new List<VideoClip>
35         {
```

```
36                  pompeGenou,
37                  stepEasy,
38                  stepMedium,
39                  stepHard,
40                  hancheEasy,
41                  hancheMedium,
42                  kickHard,
43                  squatEasy
44              };
45              anim = gameObject.GetComponent<ChangeAnimation>();
46              videoPlayer = GameObject.Find("VideoSource").GetComponent<
        VideoPlayer>();
47              ClearOutRenderTexture(videoPlayer.targetTexture);
48              isRunning = false;
49          }
50
51          IEnumerator GoEx()
52          {
53              Debug.Log("Start Pause");
54              point++;
55              instructions.text = "Temps d'exercice";
56              yield return new WaitForSeconds(seq[point - 1][1]);
57              instructions.text = "";
58              Debug.Log("End Pause");
59              yield return new WaitForSeconds(delay);
60              Sequence();
61              yield return null;
62          }
63
64          IEnumerator RestEx()
65          {
66              Debug.Log("Start Pause");
67              point++;
68              instructions.text = "Temps de repos";
69              yield return new WaitForSeconds(seq[point - 1][1]);
70              anim.Punching(false);
71              instructions.text = "";
72              Debug.Log("End Pause");
73              yield return new WaitForSeconds(delay);
74              Sequence();
75              yield return null;
76          }
77
78          IEnumerator PlayVideo()
79          {
80              Debug.Log("Start Video");
81              point++;
82              videoPlayer.clip = videoLinks[seq[point - 1][1]];
83              videoPlayer.Play();
```

```
84          Debug.Log((float)videoPlayer.clip.length);
85          instructions.text = "Instructions";
86          yield return new WaitForSeconds((float)videoPlayer.clip.length);
87          videoPlayer.Stop();
88          ClearOutRenderTexture(videoPlayer.targetTexture);
89          instructions.text = "";
90          Debug.Log("End Video");
91          yield return new WaitForSeconds(delay);
92          Sequence();
93          yield return null;
94      }
95
96      IEnumerator Walking()
97      {
98          Debug.Log("Start Walking");
99          point++;
100         anim.Walking(true);
101         instructions.text = "Instructions";
102         yield return new WaitForSeconds(seq[point - 1][1]);
103         anim.Walking(false);
104         instructions.text = "";
105         Debug.Log("End Walking");
106         yield return new WaitForSeconds(delay);
107         Sequence();
108         yield return null;
109     }
110
111     IEnumerator StrafeR()
112     {
113         Debug.Log("Start StrafeR");
114         point++;
115         anim.StrafingR(true);
116         instructions.text = "Instructions";
117         yield return new WaitForSeconds(seq[point - 1][1]);
118         anim.StrafingR(false);
119         instructions.text = "";
120         Debug.Log("End StrafeR");
121         yield return new WaitForSeconds(delay);
122         Sequence();
123         yield return null;
124     }
125
126     IEnumerator StrafeL()
127     {
128         Debug.Log("Start StrafeL");
129         point++;
130         anim.StrafingL(true);
131         instructions.text = "Instructions";
132         yield return new WaitForSeconds(seq[point - 1][1]);
```

```
133             anim.StrafingL(false);
134             instructions.text = "";
135             Debug.Log("End StrafeL");
136             yield return new WaitForSeconds(delay);
137             Sequence();
138             yield return null;
139         }
140 
141         IEnumerator Jumping()
142         {
143             Debug.Log("Start Jumping");
144             point++;
145             anim.Jumping(true);
146             instructions.text = "Instructions";
147             yield return new WaitForSeconds(seq[point - 1][1]);
148             anim.Jumping(false);
149             instructions.text = "";
150             Debug.Log("End Jumping");
151             yield return new WaitForSeconds(delay);
152             Sequence();
153             yield return null;
154         }
155 
156         IEnumerator Punching()
157         {
158             Debug.Log("Start Punching");
159             point++;
160             anim.Punching(true);
161             instructions.text = "Instructions";
162             yield return new WaitForSeconds(seq[point - 1][1]);
163             anim.Punching(false);
164             instructions.text = "";
165             Debug.Log("End Punching");
166             yield return new WaitForSeconds(delay);
167             Sequence();
168             yield return null;
169         }
170 
171         IEnumerator Pushuping()
172         {
173             Debug.Log("Start PushUping");
174             point++;
175             anim.PushUping(true);
176             instructions.text = "Instructions";
177             yield return new WaitForSeconds(seq[point - 1][1]);
178             anim.PushUping(false);
179             instructions.text = "";
180             Debug.Log("End PushUping");
181             yield return new WaitForSeconds(delay);
```

```
182         Sequence();
183         yield return null;
184     }
185 
186     IEnumerator Squating()
187     {
188         Debug.Log("Start Squating");
189         point++;
190         anim.Squating(true);
191         instructions.text = "Instructions";
192         yield return new WaitForSeconds(seq[point - 1][1]);
193         anim.Squating(false);
194         instructions.text = "";
195         Debug.Log("End Squating");
196         yield return new WaitForSeconds(delay);
197         Sequence();
198         yield return null;
199     }
200 
201     public void Sequence()
202     {
203         if (point < seq.Count)
204         {
205             switch (seq[point][0])
206             {
207                 case 0:
208                     StartCoroutine("GoEx");
209                     break;
210                 case 1:
211                     StartCoroutine("RestEx");
212                     break;
213                 case 2:
214                     StartCoroutine("PlayVideo");
215                     break;
216                 case 3:
217                     StartCoroutine("Walking");
218                     break;
219                 case 4:
220                     StartCoroutine("StrafeR");
221                     break;
222                 case 5:
223                     StartCoroutine("StrafeL");
224                     break;
225                 case 6:
226                     StartCoroutine("Jumping");
227                     break;
228                 case 7:
229                     StartCoroutine("Punching");
230                     break;
```

