RAPPORT TECHNIQUE DE PROJET DE FIN D'ÉTUDES
PRÉSENTÉ À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
DANS LE CADRE DU COURS GTI795 PROJET DE FIN D'ÉTUDES EN TI

DÉTECTION DE MODÈLE HUMAIN POUR APPLICATION DE RÉALITÉ AUGUMENTÉE

Vincent BOITEAU-ROBERT
BOIV14029409
Marc-Antoine HÉBERT
HEBM14019401
Julien LEMONDE
LEMJ20059208
Alexandre MALO
MALA0702940

DÉPARTEMENT DE GÉNIE LOGICIEL ET TI

Professeur-superviseur
Carlos Vazquez

MONTRÉAL, 25 AVRIL 2018
HIVER 2018
UNIVERSITÉ DU QUÉBEC

REMERCIEMENT

Cela fait maintenant 6 ans que 3 des 4 membres de notre équipe travaillons ensemble dans un but professionnellement commun. Nous avons vécu nos premiers travaux d'équipes lors de nos études collégiales et nous sommes très fiers de pouvoir dire que nous terminons une étape de nos vies en même temps. Nous avons rencontré le quatrième membre de cette fabuleuse équipe lors notre tout premier cours à l'École de Technologie Supérieure.

S'étant suivis de session en session tout au long de notre Baccalauréat en génie des technologies de l'information, nous voulions absolument pouvoir concevoir notre projet de fin d'études entre nous. C'est grâce à l'aide de madame Lucie Caron et de monsieur Jean-Marc Robert nous avons pu réaliser ce projet de grande envergure entouré de coéquipier dont la synergie est plus que remarquable.

Nos plus grands remerciements iront tout de même à monsieur Carlos Vasquez pour nous avoir supportés tout au long du projet. Afin de pouvoir effectuer le projet de fin d'études avec notre équipe prédéterminée, nous avions besoin de trouver un professeur du département de TI qui aurait été intéressé à nous supporter lors de la réalisation de notre projet. Notre premier choix a été de contacter rapidement monsieur Vasquez, car son cours d'application de technique numérique nous avait bien intéressés tous. Le fait d'avoir choisi monsieur Carlos Vasquez s'est vu être un très bon choix comme professeur-superviseur de PFE, car il nous a très bien supportés tout au long des étapes de conception de ce projet.

RÉSUMÉ

DÉTECTION DE MODÈLE HUMAIN POUR APPLICATION DE RÉALITÉ AUGUMENTÉE

Vincent BOITEAU-ROBERT
BOIV14029409
Marc-Antoine HÉBERT
HEBM14019401
Julien LEMONDE
LEMJ20059208
Alexandre MALO
MALA0702940

Le projet consite à détecter un être humain à l'aide d'une caméra 3D afin d'être en mesure d'aller récupérer la texture et la profondeur. Pour ensuite convertir l'humain détecté en modèle 3D. Une fois le modèle bâti, le projet consite aussi à intégrer le modèle humain dans une application de réalité augmenté. Celle-ci repose sur framework ARKit de Apple. Le tout, encapsuler dans une seule et même application unique.

Ce document contient une introduction sur le sujet ainsi qu'une revue de la documentation utilisé pour la réalisation de ce projet. De plus, la méthodologie de travail utilisé et le processus de conception y sont documentés. Le tout se termine avec une discussion et une conclusion.

Table des matières

Liste des tableaux

Table des figures

LISTE DES ABBRÉVIATIONS

API - Application Programming Interface ou en français Interface de programmation

AR - Augmented Reality ou en français Réalité Augmentée

AV - audio-vidéo

ÉTS - École de Technologie Supérieure

PFE - PRojet de fin d'études

SDK - Software Development Kit ou en français Trousse de développement logiciel

ZIP - Format de fichier compressé

INTRODUCTION

La performance des caméras des appareils mobiles est en constante croissance depuis quelques années. Ce qui permet d'envisager l'usage de ces appareils dans des nouveaux domaines d'activités. Un de ces domaines est la réalité augmentée, où la superposition d'éléments virtuels sur une capture de l'environnement réel en temps réel. Une démocratisation du développement d'application utilisant cette technologie s'est faite au cours de l'année précédente, 2017. En effet, la nouvelle interface de programmation applicative ARKit conçue par Apple offre un environnement de développement simplifié au développeur d'applications (Statt, N., 2017). Toutefois, ces technologies permettent l'interaction avec des modèles générés de façon synthétique. Un défi qui n'a pas encore de solution reconnue est l'usage de modèles provenant de l'environnement de l'utilisateur dans la réalité augmentée. Certaines technologies offrent des pistes de solution, mais leur taux d'adoption et le niveau de compatibilité avec les autres outils de l'environnement de développement Apple est limité.

Une des technologies les plus intéressantes est le Structure Sensor(Molitch-Hou, M., 2016). Celle-ci consiste en un périphérique qui doit être attaché à l'appareil. Le périphérique ajoute une caméra et un senseur 3D. Avec l'usage de la caméra de l'appareil, celui de l'appareil on peut avoir une meilleure compréhension de l'environnement 3D. L'outil vient avec une trousse de développement logiciel permettant de prendre les coordonnées d'un objet sous quatre dimensions, x, y, z et la couleur. Toutefois cette solution n'est pas très connue et a un nombre de ressources limitées.

Les deux technologies décrites plus haut apportent chacune des possibilités très intéressantes pour le développement d'application de réalité augmentée. L'usage des fonctionnalités d'une dans l'autre serait très intéressant. Toutefois au moment où l'équipe a débuté le projet,

il n'y a pas d'interface une telle combinaison. Les deux interfaces utilisent la caméra, par contre chacun utilise un utilitaire différent pour le contrôle de celle-ci. Les deux utilisent des modèles numériques 3D, mais chacun a choisi une norme différente. En effet, le Structure Sensor permet l'extraction en Modèle I/O. Alors que le ARKit utilise les modèles de type scène qui est un format propriétaire à Apple. Le principal défi du projet consistera à la combinaison de plusieurs interfaces de programmation pour créer un outil fonctionnel.

Comme décrit dans la section précédente, le principal objectif du projet est la création d'une interface entre deux technologies mobiles, soit un périphérique d'acquisition de modèle 3D, le Structure Sensor et la trousse de développement de réalité augmentée d'Apple. Autrement dit, l'équipe s'attend en fin de session d'avoir produit une application mobile permettant la prise de modèle à l'aide du senseur. L'application permettra d'ajouter un modèle extrait sur une surface plane. L'application permet cette chaîne d'action avec une expérience utilisateur agréable et naturelle à l'utilisateur.

L'application décrite ci-haut représente les attentes pessimistes par rapport aux résultats finaux. Toutefois, une telle application apporte un éventail de possibilités sur la manipulation de l'environnement augmentée. Par exemple, une des retombées envisageables et commercialisables est la numérisation de catalogue de magasin comme Ikea. L'usager peut donc ajouter un item dans une pièce de sa maison. Si plusieurs couleurs sont disponibles pour un modèle, chacune des variantes sera disponible à la modélisation.

Une autre ouverture technologique, qu'une telle application apporterait, est dans un contexte d'animation 3D. En effet, la création d'un modèle permettrait l'ajout d'un squelette dans celui-ci. Une fois le squelette créé l'animation du modèle pourrait être fait. Cette possibilité est particulièrement intéressante pour l'équipe responsable du projet. Dans l'éventualité où la vision de base du projet sera complétée avant la fin de la session d'Hiver 2018, l'équipe compte se concentrer sur la création de ce squelette et possiblement la manipulation de celui-ci. La création de positionnement clé des nœuds du squelette pourra être enregistrée afin de permettre l'animation. L'animation libre est un trop grand défi étant donné les contraintes de temps.

L'objectif principal, une application sur tablette Apple permettant de scanner un objet de l'environnement réel à l'aide du Structure Sensor et le présenter sur une surface plane en réalité augmentée à l'aide de l'ARKit, semble réaliste pour l'équipe de développement. Toutefois, le défi reste important étant donné le manque de cohérence entre les différentes technologies. Le manque de documentation et de support pour le Structure Sensor risque d'amener un ensemble de défi supplémentaire.

Par contre, l'équipe de développement considère le temps disponible à la résolution d'un tel enjeu adéquat. L'expérience en recherche d'information et apprentissage autonome acquise au cours des formations et stages des membres de l'équipe laisse envisager un succès.

Pour ce qui est en la complétion des objectifs accessoire, les limitations de temps rendent l'estimation de succès plus difficile. Par contre, il est clair que la réalisation de ceux est apporterait un plus considérable à l'expérience de chacun des membres. La motivation de l'équipe face à l'enjeu améliore les probabilités de terminer ces objectifs. Ce rapport consiste donc en un suivi des étapes qui permettront de valider ou invalider les hypothèses décrites si haut. Afin de résoudre un tel défi, l'équipe a dû se fier sur les documentations disponibles en lien avec les technologies ciblées. Un résumé de celles qui ont été utiles à la résolution du problème sera présenté.

Par la suite, une présentation de la méthodologie utilisée par l'équipe sera décrite. Les étapes ayant permis la conception de la solution seront par la suite présentées. Une analyse sur le produit obtenu sera explorée. Afin de clore le projet, les possibilités que la solution apportée à un groupe voulant poursuivre ou implémenter un projet semblable seront expliquées. L'équipe fera ces recommandations en fonction de l'expérience acquise lors du développement.

REVUE DE LA DOCUMENTATION

Afin de réaliser le projet, l'équipe de développement a dû faire des recherches préliminaires. Les technologies utilisées n'étaient pas maitrisées par l'équipe au démarrage du projet. Donc, les recueils de documentation et les projets exemples présents sur internet ont été une bonne source d'information.

Cette section présentera donc ces ressources pour les grandes parties du projet, soit le périphérique Structure Sensor, la trousse de développement logiciel de réalité augmentée d'Apple et la gestion de projet.

Structure Sensor

Occipital, la compagnie propriétaire du périphérique, publie régulièrement une version de leur trousse de développement logiciel exposant des interfaces de programmation. La dernière version publiée par la compagnie est la version 0.7.1 en février 2018. (Occipital, 2018) La version du SDK étant inférieure à 1.0.0, on comprend que la première version complète est encore en développement.

Dans l'état actuel, le SDK fournit quand même une liste d'interface disponible dans deux fichiers, Structure et StructureSLAM. Le premier est composé des différentes interfaces de contrôle du périphérique. Alors que le deuxième est un regroupement des classes permettant le contrôle des mailles en position tridimensionnelles et les textures de ceux-ci.

Ces interfaces seront la base de la conception du module de prise de modèle grâce au périphérique. Les tâches de chacune des interfaces ont été identifiées lors de la lecture du code

des entêtes des interfaces en Objective-C. L'équipe a produit un diagramme de classe afin d'améliorer la compréhension des responsabilités de chacune des interfaces et les relations entre celles-ci. Celui-ci est présenté à la page suivante.

