

ECOLE POLYTECHNIQUE DE L'UNIVERSITÉ FRANÇOIS RABELAIS DE TOURS

Département Informatique

64 avenue Jean Portalis

37200 Tours, France

Tél. +33 (0)2 47 36 14 14

polytech.univ-tours.fr

Projet Recherche & Développement

2017-2018

Applications de médiation pour l'utilisation de périphériques sans contact en musée

Tuteurs académiques

Gilles VENTURINI

Barthelemy SERRES

Étudiant

Logan VERECQUE (DI5)

2 avril 2018



Liste des intervenants

Nom	Email	Qualité
Logan VERECQUE	logan.verecque@etu.univ-tours.fr	Étudiant DI5
Gilles VENTURINI	gilles.venturini@univ-tours.fr	Tuteur académique, Département Informatique
Barthelemy SERRES	barthelemy.serres@univ-tours.fr	Tuteur académique, Département Informatique



Avertissement

Ce document a été rédigé par Logan Verecque susnommé l'auteur.

L'Ecole Polytechnique de l'Université François Rabelais de Tours est représentée par Gilles Venturini et Barthélemy Serres susnommés les tuteurs académiques.

Par l'utilisation de ce modèle de document, l'ensemble des intervenants du projet acceptent les conditions définies ci-après.

L'auteur reconnaît assumer l'entière responsabilité du contenu du document ainsi que toutes suites judiciaires qui pourraient en découler du fait du non respect des lois ou des droits d'auteur.

L'auteur atteste que les propos du document sont sincères et assument l'entière responsabilité de la véracité des propos.

L'auteur atteste ne pas s'approprier le travail d'autrui et que le document ne contient aucun plagiat.

L'auteur atteste que le document ne contient aucun propos diffamatoire ou condamnable devant la loi.

L'auteur reconnaît qu'il ne peut diffuser ce document en partie ou en intégralité sous quelque forme que ce soit sans l'accord préalable des tuteurs académiques et de l'entreprise.

L'auteur autorise l'école polytechnique de l'université François Rabelais de Tours à diffuser tout ou partie de ce document, sous quelque forme que ce soit, y compris après transformation en citant la source. Cette diffusion devra se faire gracieusement et être accompagnée du présent avertissement.



Pour citer ce document

Logan Verecque, *Applications de médiation pour l'utilisation de périphériques sans contact en musée*, Projet Recherche & Développement, Ecole Polytechnique de l'Université François Rabelais de Tours, Tours, France, 2017-2018.

```
@mastersthesis{
  author={Verecque, Logan},
  title={Applications de médiation pour l'utilisation de périphériques sans contact en
  musée},
  type={Projet Recherche \& Développement},
  school={Ecole Polytechnique de l'Université François Rabelais de Tours},
  address={Tours, France},
  year={2017-2018}
}
```


Table des matières

Liste des intervenants	a
Avertissement	b
Pour citer ce document	c
Table des matières	i
Table des figures	vi
1 Introduction	1
1 Acteurs, enjeux et contexte	1
2 Objectifs	1
3 Hypothèses	2
4 Base méthodologique	2
2 Description générale	3
1 Environnement du projet	3
1.1 Application existante de médiation Kinect.....	3
1.2 Application interactive Kinect sur statue	3
1.3 Application interactive Leap Motion sur tableau	4
1.4 Application interactive Leap Motion sur statue	4
2 Caractéristiques des utilisateurs	4
2.1 Administrateur du système	4
2.2 Visiteur de l'exposition.....	4
3 Fonctionnalité du système.....	5
3.1 Application de médiation Kinect	5

3.2	Application de découverte Leap Motion	5
4	Structure générale du système	6
3	État de l'art	8
1	Kinect	8
1.1	Kinect dans les musées du monde	8
1.2	Développer avec Kinect	16
1.2.1	Aider les musées, aider l'Histoire	16
1.2.2	Aide au développement	16
1.2.3	La gestuelle avec Kinect	17
1.2.4	Alternative et futur	17
2	Leap Motion	17
2.1	Leap Motion dans les musées du monde	17
2.2	Développer avec Leap Motion	19
2.2.1	Vue d'ensemble	19
2.2.2	Langages	21
2.2.3	Agatha	22
2.2.4	Architecture	22
4	Analyse et conception	23
1	Généralités	23
2	Diagrammes de Classes	23
2.1	Application de médiation Kinect	23
2.2	Application de découverte Leap Motion	25
2.2.1	UIInterface	26
2.2.2	Draw	26
2.2.3	ImageManipulation	26
2.2.4	LeapMotion	26
2.2.5	HandSkeleton	26
2.2.6	Configuration	27
5	Mise en œuvre	28
1	Implémentation	28
1.1	Limites	28
1.1.1	Humaines	28
1.1.2	Techniques	28
1.2	Risques	29
1.3	Choix technique	30
1.4	Choix design	30
2	Stratégie et performance	30

2.1	Stratégie	30
2.2	Performance	31
3	Analyse des résultats	31
3.1	Fonctionnement	31
3.2	Écran de veille.....	32
3.3	Étape 1 - Bouger!	32
3.4	Étape 2 - Déplacer la souris	33
3.5	Étape 3 - Serrer les poings	34
3.6	Étape 4 - Déplacer un objet	35
3.7	Étape 5 - Zoom/Dézoom	36
3.8	Étape 6 - Tourner un objet.....	36
3.9	Sortie de zone.....	37
3.10	Écran de fin.....	37
3.11	Conclusion	38
6	Bilan et conclusion	39
1	Travail réalisé.....	39
1.1	Première partie - Semestre 9	39
1.1.1	Analyse de l'existant	39
1.1.2	État de l'art.....	39
1.1.3	Interfaces	40
1.1.4	Analyse et conception.....	40
1.2	Seconde partie - Semestre 10.....	40
1.2.1	Application de médiation Kinect	40
1.2.2	Application de découverte Leap Motion.....	40
1.2.3	Tests et qualité	41
2	Gestion de projet	41
2.1	Planning du Semestre 10	41
2.2	Diagramme de Gantt	41
7	Annexes	42
1	Spécifications fonctionnelles.....	42
1.1	Application de médiation Kinect	42
1.1.1	Étape 1 — Déplacement des mains.....	42
1.1.2	Étape 2 — Contrôle du curseur de la souris	42
1.1.3	Étape 3 — Ouverture et fermeture des poings	43
1.1.4	Étape 4 — Déplacement d'une image	43
1.1.5	Étape 5 — Changement de taille d'une image.....	44
1.1.6	Étape 6 — Rotation d'une image.....	44
1.1.7	Entrée dans le champ de détection de la caméra Kinect	45

1.1.8	Sortie du champ de détection de la caméra Kinect.....	45
1.2	Application de découverte Leap Motion.....	46
1.2.1	Menu	46
1.2.2	Dessiner	46
1.2.3	Manipulation d'images.....	46
1.2.4	Entrée dans le champ de détection du Leap Motion	47
1.2.5	Sortie du champ de détection Leap Motion	47
2	Spécifications non fonctionnelles.....	48
2.1	Contraintes de développement et conception.....	48
2.1.1	Matériels	48
2.1.2	Langages de programmation.....	48
2.1.3	Logiciels et bibliothèques	48
2.1.4	Environnement	48
2.2	Contraintes de fonctionnement et d'exploitation	48
2.2.1	Performances	48
2.2.2	Capacités.....	49
2.2.3	Modes de fonctionnement	49
2.2.4	Contrôlabilité.....	49
2.2.5	Sécurité	50
2.2.6	Intégrité	50
2.3	Maintenance et évolution du système.....	50
3	Interfaces	50
3.1	Interfaces matériel/logiciel	50
3.2	Interfaces homme/machine.....	52
3.2.1	Application de médiation Kinect - Description.....	52
3.2.2	Application de médiation Kinect - Maquettes des interfaces.....	53
3.2.3	Application de découverte Leap Motion - Description	54
3.2.4	Application de découverte Leap Motion - Maquettes des interfaces	55
3.3	Interfaces logiciel/logiciel	56
4	Gestion de projet	56
4.1	Méthodologie agile	56
4.2	Livrables.....	57
4.3	Analyse de faisabilité.....	57
4.4	Suivi de projet.....	57
4.5	Outils	58
4.5.1	Versionning.....	58
4.5.2	Organisation	58
4.6	Diagramme de Gantt / Semestre 9 - PRD1	58
4.6.1	Sprint 1	58

4.6.2	Sprint 2	58
4.6.3	Sprint 3	58
4.6.4	Sprint 4	59
4.6.5	Sprint 5	59
4.6.6	Sprint 6	59
4.6.7	Sprint 7	59
4.6.8	Diagramme de Gantt	59
4.7	Diagramme de Gantt / Semestre 10 - PRD2	60
4.7.1	Sprint 1	60
4.7.2	Sprint 2	60
4.7.3	Sprint 3	60
4.7.4	Sprint 4	60
4.7.5	Diagramme de Gantt	61
5	Guide d'installation.....	61
5.1	Matériel nécessaire	61
5.2	Outils de développement	62
6	Guide du développeur.....	62
6.1	Généralités	62
6.1.1	Langage, librairies et convention de nommage	62
6.1.2	Documentation	63
6.2	Structure du système	64
7	Manuel Utilisateur.....	66
7.1	Mise en place	66
7.2	Généralités	66
7.3	Étape 1 — Bouger	66
7.4	Étape 2 — Déplacer la souris	66
7.5	Étape 3 — Serrer les poings.....	67
7.6	Étape 4 — Déplacer un objet.....	67
7.7	Étape 5 — Agrandissement/Réduction.....	67
7.8	Étape 6 — Tourner un objet	68
7.9	Fin du tutoriel.....	69
7.10	Sortie de la zone de contrôle.....	70
8	Cahier de tests	70
8.1	Scénarios de test.....	70
8.2	Résultats des tests.....	73
	Webographie	89
	Bibliographie	90

Table des figures

2 Description générale

1	Application de médiation Kinect - Diagramme des cas d'utilisation	5
2	Application de découverte Leap Motion - Diagramme des cas d'utilisation	6
3	Application de médiation Kinect - Diagramme de Séquence	7
4	Application de découverte Leap Motion - Diagramme de Séquence	7

3 État de l'art

1	Exposition Marioneta - Children's Museum of Pittsburgh.....	9
2	Ptérosauresy - American Museum of Natural History of New-York	10
3	Dessin sur toile blanche - Museum of Art of Cleveland.....	11
4	Manipulation de poteries d'argile - Museum of Art of Cleveland	11
5	Projection de peinture sur toile - Museum of Art of Cleveland	11
6	Digital Interlud - Musée Petiet de Limoux	12
7	Map Interaction - Museum for Communication of Frankfurt am Main	13
8	DinoStomp - Museum of Science and History of Fort Worth.....	14
9	DinoStomp - Museum of Science and History of Fort Worth.....	14
10	ARSandbox - Sacramento City College, Los Rios, CA	15
11	ARSandbox - Bold Park Community School - Perth, Australia	15
12	Correspondance des langages face aux exemples de code de Kinect	16
13	Comparaison Microsoft Kinect Vs ASUS Xtion	17
14	Effectorium - Mendelssohn House.....	18
15	Visite virtuelle - Atelier Antoine Bourdelle.....	19
16	Leap Motion - Système de coordonnées.....	20
17	Leap Motion - Reconnaissance d'une main	20

18	Leap Motion - Reconnaissance des doigts	20
19	Leap Motion - Reconnaissance d'un objet	21
20	Leap Motion - Liste de mouvements possibles	21
21	Leap Motion - Architecture d'une application	22
4	Analyse et conception	
1	Application de médiation Kinect - Diagramme de classes	24
2	Application de médiation Kinect - Diagramme de classes final	25
3	Application de découverte Leap Motion - Diagramme de classes	26
5	Mise en œuvre	
1	Écran de veille	32
2	Étape 1 - Bouger!	33
3	Étape 2 - Déplacer la souris!	34
4	Étape 3 - Serrer les poings	35
5	Étape 4 - Déplacer un objet	35
6	Étape 5 - Zoom/Dézoom	36
7	Étape 6 - Tourner un objet	37
8	Écran de sortie de zone	38
7	Annexes	
1	Application de médiation Kinect - Architecture de déploiement	51
2	Application de médiation Kinect - Diagramme de déploiement	51
3	Application de découverte Leap Motion - Architecture de déploiement	51
4	Application de découverte Leap Motion - Diagramme de déploiement	52
5	Maquette de l'interface de l'écran d'accueil et d'attente d'utilisateur	53
6	Maquette de l'interface de l'étape 1 du tutoriel	53
7	Maquette de l'interface de l'étape 2 du tutoriel	53
8	Maquette de l'interface de l'étape 3 du tutoriel	53
9	Maquette de l'interface de l'étape 4 du tutoriel	53
10	Maquette de l'interface de l'étape 5 du tutoriel	54
11	Maquette de l'interface de l'étape 6 du tutoriel	54
12	Maquette de l'interface de l'écran de réussite et de fin du tutoriel	54
13	Maquette de l'interface de l'écran indiquant la sortie de l'utilisateur de la zone de contrôle de Kinect	54
14	Maquette de l'interface de l'écran d'accueil et d'attente d'utilisateur	55
15	Maquette de l'interface du menu principal	55
16	Maquette de l'interface du module de dessin	55
17	Maquette de l'interface du module de manipulation d'image	56

18	Maquette d'exemple de la perte de reconnaissance du Leap Motion.....	56
19	Diagramme de Gantt	59
20	Légende - Diagramme de Gantt	61
21	Diagramme de Gantt S10	61
22	Architecture de l'installation	62
23	Exemple documentation IntelliSense dynamique	63
24	Regions module Animation.....	63
25	Diagramme de classe	64
26	Squelette de l'utilisateur	67
27	L'utilisateur déplace la statue	68
28	Zone de réception de la statue	68
29	Exemple rotation (vue de côté)	69
30	Exemple rotation (vue de face).....	69
31	Exemple rotation (vue de dessus)	69
32	Code couleur.....	73
33	Résultat Test Sprint 1 (1).....	74
34	Résultat Test Sprint 1 (2).....	75
35	Résultat Test Sprint 1 (3).....	76
36	Résultat Test Sprint 1 (4).....	77
37	Résultat Test Sprint 2 (1).....	78
38	Résultat Test Sprint 2 (2).....	79
39	Résultat Test Sprint 2 (3).....	80
40	Résultat Test Sprint 2 (4).....	81
41	Résultat Test Sprint 3 (1).....	82
42	Résultat Test Sprint 3 (2).....	83
43	Résultat Test Sprint 3 (3).....	84
44	Résultat Test Sprint 3 (4).....	85
45	Résultat Test Sprint 4 (1).....	86
46	Résultat Test Sprint 4 (2).....	87
47	Résultat Test Sprint 4 (3).....	88

1

Introduction

Ce document présente la continuité de la conception d'une application permettant l'apprentissage rapide de l'outil Kinect aux visiteurs de l'exposition prévue au Musée des Beaux-Arts de Tours ainsi que la création d'une application de découverte de la technologie Leap Motion. Cette première application fait suite au travail réalisé par M. Arnaud Talon. Elle devra rendre possible une initiation simple, rapide et efficace aux visiteurs souhaitant utiliser les installations utilisant Kinect. La seconde quant à elle devra montrer les possibilités offertes par le Leap Motion aux visiteurs. Ce projet est encadré par M. Gilles Venturini et M. Barthelemy Serres, enseignants chercheurs à l'école Polytechnique de Tours.

1 Acteurs, enjeux et contexte

À une époque où la population se désintéresse de plus en plus de l'art, certaines organisations se doivent de réagir. C'est le cas notamment des musées qui n'hésitent plus aujourd'hui à innover et moderniser leurs expositions afin de relancer leur attractivité et toucher de plus en plus de monde voire pourquoi pas attirer un nouveau public. Grâce à l'émergence des nouvelles technologies de réalités augmentées et virtuelles, les musées ont donc pu peu à peu s'approprier de nouvelles façons de concevoir leurs expositions et ainsi les rendre plus ludiques et attractives. Nous voyons donc émerger depuis quelque temps des œuvres ou écrans si ce n'est tout simplement des salles tout entières manipulables par des utilisateurs. Si le public est en majorité conquis par ces nouveautés, leur permettant de s'intéresser à des créations qu'ils n'auraient certainement pas été voir sinon certains restent réticent face à une technologie qui peut faire peur en matière d'ergonomie et de facilité d'utilisation.

Notre premier outil leur permettra de s'initier en douceur à la technologie Kinect en particulier en quelques minutes tandis que notre deuxième application leur permettra de découvrir quelques possibilités qu'offre la technologie Leap Motion à partir de quelques applications de bases.

2 Objectifs

Ce projet a pour objectif de reprendre le travail réalisé précédemment par Arnaud Talon et de le mener à son terme. Cette application, qui sera accessible à l'entrée du musée, devra initier les

visiteurs à l'outil Kinect qu'ils auront l'occasion de manipuler durant l'exposition. Elle devra permettre aux utilisateurs de comprendre les mouvements qu'ils pourront utiliser avec Kinect lors de l'exposition. Parmi ces mouvements, on retrouve les classiques : translation, rotation, agrandissement de taille, réduction de taille. Dans cette optique, ceux-ci auront l'occasion de manipuler le Kinect pendant cette phase d'apprentissage, en reproduisant les mouvements demandés à l'écran. Cette application sera donc interactive, totalement intuitive et ergonomique afin d'accompagner l'utilisateur dans son apprentissage de l'outil.

Concernant le Leap Motion, les utilisateurs devront pouvoir découvrir la technologie par eux-mêmes, à travers des applications simples telles que dessiner ou manipuler différentes images. En effet, la manipulation du Leap Motion étant relativement proche de celles effectuées sur smartphone elle est beaucoup plus intuitive et simple à prendre en main. Cependant, le but est tout de même qu'ils soient à terme au niveau pour utiliser les applications majeures de l'exposition. Nous essaierons donc implicitement de leur apprendre les mouvements et les bonnes façons de procéder.

3 Hypothèses

Deux éléments cruciaux rentrent en compte. Dans un premier temps, il faudrait avoir toujours à disposition une caméra Kinect prête à l'emploi en renfort. En effet, la caméra n'étant plus produite par Microsoft il sera compliqué de la remplacer en urgence si celle-ci venait à mourir pour une quelconque raison. Dans un deuxième temps, l'application de médiation Kinect est primordiale sur le reste, si celle-ci prend du retard que ce soit lié au développement ou à une phase de test elle prendra le dessus sur le développement de l'application Leap Motion.

4 Base méthodologique

Au niveau des langages utilisés pour nos logiciels, la question ne se pose pas pour l'application de médiation Kinect, nous allons reprendre le code en VB.NET existant. En revanche, nous opterons plutôt pour le langage C# pour l'application Leap Motion, ce qui nous permettra notamment de récupérer des bibliothèques ou des outils déjà existants pour nous aider.

Nous utiliserons la méthodologie Agile et plus particulièrement la méthode SCRUM pour la réalisation. Cela nous permettra de nous donner des objectifs concrets et à court terme tout en nous permettant de nous assurer à chaque fois une phase de test concrète. Nous utiliserons l'outil web Trello afin d'illustrer tout ça. Cela nous permettra de gérer de manières simples les différents sprints, les différentes tâches et gérer l'avancée de notre projet de manière efficace. Trello est également tout indiqué pour gérer les changements et les imprévus en cours de projet.

Le gestionnaire de version que nous utiliserons sera Gitlab. Il nous permettra de maintenir à jour les différentes versions de notre outil de la même manière que Github, mais il a en plus la particularité de permettre d'utiliser un dépôt privé gratuitement. Ainsi, l'outil ne sera pas exposé sur le réseau. Si nous avons besoin de réaliser différents traitements d'images, nous utiliserons le logiciel Gimp. L'ensemble des graphiques ou diagrammes seront réalisés grâce à l'outil en ligne draw.io qui permet en plus d'être gratuit, de sauvegarder directement nos travaux sur des plateformes différentes telles que Google Drive, Github, Dropbox, OneDrive ou encore Trello.

Enfin, les documents écrits seront réalisés en LaTeX.

2

Description générale

1 Environnement du projet

Le projet d'application pour Kinect se situe en amont des applications Kinect pour le musée des beaux-arts de Tours. L'intérêt est que les utilisateurs utilisent cet outil afin de se former et ainsi pouvoir ensuite utiliser pleinement et sereinement les différentes applications développées par M. Gilles Venturini et M. Barthélemy Serres qui interagissent avec les œuvres du musée.

Ce projet est la suite du projet de recherche et développement réalisé par M. Arnaud Talon, il poursuivra donc la même logique pour les outils de développement : il sera reconduit en VB.NET par l'intermédiaire de l'outil de développement Visual Studio.

En revanche, l'application Leap Motion est intégralement nouvelle par rapport à l'existant et s'intègre de manière totalement indépendante par rapport à notre projet. L'objectif est similaire, mais la manière de faire change. Il n'est plus question d'un tutoriel complet scénarisé, mais plutôt d'une série de petites applications de découverte utilisable à souhait et n'importe quand. Nous partirons ici sur le développement d'un module de dessin et d'un module de manipulations d'image, mais il n'est pas exclu que nous utilisions des modules déjà existants.

1.1 Application existante de médiation Kinect

L'existant qui nous intéresse et qui nous concerne directement est le travail réalisé par M. Arnaud Talon. Comme rapporter dans les différents écrits qu'il a pu rédiger le développement de l'application n'a pas pu aller à son terme, mais a quand même été suffisamment loin pour pouvoir repartir avec des bases solides et commencer à développement tout de suite. Dans cette production, on retrouve donc :

- La calibration et la reconnaissance des mouvements en dehors de celui de rotation
- Les bases des interfaces des différentes étapes du tutoriel en dehors de la rotation

1.2 Application interactive Kinect sur statue

C'est l'application pour laquelle nous faisons notre application d'initiation à la technologie Kinect. Cette application développée par M. Barthélemy Serres sous l'outil de développement

Unity permet de visualiser une statue en 3D et d'interagir avec elle à distance. Nous pouvons la déplacer et la tourner à volonté selon les mouvements décrits lors du tutoriel et obtenir des informations sur elle et son histoire.

1.3 Application interactive Leap Motion sur tableau

Cette application est développée par M. Gilles Venturini. Elle permet, à partir de la technologie Leap Motion, d'interagir à distance avec un tableau fixé sur un mur. Le programme projette une visualisation du tableau qui vient se superposer, grâce au vidéoprojecteur, sur le tableau original. Ainsi, lorsque l'utilisateur utilise l'application il a ce sentiment d'expérimenter sur l'œuvre originale. L'objectif de cette application est de parcourir différentes parties de l'œuvre à distance grâce au Leap Motion et d'obtenir des informations historiques sur celles-ci. On pourra, par exemple, découper un tableau en quatre parties qui correspondent à quatre aspects précis ou symboles historiques et obtenir des informations sur chacun d'entre eux.

1.4 Application interactive Leap Motion sur statue

Cette application est également développée par M. Gilles Venturini. Celle-ci est presque semblable en tout point à la précédente sauf que l'on utilise une représentation d'une statue au lieu d'un tableau. La possibilité en plus qui est offerte est de pouvoir repeindre la statue grâce à la technologie Leap Motion.

2 Caractéristiques des utilisateurs

Il existe deux types d'utilisateurs possibles : les administrateurs des applications et les visiteurs de l'exposition du Musée des Beaux-Arts de Tours.

2.1 Administrateur du système

Les administrateurs auront la main sur les différentes applications, permettant de configurer ou reconfigurer celles-ci à tout moment. Parmi ceux-ci il y aura nous, les développeurs des applications mais également des responsables du musée ou encore des historiens. Néanmoins, il est évident que lorsque les portes du musée sont ouvertes les administrateurs n'ont plus accès au système sauf en cas de problèmes techniques ou urgence absolue.

2.2 Visiteur de l'exposition

La difficulté vient du fait que ceux-ci sont de tout âge, de tout sexe et de tout horizon. De plus dans la logique où n'importe qui pourrait être présent cela inclut également les personnes victimes de handicap. Parmi ce lot de personne, certains connaîtront déjà la ou les technologies alors que d'autres les découvriront totalement sur place. Dans tous les cas, ce ne seront que des utilisateurs ponctuels. En effet une fois qu'un utilisateur a utilisé notre tutoriel, il n'a plus d'intérêt d'y retourner et doit maintenant se diriger vers les applications principales de l'exposition du musée. Dans le cas de la découverte du Leap Motion, il n'y a pas vraiment de limites : les utilisateurs pourront toujours revenir s'il le souhaitent.