```
231                 case 8:
232                     StartCoroutine("Pushuping");
233                     break;
234                 case 9:
235                     StartCoroutine("Squating");
236                     break;
237                 default:
238                     break;
239             }
240         }
241         else { StopSequence(); }
242     }
243 
244     void StartSequence()
245     {
246         if (!anim.Running)
247         {
248             exercicesManagements.running = true;
249             anim.Running = true;
250             point = 0;
251             isRunning = true;
252             Sequence();
253         }
254     }
255 
256     void StopSequence()
257     {
258         exercicesManagements.running = false;
259         StopAllCoroutines();
260         anim.Walking(false);
261         anim.Jumping(false);
262         anim.StrafingL(false);
263         anim.StrafingR(false);
264         anim.Punching(false);
265         anim.PushUping(false);
266         anim.Squating(false);
267         isRunning = false;
268         videoPlayer.Stop();
269         ClearOutRenderTexture(videoPlayer.targetTexture);
270         instructions.text = "";
271         Debug.Log("EndSeq");
272         anim.Running = false;
273         seq.Clear();
274     }
275 
276     public void ButtonClick()
277     {
278         if (!isRunning) { StartSequence(); }
279         else { StopSequence(); }
```

```
280     }
281
282     public void ButtonClickWalkingEasy()
283     {
284         if (!isRunning)
285         {
286             anim.WalkSpeed(0.8f);
287             seq = new List<int[]> {
288                 new int[] { 3, 30 },
289                 new int[] { 0, 30 },
290                 new int[] { 1, 30 },
291                 new int[] { 0, 30 },
292                 new int[] { 1, 30 },
293                 new int[] { 0, 30 },
294                 new int[] { 1, 30 }};
295
296             StartSequence();
297         }
298         else { StopSequence(); }
299     }
300
301     public void ButtonClickWalkingMedium()
302     {
303         if (!isRunning)
304         {
305             anim.WalkSpeed(1.0f);
306             seq = new List<int[]> {
307                 new int[] { 3, 30 },
308                 new int[] { 0, 30 },
309                 new int[] { 1, 30 },
310                 new int[] { 0, 30 },
311                 new int[] { 1, 30 },
312                 new int[] { 0, 30 },
313                 new int[] { 1, 30 },
314                 new int[] { 0, 30 },
315                 new int[] { 1, 30 }};
316             StartSequence();
317         }
318         else { StopSequence(); }
319     }
320
321     public void ButtonClickPushupMedium()
322     {
323         if (!isRunning)
324         {
325             seq = new List<int[]> {
326                 new int[] { 2, 0 },
327                 new int[] { 0, 30 },
328                 new int[] { 1, 30 },
```

```
329                 new int[] { 0, 30 },
330                 new int[] { 1, 30 },
331                 new int[] { 0, 30 },
332                 new int[] { 1, 30 },
333                 new int[] { 0, 30 },
334                 new int[] { 1, 30 }};
335             StartSequence();
336         }
337         else { StopSequence(); }
338     }
339
340     public void ButtonClickPushupHard()
341     {
342         if (!isRunning)
343         {
344             seq = new List<int[]> {
345                 new int[] { 8, 30 },
346                 new int[] { 0, 30 },
347                 new int[] { 1, 30 },
348                 new int[] { 0, 30 },
349                 new int[] { 1, 30 },
350                 new int[] { 0, 30 },
351                 new int[] { 1, 30 },
352                 new int[] { 0, 30 },
353                 new int[] { 1, 30 }};
354             StartSequence();
355         }
356         else { StopSequence(); }
357     }
358
359     public void ButtonClickStepEasy()
360     {
361         if (!isRunning)
362         {
363             seq = new List<int[]> {
364                 new int[] { 2, 1 },
365                 new int[] { 0, 60 },
366                 new int[] { 1, 60 },
367                 new int[] { 0, 60 },
368                 new int[] { 1, 60 }};
369             StartSequence();
370         }
371         else { StopSequence(); }
372     }
373
374     public void ButtonClickStepMedium()
375     {
376         if (!isRunning)
377         {
```

```
378             seq = new List<int[]> {
379                 new int[] { 2, 2 },
380                 new int[] { 0, 60 },
381                 new int[] { 1, 60 },
382                 new int[] { 0, 60 },
383                 new int[] { 1, 60 }};
384             StartSequence();
385         }
386         else { StopSequence(); }
387     }
388
389     public void ButtonClickStepHard()
390     {
391         if (!isRunning)
392         {
393             seq = new List<int[]> {
394                 new int[] { 2, 3 },
395                 new int[] { 0, 60 },
396                 new int[] { 1, 60 },
397                 new int[] { 0, 60 },
398                 new int[] { 1, 60 }};
399             StartSequence();
400         }
401         else { StopSequence(); }
402     }
403
404     public void ButtonClickHancheEasy()
405     {
406         if (!isRunning)
407         {
408             seq = new List<int[]> {
409                 new int[] { 2, 4 },
410                 new int[] { 0, 30 },
411                 new int[] { 1, 30 },
412                 new int[] { 0, 30 },
413                 new int[] { 1, 30 },
414                 new int[] { 0, 30 },
415                 new int[] { 1, 30 }};
416             StartSequence();
417         }
418         else { StopSequence(); }
419     }
420
421     public void ButtonClickHancheMedium()
422     {
423         if (!isRunning)
424         {
425             seq = new List<int[]> {
426                 new int[] { 2, 5 },
```