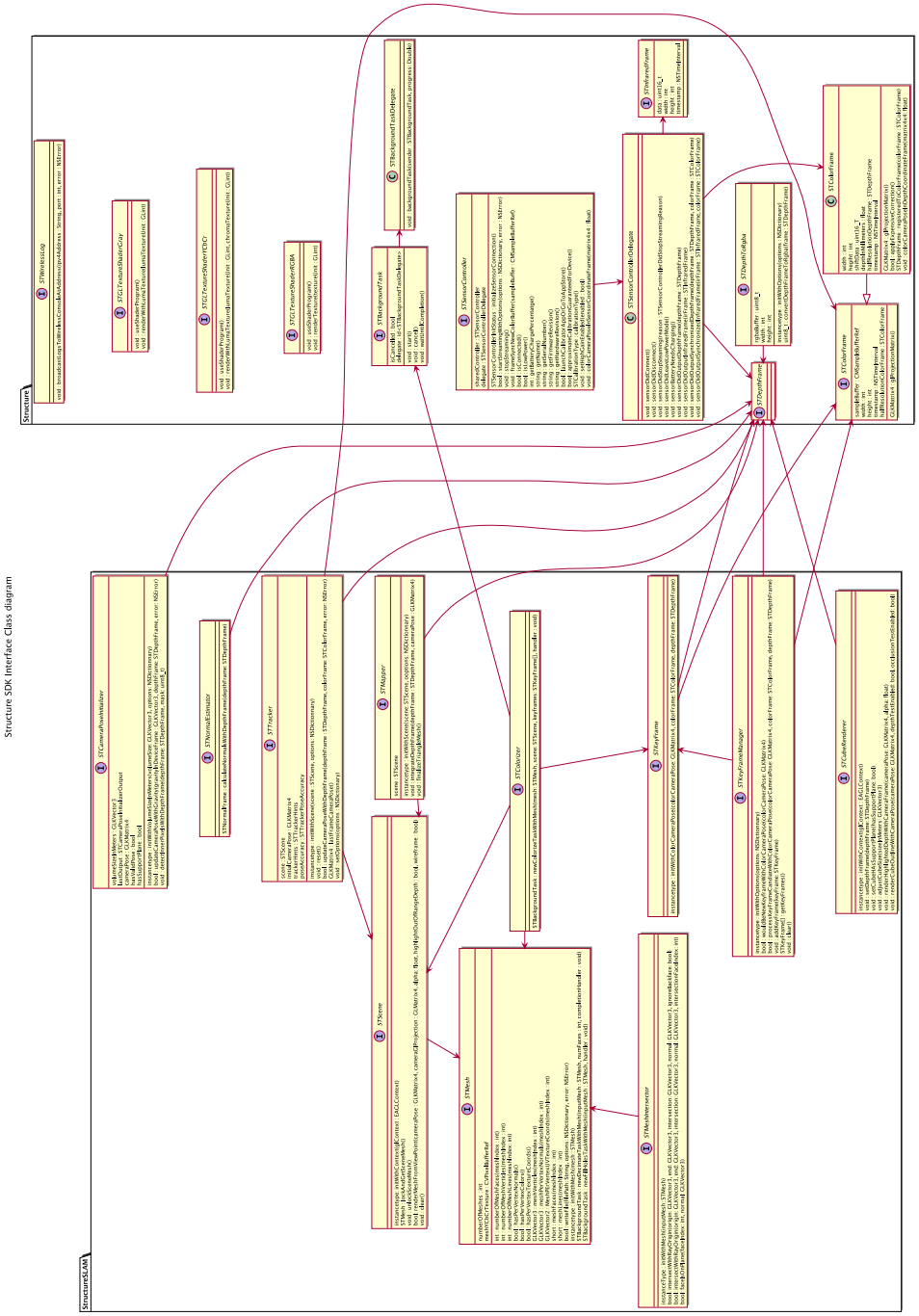

FIGURE 1 – Diagramme de classes du Structure Sensor SDK

Dans le diagramme, on constate que plusieurs patrons de conception ont été utilisés et exposés dans l'interface. La classe STSensorControllerDelegate servira de point d'ancrage entre notre application et le Structure Sensor. Cette interface permet donc de réduire les communications ou les recherches d'informations dans la logique d'affaires. La plupart de l'usage du SDK dans l'application devra être fait par cette classe.(Best-Practice-Software-Engineering, 2013)

Le Structure Sensor utilise les DepthFrame et ColorFrame pour comprendre l'environnement dans lequel il est utilisé. Ces concepts ont été introduits par Kinect un des premiers Senseurs 3D ayant connu un succès commercial étant donné son intégration dans l'environnement Xbox 360 de Microsoft. (Microsoft, 2014)

Le Structure SDK contient aussi plusieurs exemples d'application afin de guider les développeurs dans le développement de leurs applications. Le plus intéressant pour l'objectif du projet est le Scanneur. Celui-ci permet de convertir un objet dans l'espace de test en un modèle 3D et l'envoyer par courriel. D'autres fonctions incluses dans cet exemple sont la colorisation du modèle et les différentes options de capture. L'espace de capture est aussi indiqué par un cube et un plan utilisant les couleurs pour indiquer les objets qui peuvent être capturés.

Les principales sections du SDK utilisé dans cet exemple sont le STStructureSensorDelegate pour le contrôle du périphérique. Les STDepthFrames et les STColorFrames ou en français les grilles de profondeurs et les grilles de couleurs permettant de créer le modèle STMesh. Une fois le modèle scanner en maille celui-ci est présenté dans le MeshViewController grâce au rendu de maille, STMeshRenderer. Le langage de programmation utilisé dans la globalité du SDK est l'Objective-C.

Dans le but d'utiliser des technologies qui sont à jour avec les pratiques de l'industrie, l'équipe de développement souhaite utiliser le nouveau langage de programmation Swift. Celui-ci est le nouveau langage proposé par Apple pour le développement mobile. Le langage est déjà à sa quatrième version majeure et a pour but de simplifier la syntaxe qui était auparavant utilisée soit l'Objective-C.(Kremenek, T., 2017) Un membre de la communauté de développeur Structure a produit un port de l'exemple de Scanneur en Swift 2. L'équipe devra donc faire

les changements nécessaires à cette version pour la compiler avec la version 4.(Worley, C., 2016)

L'exemple de scanneur qui sera utilisé comme base du module de scan utilise plusieurs autres APIs par Apple. Ces APIs sont documentées sur le site de Apple dans la section pour développeurs. (Apple, 2018c) L'équipe utilisera donc cette ressource en ligne pour accomplir la tâche.

Apple ARKit

En juin 2017, Apple, une compagnie informatique californienne a lancé la 11e version de son système d'exploitation pour les développeurs. Cette nouvelle version apporte aux utilisateurs plusieurs nouvelles fonctionnalités, dont un gestionnaire de fichiers, appuyer fort sur une icône pour avoir plus d'options et aussi la possibilité d'apporter la réalité augmentée aux appareils mobiles. Il s'agit de la naissance de la librairie ARKit de Apple. (Clover, 2018)

Apple a créé l'ARKit en voyant le marché émergent des jeux en réalité augmentée depuis les dernières années. Le problème avec ces jeux c'est qu'il faut avoir un ordinateur très performant pour pouvoir y jouer. En apportant de la réalité augmentée aux appareils mobiles, Apple rend plus abordable ce genre de jeux et offre de belles possibilités aux développeurs.

Cette librairie utilise l'algorithme SLAM (Simultaneous Localization and Mapping) afin de se retrouver dans l'espace. Cet algorithme est utilisé dans les robots ou véhicules autonomes. Il sert à créer une carte virtuelle de l'environnement réel. Une fois cette carte générée, l'algorithme la garde à jour et garde le positionnement de l'appareil dans l'environnement. C'est grâce à cet algorithme qu'il est possible de donner des coordonnées cartographiques à un modèle généré par ordinateur et de l'apercevoir dans l'environnement réel. (Thrun, S. & Leonard, J. J., 2008)

De plus, l'apprentissage de l'ARKit se fait très bien, le site d'Apple possède une grande quantité de documentation sur chaque fonction provenant de la librairie et il existe une grande variété d'exemples pour comprendre l'utilisation. Cette librairie est disponible dans

le nouveau langage Swift d'Apple et dans l'ancien qui est l'objective-c. L'équipe a choisi d'utiliser la version la plus récente de Swift pour éviter des problèmes discontinuités.

MÉTHODOLOGIE DE TRAVAIL

La démarche entreprise par l'équipe de développement pour compléter le projet a été basée sur la combinaison des acquis obtenus dans les projets résolus en stage et au cours du processus universitaire. Toutefois la nature du projet de fin d'études comprend des contraintes et conditions qui ont dû être prises en compte. L'équipe a choisi de travailler sous formes Agile. Cette méthodologie rend le projet plus flexible qu'une méthodologie classique.

Agile

Afin de pouvoir mener à bien le projet et d'assurer la possibilité de maintenir une flexibilité dans le projet, le choix de la méthodologie Agile s'est avéré évident. En effet, celle-ci prône 4 valeurs fondamentales (Alliance, A., 2001) :

- Individus et interactions plutôt que processus et outils
- Fonctionnalités opérationnelles plutôt que documentation exhaustive
- Collaboration avec le client plutôt que contractualisation des relations
- Acceptation du changement plutôt que conformité aux plans

Ces valeurs étaient primordiales pour l'équipe. De plus, le principe d'agile consiste en découper le projet en plusieurs étapes appelées "Sprint". Chacun de ces *sprints* doit aboutir à la création d'un élément fonctionnel du projet. Le projet a été divisé en quatre *sprints* distincts et où les *sprints* comportaient des objectifs, réalisable dans un délai de 3 semaines étaient planifiées. À la fin du *sprint*, une rétrospective était effectuée afin d'évaluer les accomplissements réalisés pour cette étape. La suite des choses est planifiée en fonction des résultats observés pendant la rétrospective.

En procédant ainsi, il est possible d'ajuster les objectifs du projet et des *sprints* en fonction de l'avancement en cours.]section*Outils de suivi

Gestion du code

Tout le code pour le projet, que ce soit la base fournit par le SDK de Structure Sensor ou l'application de réalité augmentée basé sur le Framework ARKit d'Apple, celui-ci était hébergé sur le site de gestion de code *GitHub*. En effet, un outil comme *GitHub* permet de suivre toutes les modifications apportées à chaque ligne de code et de retracer les auteurs des changements à travers le temps. De plus, il permet aussi d'ouvrir des branches afin de développer de nouvelles fonctionnalités en parallèle avec une seconde branche.

Une autre possibilité très intéressante est le fait que le code est public et ouvert à tous, donc un autre développeur peut récupérer l'avancement actuel et continuer le développement puis de proposer d'intégrer ses changements.

Gestion des tâches

Afin de pouvoir faire une bonne gestion du suivi des tâches, un outil de *GitHub* a été utilisé. En effet, des issues ont été créées pour chacune des tâches à accomplir. Des catégories ont aussi été définies afin d'y classer chacune des tâches. Par exemple, toutes les tâches à accomplir lors du *sprint #1* ont été placées dans cette catégorie. De plus, l'outil permet d'assigner chacune des tâches à une ou des personnes afin que chacun sache ce qu'il doit accomplir et d'en connaître les détails.

Une autre fonction pratique est le fait qu'il est possible de lier des sections de code avec la tâche associée afin d'obtenir une meilleure vue globale de ce qui a été accompli.