3 Fonctionnalité du système

3.1 Application de médiation Kinect

L'utilisateur pourra réaliser le tutoriel Kinect de l'application, qui consiste en une succession d'étapes lui permettant de se rendre compte des possibilités offertes et de comment les prendre en mains. Au terme de celui-ci, l'utilisateur sera en mesure de :

- Contrôler le curseur de la souris à distance
- Attraper, déplacer et tourner une image
- Agrandir et réduire une image

Ces fonctionnalités peuvent être retranscrites à travers le diagramme de cas d'utilisation suivant :

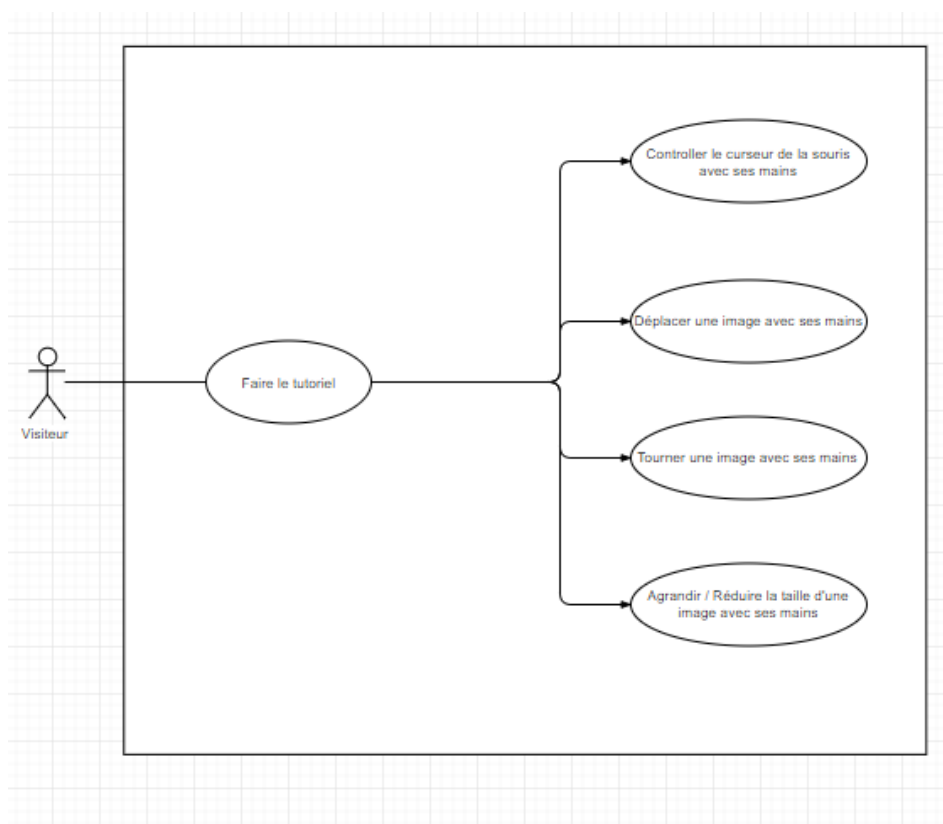


Figure 1 – Application de médiation Kinect - Diagramme des cas d'utilisation

3.2 Application de découverte Leap Motion

Le visiteur pourra utiliser cette application comme bon lui semble sans contrainte de scénarisation. Il aura le choix entre deux modules : dessin et manipulation d'image. Le temps de développement étant relativement assez court, le module de manipulation d'image sera plutôt en option si le temps nous le permet.

Dans le module de dessin, l'utilisateur sera face à une toile blanche. Il aura à sa disposition plusieurs outils pour dessiner :

- Une palette prédéfinie des couleurs de base
- Un mode « crayon » pour dessiner

- Un mode « gomme » pour effacer
- Plusieurs tailles de traits pour dessiner ou effacer
- Possibilité d’effacer complètement un dessin

Au sein du module de manipulation d’image, l’utilisateur pourra réaliser les fonctions les plus basiques en matière de traitement d’image :

- Déplacer
- Tourner et retourner
- Agrandir et réduire

Les fonctionnalités de ces deux modules peuvent être retranscrites à travers le diagramme de cas d’utilisation suivant :

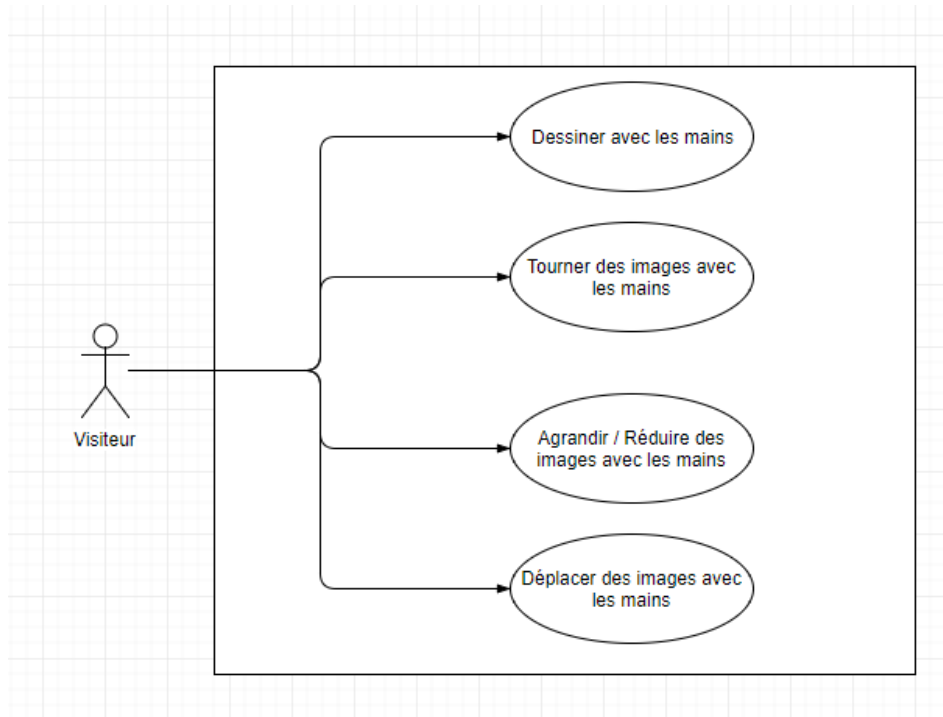


Figure 2 – Application de découverte Leap Motion - Diagramme des cas d’utilisation

4 Structure générale du système

Nous allons montrer les interactions entre les visiteurs et les systèmes dans le cadre d’un scénario de base, sans soucis techniques ou autre.

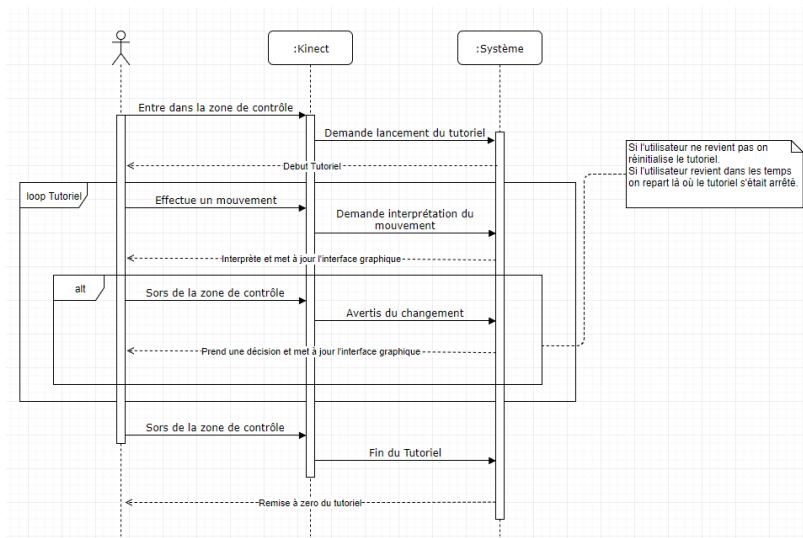


Figure 3 – Application de médiation Kinect - Diagramme de Séquence

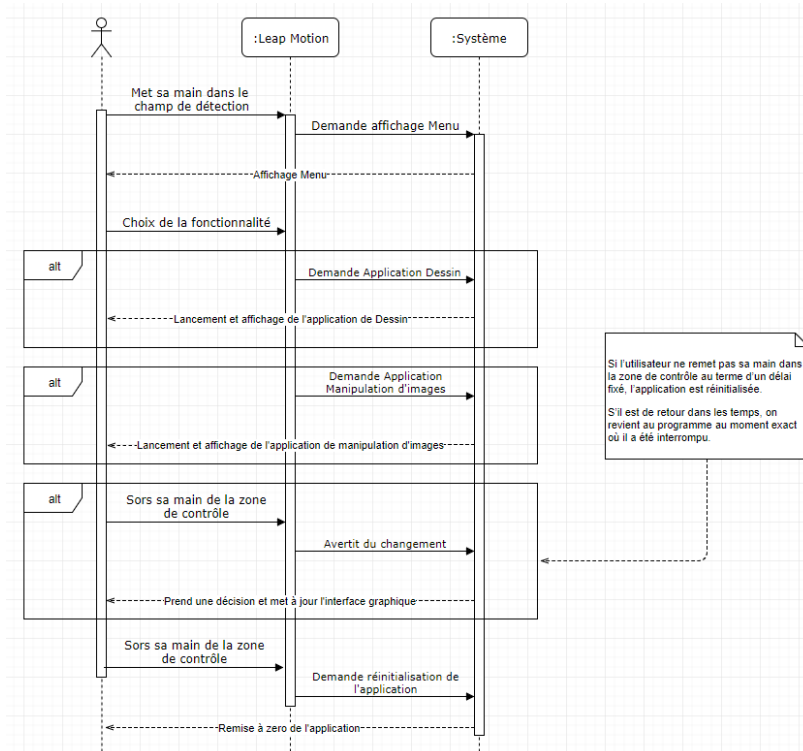


Figure 4 – Application de découverte Leap Motion - Diagramme de Séquence

3

État de l'art

Nous allons détailler à travers cette section les différentes applications existantes pour les technologies Kinect et Leap Motion. Nous verrons l'intérêt de ces applications dans le monde des musées et pour notre projet.

1 Kinect

Au moment où je rédige ces lignes, Kinect est une technologie "morte" ou obsolète. En effet, Microsoft a annoncé l'arrêt de la production de sa caméra en octobre 2017 après presque 7 ans de commercialisation puisque l'accessoire du géant américain est sorti le 4 novembre 2010 aux États-Unis et le 10 novembre 2010 en Europe. Néanmoins, nous allons tout de même étudier comment les musées ont-ils pu se servir de cet outil pendant cette période.

1.1 Kinect dans les musées du monde

Marioneta

En 2015, le Children's Museum de Pittsburgh (USA) a mis en place une installation utilisant la Microsoft Kinect pour permettre aux visiteurs d'incarner une collection de poupées marionnettes antiques dans un environnement virtuel. L'idée était de créer une expérience où les éléments du monde réagissent aux actions des utilisateurs à travers ces marionnettes. On incarne donc la marionnette de notre choix parmi les modèles proposés et nous pouvons interagir avec les éléments du décor affichés à l'écran. Par exemple, si nous sommes dans le décor d'automne alors il est possible avec notre poupée « d'exploser » une citrouille en mettant un coup de pied dedans. Kinect nous permet ici de pouvoir manipuler des marionnettes basées sur une collection donnée au Musée par Margo Lovelace dont la plupart sont, bien évidemment, trop délicates pour être jouées directement.

Nous pouvons aussi remarquer sur l'image ci-dessous un aspect intéressant mis en place par le musée. En effet, celui-ci a délimité sur le sol des zones de couleurs violettes et bleues associées à des empreintes de pas. Le principe ici est très simple, la zone bleue à l'arrière est la zone permettant de prendre le contrôle et de jouer avec les poupées. La zone violette, quant à elle, permet de changer de modèle de poupée. En effet, lorsque l'utilisateur avance dans la zone violette l'application détecte le changement de taille de l'utilisateur et lance en conséquence le

menu de sélection des poupées. Nous pourrions très bien imaginer faire la même chose dans le cas où nous aurions à notre disposition plusieurs statues ou tableaux différents.

À noter également que dans l'application les squelettes Kinect des utilisateurs sont constamment affichés en bas à droite de l'écran. Les utilisateurs ont donc toujours un œil sur leurs mouvements dans l'application.



Figure 1 – *Exposition Marioneta - Children's Museum of Pittsburgh*

Une vidéo de présentation de l'application est disponible ici :

[Marioneta: 3 min promo video](#)

Ptérosaures, Vols à l'ère des dinosaures

Dans le même esprit que pour Marioneta, l'American Museum of Natural History de New York (USA) avait mis en place en 2014 une application permettant de se mettre dans la peau des ptérosaures, ces reptiles volants géants qui régnaient dans le ciel à l'époque des dinosaures. Nous avons donc ici un simulateur de vol par reconnaissance de mouvement grâce à Kinect. [\[WWW0\]](#)

Deux types de jeux sont même disponibles :

- Attraper des insectes dans une forêt vierge.
- Attraper des poissons en mer.

Les deux jeux proposés ne sont pas anodins, car ils permettent de replacer la créature dans son environnement natal et de se rendre compte de son mode de vie et de son quotidien.

L'application Kinect ne gère donc ici que le haut du corps de l'utilisateur. Celui-ci contrôle les ailes de la créature avec les bras et les mouvements et rotations du haut du corps permettent d'orienter la créature. Par exemple, si l'utilisateur se penche en avant alors le ptérosaure va faire un pic vers le sol. À l'inverse s'il se penche vers l'arrière le ptérosaure va se diriger vers le ciel. Évidemment, battre des ailes (et donc ici des bras) reste la meilleure option pour regagner de l'altitude. Par ailleurs, l'application gère ici la vitesse d'exécution de l'utilisateur. En effet, la rapidité avec laquelle l'utilisateur bat ses bras détermine à quelle vitesse l'oiseau grimpe. Cette application a été réalisée sous Unity.

Une idée intéressante ici est que l'application attend un mouvement précis de la part de l'utilisateur pour démarrer. En effet, tant que l'utilisateur ne fait pas un mouvement de battement d'ailes le ptérosaure ne s'envole pas. L'idée est bonne, car elle est extrêmement intuitive et permet de s'identifier instantanément à la créature que nous contrôlons.

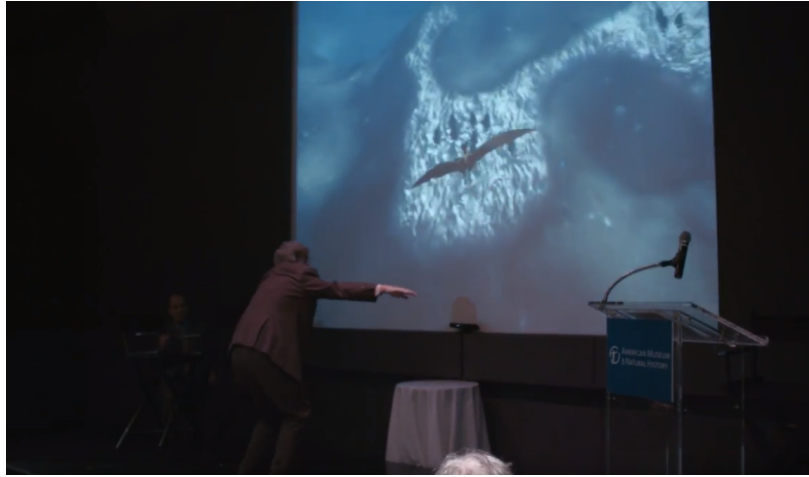


Figure 2 – Pterosauresy - American Museum of Natural History of New-York

Une vidéo de présentation de l'application est disponible ici :

[Fly Like a Pterosaur in Kinect-Powered Simulator](#)

ArtLens 2.0 - Studio Play

Le Museum of Art de Cleveland (USA) est l'un des musées qui s'est le plus modernisés. En effet, celui-ci a mis en place ArtLens, une plateforme AR qui utilise un logiciel de reconnaissance d'image pour reconnaître une sélection d'œuvres en deux dimensions. L'application fournit ensuite des détails sur la conservation et des interprétations de l'œuvre. L'œuvre d'art peut être scannée à une distance de 50 pieds (environ 15 mètres), ce qui permet aux visiteurs d'examiner les détails d'une œuvre avant même de s'en approcher. Néanmoins, le musée de Cleveland a voulu aller encore plus loin et s'est lui aussi laissé tenté par la technologie Kinect. Pour cela, il a créé Studio Play : un jeu immersif pensé pour l'art. Studio Play est une suite d'expériences qui permet aux visiteurs de se « connecter » avec les œuvres d'art du musée, notamment grâce à la caméra de Microsoft. Le musée a travaillé avec Design I/O, qui crée des installations interactives, afin de personnaliser les logiciels qui utilisent les caméras 3D Kinect de manière innovante pour améliorer l'expérience des visiteurs.

Ainsi, les utilisateurs peuvent faire une très grande série d'actions comme créer leur propre œuvre d'art en utilisant leurs propres mains, mais également manipuler l'argile dans un espace 3D ou encore projeter de la peinture virtuelle sur des toiles numériques 4K et ainsi revisiter de grands classiques de la peinture.

Toutes ses applications sont extrêmement ludiques et visiblement intuitives. Les visiteurs apprennent à se servir des applications en découvrant ce qu'il se passe lorsqu'ils font un mouvement. Cet aspect a le bénéfice de laisser intact le bonheur de découvrir par soi-même une technologie.



Figure 3 – *Dessin sur toile blanche* - Museum of Art of Cleveland

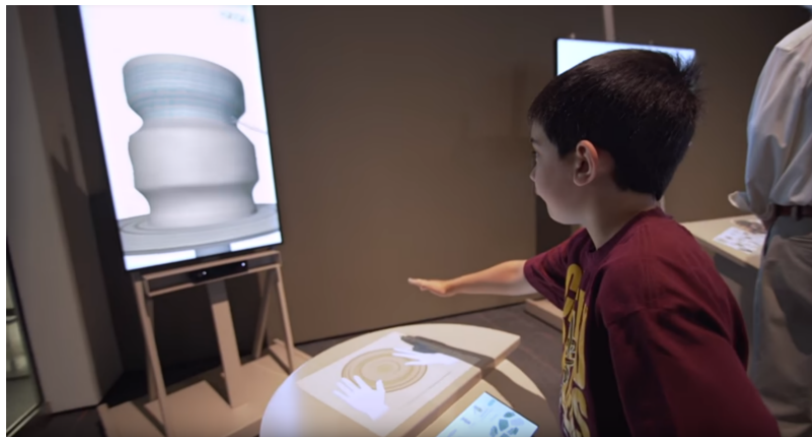


Figure 4 – *Manipulation de poteries d'argile* - Museum of Art of Cleveland



Figure 5 – *Projection de peinture sur toile* - Museum of Art of Cleveland

Nous pouvons découvrir l'ensemble des possibilités offertes par cette exposition dans cette vidéo :

[Studio Play | The Cleveland Museum of Art](#)

Digital Interlude

Lors de la Nuit des Musées 2012, le Musée Petiet de Limoux (France) a mis en place « Digital Interlude », une installation interactive permettant aux utilisateurs d'interagir avec la collection

du musée. À l'aide de la caméra Kinect et d'une diffusion sur écran géant, les utilisateurs manipulent le son et les visuels de l'installation en interaction avec la collection du musée.

L'idée est la suivante : le spectateur entre dans le champ du capteur (entre les signaux jaune et rouges) provoquant ainsi l'apparition d'un hologramme sur l'écran. L'expérience de chaque utilisateur varie d'un utilisateur à l'autre, car les sons et les visuels de l'installation changent avec le mouvement de l'utilisateur et sa proximité avec le capteur.

Parmi les exemples trouvés pour rédiger cet état de l'art, le musée de Limoux fut le seul à utiliser les capacités sonores du capteur Kinect. En effet, la caméra est capable de gérer la reconnaissance vocale. Ce qui pourrait être très utile notamment pour les personnes souffrant de handicap corporelles.



Figure 6 – *Digital Interlud* - Musée Petiet de Limoux

Une petite vidéo très succincte de cette fonctionnalité :

[Museum Night - Kinect Installation](#)

Map Interaction

Des étudiants de l'université de Münster en Allemagne ont mis au point un module type « Google Map » de cartographie utilisable entièrement avec Kinect pour le Museum for Communication de Francfort-sur-le-Main (Allemagne). Le module n'est pas très impressionnant en soi, mais il se rapproche des applications développées pour notre projet. Nous pouvons donc y trouver des boutons très larges permettant une utilisation simple et efficace, mais également un retour vidéo très propre en bas à gauche.

Le module reprend les fonctionnalités basiques de cartographie : zoom, dézoom, vu satellite, etc.



Figure 7 – Map Interaction - Museum for Communication of Frankfurt am Main

Une petite vidéo de cette fonctionnalité :

Gestural Interaction With Maps - Microsoft Kinect and esri technology

DinoStomp

Ce que nous a proposé le Museum of Science and History de Fort Worth (USA) en novembre 2016 est assez impressionnant. En effet, l'application DinoStomp est une expérience dans laquelle nous pouvons par groupe allant jusqu'à dix personnes maximum interagir avec des dinosaures de toutes les époques mésozoïques dans un paysage 3D imaginaire. Pour réaliser cela, le matériel est composé d'un mur vidéo de 8' de haut et de 20' de large, mais surtout de trois caméras de reconnaissance de mouvement Microsoft Kinect. [WWW0]

L'idée est que les dinosaures de la scène 3D suivent et interagissent avec les utilisateurs lorsqu'ils entrent dans la portée des capteurs, rugissant et bondissant selon le mouvement reconnu. La scène est entièrement animée et possède des éléments d'IA qui la rendent plus dynamique.

L'interaction principale dans l'exposition est accomplie par le suivi des personnes qui marchent devant le mur avec des capteurs Kinect. Apparemment, le suivi précis du corps était compliqué à gérer, compte tenu des conditions d'éclairage de la galerie et de la courbe du mur. C'est pour cela qu'en fin de compte, trois caméras Kinect sont utilisées et couplées pour résoudre ce problème.

L'application met en scène des dinosaures qui ont chacun leur propre personnalité lorsqu'ils réagissent et jouent avec le public. Les événements dits « épiques » se produisent lorsque certains personnages de l'application apparaissent à des moments différents selon le nombre de participants, comme le Tyrannosaure ou le Brachiosaure par exemple.

Les plus petits dinosaures, également connus sous le nom de vélociraptors et d'ankylosaures, sont programmés pour être mis en mouvement par l'un des dispositifs Kinect. Cette interaction permet aux participants d'avoir leurs mouvements mimés par des raptors en 3D et rend la scène plus participative et amusante. Ce genre d'interaction permet aussi de diminuer le « facteur de peur » pour les plus jeunes, car ils sont en mesure de contrôler les dinosaures dans la scène avec leur mouvement.

Il est ici très intéressant de voir qu'il est possible de coupler plusieurs caméras Kinect pour agrandir le champ d'action. De plus, les caméras sont placées très en hauteur avec un angle vers bas assez conséquent. Toutefois, je n'ai pas réussi à trouver plus d'informations à ce sujet.



Figure 8 – DinoStomp - Museum of Science and History of Fort Worth



Figure 9 – DinoStomp - Museum of Science and History of Fort Worth

La vidéo officielle de présentation du projet :

[DinoStomp Interactive Video Wall at the Fort Worth Museum of Science and History](#)

ARSandbox

ARSandbox est un projet très ambitieux combinant des applications de visualisations avec une exposition pratiques sur les bacs à sable pour enseigner les concepts des sciences de la Terre. Ce sandbox en réalité augmentée permet aux utilisateurs de créer des modèles de topographie en façonnant du sable réel, qui est ensuite augmenté en temps réel par une carte de couleur, des courbes de niveau topographiques et de l'eau simulée. Le système enseigne des concepts géographiques, géologiques et hydrologiques tels que la lecture d'une carte topographique, la signification des courbes de niveau, des bassins versants, des bassins versants, des digues, etc.

Le produit ainsi réalisé a été utilisé dans de très nombreux musées ou organismes à travers le monde, notamment en France. Les organismes en France ayant utilisé ce système sont :

- L'exploradome de Paris
- La Maison des Minéraux de Saint-Herriot
- Science Animation Midi Pyrénées de Toulouse
- L'Université Géosciences de Montpellier
- Le CEREGE de Marseille

Dans l'idée donc la caméra Kinect est postée au-dessus du bac à sable et capture en temps réels le sable présent dans le bac. La capacité de la caméra à gérer la profondeur est parfaitement utilisée ici pour détecter les creux dans le sable indiquant par exemple l'intention de créer un fleuve ou une rivière ou à l'inverse une bosse de sable pour créer une montagne ou même un volcan. L'application traite ensuite les informations et renvoie grâce au vidéoprojecteur la carte mise à jour en temps réel.

Cerise sur le gâteau, le projet est libre, distribué sous licence GNU et donc téléchargeable à l'adresse disponible en Webographie. L'ensemble est parfaitement documenté et prêt à être utilisé après configuration également documenté.

[WWW0]

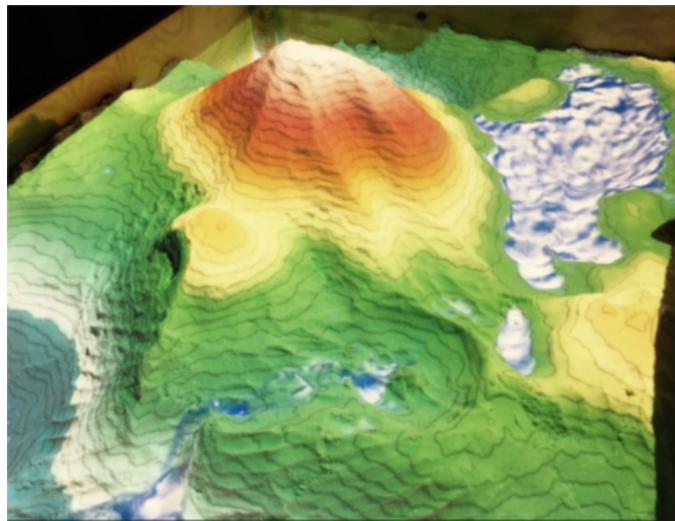


Figure 10 – ARSandbox - Sacramento City College, Los Rios, CA

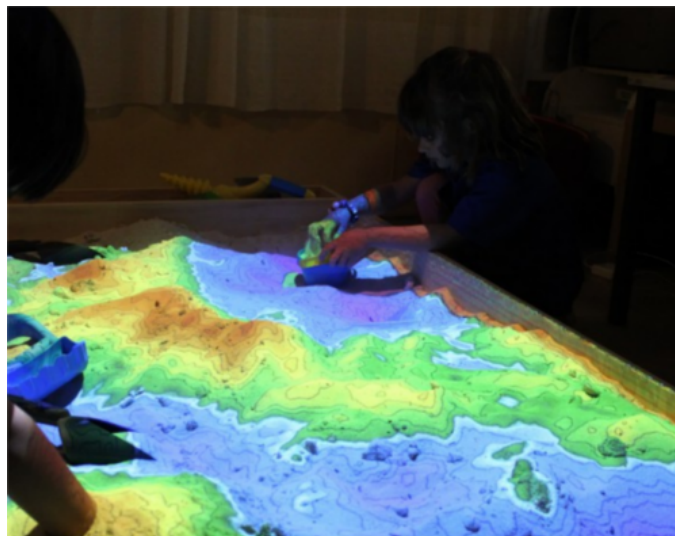


Figure 11 – ARSandbox - Bold Park Community School - Perth, Australia

Une vidéo complète d'utilisation de l'application est disponible ici :
[Augmented Reality Sandbox with Real-Time Water Flow Simulation](#)

1.2 Développer avec Kinect

1.2.1 Aider les musées, aider l'Histoire

Afin de connaître l'origine de petits trous sur le crâne d'un « T. rex » conservé au Museum Field de Chicago (USA), des chercheurs ont demandé l'aide du MIT afin de mettre au point un système de scan 3D, car évidemment, le squelette est fragile, difficile à manipuler vu sa taille et son accès est limité. Ce système utilise la caméra Kinect. [WWW0]

Ce qui est intrigant ici c'est que le capteur de Microsoft n'a pas enregistré des mouvements d'objets animés. En effet, à l'inverse, le Kinect a été déplacé tout autour de l'objet fixe (le crâne du T. rex) à partir d'un système de harnais ou d'un système de trépied.

