Hes·so VALAIS WALLIS
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences Western Switzerland

Travail de bachelor 2012

Filière Informatique de gestion

Use the Kinect platform to increase IPv6 devices awareness in a room

Etudiant : Marc Lyness

Professeur : Dominique Genoud

www.hevs.ch

Préface

Le smart grid est la signification que l'on donne à un réseau de distribution d'électricité intelligent. A l'aide des nouvelles technologies informatiques, son but est d'optimiser la production, la distribution et avant tous, la consommation d'énergie. De nos jours, ce sujet devient une priorité. Au vu du réchauffement planétaire ou encore, de l'augmentation du prix de l'électricité, des solutions doivent rapidement être trouvées.

L'enjeu de ces prochaines années, va être de pouvoir harmoniser l'ensemble de nos composants du quotidien. Par exemple, les panneaux solaire, l'éclairage ou encore, les stores. Pour l'instant, il est difficilement possible de communiquer avec ces différents appareils. Il va falloir trouver un moyen de récupérer leurs données utiles, qui, aideront à mieux planifier notre consommation.

Actuellement, la HES-SO Valais participe au projet européen « Internet of Things 6 »(IoT6)[1]. La recherche vise à étudier l'arrivée de l'« Internet protocole version 6 »(IPv6), qui devrait profiter à l'internet des objets. L'idée derrière cette notion, est de rendre nos composants physiques intelligents, autonomes et capables de prendre seul leurs décisions. Les connectés ensuite sur le même réseau que des entités numérique, leurs offriraient la capacité d'échanger des données.

Page de garde :

Image 1 : page de garde

Source : http://eetimes.com/ContentEETimes/Images/digital/041612/1620diag_pg21.jpg

[1] http://www.iot6.eu/

Résumé

Introduction

Un composant électronique innovateur, la Kinect, sortie début 2012, commence à être commercialisé. Jusqu'à maintenant, elle était utilisée pour détecter les mouvements de personne jouant aux jeux vidéo. Cependant, de nombreux projets vont rapidement la détournée de son utilité première.

Ce travail de bachelor rentre dans le projet IoT6 que mène conjointement la HES-SO du Valais avec la communauté européenne. Le but est, à l'aide du standard IPv6, d'améliorer la communication entre nos différents composants électroniques afin d'en obtenir des données utiles. Il sera ensuite possible, grâce à des décisions prises sur la base de ces informations, d'améliorer la gestion de notre consommation d'énergie.

Ainsi, la Kinect peut-elle être utilisée pour gérer intelligemment une pièce ? Peut-elle nous indiquer le nombre de personne présente dans la pièce afin qu'on puisse adapter le chauffage ? Peut-elle émettre un signal d'urgence si elle détecte quelqu'un allongé au sol ? De nombreux scénarios sont imaginables, mais tous ne seront pas réalisables.

Méthodes

L'approche de ce travail est, d'assimiler, dans un premier temps, cette nouvelle technologie qu'est la Kinect. Comme ses capacités ne sont pas encore réellement connues, la recherche d'informations et la réalisation de nombreux tests seront essentiels.

Au fur à mesure du travail, les discussions avec les clients seront enrichies des découvertes faites. Ainsi, brique par brique, les outils développés produiront au final l'application qu'ils désirent. Les informations récoltées, permettront également, à ceux qui le souhaitent, d'appréhender plus rapidement le développement d'application pour la Kinect.

Conclusion

L'application développée nous permet au final de récolter les informations suivantes : de prendre les mesures de quelqu'un, de connaître le nombre de personne présente dans la pièce, celle qui se déplace, celle qui sont à l'arrêt ou encore, celle qui sont assises. Il également possible de savoir les heures auxquelles les gens sont passés.

Mots-clés

Internet-of-Things ; Senseur ; Kinect ; Surveillance

Table des matières

1. Présentation du travail

1.1. Contexte

Proposé par MM. Bocchi et Genoud, ce travail de bachelor rentre dans le cadre du projet européen *IoT6* auxquelles la HES-SO Valais participe[2]. Les recherches visent à explorer le nouveau standard IPV6 appliqué à l'internet des objets dans des domaines tel que la gestion énergétique intelligente, la E-Santé ou encore les réseaux mobiles.

Ce projet rentre dans le domaine d'activité *eEnergy*[3]. Les travaux consistent aux développements de systèmes d'informations intelligents permettant la récolte d'informations de nos divers appareils électroniques afin d'en gérer efficacement leurs usages.

L'objectif de ce travail sera d'estimer la pertinence d'intégrer ou non une Kinect dans un tel système. Quel genre de données pouvons-nous récolter avec ? Pouvons-nous décider à l'aide de ce périphérique de chauffer une pièce selon le nombre de personnes présentes ou éteindre la lumière si personne ne s'y trouve ?

Pour une durée de 360 heures, cette recherche s'effectue dans le cadre d'un travail de bachelor en informatique de gestion à la HES-SO // Valais.

1.2. Cahier des charges

Dans le but de fixer d'un commun accord les objectifs et de permettre une meilleure planification du travail à rendre, un cahier des charges a été soumis fin mars 2012. L'auteur et le responsable du projet ont lu et accepté ledit document.

Il en est ressorti les phases importantes suivantes :

- Se familiariser avec la Kinect PC et son SDK
- Explorer les capacités de la Kinect aux travers de démos ou ateliers en ligne
- Fournir régulièrement des applications, en tirer les conclusions et les débouchés
- Fournir une application de détection réalisable selon les résultats de l'analyse
- Tirer une conclusion quant à la pertinence d'utiliser ou non une Kinect

[2] http://iig.hevs.ch/valais/iot6.html
[3] http://iig.hevs.ch/valais/eenergy-697.html

2. Comprendre l'environnement Kinect

Dans le cadre de ce chapitre, la Kinect va être présentée afin de mieux comprendre ce composant qui sera au cœur de ce travail. Car si une poignée d'entre nous la possède pour leur console de jeux, qu'est réellement ce nouveau périphérique qui débarque sur nos ordinateurs ?

Nous commencerons par expliquer comment un tel outil a vu le jour, nous verrons notamment à quel point l'engouement suscité autour d'elle fut fulgurant. Ensuite, nous passerons en revue des projets en cours, dans divers domaines, afin de découvrir les idées innovantes proposées.

Nous finirons de nous familiariser avec la Kinect en la décortiquant. Composant hardware, interface de programmation (API), informations obtenues ou encore contraintes, cette analyse va nous permettre de mieux comprendre, par la suite, les décisions à prendre au niveau de l'application.

Image 2 : la Kinect

Source : http://i.msdn.microsoft.com/dynimg/IC568992.png

2.1. Comment tout a commencé

Un secret bien gardé jusqu'à présent est dévoilé lors de l'E3[4] 2009, le salon international du jeu vidéo. Le projet nommé à l'époque « Natal », signifiant « naissance », représente aux yeux de Microsoft la nouvelle innovation en termes de divertissement vidéo ludique.

Car si Nintendo a longtemps été le pionner dans les capteurs de mouvement avec sa Wii, la firme américaine espère bien la concurrencer en étendant sa base de clients console constituée, jusqu'à maintenant, essentiellement de joueurs masculins âgés entre 18 et 30 ans [1].

Kudo Tsunoda, directeur création pour la Xbox, nous démontre lors de son intervention à l'E3 comment la Kinect facilite l'expérience avec les jeux vidéo. Selon lui, ce qui retient souvent les gens de jouer est le fait d'utiliser une manette parfois trop compliquée à leurs yeux [2].

[4] http://www.e3expo.com/

La démo est efficace, la Kinect parvient à détecter la personne qui se trouve devant la console, et est capable d'afficher son avatar qui se met à gesticuler comme elle. Deux démos d'applications ensuite nous montrent ensuite, d'abord, le renvoi de balles virtuelles à l'aide de nos mouvements, puis, une personne citant à haute voix la couleur qu'il désire pour peintre sur sa toile virtuelle [2].

Microsoft fait fort : non seulement elle se place sur le même marché que ces concurrents mais s'affranchit des manettes habituellement nécessaires et introduit une toute nouvelle façon de communiquer avec sa console de salon.

La technologie de l'époque permet de traquer jusqu'à quatre personnes et propose la lecture de 48 points de notre squelette. Alex Kipman, directeur du projet Natal, y voit une aubaine pour les développeurs de jeux d'intégrer cette technologie afin d'enrichir l'expérience [3].

Lors de l'E3 2010, Microsoft met en scène, de manière dantesque, l'annonce du nouveau nom du projet : Kinect. Mélange de « Kinetic » (énergie lié au mouvement) et « Connection », représentant bien la technologie du produit, selon les dires de Stephen Toulouse [4] [5].

Le budget marketing est colossal, un demi-milliard est prévu uniquement pour la promouvoir. En comparaison, la console Xbox avait reçu l'équivalent en 2000, principalement pour réduire son coût de lancement. Et le succès est au rendez-vous. Entre le 4 novembre 2010 et le 3 janvier 2011, la Kinect se vend à plus de 8 millions d'exemplaires, l'équivalent de 133'333 unités par jour, un record faisant d'elle le composant électronique le plus vite vendu dans un laps de temps aussi court [6] [7].

Un des facteurs de ce succès provient notamment de l'engouement de la communauté du web. A peine sortie sur console, la Kinect était déjà portée sur Linux, Mac ou encore Windows, répondant ainsi à l'appel d'une société offrant 3'000 dollars à la première personne fournissant les pilotes. Intelligemment, Microsoft n'ira pas à l'encontre de ces différents projets et y voit l'opportunité de faire germer sur la toile de nombreuses idées et projet pour son périphérique [8].

Microsoft finira par proposer elle-même une version PC de la Kinect, le 1 février 2012, avec un kit de développement(SDK) disponible gratuitement [24]. Ainsi il reprenne plus ou moins la main sur les projets qui jusqu'à maintenant se basait sur des programmes de tiers permettant de cracker la Kinect de la Xbox live. Afin d'entretenir l'engouement autour de son produit, Microsoft propose des stages d'initialisation, des programmes scolaires pour les jeunes étudiants, soutiennent des projets innovateurs ou encore, créent des partenariats avec de grands groupes industriels.

Il est impossible à ce jour de dire si le buzz va continuer mais fort est de constater que Microsoft, et il faut le souligner, elle ouvre grand les portes pour le développement sur un bijou technologique coûtant la modique somme de 250$.

2.2. Les recherches actuelles

Quand aujourd'hui vous parlez avec quelqu'un d'écran tactile ou de reconnaissance vocale, nul doute qu'il vous évoquera les produits d'« Apple ». Mais peu de gens savent que Microsoft y travaille depuis de nombreuse années, n'ayant jamais vraiment trouvé de marché pour ses produits.

Chef de l'innovation et de la recherche chez Microsoft, Craig Mundie nous donne son point de vue sur la question. Selon lui, les tablettes tactiles ne sont qu'une étape. Demain, nous serons capables de parler directement à l'ordinateur ou lui faire un signe pour lui donner un ordre compréhensible pour ce dernier, le clavier et la souris eux, devraient disparaitre. « Vous direz « Ou peut-on manger ce soir ? » au lieux de taper dans un moteur de recherche « Meilleurs restaurants ? » » [9].

Avec la Kinect, l'avantage clairement identifié est le fait de pouvoir offrir une interaction homme-machine encore plus poussée. Les designers et développeurs parlent de «Natural User Interface» (NUI) quand l'interaction devient invisible. Ici, les périphériques devraient donc disparaître, remplacés par des ordres vocaux ou des gestes de l'utilisateur.

Les idées d'implémentation ne manquent pas et de nombreuses entreprises s'y essaient dans le cadre de projets expérimentaux. La banque « LCL » du « Crédit Agricole » en France a développé des bornes interactives qui à l'aide de la reconnaissance des mouvements permet l'accès à ses services et produits. Un magasin britannique[5] a pour sa part développé une cabine virtuelle permettant l'essayage virtuel de vêtements [10] [11].

Image 3 : Bodymetrics propose la cabine virtuelle

Source : http://www.instablogsimages.com/1/2012/01/16/bodymetrics_kinect_based_virtual_shopping_application_zvalv.jpg

[5] http://bodymetrics.com

Demain, nos présentations seront animées par nos gestes ou notre voix, notre physiothérapeute sera équipé d'une Kinect pour s'assurer que notre geste est le bon ou nous serons à même de scanner notre corps afin de commander des vêtements exactement à notre taille. Microsoft, dans ses vidéos de promotion[6], va même jusqu'à nous faire croire que la Kinect sera capable de simuler un violon sur lequel nous pourrions jouer par la gestuelle.

Fantaisiste ? L'avenir en témoignera. Le gros du travail maintenant est de connaître réellement la capacité d'un tel composant pour fournir les services et applications utiles au client final. Microsoft y travaille activement, la Xbox en est une illustration. Des services tels que la navigation par la voix de ces médias ou la perspective de la combinaison avec des lunettes 3D pour la prochaine version de la console semblent prometteurs [12].

2.2.1 Le projet KinectAccelerator

Début Novembre 2011, Microsoft met au concours une invitation pour venir développer son application dans ses bureaux de Seattle. Pensant recevoir une centaine de candidature, ce seront plus de 500 qui seront soumises. Finalement, 11 projets seront retenus.

Le principe du projet KinectAccelerator[7] est le suivant : trois mois pour développer son business plan et proposer une démo. Sans compter la gratuité des locaux, les teams présents bénéficient de la présence de conseillers de chez Microsoft que ce soit dans les secteurs marketing, management voire encore mieux, les gens qui travaillent directement sur le SDK de la Kinect.

Au vu des images et vidéos fournies par le site de Microsoft[8], cette aventure humaine semble avoir porté ses fruits et permis de faire la promotion d'idées innovantes. Fin juin 2012, la « Demo Day » voyait les équipes présenter leurs solutions aux nombreux investisseurs présents. Passons en revue quelques-uns de ces projets.

Ikkos

Ikkos[9] développe une application dans le domaine sportif. La problématique soulevée est la suivante : un coach sportif doit faire comprendre à son élève, par la parole, comment perfectionner son geste. Cette communication imagée perd en information, car une personne essaie d'expliquer ce qu'elle a vu, l'autre essaie de se faire une idée et de corriger ensuite le mouvement.

[6] http://www.xbox.com/en-US/Kinect/Kinect-Effect
[7] http://www.microsoft.com/bizspark/kinectaccelerator
[8] http://www.microsoft.com/en-us/kinectforwindows/discover/gallery.aspx
[9] http://www.ikkostraining.com

Le programme permet à l'élève de voir directement le mouvement qu'il a réalisé et de le comparer avec le geste parfait calqué par-dessus. Ainsi en voyant directement le résultat, le cerveau assimile beaucoup plus facilement. Actuellement utilisé dans le cadre de la natation, il pourrait s'appliquer à d'autres sports, comme le golf par exemple.

Jintronix

Jintronix[10] développe une application dans le domaine médical. Il crée des exercices de physiothérapie dans des mondes virtuels. Des lunettes 3D permettent de voir des formes et des gants d'interagir avec. La Kinect, elle, donne des informations sur la motricité globale des membres inférieurs et supérieurs du patient.

L'idée est de permettre aux victimes d'accidents vasculaires cérébraux de réapprendre les mouvements de tous les jours. Le fait que les patients puissent voir les progrès réalisés est une manière efficace de stimuler leur participation et leur motivation.

Kimetric

Kimetric[11] est un outil pour les commerces de détails. Branchez une Kinect dans votre vitrine et laissez le programme vous dire le trafic de client. Ou encore, traquez leur parcourt dans votre magasin et le temps qu'ils y sont restés.

Il vise aussi à se servir des infos de la Kinect afin de déterminer le sexe, la grandeur, la taille et l'âge de la personne. Ensuite, à l'aide d'un écran digital, la publicité serait adaptée à la personne se trouvant en face, offrant ainsi une nouvelle façon d'interagir avec le client.

Styku

Styku[12] fait dans le textile. A l'aide de la Kinect, il reproduit en 3D les formes exactes d'une personne. Avec ces informations, il crée de toutes pièces le vêtement souhaité pour le vendre ensuite au client final. Cela prévisage un avenir radieux pour la vente en ligne de vêtements.

Übi

Übi[13] permet de transformer n'importe quel projecteur en un écran 3D tactile géant, que ce soit projeté sur un mur ou sur une table. Son application touche plusieurs domaines tels que l'éducation, les services publics, les hôpitaux et plus encore. La façon de consulter des horaires, par exemple, pourrait évoluer vers des panneaux d'affichages interactifs et faciles à mettre en place.

