



PROJET 4INFO

Sport-Santé et numérique :
Application intégrant une reconnaissance prédictive de
geste pour lutter contre la sédentarisation

STAY IN MOTION

Rapport final

Auteurs :

Firmin CADOT
Teddy GESBERT
Léandre LE BIZEC

Encadrants :

Éric ANQUETIL, professeur à l'INSA et chercheur à l'IRISA
William MOCAËR, doctorant à l'IRISA
Richard KULPA, maître de conférence à Rennes 2 et chercheur à l'IRISA

En collaboration avec les étudiants de DIGISPORT :

Guillaume COBAT
Marie-Aurélie CASTEL
Kilian BERTHOLON

Table des matières

1	Introduction	2
2	Rappel du contexte	2
3	Bilan fonctionnel	3
3.1	Retour sur le cahier des charges	3
3.2	Etat de l'application	5
3.2.1	SIM	5
3.2.2	Démonstrateur	6
4	Bilan de conception	8
4.1	Application Stay in Motion	8
4.1.1	Ajout d'un package de gestion des gestes	8
4.1.2	Ajout d'element pour la gestion des scènes	9
4.2	Démonstrateur associé au moteur de reconnaissance de gestes 3D	10
4.2.1	Nouveau package : 3D BarChart	11
4.2.2	Architecture finale	13
5	Bilan de planification	15
5.1	Rappel du versionnage des applications	15
5.2	Avancée finale du projet	15
5.3	Bilan des outils utilisés	16
6	Manuel Utilisateur	17
6.1	Application Stay in Motion	17
6.2	Démonstrateur associé au moteur de reconnaissance de gestes 3D	18
7	Manuel d'installation	21
7.1	Pré-requis pour l'installation	21
7.2	Installation de l'environnement	21
7.2.1	Pilotes et SDK	21
7.3	Installation des différentes parties du projet	22
7.3.1	Serveur	22
7.3.2	Client (Projet Unity)	22
8	Conclusion	22

1 Introduction

Suite aux phases de pré-étude et de spécification [9] ainsi qu'à la phase de planification [8], au cours desquelles nous avons pu définir les besoins et les objectifs du projet, préciser les fonctionnalités et spécifications, ainsi que proposer une première planification du projet, nous avons travaillé sur la conception de notre projet. Cette étape indispensable nous a permis de détailler à la fois l'architecture de notre projet et l'ensemble des solutions logicielles choisies.

Ce document, complémentaire aux rapports précédents, regroupe plusieurs éléments. Nous y présentons l'architecture finale de notre projet, les mises à jour des spécifications fonctionnelles et de la conception, un bilan de la planification, et enfin un manuel utilisateur et la documentation technique.

2 Rappel du contexte

La réalisation de l'application "Stay In Motion" (SIM) et du démonstrateur associé au moteur de reconnaissance de gestes 3D pour la recherche s'inscrit dans le cadre de notre deuxième année au sein du département informatique de l'INSA de Rennes. Toujours dans le but de nous préparer au travail d'ingénieur, ce projet a pour objectif de nous confronter à des problématiques techniques et de gestion de projet en nous mettant au défi de mener à bien chaque étape du processus jusqu'à sa concrétisation.

Le projet que nous menons tire son origine d'un enjeu majeur : le manque d'activité physique chez une partie de la jeune population. En effet, selon l'ANSES, deux tiers des jeunes âgés de 11 à 17 ans présentent un risque sanitaire associé à la sédentarité et à l'inactivité physique [1]. Cette problématique tend à se détériorer d'année en année. Parallèlement à cette baisse de la pratique sportive, l'utilisation du numérique ne cesse de croître [5]. De plus, ces dernières années, un intérêt particulier pour l'intelligence artificielle est apparu, et elle est désormais utilisée dans de nombreux objets numériques de notre quotidien. Ainsi, l'idée d'imaginer une application où l'utilisateur pourrait effectuer des séances de sport en lien avec un environnement virtuel et qui utilise un système d'intelligence artificielle pour renforcer l'interaction en temps réel pourrait être considérée comme attrayante pour un jeune public et ainsi leur donner goût à l'effort physique.

L'application que nous avons développée s'inscrit dans un cadre précis. En 2018/2019 a été réalisé le projet "DERG3D" [6] qui est une application servant de démonstrateur au moteur de reconnaissance de geste de Yacine Boulahia [4]. S'en est suivi le projet "R3G" [3] en 2020/2021, qui est une suite logicielle plus complète proposant des outils de visualisation plus avancées, ainsi qu'un démonstrateur adapté au nouveau moteur de reconnaissance de gestes [10]. L'année dernière, en 2021/2022, s'est formé le projet "MOP" [2] qui est une preuve de concept de l'application visant à lutter contre le manque d'activité physique chez une partie de la jeune population. Nous avons travaillé sur ces deux objectifs :

- Développer la preuve de concept "MOP" à l'état d'application : "SIM" ;
- À partir d'une brique logicielle de "R3G", créer un démonstrateur destiné à la recherche.

Ce contexte est explicité en figure 1.

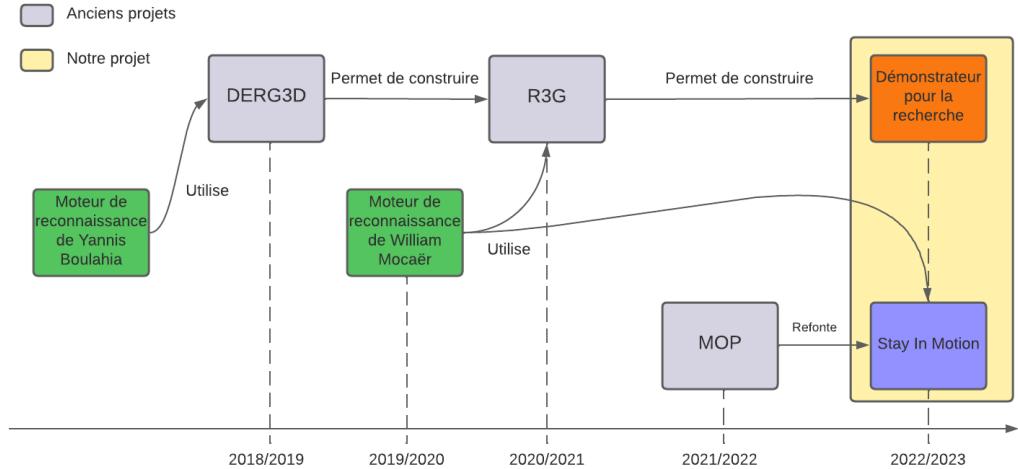


FIGURE 1 – Contexte de développement de notre projet

Le démonstrateur associé au moteur de reconnaissance de gestes 3D a pour objectif d’assister l’équipe IntuiDoc afin d’améliorer le système de reconnaissance de gestes. Pour cela, le démonstrateur permettra de tester de manière interactive la capacité du moteur à décider rapidement du geste qui sera effectué par l’utilisateur. L’évolution de la prise de décision devra être visuelle afin de permettre de suivre la compétition entre les différentes classes de gestes possibles. Le démonstrateur devra être robuste et capable d’utiliser correctement les résultats du moteur. Nous pourrons également expérimenter sa qualité et sa précision. Cette partie du projet sera donc un outil spécifique à la recherche.

3 Bilan fonctionnel

Tout au long de notre travail sur ce projet, le cahier des charges établi dans le rapport de pré-étude et spécifications fonctionnelles, a été mis à jour, pour aboutir à une version finale dans le rapport de conception [7].

3.1 Retour sur le cahier des charges

Afin de dresser le bilan fonctionnel de ce projet, nous allons reprendre le cahier des charges dans sa version finale et le commenter à l'aide du code couleur suivant :

- Bleu : Application “SIM”;
- Orange : Démonstrateur pour la recherche;
- Gris : Le développement de cette fonctionnalité diffère partiellement de ce qui avait été initialement prévu ou n'a pas abouti.

Le cahier des charges présenté en table 1 suit cette consigne.