```
427                 new int[] { 0, 30 },
428                 new int[] { 1, 30 },
429                 new int[] { 0, 30 },
430                 new int[] { 1, 30 },
431                 new int[] { 0, 30 },
432                 new int[] { 1, 30 },
433                 new int[] { 0, 30 },
434                 new int[] { 1, 30 }};
435             StartSequence();
436         }
437         else { StopSequence(); }
438     }
439 
440     public void ButtonClickHancheHard()
441     {
442         if (!isRunning)
443         {
444             seq = new List<int[]> {
445                 new int[] { 2, 6 },
446                 new int[] { 0, 30 },
447                 new int[] { 1, 30 },
448                 new int[] { 0, 30 },
449                 new int[] { 1, 30 },
450                 new int[] { 0, 30 },
451                 new int[] { 1, 30 },
452                 new int[] { 0, 30 },
453                 new int[] { 1, 30 }};
454             StartSequence();
455         }
456         else { StopSequence(); }
457     }
458 
459     public void ButtonClickSquatEasy()
460     {
461         if (!isRunning)
462         {
463             seq = new List<int[]> {
464                 new int[] { 2, 7 },
465                 new int[] { 0, 30 },
466                 new int[] { 1, 30 },
467                 new int[] { 0, 30 },
468                 new int[] { 1, 30 },
469                 new int[] { 0, 30 },
470                 new int[] { 1, 30 },
471                 new int[] { 0, 30 },
472                 new int[] { 1, 30 }};
473             StartSequence();
474         }
475         else { StopSequence(); }
```

```
476     }
477
478     public void ButtonClickSquatingHard()
479     {
480         if (!isRunning)
481         {
482             seq = new List<int[]> {
483                 new int[] { 9, 30 },
484                 new int[] { 0, 30 },
485                 new int[] { 1, 30 },
486                 new int[] { 0, 30 },
487                 new int[] { 1, 30 },
488                 new int[] { 0, 30 },
489                 new int[] { 1, 30 },
490                 new int[] { 0, 30 },
491                 new int[] { 1, 30 }};
492             StartSequence();
493         }
494         else { StopSequence(); }
495     }
496
497     private void ClearOutRenderTexture(RenderTexture renderTexture)
498     {
499         RenderTexture rt = RenderTexture.active;
500         RenderTexture.active = renderTexture;
501         GL.Clear(true, true, Color.clear);
502         RenderTexture.active = rt;
503     }
504 }
```

ChangeAnimation

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class ChangeAnimation : MonoBehaviour
6  {
7      [SerializeField] private Animator anim;
8      [SerializeField] private bool isWalking;
9      [SerializeField] private bool isJumping;
10     [SerializeField] private bool isStrafingR;
11     [SerializeField] private bool isStrafingL;
12     [SerializeField] private bool isPunching;
13     [SerializeField] private bool isPushUping;
14     [SerializeField] private bool isSquating;
```

```
15      [SerializeField] private bool running;
16
17      void Start()
18      {
19          running = false;
20          anim = GetComponent<Animator>();
21      }
22
23      void Update()
24      {
25          if (isWalking) { anim.SetBool("isWalking", true); }
26          else { anim.SetBool("isWalking", false); }
27          if (isJumping) { anim.SetBool("isJumping", true); }
28          else { anim.SetBool("isJumping", false); }
29          if (isStrafingL) { anim.SetBool("isStrafingL", true); }
30          else { anim.SetBool("isStrafingL", false); }
31          if (isStrafingR) { anim.SetBool("isStrafingR", true); }
32          else { anim.SetBool("isStrafingR", false); }
33          if (isPunching) { anim.SetBool("isPunching", true); }
34          else { anim.SetBool("isPunching", false); }
35          if (isPushUping) { anim.SetBool("isPushuping", true); }
36          else { anim.SetBool("isPushuping", false); }
37          if (isSquating) { anim.SetBool("isSquating", true); }
38          else { anim.SetBool("isSquating", false); }
39          if (!isWalking && !isJumping && !isStrafingL && !isStrafingR && !
        isPunching) { anim.SetBool("isIdle", true); }
40          else { anim.SetBool("isIdle", false); }
41      }
42
43      public void Walking(bool isWalking) { this.isWalking = isWalking; }
44      public void Jumping(bool isJumping) { this.isJumping = isJumping; }
45      public void StrafingR(bool isStrafingR) { this.isStrafingR = isStrafingR
        ; }
46      public void StrafingL(bool isStrafingL) { this.isStrafingL = isStrafingL
        ; }
47      public void Punching(bool isPunching) { this.isPunching = isPunching; }
48      public void PushUping(bool isPushUping) { this.isPushUping = isPushUping
        ; }
49      public void Squating(bool isSquating) { this.isSquating = isSquating; }
50      public bool Running { get => running; set => running = value; }
51
52      public void WalkSpeed(float speed)
53      {
54          anim.SetFloat("walkingSpeed", speed);
55      }
56  }
```

ShuffledWords (exemple d'exercice avec reconnaissance vocale)

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using SpeechLib;
using UnityEngine.UI;
using TMPro;
using UnityEngine.Windows.Speech;

public class ShuffledWords : MonoBehaviour
{
    public TextMeshProUGUI txt;
    private SpVoice voice;

    private List<string[]> words = new List<string[]> {
        new string[] { "saison","été" }, //Mot exemple
        new string[] { "animaux","loup" },
        new string[] { "couleur","vert" },
        new string[] { "vêtements","jupe" },
        new string[] { "fruit","kiwi" },
        new string[] { "boisson","café" },
        new string[] { "dessert","tarte" },
        new string[] { "ville","paris" },
        new string[] { "animaux","chien" },
        new string[] { "mois de l année","juin" },
        new string[] { "légume","navet" },
        new string[] { "mobilier","table" },
        new string[] { "jeux","tarot" },
        new string[] { "pays","italie" },
        new string[] { "sport","tennis" },
        new string[] { "couleur","violet" }
    };
    private List<string> shuffledWords = new List<string>();

