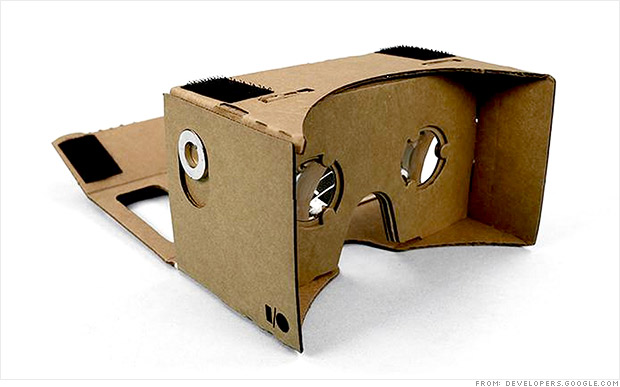
Rapport Vertigo



Domingos Ana Sofia

HEPIA 2014-2015

Projet de Semestre

Table des matières

[Informations 4](#_Toc413780652)

[Professeurs 4](#_Toc413780653)

[Elève 4](#_Toc413780654)

[Projet 4](#_Toc413780655)

[Description détaillée 4](#_Toc413780656)

[Hébergement 5](#_Toc413780657)

[Modèle de la vue 6](#_Toc413780658)

[Communication entre les composants 6](#_Toc413780659)

[CardBoard 6](#_Toc413780660)

[Description 6](#_Toc413780661)

[Utilité dans le projet 6](#_Toc413780662)

[Etudes, recherches & tests 7](#_Toc413780663)

[Montage 7](#_Toc413780664)

[Tests 7](#_Toc413780665)

[X3DOM 7](#_Toc413780666)

[Description 7](#_Toc413780667)

[Utilité dans le projet 7](#_Toc413780668)

[Etudes, recherches & tests 8](#_Toc413780669)

[Bilan 8](#_Toc413780670)

[Capteurs de mouvements 8](#_Toc413780671)

[Description 8](#_Toc413780672)

[Utilité dans le projet 8](#_Toc413780673)

[Etudes, recherches & tests 8](#_Toc413780674)

[Types 8](#_Toc413780675)

[Balle qui saute 9](#_Toc413780676)

[Bilan 9](#_Toc413780677)

[Github 9](#_Toc413780678)

[Description 9](#_Toc413780679)

[Utilité dans le projet 9](#_Toc413780680)

[WebHook sur Github 10](#_Toc413780681)

[Description 10](#_Toc413780682)

[Utilité dans le projet 10](#_Toc413780683)

[Bilan 10](#_Toc413780684)

[WebSocket (NodeJS & MeteorJS) 10](#_Toc413780685)

[Description 10](#_Toc413780686)

[NodeJS 10](#_Toc413780687)

[MeteorJS 10](#_Toc413780688)

[Utilité dans le projet 11](#_Toc413780689)

[Etudes, recherches & tests 11](#_Toc413780690)

[NodeJS 11](#_Toc413780691)

[MeteorJS 11](#_Toc413780692)

[Bilan 11](#_Toc413780693)

[Kinect 11](#_Toc413780694)

[Description 11](#_Toc413780695)

[Utilité dans le projet 11](#_Toc413780696)

[Etudes, recherches & tests 11](#_Toc413780697)

[Kinect v1 Prérequis 11](#_Toc413780698)

[Kinect v2 Prérequis 12](#_Toc413780699)

[API 12](#_Toc413780700)

[Détection d’une main EN COURS 12](#_Toc413780701)

[Bilan 12](#_Toc413780702)

[Conclusion 12](#_Toc413780703)

[Suivi projet 12](#_Toc413780704)

[Continuité 13](#_Toc413780705)

[Bilan du projet de semestre 13](#_Toc413780706)

[Annexes 13](#_Toc413780707)

Informations

* **Domingos Ana Sofia**: ana-sofia.domingos@etu.hesge.ch

Etudiante ITI travaillant sur le projet Vertigo en travail de semestre

* **Gobet Jeremy** : [jeremy.gobet.72@gmail.com](mailto:jeremy.gobet.72@gmail.com)

Assistant ITI ayant proposer le sujet en travail de semestre

* **Malandain Stephane** : [stephane.malandain@hesge.ch](mailto:stephane.malandain@hesge.ch)

Professeur ITI responsable de ce travail de semestre

Projet[[1]](#footnote-2)

Le but de ce projet est d’utiliser la Google CardBoard ~~de Google~~ et d’utiliser X3DOM pour modéliser une vue en 3D sur une page Web.