La technique complète de scan est disponible dans l'article disponible en Webographie : [Das2017:2]

Ce qui est intéressant à travers cet article également c'est qu'il détaille l'ensemble des configurations de la caméra Kinect en matière de détection. La caméra de Microsoft possède donc une caméra 1080p fonctionnant à 30 Hz. Le capteur de profondeur est lui sur une résolution de 512x424. Les distances de profondeur minimales et maximales sont respectivement de 0,5 m et 4,5 m. Enfin, le champ de vision horizontal est de 70 ° et le champ de vision vertical est de 60 °.

1.2.2 Aide au développement

Afin d'aider sa communauté au développement d'application utilisant Kinect, Microsoft a publié en open source des échantillons de code de son capteur de mouvements et de reconnaissance vocale. Les exemples de code publiés par Microsoft montrent comment utiliser les fonctionnalités de Kinect, notamment l'audio, les interactions de base, la profondeur, l'infrarouge, le suivi du visage et des gestes, le suivi du squelette et bien plus. Le tableau ci-dessous illustre sur quel langage peut être utilisé chaque exemple de code déposé par Microsoft.

Sample	C#	C++	VB	WPF	DirectX	Additional information
Audio Basics	Yes	Yes	Yes	Yes	Yes	Available in 1.6.0
Audio Capture Raw	No	Yes	No	No	No	Available in 1.6.0
Audio Explorer	No	Yes	No	No	Yes	Available in 1.6.0
Basic Interactions	Yes	No	No	No	No	Available in 1.6.0
Color Basics	Yes	Yes	Yes	Yes	Yes	Available in 1.6.0
Depth Basics	Yes	Yes	Yes	Yes	Yes	Available in 1.6.0
Depth	No	Yes	No	No	Yes	Available in 1.6.0
Depth with Color	No	Yes	No	No	Yes	Available in 1.6.0
Face Tracking	Yes	No	No	Yes	No	Available in 1.6.0, Requires Face Tracking
Face Tracking Basics	Yes	No	No	Yes	No	Available in 1.6.0, Requires Face Tracking
Face Tracking Visualization	No	Yes	No	No	Yes	Available in 1.6.0, Requires Face Tracking
Green Screen	Yes	Yes	No	Yes	Yes	Available in 1.6.0
Infrared Basics	Yes	Yes	No	Yes	Yes	Available in 1.6.0
Kinect Explorer	Yes	No	No	Yes	No	Available in 1.6.0
Shape Game	Yes	No	No	Yes	No	Available in 1.6.0
Skeletal Viewer	No	Yes	No	No	Yes	Available in 1.6.0
Skeleton Basics	Yes	Yes	Yes	Yes	Yes	Available in 1.6.0
Slideshow Gestures	Yes	No	No	Yes	No	Available in 1.6.0
Speech Basics	Yes	Yes	Yes	Yes	Yes	Available in 1.6.0
Tic Tac Toe	Yes	No	No	Yes	No	Available in 1.6.0
WPF D3D Interop	Yes	Yes	No	Yes	Yes	Available in 1.6.0
XNA Basics	Yes	No	No	Yes	No	Available in 1.6.0, Requires XNA

Figure 12 – Correspondance des langages face aux exemples de code de Kinect

1.2.3 La gestuelle avec Kinect

Cette section est détaillée dans le rapport de M. Arnaud Talon. Il existe également un article très intéressant qui décrit notamment les niveaux d'intuitivités et de fatigues qu'engendrent les différentes gestuelles propres à Kinect. Malheureusement, cet article est payant et je n'ai pas le temps de le traiter dans le peu de laps de temps où la lecture était gratuite. [0]

1.2.4 Alternative et futur

Comme énoncé dès le départ, la technologie Kinect est désormais arrivée à son terme. Nous avons tout de même vu qu'elle aura contribué à de très nombreux projets sous de multiples formes. Enfin, si Microsoft a décidé de se pencher exclusivement sur le développement de son casque de réalité augmentée HoloLens, nous pouvons nous pencher quelques instants sur les alternatives propres à la reconnaissance de forme par caméra.

ASUS Xtion

La caméra ASUS Xtion est également une caméra de reconnaissance de mouvements à l'instar de Kinect. Les différences entre les deux sont résumées dans le tableau suivant :

Device	Pros	Cons
Microsoft Kinect	<ul style="list-style-type: none"> High quality of device drivers Stable work with various hardware models Has motor that can be controlled remotely by IPI Recorder application: this makes device positioning more convenient 	<ul style="list-style-type: none"> Bigger size (12" x 3" x 2.5" against 7" x 2" x 1.5") Higher weight (3.0 lb against 0.5 lb) Require ACDC power supply Higher interference with another Kinect sensor in "Dual depth sensor" configuration Lower RGB image quality in comparison with MS Kinect
ASUS Xtion / PrimeSense Carmine	<ul style="list-style-type: none"> More compact (7" x 2" x 1.5" against 12" x 3" x 2.5") Lighter weight (0.5 lb against 3.0 lb) Does not require power supply except USB Lower interference with another ASUS Xtion / PrimeSense Carmine sensor in "Dual depth sensor" configuration Better RGB image quality 	<ul style="list-style-type: none"> Less popular device Lower drivers quality Does not work with some USB controllers (especially USB 3.0) No motor, allow only manual positioning

Figure 13 – Comparaison Microsoft Kinect Vs ASUS Xtion

Structure

La caméra Structure est une caméra qui promet un balayage 3D rapide d'objets et de personnes, des cartes 3D d'espaces intérieurs ou encore des expériences de réalités augmentées où le fantastique devient impossible à distinguer de la réalité. Elle a l'avantage de pouvoir, grâce à un appareil spécial, de se connecter à un casque de réalité augmentée.

Cette caméra possède son site web :

structure.io

2 Leap Motion

De la même manière que pour la caméra Kinect, le dispositif Leap Motion gère la reconnaissance de mouvements à l'exception près qu'il se concentre sur les gestes de la main. Évidemment, s'il se concentre uniquement sur cet aspect c'est qu'il est également extrêmement précis.

2.1 Leap Motion dans les musées du monde

Sur le site officiel Leap Motion, il n'y a qu'un seul post avec le tag « Museum ». Cela concerne la façon de créer sa propre installation artistique avec Leap Motion. Bien que certainement intéressant cet aspect n'est pas très utile pour notre projet.

Effectorium

La Maison Mendelssohn de la ville allemande de Leipzig a rouvert ses portes le 3 février 2014. L'exposition informe les visiteurs sur la vie et les œuvres du musicien et compositeur Felix Mendelssohn Bartholdy, qui a vécu et travaillé à Leipzig pendant de nombreuses années.

Le musée offre une variété d'expositions interactives qui permettent aux visiteurs d'explorer les œuvres de Mendelssohn Bartholdy. Un de ses points forts est l'installation d'une salle interactive : « Effectorium ».

Dans le « Effectorium », les visiteurs peuvent utiliser un bâton, la technologie Leap Motion et un écran tactile pour « diriger » un orchestre numérique. Le Leap Motion enregistre les mouvements de bâtons et utilise les actions pour contrôler la musique. Les « chefs d'orchestre » peuvent également influencer différents facteurs tels que le volume pour l'ensemble de l'orchestre et pour chaque groupe d'orchestre ou de chœur. Ils peuvent également contrôler l'éclairage de la pièce, l'éclairage LED changeant de couleur et d'intensité en fonction de leurs mouvements.

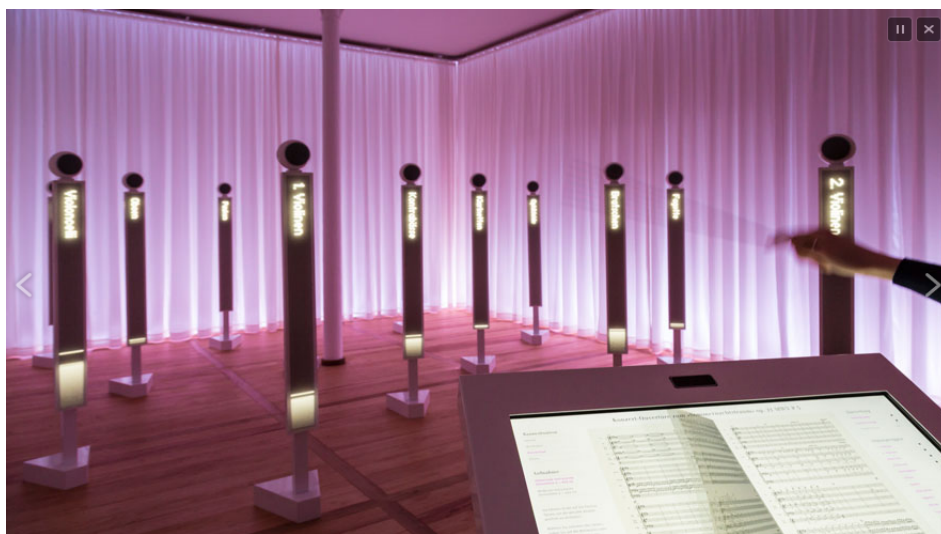


Figure 14 – Effectorium - Mendelssohn House

Une vidéo de présentation est disponible ici :

[Mendelssohn Effektorium - Virtual orchestra for Mendelssohn-Bartholdy Museum Leipzig](#)

Visite virtuelle

Cette application, en revanche, est l'illustration parfaite de notre projet et de nos applications Leap Motion. L'objectif de Gabriel Acoca est de réaliser des visites virtuelles de bâtiments ou d'objets (comme par exemple des voitures) le tout entièrement utilisable et déplaçable grâce au Leap Motion.

Ce genre d'outil s'avère extrêmement utile dans plusieurs cas. On pensera tout d'abord aux personnes à mobilité réduite qui pourront découvrir ces lieux sans trop faire d'effort, mais nous pourrions également imaginer simuler des pièces secrètes ou interdites au public comme c'est très souvent le cas dans des châteaux ou des anciennes maisons d'artistes.

Nous avons ici un exemple en vidéo :

[Visite virtuelle pilotée par Leap Motion](#)

Mais nous avons encore mieux avec des démos complètes jouables, nous avons ici l'Atelier du peintre Antoine Bourdelle.

[Visite virtuelle - Atelier Antoine Bourdelle](#)

Nous avons ici une galerie d'art quelconque.

[Visite virtuelle - Galerie d'arts](#)

Et nous avons ici le Musée Fournaise au complet.

[Visite virtuelle - Maison Fournaise](#)



Figure 15 – Visite virtuelle - Atelier Antoine Bourdelle

2.2 Développer avec Leap Motion

2.2.1 Vue d'ensemble

Le système Leap Motion reconnaît et suit les mains, les doigts et les outils semblables à des doigts comme un bâton ou un crayon par exemple. L'appareil fonctionne avec une très haute précision et un taux de trames de suivi et rapporte des positions, des gestes et des mouvements discrets.

Le contrôleur Leap Motion utilise des capteurs optiques et de la lumière infrarouge. Les capteurs sont dirigés le long de l'axe y — vers le haut lorsque le contrôleur est dans sa position de fonctionnement standard — et ont un champ de vision d'environ 150 degrés. La portée effective du Leap Motion s'étend d'environ 25 à 600 millimètres au-dessus de l'appareil.

Système de coordonnées

Leap Motion utilise un système de coordonnées cartésiennes pour droitier. L'origine est centrée au sommet du Leap Motion Controller. Les axes x et z se trouvent dans le plan horizontal, l'axe des x étant parallèle au grand côté de l'appareil. L'axe y est vertical, avec des valeurs positives augmentant vers le haut (contrairement à l'orientation vers le bas de la plupart des systèmes de coordonnées graphiques). L'axe z a des valeurs positives croissant vers l'utilisateur.

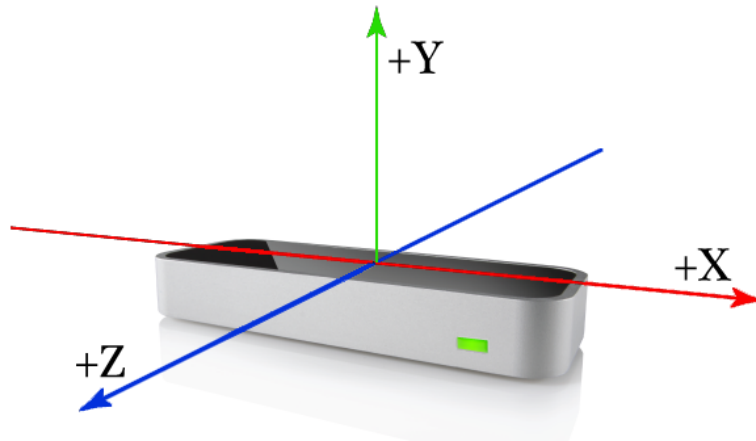


Figure 16 – Leap Motion - Système de coordonnées

Suivi de mouvements

Le modèle pour la main fournit des informations sur l'identité, la position et d'autres caractéristiques d'une main détectée notamment le bras auquel la main est attachée et les listes des doigts associés à la main.

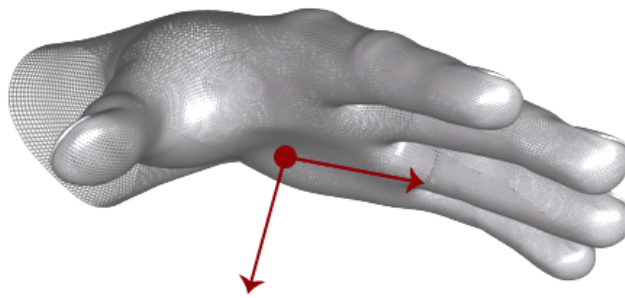


Figure 17 – Leap Motion - Reconnaissance d'une main

Le Leap Motion utilise un modèle interne d'une main humaine pour fournir un suivi prédictif même lorsque des parties d'une main ne sont pas visibles. Le modèle fournit des positions pour les cinq doigts, même lorsque l'ensemble de la main n'est pas visible.

Le contrôleur Leap Motion fournit des informations également sur chaque doigt d'une main. Si tout ou partie d'un doigt n'est pas visible, les caractéristiques du doigt sont estimées à partir des observations récentes et du modèle anatomique de la main. Les doigts sont identifiés par le nom du type, c'est-à-dire le pouce, l'index, le milieu, l'anneau et le petit doigt.

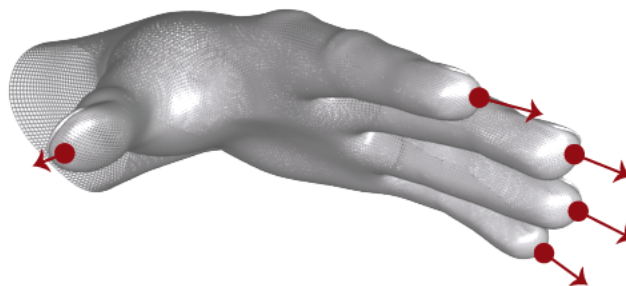


Figure 18 – Leap Motion - Reconnaissance des doigts

Enfin, Leap Motion est capable de détecter des objets pris en main par un utilisateur. Il faut

néanmoins pour cela que l'objet soit plus long, et plus fin qu'un doigt d'une main. Il vaut mieux également que l'objet soit cylindrique.

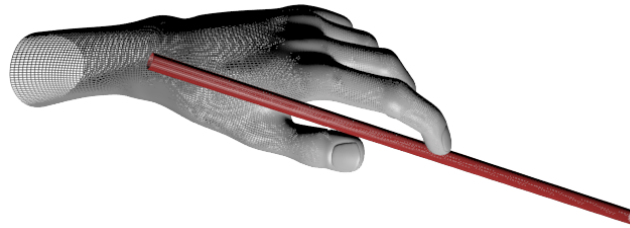


Figure 19 – *Leap Motion - Reconnaissance d'un objet*

En ce qui concerne les mouvements que peut analyser le capteur, nous pouvons trouver ci-dessous une liste des gestes de bases.

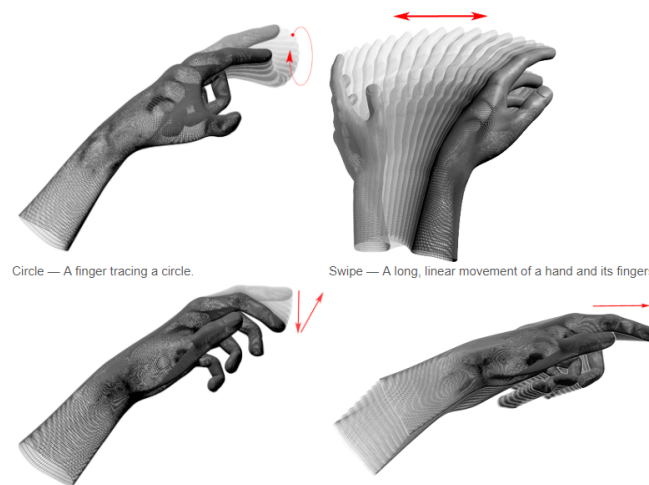


Figure 20 – *Leap Motion - Liste de mouvements possibles*

Nous pouvons d'ailleurs observer que le Leap Motion gère parfaitement l'appui d'un doigt vers l'avant simulant donc un clic de souris. Cet aspect nous sera très utile pour le développement de notre application de découverte.

2.2.2 Langages

Le développement sur le Leap Motion a fortement évolué au fil des années, notamment en le rendant beaucoup plus facile et accessible. Aujourd'hui, sur le site officiel Leap Motion une documentation complète, un guide d'installation et un guide du développeur sont disponibles pour l'ensemble des langages supportés par le dispositif. Parmi ces langages, on trouve :

- C++
- C#
- Unity
- Apple Objectif-C
- Java
- Python
- JavaScript
- Unreal Engine

2.2.3 Agatha

Agatha est un module permettant à la base de présenter des images de manières dynamiques et originales. Il s'avère en fin de compte que c'est exactement ce dont nous avons besoin pour notre module de manipulation d'image. En effet, Agatha permet de sélectionner, déplacer, arranger, pivoter et zoomer des images uniquement avec des gestes de la main et des mouvements de doigt. Ce module est entièrement gratuit et disponible sur le site officiel Leap Motion. Ce type d'outil pourra nous être utile afin de gagner du temps de développement ou sera finalement peut-être utilisé en tant que tel si les délais ne nous permettent pas d'implémenter notre propre outil.

Une vidéo de présentation est disponible ici :

[Agatha - Image Presenter for Leap Motion - Teaser](#)

2.2.4 Architecture

Le logiciel Leap Motion fonctionne en tant que service (sous Windows) ou démon (sous Mac et Linux). Le logiciel se connecte au périphérique Leap Motion Controller à partir du port USB. Les applications compatibles Leap accèdent au service Leap Motion pour recevoir des données de suivi de mouvement. Le Leap Motion SDK fournit deux types d'API pour obtenir les données Leap Motion : une interface native et une interface WebSocket. Ces API permettent de créer des applications compatibles Leap Motion.

L'interface d'application native est fournie grâce à une bibliothèque chargée dynamiquement. Cette bibliothèque se connecte au service Leap Motion et fournit les données de suivi à l'application.

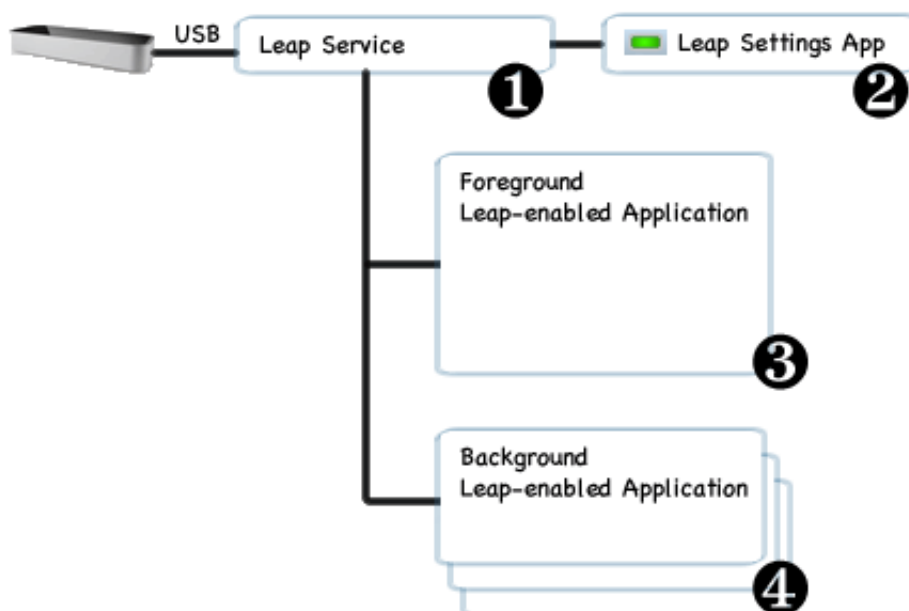


Figure 21 – Leap Motion - Architecture d'une application

4

Analyse et conception

1 Généralités

Semestre 9

Comme énoncé dès le départ, le développement de l'application de médiation pour Kinect avait déjà été commencé par M. Arnaud Talon. Mon rôle ici est de reprendre son travail, et de le mener à son terme. Cela inclut de reprendre chaque étape du tutoriel pour l'améliorer pour la rendre plus ergonomique, attrayante, intuitive et plus robuste. Cela implique également de rajouter de nouvelles fonctionnalités comme une nouvelle étape pour apprendre à un utilisateur à tourner une image ou encore ajouter des images animées ou des vidéos pour montrer concrètement le mouvement à réaliser. Toutefois, ces modifications ne viendront pas perturber la structure actuelle de l'application, qui a été pensée pour pouvoir accepter ces améliorations.

Semestre 10

Suite aux différents retards et changements qui sont intervenus sur le projet, il a été convenu d'un accord commun entre M. Venturini et moi-même que l'application de découverte Leap Motion soit mise de côté pour ne pas dire totalement abandonnée. De ce fait, l'intégralité du semestre 10 a été portée sur l'application de médiation Kinect et l'analyse sur l'application Leap Motion n'a pas bougé.

2 Diagrammes de Classes

2.1 Application de médiation Kinect

Semestre 9

Comme exposé au préalable, le diagramme de classe existant pour l'application de médiation Kinect ne devrait pas connaître de changement pour cette poursuite du projet. Il sera donc réutilisé.

Les six classes qui composent ce diagramme ne devraient pas changer. Les modifications à effectuer interviendront à l'intérieur de celle-ci sans avoir à modifier la structure du système. Nous pouvons faire une petite revue de celle-ci :

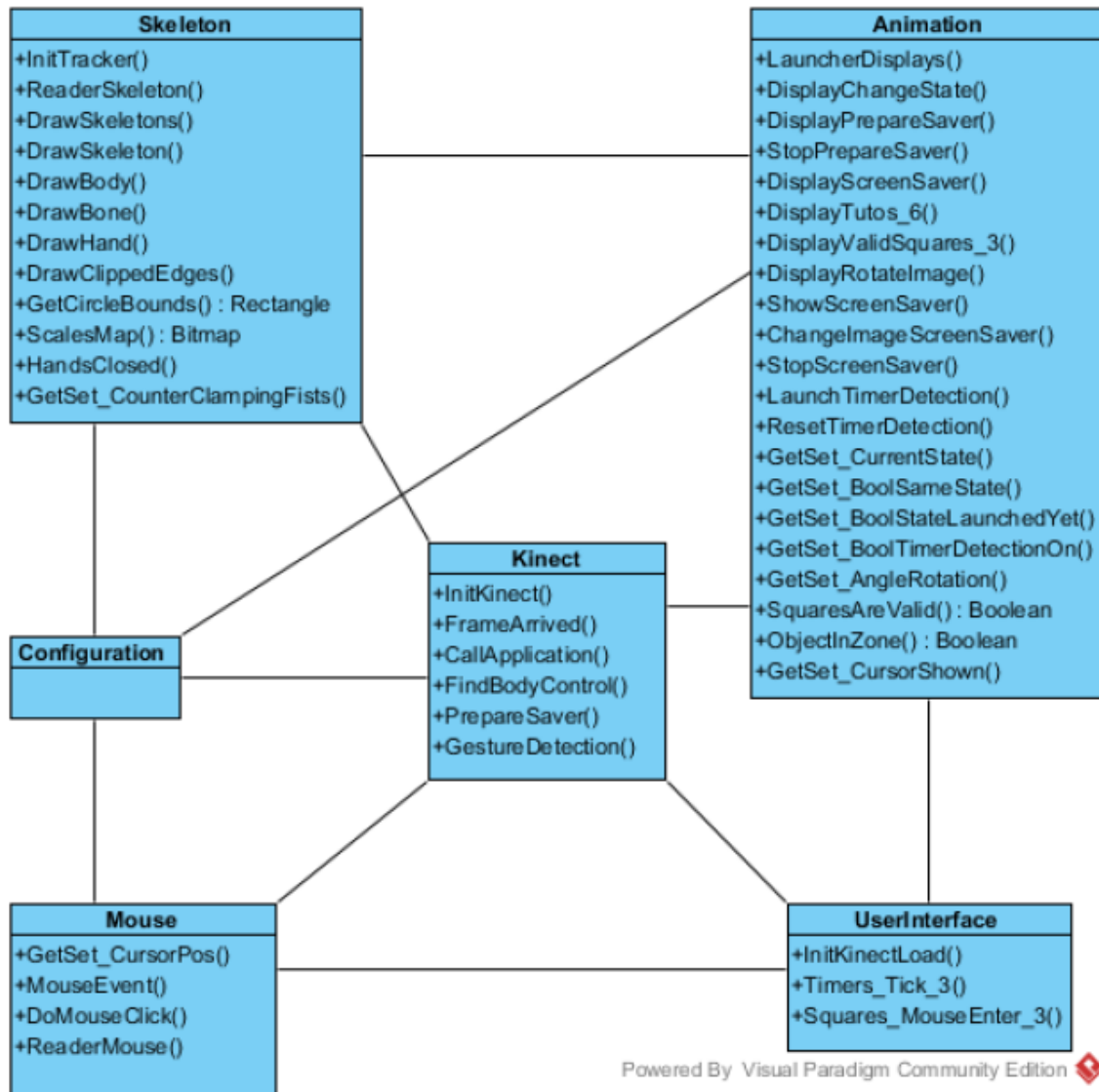


Figure 1 – Application de médiation Kinect - Diagramme de classes

- **UserInterface** : Gère l'interface graphique de l'application.
- **Kinect** : Fais le lien entre l'application et la caméra Kinect.
- **Mouse** : Gère les méthodes de contrôle du curseur de la souris.
- **Animation** : Gère les éléments graphiques et audio à effectuer selon les actions de l'utilisateur (créé pour éviter de surcharger la classe **UserInterface** ou **Kinect**).
- **Configuration** : Gère l'ensemble des paramètres de l'application, comme les liens vers les fichiers audio ou encore la sensibilité du curseur de la souris.