    [SerializeField] private int listenedWord;

    private PhraseRecognizer recognizer;
    [SerializeField] private ConfidenceLevel confidence = ConfidenceLevel.
    Medium;
    public string word;

    public bool isSpeaking;
    public bool isListening;

    public int nbRounds;
    private float timer;
    private bool timeR;
```

```
46
47      public float delay = 1f;
48
49      public ExercicesManagements exercicesManagements;
50
51      void Start()
52      {
53          word = null;
54          isSpeaking = false;
55          isListening = false;
56          voice = new SpVoice();
57          timer = 0.0f;
58          timeR = false;
59
60          listenedWord = 0;
61          for (int i = 0; i < words.Count; i++)
62          {
63              string[] temp = words[i];
64              int randomIndex = Random.Range(i, words.Count);
65              words[i] = words[randomIndex];
66              words[randomIndex] = temp;
67              Debug.Log(words[i][1]);
68          }
69          for (int i = 0; i < words.Count; i++)
70          {
71              char[] tempC = words[i][1].ToCharArray();
72              for (int j = 0; j < tempC.Length; j++)
73              {
74                  char temp = tempC[j];
75                  int randomIndex = Random.Range(j, tempC.Length);
76                  tempC[j] = tempC[randomIndex];
77                  tempC[randomIndex] = temp;
78              }
79              shuffledWords.Add(new string(tempC));
80          }
81      }
82
83      void Update()
84      {
85          if (Input.GetKeyDown(KeyCode.Space) && (isListening || isSpeaking))
86          {
87              StopEx();
88          }
89          if (timeR)
90          {
91              timer += Time.deltaTime;
92          }
93      }
94
```

```
95      private void OnApplicationQuit()
96      {
97          StopListening();
98      }
99
100     private void Recognizer_OnPhraseRecognized(PhraseRecognizedEventArgs
        args)
101     {
102         word = args.text;
103         txt.text = "Mot reconnu : " + word;
104     }
105
106     public void ButtonClick()
107     {
108         if (!isSpeaking && !isListening) { StartEx(); }
109         else { StopEx(); }
110     }
111
112     private void StartEx()
113     {
114         exercicesManagements.running = true;
115         isSpeaking = true;
116         nbRounds = 3;
117         CSVWriter.WriteData("Enonciation de lettres", "Categorie", "Temps de
        reponse");
118         StartCoroutine("ReadWord");
119     }
120
121     IEnumerator ReadWord()
122     {
123         nbRounds--;
124         isSpeaking = true;
125         voice.Speak(words[listenedWord][0].ToString(), SpeechVoiceSpeakFlags
        .SVSFlagsAsync);
126         yield return new WaitForSeconds(1);
127         for (int i = 0; i < shuffledWords[listenedWord].Length; i++)
128         {
129             Debug.Log("Speak Start");
130             voice.Speak(shuffledWords[listenedWord][i].ToString(),
        SpeechVoiceSpeakFlags.SVSFlagsAsync);
131             yield return new WaitForSeconds(delay);
132             Debug.Log("Speak End");
133         }
134         isSpeaking = false;
135         if (!isListening) { StartCoroutine("ListenWord"); }
136         yield return null;
137     }
138
139     IEnumerator ListenWord()
```

```
140     {
141         if (recognizer == null)
142         {
143             recognizer = new KeywordRecognizer(new string[] { words[
    listenedWord][1] }, confidence);
144             recognizer.OnPhraseRecognized += Recognizer_OnPhraseRecognized;
145         }
146             recognizer.Start();
147         isListening = true;
148         timer = 0.0f;
149         timeR = true;
150         txt.text = "En écoute";
151         while (word == null && isListening)
152         {
153             yield return new WaitForSeconds(0.5f);
154         }
155         if (isListening) { Debug.Log("Word OK"); }
156         timeR = false;
157         CSVWriter.WriteData(words[listenedWord][1], words[listenedWord][0],
    timer.ToString());
158         StopListening();
159         if (listenedWord < words.Count) { listenedWord++; }
160         else { listenedWord = 0; }
161         Debug.Log("End Listening");
162         StopEx();
163         if (nbRounds > 0) { StartCoroutine("ReadWord"); }
164         yield return null;
165     }
166 
167     public void StopEx()
168     {
169         exercicesManagements.running = false;
170         if (isSpeaking) { voice.Skip("Sentence", int.MaxValue);
    StopCoroutine("ReadWord"); }
171         if (isListening) { StopListening(); StopCoroutine("ListenWord"); }
172         CSVWriter.WriteData("");
173     }
174 
175     public void StopListening()
176     {
177         if (isListening) { isListening = false; }
178         if (word != null) { word = null; }
179         if (recognizer != null && recognizer.IsRunning)
180         {
181             recognizer.Stop();
182             recognizer = null;
183         }
184         txt.text = "";
185     }
```

```
186 }
```

Go/no Go

```
1  using System.Collections;
2  using UnityEngine;
3  using UnityEngine.UI;
4  using TMPro;
5
6  public class GoNoGo : MonoBehaviour
7  {
8      public Image cube;
9      public int delay = 2;
10     public int nbRounds = 5;
11     int rounds;
12     public int rolls = 0;
13     public bool isRunning;
14     public TextMeshProUGUI txt;
15
16     private float timer;
17     private bool timeR;
18
19     public ExercicesManagements exercicesManagements;
20
21     void Start()
22     {
23         cube.enabled = false;
24         isRunning = false;
25         timer = 0.0f;
26         timeR = false;
27     }
28
29     void Update()
30     {
31         if (Input.GetKeyDown("space") && isRunning && cube.color == Color.
       blue)
32         {
33             txt.color = Color.green;
34             timeR = false;
35             txt.text = "Bien joué !";
36             CSVWriter.WriteData((6 - rounds).ToString(),timer.ToString(), 1.
       ToString());
37         }
38         else if (Input.GetKeyDown("space") && isRunning && cube.color ==
       Color.black)
39         {
```