TABLE 1 – Cahier des charges mis à jour

Fonctionnalités			
N°	Intitulé	Critère	Priorité
Refonte de l'application côté développeur			1
N°1.1	Généricité des capteurs	Architecture permettant d'ajouter de manière normée un nouvel appareil de capture.	1
Refonte ergonomique de l'application			2
N°2.1	Coach virtuel	Avatar 3D montrant le mouvement à faire.	2.1
N°2.2	Modèle 3D	Avatar 3D représentant l'utilisateur sportif.	2.1
N°2.3	Avancement de la séance	Indication permettant de connaître l'avancement de la séance.	2.2
N°2.4	Timer	Indication permettant de connaître le temps restant d'un exercice ou d'un temps de repos.	2.2
N°2.5	Barre de progression	Indication permettant de connaître le score de l'utilisateur sportif.	2.2
N°2.6	Nombre de répétitions	Indication permettant de connaître le nombre de répétitions de l'utilisateur sportif.	2.2
N°2.7	Environnement	Mise en place du fond et des couleurs.	2.2
Généricité			3
N°8.2	Généricité des capteurs	Notre démonstrateur doit pouvoir afficher fonctionnellement l'utilisateur et les résultats de l'IA en temps réel peu importe le capteur de mouvement utilisé.	3.1
N°8.1	Généricité du démonstrateur	Gestion de la variation du nombre de classes de gestes et des différentes natures de gestes.	3.2
Affichage dynamique des gestes en compétition			4
N°6.1	Représentation des résultats	L'utilisateur encadrant peut visualiser les résultats de l'IA grâce à l'affichage des histogrammes correspondant aux différents gestes en compétition.	4
N°6.2	Seuil d'activation des histogrammes	Sélection entre les différents gestes de la base de données effectuée avant ou pendant l'affichage de l'histogramme grâce à un filtre.	4
Ajout d'un exercice implémentant l'IA			5
N°3.1	Coach virtuel	Avatar 3D réagissant aux mouvements faits par l'utilisateur sportif.	5.1
N°3.2	Modèle 3D	Avatar 3D permettant à l'utilisateur sportif de se visualiser.	5.1
N°3.3	Nombre de coups réussis	Nombre de coups reconnus par l'IA.	5.2
N°3.4	Nombre de coups qui ont touché	Nombre de coups reconnus par l'IA n'ayant pas déclenché la réaction du coach virtuel à temps.	5.2

L'ensemble des fonctionnalités qui ne sont pas grises ont été développées et fonctionnent comme décrit dans la section critères. Le point 8.1 n'a été que partiellement développé. Il est possible d'utiliser le démonstrateur avec différentes bases de données, mais cela nécessite plusieurs actions au niveau du code. Cette fonctionnalité n'est donc pas tout à fait générique. L'ajout d'un exercice implémentant l'IA (3.1 à 3.4) n'a pas été développé. Nous reviendrons sur ce point dans le bilan de planification.

3.2 Etat de l'application

Pour réaliser nos deux applications, nous sommes partie d'une base existante puis nous avons maquetté une nouvelle interface. Dans cette partie nous allons comparer ces trois entités :

- L'application existante ;
- La maquette réalisée dans la phase de spécification fonctionnelle ;
- Le rendu final ;

3.2.1 SIM

Les figures 2, 3 et 4 illustrent ces trois entités pour l'application “SIM”.

Après la refonte de l'architecture de la preuve de concept “MOP”, nous nous sommes penchés sur la refonte ergonomique de l'application existante présentée en figure 2. Nous avons alors produit la maquette de “SIM” présentée en figure 3. En cette fin de phase de production, nous sommes arrivés au résultat présenté en figure 4. Le résultat final comporte les principaux éléments de la maquette et est ergonomiciquement proche de cette dernière.



FIGURE 2 – Application existante “MOP”

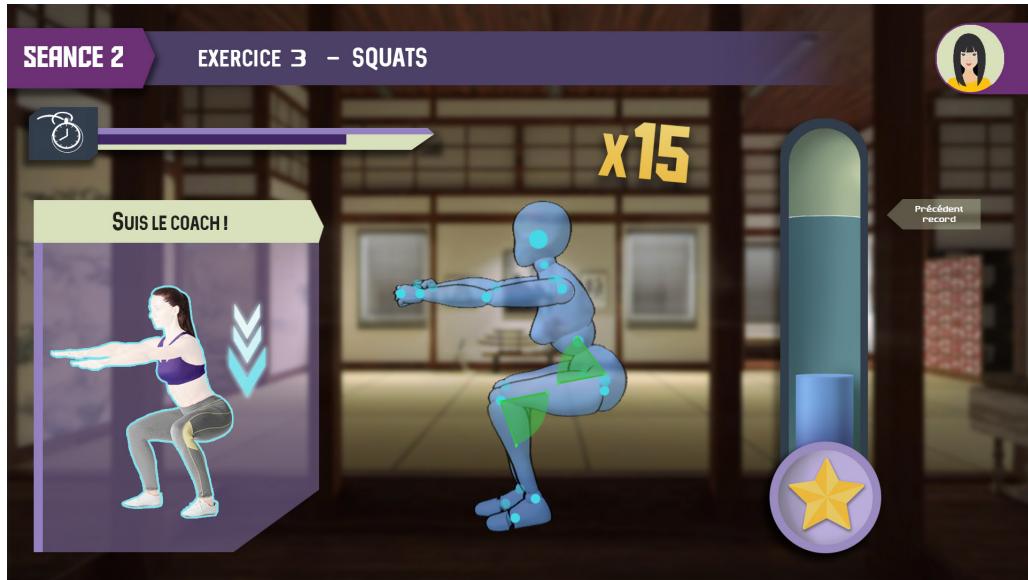


FIGURE 3 – Maquette de “SIM”

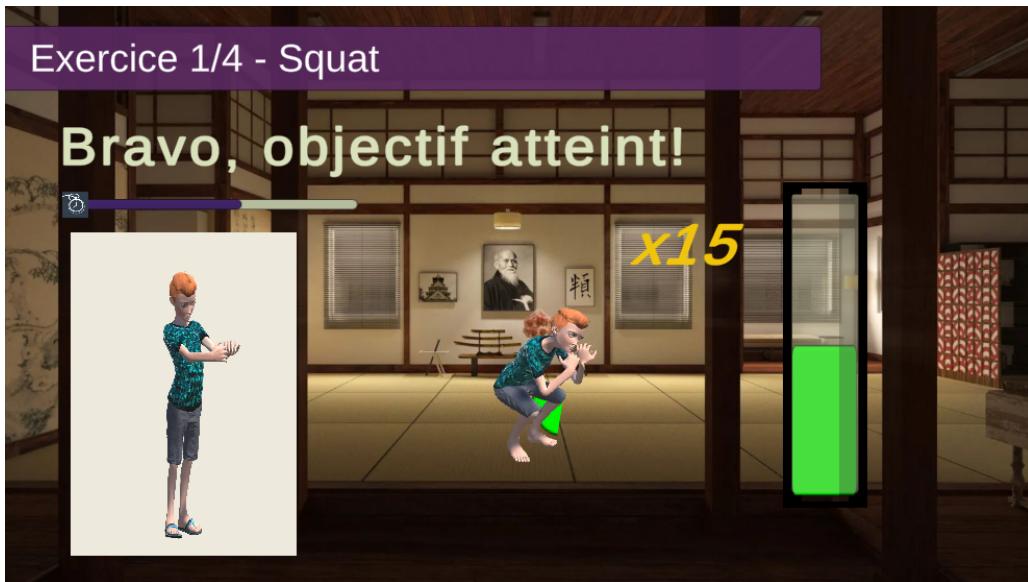


FIGURE 4 – Rendu final de "SIM"

3.2.2 Démonstrateur

Les figures 5, 6 et 7 illustrent respectivement l'application existante, la maquette et le rendu final pour le démonstrateur.

La base existante présentée en figure 5 n'était pas ergonomique et ne permettait pas l'analyse dynamique des résultats renvoyés par le module de reconnaissance de gestes. Nous avons imaginé la maquette présentée en figure 6 afin de répondre à ces besoins. Nous avons ensuite produit le rendu final présenté en figure 7. Ce rendu final améliore l'ergonomie de la base existante tout en restant suffisamment proche. Il permet désormais l'analyse dynamique des résultats envoyés par le module de reconnaissance de gestes. Nous avons décidé de garder quelques éléments de l'interface de l'existant pour avoir des icônes de geste plus lisibles que ceux sur la maquette 6.