[10] http://www.jintronix.com/site
[11] http://www.kimetric.com
[12] http://www.styku.com/business
[13] http://www.ubi-interactive.com

2.2.2 La robotique

Parlons d'appareils robotiques qui tirent avantage de la Kinect. Ces appareils peuvent maintenant être dotés d'une véritable paire d'yeux qui seront chargés de scanner les environs. Willow Garage[14] est une entreprise qui propose ces robots et certains se voient maintenant équipés d'une Kinect.

Image 4 : Turtle Bot

Source : http://www.willowgarage.com/turtlebot

Le premier modèle qui est développé se nomme le « TurtleBot ». Cette petite table est montée sur une base motorisée lui permettant de se déplacer. Couplé à une application développée et une Kinect, de nombreuses possibilités sont imaginables.

Ainsi, elle pourrait cartographier son environnement et programmer son parcours de la cuisine au salon par exemple. Une autre proposition serait de lui demander de traquer une personne afin de la suivre. Alors peut-être que l'intérêt est dur à voir sur une simple table, mais appliquer le même principe pour une chaise roulante, devient soudainement beaucoup plus intéressant.

Un autre modèle développé est utilisé dans le cadre d'un projet au « Massachussets Institute of Technology » (MIT) : le PR2. Ce robot à deux bras mécaniques peut se déplacer ou encore saisir divers objets. Mais jusqu'à maintenant, tout cela manquait de précision et les senseurs utilisés coutaient dans les 7'000 $. La Kinect y a été intégrée avec succès et ce robot est maintenant capable de préparer tout seul des cookies [13].

Image 5 : PR2 faisant des cookies

Source : http://news.cnet.com/2300-11386_3-10010451.html?tag=mncol

[14] http://www.willowgarage.com

2.2.3 La Suisse dans tout ça ?

Les universités et hautes écoles suisses ont embrassé avec ferveur les débuts de la Kinect Xbox en développant rapidement à l'aide des utilitaires permettant de la hacker. Voici divers projets intéressants se déroulant par chez nous.

Doctorant au département électricité de l'EPFL à Lausanne, Alexandre Alahi a conçu un algorithme dans le but de créer un système de surveillance. A l'aide de deux Kinect connectées, il se sert des données de profondeur afin de détecter les mouvements de personnes, même dans la pénombre. Il se réjouit d'accéder à un tel senseur pour un prix dérisoire [14] [15].

A l'institut de médecine légale de l'Université de Berne est développé un projet sous le nom de « VirtopsyProject ». Il utilise différents codes et applications open source sous linux dans le cadre de l'imagerie médicale. Il permet aux chirurgiens en salle d'opération de manipuler des documents à l'écran sans contact avec une souris ou un clavier. Les possibilités en termes de scannage sont aussi à l'étude afin de savoir si la Kinect pourrait remplacer de coûteux appareils dans la modélisation 3D de corps lors des autopsies [15] [16].

Lors du transat festival 2012[15] se déroulant à Lausanne, j'ai eu la surprise de découvrir des projets artistiques incorporant la Kinect. Sur l'image 6, on peut voir les différents stands visités lors de la manifestation.

Image 6 : images prise durant le Transat Festival 2012 à Lausanne

Au point 1, le projet réalisé par une étudiante en Master à la haute école d'art et de design de Genève, propose aux visiteurs de marcher sur un sol qui déclenche des animations sur leur passage. J'ai rapidement pu constater les limites de l'angle de la Kinect ne couvrant pas l'entier de la surface prévue.

[15] http://www.transat-festival.ch

Parlons maintenant de deux projets réalisés dans le cadre d'un partenariat entre l'EPFL et l'ECAL qui vise à promouvoir la relation entre l'ingénierie et le design[16]. Au point 2, le projet « Body Impact » de Thomas Eberwein. Notre corps se trouve projeté à l'écran sous forme d'un flux de particules et chacun de nos gestes crée des turbulences et laisse des traces.

Au point 3, le projet « Tattooar » de Cem Sever, Happy Pets et Marc Hussler nous propose un concept de miroir digital. En appliquant un tatouage sur notre peau, il s'anime à l'écran lorsqu' il est présenté devant l'appareil. La démo m'a fait réaliser certaines limites de la Kinect. En effet, ma peau très claire qui reflétait à cause d'une lumière mal placée derrière l'appareil ou encore des problèmes de calibrages de la position du bras empêchait l'animation de se lancer à l'écran. On réalise vite comme il est difficile de répondre à toutes les situations.

2.3. La Kinect

A travers ce chapitre, la technologie de la Kinect développée par la compagnie « PrimeSense [17][18]», leader dans la détection 3D et de la reconnaissance, dont le partenariat avec Microsoft a permis la production en masse de ce périphérique, va être abordée. Les informations qui suivent, sont basées sur les renseignements fournis sur le site de Microsoft[19].

2.3.1 Prérequis

Afin d'utiliser la Kinect, il vous faut télécharger une version du SDK se trouvant à l'adresse suivante :

http://www.microsoft.com/en-us/kinectforwindows/develop/developer-downloads.aspx

Le SDK fournit les drivers permettant de communiquer avec la Kinect. Il offre également une interface de programmation (API). Un ensemble d'outils, le SDK tools kit, est aussi disponible par le biais d'un second exécutable d'installation. Il fournit différents codes de démonstrations, de la documentation ainsi qu'un logiciel permettant l'enregistrement de séquences prises avec la Kinect.

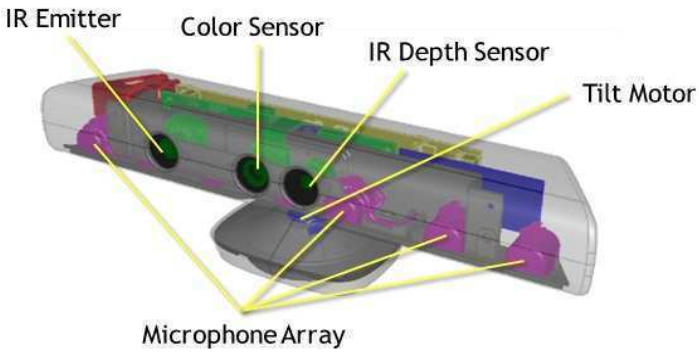
N'hésitez pas à consulter le site pour connaître les différents systèmes supportés et outils nécessaires au développement.

[16] http://www.epfl-ecal-lab.ch/page-69392-en.html
[17] http://www.primesense.com
[18] http://www.wipo.int/patentscope/search/en/WO2007043036
[19] http://msdn.microsoft.com/en-us/library/hh855352.aspx

2.3.2 KinectSensors

La « Kinect Sensors » est le nom donné au périphérique physique. Elle contient différentes caméras et quatre micros le long de sa forme verticale qui repose sur une base motorisée.

Image 7 : Kinect sensors



Source : http://i.msdn.microsoft.com/dynimg/IC584396.png

Afin de connecter une Kinect, un branchement de « Bus universel en série » (USB), ainsi qu'un adaptateur spécifique fourni avec, est nécessaire. Ce dernier, la relie à la fois au PC via une prise USB standard et par une prise de courant pour l'alimenter.

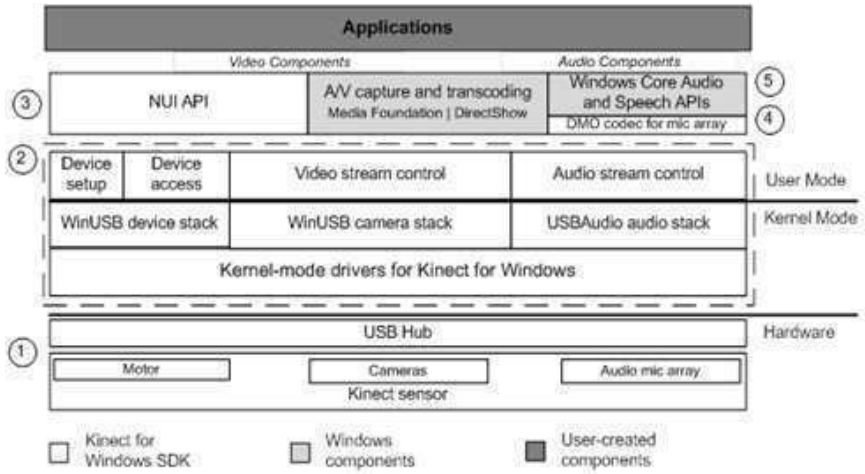
2.3.3 Architecture

Comme on peut le voir, l'architecture est divisée en trois couches. La première, celle évoquée au point 2.3.1, est le hardware (point 1, image 8). On peut voir que le moteur, les micros et les caméras sont connectés via un hub fournissant une seule sortie USB. Ceci explique le connecteur dédié fourni avec la Kinect.

Viens ensuite le mode kernel (point 2, image 8) qui permet de faire l'interface entre le hardware et le logiciel. L'équipe Microsoft c'est basé sur le Kernel de la Xbox et l'a adapté pour les ordinateurs.

Finalement, la couche « User Mode » (point 2, image8) permet de communiquer avec la Kinect (Device setup et access) et de traiter les deux types de flux : la vidéo et le son. Les cases grisées (point 5, image8) représentent les drivers Windows déjà présents sur la machine. Une Kinect branchée sur un ordinateur n'ayant aucun SDK installé, sera automatiquement reconnu comme un périphérique audio et rien d'autre.

Image 8 : architecture



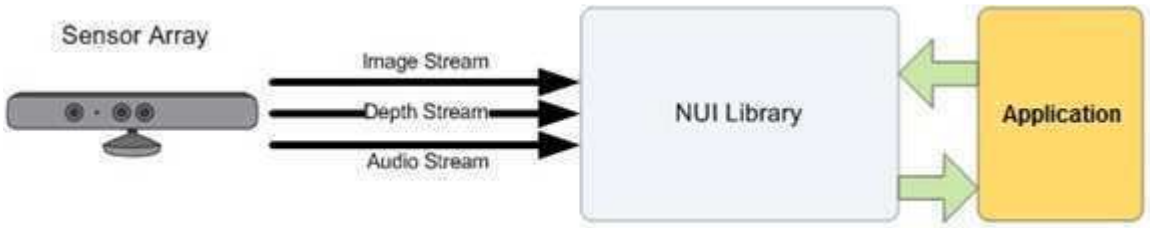
Source : http://i.msdn.microsoft.com/dynimg/IC568989.png

La version « DirectX Media Object » (DMO) (point 4, image 8) offre des pilotes avancés pour la prise de son précis, en isolant les bruits environnants de la source qui émet.

Le « NUI API » (point 3, image8), lui, nous fournis l'accès aux classes et méthodes afin de traiter les flux reçus de la Kinect. Plus d'information au point 2.3.5.

2.3.4 Les Flux

Image 9 : les différents flux disponibles



Source : http://i.msdn.microsoft.com/dynimg/IC568988.png

Une fois l'installation du SDK terminée, la Kinect est prête à être utilisée. Elle renvoie trois genres de données, visible sur l'image 9, à activer dans l'application selon les besoins. Voici une description des trois flux à disposition :

Flux son

Les micros, au nombre de quatre, ont été conçus à la base pour répondre aux contraintes de la Xbox : un joueur s'adressant à la console se trouve généralement à quelques mètres de celle-ci, contrairement à un micro habituel se trouvant près de notre bouche. Comme nous avons pu le voir au point 2.3.3, les pilotes fournis, combinés à la qualité des micros, permettent une écoute précise d'une source sonore.

Hormis le fait de pouvoir enregistrer, les micros sont là avant tout pour permettre de passer des commandes vocales à la Kinect. Microsoft se sert de ses paquets de langue et les incorpore au fil de ses releases du SDK de la Kinect. Si actuellement des langues comme l'anglais ou encore le français sont disponibles, il sera prochainement possible d'avoir accès à des accents spécifiques à certaines régions.

Restant toutefois prudents : ce qui semble un outil fabuleux à implémenter peut s'avérer peu fiable. En effet, après plusieurs tests des applications fournies par le SDK, fort est de constater qu'il ne comprend pas toujours les commandes passées. Néanmoins, les micros intégrés restent de très bonne facture et nul doute qu'au fil du temps le dispositif sera encore amélioré.

Flux d'image

Servant en premier lieu à prendre des photos des joueurs en action lors de parties sur la console Xbox, la caméra « Color Sensor » de la Kinect nous permet de recevoir des images aux résolutions et tailles suivantes :

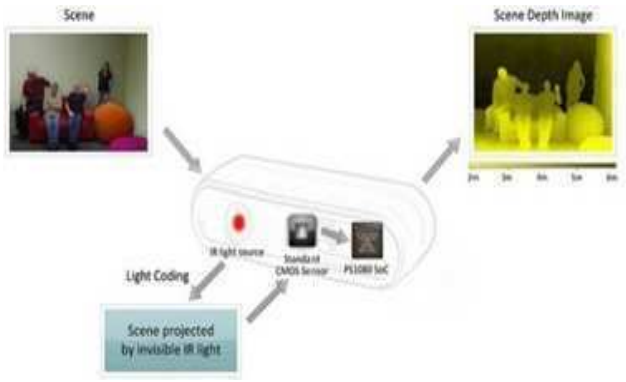
- 12 FPS: 1280X960 RGB
- 15 FPS: Raw YUV 640x480
- 30 FPS: 640x480

Un peu comme avec une webcam, accéder à ces données permet d'afficher ce que voit la Kinect en créant une image avec son flux. Selon votre application, choisissez les différentes compressions de données (RGB, YUV) et résolutions pour améliorer leur transfert entre le périphérique et votre ordinateur.

Flux de profondeur

Les deux caméras restantes, la « IR Emitter » et la « IR Depth Sensor », fournissent la majorité des informations. Elles permettent de cartographier l'espace dans le champ de vision de la Kinect en obtenant la distance entre le périphérique et chaque obstacle de la pièce.

Image 10 : fonctionnement du senseur de profondeur



Source : http://download.gameblog.fr/images/blogs/20532/21839.jpg

En quelques mots voici comment elle procède. La première caméra, bombarde des faisceaux qui, recouvrent la pièce d'un masque respectant un certain modèle (pattern). Si on filme la scène à l'aide d'une caméra infrarouge, une multitude de points apparait. La seconde elle, analyse la quantité de rayonnement infrarouge réfléchie vers la caméra. Plus la densité est forte plus l'obstacle sera près et inversement, plus elle sera faible, plus l'obstacle sera éloigné. Elle se sert donc d'un capteur infrarouge mais non pas pour la chaleur thermique habituellement utilisée [17] [18].

Les images par seconde retournées fournissent pour chaque pixel la distance cartésienne (en millimètres) entre la caméra et l'objet le plus proche dans un système de coordonnées X/Y de la taille du champ de vision de la Kinect. Les résolutions sont les suivantes :

- 30 FPS: 80x60, 320x240, 640x480

A l'aide de ces informations nous pouvons imaginer, entre autres, modéliser en 3D, traquer un mouvement ou encore ignorer certaines zones de l'image.

2.3.5 Nui API

L'API de la Kinect fourni par Microsoft possède différents outils pour son utilisation dans une application. Notons que les paragraphes qui suivent sont valables pour un développement en C#. Mais que ceux qui désirent programmer en C++ se rassurent, il est aussi possible de le faire dans ce langage.

Pour tout complément, je vous invite à vous référer au « Microsoft Developer Network » (msdn)[20] de Microsoft qui offre des informations détaillées sur l'implémentation d'une Kinect Sensors, du traitement des flux, de l'utilisation du squelette ou encore la totalité des classes et méthodes disponible dans l'API.

[20] http://msdn.microsoft.com/en-us/library/hh855347.aspx

L'essentiel pour démarrer à développer rapidement, les classes, les méthodes, vont vous être présentés ci-dessous.

Pour commencer, il faut instancier un composant de type « KinectSensor ». Le plus simple actuellement pour l'associer à un périphérique physique est de boucler sur une collection de type« kinectSensors » et de sélectionner celle ayant le statut de connecté. Pour avoir accès aux outils du SDK, il ne faut pas oublier d'ajouter dans la librairie du projet la « Microsoft.Kinect.dll » qui se trouve dans le répertoire SDK, créé lors de l'installation, dans le dossier « Assemblies ».

Cet objet, permet d'activer les flux, vu au point 2.3.4, que l'on souhaite traiter. L'ordre à de l'importance, car l'activation du traçage du squelette, par exemple, nécessite le contenu fournit par le flux de profondeur. Une mauvaise manipulation résulte par le non-retour d'information.

Tableau 1 : activer les flux

```
// Sensor stream activation ************************************************************************

// Turn on stream to receive color data streams
this.sensor.ColorStream.Enable(ColorImageFormat.RgbResolution640x480Fps30);
// Turn on stream to receive depth data streams
this.sensor.DepthStream.Enable(DepthImageFormat.Resolution640x480Fps30);
// Turn on stream to receice skeleton information
this.sensor.SkeletonStream.Enable();
// ************************************************************************************************
```

Une fois activées, nous devons traiter l'image reçue en donnée brut. Deux types d'action sont alors possibles. La première, « OpenNexFrame », permet d'obtenir la prochaine valeur fourni par le flux voire même, de passer en millisecondes le temps à l'attendre. La seconde, « AllFramesReady », permet de créer une exécution de code à chaque nouvelle donnée disponible des flux activés. Il est aussi possible de le faire pour chaque flux séparément.