Pour obtenir une description complète de ces classes se référer au rapport de M. Arnaud Talon.

Semestre 10

Ci-dessous le diagramme de classe final de l'application de médiation Kinect avec les nouveaux éléments qui ont été ajoutés tout au long du semestre et de la mise en œuvre.

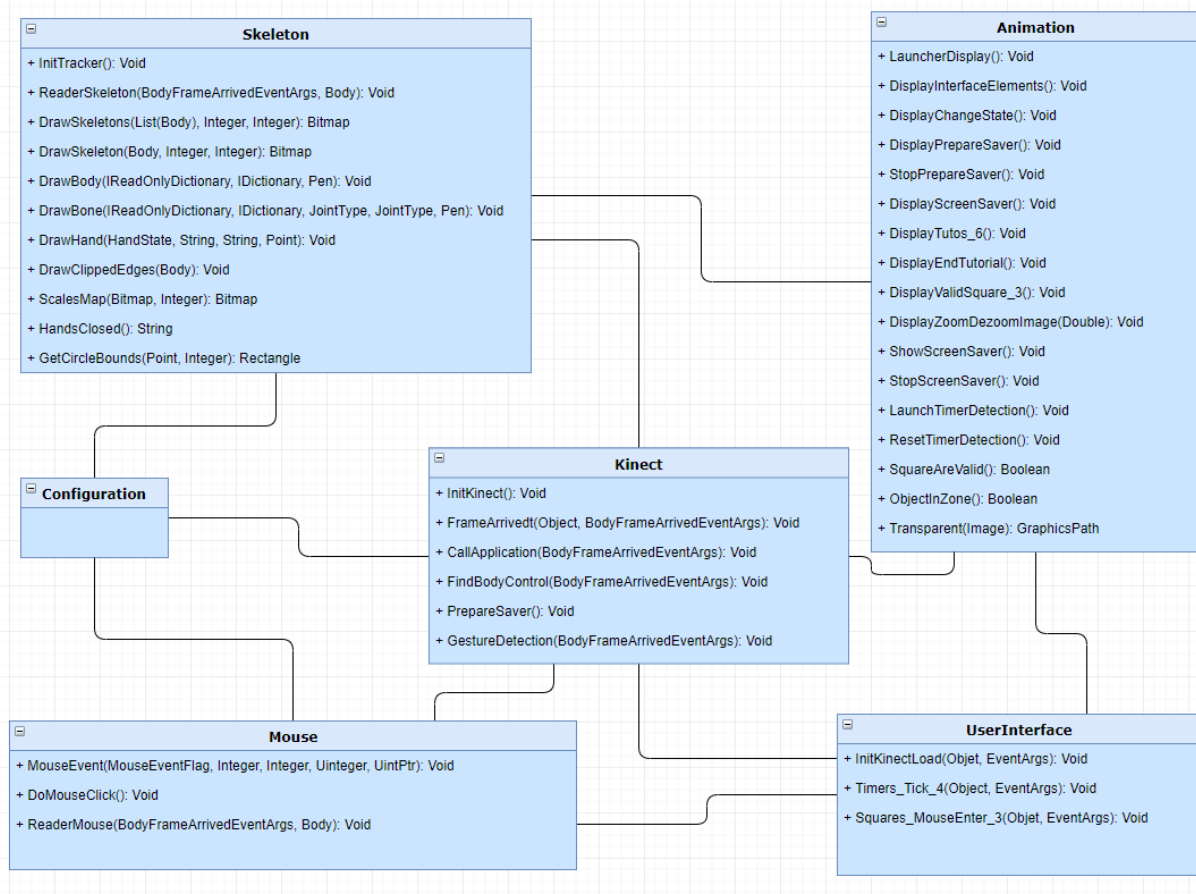


Figure 2 – Application de médiation Kinect - Diagramme de classes final

2.2 Application de découverte Leap Motion

Nous allons maintenant voir l'analyse et la conception de notre application de découverte Leap Motion. Nous pouvons dès le départ distinguer plusieurs modules se distinguant les uns des autres. Il faudra notamment gérer la reconnaissance des mains, des mouvements de celles-ci, mais également une classe pour dessiner et une autre pour la manipulation d'image.

Nous pouvons donc séparer notre application en deux parties clés :

- Une première partie pour la reconnaissance des formes et des mouvements
- Une seconde partie pour la gestion des interfaces et des fonctionnalités

Nous avons, malgré un temps sur la fin limité, pris le temps de commencer à faire un diagramme de classe. Ce diagramme est extrêmement discutable et sera mené à son terme lors des deux dernières semaines de décembre, qui coïncide avec les dates de remises de rapport et de soutenances.

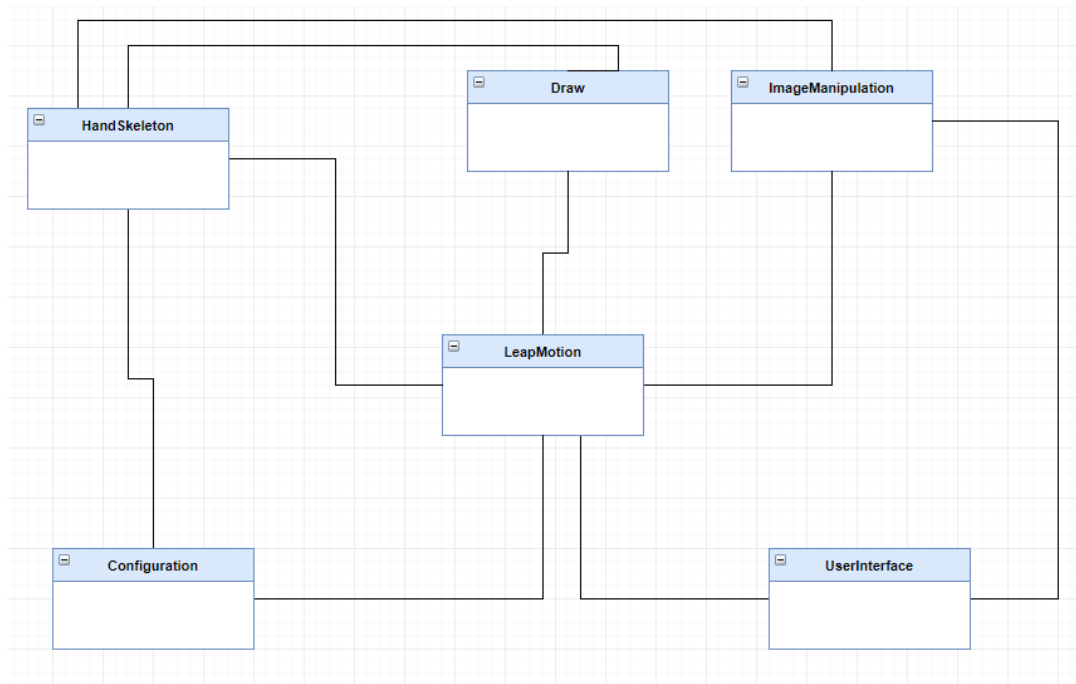


Figure 3 – Application de découverte Leap Motion - Diagramme de classes

2.2.1 UserInterface

On trouvera ici l'interface et les méthodes d'attente d'utilisateur. Elle aura donc comme rôle de gérer l'entrée et la sortie utilisateur, mais également de lancer la classe Leap Motion au lancement de l'application.

2.2.2 Draw

Classe correspondant au module de dessins.

2.2.3 ImageManipulation

Classe correspondant au module de manipulation d'image.

2.2.4 LeapMotion

Cette classe fait le lien entre l'application et le Leap Motion. Elle est lancée au début de l'application et ne s'arrête jamais. Elle gère les instructions à effectuer à chaque frame envoyée du Leap Motion.

2.2.5 HandSkeleton

Cette classe permettra la création du squelette de la main de l'utilisateur.

2.2.6 Configuration

Cette classe permettra de gérer la configuration de l'application. Dans notre cas, cela résulte simplement au chargement des images qui pourront être utilisées dans le module de manipulation d'images.

5

Mise en œuvre

Ce projet a connu quelques difficultés notamment pour la gestion du temps. En effet, la prise en main de l'existant et la réparation des erreurs sur l'application existante ont considérablement retardé le début de l'implémentation des nouvelles fonctionnalités. En conséquence, la deuxième application concernant la médiation pour le Leap Motion a tout simplement été abandonnée, d'un commun accord avec M. Venturini. D'autre part, comme prévu finalement de nombreux changements sont intervenus tout au long de l'avancée du projet notamment sur l'aspect graphique et les « épreuves » que devaient réaliser les utilisateurs.

1 Implémentation

1.1 Limites

1.1.1 Humaines

Les limites à prendre en compte sont principalement les mêmes que celles évoquées par Arnaud Talon dans son rapport.

L'application doit être assez courte pour qu'un maximum de personnes puisse la tester sans obtenir une interminable file d'attente, mais également suffisamment longue pour qu'une personne puisse comprendre efficacement les mouvements.

En conséquence, elle doit être la plus ergonomique possible et très simple à prendre en main. L'objectif, est qu'une personne comprenne rapidement le geste et comment le réaliser. Il faut aussi que ces mouvements soient accessibles à tous (dans la mesure du possible).

Enfin, il faut également que les gestes soient simples et rapides pour éviter de fatiguer les utilisateurs.

1.1.2 Techniques

Kinect envoie chaque seconde des dizaines de frames qu'il faut interpréter. Le système est donc régulièrement soumis à des risques de bugs. De nombreux tests et conditions ont été réalisés pour pouvoir gérer toute situation délicate ou imprévue dans la limite du possible.

De plus, la position de la caméra n'est pas la même pour l'exposition au musée qu'elle ne l'était pour le développement de l'existant. Nous avons donc organisé avec M. Venturini un espace de travail quasiment identique à celui qui sera au musée pour le développement afin d'être le plus proche possible des conditions de mise en production et ainsi repérer le plus tôt possible les erreurs éventuelles.

1.2 Risques

Les risques du projet ont évidemment été pensés en amont, mais surtout ont dû être limités au maximum. Néanmoins, malgré les très nombreux tests effectués, des risques d'erreurs sont toujours et demeureront toujours présents. En effet, il était parfois trop complexe de prendre le temps d'implémenter une solution, ou alors les cas étaient trop spécifiques pour qu'on y consacre du temps alors que les délais étaient très serrés. Nous dressons ici une liste non exhaustive de risques de problèmes :

- Une personne qui ne connaît absolument pas les nouvelles technologies ou même la technologie de manière générale arrive. Elle se place sur l'emplacement indiqué, mais légèrement à côté. Plusieurs fois, l'application lui dit de revenir vu qu'elle ne se situe pas bien dans la zone de contrôle, mais elle ne comprend pas trop ce qu'il faut faire. Une idée d'amélioration et de résolution du problème aurait été d'indiquer clairement à l'utilisateur : « Vous êtes trop à droite de la zone de contrôle ! » par exemple. Mais pour cela, il aurait fallu séparer chaque condition de la limitation de zone, ce qui aurait alourdi et considérablement augmenté la complexité du code. Il est donc possible qu'une personne ne comprenne pas vraiment le problème et qu'elle pense que c'est la technologie ou l'application qui bug. Néanmoins, au vu des très nombreux tests réalisés, ce problème ne devrait pas voir le jour que très très rarement.
- Une personne avec un bras dans le plâtre désire utiliser l'application. Si cette personne est capable de bouger ses deux mains, tout se déroule sans difficulté jusqu'à l'étape de contrôle du curseur. En effet, pour qu'une main prenne le contrôle du curseur de la souris, il faut qu'elle soit au moins 5 cm en avant par rapport à l'autre main, sinon rien ne se passe. Une personne avec le bras dans le plâtre aura une main au niveau du ventre donc assez avancé de base. Il faudra donc que l'autre main le soit encore plus pour bien contrôler, ce qui peut épuiser le bras et plus généralement l'utilisateur. De plus, toutes les étapes où l'on demande l'utilisation des deux mains fermée pour manipuler les images vont être beaucoup plus difficilement réalisables et pourraient générer des états indésirables pour l'application.
- Une personne handicapée ne possédant qu'un seul bras veut tester l'application. Tout se passe bien jusqu'à l'étape 3 où il faut fermer ses deux poings simultanément. Il est donc impossible de terminer l'étape et le tutoriel de manière générale.
- Une personne très grande (dans les 2 m) souhaite utiliser le logiciel. Le mur de la salle étant très proche de la caméra, il est possible que cette personne n'est pas assez de recul pour être totalement reconnue par la caméra et que le haut de son corps soit hors champ. Dans ce cas, dès la première étape il sera confronté à une difficulté majeure puisqu'il faut lever une main au-dessus de sa tête, ce qui ne sera pas pris en compte par la caméra Kinect. Il lui faudra donc se baisser voire se mettre à genoux s'il veut pouvoir poursuivre le tutoriel.

Beaucoup de cas spécifiques peuvent encore être notés. Il existera toujours des problèmes qui peuvent survenir, aussi bien au niveau physique (main difforme, membre en moins, taille trop grande...) que mental (non-compréhension des actions à effectuer, de ce qu'il faut faire, etc.). L'application a été conçue pour fonctionner avec le plus grand nombre, des enfants jusqu'aux personnes âgées, ou encore avec des personnes en fauteuil roulant. Néanmoins, il restera

toujours des cas que nous n'aurons pas rencontrés ou prévus. Ces cas n'auront pas été gérés, car trop complexes. Il ne faut pas oublier non plus de compter les bugs éventuels que nous aurions certainement du faire face. Ces bugs peuvent très bien survenir quand l'application sera disponible dans le musée et que des dizaines voire centaines de personnes la testeront chaque jour.

1.3 Choix technique

Il n'y a pas vraiment eu de choix technique. J'ai repris l'ensemble de l'existant et je l'ai poursuivi. J'ai donc continué le projet en VB.NET avec la version 2013 de Visual Studio et le framework .net.

Du point de vue de la modélisation. J'ai également repris le découpage en module qui avait été réalisé par M. Talon et M. Venturini dont l'objectif était d'obtenir une compréhension rapide et une maintenance efficace. Je me le suis approprié et je l'ai complété en suivant la façon de faire qui avait été décidé lors des spécifications.

1.4 Choix design

De la même manière que pour les choix liés à la technique, le design n'a pas vraiment posé problème. J'ai simplement respecté la « charte graphique » de l'existant et je l'ai complété et embelli pour lui donner un aspect plus professionnel. Le squelette de l'utilisateur a été réduit au niveau de sa taille, car il était « pixélisé » à l'écran et sa couleur a changé du bleu vers l'orange pour qu'il soit plus visible et discernable. J'ai également décidé d'ajouter différentes barres de progression pour aider à la compréhension de l'utilisateur sur ce qu'il est en train de faire notamment dans les périodes de temps ou dans le cas où une action est à répéter plusieurs fois.

2 Stratégie et performance

2.1 Stratégie

La stratégie était simple et clairement définie. Il fallait une application permettant de faire comprendre rapidement et intuitivement les mouvements permettant de réaliser le tutoriel et ainsi que l'utilisateur soit prêt à utiliser l'application de M. Serres.

Pour cela, nous avons séparé l'écran en deux verticalement.

- La partie gauche de l'écran contient la représentation squelettique de l'utilisateur à l'écran et les éléments permettant de réaliser les différentes épreuves associés aux différentes étapes.
- La partie droite de l'écran contient toutes les explications textuelles concernant les mouvements à réaliser et les conditions à remplir pour passer les différentes étapes.

En plus de ça, des audios sont utilisés à chaque étape pour appuyer le texte et améliorer la compréhension, mais également pour faciliter la vie des malvoyants notamment.

Le tutoriel comprend six étapes, organisées dans un ordre bien précis, pour que les personnes apprennent pas à pas tout en augmentant la difficulté graduellement.

Il avait également décidé de faire le maximum de tests utilisateurs possible. Pour cela, des tests ont été réalisés avec des collègues de la promotion, mais également avec quelques membres du personnel du musée et des étudiantes de la faculté d'Histoire. Beaucoup de problèmes et de bugs ont été détectés et résolus grâce à eux.

2.2 Performance

Finalement, les utilisateurs arrivent en grande majorité à prendre en main rapidement le tutoriel d'initiation à la technologie Kinect. Sur l'ensemble des étapes, il y a en deux qui ressort du lot en termes de difficulté : le déplacement et la rotation d'images.

Pour le déplacement d'image, on trouve deux difficultés notables :

- Kinect comprend bien plus facilement que les poings sont fermés si l'utilisateur lui montre ses paumes de mains fermées plutôt que les poings en eux-mêmes.
- Les personnes grandes (plus de 1,90 mètre) ont généralement des difficultés à déplacer l'image verticalement.

Pour la rotation d'image, c'est surtout la compréhension du geste en lui-même qui est compliqué comme nous pouvions nous en douter.

Néanmoins, toutes les autres étapes se déroulent sans aucune difficulté majeure notable. On peut donc considérer que l'objectif est atteint.

3 Analyse des résultats

3.1 Fonctionnement

Nous allons voir ici le fonctionnement général de l'application de médiation Kinect. Pour cela, un exemple semble être le plus adapté.

On lance l'application pour la première fois. L'écran de veille est alors affiché. Le dispositif Kinect ne voit personne dans l'intégralité de son champ de vision, l'écran reste alors le même et ne bouge pas.

Une personne entre dans le champ de vision de Kinect. L'application la repère et l'enregistre dans un tableau de « Body ». À chaque frame envoyée par Kinect, sa position est testée pour savoir si cette personne est dans la zone de contrôle. La personne entre finalement dans la zone de contrôle et se positionne sur l'emplacement indiqué au sol. Grâce à une condition contenue dans le code du module Kinect, le contrôle de l'application lui est donné. L'écran de veille disparaît alors pour lancer le tutoriel et la première étape de celui-ci. Au niveau de l'application, la variable « currentState » a maintenant son état à 1, signe que l'on est maintenant à l'étape 1 du tutoriel (l'écran de veille correspond à l'état 0). Une fois, cela en place, un appel à la classe Animation est réalisé pour afficher et lancer l'interface graphique et l'audio correspondant à l'étape et un appel au module Skeleton est également effectué à chaque frame envoyée par Kinect pour afficher le squelette de l'utilisateur. Le tout se réalise évidemment tant que l'utilisateur est dans la zone de contrôle.

Une fois que l'explication orale du geste est terminée, du texte et un audio apparaissent. Ils proposent la condition et l'épreuve à réaliser pour passer à l'étape suivante. L'utilisateur doit alors remplir cette condition qui sera vérifiée dans le module Kinect. Dans le cas où la condition est remplie, la variable « currentState » s'incrémente et l'étape suivante est appelée de la même manière que pour la première étape. Cette boucle se déroule alors jusqu'à la fin du tutoriel.

Si à un moment, le visiteur sort pour une quelconque raison de la zone de contrôle alors un timer de vingt secondes est déclenché et l'écran de sortie de zone est affiché pour prévenir l'utilisateur que la caméra Kinect ne le détecte plus. Si la personne revient dans les temps, alors elle retourne à l'endroit exact où elle avait quitté l'application. Dans le cas contraire, le tutoriel est remis à zéro et l'écran de veille est affiché, jusqu'à ce qu'une nouvelle personne s'installe dans la zone de contrôle.

Dans le cas maintenant où une deuxième personne entre dans la zone de contrôle alors qu'une personne s'y trouve déjà, l'application compare les positions des deux personnes et donne le contrôle à la personne se trouvant le plus proche de la caméra Kinect.

Nous allons maintenant voir en détail chaque écran.

3.2 Écran de veille

Commençons avec l'écran de veille de l'application.



Figure 1 – Écran de veille

Évidemment, cela n'est pas visible depuis la capture, mais l'image du pied est un GIF qui change cinq fois de couleurs pour donner du dynamisme à l'écran et montrer que celui-ci est actif et en attente d'utilisateur. Cela permet également normalement d'attirer le regard et donc d'éveiller la curiosité des visiteurs.

Un message est présent pour indiquer à l'utilisateur ce qu'il doit faire pour lancer l'application, mais également pour lui présenter le but de celle-ci.

Enfin, un message audio « Envie de découvrir Kinect ? Veuillez-vous positionner sur la marque au sol ! » est lancé au démarrage de l'écran de veille ou bien toutes les cinq minutes s'il n'y a aucun utilisateur pendant cette période de temps.

3.3 Étape 1 - Bouger !

La première étape du tutoriel se lance lorsqu'un utilisateur entre dans la zone contrôle.

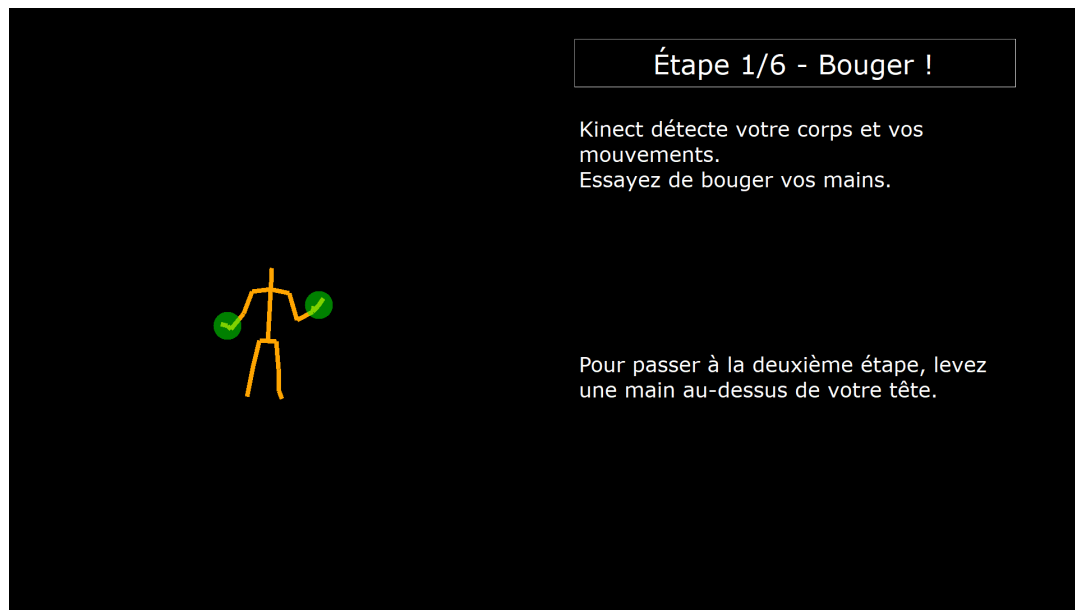


Figure 2 – Étape 1 - Bouger !

Dans l'image, nous pouvons noter trois éléments distincts :

- À gauche, le squelette de l'utilisateur. Certaines jointures du corps humain sont reconnues par la caméra Kinect (mains, coude, tête, etc.), mais pour notre application nous n'affichons que les mains. Le module Skeleton est chargé de créer les « os » un à un entre les différentes jointures. Lorsque le corps de l'utilisateur est bien visible, l'os est dessiné en orange. En revanche, si l'utilisateur met par exemple un bras derrière son dos qui n'est plus visible par la caméra Kinect alors Kinect devine l'emplacement du bras en question et le module l'affiche en blanc à l'écran.
- Ensuite, nous avons à droite l'ensemble des éléments de description de l'étape du tutoriel et du mouvement associé. Cette façon de procéder est la même pour l'ensemble des étapes du tutoriel. Nous avons tout d'abord le titre tout en haut, puis la description du mouvement juste en dessous dans le premier paragraphe et ensuite la condition à réaliser pour passer à l'étape suivante de nouveau en dessous dans le second paragraphe.
- Nous avons également un audio qui se lance en même temps que le premier paragraphe pour expliquer oralement le mouvement puis lorsque celui-ci est terminé, un second audio se lance en même temps qu'apparaît le second paragraphe pour expliquer oralement cette fois-ci la condition à réaliser. Le second paragraphe n'apparaît donc que lorsque le premier audio est terminé.

Pour passer à l'étape suivante ici, il suffit de lever une main au-dessus de sa tête.

L'objectif de cette étape est d'amorcer une première approche en douceur à la technologie Kinect pour indiquer aux utilisateurs que la caméra analyse leur mouvement lorsqu'ils bougent.

3.4 Étape 2 - Déplacer la souris

Après que l'utilisateur ait levé une main au-dessus de sa tête, la deuxième étape de lance.

Maintenant que l'utilisateur sait qu'il peut déplacer ses mains, le but de cette étape est d'utiliser cela pour réaliser quelque chose de concret.

Nous avons, comme à chaque étape, une boîte de dialogue décrivant le geste étudié lors de l'étape et une autre indiquant la condition à réaliser pour passer à l'étape suivante. L'ensemble est, naturellement, accompagné de deux audios reprenant le texte.