Communication

La communication dans ce type de projet est primordiale et afin de rester en constante communication, l'outil de communication le plus populaire du marché, c'est-à-dire *Slack* a été choisi et utilisé tout au long du projet. Il permettait à chacun des membres de l'équipe de communiquer et de suivre les communications concernant le projet. De plus, cet outil permettait de créer des canaux de discussion séparés lors de discussion spécifique concernant un sujet en particulier.

Aussi, le professeur-superviseur possédait un compte sur le *Slack* du projet, ce qui lui permettait de suivre l'avancement du projet et offrait la possibilité à l'équipe et à ses membres de communiquer facilement avec lui.

PLANIFICATION

Mise en place

Étant donné que la totalité des membres de notre équipe avait de l'expérience dans le développement applicatif par la méthodologie Agile, nous sommes lancés dans ce projet avec cette façon de faire. Agile étant une méthodologie plutôt permissive dans son exécution, nous avons pu conserver seulement certaines parties de ce mode de travail. En effet, dès les premières rencontres d'équipe pour le choix du sujet du PFE, nous avions décidé que notre projet allait comporter 4 itérations avec présentation au client des avancements. Ici, le client a été remplacé par le professeur supervisant le projet. L'avantage de vouloir fonctionner avec la méthodologie Agile est que nous pouvions rectifier le tir si nos avancements de livrable en livrable de correspondait pas aux attentes du professeur superviseur.

Séparation des tâches

addcontentslinetocsectionSéparation des tâches Dès que notre sujet pour le projet a été choisi, nous nous sommes lancé dans un survol des tâches qui allaient à être réaliser et nous avons rapidement constaté que la plupart des livrables se séparaient en deux catégories complètement indépendantes l'une de l'autre. Afin de facilité et le développement et séparer les responsabilités et réduire les risques de ces sous-domaines, nous avons divisé les membres de l'équipe en deux sous-équipes pour le lancement du projet. Ses deux équipes s'occupaient des deux grandes familles de tâches qui entouraient les éléments les plus risqués du projet. Ces réalisations étaient risquées par le fait qu'il s'agit de grand inconnu pour l'équipe. Ces

deux familles de tâches étaient : Le développement du programme AR ; Le développement du programme interagissant avec le scanneur matériel. Le but de ses deux équipes était de partir chacun de leur côté pour le commencement afin d'éliminer les inconnus face à ses technologies dès le début du projet. Plus le projet avançait, plus il fallait s'intégrer en une seule équipe pour rejoindre ses deux idées et en faire une seule solution complète.

Analyse préliminaire

Tout d'abord, aucun des membres de l'équipe n'avait des connaissances dans les technologies qui devaient être traitées dans les projets. Il a donc été nécessaire de faire une recherche préliminaire afin de gagner une compréhension nécessaire à la planification du projet. La recherche a permis de répondre à des enjeux sûrs qu'elles étaient les principales parties du projet ? Quelles technologies étaient incluses implicitement aux ARKit et Structure Sensor ? Quel environnement de développement devait être utilisé ? Est-ce que les objectifs étaient réalistes dans les contraintes de temps et de ressources ?

Suite à cette recherche, le projet ayant été jugé comme réalisable a été séparé en plusieurs modules. C'est à dire des parties du projet qui peuvent être développées indépendamment et être assignées à différents étudiants. Ayant conscience de la possibilité de bloquants au cours du développement cette technique permet de ne pas limiter l'avancement du projet complet, dès qu'il y a un problème. Les différents modules identifiés sont : le développement du scanneur avec le Structure Sensor, le développement de l'environnement AR et le développement du squelette dans le modèle humain. Le dernier module a été identifié comme accessoire étant donné l'importance de l'enjeu. De plus, le développement des deux premiers modules et leur combinaison sont déjà importants en taille et complexité.

Une fois les deux modules principaux identifiés, une séparation de l'équipe de développement a été faite. La séparation visait à permettre à chacun de se concentrer sur un type d'enjeu et de diminuer la grosseur des défis. De plus, cette méthode a permis de répondre à des contraintes qui entourait le projet et les technologies disponibles. En effet, un seul périphérique Structure Sensor était disponible. De plus, trois développeurs sur quatre avaient accès à un appareil

Apple mobile.

Une fois les sous-équipes identifiées, celles-ci ont dû séparer et lister les tâches à accomplir, les ressources documentaires disponibles autour de leur technologie et lister les risques que leurs technologies impliquent. Ces informations ont été ajoutées au plan de projet avec la description du projet et la mise en situation. Les tâches ont été par la suite séparées en trois jalons de développement.

Pour le module de Structure Sensor, les trois jalons consistent en le transfert du projet Scanneur de Swift 2 à Swift 4 pour le premier. Pour le second, la cible était l'intégration dans la structure de l'application AR et le transfert du modèle dans l'application. Le troisième consistait en la conception d'une interface visuelle adéquate et une possibilité de faire un début de squelette sur le modèle humain.

Pour le module de Structure Sensor, les trois jalons consistent en le transfert du projet Scanneur de Swift 2 à Swift 4 pour le premier. Pour le second, la cible était l'intégration dans la structure de l'application AR et le transfert du modèle dans l'application. Le troisième consistait en la conception d'une interface visuelle adéquate et une possibilité de faire un début de squelette sur le modèle humain.

Itérations

Chaque jalon représentant une partie du travail à faire contiendra toujours un certain nombre commun de regroupements de tâches. Pour commencer, avant d'entreprendre toute forme de travail, des recherches doivent être faites. Les éléments concernant les différentes méthodes de réalisation d'une tâche et les avantages et désavantages de chacune de celles-ci doivent être identifiés.

Par la suite, une solution doit être sélectionnée en comparant les avantages et désavantages. Le contexte de développement de l'application doit être aussi pris en cause. Une fois la solution choisie, l'adaptation de la conception de la solution doit être faite pour le cadre du projet. Si la solution s'intègre au sein d'une autre solution, les documents de conceptions

de la solution parentes sont adaptés. La conception est donc souvent itérative à travers les jalons et les tâches.

Par la suite, la solution est intégrée dans l'application à l'aide de l'éditeur spécialisé XCode. La compilation et la mise en marche sont testées sur un appareil mobile Apple au fur à mesure du développement afin d'éviter les mauvaises surprises en fin de jalon. Une fois une tâche terminée, le cas d'utilisations principal de l'application est testé pour s'assurer qu'aucune autre fonctionnalité n'a été brisée.

Le jalon est par la suite terminé avec une revue du travail fait et avec une présentation préparée par l'équipe pour le superviseur du projet. Dans ces revues on discute du travail fait, des bloquants et du travail à faire. Les artefacts de conception et documents de projets doivent aussi être mis à jour.

Clôture

Le développement applicatif terminer, certaines tâches doivent être faites afin de clore le projet. Parmi celles-ci est la revue des résultats obtenue par l'équipe de développement. Cette tâche a pour but d'évaluer l'état de l'application par rapport au résultat qui était attendu en début de projet. Dans le cas de différences entre les attentes et le produit final, un bilan de justification doit être tenu.

Par la suite, l'équipe devra lister ces recommandations qui pourraient servir à une autre équipe voulant poursuivre le projet. Le contexte de projet de fin d'études permet en effet la poursuite d'un tel projet. Alors un transfert de connaissance doit être préparé afin de limiter les frictions des futures parties prenantes au projet. Les deux dernières tâches permettent la préparation de la présentation qui sera faite en fin de projet devant d'autres étudiants finissants.

PROCESSUS DE CONCEPTION

Définition des exigences

La démarche décrite dans la dernière section permettra de structurer les tâches de conception exécutées par chaque membre de l'équipe. Ainsi qu'avec les informations obtenues dans la revue documentaire, une solution à l'enjeu soulevé en début de projet a été conçue. Les besoins comblés, le fonctionnement de la solution, les différentes solutions explorées et la solution retenue et prototypée seront présentés.

Les responsabilités que la solution développée devra couvrir consistent en : la prise de modèle dans l'environnement réel, la présentation de ces modèles dans l'environnement réel pris par la caméra, la compréhension des coordonnées de l'environnement capturé, la modification du modèle et la reconnaissance des mouvements de l'appareil. La technologie Structure Sensor peut répondre aux responsabilités de prise de modèle dans l'environnement et compréhension des coordonnées de l'environnement réel. Par contre, les autres responsabilités devront être résolues par d'autres moyens.

Deux possibilités se sont révélées comme intéressantes lors des recherches. La première consistant dans le développement d'une solution personnalisée permettant la gestion du modèle déjà scannée. Cette gestion devrait permettre l'ajout du modèle dans un environnement contrôlé à l'aide des coordonnées acquises avec le Structure Sensor. Une fois ajouté dans l'environnement, le modèle doit pouvoir être déplacé, mis à l'échelle et être pivoté. Ce modèle devra pouvoir être supprimé de l'environnement. Toutes ces différentes tâches demandent donc un cadre de développement permettant la transformation des données provenant des périphé-

riques de capture en scène tridimensionnelle avec lesquels les développeurs interagissent.

Un outil répondant à ce besoin a déjà été publié et il consiste dans la deuxième option. Cet outil est la trousse de développement d'Apple, l'ARKit. Celui-ci fait le pont entre les interfaces du périphérique de capture et la trousse de gestion de scène SceneKit. Cette solution est très intéressante puisqu'elle diminue grandement le poids du travail à faire. Par contre, en ajoutant une interface avec une nouvelle frontière, on ajoute aussi des contraintes au développement. De plus, l'ARKit permettant son usage avec des appareils n'ayant pas de Structure Sensor représente des possibilités fonctionnelles plus limitées. Un exemple de ces limitations est la détection d'objet en premier plan de la scène. Autrement dit, cacher un modèle numérique par un objet réel en le plaçant dans le monde réel est malheureusement impossible.

Afin de diminuer la grosseur du travail à faire, la solution de l'ARKit a été sélectionnée. Les limites que celle-ci importe ne seront pas atteintes dans le contexte du projet. Toutefois, une poursuite de projet pourrait atteindre ces limites.

ID	Titre	Description
EX1	Application mobile	L'application doit être accessible d'une tablette iPad.
EX2	Accès caméra	L'application doit avoir accès à la caméra de l'appareil l'hébergeant.
EX3	API Structure	L'application doit faire façade à l'API de Structure Sensor.
EX4	Scanner 3D	L'application doit pouvoir scanner des modèles 3D via le Structure Sensor.
EX5	Insertion de modèle	L'application doit pouvoir ajouter un modèle 3D sur une surface plane à l'aide de l'ARKit.
EX6	Transfert de modèle Structure à ARKit	L'application doit pouvoir exporter un modèle scanner à un modèle utilisable par l'ARKit.
EX7	Accès au disque	L'application doit avoir accès aux disques de la tablette.
EX8	Sauvegarde sur le disque	L'application doit pouvoir sauvegarder les modèles sur le disque.