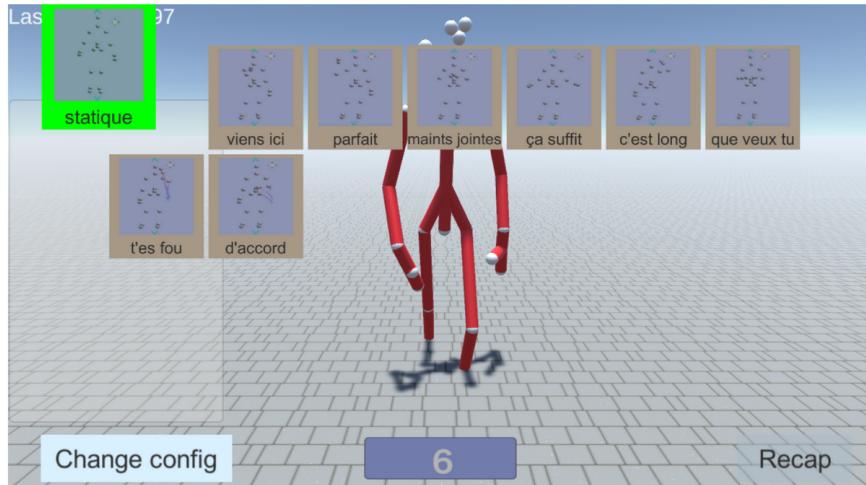


FIGURE 5 – Application existante du démonstrateur

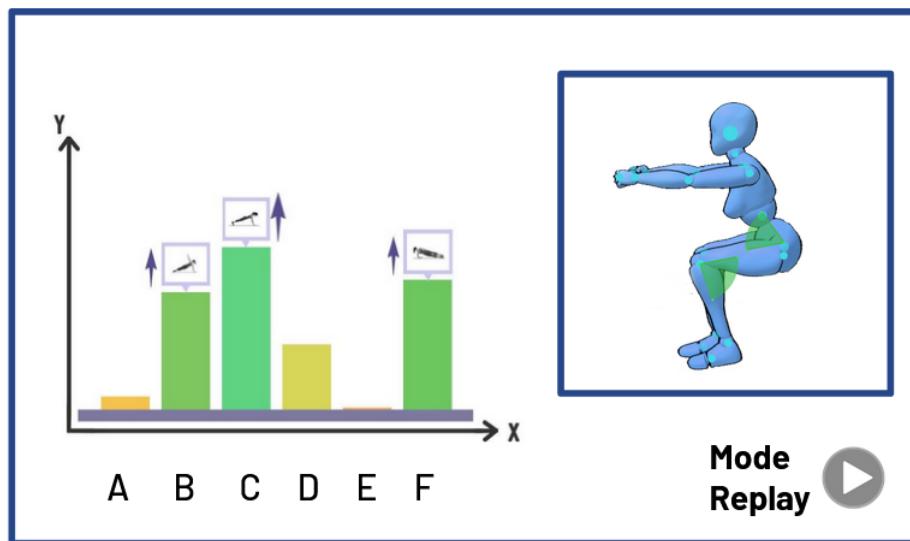


FIGURE 6 – Maquette du démonstrateur

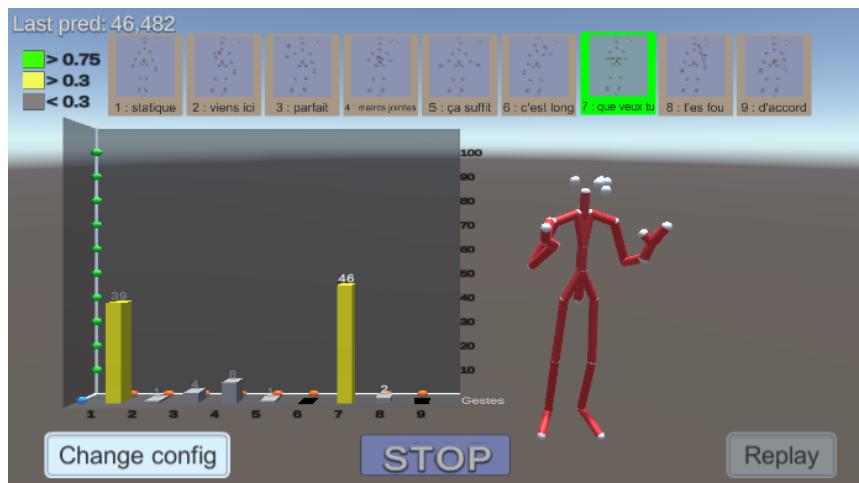


FIGURE 7 – Rendu final du démonstrateur

4 Bilan de conception

4.1 Application Stay in Motion

L'architecture de l'application "SIM" a quelque peu évolué par rapport à celle prévue dans le rapport de conception [7]. Nous avons travaillé sur deux points afin d'améliorer notre conception :

- L'ajout d'un package pour la gestion des gestes permettant de :
 - enregistrer des geste ;
 - rejouer des gestes ;
 - l'ajout d'élément pour mieux gérer les transitions de scènes.

4.1.1 Ajout d'un package de gestion des gestes

Afin de pouvoir animer notre coach, il est nécessaire de pouvoir enregistrer des mouvements et de pouvoir les rejouer. Nous avons donc décidé d'utiliser les modules "Acquisition" et "GesteReplay" du projet "R3G" [3], déjà utilisés dans l'ancien projet "MOP" [2]. Ces modules nous permettent d'enregistrer les positions enregistrées par la Kinect sous forme d'un document texte. À partir de ce document texte, il est possible de créer des sphères représentant les articulations, puis de les animer. On dispose alors d'un nuage de points dynamique représentant un squelette en mouvement. Ce squelette est par exemple visible sur la figure 2.

Le nuage de points représentant le squelette étant peu ergonomique, nous avons décidé de le remplacer par un avatar 3D. Cependant, nous avons rencontré un problème majeur : comme expliqué dans notre rapport de conception [7] dans la partie 3.2, l'avatar 3D suit les mouvements des articulations détectées par la Kinect en ne considérant que la rotation des articulations en mouvement. Cependant, les mouvements enregistrés par le module de gestion des gestes ne considéraient que la position des articulations à chaque instant et non la rotation. Il a donc fallu modifier ce module afin de récupérer la rotation, puis l'adapter à la génération d'avatar pour une séquence enregistrée et rejouée, et non une séquence captée en direct par la Kinect. Nous avons donc légèrement modifié l'architecture générale de notre projet, comme illustré sur la figure 8. La figure 9 présente le package de gestion des gestes.

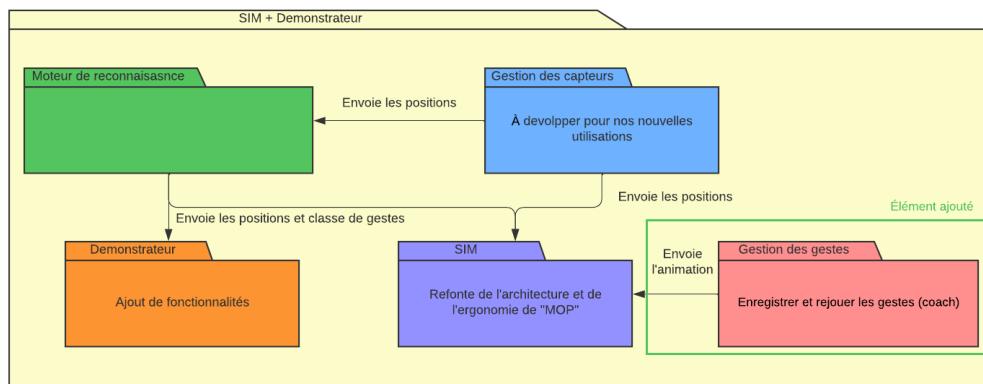


FIGURE 8 – Architecture générale de "SIM"

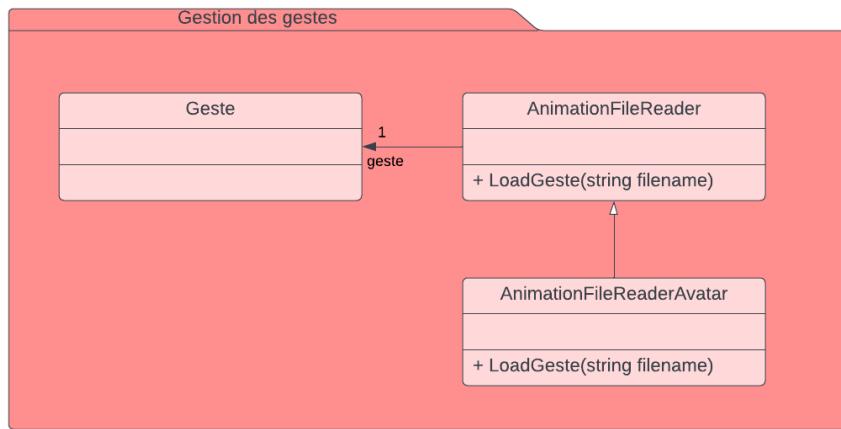


FIGURE 9 – Package de gestion des gestes

La classe Geste définit ce qu'est un geste. La classe AnimationFileReader permet de lire les fichiers texte et de créer des nuages de points dynamiques représentant un squelette en mouvement grâce à la fonction LoadGeste prenant en paramètre le chemin du fichier texte. Cependant, la classe AnimationFileReaderAvatar a été ajoutée pour permettre la liaison entre un avatar 3D et le nuage de points dynamique enregistré. Cette classe permet de lire un fichier texte et de récupérer le fichier de rotation associé. La fonction LoadGeste de cette classe a été redéfinie pour prendre en argument le chemin du fichier texte et permettre cette liaison.

4.1.2 Ajout d'élément pour la gestion des scènes

Le seul ajout que nous avons apporté visait à répondre au problème de changement de scène où le RecoManager ne parvenait pas à réinstancier une nouvelle capture de mouvement étant donné que celle-ci était déjà instanciée. Pour résoudre ce problème, nous avons donc utilisé la classe DontDestroyOnLoad, qui est une classe statique indépendante du scène manager. Ainsi, cette classe permet d'ajouter des GameObjects en tant qu'objets permanents dans cette classe tout au long des différentes scènes. Cela a donc changé la manière de modifier les GameObjects (voir figure 10) ainsi que la structure associée au changement de scènes (voir figure 11).