```
40              txt.color = Color.red;
41              txt.text = "Raté";
42              CSVWriter.WriteData((6 - rounds).ToString(), "rate", 0.ToString
    ());
43          }
44          if (timeR)
45          {
46              timer += Time.deltaTime;
47          }
48      }
49
50      private void ChangeColor()
51      {
52          if (cube.color == Color.black) { cube.color = Color.blue; }
53          else { cube.color = Color.black; }
54      }
55
56      private void StartGNG()
57      {
58          exercicesManagements.running = true;
59          cube.color = Color.black;
60          cube.enabled = true;
61          isRunning = true;
62          CSVWriter.WriteData("GoNoGo", "Temps Reaction", "Reussite");
63          StartCoroutine("GNG");
64      }
65
66      private void StopGNG()
67      {
68          exercicesManagements.running = false;
69          isRunning = false;
70          if (cube.isActiveAndEnabled) { cube.enabled = false; }
71          CSVWriter.WriteData("");
72          txt.color = Color.blue;
73          txt.text = "";
74      }
75
76      IEnumerator GNG()
77      {
78          rounds = nbRounds;
79          rolls = 0;
80          yield return new WaitForSeconds(delay);
81          while (rounds > 0)
82          {
83              txt.text = "";
84              int rand = Random.Range(0, 3);
85              if ((rand == 0 && cube.color == Color.black) || rolls >= 4)
86              {
87                  ChangeColor();
```