Tableau 2 : gérer les informations reçues

```
// Event handler **********************************************************************************

// Color streams : raise an event each time a color frame data is available
this.sensor.ColorFrameReady += new EventHandler<ColorImageFrameReadyEventArgs>(sensor_ColorFrameReady);

/// <summary>
/// Manage and store color frame data
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void sensor_ColorFrameReady(object sender, ColorImageFrameReadyEventArgs e)
{
    using (ColorImageFrame colorFrame = e.OpenColorImageFrame())
    {
        this.ColorImageFrame.Source = this.frameViewerController.colorFrameImage(colorFrame, "ColorImageFrame");
    }
}
```

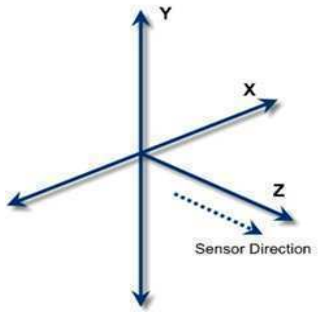
Finalement, il vous faudra démarrer la « Sensor » au moyen de la méthode Start(). Pour l'arrêter, la méthode Stop() est disponible.

Skeleton

Cette classe, qui traite le flux de profondeur, permet de traquer la position (coordonnées X,Y,Z) d'un être humain (impossible actuellement d'appliquer sur toute autre espèce vivante). Afin d'obtenir les informations relatives aux squelettes il vous faut l'activer, comme on peut le voir sur le tableau 1.

Ainsi, à chaque donnée de profondeur disponible, une pour les squelettes l'est également. Le traçage est actuellement limité à six personnes mais ceci évoluera sûrement dans les futurs SDK. Pour chaque personne tracée, un ID unique est fourni. Un squelette spécifique sera accessible à l'aide de la classe de la « NUI API » de Microsoft, « Skeleton ».

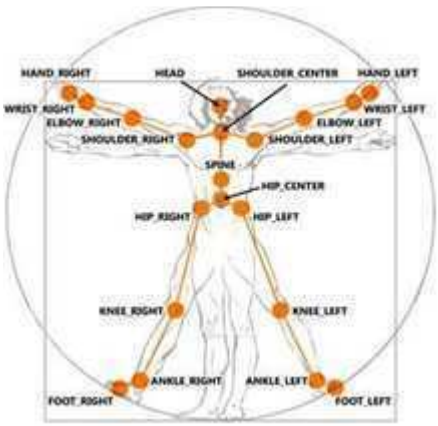
Image 11 : système de coordonnées



Source : http://i.msdn.microsoft.com/dynimg/IC534689.png

Deux statuts de traçage sont dès lors possibles : un actif et un autre passif. Le statut actif permet de récupérer la totalité des points du corps de la personne, au statut passif qui, lui, ne fournit que la position au niveau du torse. Chaque point donne la position X,Y,Z. Le plan de coordonnées utilisé est visible sur l'image 11.

Image 12 : points du squelette disponible



Source : http://www.iprogrammer.info/images/stories/Core/Hardware/KinectSkeleton/bodyparts.jpg

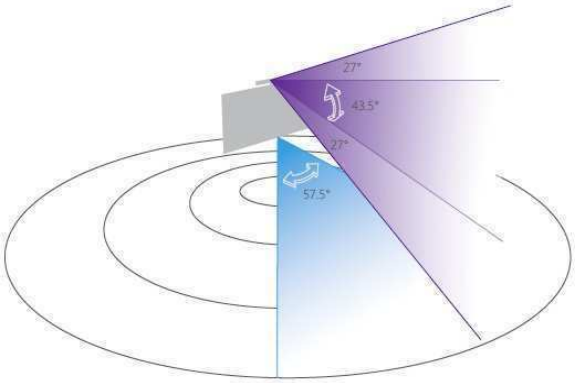
Afin de calculer au mieux la position du squelette, l'API utilise un algorithme pour connaitre l'emplacement du sol. Une personne se trouvant au milieu du champ de la Kinect sera mieux tracée que sur les côtés. Les points du squelette sont difficilement calculés lorsque qu'on ne se tient pas de front ou de dos devant la Kinect, où encore, si nos membres ne sont pas clairement discernables. Une position assise, par exemple, prête à confusion, beaucoup de points se mélangeant par leur proximité.

2.3.6 Contraintes

Voici quelques recommandations afin d'éviter d'endommager la Kinect :

- Toujours la poser sur un emplacement stable qui ne vibre pas.
- Ne pas la placer sur ou devant des haut-parleurs ou sur une surface bruyante.
- La déplacer en la prenant au niveau de la base.
- Ne pas ajuster l'angle horizontal à la main mais à l'aide du moteur intégré.
- Ne pas la placer en face d'une source lumineuse trop forte.
- Ne pas la placer près de source de chaleur (elle supporte entre 5° et 35° degrés Celsius).

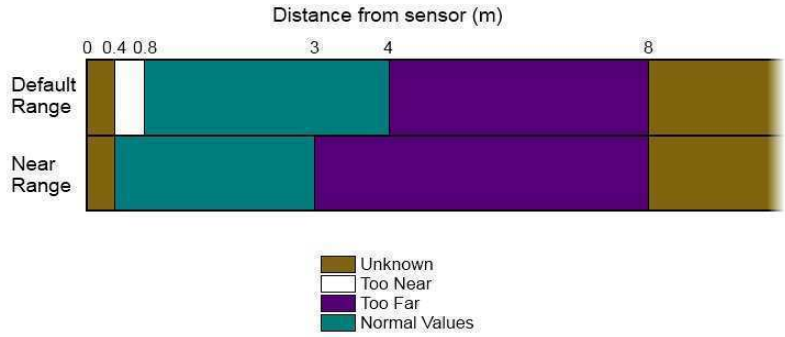
Image 13 : angle de vision de la Kinect



Source : http://go.microsoft.com/fwlink/?LinkID=247735

La base motorisée permet un déplacement de plus ou moins 27 degrés verticalement pour un angle de vue de 43.5 degrés et de 57.5 degrés horizontalement, visible sur l'image 13. Sur l'image 14, on peut voir les différentes distances que gère la Kinect. Elle peut diminuer selon l'angle opté pour la Kinect.

Image 14 : distance de traçabilité



Source : http://i.msdn.microsoft.com/dynimg/IC568993.png

Il est recommandé de placé la Kinect de manière à ce qu'elle puisse voir le sol afin d'en améliorer le traçage. Des objets entre la Kinect et la personne, des vêtements amples ou encore réfléchissants réduisent fortement les performances. Le fait d'être masqué un court instant ou de sortir du champ de vision fera perdre l'ID du squelette tracé. Lorsqu'elle sera à nouveau visible, un nouvelle ID lui sera attribué.

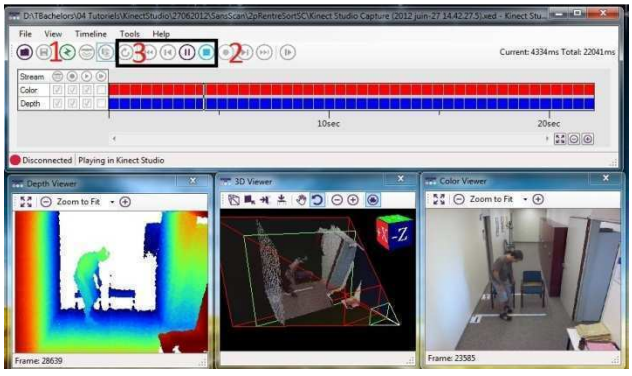
2.3.7 Les outils disponibles

Kinect Studio

Un studio d'enregistrement fait son apparition à partir de la version 1.5 du SDK Kinect de Microsoft. Car si jusqu'à maintenant il fallait à chaque fois reproduire une scène de test, se lever ou avoir des cobayes pour tester le programme, voyons comment ce studio va faciliter notre développement.

Ce studio va nous permettre de sauvegarder des séquences de données. Comme vous pouvez le constater sur l'image 15, le programme peut lire des données préenregistrées ou des images provenant directement d'une Kinect connectée. L'icône sous forme d'éclair au point 1 permet d'associer vos prochaines séquences à une application récoltant les informations.

Image 15 : le logiciel Kinect studio



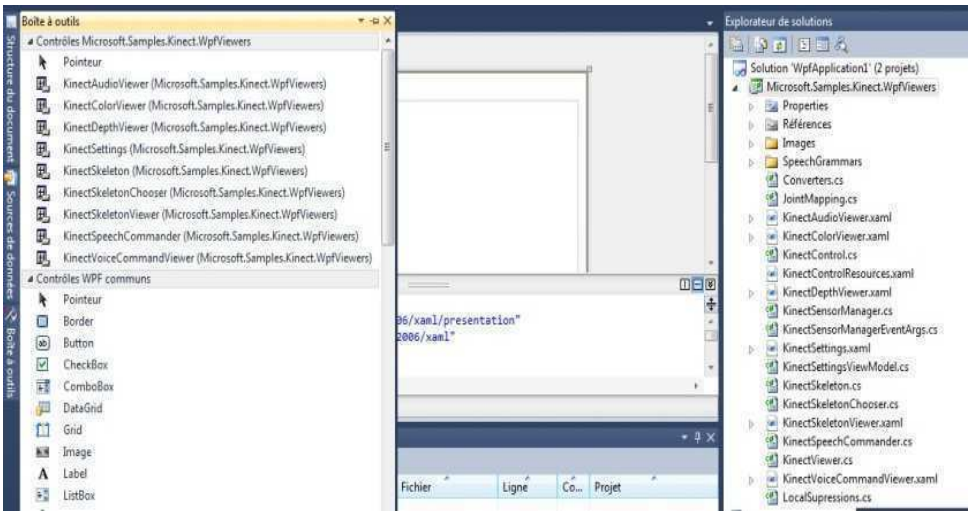
Lorsqu'une application tourne et est associée au studio, l'icône au point 2 sera disponible pour lancer un enregistrement. Une fois terminé, vous pourrez sauvegarder le fichier qui sera de format XED implémenté par la Kinect Studio. Pour une application qui sauvegarde tous les flux disponibles, il faut compter un fichier de 600'000 Kilobytes pour un enregistrement de vingt secondes.

Le groupe d'icônes au point 3 offre des fonctionnalités comme de passer la séquence en boucle, la mettre en pause ou l'arrêter. La grande force de ce studio est de pouvoir repasser les enregistrements dans n'importe quelle autre application que vous développerez. Il est donc fortement conseillé, lors d'enregistrements, d'utiliser un programme qui récolte le maximum d'informations pour les avoir ensuite à disposition. L'exemple « Kinect Explorer » fourni avec le SDK s'y prête bien.

Microsoft.Samples.Kinect.WpfViewers

Dans la démo, « KinectExplorer », fournit avec le SDK de la Kinect, on peut voir un concentré de composants graphiques qui permettent de manipuler les différentes flux. Microsoft nous met à disposition ces outils.

Image 16 : aperçu des composants



Afin d'en profiter il faut ajouter un projet existant, qui se trouve dans le dossier « *\Microsoft SDKs\Kinect\Developer Toolkit v1.5.1\Samples\C# », du nom de KinectWpfViewers. Une fois inclus à votre projet, des composants deviennent disponibles dans la boîte à outils de Visual Studio, comme on peut le voir sur l'image 16.

Le fait d'utiliser cet outil dans ce projet n'a pas été retenu. En effet, le temps passé à comprendre le fonctionnement de ces composants, ne valait pas la peine par rapport à leur réelle utilité. Je n'utilisai que le quart des outils disponibles. A noter encore, que lors du passage de la version SDK 1.0 à la version 1.5, la majorité de mon code avait cessez de fonctionner. Dès lors, j'ai préféré développer mes propres outils.

Microsoft.Kinect.Toolkit

Ce toolkit est apparu lors de la version 1.5 du SDK de la Kinect. Pour l'utiliser, il faut ajouter un projet existant, se trouvant dans le même répertoire que précédemment, du nom de « Microsoft.Kinect.Toolkit ». Il offre un composant permettant de gérer la connexion d'une Kinect.

Microsoft.Kinect.Toolkit.FaceTracking

Ce toolkit est également arrivé lors de la version 1.5 du SDK de la Kinect. Pour l'utiliser, il faut ajouter un projet existant, se trouvant dans le même répertoire que précédemment, du nom de « Microsoft.Kinect.Toolkit.FaceTracking ». Il offre tous les outils pour traquer un visage humain.

2.4. Alternatives

2.4.1 OpenNI

Comme discuté plus tôt dans ce document, PrimeSense est la compagnie qui a fourni la technologie à Microsoft afin de produire la Kinect pour la Xbox. Mais de son coté, elle a également mis à disposition son propre outils.

Le Framework OpenNI[21] a pour but de regrouper les nombreuses API existantes permettant la création d'interfaces NUI. Il est utilisable actuellement sous Windows, Ubuntu et MacOSX. Afin d'avoir les fonctionnalités comme traquer des gens, des gestes de la main ou encore passer des commandes vocales, OpenNI propose leur Middleware se nommant « NITE ». PrimeSense est également associé avec ASUS, qui, fournit un composant matériel, XtionPro[22], semblable à une Kinect.

Microsoft SDK VS OpenNI/NITE

Après la lecture de divers commentaires sur des forums, on peut mettre le doigt sur quelques différences, qui, avec le temps, disparaîtront. A l'avantage d'OpenNI, on peut citer le fait qu'il soit multiplateforme. Il a aussi beaucoup plus de mise à jour ou de projets en ligne qui l'utilisent, du fait qu'il soit présent depuis plus longtemps. Il peut également être utilisé avec la Kinect, la Kinect Xbox ou encore la Xtion Pro, là où le SDK ne fonctionne qu'avec du matériel Microsoft [19].

[21] http://openni.org
[22] http://event.asus.com/wavi/product/WAVI_Pro.aspx

Le SDK de Microsoft propose le traçage du squelette en mode assis, le « near » mode ou encore le traçage de points sur le visage. Il fournit beaucoup de documentation, de démos et des outils tels que l'obtention de la position des points d'un squelette là où il faut tout faire soi-même sur OpenNI. Il permet aussi de développer en C#, qui reste un langage plus abordable, et non pas uniquement en C++

Au final, pas de grande différence. Je pense que le SDK de Microsoft reste tout de même le plus simple à prendre en main et à obtenir des résultats rapidement étant données au vu des langages proposé et de la documentation fourni.

2.4.2 Java/Mac OS

Jusqu'à maintenant, les outils discutés pour utiliser la Kinect ne proposent que peu de solutions pour les développeurs souhaitant travailler sur un système Mac OS. Les deux projets suivants vont résoudre ce problème.

OpenKinect

Ce projet a été développé par Daniel Shiffman, Hector Martin, et la communauté OpenFramework[23]. Il s'agit d'un wrapper en Java qui offre l'accès aux données de la Kinect, tels que le color image ou encore le depth image [20].

SimpleOpenNI

Ce projet est conçu par Max Rheiner[24], professeur et chercheur à l'université d'art de Zürich dans les domaines de l'interaction physique avec les ordinateurs et le développement d'interface NUI.

Comme son nom l'indique, « SimpleOpenNI »[25] propose une version légère du frameWork « OpenNi » et du processeur Nite, de nouveau par le biais d'une classe java wrapper. Le site propose, outre le code source, une documentation et wiki remplies d'informations utiles à son fonctionnement.

2.4.3 LeapMotion

LeapMotion[26] se présente sous la forme d'un petit boîtier à placer près de votre écran. Selon le site de la compagnie, il permet d'interagir avec son ordinateur de manière plus efficace qu'une souris et serait plus sensible qu'un écran tactile. Il permet ce contrôle dans un environnement à trois dimensions à l'aide de la détection du mouvement de nos mains et de nos doigts. Il est aussi capable de détecter des éléments comme un stylo.

[23] http://www.openframeworks.cc
[24] http://iad.zhdk.ch/en/people/max-rheiner
[25] http://code.google.com/p/simple-openni
[26] http://leapmotion.com/

Le périphérique se branche via USB. Après l'installation de leur logiciel et un calibrage, il sera prêt à l'utilisation. Il crée un espace de $8m^3$ par appareil et peut être mis en réseau avec d'autres appareils pour augmenter la zone. Il permet aussi de programmer des séquences de mouvements réutilisables et sa précision permettrait, par exemple, de créer une signature digitale à l'aide de son stylo.