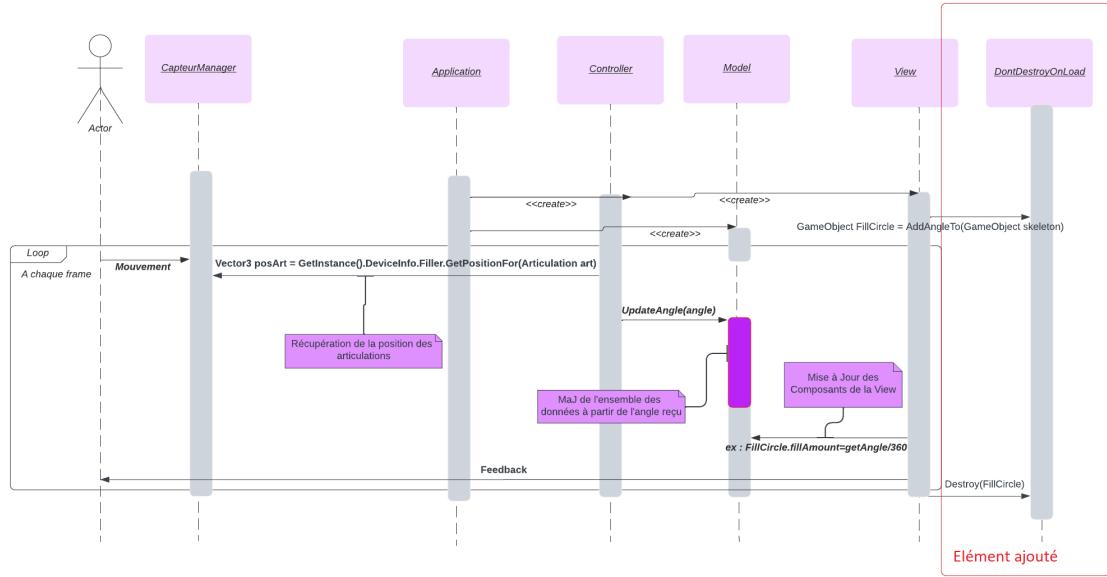


FIGURE 10 – Nouveau diagramme de séquence d'un exercice de squat

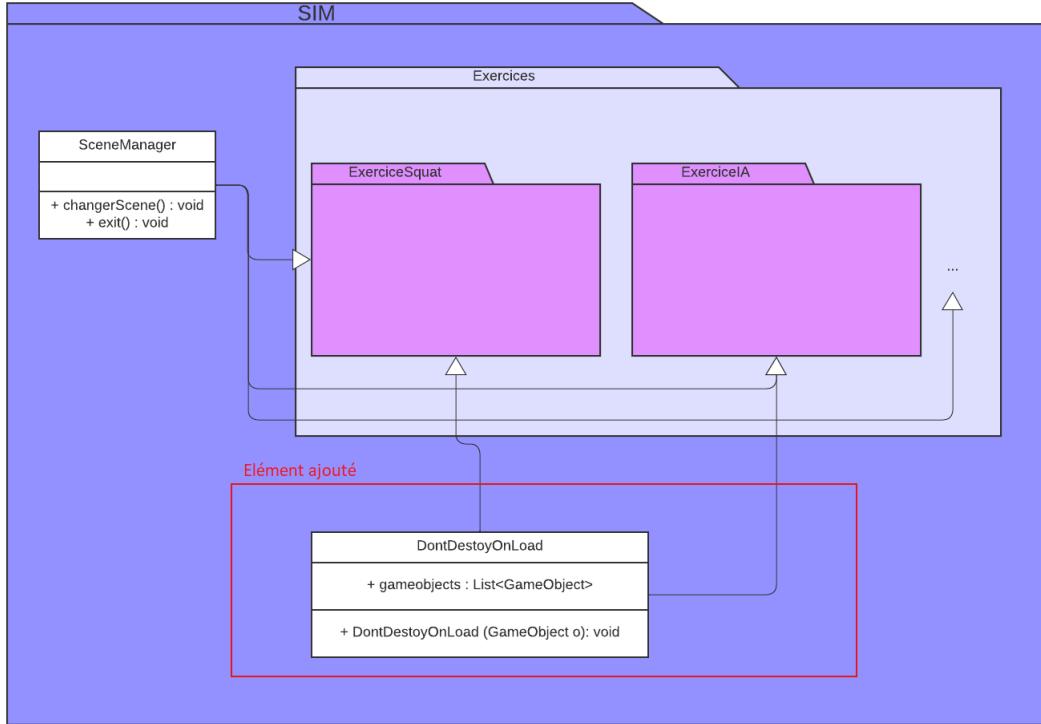


FIGURE 11 – Nouveau diagramme UML de la structure de "SIM" avec le sceneManager

4.2 Démonstrateur associé au moteur de reconnaissance de gestes 3D

L'architecture de l'application de démonstration a évolué au cours de la phase de développement et se s'est adaptés en fonction des besoins du client et des problèmes

rencontrés.

4.2.1 Nouveau package : 3D BarChart

Le principal changement d'architecture par rapport à celle décrite dans le rapport de conception [7] est l'ajout d'un package GUI (interface graphique utilisateur) de Unity, 3D Interactive Bar Chart (voir <https://assetstore.unity.com/packages/tools/gui/3d-interactive-barchart-157887>).

Ce package offre une large gamme de fonctionnalités pour la création d'histogrammes interactifs en 3D. L'utilisation de ce package a permis de simplifier la conception de l'interface utilisateur du démonstrateur et d'améliorer l'expérience utilisateur en offrant des graphiques à la fois dynamique et esthétiques. Les fonctionnalités offertes par ce package incluent la personnalisation des couleurs, la prise en charge de plusieurs séquences de données, ainsi que la possibilité de mettre à jour les données du graphique en temps réel à l'aide de plusieurs script. En figure 12, on peut observer un exemple d'interface graphique proposé par ce package.

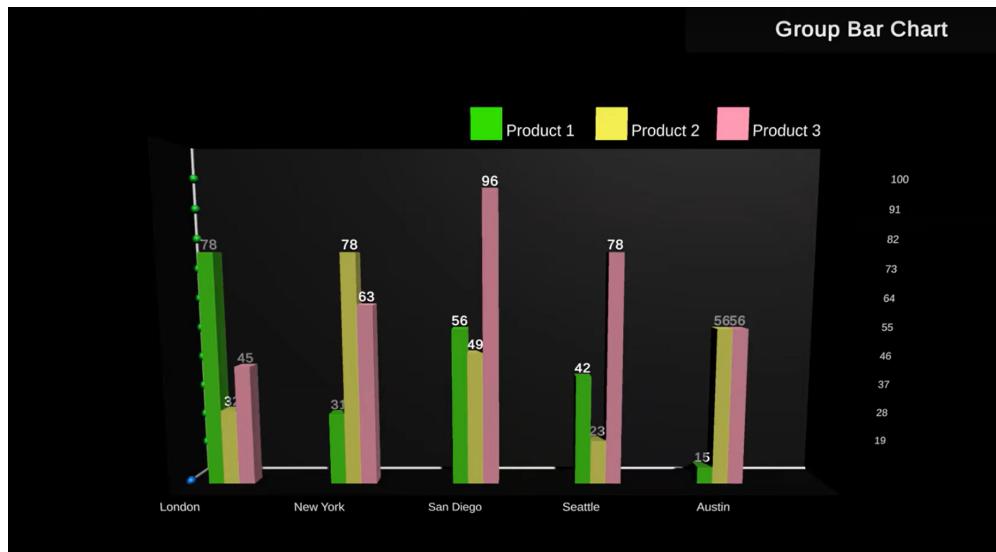


FIGURE 12 – Exemple d'interface graphique proposé par ce package

Cependant, l'utilisation de ce package n'a pas été sans défis. En effet, en raison de la grande quantité de fonctionnalités offertes et de la documentation peu fournie, la prise en main du package a été difficile. Nous avons dû consacrer un temps important pour comprendre comment fonctionnaient les différentes classes du package et comment les intégrer à notre application. De plus, nous avons constaté que certaines fonctionnalités ne répondraient pas à nos besoins spécifiques et avons donc dû procéder à des modifications de classes et de prefabs pour s'adapter en conséquence. On retrouve en figure 13 un diagramme UML décrivant les classes principales de ce nouveau package.