EX9	Liste des modèles	L'application doit pouvoir lister les modèles disponibles
EX10	Ouverture sans caméra	L'application doit pouvoir ouvrir sans avoir directement accès à la caméra.
EX11	Ouverture sans Structure Sensor	L'application doit pouvoir ouvrir sans avoir directement accès au Structure Sensor.
EX12	Icon	L'application doit être présentée avec une icône sur la tablette.
EX13	Données de distance	L'application doit permettre le scan avec une prise de données sur la distance.
EX14	Transfert de la distance	L'application doit permettre le transfert des données de distance du modèle scanner au modèle ARKit.
EX15	Présentation de distance	L'interface utilisant l'ARKit doit permettre de présenter la distance du modèle à l'utilisateur.
EX16	Interface accueil	L'application doit présenter une interface d'accueil efficace.
EX17	Interface Navigation	L'application doit présenter une navigation naturelle à l'utilisateur.
EX18	Interface de scan	L'application doit présenter une interface adéquate pour la prise de modèle 3D.
EX19	Interface de manipulation 3D	L'application doit présenter une interface adéquate pour la manipulation de réalité augmentée.
EX20	Interface catalogue	L'application doit présenter un catalogue des modèles.
EX21	Interface descriptive d'un modèle	L'application doit présenter une interface adéquate de description de modèle adéquate.
EX22	Normes Apple	L'application doit présenter une expérience utilisateur suivant les lignes directrices des applications mobiles Apple.
EX23	Code testable	Les fonctionnalités et modules de l'application doivent être testables de façon unitaire.

EX24	Swift et Objective-C	L'application doit être programmée en Swift et Objective-C.
EX25	Normes de programmation	L'application doit respecter les normes de programmation des langages Swift et Objective-C.
EX26	Exportation dans un fichier scène	L'application doit permettre l'export de modèle ARKit sur un ordinateur.
EX27	Exportation dans un fichier objet	L'application doit permettre l'export de modèle Structure Sensor sur un ordinateur.
EX28	Temps de démarrage	L'application doit ouvrir dans les temps normaux d'une application (2 secondes maximum).
EX29	Fluidité	L'application doit être fluide dans ces animations et navigations.
EX30	Exigences de sécurité par l'Apple Store	L'application doit respecter les exigences de sécurité Apple ce qui permettrait de la publier sur le App Store.
EX31	Données usagers sécurisés	Les données de l'usager doivent toujours être sécurisées et non accessibles de l'extérieur de l'application.
EX32	Création de nœuds	L'application doit permettre l'identification de nœuds dans un modèle.
EX33	Liaison entre les nœuds	L'application doit permettre la création de segments entre les nœuds d'un modèle.
EX34	Mouvement de nœuds	L'application doit permettre le mouvement d'un nœud dans un modèle.
EX35	Transfert du mouvement sur le modèle AR	L'application doit être capable de calculer l'effet du mouvement d'un nœud sur le modèle AR.
EX36	Détection de squelette dans un modèle (simple)	L'application doit être capable de détecter un squelette de nœuds dans un modèle simple scanner.
EX37	Détection de squelette dans un modèle (humain)	L'application doit être capable de détecter un squelette de nœuds dans un modèle humain scanner.

EX38	Limitation des mouvements	L'application doit être capable de limiter les mouvements et angles entre les segments et nœuds.
EX39	Fluidité des squelettes	L'application doit rester fluide lors de l'édition d'un squelette et l'animation de celui-ci.

TABLE 1: Liste des exigences

Risques et hypothèses

Dans un projet de cette envergure, les risques sont nombreux et importants. Afin d'assurer une bonne planification et une bonne gestion de ceux-ci, une réunion a été convoquée pour identifier ces risques et élaborer une stratégie de mitigation dans le but d'atténuer leurs impacts. La liste ci-dessous présente les différents risques qui ont été ciblés avec leur cote d'impact et de probabilité ainsi qu'un résumé de la stratégie de mitigation prévue.

Risque	Impact	Probabilité	Migration é atténuation
Les difficultés d'implémentation des interfaces de programmation peuvent bloquer le développement.	Très important	Très probable	Afin de réduire l'impact de ce risque, les attentes du produit final ont été réduites afin de laisser assez liberté à la résolution de problème. De plus, des rôles ont été assignés à des membres de l'équipe en fonction des interfaces de programmation ce qui permet le travail de chacun en parallèle.

La dépendance à une technologie matérielle, le Structure Sensor, peut amener une réduction dans le temps de développement.	Très important	Très probable	Afin de limiter cette dépendance, l'équipe a décidé de manipuler les modèles 3D à l'aide de la trousse de réalité augmentée Apple. Les développeurs ayant chacun accès à un appareil Apple certains pourront se concentrer sur l'ARKit sans avoir besoin du senseur.
Les tests automatisés d'interface mobile sont complexes à implémenter.	Important	Probable	Des tests unitaires seront développés pour toutes les fonctions qui ne sont pas directement en contrôle de l'interface graphique. Les responsabilités du code côté interface seront limitées et feront principalement appel à des fonctions tests de façon unitaire.
Il y a un manque de documentation pour le Structure Sensor ce qui peut limiter la vitesse de développement.	Important	Probable	Une validation des ressources disponibles a été faite préalablement à la validation de la sélection du projet. La documentation officielle est en effet dommage, toutefois le nombre d'exemples disponible et la présence d'un forum de développeur laissent croire que les problèmes seront résolus.

Il n'y a pas de propriétaire du produit ou client officiel, la vision du produit final peut donc être perdue au cours du développement.	Peu important	Peu probable	Lors de la première rencontre d'équipe, chacun des membres à identifier ces attentes par rapport au produit final. Un résumé de la vision de chacun et une synthèse de la solution attendue ont été pris en note.

TABLE 2: Liste des risques

En plus des risques (présenté ci-haut), une liste d'hypothèses a aussi été produite dans le but de prédire et de prévoir l'impact de certains éléments sur le projet. Celle-ci est présentée ci-dessous et contient la catégorie ainsi que l'auteur de cette hypothèse.

ID	Titre	Description
HYP1	Interface de programmation	S'assurer du bon fonctionnement des interfaces de programmation et le matériel sera la plus grande difficulté de la première moitié du projet.
HYP2	Détection de squelette	La détection de squelette sera seulement une preuve de concept fonctionnant dans des conditions précises.
HYP3	Mouvement et calcul	Le mouvement des squelettes et modèles est une tâche très exigeante en termes de calcul et impliquera des pertes de performance.
HYP4	Communauté de développeurs d'application pour l'ARKit d'Apple	L'ARKit étant développé par Apple doit avoir une grande communauté de développeurs et un bon support ce qui le rend plus intéressant qu'un simple modèle Structure Sensor.

TABLE 3: Liste des hypothèses

Avancement itératif

Itération 0

Conception

Composantes à intégrer Un travail préliminaire à la réalisation du travail fu de lister les différentes composantes qui devaient être intégrer dans notre application. Cette liste permet au développeur de comprendre l'envergure du travail à accomplir. De plus, elle permet une compréhension simple de chaque partie du projet même, si on n'a pas travaillé dessus.

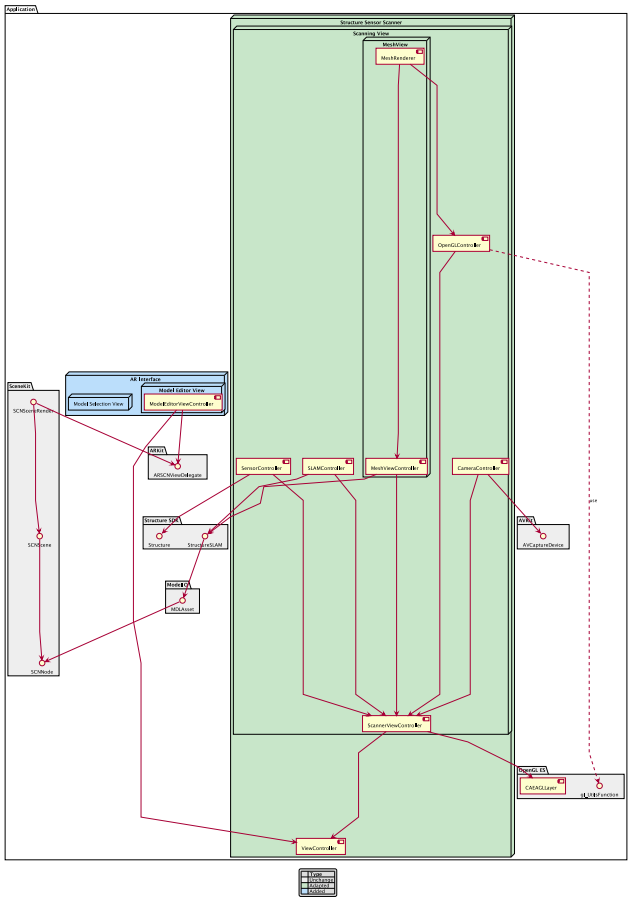

FIGURE 2 – Diagramme des composantes de l'application

Comme on peut le voir, le produit, lorsqu'implémenter avec l'ARKit, consiste en une appli-

cation intégrant plusieurs interfaces. Ce qui représente définitivement un aspect facilitant au développement. Toutefois, un tel choix présente des limites et demande une adaptation de l'application pour réduire la friction aux frontières de chacune.

Les différentes interfaces font face à une variété d'enjeux et intègre une variété de technologies produite non seulement par Apple. L'interface de réalité augmentée récemment développée par Apple, ARKit. Cette librairie a pour principale tâche, l'intégration des fonctionnalités de la caméra et de la détection de mouvement afin de créer une expérience de réalité augmentée.(Apple, 2018a)

Cette API est utilisée en coopération avec le SceneKit l'interface de gestion d'environnement 3D dans iOS aussi produit par Apple. Celle-ci simplifie les principaux défis du développement 3D comme les animations, la simulation physique, les effets de particules et le rendu réaliste avec gestion d'effet de lumière. L'usage d'une interface produite par le propriétaire de l'appareil hôte de l'application permet de s'assurer d'une optimisation fournie.(Apple, 2018f)

Les modèles utilisés par les scènes proviennent de modèle 3D importé par l'interface de Model I/O, ils sont convertis par la suite en ScnNode. Le modèle permet la gestion des textures, des fichiers, des caméras et des matériaux.(Apple, 2018d) Ce genre de modèle n'est pas propriétaire à Apple, il peut donc être utilisé par Unity(Unity, 2016) et Blender(Marklew, R., 2013). L'importation des modèles 3D se fait à l'aide du senseur Structure qui est une des principales technologies du projet. Ce périphérique utilise la caméra de la tablette et la caméra sur le périphérique pour avoir une meilleure compréhension de l'environnement 3D. Le périphérique contient aussi un senseur infrarouge pour faciliter l'interprétation de l'environnement. Un kit de développement logiciel est fourni par les développeurs du périphérique permettant le contrôle du senseur et l'exportation en modèle.(Occipital, 2018) Les images capturées par le senseur et la caméra sont par la suite présentées par l'interface de programmation AVKit. La classe AVCaptureDevice est ainsi utilisée pour présenter à l'usager l'environnement pris par l'appareil.(Apple, 2018b) Afin d'offrir un rendu plus réaliste, la librairie OpenGL est aussi utilisée.(Apple, 2018e)

Les liens entre la logique de notre application et celle des API se font par des extensions

aux contrôleurs des vues principales de notre application et par l'intégration de certaines interfaces. Les interfaces de Structure, OpenGL, AVCaptureDevice se font par l'extension du contrôleur de vue du scanneur. Pour ce qui est de l'ARKit et SceneKit, les classes respectant le patron « delegates » pour chacun ont été implémentées dans le contrôleur de la vue de manipulation de modèle.