Le leapMotion devrait être un sérieux concurrent de la Kinect sur le marché des interfaces NUI. Sa soi-disant précision permettant de détecter avec efficacité nos mains met un sérieux coup au périphérique de Microsoft qui peine actuellement sur ce point-là [21].

3. Use case

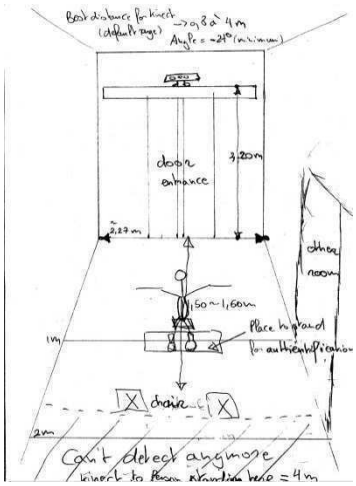
Le cadre du travail est, dans les limites de la Kinect, de pouvoir traiter les informations qu'elle fournit afin de communiquer à d'autres composants des signaux qui permettront de prendre des décisions. Par exemple, le chauffage d'une pièce serait adapté au nombre de personnes présentes dans une pièce.

Après une première phase d'analyse, agrémenté de test sur les démos fournit ou encore de développement de petits utilitaires, les possibilités s'offrant à nous commencent à se dessiner. Le use case qui suit a été pensé lors d'une séance à l'HES-SO de Sierre, le 25.06.2012, en présence de MM. Bocchi et Genoud, expliqué au point 4.5.

3.1. But

L'idée est de placer une Kinect au-dessus d'une porte et de filtrer le trafic. Les informations reçues seront traitées afin de savoir le nombre de personnes présentes dans la pièce, si elles sont en mouvement, ou encore, si elles sont assises. Visible sur l'image 17 (en annexe), un croquis de la scène, où l'ensemble des tests seront réalisés, avec les mesures réelles indiquées.

Image 17 : croquis réalisé pour les tests



L'application traitera deux des types de flux fournis par la Kinect. Premièrement, celui de la caméra qui donnera une information visuelle de la pièce.

Deuxièmement, celui de la caméra de profondeur qui permettra de détecter, avec l'API fourni, si un squelette (donc un humain) est détecté. Pour déterminer si la personne se déplace, la position donnée par l'API de la Kinect sera prise entre deux intervalles. Ensuite, si une marge infime existe entre les deux positions, on estimera que la personne est immobile.

Le squelette fournit aussi les positions de différents points du corps. Avec, nous pouvons déterminer la position des membres du corps d'une personne ou encore sa taille. L'idée sera, à l'aide de ces données, d'estimer si une personne est assise.

Nous aimerions aussi pouvoir, à l'aide du squelette d'une personne, lui associer une identité préalablement enregistrée. Comme on peut le voir sur le croquis ci-dessus, la personne devrait être scannée à une certaine distance avec une certaine position.

3.2. Acteurs

3.2.1 Kinect

Cet acteur est responsable de la surveillance d'une pièce. Il doit émettre les informations à l'application de façon ininterrompue. Sa position et son angle de vue garantissent une vue idéale et restent fixes.

3.2.2 Personne tracée

Cet acteur correspond à un squelette tracé dont nous gérons les informations. Nous connaitrons sa position, la forme de son squelette ou encore si il se déplace.

3.3. Stories

Plusieurs scènes et scénarios ont été enregistrés à l'aide du Kinect Studio. Ainsi, une base de séquences, ayant en commun les mêmes conditions d'enregistrement, seront disponibles lors du développement de l'application. Voici les différentes stories imaginées lors de ce tournage.

3.3.1 Gestion d'une personne

Une personne entre dans la pièce et va s'asseoir sur une chaise. Ensuite, elle se relève et repart vers la sortie. La même story est filmée avec un arrêt de la personne lors de son entrée, afin de la scanner de dos pour l'identifier.

3.3.2 Gestion de deux personnes

Comme précédemment, une première personne rentre et va s'asseoir sur une chaise. Elle est ensuite rejointe par une seconde personne qui fait de même. Après un court laps de temps, elles se lèvent, interagissent entre elles et rejoignent la sortie l'une après l'autre.

Une version avec un scan des personnes a été réalisée ainsi qu'avec plusieurs croisements entre elles afin d'étudier la perte du traçage momentané du squelette.

3.4. Scènes de test

Comme indiqué au point 2.3.7, le « Kinect studio » va nous être utile pour enregistrer un maximum de données afin de les utiliser lors du développement de notre application. Pour cela, la démo « Kinect Explorer » a été utilisée et la scène ci-dessous a été mise en place.

Image 18 : la scène de test

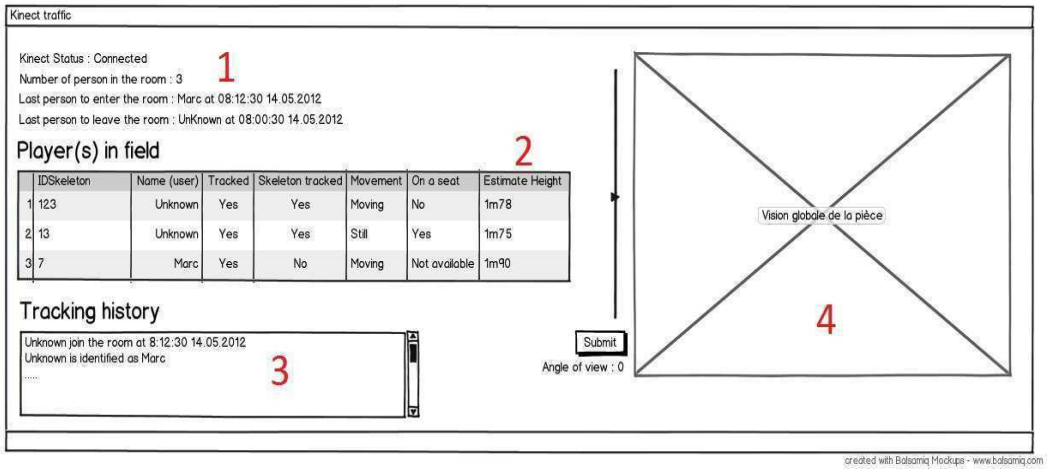
La première image nous montre la pièce filmée. Elle fait 2,27 mètre de large, propose une entrée principale et une porte sur le côté donnant accès à une autre pièce. La deuxième image donne une idée de ce que verra la Kinect. Celle-ci, comme on peut le voir sur la troisième image, se situe au-dessus de la porte d'entrée à 2,20 mètre de hauteur pour un angle de -27 degrés (le minimum possible).

Ce corridor a été aménagé pour les tests, comme on peut le voir sur les images 3 et 4, en disposant des chaises nécessaires aux stories décrites au point 3.3. Les lignes au sol représentent les mesures, prises depuis le bas de la porte, pour une distance de 1 mètre puis 2 mètre. Bien que la Kinect propose une distance maximum de 4 mètre, le fait d'avoir baisser son angle fait perdre le traçage d'un squelette se trouvant au-delà des 2 mètre, ici, derrière les chaises.

3.5. ProtoEcran

Voici, visible sur l'image 19, le proto-écran de l'interface créée suite aux discussions avec les clients. Les Informations disponibles seront affichées en quatre parties distinctes.

Image 19 : interface de la future application



La partie 1 concerne les informations générales qui indique si la Kinect est connectée, le nombre de personnes présentes dans la pièce ou encore, les dernières personnes qui sont entré ou sorti.

La partie 2 affiche les informations concernant les personnes tracées par l'application. Elle permet de connaitre leur ID fournit par l'API, leur identifiant qui peut être inconnu si la personne n'est pas reconnue, de savoir si la personne est bien traquée, si elle est traquée totalement au niveau du squelette, si elle se déplace, si elle est assise et pour finir, une estimation de sa taille.

La partie 3 offre un historique du trafic dans la pièce. Il permet de connaître les gens qui sont venus et repartis ou encore les identifications qui se sont déroulées.

La partie 4 est l'affichage de l'image montrant la pièce ainsi que les squelettes tracés dessinés par-dessus. La barre verticale permet de soumettre, à l'aide du bouton « Submit », l'angle de la Kinect désiré.

4. Réalisation

L'idée de ce chapitre est de présenter le déroulement du projet et de mieux comprendre les décisions prises. La particularité du sujet abordé est, au fur et à mesure de la prise en main et des mises à jour de ce produit d'innovation, le cahier des charges évoluait constamment.

Des rencontres avec les clients, MM. Bocchi et Genoud, ont été mises en place. Elles se déroulaient de la manière suivante : à l'aide d'un support de présentation, je discutais des points abordés, des difficultés rencontrées et présentait à la fin une démo. A la suite de quoi une discussion prenait place avec les clients, afin de décider de la direction à prendre pour leur application.

Cette façon de travailler ressemble beaucoup aux méthodes agiles et plusieurs sprints se sont déroulés lors de ce travail. Un chapitre pour chacun des sprints explique le travail effectué durant les quelques semaines, les résultats découverts, les prises de positions et les éléments importants discutés lors des rencontres avec les clients.

4.1. 1er sprint : 12 mars au 30 avril 2012

Ce premier sprint est l'un des plus longs, car, durant cette période, il a fallu mettre en place le projet, se documenter et attendre l'arrivée d'une Kinect. Les semaines étant encore courte, 14 heures, les rendez-vous sont pour l'instant espacés.

Une première discussion, le 16 mars 2012 avec M. Bocchi, m'a permis d'en savoir plus sur les attentes de ce travail et sur le projet global, qui cherche à tirer profit d'une Kinect. Rapidement, il m'est demandé de fournir un cahier des charges avec les grands lignes directrices ainsi que de trouver un site de vente en ligne pour commander la Kinect. Il me prodigue aussi quelques conseils et directives, ainsi que détails sur les avancées du projet à fournir avant chaque rencontre, un résumé du travail effectué, les problèmes rencontrés et solutions proposées et, si possible, préparer une démonstration.

Les deux premières semaines sont donc consacrées à la rédaction du cahier des charges et à la mise en place d'un planning (tous deux en annexes) pour visualiser les 360 heures consacrées durant les 22 semaines à disposition. Du temps est aussi pris pour la création d'une image virtuelle, le client souhaitant pouvoir avoir un environnement de travail à disposition à la fin de ce travail.

Lors de ma rencontre du 30 mars 2012 avec M. Bocchi, on m'informe des problèmes de la commande de la Kinect qui, occasionne du retard. Durant cette attente, le temps est consacré à la recherche d'informations sur le fonctionnement de la Kinect, l'état de l'art ou encore, de trouver les futurs tutoriaux qui permettront de mieux assimiler la technologie. Pour ma part, j'évoque pour la première fois le problème de la virtualisation. Il semblerait, selon divers forum, que la Kinect ne fonctionnerait pas avec un système virtualisé.

Le 21 avril 2012, je mets la main sur la Kinect version Window PC. Cela fait maintenant un mois qu'elle est officiellement sortie et je fais partie des premiers privilégiés à découvrir ce périphérique. Mes premiers pas sont essentiellement basés sur la découverte des démos fournies par le Kinect SDK ou encore, de trouver un moyen d'afficher une image à l'aide des flux de données « Color Frame » et « Depth Frame ».

Le 30 avril 2012, une rencontre avec MM. Bocchi et Genoud marque la fin de ce premier sprint. Je commence par montrer les différentes démos fournies par Microsoft et nous constatons rapidement certaines failles de la Kinect. Par exemple, notre corps est traqué certes, mais le programme devient vite confus avec notre environnement, gêné par une chaise ou par le fait que nous sommes positionnés trop près de la Kinect.

L'idée proposée, au début de ce projet, de surveiller une pièce est revue à la baisse. Selon la documentation fournie par le SDK de la Kinect, la portée maximum pour qu'elle soit opérationnelle se limite à quatre mètres. Les clients me proposent donc un nouveau scénario : l'idée serait, dans un premier temps, de pouvoir connaître le nombre de personne se trouvant dans une pièce, de savoir si elles se déplacent ou si elles sont assises. Dans un second temps, il faudrait analyser les possibilités d'identifier une personne.

Ils me conseillent aussi, au vu des contraintes de la Kinect, de préparer une scène de test reproductible où aucun objet n'entraverait le traçage des personnes et où les conditions, de l'éclairage notamment, serait idéales.

A la question de savoir si le traitement vocal pourrait servir dans la reconnaissance d'une personne ou pour passer des commandes à l'application, les clients ne désirent pas l'intégrer. D'ailleurs, selon eux, ce domaine est plus complexe qu'il n'en a l'air pour un résultat pas forcement garanti.

Pour finir, deux questions majeures sont soulevés à savoir si la Kinect peut fonctionner sur une image virtuelle ou encore, comment l'API de Microsoft gère le traçage des personnes. Elles seront abordées lors de la prochaine rencontre.

4.2. 2ème sprint : 30 avril au 21 mai 2012

Durant ce deuxième sprint, la recherche d'informations continue. De nombreux forums, vidéos et ateliers pratique sont analysés. Une longue recherche, de projets existants, permettant d'enregistrer des séquences de la Kinect pour les reproduire par la suite, ne sera pas concluante. Il me faut donc encore et toujours me lever, que la Kinect puisse me détecter, à chaque fois que je souhaite tester mon code.

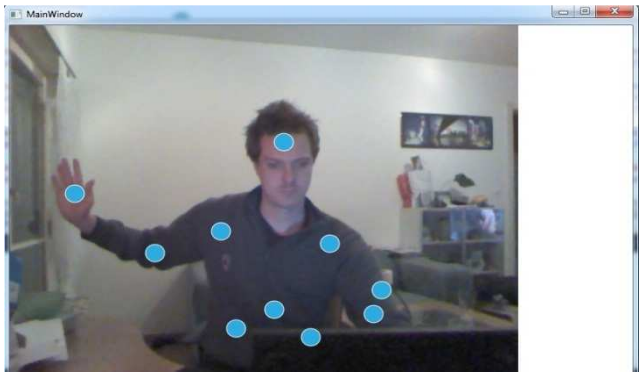
J'analyse les différentes possibilités qui s'offrent à moi pour identifier une personne. La première serait d'analyser l'image à l'aide d'algorithmes avancés pour reconnaître le visage. Cette option trop poussée, hors de notre sujet et n'utilisant que trop peu la Kinect est abandonnée. Tout comme l'est l'idée de reconnaître une personne à l'aide de son contour. En effet, des habits voir une coupe de cheveux différentes rendraient cette tâche impossible.

Finalement l'idée viendra d'un projet découvert durant mes recherches ; L'auteur est Greg Duncan, qui a posté sur le site « Coding4Fun », un site d'informations sur les applications développées pour la Kinect. Il propose, à l'aide des différents points mesurés sur notre corps, d'indiquer notre taille approximative. En effet, après quelques tests, la taille correcte semble se trouver lorsque nous nous trouvons à un mètre de la Kinect [22].

Après analyse de ce code, je constate qu'il s'agit de règles de trigonométrie. En appliquant les formules, je suis maintenant capable d'obtenir les mesures suivantes d'une personne : taille, longueur d'un bras, d'une jambe, du torse et largeur des épaules.

De plus en plus à l'aise avec les composants Kinect Microsoft, je suis maintenant capable d'afficher les images des flux transformé. Désirant voir à l'image les différents points du squelette mesuré par la Kinect, je m'inspire d'une démo fournie avec le SDK. Un canevas, sorte de cadre transparent, est placé par-dessus l'image, ce qui permet de dessiner un squelette en surimpression.

Image 20 : test sur les points du squelette



Le développement de ces deux démos, m'ont permis d'assimiler la classe « Skeleton ». Il est ainsi possible d'obtenir les positions dans l'espace de chacun des points d'un squelette activement tracé.

Le 21 mai 2012 à Sierre, je présente ma revue de sprint à MM. Genoud, Bocchi et Wannier. Je commence par répondre aux questions de la dernière séance. Après des tests effectués, je confirme qu'il n'est malheureusement pas possible de travailler avec une image virtuelle. Microsoft promet d'y remédier dans prochaine mise à jour pas encore annoncée. Du coup, l'application finale sera rendue avec les explications nécessaires à l'installation.

La seconde question concernait le traçage des squelettes. Le nombre vérifié, est bien de deux personnes tracées de manière active, et de quatre personnes de façon passive. Un doute subsiste encore afin de savoir si des ID sont attribués et, si oui, comment sont-ils gérés. Une réponse sera fournie lors de la prochaine séance.

Etant donné la distance de un mètre nécessaire pour mesurer la taille d'une personne correctement, les clients proposent de mettre des contraintes définies lors des tests. On fera en sorte de toujours scanner à la bonne distance. Je propose aussi, qu'un scan soit déclenché lorsque la personne place les deux bras à l'horizontale.