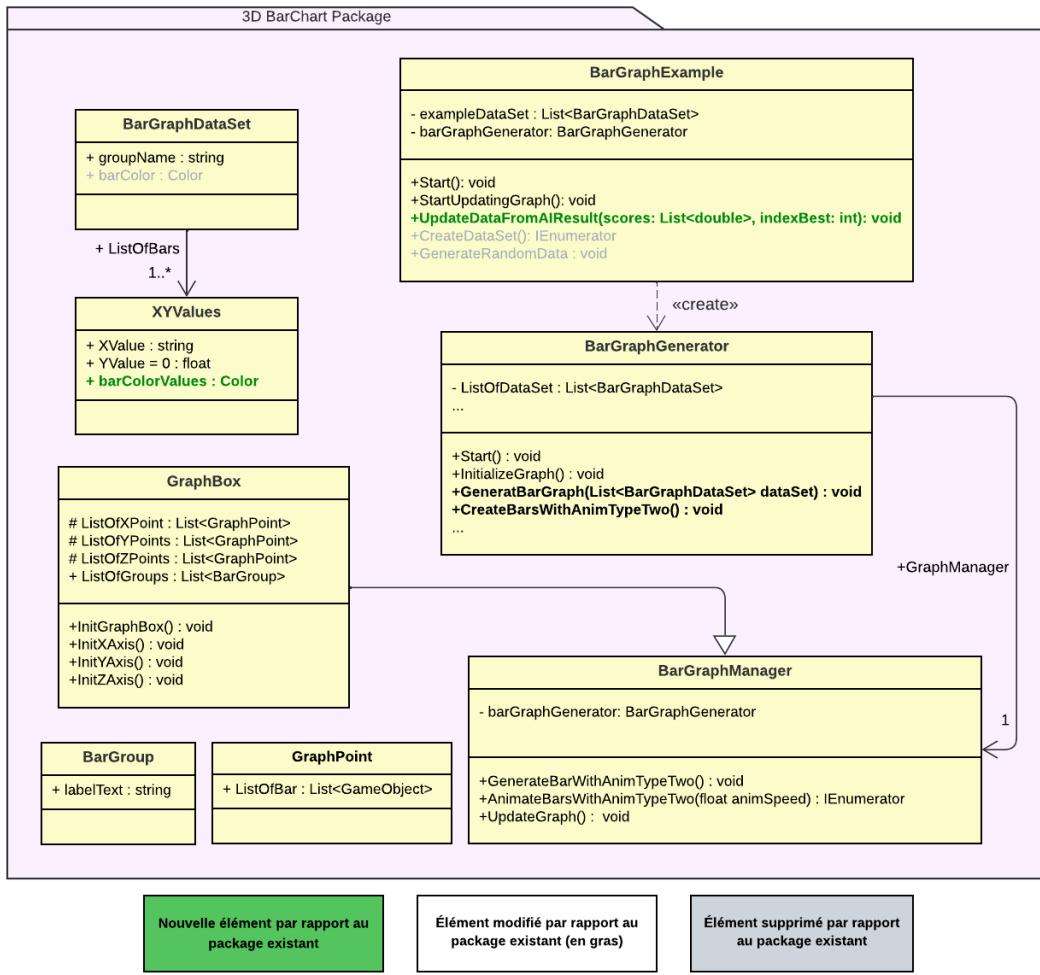


FIGURE 13 – Diagramme UML simplifié de l'architecture du package 3D BarChart

La classe principale du package est la classe **BarGraphGenerator** qui est responsable de la création et de la mise à jour de l'ensemble des histogrammes. Cette classe possède en attribut une liste d'objet de la classe **BarGraphDataSet**. Ces objets permettent de stocker les données nécessaires à la construction d'histogramme.

Les éléments en vert correspondent aux ajouts que l'on a ajouté par rapport au code du package déjà existant. La méthode **UpdateDataFromAIResult()** de la classe **BarGraphExample** permet de connecter ce nouveau package avec le moteur de reconnaissance de geste du démonstrateur. Cette méthode est utilisée pour mettre à jour l'histogramme en fonction des résultats de l'IA. Elle prend en entrée une liste de scores ainsi que l'index de la meilleure prédiction (c'est-à-dire l'index de l'élément de la liste avec le score le plus élevé).

Ensuite, elle met à jour leur hauteur en utilisant les scores de la liste. Elle attribut également une couleur en fonction du score à chaque classe de l'histogramme pour rendre la visualisation plus facile à interpréter.

Enfin, elle utilise une méthode de la classe **BarGraphGenerator** pour ajouter les nouvelles données au graphique.

4.2.2 Architecture finale

Pour obtenir la fonctionnalité de l'affichage dynamique des geste en compétition, nous avons donc intégré ce nouveau package dans l'architecture globale du démonstrateur et d'ensuite lié les prédictions du moteur avec l'affichage dynamique proposé par le package. On peut observer le rendu final à la figure 7 Pour cela, nous avons ajouté en attribut de la classe DemonstrationView une instance BarGraph de la classe BarGraphExample comme décrit sur la figure 14. Cet ajout nous permet d'appeler la méthode **UpdateDataFromAIResult()** directement depuis la méthode UpdateView() de la classe DemonstrationView.

La méthode updateView() récupère donc pour chaque frame la liste de prédictions du moteur de reconnaissance et cette liste est ensuite utilisé en paramètre de la méthode **UpdateDataFromAIResult()**. Elle inclut également une fonction d'attente pour éviter le chevauchement des prédictions. L'utilisation de ce package simplifie donc le processus de l'affichage dynamique dans notre application. Cela nous permet de ne avoir à modifier les classes DemonstrationController et DemonstrationModel comme nous avions prévu dans le rapport de conception. Mise à part l'ajout de ce nouveau package, la structure globale de l'application reste la même.

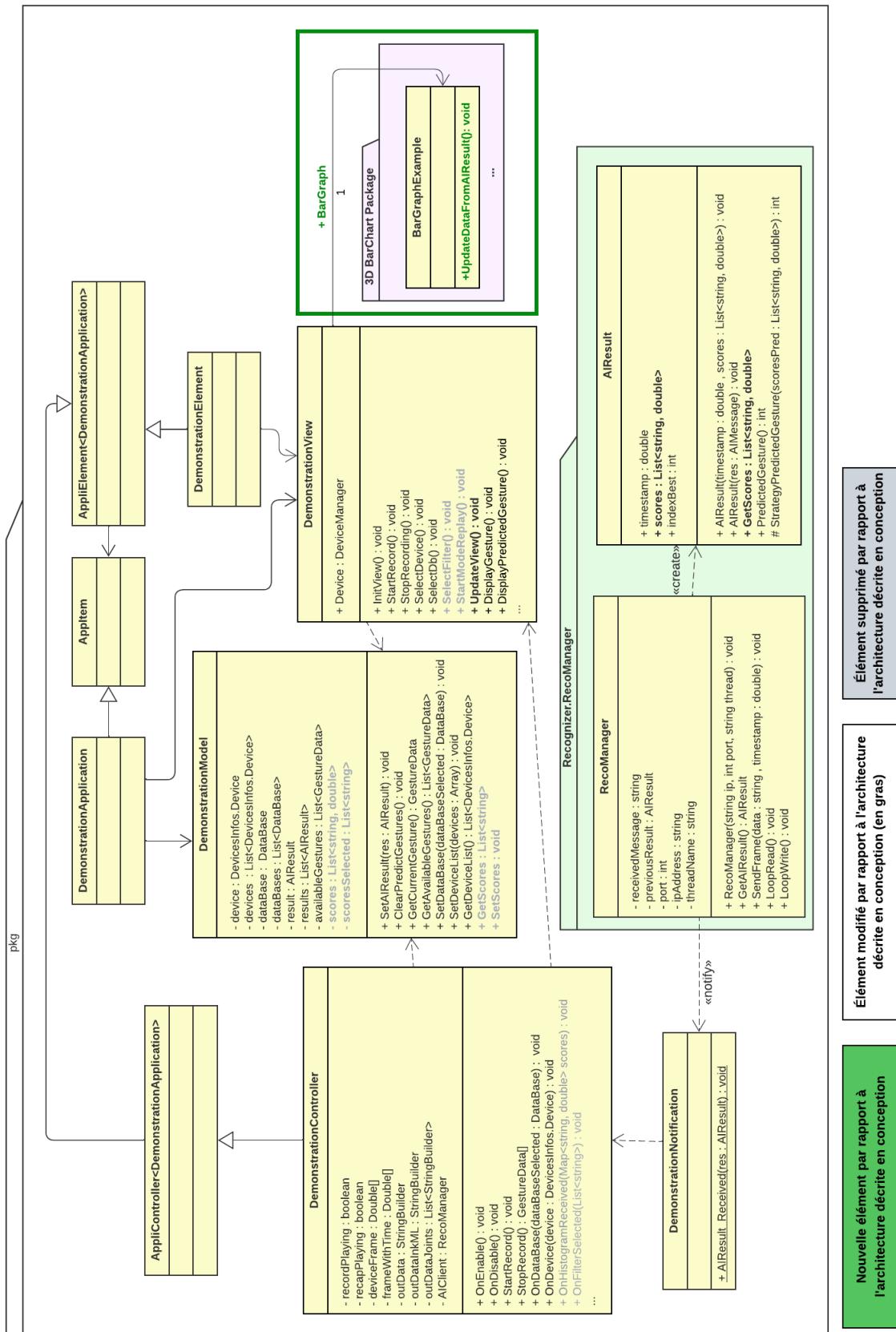


FIGURE 14 – Diagramme UML de l'architecture finale du démonstrateur

5 Bilan de planification

Au début du projet, nous avons organisé sa mise en place en planifiant le temps nécessaire pour réaliser les différentes fonctionnalités et livrables. D'après le rapport de planification [8], nous avons prévu un total de 405 heures de travail avec un effectif de 3 personnes et une charge de travail de 10 heures par semaine. Suite à un changement de calendrier scolaire, la date limite finale de rendu des projets est passé du 11/05 au 14/04, nous amputant ainsi près de 120 heures de travail. Dans cette section, nous allons vous expliquer comment le projet s'est réellement déroulé au sein de l'équipe, en comparaison avec la planification initiale établie en Novembre.