Chacun des contrôleurs est lié à une vue dans le storyboard et par les liens créer par la logique ou par ceux créer dans le storyboard, ce qui permet à l'usager de transférer d'une à l'autre.

Réalisation Dans la méthodologie AGILE, les projets se doivent de commencer avec une itération 0. Le but de cette itération est de devoir élaborer les spécifications et la portée du projet avec le client. Cette itération a débuté avec une rencontre avec monsieur Carlos Vasquez afin de savoir ce qui serait inclus dans cette session de développement. Ce qui est sorti de ses rencontres avec le client et avec les coéquipiers de l'équipe était la liste des exigences présentée précédemment et un calendrier des itérations. Carlos semblait ravi de l'idée de faire plusieurs points de contrôle avec lui tout au long de la session afin de rectifier le tir sur le travail accompli et restant à accomplir.

Itération 1

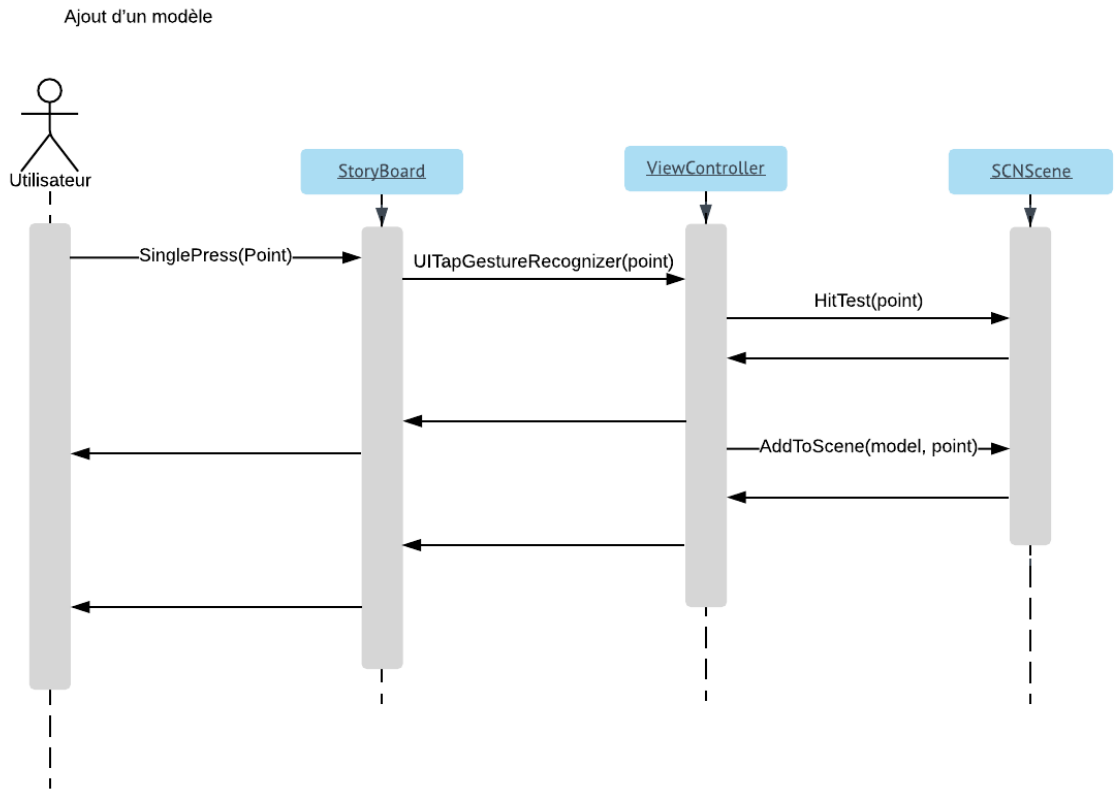

FIGURE 3 – Diagramme de séquence d'ajout d'un modèle

Conception Dans le diagramme ci-haut, on peut voir l'interaction entre les classes afin de pouvoir ajouter un modèle dans une scène de réalité augmentée. Quand un utilisateur clique sur l'écran, l'action est détecté par le storyboard. Un storyboard est la classe qui gère les composantes visuelles d'une application IOS. Il y a plusieurs types de geste pouvant être détecté par le storyboard, mais pour ce scénario, seulement le simple toucher est détecté. Une

fois détecté, le storyboard envoie le type de geste au contrôleur de la vue. Ici ce contrôleur reçoit un geste de type “UITapGesture” et le méthode “UITapGestureRecognizer(point)” est appelée. L’argument point de la méthode possède plusieurs informations sur le geste dont les coordonnées spatiales pour savoir où déposer le modèle. Dans cette méthode, il y a un appel à la classe “SCNScene” qui possède une méthode afin de savoir si le point est dans la zone de surface détecté auparavant. Si c’est le cas, on ajoute le modèle à la scène virtuel, ce qu’y va générer la vue avec le nouveau modèle à la position souhaité. Pour être détecté comme un simple touché, il y a une limite de 0.3 secondes, sinon le geste est un long touché ce qu’y résulte en un autre type de geste qui sera traité plus tard.

Scan Lors de la première itération, le processus de conception du module de scan a aussi débuté. La procédure suivante devra être implémentée pour permettre un scan avec le Structure Sensor.

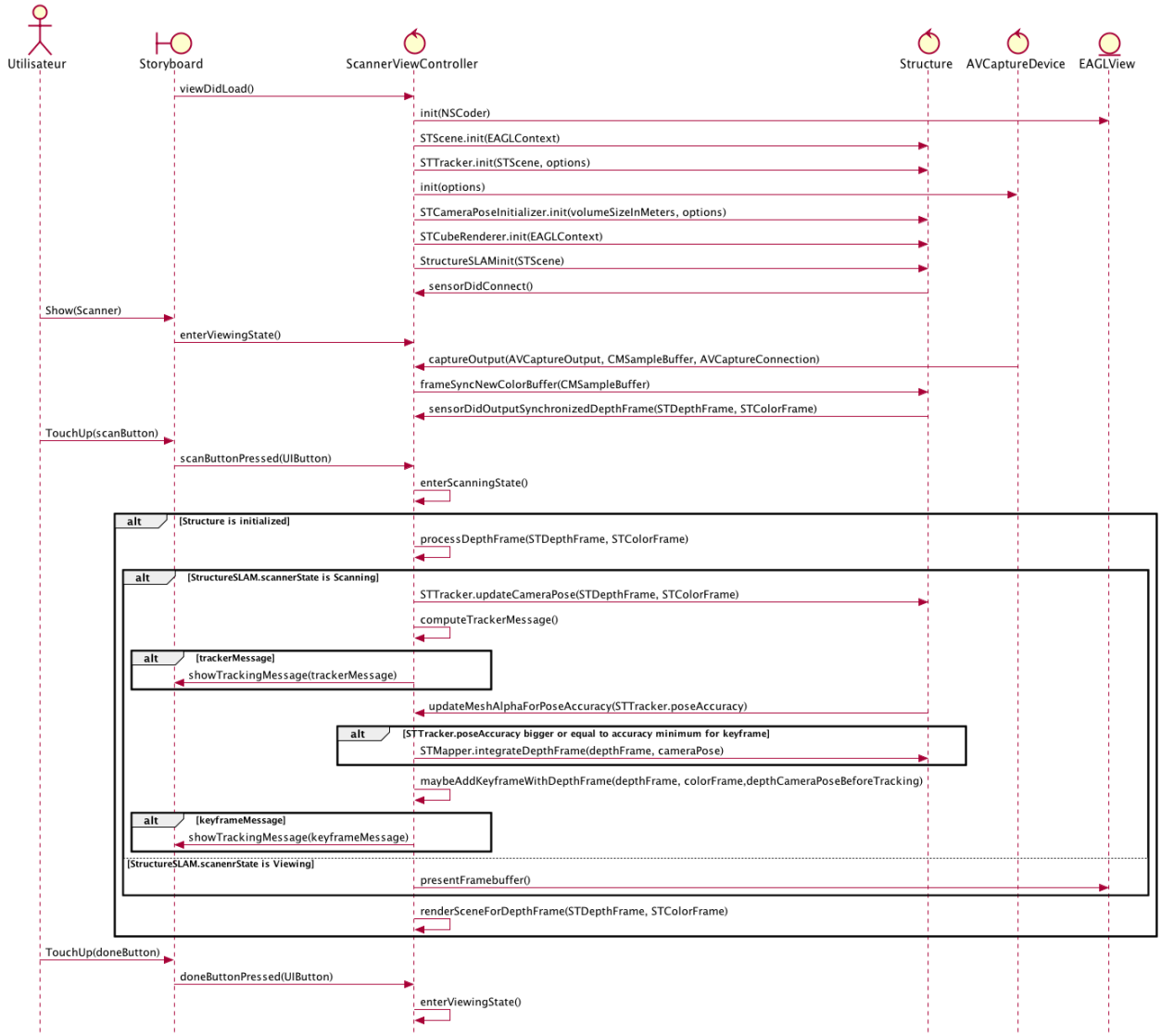

FIGURE 4 – Diagramme de séquence d'un scan d'objet

Le diagramme de séquences pour le scan d'un modèle montre les étapes nécessaires à produire un modèle d'objet scanner. Les premières étapes avant la première interaction avec l'usager sont effectuées au lancement de l'application. Elles consistent en la procédure d'initialisation de toutes composantes nécessaires à un scan. Les différentes interfaces du Structure Sensor,

le traqueur, la scène, le rendu du cube du heat map et le convertisseur de trame en maille, ainsi que la caméra et la vue en OpenGL.

Au relâchement du doigt de l'utilisateur sur bouton de navigation vers l'interface de scan, le storyboard de l'application va transférer le contrôle à la scène de scan. À ce moment, l'application doit s'identifier comme client de la caméra de l'appareil. De plus, elle demande le démarrage de la session de capture du périphérique Structure Sensor. La trame de couleur prise par la caméra de la tablette est envoyée à chaque intervalle de capture au périphérique. À la réception, le périphérique va trouver la trame de profondeur prise à l'instant de la trame de couleur et renvoie les deux au contrôleur de vue. Selon le mode actif, le scénario diverge. Dans le mode de visualisation, on ne fait que présenter les deux trames dans la vue OpenGL. Suite à la réussite de ces différentes étapes, la vue principale présente l'environnement devant l'utilisateur avec un cube avec un heatmap facilitant la communication des points d'intérêt à l'usager.

Au relâchement du bouton « Scan », l'utilisateur, l'application entre en mode de capture. Dans ce mode, chaque combinaison de couleur et profondeur est évaluée comme étant un moment clé par le convertisseur de trames en maille et le traqueur. Si les deux trames sont en effet clé, les nouvelles coordonnées sont ajoutées à la maille. Le traqueur et le convertisseur peuvent détecter des difficultés causées par l'usager. Quand c'est le cas, les avertissements doivent être présentés à l'utilisateur. Pour compléter un scan l'utilisateur doit peser et relâcher le bouton « Done ».