4.3. 3ème sprint : 21 mai au 11 juin 2012

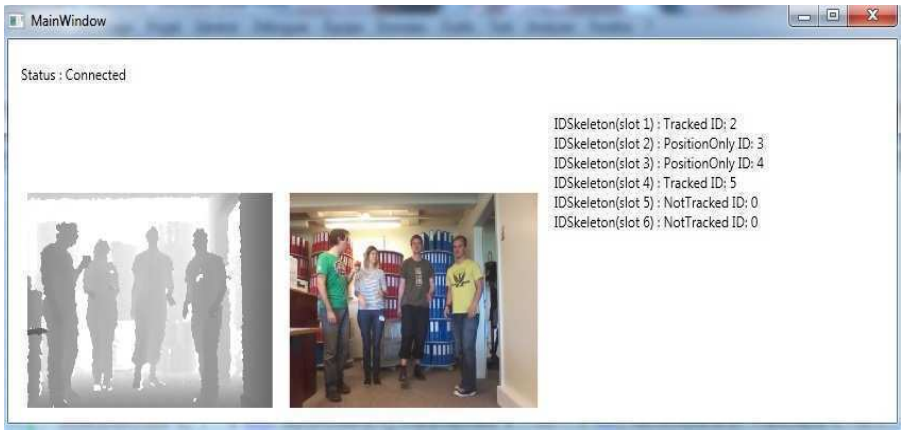
Ce nouveau sprint voit débarquer la sortie de la nouvelle version, 1.5, du SDK de la Kinect. Les composant fournis par Microsoft ne fonctionnent plus de la même manière et demandent de grandes modifications dans le code. Pour éviter ce genre de soucis à l'avenir, je décide de ne développer que mes propres outils afin de ne pas être dépendant de leurs composants.

Les nouveautés majeures disponibles sont la possibilité de tracer des points du visage et un studio pour enregistrer des séquences. Après quelques recherches, j'abandonne l'idée de pouvoir utiliser le scan du visage pour identifier une personne. La précision des données est trop légère et un tel outil serait beaucoup trop complexe à mettre en place.

Le studio qui permet de sauvegarder des séquences tombe à points nommé. J'en profite, à l'aide d'une des démos fournies par le SDK, de récolter un maximum de données. Avec l'aide de plusieurs de mes collaborateurs de travail, j'enregistre leur passage, tour à tour, en leur demandant de faire une rotation sur eux-mêmes à une distance de un mètre de la Kinect. Je profite aussi de faire quelques séquences avec quatre personnes à l'image, qui se croisent et se recroisent, que j'utiliserai comme base de test pour analyser le comportement du traçage.

Le lundi 11 juin 2012, c'est un peu dépité que je rencontre M. Genoud sur Sierre. En effet, je n'ai pas de démo à présenter cette semaine car la nouvelle version rend difficile l'usage des composants que je n'arrive pas à assimiler. On se demande s'il faudrait rester sur la 1.5, question à laquelle on ne peut répondre que par oui, tant le nouveau studio d'enregistrement va nous être utile.

Image 21 : premier enregistrements



A la fin de cette séance, le client me précise un scénario avec comme objectif, les points essentiels suivants à mettre en place d'ici à la fin du projet : déterminer le nombre de personnes dans une pièce, savoir si une personne se déplace, si elle est à l'arrêt ou encore, si elle est assise.

4.4. 4ème sprint : 11 juin au 25 juin 2012

Après une analyse pour déterminer les classes nécessaire ou encore, la création d'un prototype de l'interface de la future application, ce sprint voit le développement des premiers outils nécessaires à l'application finale.

Je commence par parcourir l'aide en ligne du SDK de la Kinect. J'y trouve les informations nécessaires pour créer une classe me permettant d'afficher une image selon le type de flux reçu. Je développe ensuite une classe pour manipuler les informations fournies par le flux du squelette et une autre, pour conserver des informations sur les personnes présentes dans la pièce.

En déterminant le nombre de squelettes tracés, je résous l'un des objectifs de mon scénario, car je sais maintenant combien de personnes sont présentes dans la chambre. Un deuxième est résolu après réflexion. En me basant sur la mesure prise au niveau du torse, disponible autant pour le mode actif que passif d'un squelette tracé, j'enregistre une valeur que je compare, après un certain laps de temps, avec la nouvelle reçue. Si, avec une certaine marge, la différence est faible, alors nous estimons que la personne ne se déplace plus.

Le 25 juin 2012, je me rends à Sierre pour un nouveau rendez-vous avec MM. Bocchi et Genoud. A l'aide de ma démonstration, je réponds à leur question concernant le traçage de la Kinect. On constate rapidement que le moindre croisement entre deux personnes nous fait perdre l'identification, remplacée par un nouveau. Dès lors, devons-nous persister à vouloir identifier une personne sachant que nous pourrions perdre son traçage peu après ?

Nous revoyons le scénario une dernière fois. L'idée va être de placer une Kinect à l'entrée d'une pièce. Elle devra nous dire, à une distance idéale et sans obstacle, le nombre de personnes présentes. Nous imaginons aussi la disposition de la pièce. En face de l'entrée, on trouvera un canapé, voire des chaises, et notre application nous indiquera si une personne est assise.

Une personne devra pouvoir être scannée. L'environnement imaginé est idéal, car une vision frontale pour tracer les points est optimale. Cependant, j'émets un gros doute sur la gestion des croisements. Peut-on imaginer réattribuer un ancien ID à un nouveau tracé ayant presque les mêmes mesures. De même, on pourrait imaginer prendre des échantillons de couleurs sur des parties du corps, pour savoir s'il s'agit à nouveau de la personne au pantalon bleu et à la chemise rouge. Les clients prennent note de mes remarques mais me précisent que pour eux, la gestion des croisements n'est pas prioritaire.

Nous nous quittons ce jour-là avec comme objectif de reproduire le scénario réalisé ensemble et avec le croquis ci-dessous qui a découlé de notre discussion.

Image 22 : croquis réalisé

4.5. 5ème sprint : 25 juin au 17 juillet 2012

Ce sprint commence par une journée de tournage pour enregistrer un maximum de séquences qui serviront de support pour le développement de l'application. La scène est installée et mesurée afin d'avoir des repaires. En plaçant la Kinect en hauteur, je constate que le fait de l'incliner me fait perdre une distance considérable. Du coup, pour être sûr de tracer correctement les gens, les chaises seront placées dans un espace qui n'ira pas plus loin que deux mètres.

Des scénarios sont imaginés. Au fur et à mesure, je note quelle scène avec quelle personne est réalisée pour être sûr de ne manquer d'aucune information utile par la suite. Les séquences sont courtes, 20 secondes environ, et mettent en scène une ou deux personnes. Parfois elles se laissent scanner et généralement, elles utilisent une des chaises pour s'asseoir. Des gens se croisant ont aussi été filmés au cas où il resterait du temps pour trouver une solution à ce problème.

La prochaine étape est la volonté de trouver un moyen pour dessiner, par-dessus l'image, le squelette des personnes tracées. L'idée est de déterminer visuellement, si les points mesurés sont fiables, et donc correctement alignés sur la personne que l'on voit. Et comme je n'arrive plus à utiliser les composants fournis par Microsoft, il me faut ma propre classe. La solution viendra du web où Je trouve un article d'un développeur qui s'intéresse lui aussi à la Kinect : Renaud Dumont. Sa classe, une fois adaptée, me permet de dessiner sans problème les squelettes [23].

Une réflexion m'attend ensuite concernant la meilleure façon de détecter si une personne est assise. Après une recherche sur le web de projets existants ayant la même problématique, je constate qu'on utilise souvent la trigonométrie. L'idée est simple : si vous constatez que l'angle à la hauteur de votre genou est de 90 degrés, il y a de fortes chances pour que la personne soit assise.

Bien sûr, dans notre cas, nous sommes de face. Je développe tout de même une classe regroupant des outils de mesure. Je suis maintenant capable de calculer, en plus de la longueur entre plusieurs points, un angle à l'aide de trois points. Malheureusement, les tests ne sont pas concluants. Premièrement, parce que la Kinect s'emballe lorsque une personne est assise. Elle n'est plus capable de fournir correctement les points du squelette, rendant les mesures peu fiables. Deuxièmement, parmi les séquences tournées, l'une d'elle montre une personne qui, une fois assise, tend sa jambe. Ce cas montre que l'angle au niveau du genou ne permet pas à chaque fois de différencier le fait d'être assis ou d'être debout.

Au vu des tests, il faut trouver une autre solution. Une seconde idée est essayée: on va se servir de la taille. Pour l'avoir, vu que la personne ne sera pas forcement toujours à la bonne place, la grandeur est constamment mesurée et n'est retenue que si elle correspond à la valeur maximale

calculée jusqu'à présent. Pour estimer, ensuite, si une personne est assise, il suffit de surveiller la valeur que prend la taille. Si on se retrouve avec une valeur plus basse de 20 centimètre par rapport à sa valeur maximale, alors nous sommes bien en présence de quelqu'un posé sur une chaise.

Le 17 juillet 2012, je rencontre pour la dernière fois MM. Genoud et Wannier à Sierre. Je présente le travail cité ci-dessus et propose une nouvelle idée : créer un historique du trafic de la pièce. Le but serait de savoir à quelle heure il y a du monde présent. Les clients approuvent l'avancée du projet ainsi que le prototype final proposé. Je fais part de mes inquiétudes au sujet de l'identification d'une personne. Selon M. Genoud, ce sujet pourrait, à lui seul, être le sujet d'un second travail. Pour leur projet, il se contenterait d'avoir déjà la possibilité d'enregistrer des mesures.

Durant la soirée, je reçois l'avis de M. Genoud concernant le code source que j'ai développé. Il en ressort que je dois mieux le documenter, en expliquer la structure dans le travail de bachelor et que finalement, il aimerait que quelques données soient sauvegardées dans une base de données. J'en prends bonne note et je les mettrai en place lors du 6ème et dernier sprint.

4.6. 6ème sprint : 17 juillet au 13 aout 2012

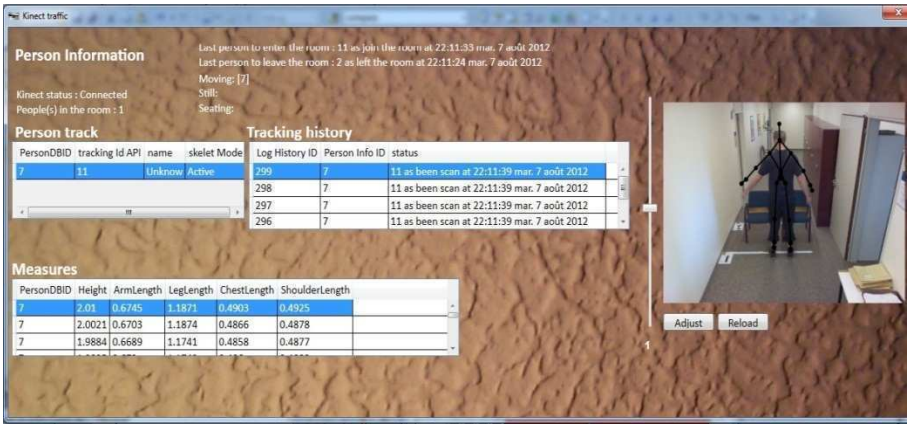
Des mesures ou encore un dessin d'un squelette mal exécuté se produisaient lorsqu'une personne se trouvait trop près des bords du champ de vision de la Kinect. Une méthode nommée « ClippedEdges() », fournie par Microsoft, permet de savoir si quelqu'un en est proche. Elle est implémentée. Cependant, ça ne sera pas suffisant. Malheureusement, la méthode, parfois, semble croire un instant que le squelette peut être tracé. Il en résulte une double apparition de la personne. La solution est de mettre un détecteur au niveau de la position du torse. Ainsi, si la position ne se trouve pas dans une certaine zone sûre, la personne n'est pas tracée.

La gestion des croisements de personnes est revue une dernière fois. J'essaie de garder plus longtemps les ID qui ne sont plus tracés, afin de les comparer ensuite au nouveau. Si leurs tailles sont plus ou moins les mêmes, elles devraient concerner la même personne. Le test n'est pas concluant et la gestion de l'historique du trafic devient du coup un vrai casse-tête. Une personne ajoutée est-elle forcément une personne rentrante ou alors, une même personne déjà présente ? Cette faille ne sera pas résolue.

Un sous-projet est ensuite créé afin de mettre en place la génération d'une base de données. Un souci inattendu survient lors de la sauvegarde des premiers éléments et le rafraîchissement des dataGrid utilisées. Un détail oublié, ma classe se voit updater 30 fois par seconde, ce qui du coup provoque de très gros ralentissements au niveau de l'application. La solution va être d'utiliser un compteur et de ne pas exécuter ce genre d'opération à tout bout de champ. Ma table permet également d'enregistrer l'ensemble des points mesurés. Une sauvegarde concluante qui n'a pas été retenue, car elle consommait beaucoup trop de ressources. Mais cela reste une possibilité.

Afin de manipuler et sauvegarder mes données en base de données, un service, utilisant des « Entities », a été mise en place. L'interface est revue, certain champ semble ne plus être nécessaire à la sauvegarde.

Image 23 : application finale



Les dernières tâches seront le nettoyage du code, la génération d'une doc, la rédaction des derniers points du travail ainsi qu'une relecture.

5. Application KinectTraffic

5.1. Structure du projet

5.1.1 Programmes utilisés

Le programme a été développé à l'aide du logiciel « Microsoft Visual studio 2010 Premium ». La version « Entity Framework » utilisée est la 4.0. La version de « Microsoft SQL Server 2008 R2 management studio» a été combinée avec « Entity Framework 4.1 » pour la gestion d'une base de données. L'application tourne avec la version 1.5 du SDK et la 1.5.1 du kit d'outils de la Kinect.

5.1.2 Installation et lancement de l'application

Afin de lancer l'application, il vous faut préalablement avoir installé les programmes cités ci-dessus. Les drivers de la Kinect ou encore, du « Entity Framework 4.1 », sont disponibles dans le dossier des « Drivers » sur le cd. Attention, la Kinect ne tourne que sur un système Windows Seven. Si vous avez installé une autre version du SDK, il est préférable de la désinstaller avant.

Une fois votre environnement de travail prêt, aller dans le dossier « Programme » pour exécuter le lien sln, « WpfKinectTraffic ». L'application se charge dans votre « Visual Studio » et trois projets différents sont disponibles. Une fois le programme démarré, des séquences sont disponible dans le dossier « Videos, sur le cd. La section 2.3.7 vous explique comment utiliser le Kinect studio afin de tester l'application.

5.1.3 Projets

Le programme est composé de trois projets. Le premier, est « WpfKinectTraffic ». Ceci est le cœur du programme. Il regroupe toutes les classes nécessaires pour afficher et transformer les données reçues de la Kinect. Voici ci-dessous une description sommaire de l'ossature.

MainWindow

Cette classe est le point de départ. C'est ici que sont mis en place les différents labels, images et datagrids pour afficher les informations. On y trouve aussi l'instanciation des différents contrôleurs qui se chargent de traduire les infos reçues de la Kinect. La gestion des events, principalement celle concernant les flux reçus de la Kinect, y sont également gérés.

PersonDataController

Les informations fournies par la Kinect, concernant le squelette, sont transmises à ce contrôleur. Il regroupe les méthodes qui permettent de déterminer les mouvements et positions de la personne tracée. Il gère également les données qui seront sauvegardées à l'aide du contrôleur « SkeletonController » dans la base de données.

SkeletonController

Ce contrôleur a pour rôle de traiter un ensemble de squelette tracé par l'API et de pouvoir gérer sa propre collection du type de la classe « PersonDataController ». Il permet de gérer la prise de données, d'updater ou encore sauvegarder en base de données à différents intervalles de temps.

FrameViewersController

De façon à pouvoir afficher une image résultant du flux « color » ou « depth », il nous fallait un contrôleur. Son rôle est de gérer une collection qui possède le nom du composant image utilisé dans le « MainWindow » ainsi que la source des données fournies par la Kinect sensor. A chaque itération, il se charge de transformer le flux et de rafraîchir l'image.