5.1 Rappel du versionnage des applications

Nous avons fait le choix lors de la planification de versionner les différents modules SIM et le démonstrateur et de leur attribuer un niveau de priorité. Ainsi, comme présenté en figure 15, l'application SIM a été divisée en deux versions : la première est une refonte totale de l'application MOP, la seconde est un ajout de fonctionnalité incluant un exercice qui implémente l'IA que nous utilisons. Une dernière version avait initialement été prévu et visait à ajouter un côté modulable à notre application par le biais d'un mode Auteur permettant de modifier une séance et par la possibilité d'ajout d'exercice dans la base de donnée. Cependant cette troisième version a été abandonnée après changement du calendrier scolaire. Ici l'objectif final du projet était d'arriver avant tout à une version 1 tandis que la version 2 permettait d'apporter plus d'interactivité au sein de l'application, mais restait un objectif auxiliaire..

Côté démonstrateur, une principale fonctionnalité était souhaitée : cette fonctionnalité consistait en l'affichage dynamique des résultats de l'IA, et ce d'une manière générique. Pareillement à SIM, une version auxiliaire était initialement souhaitée, et visait à faire apparaître ce même affichage à posteriori, dans un "replay" des gestes effectués, cependant cette version a elle aussi été abandonnée suite au nouvel emploi du temps.

Version	Module	Fonctionnalités	Priorité initial	Priorité final	Réalisation
V1	SIM	Refonte Architecture Application	Recommandé	Recommandé	100%
		Refonte Ergonomie Application			100%
	Démonstrateur	Affichage Dynamique	Souhaité	Auxiliaire	50%
		Généricité			0%
V2	SIM	Ajout d'un exercice implémentant l'IA	Souhaité	Auxiliaire	0%

FIGURE 15 – Tableau prévisionnel des différentes versions du projet, détaillé par module et fonctionnalité.

5.2 Avancée finale du projet

La production finale de notre projet correspond ainsi globalement à 90% de la version 1 initialement prévu (voir figure 15). Si nous avons accumulé un certain retard sur la planification, cela est principalement dû à deux problèmes rencontrés au cours du projet.

En effet, dès le début, nous avons été confrontés à un problème de sous-estimation du nombre d'heures de travail nécessaires au lancement du projet, en particulier pour la refonte de l'architecture de l'application ainsi que pour l'ajout d'un avatar 3D

pour le coach virtuel et la représentation de l'utilisateur. Au total, cela nous a pris 60 heures, alors que nous avions initialement prévu 40 heures pour ces différentes tâche.

Puis, lors de la suite du projet, une deuxième grande difficulté est apparue : Le changement de calendrier scolaire a entraîné une condensation de nos emplois du temps respectif ce qui nous a empêché de tenir les 10h de travail/semaine/étudiant prévu. Nous sommes ainsi descendu à 7h de travail/semaine/étudiant sur plusieurs semaines, freinant ainsi considérablement l'avancée de notre projet.

En fin de compte, nous avons réussi à atteindre l'objectif principal pour l'application SIM en créant une application fonctionnelle et ergonomique. Cependant, en raison des difficultés rencontrées, nous avons dû abandonner la version 2. En plus de cela, le retard accumulé sur la version 1 nous a laissé moins de ressources pour travailler sur le démonstrateur. Malgré tout, nous avons quand-même réussi à créer une première version fonctionnelle du démonstrateur, qui est compatible avec différents capteurs, mais qui n'est pas générique en ce qui concerne les résultats, c'est-à-dire la base de gestes à reconnaître.

Le développement de ce projet à ainsi représenté une charge totale de travail de 241 heures (voir figure 17) répartis à trois personnes sur le second semestre. Plus de la moitié de ce temps était ainsi consacré à des temps de réunion et à la rédaction des différents livrables. Tandis que près de 40% du temps de développement était consacré à de la réflexion/recherche d'information. Cela donne ainsi avec l'ensemble du travail effectué au S7 (voir figure 16) une charge totale de 603 heures sur l'ensemble de l'année

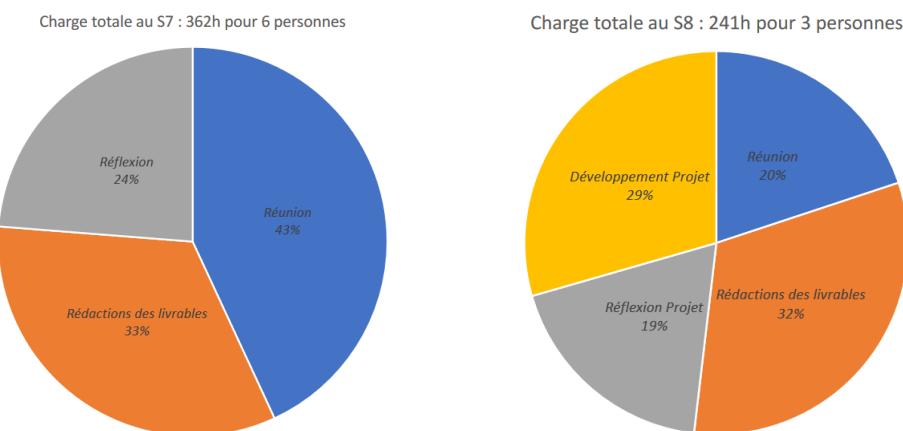


FIGURE 16 – Charge totale de travail au S7 et répartition par type de tâche FIGURE 17 – Charge totale de travail au S8 et répartition par type de tâche

5.3 Bilan des outils utilisés

Tout au long de notre projet, plusieurs outils ont été utilisés afin de communiquer au sein du groupe et pour la gestion des tâches :

- **Teams** : logiciel de gestion de projet et de communication professionnel.
- **Gitlab** : plateforme d'hébergement et de suivi de projet permettant de centraliser les documents.

- **Microsoft Project** : logiciel de gestion de projets qui a permis de planifier et piloter notre projet.
- **Google Drive** : plateforme de stockage et d'édition de fichiers.
- **Overleaf** : Editeur collaboratif LaTeX en ligne.

6 Manuel Utilisateur

Dans ce manuel utilisateur, nous présenterons dans un premier temps l'application Stay In Motion puis le démonstrateur associé au moteur de reconnaissance de gestes 3D. Nous détaillerons le fonctionnement de chacune de ces applications et vous guiderons à travers les étapes nécessaires pour les maîtriser facilement. Ce manuel a pour objectif de vous fournir toutes les informations requises pour une expérience utilisateur optimale et une compréhension approfondie des fonctionnalités offertes par ces deux applications.

6.1 Application Stay in Motion

Lorsque l'application "SIM" est lancée, l'écran d'accueil présenté en figure 18 apparaît.



FIGURE 18 – Menu principal de SIM

Le bouton "Mode utilisateur" permet de renvoyer l'utilisateur vers la séance de sport. En clinquant dessus, la séance commence. Les deux autres boutons "Mode auteur" et "Paramètre" ne sont pas cliquable. Nous les avons créé en prévision des modes auteur et paramètre prévus dans la première version du cahier des charges établi dans le rapport de pré-étude et spécification fonctionnelle (voir figure [9]). Ces deux modes ont finalement été abandonné lors de la phase de planification. Nous les avons laissé pour les futurs groupes.

Une fois la séance lancée, une série d'exercice se lance. Chaque exercice est de la forme de l'exercice "Squat" présenté en figure 4.

Une fois tous les exercices terminés, un retour utilisateur est fait, ce retour est présenté en figure 19.



FIGURE 19 – Récapitulatif de la séance

6.2 Démonstrateur associé au moteur de reconnaissance de gestes 3D

Tout d'abord avant de commencer la séance de démonstration, l'utilisateur doit sélectionner le capteur qu'il souhaite utilisé pendant la démonstration parmi les capteurs compatibles avec l'application décrit en figure 20 (Kinect V1, Kinect V2, LeapMotion, Kinect Azure ou SimulatorKinect) .