Réalisation

Scanner Dans cette première itération, l'avancement a été considérable au niveau de la sous-équipe responsable des technologies AR. Lors de ces 3 semaines, deux applications distinctes ont été développées afin de permettre une indépendance dans les deux approches technologiques. À la démonstration au client de cette itération, l'équipe avait déjà une détection de plan horizontal fonctionnel et pouvait déjà déposer un objet téléchargé sur ce plan. L'équipe de réalité augmentée ont dû trouver des modèles temporaires afin de tester l'ajout

d'objet dans la scène.

Réalité augmentée Dans cette première itération, l'avancement a été considérable au niveau de la sous-équipe responsable des technologies AR. Lors de ces 3 semaines, deux applications distinctes ont été développées afin de permettre une indépendance dans les deux approches technologiques. À la démonstration au client de cette itération, l'équipe avait déjà une détection de plan horizontal fonctionnel et pouvait déjà déposer un objet téléchargé sur ce plan. L'équipe de réalité augmentée ont dû trouver des modèles temporaires afin de tester l'ajout d'objet dans la scène.

Test et Validation À ce stade du développement, encore très peu de tests avaient été élaborés, car il était seulement question de faire fonctionner des technologies dont l'équipe n'était pas du tout familière. Cependant, avant la démonstration l'équipe a testé sur plusieurs styles de surface leurs méthodes de détections de plan 3D afin de s'assurer que la démonstration se déroule avec succès.

Itération 2

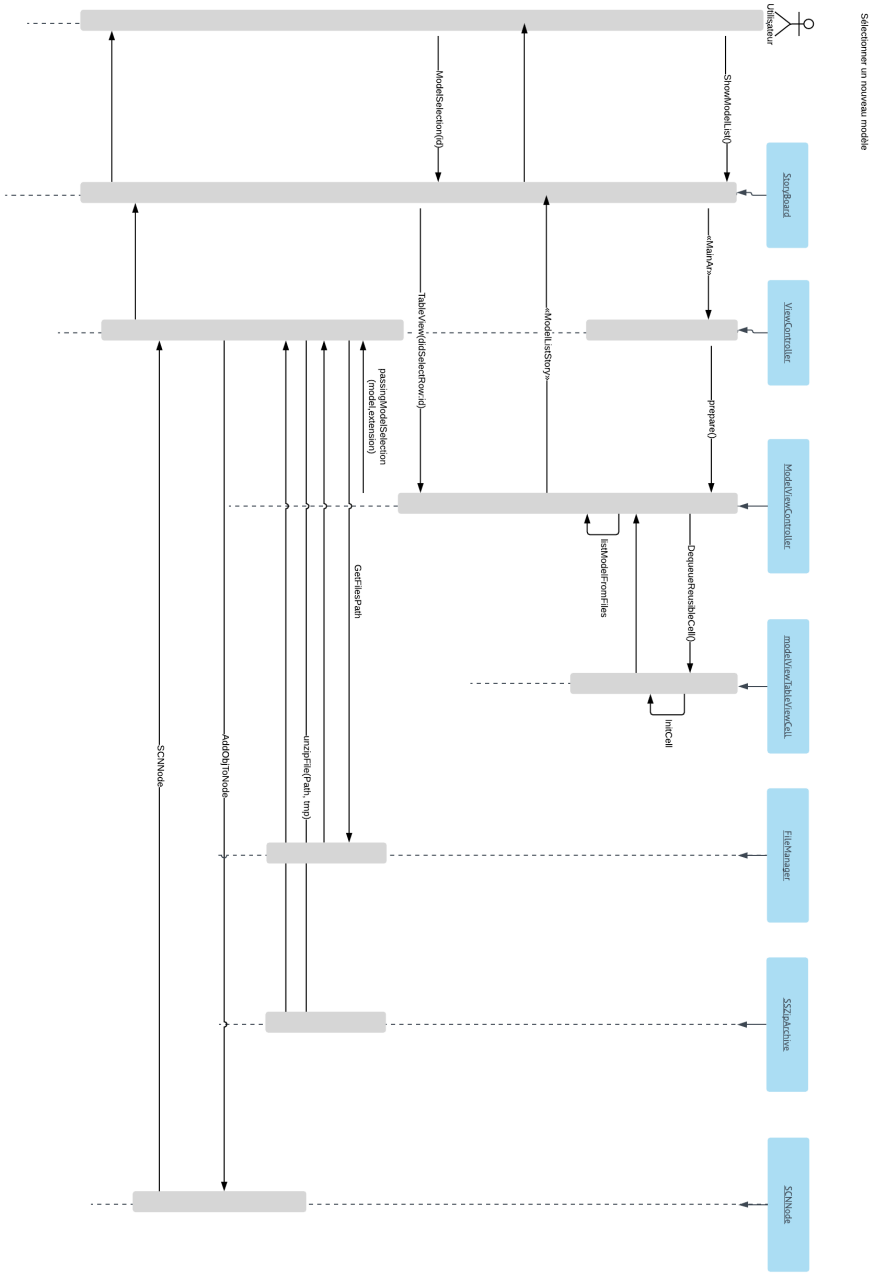

FIGURE 5 – Diagramme de séquence de sélection de modèle

Conception Ce diagramme de séquence montre encore une fois la connexion entre plusieurs classes du projet. Il peut sembler complexe, mais l'action décrit dedans ne l'est pas. Il

y a seulement beaucoup de dépendance de classe pour ce processus. Comme pour toute application IOS, le point d'entré des actions est le storyboard. L'utilisateur clique sur le bouton contenant le nom du modèle afin de pouvoir voir une liste des modèles disponibles. Il s'agit du seul bouton de l'interface principale, l'idée était d'avoir l'expérience utilisateur la plus épurée possible afin de ne pas cacher les capacités de la réalité augmentée. Cet action appelle le contrôleur principal de l'application ("viewController"), afin d'initialiser une nouvelle vue avec un nouveau contrôleur. Il s'agit du "modelViewController". C'est dans ce contrôleur que la liste est générée en listant tous les fichiers d'un répertoire avec la classe FileManager. Avec cette liste, un élément de type "TableView" est créé et ajouté à la vue. L'utilisateur peut alors cliquer sur la cellule du tableau pour sélectionner le modèle qu'il souhaite avoir. C'est alors que le contrôleur utilise la classe "SSzipArchive" pour décompresser le fichier du modèle dans un le répertoire temporaire du téléphone, afin d'être utilisé comme une cache. Si le fichier existe déjà, il n'y aura pas de décompression. Ensuite, il retourne au contrôleur principal le choix de l'utilisateur et l'extension du fichier, car les fichiers disponible sont : .obj ou .scn. C'est alors que le contrôleur lit le fichier pour générer un objet de type "SCNNode". Ce type d'objet est utilisé pour modéliser le modèle dans l'environnement virtuel. Le modèle est alors prêt à être ajouté à la scène. Il suffit d'un simple touché sur une surface détecté pour déclencher l'ajout tel qu'expliqué dans la section précédente.

Réalisation

Scanner Cette itération a eu une drôle de formule, car dès son début, l'équipe avait enfin réussi à faire fonctionner le code du scanneur qui permettait de se connecter à l'API du StructureIO. Avec de la chance, la présentation de l'itération 1 a dû être retardée et Carlos a donc pu constater l'avancement du scanneur qui était presque fonctionnel lors de la démonstration technique. La sous-équipe du scanneur commençant leur itération avec un avancement important au niveau des fonctionnalités, ils ont eu un regain de motivation et l'accomplissement final pour la deuxième itération est plus qu'impressionnant. Étant donné que le code devait être réécrit dans une nouvelle version de Swift, certains problèmes étaient toujours présents lors de la démonstration de la première itération. En effet, lors de la pré-

cédente démonstration devant Carlos Vasquez, la synchronisation de la caméra infrarouge avec celle de l'IPAD fourni n'était pas fonctionnelle et c'est pour cette raison que les modèles scannés à l'intérieur de l'application étaient de couleur grise et sans texture. Pour cette itération, l'objectif principal a été de réparer ce problème de synchronisation. Le problème a finalement été résolu suite à de longues heures de débogage. C'était relié sans surprise par un problème lié au fait que les fonctionnalités de Swift 2 et 4 sont parfois très différentes. Pour être plus précis, il s'agissait d'un problème de traitement de pointeur mémoire de l'iPad qui était mal implémenté. C'est donc avec grande fierté que l'équipe a pu présenter un scanneur fonctionnant avec des couleurs et textures pour la deuxième démonstration.

Réalité augmentée Lors de cette itération l'équipe s'occupant de l'application utilisant le ARKit s'est confectionné un nouveau patron stratégie connecté à menu déroulant afin que l'utilisateur de l'application puisse choisir le modèle de son choix. Ce petit ajout à l'expérience utilisateur pourrait sembler banal à première vue, mais quand nous constatons les patrons de conceptions derrière tous ses appels système, nous comprenons bien la complexité de ce nouvel ajout aux fonctionnalités. C'est aussi à cette itération que l'équipe à pu réaliser la complexité des étapes à venir et a décidé de retirer des requis de projet final afin que la réalisation reste raisonnable et faisable dans suivant la restriction de temps imposé à l'équipe. C'est en effet lors de la deuxième démonstration à Carlos Vasquez que l'équipe et le client ont décidé de reculer sur tout ce qui était la gestion du squelette du modèle 3D et de ne pas s'attaquer à la modification architecturale des modèles sélectionnés.

Test et Validation Lors de cette itération plusieurs tests ont été nécessaire afin de s'assurer de toutes les fonctionnalités suivaient bien toutes les exigences et leurs requis. Plusieurs tests d'exécution se sont faits pour les scans de couleurs afin de bien s'assurer que la synchronisation de la caméra de l'iPad et du Structure était idéale. Des tests manuels de style exploratoires ont aussi été suivis pour l'application roulant le ARKit et cela sur plusieurs périphériques différents afin de constater que tous les appareils agissaient de façon semblable, mais pas exactement identique. À ce moment il y aurait pu y avoir des problèmes comportementaux, mais tout semblait beau.