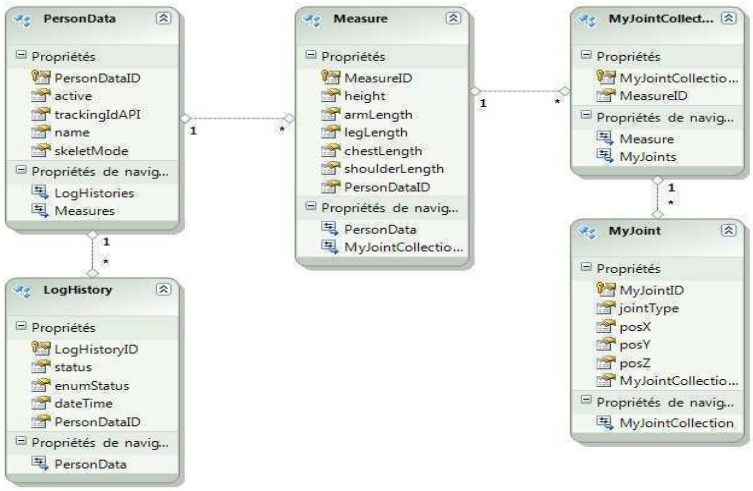
Le second projet est le « KinectTrafficModel ». Il contient les classes, permettant de générer une base de données. Un « drop table » n'a pas été implémentée donc, en cas de modification de certaines propriétés, il vous faudra effacer la base de données avant de relancer l'application.

Le dernier projet est le « TestProjetKinectTraffic ». Afin de s'assurer de la véracité des résultats de trigonométrie obtenue par la classe « BodyMeasurementTools », une série de tests y sont effectués.

5.2. Base de données

Sur l'image 24, vous trouverez le schéma de la base de données suivi d'une explication sur la raison de la présence de chaque table.

Image 24 : aperçu de la base de données



PersonData

Cette table enregistre les informations d'une personne tracée par l'application. Le champ « active » permet de savoir si elle l'est actuellement et, dans ce cas, le programme affiche ses données. Le champ « name », comme son nom l'indique, représente le nom de la personne qui, par défaut, est fixé à « Unknown » (inconnu). Il permet, si à l'avenir une reconnaissance de personne est possible ou si on souhaite attribuer manuellement des données à une personne, de préciser le nom.

Le champ « trackingIdAPI » est l'ID fourni par la Kinect au squelette tracé. Le champ « skeletMode » nous indique si le mode de traçage est passif ou actif. « PersonData » est composée aussi de deux tables décrites ci-dessous.

Measure

Chaque fois qu'une personne est scannée les mesures suivantes sont sauvegardées : taille, longueur de bras, de jambe, du torse ainsi que largeur des épaules. Si on le souhaite, il est possible, à l'aide des tables « MyJointCollection » et « MyJoint », de sauvegarder chaque position du squelette tracé fourni par l'API de la Kinect.

LogHistory

Cette table enregistre les évènements datés de certains cas énumérés comme l'entrée ou la sortie d'une personne dans une pièce, ou encore lorsque elle est scannée.

5.3. Méthodes développées

Parlons maintenant plus en détail des méthodes développées pour obtenir les différentes informations affichées par l'application. Elles seront traitées par thème avec les références des classes concernées.

5.3.1 Stocker les informations d'une personne

La class « MainWindow » recevra 30 fois par secondes de nouvelles informations des différents flux provenant de la Kinect. Il faut donc trouver un moyen de traiter ces informations sans pour autant alourdir la vitesse d'exécution de l'application.

Les informations nécessaires concernant une personne ne se trouvent que dans un seul flux : « SkeletonFrame ». Elles sont utilisées soit pour la mesure des déplacements, soit pour enregistrer des données dans la base. La classe « SkeletonController » s'en occupe.

A chaque image reçu, la méthode « SkeletUserUpdate() » qui se trouve dans la classe « SkeletonController » permet de mettre à jour notre propre jeu de données concernant les squelettes tracés. Comme expliqué au point 2.3.5, la méthode chargée de surveiller l'arrivée de nouvelle données se prénomme « Sensor_SkeletonFrameReady() » et se trouve dans la classe « MainWindow ».

Que fait exactement la méthode « SkeletUserUpdate() ». Elle commence par faire une copie des derniers squelettes tracés par l'API à l'aide de la méthode « UpdateFrame() ». Ne seront gardés que les personnes tracées et celles ne se situant pas trop près des bords, déterminer par la méthode « IsInKinectCameraBorder() ».Ensuite, la méthode « AddPersonToDictionary() » compare chaque personne tracée de cette copie à notre propre tableau de type « PersonDataController ». Si la valeur est nouvelle, elle est ajoutée, sinon, elle est mise à jour avec les nouvelles valeurs.

Ensuite la méthode « RemovePerson() » se charge d'effacer les personnes qui ne sont plus tracées. Pour cela, on utilise la propriété « timeStamp » fournit par l'API. Ce nombre, représente en seconde le temps écoulé et sa valeur est fixée par l'API. Si une valeur de notre tableau n'a pas vu cette valeur être rafraîchie, alors elle n'est plus tracée. Elle est supprimée de la collection et désactivée de la base de données.

Si l'application devait s'interrompre suite à une fermeture de la fenêtre, la méthode « Window_Closing() » dans la classe « MainWIndow » s'en charge en appelant la méthode « EndProgramRemovePerson() » se trouvant dans la classe « SkeletonController ». Cette méthode parcourt les squelettes encore tracé pour les désactiver et mettre à jour la base de données. A noter qu'elle enregistre un historique pour chaque personne, comme expliquée au point 5.3.8, pour enregistrer le fait que ces personnes ne sont plus tracées suite à l'arrêt brutal du programme.

5.3.2 Afficher l'image et dessiner un squelette par-dessus

Pour afficher l'image sans passer par les composants fournis dans les outils du SDK, le code fourni par Microsoft sur son site « MSDN » fera très bien l'affaire. Les lignes concernées sont présentes dans les méthodes « ColorImageFrame() » et « DepthImageFrame() » dans la classe « frameViewerController ». Elles permettent de transformer le flux reçu en un bitmap, ce qui représente un tableau de bits.

Ce contrôleur contient deux dictionnaires, un pour chaque type de frame. Si on souhaite afficher un des flux, on commence par choisir la bonne classe pour stocker les données, « ColorData » ou « DepthData », en indiquant la Kinect sensor à utiliser. Ensuite, ces données sont associées à une clé de type String, l'idéal étant de mettre le nom de l'image source qui s'affichera. Finalement, à chaque arrivée de nouveaux flux, la méthode écrite par Microsoft nous retournera le bitmap que nous pourrons appliquer à la source de notre image qui se chargera de l'interpréter et de l'afficher.

Le fait d'avoir créé un dictionnaire à qui l'on passe un type de données pour une certaine Kinect sensor est uniquement dans l'optique du jour où l'option de plusieurs Kinect connectées en même temps sera disponible. On pourra ainsi demander, pour cette source précise, que telle ou telle image soit affichée.

Sans vouloir encore une fois passer par les composants fournis par Microsoft, on souhaite pouvoir dessiner un squelette par-dessus l'image. Après quelques recherches, la solution de placer un canevas superposé à l'image est préférée. La position des points et les traits entre eux sont donc dessinés sur le canevas à l'aide d'une ellipse, composant de dessin dans Visual studio.

Pour faire cela, la classe « SkeletonDrawing », développée entièrement par Renaud Dumont, est utilisée. Après quelques modifications, cet objet devient une propriété de la classe « PersonDataController » et la référence du canevas utilisé, lui est passé en paramètre. A chaque image reçue, on rafraichit le dessin pour chaque membre du corps tracé. Si la personne n'est plus présente, la méthode « DeletePersonData() » dans la classe « PersonDataController » appelle la méthode Erase() du composant de type « SkeletonDrawing » [23].

5.3.3 Connaître le nombre de personnes présentes

Dès l'instant où une personne se déplace dans la pièce, la Kinect la détecte, soit complétement (informations complètes du squelette), soit uniquement par la position d'un point au niveau de son torse. Bien sûr tout ça dans les limites, cité dans le chapitre 2.3.5 qui décrit l'API, de six personnes qui peuvent être tracées à la fois.

L'idée ici est donc, à chaque image que nous recevons concernant le squelette, de mettre à jour notre tableau contenant les personnes tracées comme expliqué au point 5.3.1. La longueur de ce tableau nous indique au final le nombre de personnes présentes dans la pièce.

5.3.4 Tracer le mouvement d'une personne

Pour la détection du mouvement, nous allons nous servir d'une valeur que toute personne tracée fournit : la position de base au niveau du torse. Ainsi, nous n'avons pas à nous soucier du mode de traçage actif ou passif.

Dans un premier temps, nous devons sauvegarder une ancienne valeur pour la comparer ensuite, après un laps de temps, avec de nouvelles données. Lors de la mise à jour des squelettes tracés, comme expliqué au point 5.3.1, la méthode « StorePersonsPosition ()» dans la classe « SkeletonController » se charge de parcourir chaque données de notre dictionnaire des personnes tracées, de type « PersonDataController », et, d'appeler leurs méthode « StoreOldPosition() ».

Maintenant, lorsque nous mettons à jour les informations concernant une personne se trouvant dans notre dictionnaire, comme expliqué au point 5.3.1, nous mettons à jour la propriété « still » de la classe « PersonDataController ». Cette propriété indique, à l'aide de la méthode « IsStill() », si une personne est figée à l'aide des calculs suivant : les positions x, y et z, actuelles et anciennes, sont comparés entre elles. Si la différence de l'une de ces valeurs dépasse le seuil de 0.04 centimètres, alors nous pouvons estimer qu'il y a eu un déplacement.

Stocker d'anciennes valeurs et tester si une personne est à l'arrêt, ne devraient jamais se faire en même temps, car les données seraient exactement les mêmes et fausserait donc les résultats. Pour cela, un compteur, qui s'incrémente à chaque nouveau jeu de données, permet de filtrer, à l'aide d'un modulo, l'appel ou non de l'une des méthodes.

5.3.5 Connaître la taille d'une personne

La taille devrait plus ou moins être connue sans qu'on ait besoin que la personne soit en position de scan pour l'obtenir. Pour cela, nous allons à chaque update des valeurs concernant une personne, ne retenir que la valeur maximum entre la nouvelle et l'ancienne, car malheureusement, une mesure trop proche de la Kinect nous indique une valeur trop base.

La classe « PersonDataController » la conserve dans la propriété evaluateHeight. Pour la mesurer, nous faisons appel à la méthode « Height() » qui se trouve dans la classe « BodyMeasurementTools » en passant en paramètre la classe « Skeleton », que chaque personne possède comme propriété. Cette classe fournit tous les points mesurés du squelette.

5.3.6 Savoir si une personne est assise

Pour estimer si une personne est assise ou non, il faut vérifier que la personne ne bouge plus et qu'elle perd soudainement au minimum 20 centimètres. Cette valeur est calculée au même endroit et instant qu'expliqué dans le chapitre 5.3.4 sur le calcul du déplacement d'une personne.

La propriété « OnSeat » de la personne en cours est mise à jour avec la méthode « IsOnSeat() » de la classe « PersonDataController ». Elle compare la taille actuelle avec la propriété « evaluateHeight » afin de savoir si la différence est plus grande que la marge paramétrée. Si c'est le cas, et que la personne ne bouge pas, alors on estime qu'elle est assise.

5.3.7 Scanner une personne

Pour le scan d'une personne, trois choses sont sûre et établies dans notre scénario : on la verra de dos, elle devra avoir les deux bras à l'horizontale et sera trouvera à un mètre de la Kinect.

Pour cela, au moment de la mise à jour d'une personne, si la personne peut être scannée, on enregistre une série de mesures calculées à l'aide de la classe « BodyMeasurementTools » avec les méthodes suivantes : « Height() », « ShoulderLength() », « ChestLength() », « ArmLength() » et « LegLength ». Elles sont appelées à l'aide de la méthode « AddMeasure() » se trouvant dans la classe « PersonDataController » et stockées en même temps en vue de l'update de la personne en base de données.

Afin de savoir si le scan peut avoir lieu, on appelle la méthode « CanBeScan() » dans la classe « PersonDataController ». Elle compare, sur l'ensemble des deux bras, la position Y au niveau de l'épaule, coude, poignet et main. S'ils se trouvent au même niveau, nous sommes bel et bien en présence de la position désirée.

Image 25 : aperçu d'un scan

5.3.8 Obtenir un historique du trafic

Pour connaître les heures de passages des personnes, une classe « Log » nous fournit des informations comme l'heure, la date ou encore la raison, qui sont énumérées dans la classe EnumLogStatus, comme « Join », « Leave » ou encore « Scan ».

Quand une personne est ajoutée ou supprimée, comme expliqué dans le chapitre 5.3.1, nous utilisons la raison « Join » ou « Leave ». En même temps, les propriétés « lastLogToEnter » et « lastLogToLeave », de la classe « SkeletonController » sont mises à jour. Elles sont utilisées par les méthodes « DisplayLastIn() » et « DisplayLastOut() », utilisées pour afficher les informations dans la classe « MainWindow ». Si elle concerne un scan, comme expliqué dans le chapitre 5.3.7, alors la raison sera « Scan ».

La méthode « LogToLogHistory() » dans la classe « Log », permet de le stocker dans la classe « PersonDataController » en vue de la mise à jour de la personne en base de données.

5.3.9 Améliorations possibles

Nous sommes capables de tracer des personnes et d'en tirer des informations. Cependant, quelques points peuvent encore être améliorés. La contrainte discutée à la section 2.3.6, au sujet de la perte de l'ID de personnes qui se croisent en est une. Les pistes données dans la section 4.6 sont là pour être explorées afin de régler un problème pas si évident.

Un autre point qui n'a pu être qu'approché, est la reconnaissance d'une personne d'après les mesures récoltées. L'application offre les outils pour obtenir ces informations. Il faut dès lors réfléchir sur la meilleure façon de les traiter pour arriver à ce but. Au départ, l'idée était de prendre plusieurs mesures distinctes, comme discuté à la section 5.3.7. Mais qui sais, avec la prise d'information depuis plusieurs angles, pourquoi ne pas carrément modéliser les personnes en 3D, comme le font déjà d'autres projets, et les utiliser comme référence pour reconnaitre la personne.

6. Conclusion

Dans la première partie de ce rapport, le but a été d'analyser la Kinect afin de connaître ses réelles capacités. Cette phase a permis de mieux appréhender le développement avec ce composant. Au fur et à mesure de mon analyse, les classes et méthodes dont j'allais avoir besoin ont commencé à apparaître dans la masse d'informations à traiter.

L'usage des micros pour donner des ordres à une application a été écarté de suite. Pas encore au point, certaines démos restaient tout simplement insensible à mon accent. Le flux de profondeur, bien qu'employée par la classe « Skeleton », n'a pas été exploré. L'API fournissait suffisamment d'outils pour gérer le traçage de personnes contrairement à d'autre Framework, abordé au point 2.4, qui eux, nécessite de traiter nous-mêmes les données pour obtenir la position du corps.

Cependant, les informations que fournit ce flux, ne doivent pas être négligées. Elles s'avèrent très efficaces lorsque l'on cherche à modéliser des choses en 3D, offrant des informations précises de chaque distance se trouvant entre la Kinect et un obstacle. La Kinect qui pourrait, ce qui reste encore à être déterminé, à l'avenir remplacer des scanneurs coûteux, comme discuté au point 2.2, ou les robots commencent déjà eux, à en être équipés. Divers projets sur le web ont déjà montré la possibilité d'apprendre à la Kinect à reconnaitre certain type d'objet ou encore, des symboles dessinés sur une feuille. Ce qui montre bien le réel potentiel de la Kinect sur ce point.

Durant ce travail, je me suis efforcé de rédigé une documentation permettant de s'imprégner de l'environnement de la Kinect avec les éléments essentiel pour comprendre son fonctionnement. La difficulté étant de bien cibler, sachant que son SDK va sûrement encore beaucoup évoluer. Les pistes fournies, offre selon moi les sources nécessaires pour commencer à travailler avec une Kinect.

Les rencontres avec les clients auront permis de prendre les directives afin de fournir l'application qu'ils souhaitaient. Les décisions, prises en commun, se basaient généralement sur l'expérience que j'avais acquise jusque-là, obtenus notamment suite au développement de petits programmes, que les résultats soient positifs ou négatifs. Noter que la documentation de Microsoft disponible en ligne est idéale pour commencer à développer, offrant les bases nécessaires pour obtenir les premières données. Contrairement à leurs démos, dont les codes sources sont difficiles à comprendre. Leurs utilités ne sont cependant pas remises en question, tant ils permettent de se faire rapidement une idée sur les futurs développements possible.

Des contraintes ont rapidement été décelées. Elles représentent, à mes yeux le souci majeur de la Kinect. Malgré le fait déjà exceptionnel de pouvoir tracer le squelette d'une personne, la technique n'est encore pas au point. Il y encore trop de confusion lorsque la position de la personne est inadéquate, comme se mettre devant une chaise ou ne pas être idéalement placé devant la Kinect en étant trop prêt ou trop loin. Du coup, nous sommes actuellement obligés de délimiter la surveillance d'une pièce, en espérant que les personnes ne se croissent pas et que leur corps soit toujours visible, ce qui n'est pas idéal.