~~Pour les gens qui connaissent l’Occulus Rift, la CardBoard de Google la verison ship et home made. Pour ceux qui ne connaissent pas,~~ la CardBoard est un boîtier dans lequel on peut disposer un smartphone et s'en servir pour constituer une paire de lunette 3D de réalité virtuelle ~~simuler de la 3D.~~

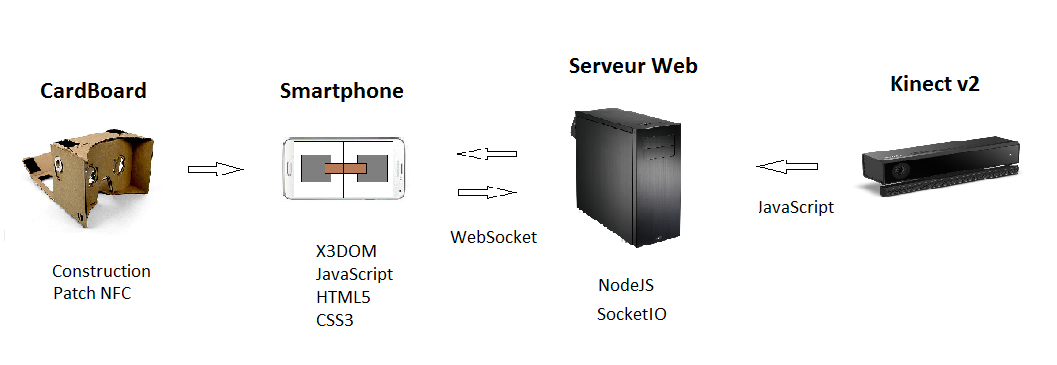
Le projet que nous avons décidé de réaliser, avec l'assistant Jérémy Gobet, est l’implémentation~~, la plus réalise possible,~~ d’un exercice qui permet de vaincre la peur du vide.   
Nous allons pour cela, modéliser la vue d’une planche de 3 mètres reliant 2 immeubles sur laquelle les personnes devront traverser pour arriver à l’immeuble d’en face.

Description détaillée

Pour rendre l’exercice ~~plus~~ réalise et ressentir les sensations de la planche sous nos pieds et de la peur de tomber, nous allons placer une planche sur le sol et les personnes devront marcher dessus en portant la Google CardBoard. Un appareil (Kinect) va récupérer les mouvements et les déplacement de la personne sur la planche pour les répercuter dans le monde virtuel. On peut aussi imaginer ajouter un ventilateur et des écouteurs pour augmenter l'immersion dans le monde virtuel.

Pour ce projet, ~~nous allons~~ je vais travailler sur plusieurs axes (technologies et langages) car les périphériques doivent communiquer entre eux. Le schéma ci-dessous nous montre les axes sur lesquelles ~~nous allons~~ je vais travailler ainsi que la communication entre les périphériques.

Schéma projet



(améliorer la qualité/taille du texte dans les images)

Afin de pourvoir avoir un projet complet, ~~nous l’avons~~ je l'ai séparé en 2 étapes.

La première, lors du projet de semestre, durant laquelle ~~nous allons~~ je vais étudier, définir et tester toutes les technologies et composants dont ~~nous aurons~~ j'aurai besoin pour arriver au bout de ce projet.   
Durant cette partie, ~~nous allons~~ je vais construire les bases ~~de notre~~ du projet. C’est-à-dire, la construction des lunettes, avoir une vue 3D qui s’adapte aux lunettes et aux mouvements et la communication entre les composants/technologies mise en place.

La deuxième, lors du projet de Bachelor, durant laquelle ~~nous allons~~ je vais travailler sur la partie de déplacement de la personne en utilisant une Kinect et la mise en place de la vue finale. ~~nous allons~~ Je vais donc reporter le déplacement de la personne sur la planche dans la vue sur la CardBoard.