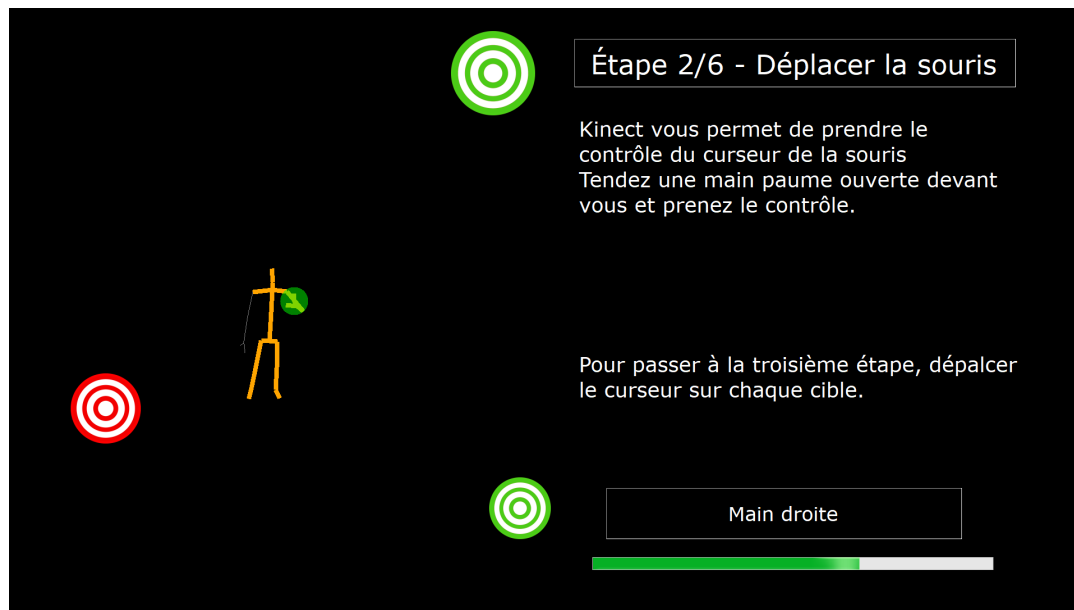


Figure 3 – Étape 2 - Déplacer la souris !

Le but de cette étape est d’initier l’utilisateur au contrôle du curseur de la souris à distance. Pour cela, il doit tendre une main devant lui et bouger sa main comme il a pu le faire à l’étape précédente.

Trois nouveaux éléments sont apparus pour cette étape.

- Trois cibles que l’utilisateur devra atteindre avec le curseur de la souris pour valider celles-ci et valider l’étape. Les cibles sont par défaut de couleur rouge et deviennent vertes lorsque le curseur de la souris est passé sur celles-ci.
- Un cadre sous le deuxième paragraphe indiquant quelle main d’utilisateur a le contrôle du curseur. Cette main correspond à celle étant la plus proche de la caméra Kinect.
- Une barre de progression à trois crans indiquant le nombre de cibles atteintes par l’utilisateur. Par exemple, sur l’image, l’utilisateur a touché deux cibles et la barre de progression est pleine aux deux tiers.

3.5 Étape 3 - Serrer les poings

L’utilisateur a atteint les trois cibles et l’étape trois du tutoriel s’affiche.

Cette étape demande maintenant une interaction avec les mains. L’objectif de cette étape est de faire comprendre à l’utilisateur que Kinect ne se contente pas de détecter la position des mains dans l’espace, mais qu’il est également capable de détecter si les mains sont ouvertes ou fermées. Cette étape est importante, car cette manipulation est à la base de tous les mouvements que pourra utiliser l’utilisateur par la suite.

Depuis le début de l’application, l’utilisateur peut voir qu’il y a un cercle vert autour de ses mains sur la représentation squelettique affichée à l’écran. L’idée ici est simple, si l’utilisateur a les mains ouvertes, le cercle est vert. Dans le cas contraire et que les mains sont fermées, le cercle est rouge. Nous avons donc mis ici un indicateur pour que l’utilisateur sache à tout moment comment réagit Kinect face à l’ouverture ou la fermeture de ses poings.

Enfin, nous avons mis un cadre avec un compteur indiquant le nombre de fois où l’utilisateur a fermé ses deux poings simultanément. Une fois arrivé à cinq, la condition est remplie et on estime que l’utilisateur a compris la manipulation et peut passer à l’étape quatre.



Figure 4 – Étape 3 - Serrer les poings

3.6 Étape 4 - Déplacer un objet

L'utilisateur sait maintenant déplacer ses mains et ouvrir et fermer ses poings. L'étape quatre se lance et nous allons maintenant nous intéresser aux différents mouvements qui seront utilisés sur l'application de M. Serres.



Figure 5 – Étape 4 - Déplacer un objet

L'objectif de cette étape est de combiner l'ensemble des manipulations qu'a pu réaliser l'utilisateur jusqu'à maintenant. L'idée est de prendre la statue et de la déplacer sur son socle. Pour cela, l'utilisateur doit tendre les deux mains devant lui, « attraper » l'objet en fermant les poings et, tout en gardant les poings fermés, déplacer ses poings dans l'espace pour déplacer la statue.

La difficulté vient ici du fait qu'il est plus facile de réaliser le déplacement en plusieurs fois plutôt que de tout faire en une fois. Malheureusement, cela est compliqué à expliquer sachant que la

description initiale du geste prend une place conséquente sur l'écran.

Une fois la statue sur le socle, l'utilisateur arrive sur l'étape cinq du tutoriel.

3.7 Étape 5 - Zoom/Dézoom

L'utilisateur sait maintenant déplacer un objet dans l'espace. Cette cinquième étape lui permettra d'apprendre à agrandir et réduire la taille de l'objet.

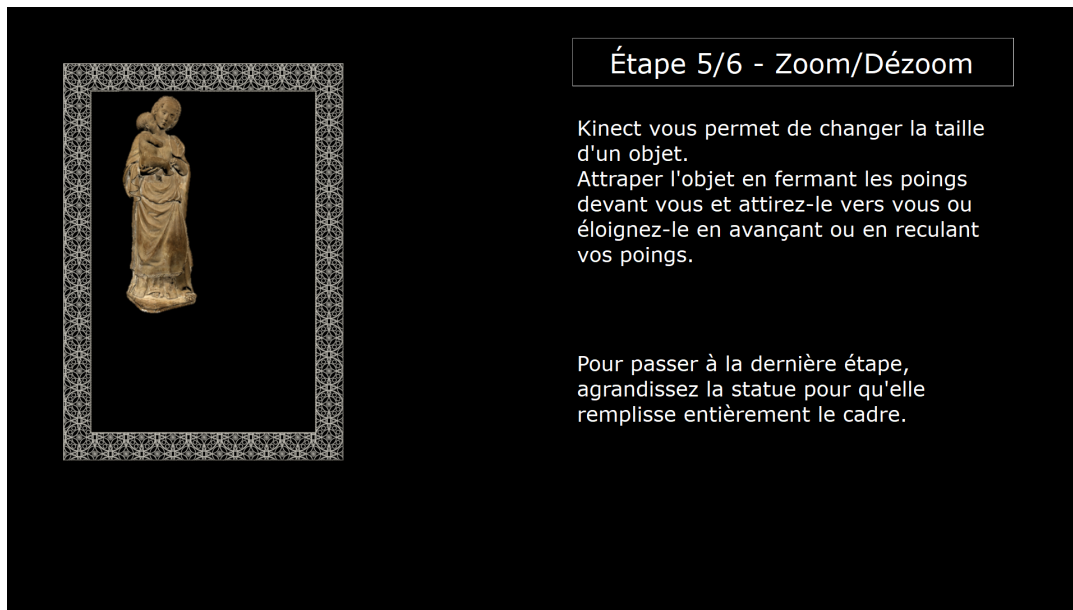


Figure 6 – Étape 5 - Zoom/Dézoom

L'objectif de cette étape est d'apprendre les mouvements permettant d'agrandir la taille de l'image et de zoomer ou au contraire de réduire la taille de celle-ci et de dézoomer.

A l'origine, le mouvement était de tendre les mains devant soi, de fermer les poings et d'écartier les poings pour agrandir la taille de l'image ou bien se rapprocher ses poings pour réduire l'image. Néanmoins, les tests réalisés sur l'application de M. Serres lors de la journée Porte Ouverte de l'Université de Tours et de Polytech Tours ont convaincu M. Venturini et M. Serres de changer ce mouvement, car les utilisateurs trouvaient plus intuitif le mouvement de ramener ses poings vers soi plutôt que de les écartier.

Nous avons donc changé et gardé ce dernier. En conséquence, dorénavant pour agrandir la taille de l'image l'utilisateur doit tendre les mains devant lui, fermer les poings et ramener ses poings vers lui. Évidemment, si l'utilisateur approche ses poings à l'inverse, vers la caméra, alors la taille de l'image sera réduite.

La condition pour passer cette étape est d'agrandir suffisamment l'image pour qu'elle remplisse entièrement le cadre.

3.8 Étape 6 - Tourner un objet

L'utilisateur est maintenant face à la dernière étape du tutoriel. Cette étape n'était pas du tout présente à la base dans l'existant, mais elle avait tout de même été réfléchi et prête à être ajoutée.



Figure 7 – Étape 6 - Tourner un objet

L'objectif de cette étape est d'apprendre le mouvement permettant de tourner une image 3D. Nous pouvons noter une différence majeure avec l'application de M. Serres. Dans cette application, nous tournons une image 3D dans un plan 2D tandis que dans celle de M. Serres nous tournons en 3D autour de la statue. Néanmoins, le mouvement et l'idée restent les mêmes.

Le mouvement est assez complexe et c'est pour cela qu'il est demandé en dernier. Il faut tendre les mains devant soi, fermer les poings et déplacer ses poings comme si l'on voulait tourner un volant horizontal (plus de détails dans le manuel utilisateur). Lorsque ce mouvement est réalisé, la statue tourne sur elle-même à droite ou à gauche selon le sens de rotation. Lorsque la statue tourne, la barre de progression en bas de l'écran se remplit.

La condition est de tourner la statue jusqu'à ce que la barre de progression soit pleine. Le sens et la vitesse du mouvement ne rentrent pas en compte.

3.9 Sortie de zone

Cet écran est affiché lorsque l'utilisateur sort de la zone de contrôle, que ce soit volontaire ou non. Un timer de vingt secondes est déclenché permettant à l'utilisateur de revenir dans la zone s'il le souhaite. Si l'utilisateur revient, le tutoriel revient à l'écran exact où le visiteur l'a quitté. En revanche, si les vingt secondes sont écoulées l'écran de veille est affiché.

Dans un souci de clarté, nous avons mis la même image que pour l'écran d'accueil afin d'indiquer que le problème vient de la non-présence d'un individu dans la zone de contrôle.

Un fichier audio est également utilisé pour prévenir l'utilisateur qu'il n'est plus dans la zone et une barre de progression augmente la visibilité et la compréhension du timer.

3.10 Écran de fin

Certains aléas ont fait que je n'ai pas d'image de l'écran de fin à présenter ici. Néanmoins, l'écran de fin est le même que pour l'écran de veille ou de sortie de zone à l'exception près que le GIF est remplacé une image de félicitations et que le message indiqué félicité l'utilisateur et l'invite à continuer l'exposition.



Figure 8 – Écran de sortie de zone

Un fichier audio est également là pour féliciter l'utilisateur et l'orienter vers la suite de l'exposition.

3.11 Conclusion

Pour conclure, le tutoriel dure entre quatre et sept minutes selon la vitesse de l'utilisateur. Le temps vient essentiellement de la durée des différents fichiers audios et de la difficulté d'une personne à effectuer un mouvement. De plus, le fait de devoir attendre vingt secondes lorsque quelqu'un quitte l'application avant d'avoir fini pour que le tutoriel soit remis à zéro est également du temps de perdu. Cependant, cette durée a été décidée d'un accord commun entre M. Venturini et moi-même.

6

Bilan et conclusion

1 Travail réalisé

1.1 Première partie - Semestre 9

Ce semestre a été l'occasion de travailler sur plusieurs aspects du projet. La vraie difficulté que j'ai rencontrée fut l'annonce de la création d'une application de découverte du Leap Motion à un mois du terme du PRD1. J'ai donc dû ajouter à mon projet une nouvelle application et ajouté celle-ci dans mon cahier de spécifications.

1.1.1 Analyse de l'existant

La première partie a été de prendre en main l'application existante développée par M. Arnaud Talon l'année dernière. Bien que suffisamment documentée, cette partie fut plus longue que ce que j'avais imaginé. Il a fallu également réparer quelques bugs majeurs qui s'étaient glissés au sein de celle-ci pour pouvoir tester et effectuer une démo de cette application. Les démonstrations des applications développées par M. Serres et M. Venturini ont été faites très tardivement, ce qui a considérablement retardé la rédaction de la partie 2.1. de ce rapport ainsi que l'analyse et la conception de l'application de découverte du Leap Motion, car nous nous sommes décidés sur les fonctionnalités de cette application à l'issue de ces démonstrations.

1.1.2 État de l'art

L'État de l'art a été assez difficile à réaliser pour le Kinect, étant donné que Microsoft a annoncé la mort de celle-ci en plein milieu du semestre. Il a donc fallu revoir mes plans à mi-parcours et se concentrer sur les applications qui ont été développées par le passé.

À l'inverse, l'étude du Leap Motion a été plus simple, mais réalisée un peu tardivement. Le problème a donc été de diriger mes recherches sur le Leap Motion pour des applications utilisables en musée.

1.1.3 Interfaces

La mise en place des maquettes d'interfaces a pris une part importante de temps sur ce semestre. En effet, nous essayons de développer des applications les plus ergonomiques et intuitives possibles et les interfaces sont une partie très importante à ne pas négliger. Pour cela, en étant en constante discussion avec M. Venturini nous avons mis en place plusieurs versions d'interfaces avec quelques corrections à chaque fois. Actuellement, et au sein de ce rapport en annexe nous pouvons trouver les versions définitives des interfaces pour l'application de médiation Kinect et l'application de découverte Leap Motion. Ces versions correspondent aux versions 3 et versions 2 respectivement pour Kinect et Leap Motion.

1.1.4 Analyse et conception

L'analyse et la conception pour l'application de médiation Kinect a consisté en une analyse des diagrammes existants notamment le diagramme de classe. En revanche, nous avons pris du retard sur l'analyse de l'application de découverte Leap Motion et la conception sera à compléter et terminer dès la fin du mois de Décembre pour commencer le développement dès le début du mois de Janvier.

1.2 Seconde partie - Semestre 10

Ce semestre a été l'occasion de mettre en pratique l'ensemble des spécificités (à quelques changements prêts) que j'avais moi-même réalisé lors du semestre précédent. C'est avec joie et fierté que je suis arrivé au bout du développement de l'application de médiation Kinect malgré les nombreuses difficultés rencontrées en chemin. Néanmoins, c'est toujours décevant de devoir abandonner une application notamment pour le Leap Motion, mais cela m'a permis de me concentrer exclusivement sur la première application et de fournir à l'arrivée une application performante.

1.2.1 Application de médiation Kinect

À partir de l'existant fourni par M. Arnaud Talon lors du PRD de l'année 2016-2017 je devais terminer l'application de médiation pour la technologie Kinect. Pour cela, je me suis approprié le sujet, notamment les outils et langages que sont le VB.NET et Visual Studio 2013. La veille technologique du semestre 9 m'a permis de me familiariser avec la technologie et la façon de développer. La mise en route a été compliquée puisque j'ai dû faire face à un bug qui m'a pris deux semaines pour être corrigé complètement. Néanmoins, et avec la décision de mettre de côté la deuxième application, tout est rentré dans les temps et l'application est maintenant opérationnelle et prête à être déployé au musée.

1.2.2 Application de découverte Leap Motion

Suite aux différentes difficultés rencontrées lors du développement sur l'application de médiation Kinect il a été décidé d'un accord commun avec M. Venturini de mettre de côté cette application et de s'en passer. Cette stratégie a été payante sur deux aspects puisque cela m'a permis de finir la première application dans les temps et avec un gage qualité vraisemblablement supérieure, mais également lors des tests avec le musée nous avons pu voir que cette technologie n'avait pas tant besoin que ça d'une application de médiation ou de découverte, car elle est suffisamment intuitive toute seule.

1.2.3 Tests et qualité

Les tests ont sans doute été la tâche la plus importante dans ce projet. En effet, des tests étaient effectués chaque semaine, aussi bien avec moi-même, des testeurs courants (comme l'équipe travaillant avec le MOA) ou des nouveaux n'ayant jamais vu l'application avant. Je remercie d'ailleurs chaque personne ayant réalisé le tutoriel au moins une fois, et elles sont nombreuses !

Grâce à tous ces tests continus, de nombreuses corrections ont été réalisées. Que ce soit des erreurs de compréhension des testeurs, des bugs d'applications, des erreurs de repérage de la caméra Kinect ou encore des améliorations à ajouter, chaque test a permis de faire des identifications efficaces et d'avancer pas à pas vers une application fonctionnelle et performante.

2 Gestion de projet

2.1 Planning du Semestre 10

Un diagramme de Gantt est disponible en annexe pour détailler le plan de développement du semestre 10. Néanmoins, ce semestre va démarrer comme attendu par la conception et l'analyse de l'application de découverte Leap Motion. Il faudra, au cours de ce semestre, faire très attention aux délais qui nous sont imposés. Les projets sont très ambitieux en matière de fonctionnalités, mais il ne faudra pas avoir peur de réduire un peu l'application Leap Motion si nous sentons que les délais deviennent très fragiles.

2.2 Diagramme de Gantt

Comme indiqué plusieurs fois, l'application de découverte Leap Motion a été abandonnée puisque la durée des sprints définis au semestre 9 n'était juste pas possible à tenir. J'ai donc, dans l'urgence, redéfini un nouveau diagramme de Gantt que l'on peut retrouver dans les annexes de ce rapport. Ce nouveau diagramme a pris en compte les remarques que j'ai pu recevoir lors de ma soutenance du semestre 9 notamment. J'ai donc pris la peine de bien spécifier les phases de tests et ajouter le temps nécessaire à la réalisation de la documentation.

En fin de compte, ce nouveau planning a été parfaitement respecté et nous avons pu arriver à son terme sans trop de problèmes. Étant donné qu'il y avait quatre sprints, quatre livrables ont été générés dont la version finale de l'application de médiation Kinect.

7

Annexes

1 Spécifications fonctionnelles

1.1 Application de médiation Kinect

1.1.1 Étape 1 — Déplacement des mains

Déplacement des mains

Présentation de la fonction :

- Objectif : Initier l'utilisateur au fait qu'il puisse et doive utiliser son corps pour interagir avec le système.
- Priorité : Cette fonction est primordiale dans l'application et se doit d'être la première en action.

Description de la fonction :

- Préconditions : L'utilisateur est détecté par la caméra Kinect et est en place dans la zone de contrôle.
- Postconditions : Avoir réalisé avec succès le déplacement des mains et lever une main. Être toujours dans la zone de détection et de contrôle de la caméra Kinect.
- L'application décrit les actions de déplacement des mains à travers une description orale enregistrée et un texte affiché. Une petite image animée ou une vidéo illustre également l'action à réaliser. L'utilisateur doit déplacer ses mains peu où il le souhaite et ensuite lever une main lorsque l'application le lui demande. La réalisation de cette dernière déclenche l'étape 2 et valide l'étape 1.

1.1.2 Étape 2 — Contrôle du curseur de la souris

Contrôle du curseur de la souris

Présentation de la fonction :

- Objectif : Initier l'utilisateur au fait qu'il peut déplacer le curseur de la souris grâce au déplacement de ses mains.
- Priorité : Cette fonction est primordiale dans l'application. Elle se situe en deuxième dans la hiérarchie du tutoriel.

Description de la fonction :

- Préconditions : L'utilisateur est dans la zone de contrôle et « Étape 1 – Déplacement des mains » a été réalisée.
- Postconditions : Avoir réalisé avec succès le déplacement du curseur aux endroits demandés et être toujours dans la zone de détection et de contrôle.
- L'application décrit les actions de contrôles du curseur de la souris à travers une description orale enregistrée et un texte affiché. Une petite image animée ou une vidéo illustre également l'action à réaliser. L'idée est d'appliquer immédiatement ce que l'on a appris durant l'étape 1 en déplaçant ses mains pour déplacer le curseur de la souris. L'objectif est de déplacer le curseur de la souris à différents endroits de l'écran. Nous aurons plusieurs cibles à atteindre et lorsque le curseur passera sur ces cibles elles seront validées. Une fois toutes les cibles validées, cette étape est réussie.

1.1.3 Étape 3 — Ouverture et fermeture des poings

Ouverture et fermeture des poings

Présentation de la fonction :

- Objectif : Initier l'utilisateur au fait qu'il peut ouvrir et fermer les poings pour interagir avec le système.
- Priorité : Cette fonction est primordiale dans l'application. Elle se situe en troisième dans la hiérarchie du tutoriel.

Description de la fonction :

- Préconditions : L'utilisateur est dans la zone de contrôle et « Étape 2 – Contrôle du curseur de la souris » a été réalisée.
- Postconditions : Avoir réalisé avec succès l'ouverture et la fermeture des poings demandés par l'application et être toujours au sein de la zone de détection et de contrôle.
- L'application décrit les actions d'ouvertures et de fermetures des poings à travers une description orale enregistrée et un texte affiché. Une petite image animée ou une vidéo illustre l'action à réaliser. Il s'agit ici de tout d'abord ouvrir et fermer ses poings de manière aléatoire pendant quelques secondes pour que l'utilisateur prenne conscience du mouvement. Il faut ensuite réaliser une courte série de dix ouvertures et fermetures des poings. Chaque main comptant pour un il faut donc ouvrir et fermer chaque poing cinq fois.

1.1.4 Étape 4 — Déplacement d'une image

Déplacement d'une image à distance

Présentation de la fonction :

- Objectif : Initier l'utilisateur au déplacement d'une image en l'attrapant grâce à la fermeture des poings et en la déplaçant dans l'espace.
- Priorité : Cette fonction est primordiale dans l'application. Elle se situe en quatrième dans la hiérarchie du tutoriel.

Description de la fonction :

- Préconditions : L'utilisateur est dans la zone de contrôle et « Étape 3 – Ouverture et fermeture des poings » a été correctement réalisée.
- Postconditions : Avoir réalisé avec succès le déplacement de l'image de sa position initiale jusqu'à la position demandée et être toujours au sein de la zone de détection et de contrôle.
- L'application décrit comment fonctionne le déplacement d'une image grâce à une description orale enregistrée et un texte affiché. Une petite image animée ou une vidéo renforce l'apprentissage en illustrant l'action à réaliser. Il s'agit ici de tendre les mains devant soi, l'utilisateur ferme les poings et « attrape » l'image. À partir de là, tant qu'il garde les poings fermés l'image se déplace en même temps qu'il déplace ses bras dans l'espace. L'objectif est d'amener une balle de handball de son point de départ jusqu'au but.

1.1.5 Étape 5 — Changement de taille d'une image

Changement de taille d'une image

Présentation de la fonction :

- Objectif : Initier l'utilisateur aux méthodes pour redimensionner une image
- Priorité : Cette fonction est primordiale dans l'application. Elle se situe en cinquième dans la hiérarchie du tutoriel.

Description de la fonction :

- Préconditions : L'utilisateur est dans la zone de contrôle et « Étape 4 – Déplacement d'une image » a correctement été réalisée.
- Postconditions : Avoir réalisé avec succès le redimensionnement d'une image et être toujours dans la zone de détection et de contrôle de la caméra Kinect.
- L'application décrit comment fonctionne le redimensionnement de la taille d'une image grâce à une description orale enregistrée et un texte affiché. Une petite image animée renforce l'apprentissage en illustrant l'action à réaliser. Il s'agit ici de tendre les poings pour « attraper » l'image puis en rapprochant les poings l'un vers l'autre l'utilisateur peut faire zoomer l'image. Dans le cas contraire en écartant les poings l'un de l'autre l'utilisateur agrandit l'image. Cette pratique s'inspire fortement de l'utilisation des smartphones. L'objectif de cette étape est d'augmenter la taille d'une tache de peinture apparue à l'écran suffisamment pour qu'elle recouvre le cadre autour d'elle.

1.1.6 Étape 6 — Rotation d'une image

Rotation d'une image

Présentation de la fonction :

- Objectif : Initier l'utilisateur au mouvement de rotation.
- Priorité : Cette fonction est primordiale dans l'application. Elle se situe en dernière dans la hiérarchie du tutoriel.

Description de la fonction :

- Préconditions : L'utilisateur est dans la zone de contrôle et « Étape 5 – Changement de taille d'une image » a été correctement réalisée.
- Postconditions : Avoir réalisé avec succès la rotation d'une image sur elle-même et être encore dans la zone de détection et de contrôle.
- L'application décrit comment fonctionne la rotation d'une image grâce à une description orale enregistrée et un texte affiché. Une petite image animée ou une vidéo renforce l'apprentissage en illustrant l'action à réaliser. Il s'agit ici de mettre en avant le bras

correspondant à la direction dans laquelle on veut tourner l'image (droite ou gauche) et effectuer avec celui-ci un mouvement circulaire dans l'espace de profondeur à la manière d'une rotation d'un volant avec poignée. L'objectif est de faire avancer une vidéo de quelques secondes du début jusqu'à son terme. Le mouvement de rotation remplaçant alors l'avancée automatique de celle-ci.

1.1.7 Entrée dans le champ de détection de la caméra Kinect

Gestion de l'entrée dans le champ de détection de la caméra Kinect

Présentation de la fonction :

- Objectif : Prendre en compte l'arrivée d'une personne dans la zone de détection et régler les problèmes de conflits si plusieurs personnes sont à l'intérieur en même temps.
- Priorité : Cette fonction est primordiale dans l'application. Le reste du tutoriel dépend de celle-ci.

Description de la fonction :

- Préconditions : /
- Postconditions : /
- Il y a ici deux possibilités :
 - L'application détecte un utilisateur qui rentre à l'intérieur de la zone de détection et de contrôle et lance en conséquence la première phase du tutoriel pour que l'utilisateur puisse la réaliser.
 - Une personne s'insère dans le champ de contrôle de la caméra alors qu'une autre personne s'y trouve déjà. Dans ce cas, l'application ne prend pas en compte cette personne.

1.1.8 Sortie du champ de détection de la caméra Kinect

Gestion de la sortie du champ de détection de la caméra Kinect

Présentation de la fonction :

- Objectif : Prendre en compte la sortie de l'utilisateur de la zone de détection.
- Priorité : Cette fonction est primordiale dans l'application.