Maintenant, la Kinect est un produit qui vient de voir le jour. Il est fort probable que dans les prochains mois, certaines fonctionnalités évolueront encore et que d'autres apparaîtront. On peut espérer qu'elle gagnera en précision, rendant ainsi le traçage de personnes plus facile. De plus, on peut s'attendre à ce qu'une connexion multiple de Kinect voie le jour, permettant ainsi de recevoir simultanément des informations de différents angles. Évidemment, les risques sont aussi que le code développé ne soit plus d'actualité et que certaines fonctions jusqu'à présent utilisées, ne soient plus disponibles. On compte sur Microsoft pour fournir un SDK de plus en plus stable, n'obligeant pas à chaque mise à jour de revoir tous son code.

Ce projet offre une première idée sur les capacités de la Kinect. En a vu, qu'il est possible d'obtenir des informations utiles, dans notre cas, étudier les gens présents dans une pièce. Ainsi, nous pourrions imaginer, grâce à l'historique de la présence des gens créé, estimer quand démarrer le chauffage ou l'adapter, sachant que généralement, trois personnes seront présentes dans les locaux à tel heure. Dans l'optique que nous serions capables de scanner et de reconnaitre une personne, on pourrait imaginer le chauffage adapté à son profil.

Malheureusement, pour le moment, il existe encore beaucoup de contraintes et sa précision laisse souvent à désirer. Faut-il pour autant abandonner ? Je ne pense pas, du moins pas pour l'instant. La Kinect en est à ses débuts et devrait se bonifier.

A mon avis, l'utilisation d'une seule Kinect reste idéale lorsque l'on souhaite développer une interface « NUI ». Elle permet sans trop de problème d'analyser les gestes d'une personne de face ou encore, d'observer une zone restreinte. Ainsi, on peut, à l'aide de ses mouvements, manipuler par exemple une présentation PowerPoint, zoomer sur une photo ou sur une carte géographique virtuelle, voir associer des positions du corps à une action concrète. Cependant, certains projets discutés au point 2.2, comme ceux qui scannent le corps, nous montres comme parfois, il est nécessaire d'avoir plusieurs Kinect assemblées offrant les données nécessaires supplémentaires.

Je terminerai donc par supposer que plus nous aurions de Kinect présente dans la pièce, plus nous serions à même de couvrir d'espace. On pourrait ainsi les placées au-dessus d'une porte, d'une fenêtre ou encore au plafond du couloir. Chacune, délimitées à surveiller une zone de confort optimale pour la Kinect et de s'occuper de tâches précises. Le souci majeur étant, si on souhaite tracé une personne, de l'associer à chaque ID créé par nos senseurs.

A l'avenir, en rentrant à la maison, ma Kinect de palier me reconnaissant, indiquera à la lumière de s'allumer à la densité que je préfère. Ensuite, ma Kinect du salon indiquera à la TV de se mettre en route sur ma chaine préférée si je m'assoie en face d'elle et que j'exécute un geste indiquant de la démarrer. Ce dernier scénario de sciences fictions risque de bientôt voir le jour. On le voit déjà au niveau de l'offre des téléviseurs qui proposent de plus en plus une navigation « NUI » des chaines laissant tomber les télécommandes habituelles pour nous permettre de le faire à l'aide de nos mains.

Image 26 : ce genre de scène ne sera bientôt plus de la science-fiction

Source : http://www.wiids.co.uk/wp-content/uploads/2010/03/minority-report-ui1.jpg

La suite de ce projet devrait se concentrer sur d'autre scénario que celui de placer une Kinect au-dessus d'une porte. L'idée au final, serait d'énumérer tous les endroits adéquats pour les placers. Au-dessus d'un lit pour vérifier si quelqu'un dort encore et déclencher si nécessaire le réveil matin, dans le jardin pour surveille dans le noir si une intrusion humaine a lieu et allumer les lumières ou encore, dans une pièce surveillant, idéal pour les personnes âgées, tout ce qui pourrait ressembler à un malaise et prévenir au cas échéant les secours.

Comme on peut le voir, les scénarios ne manquent pas et l'idée de relier au système de notre maison de nombreux senseurs Kinect est tout à fait réalisable au vu de son prix. « IoT » lui, devrait permettre de les faire communiquer de manière efficace avec le reste des composants électroniques déjà présents. La Kinect qui seul semble limité, peut à mon avis devenir un senseur essentiel si il est utilisé dans l'emplacement et les conditions idéales. J'espère que ce document contribuera donc à la suite des recherches sur ce sujet au sein de l'école.

7. Gestion de projet

Dans cette partie, le travail de bachelor est revu dans une optique de gestion de projet.

7.1. Déroulement

Le planning des heures consacrées est particulier concernant les temps partiels. Le travail commence plus tôt, permettant aux élèves de terminer dans les mêmes délais, tout en pouvant assurer la fin du semestre des cours et, dans mon cas, un 80% chez mon employeur. Les dates et les heures à consacrer sont les suivantes :

- Du 12.03.2012 au 02.06.2012, 14 heures par semaines
- Du 04.06.2012 au 11.08.2012, 24 heures par semaines

Ce travail s'est déroulé sur 32 semaines pour un total de 374 heures. Les 12 premières semaines se sont déroulées en parallèle avec les cours, à raison de 14 heures semaines. Deux semaines à 24 heures ont eu lieux suivi de deux semaines consacrées pour la préparation et les examens, mais qui ont finalement été utilisées pour le projet. Enfin, 7 semaines à 24 heures pour arriver à la remise du travail, le 13 août 2012.

Les précisions sur le déroulement de ce travail ont été développées au point 4.

7.2. Planification

La planification initiale, disponible en annexe, ainsi que toutes les images utilisées dans ce chapitre, a été reportée dans le tableau 3 pour être confrontée aux heures réellement effectuées. La contrainte majeure quant à la répartition des heures sur la durée du travail concernait l'inconnue autour d'un sujet comme la Kinect. Les méthodologies agiles ont rapidement pris le dessus sur ce planning, correspondant mieux à ce travail, en permettant une meilleure flexibilité.

Tableau 3 : heures planifiées versus réalisées

	Sprint	1		2		3		4		5		6	
Semaine		11 à 17		18 à 20		21 à 23		24 à 25		26 à 28		29 à 32	
		P	F	P	F	P	F	P	F	P	F	P	F
Analyse/Recherche	H	28	36	9	15.5	3	5.5	0	4.5	0	12.5	0	3
Tutoriel/Installation	H	40	17	24	9	16	7	0	0	0	0	0	0
Conception	H	0	0	0	0	8	0	4	0	8	6	8	5
Développement	H	0	0	0	9	20	11	20	24	38	35.5	20	43
Rapport	H	30	23	9	6.5	5	14.5	0	19.5	2	18	48	49
Totaux	H	98	76	42	40	52	38	24	48	48	72	96	100
		H = Heures		P = Prévu		360		F = Fait		374			

7.3. Suivis par sprint

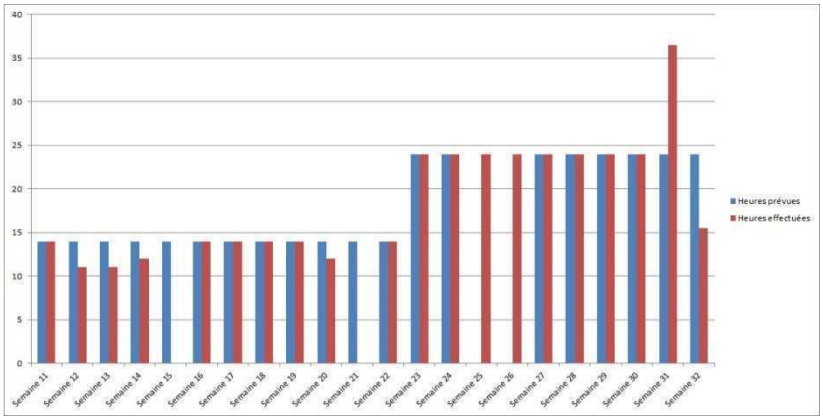
Durant tout le projet, des rendez-vous ont été échelonnés sur la base des agendas respectifs des différents acteurs, ainsi qu'aux nombres d'heures attribués.

Chaque rencontre était ponctuée par une revue de sprint permettant de faire part de l'avancée du projet, comme discuté au point 4. Une discussion fixait, à chaque fois, les objectifs à atteindre pour la prochaine rencontre.

7.4. Bilan des heures effectuées

Dans le tableau 4, un comparatif entre les heures disponibles et celles réalisées, nous montres que le travail a été régulier. Cependant, on peut constater, à deux endroits, de grande différence pour les raisons suivante ;

Tableau 4 : graphique des heures planifiées versus réalisées

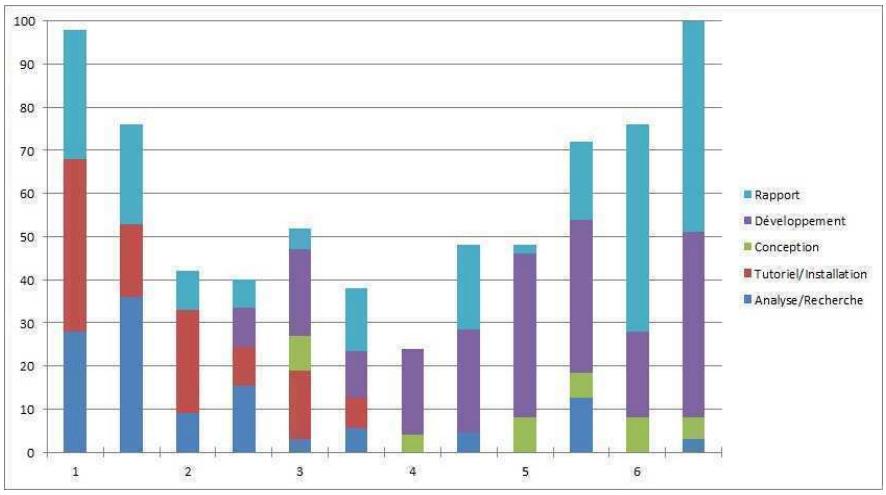


La première se situe de la semaine 12 à la semaine 15. L'arrivée tardive de la Kinect et la fin d'un projet conséquent dans un de mes modules, on eut des répercussions sur le temps attribué au travail de bachelor. Elles ont pu être compensées par les semaines 25 et 26, qui n'étaient, au départ, pas prévu dans le planning.

Le second saut, dans les heures, se situe aux semaines 31 et 32. La raison est qu'habituellement, les heures consacrées se déroulaient le week-end. Désirant imprimer mon dossier le dernier vendredi avant la remise, mon travail a été adapté pour prendre un peu d'avance sur le planning.

Le tableau 5, nous indique, par sprint, comme discuté au point 4, les tâches planifiées (barre au-dessus du numéro du sprint) par rapport à celles réalisée (barre à droit du numéro du sprint).

Tableau 5 : tâches planifiées versus réalisées, groupées par sprint



Ce qui ressort de ce graphique, est la première moitié du projet a été consacré à la recherche d'informations ou encore l'auto-formation sur la Kinect. La seconde moitié, elle, a vu la réalisation de l'application finale. Le développement a été plus conséquent au sprint 6, notamment à cause d'erreurs décelées qui ont nécessité des corrections.

8. Satisfaction personnelle

Le fait que le sujet concernait un produit à la pointe de la technologie et que le travail m'amènerait essentiellement à de la recherche, m'a tout de suite motivé à participer à ce genre de projet.

A l'heure du bilan, deux choses m'auront particulièrement enrichi. La première est la capacité à m'adapter. Le sujet a demandé beaucoup de flexibilité. Il a fallu sans arrêt prendre des décisions et adapter le cahier des charges au vu des résultats des tests effectués. L'imprévu était également une constante à gérer. La seconde, aura été d'avoir une série de rendez-vous avec les clients. Les différentes séances, où mon travail était présenté et débriefé, m'ont permis d'engranger une certaine expérience qui pourrait prochainement me servir dans le monde professionnel.

Au final, ce travail de bachelor aura été plaisant à faire, tant par le sujet traité, la liberté de mouvement accordée ainsi que la bonne entente avec les responsables du projet.

9. Remerciements

Je tiens à remercier les personnes qui ont participé, de près ou de loin, à la réalisation de ce travail.

Je tiens à remercier particulièrement Monsieur Dominique Genoud, Monsieur Yann Bocchi et Monsieur David Wannier pour avoir proposé ce thème et pour m'avoir suivi et encadré durant ce projet.

Merci aux personnes qui ont participé en tant que cobaye à mes séances d'enregistrements pour les tests. Merci pour leurs efforts et leur patience.

Merci aux personnes qui m'ont aidé à relire ce document. Merci pour leurs efforts et leur patience.

10. Déclaration sur l'honneur

Je déclare, par ce document, que j'ai effectué le travail de bachelor ci-annexé seul, sans autre aide que celles dûment signalées dans les références, et que je n'ai utilisé que les sources expressément mentionnées. Je ne donnerai aucune copie de ce rapport à un tiers sans l'autorisation conjointe du RF et du professeur chargé du suivi du travail de bachelor, y compris au partenaire de recherche appliquée avec lequel j'ai collaboré, à l'exception des personnes qui m'ont fourni les principales informations nécessaires à la rédaction de ce travail et que je cite ci-après :

- M. Genoud, professeur à la HES-SO Valais
- M. Bocchi, professeur à la HES-SO Valais

Sierre, le 13 août 2012

Lyness Marc

11. Bibliographie

[1] Leadbetter, R. (2010, Avril 3). *PrimeSense: Beyond Natal*. Consulté le Mars 15, 2012, sur http://www.eurogamer.ne: http://www.eurogamer.net/articles/digitalfoundry-primesense-article?page=2

[2] Terdiman, D. (2009, Juin 1). *Microsoft's Project Natal: What does it mean for game industry?* Consulté le Mars 15, 2012, sur news.cnet.com: http://news.cnet.com/8301-10797_3-10253892-235.html

[3] Gibson, E. (2009, Juin 5). *E3: Post-Natal Discussion*. Consulté le Mars 15, 2012, sur www.eurogamer.net:http://www.eurogamer.net/articles/e3-post-natal-discussion-interview

[4] Bramwell, T. (2010, Juin 14). *MS man: Kinect "perfect name" for Natal*. Consulté le Mars 17, 2012, sur www.eurogamer.net: http://www.eurogamer.net/articles/ms-man-kinect-perfect-name-for-natal

[5] Keighley, G. (2010, Juin 17). *Event, E3 2010: Cirque du Soleil.* Consulté le Mars 17, 2012, sur www.gametrailers.com: http://www.gametrailers.com/video/e3-2010-kinect/700279

[6] Atkinson, C. (2010, Octobre 18). *Microsoft's move : Tech giant spends big to launch Wii rival Kinect*. Consulté le Mars 18, 2012, sur www.nypost.com: http://www.nypost.com/p/news/business/microsoft_move_3gVmAyryJuD6px1dV7LeDP?CMP=OTC-rss&FEEDNAME=

[7] gwr_press. (2011, Janvier 3). *Kinect Confirmed As Fastest-Selling Consumer Electronics Device*. Consulté le Mars 18, 2012, sur community.guinnessworldrecords.com: http://community.guinnessworldrecords.com/_Kinect-Confirmed-As-Fastest-Selling-Consumer-Electronics-Device/blog/3376939/7691.html

[8] Zaffagni, M. (2010, Novembre 15). *La communauté open source s'empare de Kinect, Microsoft laisse faire*. Consulté le Mars 23, 2012, sur www.01net.com: http://www.01net.com/editorial/523549/la-communaute-open-source-s-empare-de-kinect-microsoft-laisse-faire/

[9] Chan, S. P. (2012, Avril 1). *Computing future is name of the game for Microsoft's Craig Mundie*. Consulté le Mars 27, 2012, sur seattletimes.nwsource.com: http://seattletimes.nwsource.com/html/businesstechnology/2017860635_microsoftmundie01.html

[10] Abecassis, Y. (2012, Mars 13). *Kinect pourrait révolutionner le monde professionnel*. Consulté le Mars 24, 2012, sur www.lefigaro.fr: http://www.lefigaro.fr/societes/2012/03/12/20005-20120312ARTFIG00737-kinect-pourrait-revolutionner-le-monde-professionnel.php

[11] Team, K. f. (2012, Mars 15). *Kinect for Windows Solution Leads to the Perfect Fitting Jeans*. Consulté le Mars 30, 2012, sur blogs.msdn.com: http://blogs.msdn.com/b/kinectforwindows/archive/2012/03/15/kinect-for-windows-solution-leads-to-the-perfect-fitting-jeans.aspx