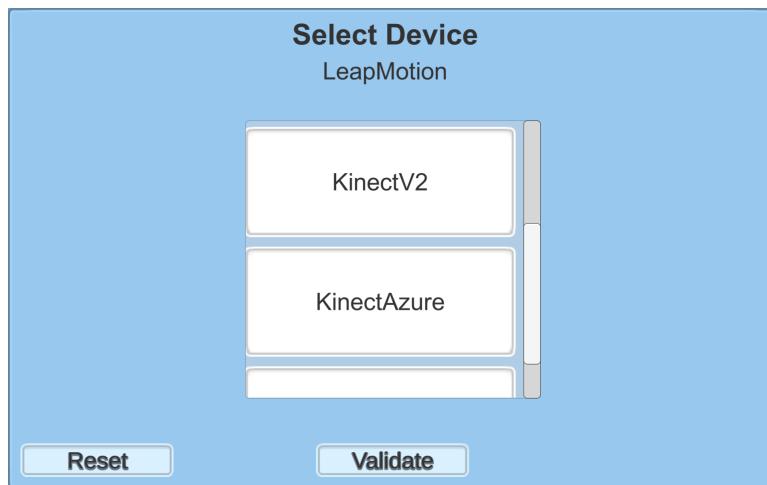


FIGURE 20 – Menu permettant la sélection du capteur utilisé lors de la démonstration

Après avoir sélectionné le capteur, les bases de données associées à ce type de capteur s'affichent (voir figure 21). L'utilisateur doit donc choisir la base de données de gestes sur laquelle il veut tester le moteur de reconnaissance. Après avoir valider la sélection du capteur et de la base de données associée, l'utilisateur arrive sur la

scène principale de la session de démonstration décrite en figure 22. Pour lancer une séance, l'utilisateur doit appuyer sur le bouton « Start ». Il est important de noter que l'application de démonstration fonctionne de pair avec un serveur qui gère les moteurs de reconnaissance. La détection de gestes ne fonctionnera pas si le serveur n'est pas connecté au module. On retrouve en haut de cette scène la liste des gestes reconnus par le moteur de reconnaissance que l'on a choisi. Ils sont décrits par une image qui représente le squelette d'un utilisateur réalisant ce geste ainsi qu'une légende. Chaque geste est connecté à une partie de l'histogramme présent sur la gauche de la scène. Le numéro présent au niveau de la légende d'un geste correspond au numéro sur l'axe X de l'histogramme. L'axe Y correspond au niveau de confiance prédits par l'IA. On peut aussi voir le squelette capturé par le capteur en direct à droite de la fenêtre.

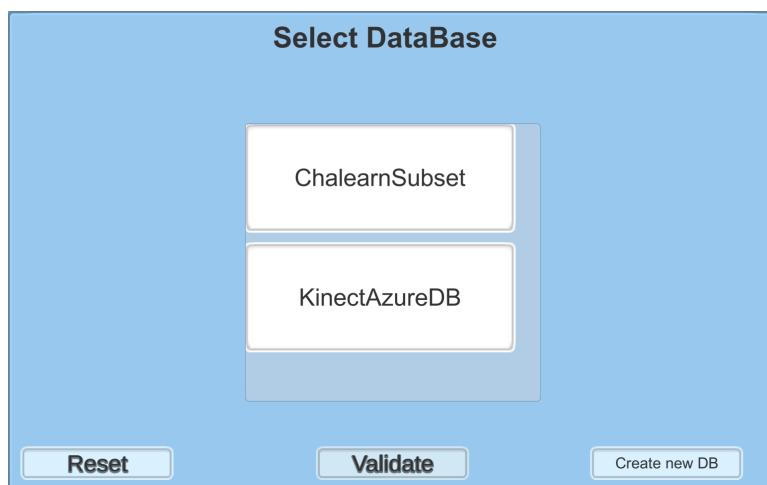


FIGURE 21 – Menu permettant la sélection de la base de données utilisé

Après avoir lancé la reconnaissance de gestes, les mouvements de l'utilisateur sont analysés par le moteur de reconnaissance et les résultats obtenus sont affichés en temps réel de façon qualitative à travers l'histogramme. Les barres de couleur verte correspondent aux geste prédits avec un niveau de confiance supérieur à 0.75 , celles de couleur jaune correspondent aux geste prédits avec un niveau de confiance supérieur à 0.3 et les barres restantes sont en gris.

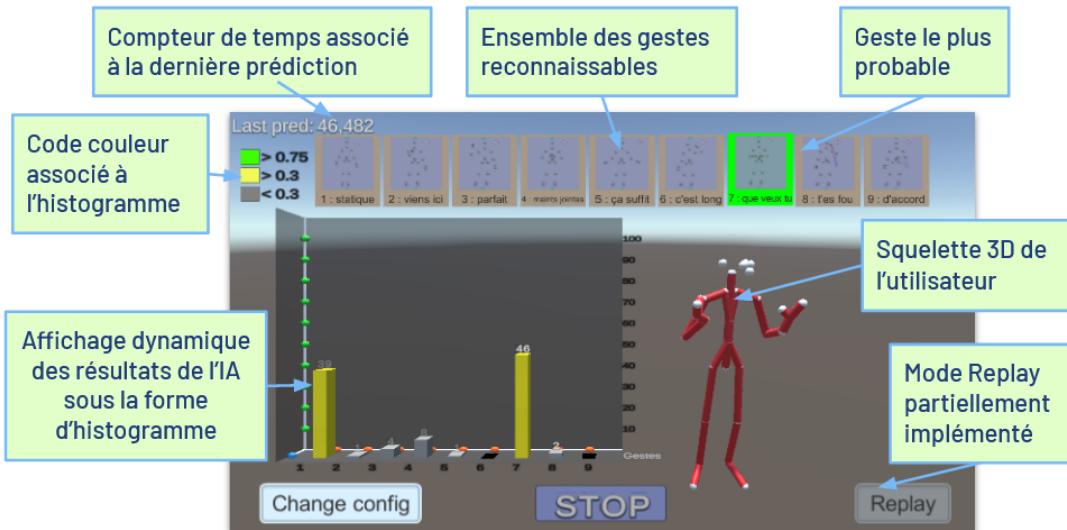


FIGURE 22 – Affichage dynamique selon les prédictions et la confiance du moteur de reconnaissance

Sur la figure 22, on peut observer l'hésitation dans les prédictions de l'IA. les gestes « Que veux-tu » et « statique » possèdent une confiance presque égale. Sur la figure 23, le moteur a fait son choix définitif avec un confiance élevé pour un unique geste.

Pour chaque nouvelle prédiction du moteur, le geste le plus probable s'avance devant les autres gestes et se colore en vert fluo.

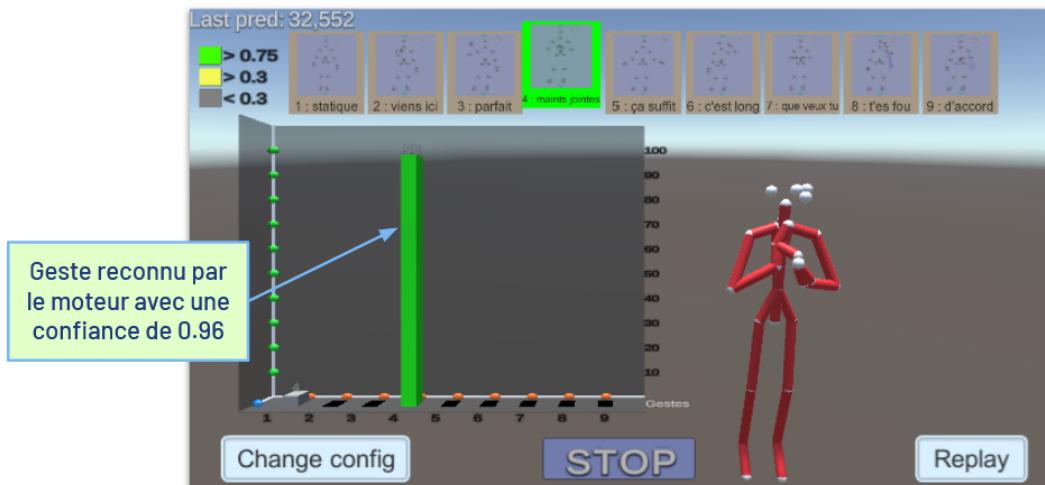


FIGURE 23 – Choix définitif du moteur de reconnaissance

Pour mettre en pause ou mettre fin à la séance de démonstration, l'utilisateur doit ensuite appuyer sur le bouton « Stop ».

7 Manuel d'installation

Ce manuel a pour objectif de vous guider pas à pas dans l'installation du projet SIM, de son serveur et de son environnement requis pour l'exécuter.

7.1 Pré-requis pour l'installation

Avant de commencer l'installation, assurez-vous que votre système répond aux pré-requis suivants :

- Un GPU NVIDIA compatible CUDA (voir <https://developer.nvidia.com/cuda-gpus>);
- Windows 10 ou 11 ;
- Suffisamment d'espace disque disponible (environ 16 Go, dont environ 11 peuvent être installés sur un disque secondaire).