Itération 3

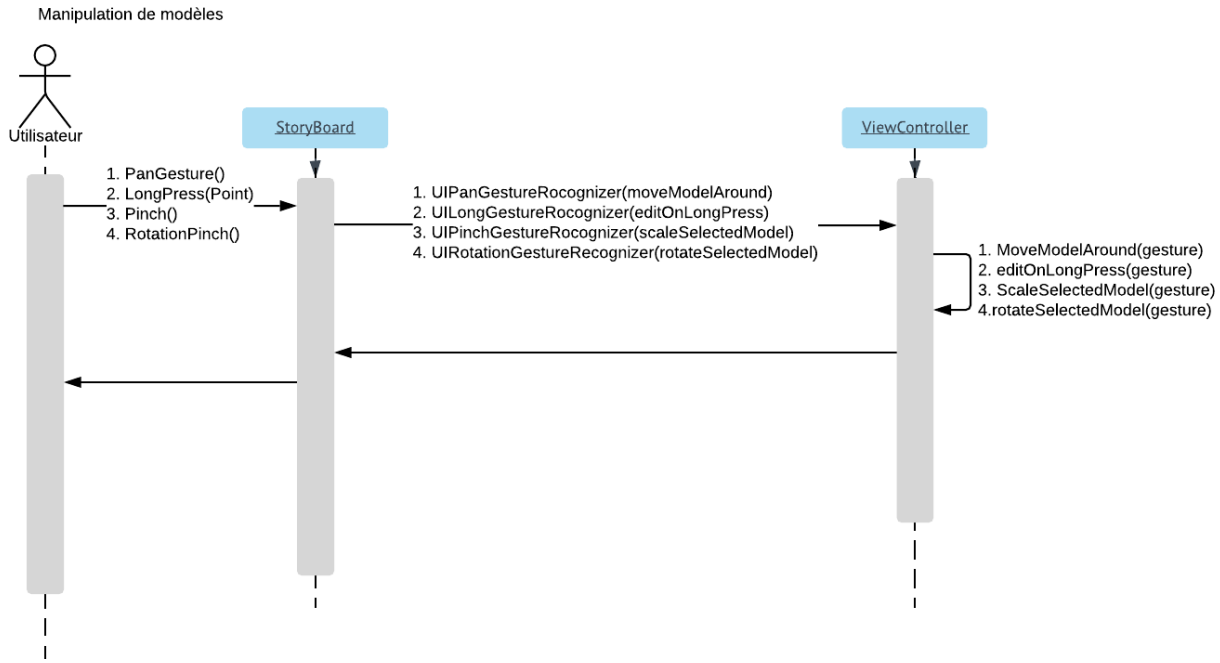

FIGURE 6 – Diagramme de séquence d'une manipulation de modèle

Conception Pour ce Diagramme de séquence, l'équipe a voulu montrer les différents gestes détecté par l'application. Au lieu de faire plusieurs fois le même diagramme, tous les gestes ont été inserer dans un seul et numéroté, car le nom des méthodes changes selon le geste. Ce comportement peut être vue comme le patron stratégie, qu'y pendant l'exécution, change la façon de répondre à un geste selon son type. Il est possible de faire un touché prolongé sur un objet pour le mettre en mode édition. Ce mode permet alors de faire les autres gestes. Ils sont tous gérés par le même contrôleur, car ce sont des gestes détectés par le storyboard de la vue dont le contrôleur principal est le même pour tous. En mode édition, il est possible de bouger son doigt, afin d'effectuer une translation sur le plan des x et z, faire une rotation de deux doigts sur l'écran pour tourner l'objet sur l'axe des y et séparer deux doigts pour faire un changement d'échelle soit positif ou négatif. Tout dépend de l'orientation du geste. Chaque méthode de mouvement possède des arguments pouvant facilement être modifiable,

afin de changer les vitesses et les facteurs de chaque transformation géométrique. L'équipe a aussi réussi à implémenter une méthode qui permet d'utiliser simultanément tous les gestes, ce qu'y ajoute un aspect positif à l'expérience utilisateur, car cela rend les gestes très fluides et facile à comprendre.

Réalisation Lors de cette itération les deux sous-équipes sont devenues une seule grande équipe afin de bien s'intégrer une dans l'autre. C'était donc la phase de l'intégration, mais ce n'est pas seulement ce qui a été réalisé dans cette itération.

En effet, pour commencer la nouvelle grande équipe s'est occupée de rejoindre les deux applications distinctes des deux équipes sous un même projet Swift afin que les deux codes cohabitent pour le reste du projet. Avec quelques problèmes techniques évidents qui suivent une telle réorganisation de code, ils se sont occupés de régler tous les problèmes le plus rapidement possible afin que l'autre sous-équipe de soi pas du tout impacté. En plus de ce travail de réconciliation de code, ils ont implémenté des traitements numériques aux objets 3D téléchargés liés avec l'interface utilisateur. En suivant des principes vus en cours à l'ÉTS, cette équipe à implémenter des ajouts remarquables à l'expérience utilisateur à la partie AR du projet. En effet, lors de la troisième démonstration devant Carlos Vasquez, l'équipe a montré qu'un utilisateur pouvait modifier la grandeur du modèle choisi à l'aide de deux doigts. La sélection du modèle, la rotation autour de l'axe Y et le déplacement sur un plan X-Z ont aussi été implémenté et démontré.

Les deux projets roulaient maintenant sous la même application, mais aucune ligne directrice ne reliait ces deux applications ensemble. C'est donc à cette itération que la dernière exigence importante a été implémentée. Lorsque l'on scan un objet à l'aide de la partie scanner de l'application, la possibilité de sauvegarder ce nouveau modèle le storage de l'application et de maintenant pouvoir l'utiliser comme un nouveau modèle dans le menu déroulant des objets disponibles à l'ajout de notre scène de réalité augmentée. Avec cette nouvelle fonctionnalité ajoutée certain problème sont survenus, car ses modèles n'étaient pas tout à fait prêts à être utilisés. C'est alors qu'avec la collaboration de tous les membres de l'équipe, il a été possible de faire les modifications numériques à même le modèle en pleine exécution.

Test et Validation À ce stade le produit était maintenant prêt à être démontré dans son état final. C'est donc à cette fin de dernière itération que les tests d'intégration ont été effectués. Ces tests d'intégration étaient de style exploratoire et consistaient à suivre un scénario de test élaboré conjointement avec les exigences réalisées. Ce scénario était de démarrer l'application qui avait été préalablement nettoyée de tous ses précédents modèles et d'y scanner un nouveau modèle quelconque, de revenir à l'interface AR et d'y sélectionner le modèle nouvellement ajouté et de la placer directement sur le plan X-Z détecté. Il fallait ensuite effectuer toutes les transformations implémentées à ce nouveau modèle et constater le fonctionnement de l'application. C'est bien à ce moment quelques défaillances sont été constaté et corrigé. En effet, comme mentionné plus tôt, les modèles scannés à l'aide du StructuteIO Sensor, étaient renversés et éloignés de quelques mètres de leur point de rotation. C'est à ce moment que tous les membres de l'équipe se sont mis sur les derniers correctifs nécessaires pour remédier à cette drôle de situation. Maintenant, lors de leur sélection dans le menu déroulant, les modèles provenant du StructureIO Sensor subissent une rotation autour de l'axe des X de 1 RAD ainsi qu'une modification de boîte de délimitation et un déplacement de leur point d'axe au centre de la face du bas de leur nouvelle boit de délimitation.

DISCUSSION

En rappel, l'objectif initial du projet était la production d'une application sur tablette Apple permettant de scanner un objet de l'environnement réel à l'aide du Structure Sensor et le présenter sur une surface plane en réalité augmentée à l'aide de l'ARKit. Le produit satisfait cette attente à la fin de la période de développement soit la fin de la troisième itération. En effet, les deux interfaces nommées sont utilisées. De plus, le cas d'utilisation impliqué par l'objectif est tout à fait possible.

Comparatif du produit planifié et actuel

Au début du processus de conception, l'équipe a identifié une cible pour le produit final. Maintenant la période de développement finalisé, l'état actuel de l'application doit être comparé avec la cible. Les exigences, les risques et les hypothèses identifiés seront alors revisités.

Exigences

Comme on peut le voir en Annexe I, le statut des exigences de l'objectif principal a pour la plupart été complété. Les écarts entre celles planifiées et celles réellement complétées sont le résultat de décisions prises lors du processus de conception. Une entente avec la résolution à entreprendre a été discutée avec tous les membres de l'équipe.

Parmi ces écarts, on trouve l'exigence ST11 portant le nom ouverture sans caméra qui indiquait que l'application devrait pouvoir être ouverte sans accès à la caméra. Cette exigence

a été annulée puisque la caméra est à la base des fonctionnalités de l'application. Donc, la gestion de l'exception semblait inutile. Surtout étant donné, le travail supplémentaire que celle-ci impliquait quand le stade de déploiement ciblé est un prototype.

L'exigence ST22 nommée Interface descriptive de modèle a aussi été annulée. Cette exigence impliquait la création d'une nouvelle interface dans l'application pour décrire le modèle. Celle-ci demandait donc un travail supplémentaire de conception, l'ajout de composante de navigation et la recherche de statistiques de fichiers internes à une application. L'utilité de cette interface ne paraissait pas suffisante pour la prioriser par rapport à d'autres, plus importantes pour le bon fonctionnement.

L'application actuelle consiste pour la plus grande partie en l'intégration de plusieurs interfaces de programmation dans un même produit. Les différentes interfaces sont testées par les institutions les déployant et publiant. Ce qui limite grandement le code qui n'est pas testé dans notre application. C'est pourquoi l'exigence code testable avec l'identifiant ST24 a aussi été mis de côté. Le niveau de déploiement prototype et le nombre de travail nécessaires à la création de tests étaient trop élevés.

Deux exigences, ST27 et ST28, qui consistent en l'exportation des fichiers créés lors des scans et des modifications dans l'interface de réalité augmentée. Celles-ci sont considérées comme partiellement fonctionnelles puisque lors de la connexion sur l'outil de développement XCode on a accès à ces fichiers. Donc, le propriétaire du prototype peut les exporter.

Les exigences ST33 à ST40 ciblaient l'objectif accessoire d'avoir un squelette de nœuds dans le modèle pour pouvoir l'animer. Cet objectif n'a pas pu être complété dans la période disponible pour l'exécution du projet. Donc, ces exigences ont été reportées à une poursuite possible du projet.

Risques

Les risques identifiés dans la planification ont bien été des causes de problèmes lors du développement. Les impacts et les probabilités prévues pour chacun respectaient bien l'échelle

réelle. Les dépendances de technologiques, le manque de documentation et le manque de leadership dans le projet ont introduit des ralentissements dans la production de l'application.

Le plus grand défi fut en effet les difficultés d'implémentation des interfaces de programmation. Le changement de l'interface de programmation du Structure Sensor impliqué par le changement de langage d'Objective-C à Swift. Un grand nombre de temps a été mis sur le transfert d'un format à l'autre.

Hypothèses

Comme indiqué dans la section précédente, la difficulté d'implémenter les interfaces de programmation a été le plus grand défi du projet. Ce qui confirme la première hypothèse HYP1. Une des raisons pour lequel l'implémentation à l'aide de l'ARKit au lieu de tout faire dans le contexte Structure Sensor était en réponse à une très petite communauté de développement sur le périphérique. Alors que le nombre de développeurs utilisant le ARKit était plus intéressant pour aller chercher de l'aide chez la communauté. Cette hypothèse, HYP4, a été en effet confirmée. En effet, 13 millions d'applications ARKit ont été téléchargées entre octobre 2017 et mars 2018. (Remi, 2018)

Les deux autres hypothèses n'ont pas pu être confirmées ou invalidées puisqu'elle adressé l'objectif accessoire du squelette humain. Par contre, la vraissembalité de celles-ci n'a pas changé aux yeux de l'équipe de travail. Donc, la validation peut être reportée à la poursuite possible du projet.