Description de la fonction :

- Préconditions : L'utilisateur est dans la zone de contrôle
- Postconditions : /
- L'application détecte que l'utilisateur présent au sein de la zone de contrôle en sort. Deux cas apparaissent ici :
 - L'utilisateur a terminé le tutoriel et une fois sorti du champ l'application retourne sur l'écran d'attente du tutoriel pour qu'une nouvelle personne y accède.
 - L'utilisateur sort du champ de contrôle en plein milieu d'une phase du tutoriel. Dans ce cas, l'application invite l'utilisateur à revenir dans la zone et attend quelques instants. Si jamais l'utilisateur ne revient pas, l'application revient sur l'écran d'accueil dans l'attente d'une nouvelle personne.

1.2 Application de découverte Leap Motion

1.2.1 Menu

Menu principal de l'application

Présentation de la fonction :

- Objectif : Permettre à l'utilisateur de choisir entre le module de dessin ou le module de manipulation d'image.
- Priorité : Primordiale dans l'application. Cependant, si l'un des modules est amené à ne pas être développé pour une quelconque raison cette fonctionnalité sera amenée à disparaître également.

Description de la fonction :

- Préconditions : L'utilisateur a sa main dans la zone de détection du capteur.
- Postconditions : L'utilisateur a choisi un des deux modules
- L'application affiche deux icônes correspondant aux deux modules. En cliquant sur l'une des deux images le programme ouvre le module souhaité en conséquence.

1.2.2 Dessiner

Module de dessin avec utilisation de l'outil Leap Motion

Présentation de la fonction :

- Objectif : Permettre à l'utilisateur de découvrir le Leap Motion à travers un module de dessin à l'image de l'outil Paint sous Windows.
- Priorité : ette fonction est primordiale dans cette application et secondaire par rapport à l'application de médiation Kinect

Description de la fonction :

- Préconditions : L'utilisateur a sa main dans le champ de contrôle du Leap Motion
- Postconditions : /
- L'application ouvre un espace de dessin vierge. Le contrôle complet se fait par le déplacement de la main dans l'espace. L'utilisateur a plusieurs choix :
 - Une sélection de différentes couleurs
 - Dessiner ou Effacer
 - Plusieurs tailles de crayons/gommes
 - Une possibilité d'effacer complètement le dessin
 - Retourner au menu

1.2.3 Manipulation d'images

Module de manipulation d'image avec l'outil Leap Motion

Présentation de la fonction :

- Objectif : Permettre à l'utilisateur de découvrir le Leap Motion à travers un module de manipulation d'image.
- Priorité : Secondaire dans cette application et secondaire par rapport à l'application de médiation Kinect.

Description de la fonction :

- Préconditions : L'utilisateur a sa main dans le champ de contrôle du Leap Motion
- Postconditions : /
- L'application ouvre une série d'images prédéfinie. L'utilisateur sélectionne alors l'une d'entre elles et peut la manipuler comme il le souhaite : agrandir, réduire, tourner, déplacer, colorier. À tout moment, il peut fermer l'image en cours et en choisir une autre. Encore une fois, le contrôle du module s'effectue à distance par simple manipulation de la main dans l'espace.

1.2.4 Entrée dans le champ de détection du Leap Motion

Gestion de l'entrée dans le champ de détection du Leap Motion

Présentation de la fonction :

- Objectif : Prendre en compte l'arrivée d'une main dans le champ de détection du Leap Motion et régler les problèmes de conflits si plusieurs mains s'y trouvent en même temps.
- Priorité : Cette fonction est primordiale dans l'application. Cette fonction assure la stabilité de tout le reste.

Description de la fonction :

- Préconditions : /
- Postconditions : /
- • Il y a ici deux possibilités :
 - L'application détecte une main qui rentre à l'intérieur de la zone de détection et lance en conséquence le menu principal.
 - Une main s'insère dans le champ de contrôle de la caméra alors qu'une autre s'y trouve déjà. Dans ce cas, l'application indique à l'utilisateur qu'il y a une main en trop dans le champ de détection.

1.2.5 Sortie du champ de détection Leap Motion

Gestion de la sortie du champ de détection du Leap Motion

Présentation de la fonction :

- Objectif : Prendre en compte la sortie d'une main de la zone de détection.
- Priorité : Cette fonction est primordiale dans l'application.

Description de la fonction :

- Préconditions : Une main est présente dans la zone de contrôle
- Postconditions : /
- L'application détecte que la main présente au sein de la zone de contrôle en sort. L'application indique alors clairement à l'écran qu'elle a perdu le contact avec la main de l'utilisateur. Elle déclenche alors un timer affiché à l'écran. Si l'utilisateur revient, l'application repart à l'endroit exact où elle s'était momentanément stoppée. Dans le cas contraire, l'application est réinitialisée sur l'écran d'accueil en attente d'une nouvelle personne.

2 Spécifications non fonctionnelles

2.1 Contraintes de développement et conception

2.1.1 Matériels

Les principales contraintes matérielles viennent de l'usage de la caméra Kinect et du dispositif Leap Motion. L'utilisation de ces outils implique d'avoir des machines suffisamment puissantes pour pouvoir les supporter et au moins un port USB3 pour chacun. De plus, la technologie Kinect étant dépréciée et n'étant maintenant plus maintenu et produite par Microsoft il ne faudra pas qu'un incident survienne sur l'appareil sous peine d'avoir des difficultés pour le remplacer.

2.1.2 Langages de programmation

L'utilisation du SDK Kinect pour Windows nous oblige à utiliser les langages Microsoft donc le VB.NET ou le C#. Évidemment pour la poursuite du projet existant Kinect nous continuerons d'utiliser le VB.NET.

2.1.3 Logiciels et bibliothèques

Le langage de programmation utilisé ne nous oblige pas, mais nous conforte dans l'idée d'utiliser l'IDE Visual Studio pour le développement. La version utilisée depuis le début pour le développement était la version 2013, mais celle-ci n'est plus disponible sur le Microsoft DreamSpark ce qui oblige à développer sur la machine qui contient déjà le projet sauf si le portage vers les versions récentes est possible sans difficulté. Afin d'utiliser et de développer avec la caméra Kinect, nous avons besoin du SDK Kinect pour Windows. De même pour développer avec le dispositif Leap Motion nous avons besoin du SDK Leap Motion pour Windows.

Les projets sont également ambitieux, mais les échéances sont proches et le temps de développement accordé au projet assez court. Cela devrait nous contraindre d'utiliser des bibliothèques existantes notamment pour le projet Leap Motion pour accélérer et simplifier le processus de production. Dans ce cas précis, nous nous retrouverons contraints par les outils trouvables libres de droits et leur utilité.

2.1.4 Environnement

Il sera possible et utile d'utiliser le logiciel Kinect Studio 2.0 pour bien mettre en place la caméra dans la pièce.

2.2 Contraintes de fonctionnement et d'exploitation

2.2.1 Performances

Au niveau de l'utilisateur

Dans les deux cas, nous sommes dans des applications de retranscriptions de mouvements en temps réel. Néanmoins, nos besoins de performances diffèrent d'une application à l'autre. Pour l'application de médiation Kinect, nous sommes faces à un tutoriel presque obligatoire que les visiteurs devront suivre avant de se lancer dans les autres applications. Le but est donc que cela se déroule rapidement en quelques minutes pour éviter de se retrouver avec une file d'attente. Les temps de réponse devront donc être quasi immédiat pour ne pas dire instantané. Le but est que les phases d'initiations s'enchaînent, mais il y a risque important qu'un utilisateur quitte l'application si celle ne réagit pas en même temps que lui. Il faut donc que l'application soit parfaitement réactive et fluide. On ne pourra pas non plus accepter un peu de latence en raison de soucis de calibration également.

Pour l'application de découverte Leap Motion, l'application est en libre accès en continu pour les utilisateurs sans limites de temps. L'objectif sera donc que celle-ci soit parfaitement fluide puisque l'on ne peut pas se permettre que l'application ne réponde pas correctement aux mouvements de la main de l'utilisateur. Cependant, les critères d'optimisations seront moins élevés.

Au niveau de l'environnement

La fréquence d'utilisation dépendra de beaucoup de facteurs : selon les jours de la semaine, les personnes présentes, le débit ou la fin de l'exposition... Tout un tas d'éléments qui font que le taux d'utilisation de l'application variera au jour le jour. Néanmoins, on peut s'imaginer et essayer de déterminer plus ou moins bien le nombre et le type de personnes présentes. Nous devrions avoir plus de visiteurs les samedis et plus d'enfants pendant les vacances scolaires par exemple.

2.2.2 Capacités

Les applications ne s'adressent qu'à une personne à la fois maximum. Elles seront donc configurées pour n'accepter qu'une seule personne dans le champ de vision ou une seule paire de mains. Cette limite s'impose d'elle-même dans un souci de complexité de mise en œuvre et surtout d'ergonomie et d'intuitivité pour les utilisateurs.

2.2.3 Modes de fonctionnement

Les machines contenant les différentes applications seront difficilement accessibles une fois déployées sur place. Par conséquent, il faudra mettre en place quelques configurations. Les applications seront par exemple lancées automatiquement lors de la mise en tension des machines. Le projet de M. Pierre Duchêne permettra le monitoring des différentes machines et applications à distance.

2.2.4 Contrôlabilité

Une application est en cours de développement par M. Pierre Duchêne afin de pouvoir monitorer et gérer l'ensemble du parc-machine présent sur place au musée. Le but est de surveiller l'état des machines à distances, les allumer ou les éteindre, mais également pouvoir agir et réagir dessus. Nous mettrons tout de même en place une gestion de fichiers de log afin de garder des informations régulières sur l'état de nos programmes.

2.2.5 Sécurité

Les applications seront disponibles sur des machines inaccessibles et hors d'atteintes pour les utilisateurs. Seules les personnes connaissant leurs emplacements (les administrateurs et une partie du personnel du musée en somme) auront la possibilité d'y accéder. Toutefois, elles seront quand même protégées par mot de passe. Les utilisateurs donc, quant à eux, n'auront aucun accès direct à la machine, car les applications avec curseur seront en plein écran et qu'il n'y a pas de clavier à disposition.

2.2.6 Intégrité

Le projet de monitoring à distance devra permettre de prévenir les problèmes de déconnexion imprévue, de pertes d'informations ou même de problèmes purement matériels. Malgré cela, il convient que nos applications ne souffrent d'aucune perte mémoire étant donné qu'elles seront en fonctionnement tout au long de la journée sans interruption.

2.3 Maintenance et évolution du système

La maintenance des systèmes aura lieu pendant toute la durée de l'exposition c'est-à-dire pendant trois mois. Il s'agira de s'assurer que les applications soient performantes et actives pendant tout ce temps. En cas de problème, il faudra être particulièrement réactif, car quoiqu'il arrive l'exposition ne s'arrêtera pas pour attendre que nos applications soient réparées. L'application de monitoring de M. Pierre Duchêne est là pour aider à la maintenance et prévenir les différents problèmes qui pourraient survenir.

Il ne devrait pas y avoir d'évolution pour le système Kinect. L'application est destinée à être utilisée pour l'exposition sans aller plus loin. Elle peut toujours servir de point d'appui pour des projets futurs, mais la dépréciation et l'abandon de la technologie par Microsoft remet fortement en cause ce dernier point.

En revanche, les applications Leap Motion pourront nous permettre d'ouvrir la voie à d'autres projets. Ces applications pourront nous servir de point d'appui étant donné que la technologie Leap Motion est en pleine extension et que son utilisation avec les casques de réalité virtuelle de plus en plus fréquente et appréciée.

3 Interfaces

3.1 Interfaces matériel/logiciel

Pour utiliser notre application Kinect, nous allons utiliser un ordinateur et une caméra Kinect évidemment, mais également un vidéoprojecteur. Notre caméra captera les mouvements de l'utilisateur et enverra les données à notre application. Le vidéoprojecteur sera là bien sûr pour projeter l'interface de l'application.

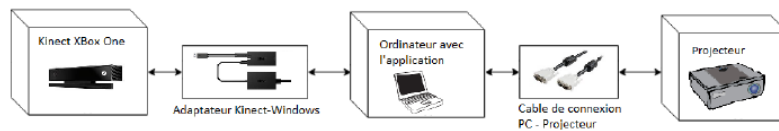


Figure 1 – Application de médiation Kinect - Architecture de déploiement

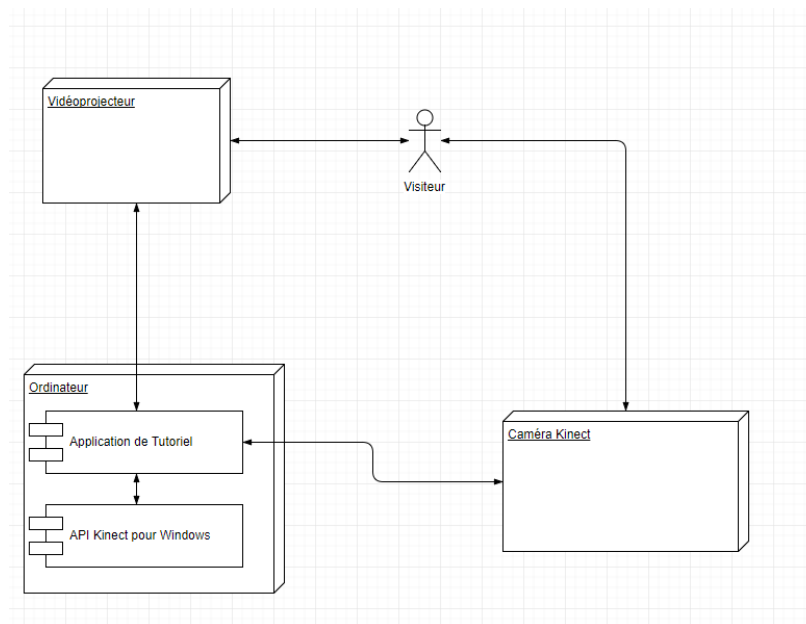


Figure 2 – Application de médiation Kinect - Diagramme de déploiement

Le même schéma est la même architecture s'applique pour le capteur Leap Motion, à l'exception de l'utilisation d'un simple câble USB3 au lieu d'un adaptateur pour le relié à l'ordinateur.

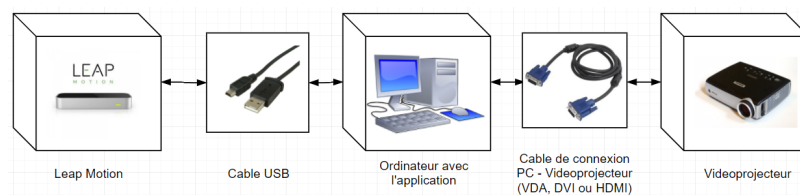


Figure 3 – Application de découverte Leap Motion - Architecture de déploiement

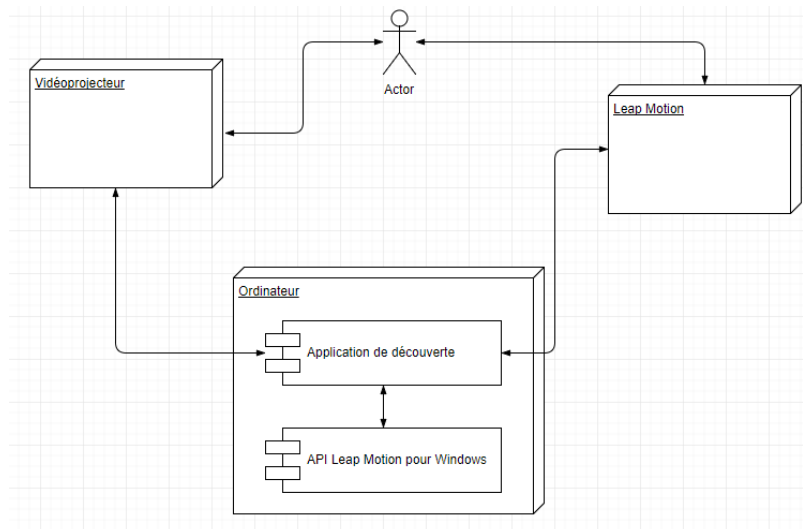


Figure 4 – Application de découverte Leap Motion - Diagramme de déploiement

3.2 Interfaces homme/machine

3.2.1 Application de médiation Kinect - Description

L'idée est que n'importe qui puisse utiliser l'application. Peu importe que le visiteur soit petit, grand, homme, femme, jeune, âgé, sous handicap, etc.

Dans un premier temps, l'utilisateur devra se placer dans une zone spécifique afin de se faire détecter par la caméra Kinect et prendre le contrôle du logiciel. L'idéal serait de placer plusieurs zones (deux ou trois) afin de gérer les différences de taille. On marquera ces zones au sol par exemple par des empreintes de pas plus ou moins grandes selon la distance pour indiquer que les plus petits seront plus proches de la caméra. L'idéal serait que ce soit cette application de tutoriel qui leur permette de trouver leur marque, ainsi ils se placeront par défaut à l'endroit qui leur correspond sur les applications principales.

Une fois en place, l'utilisateur aura le contrôle du logiciel et devra réaliser le tutoriel. L'application devra donc être parfaitement ergonomique et intuitive pour éviter toute hésitation ou toute perte de temps. Les interfaces seront donc sobres, dans l'intention que le tout soit parfaitement net, précis et concis. Le but n'est pas que la personne se décourage ou trouve l'utilisation de l'outil encore plus difficile que lorsqu'il est arrivé.

Un gros point sera donc mis sur l'ergonomie du système pour celle-ci soit la plus simple possible. Une assistance sonore sera également présente pour aider l'utilisateur. On partira donc sur des interfaces sobres, mais efficaces en termes d'ergonomie. Celles-ci sont disponibles en Annexes.

3.2.2 Application de médiation Kinect - Maquettes des interfaces



Figure 5 – Maquette de l'interface de l'écran d'accueil et d'attente d'utilisateur

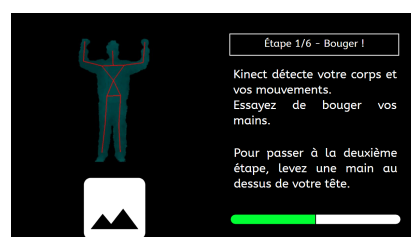


Figure 6 – Maquette de l'interface de l'étape 1 du tutoriel

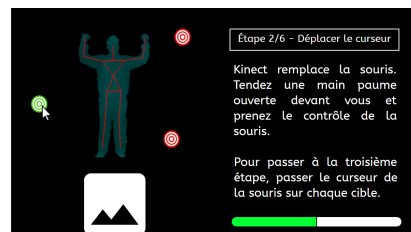


Figure 7 – Maquette de l'interface de l'étape 2 du tutoriel

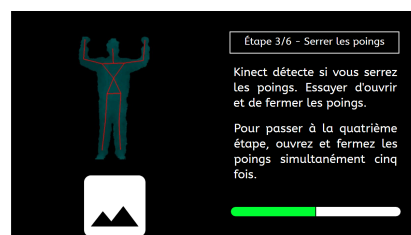


Figure 8 – Maquette de l'interface de l'étape 3 du tutoriel



Figure 9 – Maquette de l'interface de l'étape 4 du tutoriel

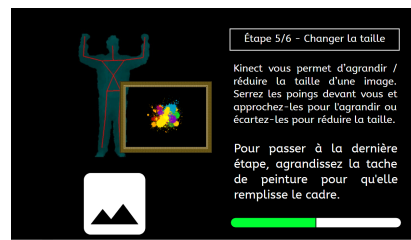


Figure 10 – Maquette de l'interface de l'étape 5 du tutoriel

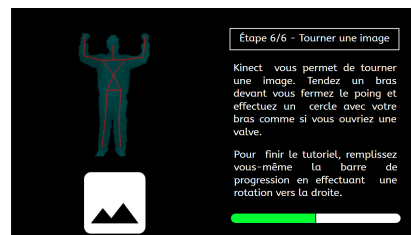


Figure 11 – Maquette de l'interface de l'étape 6 du tutoriel

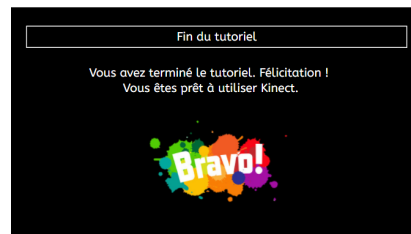


Figure 12 – Maquette de l'interface de l'écran de réussite et de fin du tutoriel



Figure 13 – Maquette de l'interface de l'écran indiquant la sortie de l'utilisateur de la zone de contrôle de Kinect

3.2.3 Application de découverte Leap Motion - Description

Il y a moins de restrictions pour cette application, car elle n'est pas scénarisée. L'objectif est donc d'avoir une application très ergonomique et intuitive pour être compris du premier coup d'œil. C'est pour cela que nous avons misé sur de grandes icônes cliquables. Pour chaque module, nous sommes également partis sur une interface très épurée avec aucun sous-menu. L'intérêt est que l'ensemble des fonctionnalités soit accessible et visible sans devoir chercher.

3.2.4 Application de découverte Leap Motion - Maquettes des interfaces



Figure 14 – Maquette de l'interface de l'écran d'accueil et d'attente d'utilisateur



Figure 15 – Maquette de l'interface du menu principal

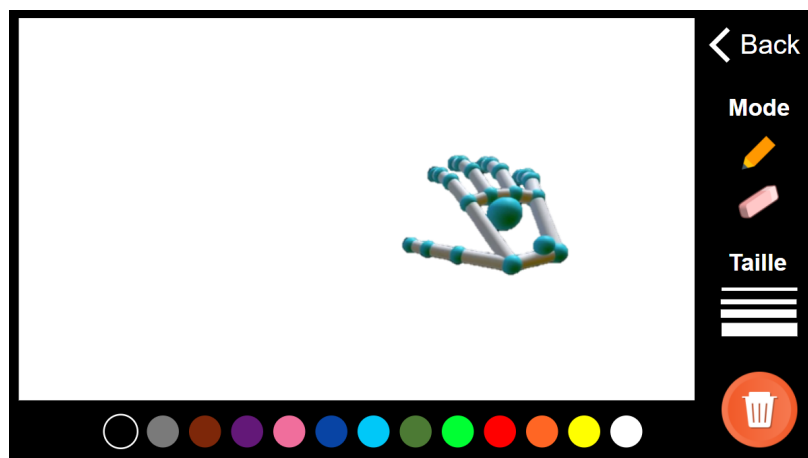


Figure 16 – Maquette de l'interface du module de dessin

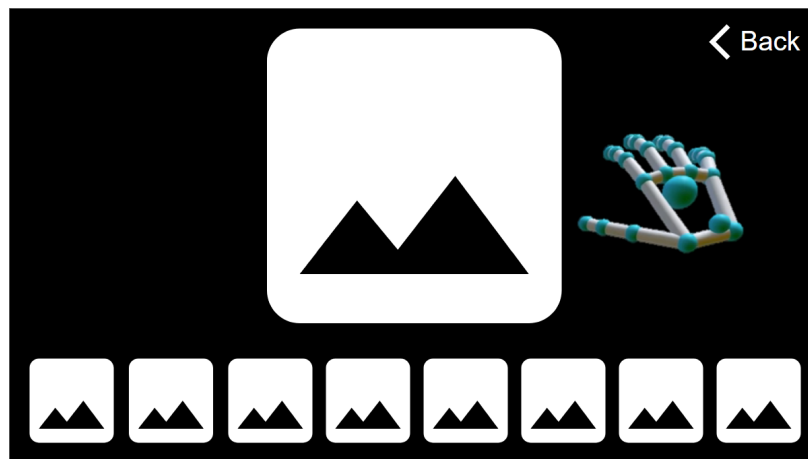


Figure 17 – Maquette de l'interface du module de manipulation d'image

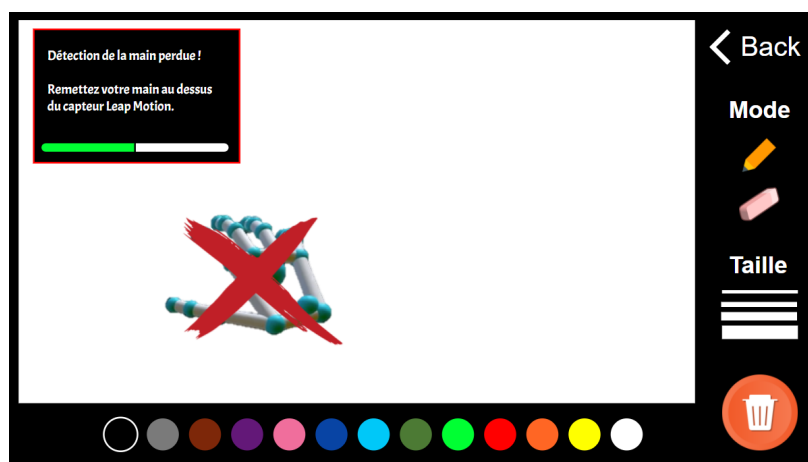


Figure 18 – Maquette d'exemple de la perte de reconnaissance du Leap Motion

3.3 Interfaces logiciel/logiciel

Les applications complèteront de manières régulières des fichiers de log pour aider le monitoring et assurer l'intégrité de celles-ci.

4 Gestion de projet

4.1 Méthodologie agile

La méthodologie agile a été choisie pour ce projet, car elle est la plus adaptée à celui-ci. En effet, nous sommes dans un premier temps face à un projet avec un objectif très concret et l'agile nous permet d'obtenir à la fin de chaque sprint un livrable de qualité que nous pourrions utiliser. De plus, l'agile est parfaitement adapté au changement et il y a de très grandes chances que les attentes du musée ne correspondent pas tout à fait à ce que nous avons mis en place. L'utilisation de la méthodologie agile nous permettra de parer à toute éventualité de changement sur notre application.

4.2 Livrables

Voici les livrables qui ont été établis au début du projet :

- État de l'art et rapports : La partie recherche du projet aboutit à différents documents (cahier de spécifications, guide du développeur, guide d'installation, manuel utilisateur, cahier de tests, état de l'art et analyse du projet). Ces documents ont été fusionnés dans le présent rapport.
- Application de médiation pour la caméra Kinect : Cette application devra être complète, testée et approuvée pour la fin du mois de mars.
- Application de médiation pour le Leap Motion : identique à l'application pour Kinect.