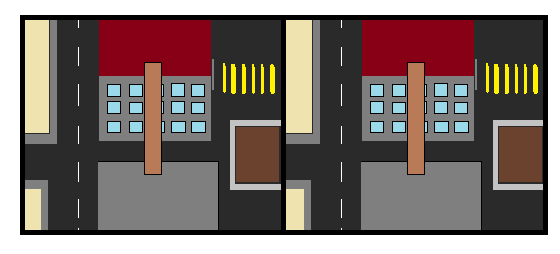
Hébergement

Lien dépôt git (privé)  : <https://github.com/PimsJay01/Vertigo>

Lien hébergement : <https://git.hepia.ovh:5001/>

Hôte : 80.80.228.13FH

Modèle de la vue



~~Nous allons~~ Je vais modéliser cette vue ou une vue qui y ressemble en utilisant ~~le langage~~ la technologie X3DOM ainsi que les langages web basiques (HTML, CSS et JavaScript) ~~l’HTML/CSS~~. Comme vous pouvez voir elle sera séparée en deux ~~par un trait~~ car c’est comme cela que nous la verrons dans les lunettes.

Nous allons essayer de rendre cette vue la plus réaliste possible.

(Je te laisse corrigé les nous en je dans la suite du texte)

Communication entre les composants

La vue que nous modélisons changera en fonction du mouvement de la tête en utilisant les capteurs de déplacement du smartphone sur lequel nous accéderons à la page de notre projet. ???

Nous avons également un serveur web ~~tournant sur un ordinateur qui envoi toute les 10 secondes~~ qui envoi de manière cyclique des messages au smartphone connecté.

Ce serveur nous permettra, lors du travail de Bachelor, de transmettre les ~~informations~~ données provenant de la Kinect ainsi nous pourrons déplacer le personnage et ses membres dans la réalité virtuelle ~~dans la vue~~ et pour y rendre le plus réel possible.

Google CardBoard

Description

Nous pouvons voir la GCardBoard comme un Oculus Rift pas cher.   
En effet, la CardBoard se construit à la maison avec très peu de matériel. Il est donc possible pour n’importe quelle personne d’en construire une.

Les CardBoard sont accessible par tout le monde à un prix très bas. Google permet via ses lunettes à tout le monde de pouvoir s’amuser avec la 3D sans avoir à se ruiner.

Utilité dans le projet

~~Le but de ce projet est d’utiliser ses lunettes de Google et pouvoir travailler sur de la 3D.~~ L'un des axes de ce projet de semestre est d'essayer les Google CardBoard et de s'en servir avec un smartphone comme lunette de réalité virtuelle.

Etudes, recherches & tests

Montage

Pour construire[[2]](#footnote-3) une Googel CardBoard il faut :

* Du carton pas trop épais (1-2mm)
* Des lentilles biconvexes
* Un tag NFC
* Un aimant en céramique
* Un aimant en néodyme
* Du velcro pour tenir le smartphone

Nous avons construit une CardBoard complète mais dans notre cas, nous n’utilisons pas tous les composants se trouvant sur la CardBoard. Seules les lentilles nous auraient suffi.

Les patrons et instructions de montage nous sont fournis par Google sur leur page[[3]](#footnote-4) parlant de la CardBoard. Il suffit simplement de télécharger les documents, les imprimé, les coller sur le carton, découper et assembler pour obtenir la CardBoard finale.

Tests

J’ai téléchargé l’application CardBoard sur Android qui m’a permet de tester si tout marchait bien et de voir le rendu possible avec un smartphone.

J’ai également testé un exemple de vue 3D ~~la vue~~ d’une salle de classe que l’on trouve sur le site de X3DOM.

X3DOM

Description

C’est un Framework open-source qui permet d’exécuter des scènes 3D sur un page WEB. Pour cela, il utiliser l’HTML5, la 3D est donc créer par des balises.

Cela permet d’avoir une scène de X3D en direct sur un code HTML DOM, nous pouvons donc manipuler le contenu 3D en utilisant les éléments DOM.

En tant que développeur Web, nous pouvons construire des pages Web et application qui incluent du contenu 3D en utilisant WebGL sans avoir besoin de plugins.

Utilité dans le projet

Nous allons utiliser ce Framework pour modéliser la vue en 3D ci-dessus. Dans notre cas, je vais modéliser cette vue sur Blender (juste la partie graphique) puis en utilisant les évènements DOM, je vais déplacer la vue en fonction des capteurs du smartphone.

Etudes, recherches & tests

Suivi de ~~certains~~ quelques tutos se trouvant sur la page[[4]](#footnote-5) X3DOM pour comprendre le fonctionnement de ce Framework ~~nouveau langage~~.