Améliorations possibles

Étant donné la durée restreinte de ce projet, il est certain que certaines étapes de peaufinage ont été abandonnées afin de laisser place à un produit fonctionnel d'un bout à l'autre. Comme mentionné plus tôt, beaucoup d'inconnus technologiques ont causés des réalignements dans la portée du projet. Dans cette section, chaque amélioration possible à notre produit courant

sera énumérée et expliquée en détails

Manipulation des nouveaux objets

Le fonctionnement de bout en bout de l'ajout d'un nouveau modèle scanné à l'environnement AR a seulement été un succès lors de la dernière itération. Pour cette raison il a été impossible de rendre ce phénomène le plus élégant possible. Dans son état actuel, un nouveau scan produit un fichier .ZIP qui est stocké à la racine du projet afin que les deux parties du logiciel (AR et Scanner) puissent y accéder. Par la suite, lors de la sélection de ce modèle en particulier dans la liste des objets disponible, ce fichier est décompressé dans un dossier temporaire et un traitement numérique est fait sur cet objet afin de le convertir en objet utilisable par le ARKit de Apple. Sans avoir eu de contrainte de temps, il aurait été beaucoup plus élégant de faire tout ce traitement numérique une seule fois lors de l'enregistrement d'un nouveau scan à notre application. Cette modification sauverait du temps de calcul CPU et mémoire de l'appareil sur lequel ce logiciel roulera dans le futur. Cette erreur de conception a été insérée au produit final lorsque des problèmes de permissions d'accès aux fichiers sont survenus et que l'équipe a voulu contourner un problème de façon à pouvoir continuer le développement sans impacter les autres composantes.

Interface utilisateur du scanneur

Comme mentionné dans les sections précédentes, des changements majeurs ont dû être effectués sur le code d'appel à l'API du Structure Sensor, car ses classes d'interfaces fournies étaient toutes conçues pour de vieilles versions de Swift. En se basant sur les projets démonstratifs de l'API, l'équipe pensait être capable de faire beaucoup de réutilisation, mais étant donné que de la réingénierie a été nécessaire dans les appels à l'API d'interface du matériel et du traitement des données, tout le côté visuel de l'application de démonstration a été conservé. Le but initial était de refaire une interface visuelle pour interagir avec le code modifié, mais le manque de temps causé par le manque de documentation pour les récentes versions de Swift ont causé du retard dans la sous-équipe responsable de la partie scanneur.

Le problème c'est que l'on pourrait croire que l'application est fournie dans l'API a été insérer dans notre produit sans aucune modification, due à son interface presque identique. Cependant, le code qui roule derrière est très différent et même que la plupart des interactions fournies par l'interface de l'API ne sont plus fonctionnelles, car nos besoins n'incluent pas une configuration en temps réel du scanneur. L'amélioration, ici, serait de créer un nouveau Storyboard, de la connecter sur notre code actuel et ainsi pouvoir se débarrasser de cette interface utilisateur générique.

Prochaines étapes

L'application produite dans le cadre du projet n'étant qu'un prototype et ayant plus d'objectifs implique une poursuite du travail possible et intéressante.

Squelette de noeuds

Le projet consistait en deux objectifs, le premier la création de l'application intégrant les deux technologies. Celui-ci a été complété. Toutefois, le deuxième qui lui était accessoire, la création d'un squelette de nœuds dans le modèle humain, n'a pas pu être complété. Les prochaines étapes seraient donc de compléter celui-ci.

Le développement d'un tel prototype pourrait permettre d'évaluer les hypothèses, HYP2 et HYP3 qui ont été identifiés au début du projet. Ces hypothèses permettront d'adresser les différentes inquiétudes par rapport à la performance d'une application comme celle produite dans le projet dans un contexte réel.

Publication d'une application

Le niveau de déploiement du produit est actuellement seulement un prototype. Donc une fois les deux objectifs complétés, il serait intéressant de déployer une version publique de l'application sur le marché d'Apple. Toutefois, certaines tâches devront être accomplies avant

d'être prêtes à une publication. Des tests devront être ajoutés à l'application. L'export des modèles scanné devra être envoyé sur iCloud pour permettre le transfert entre les appareils de l'usager. Une validation des droits associés aux périphériques produit par Occipital, la compagnie propriétaire du Structure Sensor, devra être faite.

CONCLUSION

Le développement d'application utilisant la réalité augmentée est encore très embryonnaire. Toutefois, la publication de certaines technologies comme le Structure Sensor de la compagnie Occipital et l'ARKit développé pour les appareils Apple amène des nouvelles possibilités. Chacune de ces technologies vient par contre avec ces inconvénients. La première est très peu adopté par les développeurs puisqu'elle demande l'achat d'un périphérique. Une faible communauté de développement limite donc les ressources possibles à la résolution de problème. La seconde introduit des limitations physiques qui peuvent être résolues qu'avec une amélioration physique de l'appareil. De plus, l'ARKit ne peut qu'utiliser des modèles préalablement ajoutés ou importés dans l'application.

Le produit réalisé devait donc combiner les deux technologies pour réduire les risques et les faiblesses de chacune. Le niveau de production visé de l'application n'était qu'une preuve de concept. Un objectif de détection de modèle humain avait été ajouté à l'envergure du projet au début pour concrétiser la preuve de concept. Cet objectif n'était pas nécessaire au statut de succès du projet.

L'application produite au cours de la session valide en effet une combinaison possible des deux technologies. En effet, l'utilisateur est en mesure de scanner un objet de l'environnement réel. Par la suite, celui-ci peut l'introduire dans une scène de réalité augmentée gérée par l'ARKit. Une manipulation simple du modèle est par la suite possible. La solution confirme les possibilités qu'apporte un périphérique comme le Structure Sensor. Elle valide en plus la facilité de développement introduite avec l'ARKit. L'objectif de détection de modèle humain a malheureusement dû être reporté à une poursuite possible du projet.

L'application développée dans le cadre du projet n'est qu'une preuve de concept. Par contre, elle introduit des possibilités pour des domaines d'activités tels que le jeu vidéo et les commerces de détails.

Le produit a été produit au moment où des limitations existent dans les deux technologies. Le ARKit est limité par une détection de profondeur peu performante. Alors, que le Structure Sensor manque grandement de ressource pour le développement d'application. Il est fort probable qu'Apple introduisent des appareils permettant de réaliser des meilleurs captures d'environnement réel. De plus, dans le cas où l'adoption du périphérique Structure Sensor devient importante la compagnie Occipital risque d'introduire une documentation plus complète. Si ces prévisions s'avèrent véridiques, il serait recommandé de choisir une seule solution. Ce changement limitera alors les difficultés de développement.

Bibliographie

Alliance, A. (2001). Manifesto for Agile Software Development. http://agilemanifesto.org/.

Apple. (2018a). ARKit. https://developer.apple.com/documentation/arkit.

Apple. (2018b). AVKit. https://developer.apple.com/documentation/avkit.

Apple. (2018c). Apple Developer Documentation. https://developer.apple.com/documentation.

Apple. (2018d). Model I/O. https://developer.apple.com/documentation/modelio.

Apple. (2018e). OpenGl ES. https://developer.apple.com/documentation/opengles.

Apple. (2018f). SceneKit. https://developer.apple.com/documentation/scenekit.

Best-Practice-Software-Engineering. (2013, Octobre, 04). Delegation Pattern [University of San Francisco, Lecture slides]. http://best-practice-software-engineering.ifs.tuwien.ac.at/patterns/delegation.html.

Clover, J. (2018, Mars, 29). Apple Releases iOS 11.3 With Battery Health Tool, ARKit 1.5, Business Chat, New Animoji, and More [MacRumors, article]. https://www.macrumors.com/2018/03/29/apple-releases-ios-11-3/.

Kremenek, T. (2017, septembre, 19). Swift 4.0 Released ! https://swift.org/blog/swift-4-0-released/.

Marklew, R. [Richard Marklew]. (2013, Février, 10). Importing .obj to blender 2.6 [Forum]. https://blenderartists.org/forum/showthread.php?281322-Importing-obj-files-to-blender-2-6.

Microsoft. (2014, Octobre, 21). WindowsPreview.Kinect Namespace [Documentation]. https://docs.microsoft.com/en-us/previous-versions/windows/kinect/dn758774(v%3dieb.10).

Molitch-Hou, M. (2016, Janvier, 06). Occipital Unveils New 3d Sensor & Mixed Reality Engine at CES [3D Printing Industry, Article]. https://3dprintingindustry.com/news/64337-64337/.

Occipital. (2018, Février, 05). Structure SDK version 0.7.1 [SDK Download Link]. https://developer.structure.io/sdk.

Remi. (2018, mars, 29). 13 millions d'applications ARKit ont été téléchargées en six mois [iPhonote]. https://www.iphonote.com/actu/130432/13-millions-applications-arkit-ont-ete-telechargees-en-six-mois.

Statt, N. (2017, Septembre, 12). Apple shows off breathtaking new augmented reality demos on iPhone 8 [The Verge, Article]. https://www.theverge.com/2017/9/12/16272904/apple-arkit-demo-iphone-augmented-reality-iphone-8.

Thrun, S. & Leonard, J. J. (2008). *Springer Handbook of Robotics*. Berlin, Heidelberg : Springer Berlin Heibdelberg.

Unity. (2016). How do I import Models from my 3D app ? https://docs.unity3d.com/540/Documentation/Manual/HOWTO-importObject.html.

Worley, C. [n6xej]. (2016, septembre, 29). StructureScannerSwift. https://github.com/n6xej/StructureScannerSwift.

ANNEXE I - MISE À JOUR DES EXIGENCES

ID	Statut	Titre	Note
EX1	Complété	Application mobile	
EX2	Complété	Accès caméra	
EX3	Complété	API Structure	
EX4	Complété	Scanner 3D	
EX5	Complété	Insertion de modèle	
EX6	Complété	Transfert de modèle Structure à ARKit	
EX7			
EX8	Complété	Accès au disque	données internes à l'application seulement
EX9	Complété	Sauvegarde sur le disque	données internes à l'application seulement
EX10	Complété	Liste des modèles	
EX11	Annulé	Ouverture sans caméra	Nécessité discutable pour l'état de prototype et fonction primaire de l'application demande la caméra
EX12	Complété	Ouverture sans Structure Sensor	
EX13	Complété	Icon	
EX14	Complété	Données de distance	

EX15	Complété	Transfert de la distance	
EX16	Complété	Présentation de distance	Le cadrillage du plan AR le permet.
EX17	Complété	Interface accueil	
EX18	Complété	Interface Navigation	
EX19	Complété	Interface de scan	
EX20	complété	Interface de manipulation 3D	
EX21	Complété	Interface catalogue	
EX22	Annulé	Interface descriptive d'un modèle	L'utilité de l'interface a été rediscuter.
EX23	Complété	Normes Apple	
EX24	Annulé	Code testable	Les frontière des API sont déjà tester.
EX25	Complété	Swift et Objective-C	
EX26	Complété	Normes de programmation	
EX27	Partielle	Exportation dans un fichier scène	Conversion des modèles scanner à même l'outils de développement possible.
EX28	Partielle	Exportation dans un fichier objet	Sauvegarder dans les fichiers de l'applications peut-être exporter par les développeurs.
EX29	Complété	Temps de démarrage	
EX30	Complété	Fluidité	
EX31	Complété	Exigences de sécurité par l'Apple Store	
EX32	Complété	Données usagers sécurisés	données internes à l'application seulement

EX33	Reporté	Création de nœuds	
EX34	Reporté	Liaison entre les nœuds	
EX35	Reporté	Mouvement de nœuds	
EX36	Reporté	Transfert du mouvement sur le modèle AR	
EX37	Reporté	Détection de squelette dans un modèle (simple)	
EX38	Reporté	Détection de squelette dans un modèle (humain)	
EX39	Reporté	Limitation des mouvements	
EX40	Reporté	Fluidité des squelettes	