À la fin du projet :

L'intégralité des documents a été rendue dans les temps. L'application de médiation pour Kinect a également complété, mais également testé à la fois tout au long du projet, mais également par des représentants du musée et de la faculté d'Histoire de l'Université de Tours. L'application a également été validée par M. Venturini et M. Serres avant d'être déployée et rendue. En revanche, l'application de médiation pour le Leap Motion a tout simplement été abandonnée par manque de temps. Néanmoins, nous avons pu nous apercevoir lors des différents tests réalisés avec le musée que la prise en main de l'outil est réellement plus intuitive que pour la caméra Kinect. Cela nous rassure donc sur le choix d'avoir privilégié l'application pour Kinect aux dépens de celle-ci.

4.3 Analyse de faisabilité

Il est évident que le projet était ambitieux et que réaliser deux applications différentes en trois mois n'allait pas être chose aisée.

L'application de médiation pour la caméra Kinect a été menée jusqu'à son terme. Un utilisateur peut maintenant se placer dans la zone de contrôle et réaliser le tutoriel pour apprendre la technologie. L'objectif est donc atteint.

En revanche, face aux difficultés et aux jalons prenant du retard, il a rapidement été décidé après les deux premières semaines d'abandonner l'idée de la deuxième application pour le Leap Motion. On peut donc considérer ce projet comme abandonné ou du moins laissé à l'état d'analyse.

4.4 Suivi de projet

L'avancée du projet était presque chaque semaine, selon les disponibilités, par M. Venturini et M. Serres. Le développement de l'application n'a pas été simple et parfois semé d'aléas, aussi bien au niveau de la caméra Kinect (mauvaise détection de l'utilisateur) qu'au niveau du langage vb.net en lui-même, qui est sur certains points assez restrictif notamment pour la gestion des interfaces graphiques.

Enfin, tout au long de notre projet nous avons fait tester l'application à un maximum de personne possible. Ainsi, il ne serait pas étonnant qu'un bon tiers de la promotion ait testé au moins une fois l'application et nous ait transmis à la fois une aide pour la détection des bugs et des erreurs, mais également sur les choses qui étaient bien réalisées et pertinentes.

4.5 Outils

4.5.1 Versionning

Nous avons utilisé Gitlab pour le versionning. L'idée était très simple, à chaque sprint correspondait sa proche branche en dehors de la branche master. Une fois, le sprint terminé et sa phase de test commencé, nous utilisons une nouvelle branche baptisée « TestSprint# ». Lorsque la version de l'application passait avec succès la phase de test, une nouvelle version de l'application voyait le jour. Nous avons donc 5 versions de l'application : 0.1, 0.2, 0.3, 0.4 et 1.0.

4.5.2 Organisation

De manière classique, nous avons utilisé un Trello pour ce projet. Celui-ci était supervisé par M. Venturini. L'objectif de celui-ci était de répartir mes tâches tout en ajoutant via les commentaires toutes difficultés, idée ou observation que je pouvais avoir tout au long du projet. C'était également le bon moyen de faire des rapports hebdomadaires sur le projet.

4.6 Diagramme de Gantt / Semestre 9 - PRD1

Étant donné que nous avons choisi de travailler avec une méthode agile, il faudra donc que le client ait des retours réguliers de ce qui aura été produit. Chaque partie retournée devra donc être suffisamment dense et suffisamment indépendante des autres parties pour avoir à chaque retour une nouvelle version livrable du projet. Dans les points qui suivent, nous allons expliciter les différents sprints qui composeront notre projet.

4.6.1 Sprint 1

La première tâche (premier sprint) consiste à fixer les différents bugs et problèmes existants sur l'application. À la fin de ce sprint, un test sera réalisé pour valider un état stabilisé de notre application. Une application « propre » nous permettra de reprendre le développement et de mettre en place les différentes améliorations.

4.6.2 Sprint 2

Durant cette phase, nous allons améliorer les fonctionnalités de détections des mouvements de l'utilisateur de l'application notamment dans les parties du tutoriel qui sont difficilement faisables en l'état comme c'est le cas pour les parties 4 et 6 (cf. 4. Spécifications fonctionnelles). À la fin de ce sprint, l'application devra être parfaitement fluide et devra répondre de manière satisfaisante aux mouvements de l'utilisateur. Des tests viendront une nouvelle fois attester cette évolution de notre système.

4.6.3 Sprint 3

Ce sprint consiste à développer la fonctionnalité d'apprentissage de la rotation d'une image. Cette partie devra évidemment respecter les contraintes du sprint précédent en termes de fluidité, d'ergonomie et de détection de mouvement. Le tutoriel complet en termes de fonctionnalités utilisateurs pourra alors être testé.

4.6.4 Sprint 4

Cette partie servira à l'ajout des différents modules multimédias comme les images et le son. Il faudra créer et ajouter l'ensemble des images animées ou vidéos, remplacer le retour caméra utilisateur actuellement en forme de squelette bleu, mais aussi enregistrer et ajouter tous les enregistrements sonores. Il faudra également s'assurer que cela n'impacte en rien la fluidité de l'application.

4.6.5 Sprint 5

Ce sprint servira au développement de la reconnaissance de mouvement et à la mise en place des interfaces pour l'application de découverte Leap Motion.

4.6.6 Sprint 6

Développement du module de dessin et de ses fonctionnalités.

4.6.7 Sprint 7

Développement du module de manipulation d'image et de ses fonctionnalités. Ajout d'une banque d'image fixe pour le module.

4.6.8 Diagramme de Gantt

Ce diagramme est à titre indicatif et sera certainement amené à être modifié par la suite. Nous avons ici une phase de développement de l'application de médiation Kinect (en bleu) et une phase de développement de l'application de découverte Leap Motion (en vert). Entre les deux, nous pouvons voir une petite phase verte. Ce moment correspond à la pause pédagogique du mois de février. Durant cette période, une phase de tests complète pourra être menée.

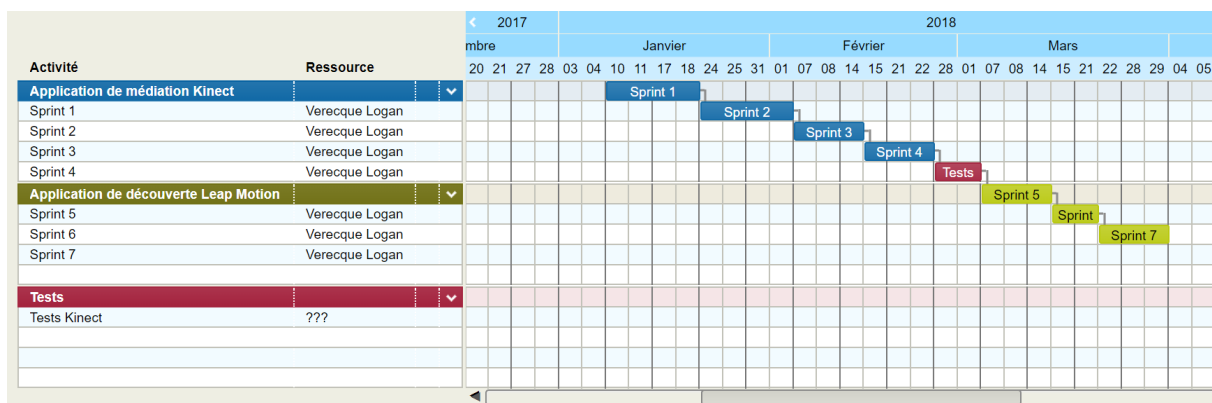


Figure 19 – Diagramme de Gantt

4.7 Diagramme de Gantt / Semestre 10 - PRD2

4.7.1 Sprint 1

La première tâche (premier sprint) consiste à fixer les différents bugs et problèmes existants sur l'application. À la fin de ce sprint, un test sera réalisé pour valider un état stabilisé de notre application. Une application « propre » nous permettra de reprendre le développement et de mettre en place les différentes améliorations.

4.7.2 Sprint 2

Durant cette phase, nous allons améliorer les fonctionnalités de détections des mouvements de l'utilisateur de l'application notamment dans les parties du tutoriel qui sont difficilement faisables en l'état comme c'est le cas pour les parties 4 et 6 (cf. 4. Spécifications fonctionnelles). À la fin de ce sprint, l'application devra être parfaitement fluide et devra répondre de manière satisfaisante aux mouvements de l'utilisateur. Des tests viendront une nouvelle fois attester cette évolution de notre système.

4.7.3 Sprint 3

Ce sprint consiste à développer la fonctionnalité d'apprentissage de la rotation d'une image. Cette partie devra évidemment respecter les contraintes du sprint précédent en termes de fluidité, d'ergonomie et de détection de mouvement. Le tutoriel complet en termes de fonctionnalités utilisateurs pourra alors être testé.

4.7.4 Sprint 4

Cette partie servira à l'ajout des différents modules multimédias comme les images et le son. Il faudra créer et ajouter l'ensemble des images animées ou vidéos, remplacer le retour caméra utilisateur actuellement en forme de squelette bleu, mais aussi enregistrer et ajouter tous les enregistrements sonores. Il faudra également s'assurer que cela n'impacte en rien la fluidité de l'application.

4.7.5 Diagramme de Gantt

Activité	Ressource
Application de médiation Kinect	
Sprint 1	Verecque Logan
Sprint 2	Verecque Logan
Sprint 3	Verecque Logan
Sprint 4	Verecque Logan
Rendu Appli	Verecque Logan
Tests	
Tests Application Globale	Musée
Tests Sprint1	Verecque Logan
Tests Sprint2	Verecque Logan
Tests Sprint3	Verecque Logan
Tests Sprint4	Verecque Logan
Documentation	
Rapport	Verecque Logan
Guide développeur	Verecque Logan
Guide d'installation	Verecque Logan
Manuel Utilisateur	Verecque Logan
Cahier de test	Verecque Logan

Figure 20 – Légende - Diagramme de Gantt

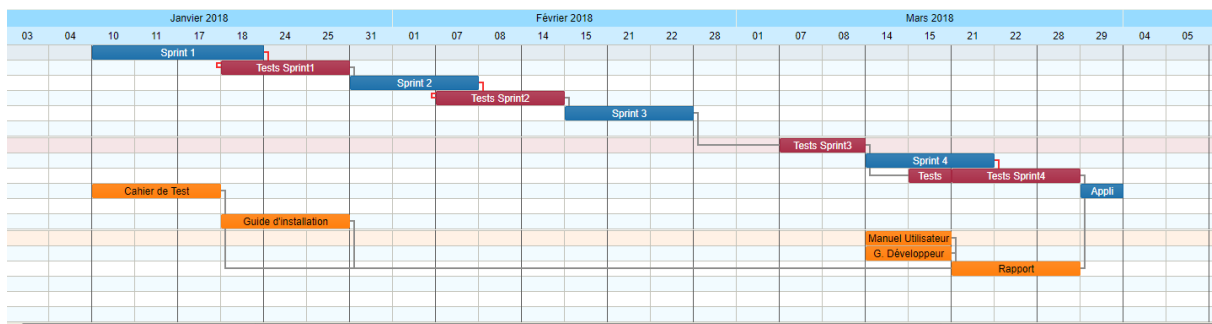


Figure 21 – Diagramme de Gantt S10

5 Guide d'installation

5.1 Matériel nécessaire

Pour utiliser l'application de médiation Kinect, nous avons besoin au minimum d'un ordinateur et d'une caméra Kinect avec l'adaptateur pour Windows. L'application peut se lancer directement sur l'écran de l'ordinateur, mais il est préférable d'utiliser un grand écran de télévision ou un projecteur. En effet, l'utilisateur est debout face à la caméra à quelques mètres de l'écran. Il est donc agréable d'avoir un écran suffisamment grand pour apprécier l'application. Le schéma ci-dessous décrit la façon de mettre en place le système.

À noter qu'il faut un ordinateur assez puissant avec au minimum 8 Go de mémoire vive, mais également obligatoirement au moins un port USB3 pour brancher la caméra Kinect sur la machine. Enfin, aucune connexion internet n'est nécessaire pour l'installation ou l'utilisation de l'application.

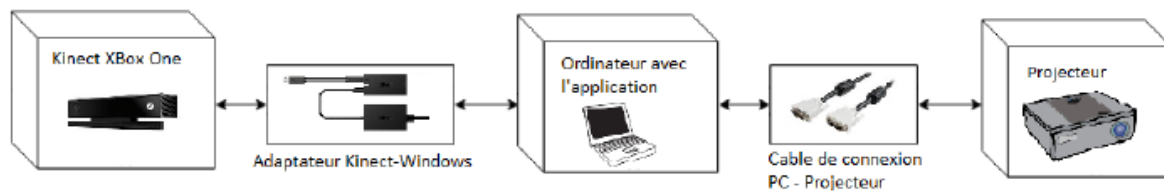


Figure 22 – Architecture de l'installation

5.2 Outils de développement

Il est nécessaire d'avoir la dernière version du SDK Kinect et un environnement de développement pour VB.NET. Celui utilisé jusqu'ici est Visual Studio 2013. Il est normalement possible d'utiliser une version plus récente de Visual Studio (la dernière étant la 2017), mais cela n'a pas été testé pour notre projet donc nous ne connaissons pas les problèmes éventuels de compatibilités. Dans le cas où vous voudriez créer un nouveau projet inspiré de celui-ci sur Visual Studio, ou sur un autre logiciel, il ne faut pas oublier d'ajouter les références importées pour le bon fonctionnement de l'application. Elles se situent dans le sous-onglet « Ajouter une référence » de l'onglet « Projet ». Ces références permettent d'utiliser les bibliothèques du SDK Kinect. Concernant les traitements d'image, il vous faut un logiciel comme Gimp qui est gratuit ou Adobe Photoshop qui est lui payant.

6 Guide du développeur

6.1 Généralités

6.1.1 Langage, librairies et convention de nommage

Le langage utilisé pour l'application est le Visual Basic avec le Framework .net. On parlera alors plutôt de VB.net. La librairie MSDN a également été utilisée, notamment pour pouvoir modifier la position du curseur de la souris. En effet, par défaut, VB.net n'autorise pas la modification de certains paramètres de l'ordinateur par sécurité.

La convention de nommage utilisée est la suivante :

Variables : minuscule pour premier mot suivi de majuscule première lettre du 2e mot. Exemple : `maVariable`

Fonctions : Première lettre majuscule pour chaque mot, le reste en minuscule Exemple : `maFonction()`

Seules exceptions :

- Les fonctions situées dans `UserInterface` sont des événements. Le nom de la fonction est alors la variable utilisée pour l'événement suivie d'un Underscore puis du nom de l'événement. Par exemple, si un bouton appelé « button » doit s'activer au survol de la souris, le nom de la fonction sera `button_OnMouseOver()`.
- Certaines variables spécifiant des coordonnées sont composées d'une seule lettre correspondant à la coordonnée. Par exemple, la coordonnée x est représentée par une variable appelée `X`, pour plus de simplicité.

6.1.2 Documentation

Le choix du langage a fortement réduit les possibilités de documentation. Néanmoins, une documentation IntelliSense a été générée. Cette documentation se présente sous la forme d'un fichier XML qui résume et présente l'ensemble des méthodes de l'application de médiation Kinect. Mais le réel intérêt de celle-ci est la documentation dynamique sous Visual Studio. En effet, lorsque cette documentation est en place celle-ci est disponible à tout moment pendant le développement. Il suffit d'appeler la méthode implémentée que l'on souhaite utiliser et les informations que nous avons rédigées apparaîtront automatiquement.

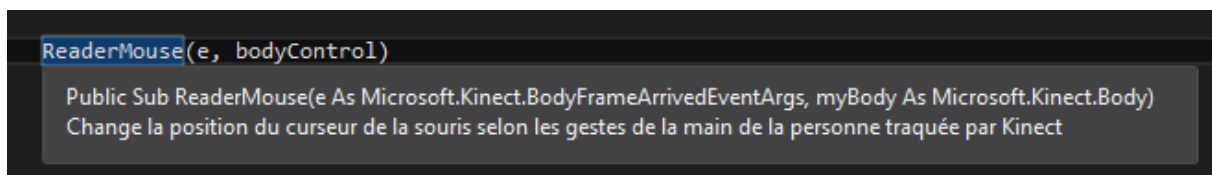


Figure 23 – Exemple documentation IntelliSense dynamique

De plus, dans le doute où cette documentation ne serait pas suffisante le code a été commenté plus que de raison. Chaque méthode, chaque variable est extrêmement détaillée pour ne pas se perdre dans la modélisation modulaire de l'application.

Enfin, chaque module a été découpé en région comme le permet le framework .net. L'idée est ici de séparer le code d'une même classe afin de gagner en clarté. Nous avons ici séparé chaque module en essayant d'être le plus pratiques possible. Par exemple pour le module Animation nous avons cinq régions : « Variables », « Conditions », « Displays », « Timers », « Others ». Dans cet exemple, toutes les méthodes du module concernant l'utilisation des timers sont rangées dans la région « Timers ». Ainsi, il est plus facile de s'y retrouver et de se repérer.

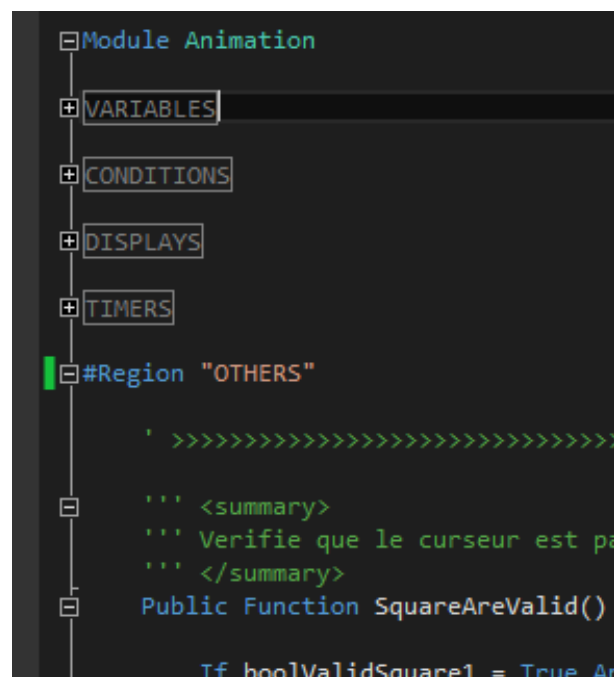


Figure 24 – Regions module Animation

6.2 Structure du système

Avant toute chose, il est important d'indiquer que la structure de l'application est basée sur un diagramme de modules et non de classes dans une perspective d'indépendance entre chaque module.

Évidemment, le concept permet une meilleure indépendance du code par rapport au matériel physique et permet de s'y retrouver plus facilement. Néanmoins, cet aspect augmente la redondance au niveau du code.

Voici ci-dessous le diagramme de modules de l'application de médiation Kinect.

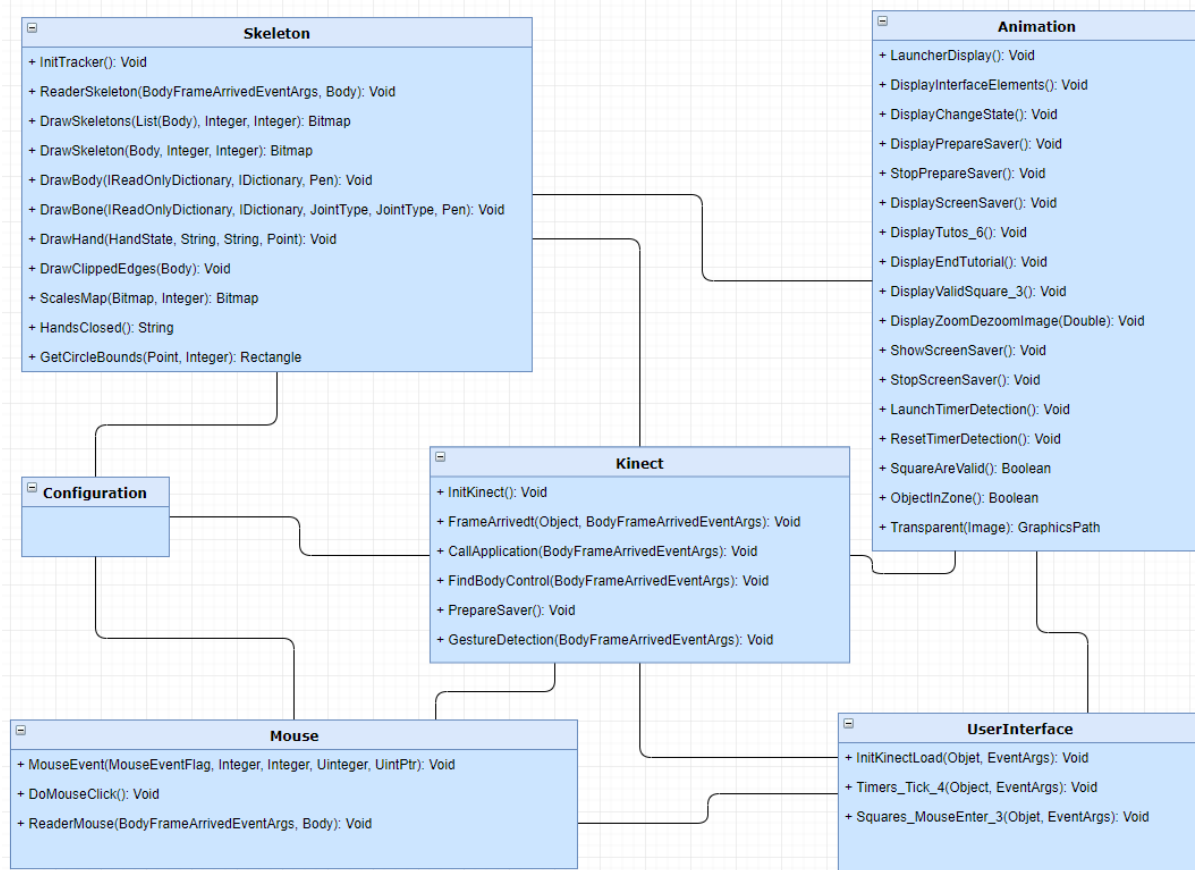


Figure 25 – Diagramme de classe

Le diagramme a été simplifié au niveau de son écriture sur un aspect particulier. En effet, les méthodes terminant par un numéro indiquent le fusionnement de plusieurs méthodes pour gagner en place sur le diagramme. Par exemple, dans le module UserInterface, quatre timers sont utilisés. Leurs fonctions ont été fusionnées en une seule sur le diagramme afin d'éviter de l'alourdir.

Nous allons maintenant détailler l'utilité de chaque module pas à pas :

UserInterface

Comme son nom l'indique, il s'agit de l'interface utilisateur. Au sein de ce module, on trouvera ce que l'on pourra appeler la fonction main du programme qui est en fait la fonction d'initialisation du Kinect et la première méthode lancée lors du lancement de l'application. Nous trouverons également dans ce module les actions déclenchées par les différents timers présents

dans l'application et enfin trois méthodes servant à l'étape numéro deux du tutoriel pour le déplacement du curseur de la souris sur les cibles.

Kinect

Ce module représente le coeur et le moteur de l'application de médiation Kinect. Il sert de pivot entre les différents modules. Il est lancé à partir de `UserInterface` au démarrage de l'application et ne s'arrête jamais. Ce module gère les instructions à effectuer à chaque frame envoyé par la caméra Kinect.

Nous trouverons donc à l'intérieur de ce module les méthodes de gestions des frames, mais surtout la méthode de détection des mouvements. C'est au sein de cette méthode que l'on vérifiera à chaque étape que l'utilisateur réussit à faire le geste demandé correctement.

Mouse

Ce module contient toutes les fonctions de contrôle du curseur de la souris. Nous utilisons le curseur de la souris pour les étapes de déplacements d'objets, à chaque fois que nous devons attraper un objet et bien évidemment lorsque l'on doit contrôler le curseur de la souris à distance. Par exemple, l'étape de déplacement d'un objet se résume à un simple glisser-déposer avec la souris cachée derrière les interactions gestuelles avec la caméra Kinect.

Skeleton

Ce module permet la création du squelette de l'utilisateur en cours. Il est capable de créer plusieurs squelettes (jusqu'à six personnes peuvent être repérées par Kinect), mais pour notre application, nous avons limité la création du squelette à la personne qui dirige. A chaque déplacement de l'utilisateur, le squelette va être mis à jour. Le but de celui-ci est évidemment de montrer à l'utilisateur ses mouvements en direct.

Animation

Pour éviter de surcharger `UserInterface` ou `Kinect`, ce module a été créé. Il gère tous les éléments graphiques et audio à effectuer selon les actions de l'utilisateur. Il comprend toutes les méthodes qui sont appelées par la classe `Kinect` selon chaque frame reçu du dispositif Kinect. L'idée était de séparer de manière concrète l'environnement matériel Kinect du contenu multimédia. C'est donc ici que l'on trouvera la méthode de gestions des textes, des images et de l'audio pour les étapes du tutoriel.

Configuration

Ce module comprend toutes les variables modifiables par l'utilisateur, comme les liens vers les fichiers audio ou encore la « douceur » du « curseur de la souris ». Il permet un accès plus rapide et simple à ces variables, si un non-informaticien souhaite juste modifier quelques configurations.

De manière générale, les modules qui sont amenés à beaucoup être modifiés sont les modules `UserInterface` pour le design et la gestion des éléments, `Animation` pour les méthodes associées à l'interface, et enfin `Kinect` pour les appels aux méthodes du module `Animation`.

7 Manuel Utilisateur

7.1 Mise en place

Pour toute information concernant la mise en place de la caméra Kinect et le branchement des différents périphériques, veuillez vous référer au guide d'installation.

Lorsque l'application est lancée, vous vous retrouvez face à l'écran d'accueil. Celui-ci est alors en attente jusqu'à ce qu'une personne vienne prendre le contrôle de la caméra Kinect. Pour cela, veuillez vous mettre en place face à la caméra, à deux mètres minimum et trois mètres maximum de celle-ci.