Bilan

Pour l’implémentation de la vue 3D, nous allons la modéliser ~~la vue~~ sur Blender. Blender est un logiciel 3D qui permet de modéliser une vue et d’exporter celle-ci ~~cette vue~~ en un fichier x3d, ce fichier contient toute la scène ~~les points de la vue.  
Après avoir~~ modélisé ~~la vue~~ sur Blender~~, nous l’importons le fichier dans notre balise X3DOM~~. Ce fichier peut ensuite être importé dans les blalise X3DOM et manipuler dans le navigateur.

Capteurs de mouvements

Description

Depuis des années, les smartphones ont des plus en plus de capteurs différents permettant de capter énormément de choses tels que les pas, le déplacement du smartphone et bien d’autres.

Utilité dans le projet

Comme nous avons décidé que notre projet se fera entièrement en JavaScript, nous avons dû trouver le moyen de détecter les mouvements de la tête via le navigateur web pour permettre d’adapter la vue lors du mouvement de la tête de la personne portant la CardBoard.

Etudes, recherches & tests

~~Vous ne le saviez sûrement pas mais~~ il existe beaucoup de capteur de mouvements différents intégré à notre smartphone.

Types

Accelerometer

Il mesure l’accélération appliquée au smartphone en prenant compte de la force de gravité. Il capte les valeurs le long des axes x, y et z.

Linear Accelerometer

Il nous fournis un vecteur tridimensionnel représentant l’accélération le long de chaque axe en excluant la gravité.

Gravity

Il mesure le vecteur à trois dimensions indiquant la direction et l’ampleur de la gravité.

Gyroscope

Il mesure la vitesse ou la rotation en radians par secondes autour des axes x, y et z du smartphone.

Uncalibrated Gyroscope

Similaire au gyroscope sauf qu’aucune compensation gyro-dérivé est appliqué à la vitesse de rotation. Il fournit des résultats plus bruts et peut inclure un certain biais. Il permet également de mesurer la dérive estimée autour de chaque axe.  
Il est utilisé pour le post-traitement et des données d’orientation de fusion.

Rotation Vector

Ce vecteur représente l’orientation de l’appareil comme une combinaison d’un angle et un axe dans lequel le smartphone a tourné autour d’un angle delta autour d’un axe.

Significant Motion

Il déclenche un événement spécifique à chaque mouvement détecté puis le désactive. Un évènement signifiant est un évènement qui peut entrainer un changement dans l’emplacement de l’utilisateur.

Step Counter

Il indique le nombre de mesures prises par l’utilisateur depuis le dernier redémarrage tandis que le capteur a été activé. Il a une latence (jusqu’à 10 secondes) mais est plus précis que le capteur de détection.

Step Detector

Il déclenche un événement à chaque fois que l’utilisateur fait un pas. La latence est inférieure à 2 secondes.

Balle qui saute

Pour tester le sensor que nous avons choisi d’utilisé, j’ai codé un petit « jeu ».

Il s’agit d’une balle dans un carré et en fonction des mouvements du smartphone, la balle bouge et rebondit sur les bords du carré.

Bilan

Nous avions d’abord décidé d’utiliser l’accéléromètre mais après des recherches, nous avons trouvé que le vecteur de rotation (Rotation Vector) sera plus adapté à notre cas car nous devons déplacer la vue en fonction de la rotation de la tête.

Github

Description

C’est un service web d’hébergement et de gestion de développement de logiciels basé sur le gestionnaire de version git.

Github permet de créer un projet sur lequel des personnes peuvent y contribuer. Il permet également d’avoir un « versionning » sur un projet, ce qui est primordial pour un projet de développement informatique ~~peut être très utile.~~

Utilité dans le projet

Permet d’avoir un « versionning » de Vertigo et de partager mon travail avec le corps d'ensignement qui suit ce projet et son avancement. ~~avec mes professeurs pour qu’ils puissent suivre mon avancement.~~

WebHook sur Github

Description

C’est une méthode qui permet étendre, personnaliser et intégrer une application web lorsqu’il se produit un événement, une notification est envoyer à une URL spécifique via HTTP POST.

Le WebHook, est utilisé généralement de 3 façons :

* Push : réception de données en temps réel
* Plugins : traitement de données et envoyer quelque chose en retour
* Pipes : recevoir des données et les transmettre

Utilité dans le projet

Lors de la modification du projet dans le dépôt git, WebHook permet modifier sur le serveur hébergeant notre site les fichiers modifiés.