```
88                  rolls = 0;
89                  Debug.Log("1");
90                  timeR = true;
91              }
92              else if(cube.color == Color.blue)
93              {
94                  ChangeColor();
95                  rounds--;
96                  Debug.Log(rounds);
97                  if (timeR) { CSVWriter.WriteData((6 - rounds).ToString(), "
    temps ecoule", 2.ToString()); }
98                  timeR = false;
99                  timer = 0.0f;
100             }
101             else
102             {
103                 rolls++;
104             }
105             yield return new WaitForSeconds(delay);
106         }
107         txt.text = "Fin GoNoGo";
108         StopGNG();
109         yield return null;
110     }
111
112     public void ButtonClick()
113     {
114         if (!isRunning) { StartGNG(); }
115         else { StopCoroutine("GNG"); StopGNG(); }
116     }
117 }
```

ExercisesManagement

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using TMPro;
5
6  public class ExercicesManagements : MonoBehaviour
7  {
8      public WalkingExercice walkingExercice;
9      public TestStroop testStroop;
10     public MirrorWords mirrorWords;
11     public GoNoGo goNoGo;
12     public NumbersManipulation numbersManipulation;
```

```
13      public CutWords cutWords;
14      public SortABC sortABC;
15      public ShuffledWords shuffledWords;
16      public GNGmulti gNGmulti;
17      public PhysicalExercise physicalExercise;
18
19      public TextMeshProUGUI txt;
20      public DataReader data;
21
22      private bool started;
23      public bool running;
24
25      void Start()
26      {
27          walkingExercice = GetComponent<WalkingExercice>();
28          testStroop = GetComponent<TestStroop>();
29          mirrorWords = GetComponent<MirrorWords>();
30          goNoGo = GetComponent<GoNoGo>();
31          numbersManipulation = GetComponent<NumbersManipulation>();
32          cutWords = GetComponent<CutWords>();
33          sortABC = GetComponent<SortABC>();
34          shuffledWords = GetComponent<ShuffledWords>();
35          gNGmulti = GetComponent<GNGmulti>();
36          physicalExercise = GetComponent<PhysicalExercise>();
37          data = new DataReader();
38          CSVWriter.CreateFile();
39          running = false;
40          txt.color = Color.white;
41          txt.text = "Appuyer sur Espace pour commencer";
42          started = false;
43      }
44
45      private void Update()
46      {
47          /*if (Input.GetKeyDown("space") && !started)
48          {
49              txt.text = "";
50              StartSimu();
51          }*/
52      }
53
54      private void StartSimu()
55      {
56          started = true;
57          int diff = data.loadedData.difficultyP;
58          switch (diff)
59          {
60              case 0:
61                  StartCoroutine("SimuEasy");
```

```
62                  break;
63              case 1:
64                  StartCoroutine("SimuNormal");
65                  break;
66              case 2:
67                  StartCoroutine("SimuHard");
68                  break;
69              default:
70                  StartCoroutine("SimuNormal");
71                  break;
72          }
73
74      }
75
76      private void EndSimu()
77      {
78          started = false;
79      }
80
81      IEnumerator SimuEasy()
82      {
83          Debug.Log("-----EASY-----");
84          StepEasy();//
        ------------------------------------------------------------------------------------------EXERCICE
85          while (running)
86          {
87              yield return new WaitForSeconds(0.5f);
88          }
89          txt.color = Color.white;
90          txt.text = "Exercice Suivant";
91          yield return new WaitForSeconds(5);
92          txt.text = "";
93          GoNoGo();//
        ------------------------------------------------------------------------------------------EXERCICE
94          while (running)
95          {
96              yield return new WaitForSeconds(0.5f);
97          }
98          txt.color = Color.white;
99          txt.text = "Exercice Suivant";
100         yield return new WaitForSeconds(5);
101         txt.text = "";
102         HancheEasy();//
        ------------------------------------------------------------------------------------------EXERCICE
103         while (running)
104         {
105             yield return new WaitForSeconds(0.5f);
106         }
107         txt.color = Color.white;
```

```
108         txt.text = "Exercice Suivant";
109         yield return new WaitForSeconds(5);
110         txt.text = "";
111         SortABC();//
        ------------------------------------------------------------------------------------------EXERCICE
112         while (running)
113         {
114             yield return new WaitForSeconds(0.5f);
115         }
116         SortABC();
117         while (running)
118         {
119             yield return new WaitForSeconds(0.5f);
120         }
121         txt.color = Color.white;
122         txt.text = "Exercice Suivant";
123         yield return new WaitForSeconds(5);
124         txt.text = "";
125         MirrorWords(0);//
        ------------------------------------------------------------------------------------------EXERCICE
126         while (running)
127         {
128             yield return new WaitForSeconds(0.5f);
129         }
130         txt.color = Color.white;
131         txt.text = "Fin de la séance";
132         CSVWriter.WriteData("Fin de Seance");
133         CSVWriter.WriteData("");
134         CSVWriter.WriteData("");
135         Debug.Log("------EndEASY------");
136         EndSimu();
137         yield return null;
138     }
139
140     IEnumerator SimuNormal()
141     {
142         Debug.Log("------MEDIUM------");
143         StepMedium();//
        ------------------------------------------------------------------------------------------EXERCICE
144         while (running)
145         {
146             yield return new WaitForSeconds(0.5f);
147         }
148         txt.color = Color.white;
149         txt.text = "Exercice Suivant";
150         yield return new WaitForSeconds(5);
151         txt.text = "";
152         CutWords();//
        ------------------------------------------------------------------------------------------EXERCICE
```

```
153             while (running)
154             {
155                 yield return new WaitForSeconds(0.5f);
156             }
157             txt.color = Color.white;
158             txt.text = "Exercice Suivant";
159             yield return new WaitForSeconds(5);
160             txt.text = "";
161             PushUpMedium();//
        ----------------------------------------------------------------------------------------------------EXERCICE
162             while (running)
163             {
164                 yield return new WaitForSeconds(0.5f);
165             }
166             txt.color = Color.white;
167             txt.text = "Exercice Suivant";
168             yield return new WaitForSeconds(5);
169             txt.text = "";
170             MirrorWords(1);//
        ----------------------------------------------------------------------------------------------------EXERCICE
171             while (running)
172             {
173                 yield return new WaitForSeconds(0.5f);
174             }
175             txt.color = Color.white;
176             txt.text = "Exercice Suivant";
177             yield return new WaitForSeconds(5);
178             txt.text = "";
179             EndSimu();
180             HancheMedium();//
        ----------------------------------------------------------------------------------------------------EXERCICE
181             while (running)
182             {
183                 yield return new WaitForSeconds(0.5f);
184             }
185             txt.color = Color.white;
186             txt.text = "Fin de la séance";
187             CSVWriter.WriteData("Fin de Seance");
188             CSVWriter.WriteData("");
189             CSVWriter.WriteData("");
190             Debug.Log("----EndMEDIUM----");
191             EndSimu();
192             yield return null;
193         }
194
195         IEnumerator SimuHard()
196         {
197             Debug.Log("----HARD----");
```

```
198         PushUpHard();//
    ------------------------------------------------------------------------------------EXERCICE
199         while (running)
200         {
201             yield return new WaitForSeconds(0.5f);
202         }
203         txt.color = Color.white;
204         txt.text = "Exercice Suivant";
205         yield return new WaitForSeconds(5);
206         txt.text = "";
207         MirrorWords(2);//
    ------------------------------------------------------------------------------------EXERCICE
208         while (running)
209         {
210             yield return new WaitForSeconds(0.5f);
211         }
212         txt.color = Color.white;
213         txt.text = "Exercice Suivant";
214         yield return new WaitForSeconds(5);
215         txt.text = "";
216         HancheHard();//
    ------------------------------------------------------------------------------------EXERCICE
217         while (running)
218         {
219             yield return new WaitForSeconds(0.5f);
220         }
221         txt.color = Color.white;
222         txt.text = "Exercice Suivant";
223         yield return new WaitForSeconds(5);
224         txt.text = "";
225         ShuffledWords();//
    ------------------------------------------------------------------------------------EXERCICE
226         while (running)
227         {
228             yield return new WaitForSeconds(0.5f);
229         }
230         txt.color = Color.white;
231         txt.text = "Exercice Suivant";
232         yield return new WaitForSeconds(5);
233         txt.text = "";
234         SquattingHard();//
    ------------------------------------------------------------------------------------EXERCICE
235         while (running)
236         {
237             yield return new WaitForSeconds(0.5f);
238         }
239         txt.color = Color.white;
240         txt.text = "Fin de la séance";
241         CSVWriter.WriteData("Fin de Seance");
```

```
242             CSVWriter.WriteData("");
243             CSVWriter.WriteData("");
244             Debug.Log("------EndHARD------");
245             EndSimu();
246             yield return null;
247         }
248 
249         public void WalkingExercice()
250         {
251             walkingExercice.ButtonClick();
252         }
253 
254         public void TestStroop()
255         {
256             testStroop.ButtonClick();
257         }
258 
259         public void MirrorWords(int diff)
260         {
261             mirrorWords.ButtonClick(diff);
262         }
263 
264         public void GoNoGo()
265         {
266             goNoGo.ButtonClick();
267         }
268 
269         public void NumbersManipulation()
270         {
271             numbersManipulation.ButtonClick();
272         }
273 
274         public void CutWords()
275         {
276             cutWords.ButtonClick();
277         }
278 
279         public void SortABC()
280         {
281             sortABC.ButtonClick();
282         }
283 
284         public void ShuffledWords()
285         {
286             shuffledWords.ButtonClick();
287         }
288 
289         public void GNGmulti()
290         {
```

```
291             gNGmulti.ButtonClick();
292         }
293 
294         public void WalkingEasy()
295         {
296             physicalExercise.ButtonClickWalkingEasy();
297         }
298 
299         public void WalkingMedium()
300         {
301             physicalExercise.ButtonClickWalkingMedium();
302         }
303 
304         public void PushUpHard()
305         {
306             physicalExercise.ButtonClickPushupHard();
307         }
308 
309         public void PushUpMedium()
310         {
311             physicalExercise.ButtonClickPushupMedium();
312         }
313 
314         public void StepEasy()
315         {
316             physicalExercise.ButtonClickStepEasy();
317         }
318 
319         public void StepMedium()
320         {
321             physicalExercise.ButtonClickStepMedium();
322         }
323 
324         public void StepHard()
325         {
326             physicalExercise.ButtonClickStepHard();
327         }
328 
329         public void HancheEasy()
330         {
331             physicalExercise.ButtonClickHancheEasy();
332         }
333 
334         public void HancheMedium()
335         {
336             physicalExercise.ButtonClickHancheMedium();
337         }
338 
339         public void HancheHard()
```

```
340     {
341         physicalExercise.ButtonClickHancheHard();
342     }
343 
344     public void SquattingHard()
345     {
346         physicalExercise.ButtonClickSquatingHard();
347     }
348 }
```