[12] Leclercq, D. (2012, Juin 18). *Xbox 720 : fuite sur les nouveautés !* Consulté le Juin 18, 2012, sur www.pcworld.fr: http://www.pcworld.fr/2012/06/18/jeux-video/xbox-720-fuite-nouveautes/528989

[13] LaMonica, M. (2011, Décembre 16). *Microsoft's Kinect: A robot's low-cost, secret weapon*. Consulté le Juin 1, 2012, sur news.cnet.com: http://news.cnet.com/8301-11386_3-57337485-76/microsofts-kinect-a-robots-low-cost-secret-weapon/

[14] Bonvin, D. (2010, Décembre 21). *Connecting Kinects for Group Surveillance*. Consulté le Avril 14, 2012, sur actu.epfl.ch: http://actu.epfl.ch/news/connecting-kinects-for-group-surveillance

[15] Mesnier, D. (2011, Mars 2). *Le capteur Kinect de Microsoft permet une interaction inédite avec le corps*. Consulté le Avril 28, 2012, sur blog.alpict.com: http://blog.alpict.com/2011/03/02/le-capteur-kinect-de-microsoft-permet-une-interaction-inedite-avec-le-corps

[16] L, J. (2012, Décembre 23). *Kinect : un hack pour manipuler les radiographies médicales*. Consulté le Avril 21, 2012, sur www.numerama.com: http://www.numerama.com/magazine/17670-kinect-un-hack-pour-manipuler-les-radiographies-medicales.html

[17] kun, S. (2010, Octobre 13). *Comment fonctionne la technologie Kinect ?* Consulté le Avril 14, 2012, sur www.gameblog.fr: http://www.gameblog.fr/article-lecteur_612_comment-fonctionne-la-technologie-kinect

[18] lilyszajn. (2012, Février 21). *Microsoft Kinect Specifications & Sensor Breakdown*. Consulté le Avril 30, 2012, sur lilyszajnberg.com/: http://lilyszajnberg.com/blog/?p=688

[19] Hinchman, W. (2011, Juin 20). *windows-kinect-sdk-vs-openni*. Consulté le Juin 14, 2012, sur labs.vectorform.com: http://labs.vectorform.com/2011/06/windows-kinect-sdk-vs-openni-2/

[20] Shiffman, D. (2010, Novembre 14). *Kinect and Processing*. Consulté le Mai 14, 2012, sur www.shiffman.net: http://www.shiffman.net/2010/11/14/kinect-and-processing

[21] Leapmotion. (s.d.). *leapmotion*. Consulté le Mai 26, 2012, sur leapmotion.com: http://leapmotion.com/

[22] Duncan, G. (2012, Mai 16). *Kinect and Processing*. Consulté le Mai 24, 2012, sur channel9.msdn.com: http://channel9.msdn.com/coding4fun/kinect/Put-down-the-measuring-tape-Using-the-Kinect-to-find-your-height

[23] Dumont, R. (2012, Avril 19). *Kinect and Processing*. Consulté le Juin 27, 2012, sur www.renauddumont.be: http://www.renauddumont.be/en/2012/kinect-sdk-1-0-3-tracker-les-mouvements-avec-le-skeletonstream

[24] Tamblyn, T. (2012, Janvier 10). *Microsoft Kinect for Windows launched*. Consulté le Août 19, 2012, sur www.t3.com: http://www.t3.com/news/microsoft-kinect-for-pc-coming-early-2012

12. Abréviations

IPv6	Internet Protocol version 6
API	Interface de programmation
SDK	Software development kit
NUI	Natural User Interface
MIT	Massachussets Institute of Technology
USB	Universal Serial Bus
DMO	DirectX Media Object
msdn	Microsoft Developer Network
IoT6	Internet of Things 6

13. Table des illustrations

13.1. Images

13.2. Tableaux

14. Annexes

14.1. Cahier des charges

Cahier des charges

Projet : Use the Kinect plateform to increase IPv6 devices awarness in a room
Cadre : Hes-So Valais / Travail de Bachelor 2012
Etudiant : Marc Lyness
Responsable : Dominique Genoud

Introduction

Ce travail s'intègre dans le cadre du projet européen IoT6. Ce dernier doit démontrer comment bénéficier du nouveau standard IPv6 afin d'améliorer la communication entre différents appareils. Eclairages, stores ou encore chauffage, sont les nombreux exemples de données à récupérer pour offrir des services utiles au client final.

L'attention se portera sur la kinect de Microsoft. Cet appareil a jusqu'à maintenant servi de détecteur de mouvement pour une console de jeux vidéo. Très vite, son usage premier a été détourné dans divers projets informatiques. Dès lors, une version PC a été mise en vente permettant à tout un chacun d'explorer les possibilités offertes par ce composant.

Ce projet sera divisé en trois étapes. La première consistera en une prise en main de la kinect et de son kit de développement (SDK) ainsi qu'à une analyse de l'état de l'art. La deuxième partie verra le développement d'applications tout en connaissant à ce moment-là, les limites de l'appareil. Finalement, une conclusion devra justifier l'utilité ou non de la kinect.

Pour une durée de 360 heures, cette recherche s'effectue dans le cadre d'un travail de bachelor en informatique de gestion à la HES-SO // Valais.

Analyse de la kinect

Dans cette première partie, le but sera de se préparer au mieux pour la phase de développement. Il plane actuellement une zone d'ombre concernant les réelles possibilités d'interaction avec un tel appareil.

La kinect sera décortiquée. Quelles sont ses capacités ? Comment interagir avec elle ? Que peut-on faire ou ne pas faire avec ? Une phase d'apprentissage, jalonnée de tutoriaux, sera nécessaire pour en comprendre le fonctionnement.

Pour cela, le SDK de Microsoft fournira les outils de base. Une contrainte, liée à la nouveauté de cet appareil, sera de trouver les sources d'informations nécessaires à l'auto-formation. Il sera nécessaire d'explorer un maximum les forums et les nombreux projets existants. L'état de l'art permettra de les dénicher et de pourquoi pas, s'en inspirer pour la partie de développement.

Développement

Une fois que la kinect sera appréhendée, il sera temps de passer à la réflexion sur le développement d'applications utiles et faisables. Sur la base de brainstorming et après validation des responsables de ce projet, l'analyse aura fourni le bagage permettant de conseiller et de développer les idées proposées.

Actuellement, aucune fonctionnalité de base n'est concrètement demandée. Bien sûr, des idées comme détecter des personnes près de la fenêtre pour baisser les stores en cas de soleil ou encore la gestion des lumières selon la présence de personnes dans la pièce ont été évoquées, mais seule la phase analytique saura nous dire s'il est réellement possible de les développer.

Toutes les deux à trois semaines, une réunion avec les responsables permettra de coordonner les informations réunies et de partager les connaissances accumulées jusque-là, en proposant à chaque fois une nouvelle démo, en quelque sorte, devenir un consultant kinect.

Conclusion

La partie finale sera principalement consacrée à la rédaction du rapport, plus particulièrement à la conclusion du projet. Sur la base de la phase de développement, une analyse devra permettre de dire si la kinect a son utilité dans le monde de la domotique ou non.

Tâches du planning

Gestion de projet

- Planning et suivi du projet
- Cahier des charges
- Documentation

Acquisition de compétences et analyse

- Se familiariser avec la kinect
- Se familiariser avec le SDK
- Utilisation des fonctions de base de la kinect et de démos existantes
- Analyse des projets existants

Développement

Par fonctionnalités :

- Analyse de la faisabilité
- Implémentation d'une démo
- Conclusion, nouveaux débouchés

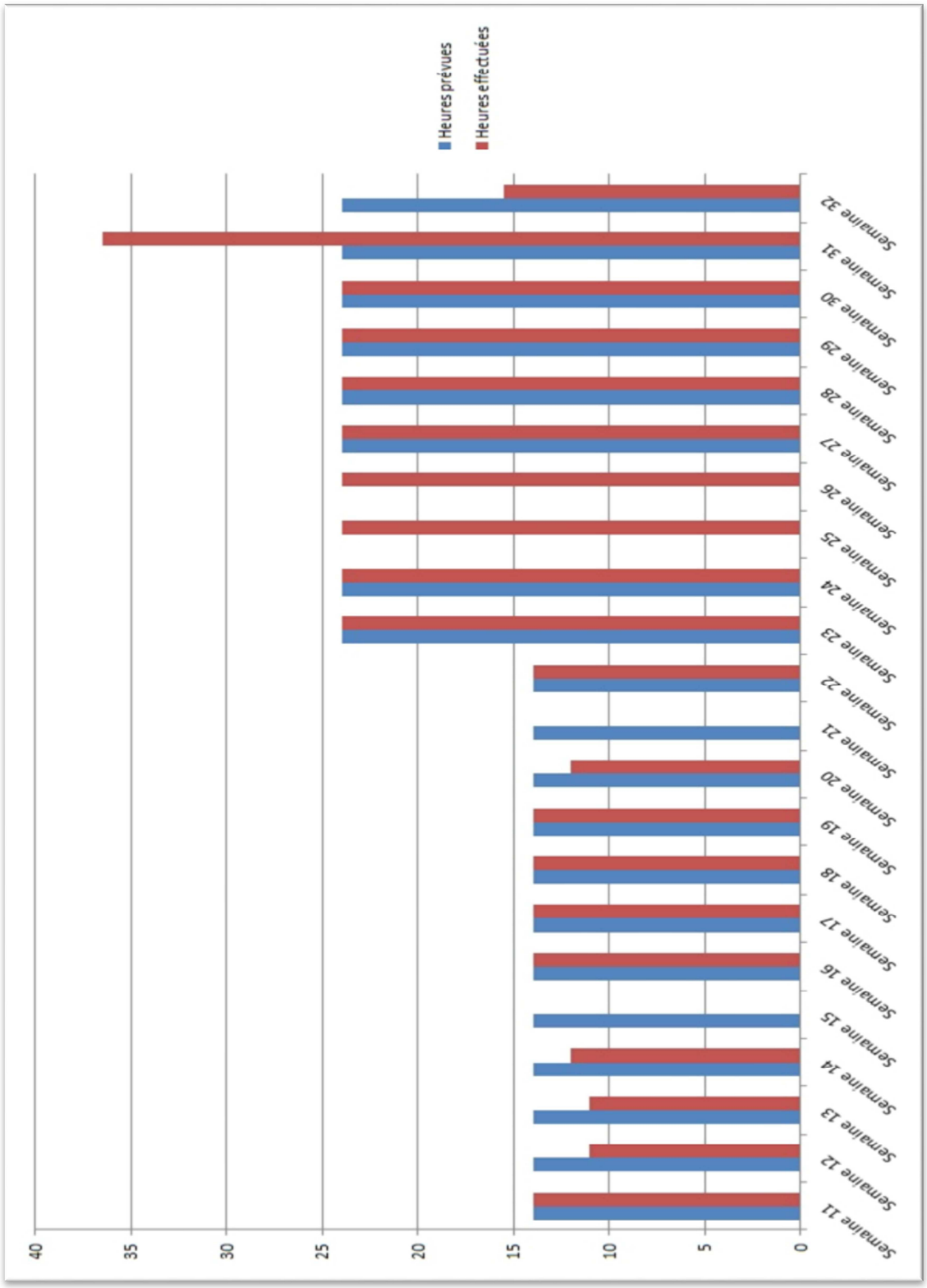
14.2. Planning initial

Planification du travail de bachelor : "Use the Kinect plateform ton icrease IPv6 devices awarness in a room"

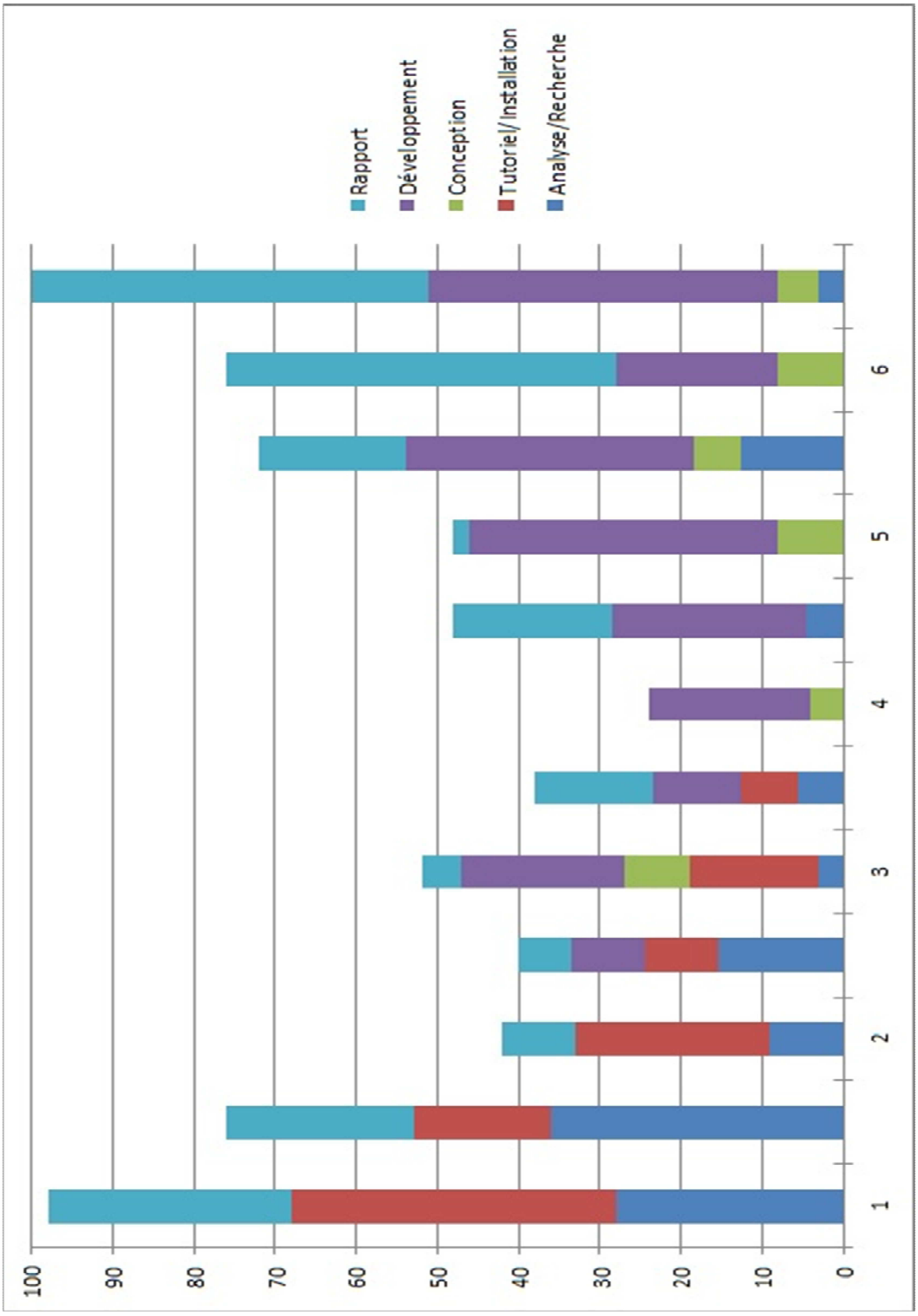
Semaine	Du	Au	Heures TB					
11	12.03.2012	17.03.2012	14	4	5	5		
12	19.03.2012	24.03.2012	14	4	5	5		
13	26.03.2012	31.03.2012	14	4	6	4		
14	02.04.2012	07.04.2012	14	4	6	4		
15	09.04.2012	14.04.2012	14	4	6	4		
16	16.04.2012	21.04.2012	14	4	6	4		
17	23.04.2012	28.04.2012	14	4	6	4		
18	30.04.2012	05.05.2012	14	3	8	3		
19	07.05.2012	12.05.2012	14	3	8	3		
20	14.05.2012	19.05.2012	14	3	8	3		
21	21.05.2012	26.05.2012	14	3	8	3		
22	28.05.2012	02.06.2012	14		8	2	4	
23	04.06.2012	09.06.2012	24				4	20
24	11.06.2012	16.06.2012	24				4	20
25	18.06.2012	23.06.2012	examen					
26	25.06.2012	30.06.2012	examen					
27	02.07.2012	07.07.2012	24			2	4	18
28	09.07.2012	14.07.2012	24				4	20
29	16.07.2012	21.07.2012	24				4	20
30	23.07.2012	28.07.2012	24				4	20
31	30.07.2012	04.08.2012	24			24		
32	06.08.2012	11.08.2012	24			24		
		Total heures	360					

	Heures plannifiées
Analyse, recherche	40
Tutoriels, installation	80
Conception	28
Développement	118
Rapport	94
Total	360
Phase 1: Analyse	120
Phase 2 : Développement	146
Rédaction	94

14.3. Planning final en semaine



14.4. Planning final en sprint



14.5. Croquis de la scène de test