Bilan

Cette méthode permet de ne pas avoir à importer manuellement via une application FTP ~~Filezilla~~ les fichiers ~~ayant changé~~ sur le serveur hébergeant du ~~notre~~ site web. Ce qui permet de gagner du temps, il suffit de push les modifications sur Github et nous pouvons directement tester sur ~~notre~~ l'hébergeur ~~les nouvelles modifications~~.

WebSocket (NodeJS & MeteorJS)

Description

NodeJS

C’est une plateforme[[5]](#footnote-6) logicielle et évènementielle en JavaScript. Elle contient une bibliothèque de serveur HTTP, ce qui permet de faire tourner et de mieux contrôler un serveur web sans avoir besoin d’un logiciel externe.

Cette plateforme permet d’utiliser plusieurs modules tel que

* Socket.io qui permet d’utiliser les WebSocket et de facilement les manipuler
* ExpressJS qui permet de créer et gérer une application web plus facilement
* AngularJS qui permet d’étendre le langage HTML par de nouvelles balises et attributs
* MongoDB qui permet de gérer des bases de données
* Et bien d’autres...

~~Cette plateforme~~ Celle-ci est de plus en plus populaire en tant que plateforme serveur.

MeteorJS

C’est un Framwork ~~une plateforme~~[[6]](#footnote-7) simple pour la construction des sites Web modernes, qui se distingue parmi les autres avec son point de vue sur la création d’application web.

Il connecte de façon transparente le client avec node/MongoDB et promet des applications rapides.

~~MeteorJS s’occupe de tout pour nous.~~

Utilité dans le projet

Nous avons besoin d’un serveur pour que nous puissions envoyer les données du Kinect au serveur hébergeant le projet vertigo via les WebSockets. De manière générale la communication se fait toujours du client vers le serveu. Le serveur ce content de répondre aux requêtes des clients. Pour ce projet, nous avons besoin que le serveur envoi des informations de manière cyclique à l'utilisateur.

Etudes, recherches & tests

NodeJS

Pour tester NodeJS et comprendre comment il fonctionne, j’ai suivi un tutoriel [[7]](#footnote-8)qui m’a permis de créer un chat client-serveur. J’ai choisi de faire ce tutoriel parce que le serveur web que nous voulons mettre en place utilise le même système de commnunication : ~~est une sorte de chat où~~ le serveur envoie des informations au client.

MeteorJS

Pour tester MeteorJS et le comprendre, j’ai suivi le tutoriel [[8]](#footnote-9)disponible sur leur site. Il m’a permet de créer un serveur web rapidement sans avoir à coder beaucoup.

Bilan

Lors des tests de ses deux plateformes, nous nous sommes rendu compte que MeteorJS ne correspondait pas à ce que nous voulions faire. En effet, MeteorJS gère tout pour nous alors que nous voulons pouvoir gérer totalement notre serveur Web.

Nous utiliserons donc NodeJS qui permet de ~~totalement~~ gérer à plus bas niveau notre serveur web. Nous allons surtout utiliser le module Socket.io qui nous permet de facilement manipuler les WebSockets. (décris ce que c'est)

Kinect

Description

C’est un périphérique initialement destiné à la console de jeux Xbox 360 permettant d'agir sur ~~de contrôler~~ des jeux vidéo sans utiliser de manette.

Depuis quelques années, Microsoft permet d’utiliser ce périphérique pour l’intégrer dans un projet ou une application.

Utilité dans le projet

Nous avons besoin d’un outil pour capter les mouvements ~~sur la planche de l’utilisateur~~???pour les reporter sur la vue en 3D.

Etudes, recherches & tests

Kinect v1 Prérequis[[9]](#footnote-10)

|  |  |
| --- | --- |
| **OS** | Windows 7 ou plus |
| **Processeurs** | 32 ou 64 bit |
| **Type Processeur** | Dual-core, 2.66-GHz ou plus rapide |
| **RAM** | 2 GB |
| **USB dédié** | 2.0 |
| **Carte graphique** | DirectX 9.0c |
| **Software** | Virtual Studio 2010 ou 2012 |

Kinect v2 Prérequis[[10]](#footnote-11)

|  |  |
| --- | --- |
| **OS** | Windows 8 ou plus haut |
| **Processeurs** | 64 bits |
| **Type Processeur** | I7 3.1 GHz ou plus |
| **RAM** | 4 GB |
| **USB dédié** | 3.0 |
| **Carte graphique** | DX11 capable |
| **Software** | Virtual Studio 2012 |

API

Lors de ma recherche, j’ai trouvé que pour les deux versions du Kinect il y a une API Java et une API JavaScript. Ses deux API sont bien documentées, nous pouvons y trouver beaucoup d’exemples.