Menu (gestion de l'interface du menu)

```
1  using System;
2  using System.Collections;
3  using System.Collections.Generic;
4  using UnityEngine;
5  using UnityEngine.SceneManagement;
6 
7  public class Menu : MonoBehaviour
8  {
9      public GameObject canvaMenu;
10     public GameObject canvaSettings;
11     public GameObject canvaDifficulty;
12 
13     private int difficultyP = 1;
14     private int difficultyC = 1;
15 
16     void Start()
17     {
18         if (!canvaMenu.activeInHierarchy) { canvaMenu.SetActive(true); }
19         if (canvaSettings.activeInHierarchy) { canvaSettings.SetActive(false
       ); }
20         if (canvaDifficulty.activeInHierarchy) { canvaDifficulty.SetActive(
       false); }
21     }
22 
23     void Update()
24     {
25 
26     }
27 
28     public void SettingsActivate()
29     {
30         canvaSettings.SetActive(true);
31         canvaMenu.SetActive(false);
32         canvaDifficulty.SetActive(false);
```

```
33      }
34
35      public void MenuActivate()
36      {
37          canvaMenu.SetActive(true);
38          canvaSettings.SetActive(false);
39          canvaDifficulty.SetActive(false);
40      }
41
42      public void DifficultyActivate()
43      {
44          canvaDifficulty.SetActive(true);
45          canvaSettings.SetActive(false);
46          canvaMenu.SetActive(false);
47      }
48
49      public void LaunchSimulation()
50      {
51          StoredInfo data = new StoredInfo();
52          DataReader dataReader = new DataReader();
53          if (dataReader.loadedData != null)
54          {
55              data = dataReader.loadedData;
56          }
57          data.difficultyP = difficultyP;
58          DataSaver.saveData(data,"storedData");
59          CSVWriter.CreateFile();
60          //SceneManager.LoadScene(1);
61          CSVWriter.WriteData("Seance du " + DateTime.Now.ToString("dd/MM"),"
        DifficulteP " + difficultyP.ToString(), "DifficulteC " + difficultyC.
        ToString());
62          CSVWriter.WriteData("");
63          SceneManager.LoadScene(1);
64      }
65
66      public void SetDifficultyP(int diff)
67      {
68          difficultyP = diff;
69      }
70
71      public void SetDifficultyC(int diff)
72      {
73          difficultyC = diff;
74      }
75
76      public void ResetDifficulty()
77      {
78          difficultyP = 1;
79          difficultyC = 1;
```

```
80      }
81
82      public void Exit()
83      {
84          Application.Quit();
85          Debug.Log("ExitApplication");
86      }
87  }
```

CSVWriter

```
1   using System;
2   using System.Collections;
3   using System.Collections.Generic;
4   using System.IO;
5   using UnityEditor;
6   using UnityEngine;
7
8   public static class CSVWriter
9   {
10      public static void CreateFile()
11      {
12          try{
13              if (!File.Exists(getPath())) { var sr = File.Create(getPath());
14                  sr.Close();
15                  //AssetDatabase.ImportAsset("Assets/Data/" + DateTime.Now.
        ToString("dd-MM") + ".csv");
16              }
17          }
18          catch (Exception) { }
19      }
20
21      public static void WriteData(string s1, string s2 = "", string s3 = "",
        string s4 = "")
22      {
23          string filePath = getPath();
24          File.AppendAllText(filePath, s1 + ";" + s2 + ";" + s3 + ";" + s4 + "
        \n");
25          Debug.Log("WRITETEXT----------");
26          //AssetDatabase.ImportAsset(filePath);
27      }
28
29      private static string getPath()
30      {
31  #if UNITY_EDITOR
```

```
32          return Application.dataPath + DateTime.Now.ToString("dd-MM") + ".csv
    ";
33 #elif UNITY_ANDROID
34          return Application.persistentDataPath+DateTime.Now.ToString("dd-MM")
     + ".csv";
35 #elif UNITY_IPHONE
36          return Application.persistentDataPath+"/"+DateTime.Now.ToString("dd-
    MM") + ".csv";
37 #else
38          return Application.dataPath +"/"+DateTime.Now.ToString("dd-MM") + ".
    csv";
39 #endif
40      }
41 }
```