7.2 Installation de l'environnement

7.2.1 Pilotes et SDK

GPU Si vous avez GeForce Experience (le gestionnaire de pilotes de NVIDIA) installé, utiliser-le pour mettre à jour les pilotes de votre GPU, sinon télécharger et installer GeForce Experience ici : <https://www.nvidia.com/en-us/geforce/geforce-experience/> ou télécharger et installer le dernier pilote pour votre GPU spécifique ici : <https://www.nvidia.com/Download/index.aspx>

Azure Kinect Télécharger et installer la dernière version du SDK Azure Kinect (nous avons utilisé le SDK Azure Kinect 1.4.1.exe) ici : <https://github.com/microsoft/Azure-Kinect-Sensor-SDK/blob/develop/docs/usage.md>. Puis installer le SDK Azure Kinect Body Tracking (nous avons utilisé 1.1.2, avec l'installateur msi) ici : <https://learn.microsoft.com/en-gb/azure/kinect-dk/body-sdk-download>

Python Si nécessaire télécharger et installer la dernière version de Python 3 (nous avons utilisé Python 3.10.7) ici : <https://www.python.org/downloads/windows/>

CUDA Télécharger et installer la dernière version de cuda (nous avons utilisé CUDA Toolkit 11.7.1) ici : <https://developer.nvidia.com/cuda-toolkit-archive>

CUDNN Télécharger la dernière version de cuDNN (nous avons utilisé cuDNN v8.5.0) ici : <https://developer.nvidia.com/cudnn>. Vous aurez besoin d'un compte de programme de développement NVIDIA (cela peut être créé gratuitement). L'installation de cuDNN est très manuelle, vous pouvez suivre ce guide : <https://docs.nvidia.com/deeplearning/cudnn/install-guide/index.html#install-windows>. Le guide vous demandera d'ajouter des chemins de répertoire à la variable PATH, voici un guide sur comment faire ceci : [https://learn.microsoft.com/en-us/previous-versions/office/developer/sharepoint-2010/ee537574\(v=office.14\)](https://learn.microsoft.com/en-us/previous-versions/office/developer/sharepoint-2010/ee537574(v=office.14)). Les étapes à suivre sont les étapes 2.1.3 à 2.3. Vous n'avez pas besoin de la partie "Add cuDNN to your Visual Studio project". N'oubliez pas de redémarrer votre ordinateur après avoir effectué cette installation afin que la modification du PATH prenne effet.

Git Si nécessaire, télécharger et installer la dernière version ici : <https://git-scm.com/download/win>

Unity Télécharger et installer la dernière version de Unity Hub (le gestionnaire de versions de Unity) : <https://unity3d.com/get-unity/download>. Télécharger la version 2021.3.7f1 de Unity ici : <https://unity3d.com/get-unity/download/archive>. Essayer d'utiliser la dernière version, mais Unity change beaucoup au fil du temps et casse souvent la compatibilité. Si votre ordinateur en possède un disque secondaire, vous pouvez l'installer là-dessus, car il prend environ 3,3 Go.

7.3 Installation des différentes parties du projet

7.3.1 Serveur

Utiliser git pour cloner le dépôt du serveur ici : https://gitlab.inria.fr/wmocaer/jcr-rnn_ownimplementation

7.3.2 Client (Projet Unity)

Utiliser git pour cloner le projet client ici : https://gitlab.insa-rennes.fr/William.Mocaer/r3g_mop_22_23

8 Conclusion

Au terme de ce projet, nous avons réalisé une application interactive s'inscrivant dans la lutte contre la sédentarité des jeunes et un démonstrateur permettant de visualiser la compétition entre différentes classes de gestes durant la phase de reconnaissance d'un module de reconnaissance de geste.

La phase de pré-étude et de spécifications fonctionnelles nous a permis de comprendre le sujet du projet ainsi que les besoins du client. Nous avons analysé le cahier des charges, puis choisi les technologies les plus appropriées pour permettre la réalisation de la suite logicielle. Nous avons ensuite défini les fonctions correspondant à chaque besoin du client. Puis, nous avons attribué à chaque fonction un ordre de réalisation et évalué les heures de développement nécessaires. Nous avons ensuite planifié les différentes tâches fonctionnelles pour réaliser le projet et estimé la charge totale de travail nécessaire, en accord avec les ressources disponibles.

Avant de débuter le développement du projet, la phase de conception nous a permis d'obtenir une bonne organisation du code du projet. Nous avons défini des diagrammes de classes et de séquences de chaque module, nous permettant d'aborder sainement la phase de développement. Enfin, cette phase de développement nous a permis d'appliquer concrètement le travail préparé en amont, avec quelques ajustements.

Ce projet nous a donc appris à travailler au sein d'une équipe de taille conséquente, à définir et à planifier le travail en prenant en compte les ressources disponibles et les freins éventuels. Nous avons également amélioré nos compétences techniques et notre capacité à travailler en groupe au cours des différentes phases du projet.

Ce projet a été pédagogique pour toute notre équipe. Nous avons pu mesurer l'ampleur du travail nécessaire pour réaliser un projet de longue durée. Nous avons développé des compétences techniques mais aussi humaines puisqu'il était nécessaire de travailler en groupe et de se répartir les tâches. Grâce à la rédaction des rapports,

nous avons appris à clarifier nos idées et à les transmettre, domaine dans lequel nous nous sommes grandement améliorés au cours de l'année.

Concernant le résultat final du projet, nous sommes globalement satisfaits du résultat obtenu, compte tenu de la charge de travail. Nous sommes heureux d'avoir travaillé tous ensemble et de fournir nos applications au client.

Références

- [1] ANSES. *Inactivité physique et sédentarité chez les jeunes : l'Anses alerte les pouvoirs publics.* 2020. URL : <https://www.anses.fr/fr/content/inactivit%C3%A9-physique-et-s%C3%A9dentarit%C3%A9-chez-les-jeunes-l'anses-alerte-les-pouvoirs-publics>. (consulté : 4 Novembre 2022).
- [2] Mehdi AOUICHA et al. *Rapport de Conception - Move on Progress.* Nov. 2021. URL : <https://insarennesfr.sharepoint.com/sites/Projet4info22-23/Documents%20partages/Anciens%20Projets/RapportConception.pdf>.
- [3] Paul BERTHELOT et al. *Rapport de Conception - Recognition of 3D Gesture.* 2021. URL : https://insarennesfr.sharepoint.com/sites/Projet4info22-23/Documents%20partages/Anciens%20Projets/Concep_R3G_BabelMarie.pdf.
- [4] Said Yacine BOULAHIA. « Reconnaissance en-ligne d'actions 3D par l'analyse des trajectoires du squelette humain ». Theses. INSA de Rennes, juill. 2018. URL : <https://theses.hal.science/tel-01857262>.
- [5] Antoine COURMONT. *Baromètre du numérique 2021 – Les chiffres des usages numériques en France.* Juill. 2021. URL : <https://linc.cnil.fr/fr/barometre-du-numerique-2021-les-chiffres-des-usages-numeriques-en-france>. (consulté : 4 Novembre 2022).
- [6] Aydinh EMRE et al. *Rapport de Conception - DERG3D.* 2018. URL : https://insarennesfr.sharepoint.com/sites/Projet4info22-23/Documents%20partages/Anciens%20Projets/_4info__Projet___DERG3D___Conception.pdf.
- [7] Cadot FIRMIN, Gesbert TEDDY et Le Bizec LÉANDRE. *Rapport de Conception - Stay In Motion.* 2023. URL : https://insarennesfr.sharepoint.com/:b/r/sites/Projet4info22-23/Documents%20partages/General/Stay_In_Motion_Conception_V1.pdf.
- [8] Cadot FIRMIN et al. *Rapport de Planification - Stay In Motion.* Nov. 2022. URL : https://insarennesfr.sharepoint.com/sites/Projet4info22-23/Documents%20partages/Anciens%20Projets/Planif_StayInMotion_ThierryRoger.pdf.
- [9] Cadot FIRMIN et al. *Rapport de Pré-étude et de spécification fonctionnelle - Stay In Motion.* Nov. 2022. URL : https://insarennesfr.sharepoint.com/sites/Projet4info22-23/Documents%20partages/Anciens%20Projets/Stay_In_MotionSpec.pdf.

- [10] William MOCAËR, Eric ANQUETIL et Richard KULPA. « Réseau Convolutif Spatio-Temporel 3D pour la Reconnaissance Précoce de Gestes Manuscrits Non-Segmentés ». In : *RFIAP 2022 - Congrès Reconnaissance des Formes, Image, Apprentissage et Perception*. Vannes, France, juill. 2022, p. 1-9. URL : <https://hal.archives-ouvertes.fr/hal-03682604>.