Détection d’une main EN COURS

Pour tester le Kinect, j’ai codé une petite application qui permet de détecter une main.

Bilan

Nous allons utiliser la Kinect v1 pour plusieurs raisons qui sont les suivantes :

* Kinect v2 trop strict
* Je travaille sur Windows 7 et ???
* Nous avons un Kinect v1 disponible donc pas besoin d’acheter la v2

Nous avons également décidé d’utiliser d’API JavaScript parce que la communication entre le smartphone et le serveur web de notre application se fait en JavaScript, il est donc plus adapter d’utiliser du JavaScript pour ne pas se perdre dans une multitude de langages ~~du côté du Kinect également~~ .

Conclusion

Suivi projet

Durant tout le long du projet, pour le suivi, j’ai tenu un cahier de bord pour chaque entretien. Cela nous a permis de suivre le flux du projet et de se rappeler ce dont nous avons parlé lors des entretiens.

Continuité

Comme je l’ai mentionné dans la description du projet, je vais continuer ce projet lors du travail de Bachelor.

Le travail a effectué pour que Vertigo soit opérationnel est le suivant :

* Détecter les mouvements de l’utilisateur lors du déplacement sur la planche via le Kinect
* Mettre en place la scène ~~vue~~ 3D finale
* Transmettre les déplacements et mouvement via le serveur web sur lequel est connecté le Kinect.
* Reporter les déplacements et mouvements détecté par le Kinect sur la vue 3D du smartphone.
* Chercher des solutions pour augmenter l'immersion de l'utilisateur dans le monde virtuel.

Bilan du projet de semestre

C’est un projet très intéressant, il m’a permis d’apprendre de nouveaux langages et technologies avec lesquels j’aimerais travailler plus tard. En effet, ce que j’ai appris lors de ce projet est de plus en plus utiliser de nos jours et est donc un sujet qu’il faut maîtriser si je veux évoluer dans mon métier.

De plus, le fait que ce projet puisse peut-être aider les gens ayant la peur du vide a rendu ce projet plus motivant.

1. Exemple du projet avec l’Oculus Rift : <https://www.youtube.com/watch?v=7oqazo3ZEnY> [↑](#footnote-ref-2)
2. Matériel pour la construction : [http://www.amazon.com/AM-CARDBOARD%C2%AE-Complete-Cardboard-Project/dp/B00LM36DUK/ref=pd\_sim\_cps\_14?ie=UTF8&refRID=1SNC11VTMPQND1MF5ZTE](http://www.amazon.com/AM-CARDBOARD®-Complete-Cardboard-Project/dp/B00LM36DUK/ref=pd_sim_cps_14?ie=UTF8&refRID=1SNC11VTMPQND1MF5ZTE)Construction : <https://www.youtube.com/watch?v=3YopUPZErwI> [↑](#footnote-ref-3)
3. Page de Google sur la CardBoard : <https://www.google.com/get/cardboard/get-cardboard.html> [↑](#footnote-ref-4)
4. Tutoriel X2DOM : <http://doc.x3dom.org/tutorials/index.html> [↑](#footnote-ref-5)
5. Site NodeJS : <https://nodejs.org/> [↑](#footnote-ref-6)
6. Site de MeteorJS : <https://www.meteor.com/> [↑](#footnote-ref-7)
7. Tutoriel NodeJS : <http://socket.io/get-started/chat/> [↑](#footnote-ref-8)
8. Tutoriel MeteorJS : <https://www.meteor.com/try> [↑](#footnote-ref-9)
9. Prérequis kinect v1 : <https://msdn.microsoft.com/en-us/library/hh855359.aspx> [↑](#footnote-ref-10)
10. Prérequis Kinect v2 : <https://msdn.microsoft.com/en-us/library/dn782036.aspx> [↑](#footnote-ref-11)