DataSaver

```
1  using System;
2  using System.IO;
3  using System.Text;
4  using UnityEngine;
5
6  public class DataSaver
7  {
8      //Save Data
9      public static void saveData<T>(T dataToSave, string dataFileName)
10     {
11         string tempPath = Path.Combine(Application.persistentDataPath, "data
    ");
12         tempPath = Path.Combine(tempPath, dataFileName + ".txt");
13
14         //Convert To Json then to bytes
15         string jsonData = JsonUtility.ToJson(dataToSave, true);
16         byte[] jsonByte = Encoding.ASCII.GetBytes(jsonData);
17
18         //Create Directory if it does not exist
19         if (!Directory.Exists(Path.GetDirectoryName(tempPath)))
20         {
21             Directory.CreateDirectory(Path.GetDirectoryName(tempPath));
22         }
23         //Debug.Log(path);
24
25         try
26         {
27             File.WriteAllBytes(tempPath, jsonByte);
28             Debug.Log("Saved Data to: " + tempPath.Replace("/", "\\"));
```

```
29          }
30          catch (Exception e)
31          {
32              Debug.LogWarning("Failed To PlayerInfo Data to: " + tempPath.
    Replace("/", "\\"));
33              Debug.LogWarning("Error: " + e.Message);
34          }
35      }
36
37      //Load Data
38      public static T loadData<T>(string dataFileName)
39      {
40          string tempPath = Path.Combine(Application.persistentDataPath, "data
    ");
41          tempPath = Path.Combine(tempPath, dataFileName + ".txt");
42
43          //Exit if Directory or File does not exist
44          if (!Directory.Exists(Path.GetDirectoryName(tempPath)))
45          {
46              Debug.LogWarning("Directory does not exist");
47              return default(T);
48          }
49
50          if (!File.Exists(tempPath))
51          {
52              Debug.Log("File does not exist");
53              return default(T);
54          }
55
56          //Load saved Json
57          byte[] jsonByte = null;
58          try
59          {
60              jsonByte = File.ReadAllBytes(tempPath);
61              Debug.Log("Loaded Data from: " + tempPath.Replace("/", "\\"));
62          }
63          catch (Exception e)
64          {
65              Debug.LogWarning("Failed To Load Data from: " + tempPath.Replace
    ("/", "\\"));
66              Debug.LogWarning("Error: " + e.Message);
67          }
68
69          //Convert to json string
70          string jsonData = Encoding.ASCII.GetString(jsonByte);
71
72          //Convert to Object
73          object resultValue = JsonUtility.FromJson<T>(jsonData);
74          return (T)Convert.ChangeType(resultValue, typeof(T));
```

```
75      }
76
77      public static bool deleteData(string dataFileName)
78      {
79          bool success = false;
80
81          //Load Data
82          string tempPath = Path.Combine(Application.persistentDataPath, "data
        ");
83          tempPath = Path.Combine(tempPath, dataFileName + ".txt");
84
85          //Exit if Directory or File does not exist
86          if (!Directory.Exists(Path.GetDirectoryName(tempPath)))
87          {
88              Debug.LogWarning("Directory does not exist");
89              return false;
90          }
91
92          if (!File.Exists(tempPath))
93          {
94              Debug.Log("File does not exist");
95              return false;
96          }
97
98          try
99          {
100             File.Delete(tempPath);
101             Debug.Log("Data deleted from: " + tempPath.Replace("/", "\\"));
102             success = true;
103         }
104         catch (Exception e)
105         {
106             Debug.LogWarning("Failed To Delete Data: " + e.Message);
107         }
108
109         return success;
110     }
111 }
```

Bibliographie

[1] Manera V. Ben-Sadoun G., Sacco G. et al. Physical and cognitive stimulation using an exergame in subjects with normal aging, mild and moderate cognitive impairment. *Journal of Alzheimer's Disease*, 53(4) :1299–1314, 2016.

[2] R. van Deursen H. Khalil, L. Quinn et al. Adherence to use of a home-based exercise dvd in people with huntington disease : Participants' perspectives. In *Physical Therapy, Volume 92, Issue 1, 1 January 2012, Pages 69–82*, pages 69–82, 2012.

[3] J-S. Vidal L. Djabelkhir, Y-H. Wu et al. Computerized cognitive stimulation and engagement programs in older adults with mild cognitive impairment : comparing feasibility, acceptability, and cognitive and psychosocial effects. *Clinical Interventions in Aging*, 12(1) :1967–1975, 2017.

Résumé — Ce rapport synthétise les recherches et le travail réalisés pour concevoir un outil numérique adapté et exploitable pour les patients atteints de la maladie de Huntington, dans le cadre du projet CoMoN. La première partie de ce rapport traite des objectifs, des contraintes et des besoins identifiés pour la conception de cet outil. Cette partie présente aussi les différents exercices sélectionnés ainsi que le déroulement d'une séance type. La deuxième partie s'attache à détailler les choix réalisés au cours de la conception de cette application et les solutions techniques mise en place pendant son développement. Enfin, la dernière partie présente les apports de l'outil numérique créé et ses innovations. Il y est aussi détaillé les évolutions prévues ainsi que les retours obtenus lors de tests.

Mots clés : Maladie de Huntington, CoMoN, Neurodégénérescence, Stimulation physique, Stimulation cognitive, Outil numérique, Unity3D, C#, Motivation, Accessibilité, Retours d'expérience

Abstract — This report synthesizes the research and work carried out to design a suitable and usable digital tool for patients with Huntington's disease, as part of the CoMoN project. The first part of this report deals with the objectives, constraints and needs identified for the design of this tool. This part also presents the various exercises selected as well as the course of a typical session. The second part focuses on detailing the choices made during the design of this application and the technical solutions implemented during its development. Finally, the last part presents the contributions of the digital tool created and its innovations. It is also detailed there the planned evolutions as well as the feedback obtained during tests.

Mots clés : Huntington's disease, CoMoN, Neurodegeneration, Physical stimulation, Cognitive stimulation, Digital tool, Unity3D, C#, Motivation, Accessibility, Feedback

Polytech Angers
62, avenue Notre Dame du Lac
49000 Angers