Il est fortement conseillé d'utiliser l'application en étant debout face à la caméra. Bien qu'il soit totalement possible d'utiliser le logiciel en étant assis, certains mouvements se trouveront plus difficiles à réaliser dans cette position.

7.2 Généralités

Une étape du tutoriel se décompose en deux phases :

- Une première phase où l'application vous explique le mouvement que vous devez réaliser et comment le réaliser.
- Une deuxième phase vous expliquant les instructions à réaliser pour passer à l'étape suivante du tutoriel.

Chaque étape du tutoriel s'accompagne d'une description textuelle et auditive pour aider à la compréhension des mouvements demandés.

Les mouvements que vous réalisez sont représentés par le squelette orange à l'écran.

Au moment où vous réussissez une étape du tutoriel, l'étape suivante se lance automatiquement sans que vous ayez besoin de faire quoi que ce soit.

7.3 Étape 1 — Bouger

Lorsque vous intégrez la zone de contrôle, la première étape du tutoriel se lance automatiquement.

Lors de cette première étape, vous êtes invités à comprendre que vous pouvez bouger votre corps, les mouvements seront alors représentés par le squelette à l'écran.

Pour passer à l'étape suivante du tutoriel, vous devrez lever la main de votre choix au-dessus de votre tête.

7.4 Étape 2 — Déplacer la souris

Lors de cette deuxième étape, l'application vous présentera le contrôle du curseur de la souris grâce à la caméra Kinect.

Une fois l'explication du mouvement passé, vous devrez déplacer le curseur de la souris sur les différentes cibles affichées à l'écran. Pour réaliser cela, vous devrez tendre une seule et unique main devant vous face à la caméra et déplacer celle-ci afin de déplacer le curseur de la souris.

7.5 Étape 3 — Serrer les poings

Cette troisième étape vous présentera le concept de fermeture des poings. Vous devrez ici fermer les deux poings simultanément cinq fois de suite. Ce concept vous permettra, par la suite, d'attraper des objets pour pouvoir les déplacer.

À noter qu'un indicateur est présent pour vous indiquer si vos poings sont fermés ou non. En effet, lorsque les paumes de mains sont ouvertes un cercle vert est indiqué autour d'elles sur le squelette présent à l'écran. À l'inverse, si les poings sont fermés alors ce cercle est rouge.

Dans le cas où il n'y aurait ni l'un ni l'autre, cela signifie que la caméra Kinect n'arrive plus à détecter les mains de l'utilisateur. Afin de prévenir au maximum cela, veuillez montrer à la caméra les paumes de vos mains fermées plutôt que les poings comme indiqué ci-dessous.

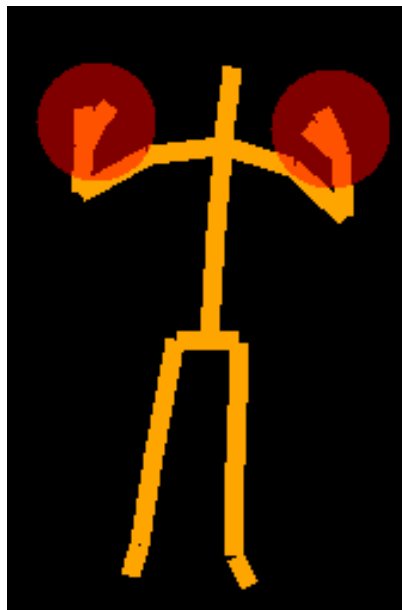


Figure 26 – Squelette de l'utilisateur

7.6 Étape 4 — Déplacer un objet

Cette quatrième étape vous initiera au déplacement d'un objet. Vous devrez ici attraper un objet en fermant les poings puis, tout en les laissant fermés, déplacer vos poings pour déplacer l'objet jusqu'à une zone bien précise. L'idée est ici de déplacer une statue affichée à l'écran pour la poser sur son socle se trouvant juste à côté.

Cette étape a été pensée pour ne pas être contraignante, ainsi, la zone à atteindre est suffisamment grande et permissive pour ne pas contraindre ou décourager l'utilisateur.

7.7 Étape 5 — Agrandissement/Réduction

Cette cinquième étape vous apprendra à agrandir ou réduire la taille d'un objet. Vous serez alors face à une image d'une statue avec une taille très réduite que vous devrez agrandir jusqu'à ce qu'elle remplit complètement le cadre dans lequel elle est contenue.

Pour réaliser cela, il faudra attraper l'objet de la même manière que pour l'étape 4 et déplacer vos poings vers vous si vous souhaitez agrandir ou vers la caméra Kinect si vous souhaitez réduire la taille de l'image.



Figure 27 – L'utilisateur déplace la statue



Figure 28 – Zone de réception de la statue

7.8 Étape 6 — Tourner un objet

Cette sixième et dernière étape vous présentera la manière de tourner un objet 3D dans l'espace. Vous devez pour cela fermer vos poings pour attraper l'objet et déplacer vos poings comme si vous tourniez un volant horizontal comme l'illustre l'image ci-dessous. L'objectif de cette étape est de tourner à loisir l'image 3D de la statue affichée à l'écran. Une barre de progression apparaîtra en bas à droite de l'écran et se remplira au fur et à mesure que l'utilisateur tourne la statue. Une fois la barre de progression pleine, l'étape est réussie et le tutoriel prend fin. N'oubliez jamais, une nouvelle fois, que les paumes de vos mains doivent faire face à la caméra lors de vos mouvements. Il ne faut pas hésiter à ouvrir les mains, se repositionner et de nouveau attraper l'objet pour effectuer le geste, un peu comme si vous tourniez le volant par à-coup.



Figure 29 – Exemple rotation (vue de côté)



Figure 30 – Exemple rotation (vue de face)

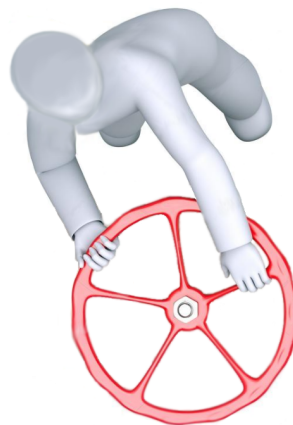


Figure 31 – Exemple rotation (vue de dessus)

7.9 Fin du tutoriel

L'écran de fin du tutoriel s'affiche lorsque toutes les étapes ont été réalisées avec succès. Celui-ci félicite l'utilisateur pour avoir effectué le tutoriel et l'invite à la sortir de la zone de contrôle pour laisser la place à l'utilisateur suivant. L'utilisateur dispose de vingt secondes pour sortir de celle-ci. Si le temps arrive à son terme, le tutoriel retourne sur l'écran d'accueil, mais comme l'utilisateur est toujours dans la zone de contrôle la première étape du tutoriel est lancée automatiquement. Si l'utilisateur sort dans le temps imparti l'application retourne sur l'écran d'accueil et attend de nouveau qu'un utilisateur entre dans la zone de contrôle.

7.10 Sortie de la zone de contrôle

Si vous sortez de la zone de contrôle de la caméra Kinect, il peut se passer deux phénomènes bien distincts.

- Dans le cas où vous étiez en cours d’une étape du tutoriel, l’application vous invite à poursuivre celle-ci en revenant dans la zone de contrôle. Un écran affichant que vous êtes sortis de la zone apparaît pour vous prévenir que vous disposez alors de vingt secondes pour rejoindre de nouveau l’emplacement utilisateur. Si vous revenez dans les vingt secondes, l’application reprend là où vous l’avez laissée et vous pouvez continuer. Si les vingt secondes sont écoulées, l’application retourne automatiquement sur l’écran d’accueil et retourne en état d’attente d’un nouvel utilisateur.
- Dans le cas où vous avez terminé le tutoriel, l’application retourne automatique sur l’écran d’accueil en attente d’un nouvel utilisateur.

8 Cahier de tests

L’idée était, à la base, de mettre en place un système d’intégration continu sur la plateforme Gitlab pour assurer la partie test tout au long du projet et éviter la régression du code. Néanmoins, cela n’a pas pu se faire pour la simple et bonne raison que cela n’était pas impossible, mais extrêmement compliqué avec le langage vb.net.

Une autre idée fut ensuite d’utiliser l’outil SonarQube pour mesurer la qualité du code source également en continu, mais après plusieurs essais je n’ai pas réussi à le mettre en place, d’autant plus que SonarQube demandait un dépôt Github et n’acceptait pas les dépôts Gitlab.

Finalement, et pour faire face à ces désillusions, j’ai tenté de tout mettre en œuvre pour réaliser des tests cohérents et efficaces pour le projet. J’ai donc procédé en plusieurs étapes. Tout d’abord, j’ai réalisé un ensemble de scénarios de test pour l’intégralité du projet puis dans un deuxième temps à chaque période de tests à la fin de chaque sprint j’ai réalisé l’intégralité des scénarios possibles tout en reportant chaque résultat dans un tableau afin de valider le sprint en cours et avoir une vision générale sur les défauts du code.

8.1 Scénarios de test

Nous allons maintenant détailler l’ensemble des scénarios de tests qui sont réalisés à chaque phase de test.

Lancement de l’application

- *Situation* : L’application se lance.
- *Résultat attendu* : Les éléments graphiques (textes et audio) de l’écran d’accueil se lancent correctement.

Écran d’accueil

- *Situation* : Un utilisateur entre dans la zone de contrôle.
- *Résultat attendu* : L’étape 1 du tutoriel se lance.

Tutoriel étape 1

- *Situation* : L'étape 1 du tutoriel s'affiche.
- *Résultat attendu* : Les éléments graphiques (squelette, textes et audio) se lancent correctement.
- *Situation* : L'explication du geste est terminée.
- *Résultat attendu* : L'application lance l'épreuve à effectuer (texte et audio).
- *Situation* : L'utilisateur effectue le geste de lever une main.
- *Résultat attendu* : L'application comprend le geste et lance l'étape 2.
- *Situation* : L'utilisateur sort de la zone de contrôle.
- *Résultat attendu* : L'application lance l'affichage de sortie de zone.

Tutoriel étape 2

- *Situation* : L'étape 2 du tutoriel s'affiche.
- *Résultat attendu* : Les éléments graphiques (squelettes, textes, audio, affichage de la main dominante) se lancent correctement.
- *Situation* : L'explication du geste est terminée.
- *Résultat attendu* : L'application lance l'épreuve à effectuer (texte, audio et cibles).
- *Situation* : L'utilisateur effectue le geste de déplacer le curseur.
- *Résultat attendu* : L'application comprend le geste et déplace le curseur de la souris en conséquence.
- *Situation* : L'utilisateur touche une cible avec le curseur de la souris.
- *Résultat attendu* : La cible touchée devient verte signifiant sa validation.
- *Situation* : L'utilisateur valide les trois cibles.
- *Résultat attendu* : L'application affiche l'étape 3 du tutoriel.
- *Situation* : L'utilisateur sort de la zone de contrôle.
- *Résultat attendu* : L'application lance l'affichage de sortie de zone.

Tutoriel étape 3

- *Situation* : L'étape 3 du tutoriel s'affiche.
- *Résultat attendu* : Les éléments graphiques (squelettes, textes, audio) se lancent correctement.
- *Situation* : L'explication du geste est terminée.
- *Résultat attendu* : L'application lance l'épreuve à effectuer (texte, audio et compteur de fermeture des poings) et le compteur de fermeture des poings a été réinitialisé.
- *Situation* : L'utilisateur effectue le geste et ferme ses poings.
- *Résultat attendu* : L'application comprend le geste, affiche un cercle rouge et incrémente le compteur.
- *Situation* : Le compteur de fermeture des poings arrive à 5.
- *Résultat attendu* : L'application affiche l'étape 4 du tutoriel.
- *Situation* : L'utilisateur sort de la zone de contrôle.
- *Résultat attendu* : L'application lance l'affichage de sortie de zone.

Tutoriel étape 4

- *Situation* : L'étape 4 du tutoriel s'affiche.
- *Résultat attendu* : Les éléments graphiques (squelettes, textes, audio et l'image de la statue) se lancent correctement.

- *Situation* : L'explication du geste est terminée.
- *Résultat attendu* : L'application lance l'épreuve à effectuer (texte, audio et socle).
- *Situation* : L'utilisateur effectue le geste.
- *Résultat attendu* : L'application comprend le geste, et déplace la statue dans l'espace.
- *Situation* : La statue atteint la zone du socle.
- *Résultat attendu* : La zone de réception est au bon endroit, l'application comprend la validation de l'étape et affiche l'étape 5 du tutoriel.
- *Situation* : L'utilisateur sort de la zone de contrôle.
- *Résultat attendu* : L'application lance l'affichage de sortie de zone.

Tutoriel étape 5

- *Situation* : L'étape 5 du tutoriel s'affiche.
- *Résultat attendu* : Les éléments graphiques (squelettes, textes, audio, l'image de la statue et le cadre) se lancent correctement.
- *Situation* : L'explication du geste est terminée.
- *Résultat attendu* : L'application lance l'épreuve à effectuer (texte, audio).
- *Situation* : L'utilisateur effectue le geste de ramener ses mains vers lui.
- *Résultat attendu* : La taille de l'image de la statue s'agrandit.
- *Situation* : L'utilisateur effectue le geste d'approcher ses mains vers la caméra Kinect.
- *Résultat attendu* : La taille de l'image de la statue se réduit.
- *Situation* : L'image de la statue remplit entièrement le cadre.
- *Résultat attendu* : L'application affiche l'étape 6 du tutoriel.
- *Situation* : L'utilisateur sort de la zone de contrôle.
- *Résultat attendu* : L'application lance l'affichage de sortie de zone.

Tutoriel étape 6

- *Situation* : L'étape 6 du tutoriel s'affiche.
- *Résultat attendu* : Les éléments graphiques (squelettes, textes, audio, l'image de la statue et barre de progression) se lancent correctement.
- *Situation* : L'explication du geste est terminée.
- *Résultat attendu* : L'application lance l'épreuve à effectuer (texte, audio).
- *Situation* : L'utilisateur effectue le geste de rotation.
- *Résultat attendu* : La barre de progression se remplit et la statue se tourne en fonction de la rotation du mouvement.
- *Situation* : La barre de progression est pleine.
- *Résultat attendu* : L'application affiche l'écran de fin du tutoriel
- *Situation* : L'utilisateur sort de la zone de contrôle.
- *Résultat attendu* : L'application lance l'affichage de sortie de zone.

Écran de fin du tutoriel

- *Situation* : L'écran de fin du tutoriel s'affiche.
- *Résultat attendu* : Les éléments graphiques (textes, audio, l'image de félicitation et la barre de progression) se lancent correctement. Le décompte de la barre de progression de vingt secondes se lance et la barre de progression se remplit chaque seconde.
- *Situation* : Le temps de sortie de la zone de vingt secondes est écoulé.
- *Résultat attendu* : La barre de progression est pleine et l'étape 1 du tutoriel s'affiche.

- *Situation* : L'utilisateur sort de la zone de contrôle.
- *Résultat attendu* : L'application lance l'affichage de l'écran d'accueil.

Écran de sortie de zone

- *Situation* : L'écran de sortie de zone s'affiche.
- *Résultat attendu* : Les éléments graphiques (textes, audio, l'image des empreintes de pas et la barre de progression) se lancent correctement. Le décompte de la barre de progression de vingt secondes se lance et la barre de progression se remplit chaque seconde.
- *Situation* : Le temps de sortie de la zone de vingt secondes est écoulé.
- *Résultat attendu* : L'application lance l'affichage de l'écran d'accueil.
- *Situation* : L'utilisateur entre dans la zone de contrôle.
- *Résultat attendu* : L'application lance l'affichage de l'étape en cours du tutoriel dans l'état exact où elle était avant que l'utilisateur ne sorte de la zone.

8.2 Résultats des tests

Cette partie présente les résultats des tests effectués lors des différents scénarios présentés précédemment. Pour chaque phase de tests de chaque sprint, nous avons effectué l'ensemble des tests possibles (sauf évidemment les tests ayant lieu sur des éléments non implémentés) afin de, bien entendu, vérifier le bon fonctionnement de l'application, mais également afin de vérifier la non-régression du test.

Le code couleur est le suivant :

Test réussi	Test réussi, mais problème détecté	Test non réussi	Élément non implémenté
-------------	------------------------------------	-----------------	------------------------

Figure 32 – Code couleur

* La mise en page LaTeX a rendu difficile la lecture des différents tableaux mais ils sont disponibles dans le cahier de test en dehors du rapport.

Nous pouvons observer que les tests du dernier sprint ont abouti à trois scénarios avec un code couleur orange. Ces trois scénarios sont particuliers et leurs cas ont été étudiés notamment dans le manuel utilisateur.

Test	Résultat Attendu	Résultat Obtenu	Commentaires
Lancement Application	Les éléments graphiques (textes et audio) de l'écran d'accueil se lancent correctement.		
Écran d'accueil	L'étape 1 du tutoriel se lance		
Tutoriel Étape 1 (1)	les éléments graphiques (squelette, textes et audio) se lancent correctement		
Tutoriel Étape 1 (2)	l'application lance l'épreuve à effectuer (texte et audio)		
Tutoriel Étape 1 (3)	l'application comprend le geste et lance l'étape 2		
Tutoriel Étape 1 (4)	l'application lance l'affichage de sortie de zone		
Tutoriel Étape 2 (1)	les éléments graphiques (squelettes, textes, audio, affichage de la main dominante et barre de progression) se lancent correctement		Barre de progression au sprint 4
Tutoriel Étape 2 (2)	l'application lance l'épreuve à effectuer (texte, audio et cibles)		Les cibles s'affichent, mais lors d'un deuxième passage sur le tutoriel ne sont plus à la bonne taille

Figure 33 – Résultat Test Sprint 1 (1)

Tutoriel Étape 2 (3)	l'application comprend le geste et déplace le curseur de la souris en conséquence		
Tutoriel Étape 2 (4)	la cible touchée devient verte signifiant sa validation		
Tutoriel Étape 2 (5)	l'application affiche l'étape 3 du tutoriel		
Tutoriel Étape 2 (6)	l'application lance l'affichage de sortie de zone		
Tutoriel Étape 3 (1)	les éléments graphiques (squelettes, textes, audio) se lancent correctement		
Tutoriel Étape 3 (2)	l'application lance l'épreuve à effectuer (texte, audio et compteur de fermeture des poings) et le compteur de fermeture des poings a été réinitialisé		
Tutoriel Étape 3 (3)	l'application comprend le geste, affiche un cercle rouge et incrémente le compteur		
Tutoriel Étape 3 (4)	l'application affiche l'étape 4 du tutoriel		
Tutoriel Étape 3 (5)	l'application lance l'affichage de sortie de zone		
Tutoriel Étape 4 (1)	les éléments graphiques (squelettes, textes, audio et l'image de la statue) se lancent correctement		La statue change de position lors d'un 2e passage sur le tuto
Tutoriel Étape 4 (2)	l'application lance l'épreuve à effectuer (texte, audio et socle)		
Tutoriel Étape 4 (3)	l'application comprend le geste, et déplace la statue dans l'espace		La statue se déplace, mais difficilement
Tutoriel Étape 4 (4)	la zone de réception est au bon endroit, l'application comprend la validation de l'étape et affiche l'étape 5 du tutoriel		La zone de réception est décalée par rapport au socle

Figure 34 – Résultat Test Sprint 1 (2)

Tutoriel Étape 4 (5)	l'application lance l'affichage de sortie de zone		
Tutoriel Étape 5 (1)	les éléments graphiques (squelettes, textes, audio, l'image de la statue et le cadre) se lancent correctement		
Tutoriel Étape 5 (2)	l'application lance l'épreuve à effectuer (texte, audio)		
Tutoriel Étape 5 (3)	la taille de l'image de la statue s'agrandit		La vitesse de changement de taille est trop lente
Tutoriel Étape 5 (4)	la taille de l'image de la statue se réduit		La vitesse de changement de taille est trop lente
Tutoriel Étape 5 (5)	l'application affiche l'étape 6 du tutoriel		L'image de la statue sort du cadre alors qu'elle devrait être bloquée dedans
Tutoriel Étape 5 (6)	l'application lance l'affichage de sortie de zone		
Tutoriel Étape 6 (1)	les éléments graphiques (squelettes, textes, audio, l'image de la statue et barre de progression) se lancent correctement		
Tutoriel Étape 6 (2)	l'application lance l'épreuve à effectuer (texte, audio)		
Tutoriel Étape 6 (3)	la barre de progression se remplit et la statue se tourne en fonction de la rotation du mouvement		
Tutoriel Étape 6 (4)	l'application affiche l'écran de fin du tutoriel		
Tutoriel Étape 6 (5)	l'application lance l'affichage de sortie de zone		

Figure 35 – Résultat Test Sprint 1 (3)

Écran fin tuto (1)	les éléments graphiques (textes, audio, l'image de félicitation et la barre de progression) se lancent correctement. Le décompte de la barre de progression de vingt secondes se lance et la barre de progression se remplit chaque seconde.		Barre de progression au Sprint 4
Écran fin tuto (2)	la barre de progression est pleine et l'étape 1 du tutoriel s'affiche		
Écran fin tuto (3)	l'application lance l'affichage de l'écran d'accueil		
Écran Hors Zone (1)	les éléments graphiques (textes, audio, l'image des empreintes de pas et la barre de progression) se lancent correctement. Le décompte de la barre de progression de vingt secondes se lance et la barre de progression se remplit chaque seconde.		Si on vient de l'étape 2 alors l'affichage du curseur de la souris est toujours présent. De même l'audio venant de l'étape en cours ne s'arrête pas.
Écran Hors Zone (2)	l'application lance l'affichage de l'écran d'accueil		Pour le moment au bout de 3 secondes au lieu de 20
Écran Hors Zone (3)	l'application lance l'affichage de l'étape en cours du tutoriel dans l'état exact où elle était avant que l'utilisateur ne sorte de la zone.		Les modifications sur les positions et tailles des images ne sont pas prises en compte. Tout est réinitialisé.

Figure 36 – Résultat Test Sprint 1 (4)

Test	Résultat Attendu	Résultat Obtenu	Commentaires
Lancement Application	Les éléments graphiques (textes et audio) de l'écran d'accueil se lancent correctement.		
Écran d'accueil	L'étape 1 du tutoriel se lance		
Tutoriel Étape 1 (1)	les éléments graphiques (squelette, textes et audio) se lancent correctement		
Tutoriel Étape 1 (2)	l'application lance l'épreuve à effectuer (texte et audio)		
Tutoriel Étape 1 (3)	l'application comprend le geste et lance l'étape 2		
Tutoriel Étape 1 (4)	l'application lance l'affichage de sortie de zone		
Tutoriel Étape 2 (1)	les éléments graphiques (squelettes, textes, audio, affichage de la main dominante et barre de progression) se lancent correctement		Barre de progression au sprint 4
Tutoriel Étape 2 (2)	l'application lance l'épreuve à effectuer (texte, audio et cibles)		Les cibles s'affichent, mais lors d'un deuxième passage sur le tutoriel ne sont plus à la bonne taille
Tutoriel Étape 2 (3)	l'application comprend le geste et déplace le curseur de la souris en conséquence		
Tutoriel Étape 2 (4)	la cible touchée devient verte signifiant sa validation		
Tutoriel Étape 2 (5)	l'application affiche l'étape 3 du tutoriel		
Tutoriel Étape 2 (6)	l'application lance l'affichage de sortie de zone		
Tutoriel Étape 3 (1)	les éléments graphiques (squelettes, textes, audio) se lancent correctement		

Figure 37 – Résultat Test Sprint 2 (1)

Tutoriel Étape 3 (2)	l'application lance l'épreuve à effectuer (texte, audio et compteur de fermeture des poings) et le compteur de fermeture des poings a été réinitialisé		
Tutoriel Étape 3 (3)	l'application comprend le geste, affiche un cercle rouge et incrémente le compteur		
Tutoriel Étape 3 (4)	l'application affiche l'étape 4 du tutoriel		
Tutoriel Étape 3 (5)	l'application lance l'affichage de sortie de zone		
Tutoriel Étape 4 (1)	les éléments graphiques (squelettes, textes, audio et l'image de la statue) se lancent correctement		La statue change de position lors d'un 2e passage sur le tuto
Tutoriel Étape 4 (2)	l'application lance l'épreuve à effectuer (texte, audio et socle)		
Tutoriel Étape 4 (3)	l'application comprend le geste, et déplace la statue dans l'espace		Le déplacement est plus fluide et mieux interpréter. Information sur ce point dans le manuel utilisateur
Tutoriel Étape 4 (4)	la zone de réception est au bon endroit, l'application comprend la validation de l'étape et affiche l'étape 5 du tutoriel		La zone de réception est décalée par rapport au socle
Tutoriel Étape 4 (5)	l'application lance l'affichage de sortie de zone		
Tutoriel Étape 5 (1)	les éléments graphiques (squelettes, textes, audio, l'image de la statue et le cadre) se lancent correctement		
Tutoriel Étape 5 (2)	l'application lance l'épreuve à effectuer (texte, audio)		
Tutoriel Étape 5 (3)	la taille de l'image de la statue s'agrandit		La vitesse de changement de taille est trop lente

Figure 38 – Résultat Test Sprint 2 (2)

Tutoriel Étape 5 (4)	la taille de l'image de la statue se réduit		La vitesse de changement de taille est trop lente
Tutoriel Étape 5 (5)	l'application affiche l'étape 6 du tutoriel		L'image de la statue sort du cadre alors qu'elle devrait être bloquée dedans
Tutoriel Étape 5 (6)	l'application lance l'affichage de sortie de zone		
Tutoriel Étape 6 (1)	les éléments graphiques (squelettes, textes, audio, l'image de la statue et barre de progression) se lancent correctement		
Tutoriel Étape 6 (2)	l'application lance l'épreuve à effectuer (texte, audio)		
Tutoriel Étape 6 (3)	la barre de progression se remplit et la statue se tourne en fonction de la rotation du mouvement		
Tutoriel Étape 6 (4)	l'application affiche l'écran de fin du tutoriel		
Tutoriel Étape 6 (5)	l'application lance l'affichage de sortie de zone		
Écran fin tuto (1)	les éléments graphiques (textes, audio, l'image de félicitation et la barre de progression) se lancent correctement. Le décompte de la barre de progression de vingt secondes se lance et la barre de progression se remplit chaque seconde.		Barre de progression au Sprint 4
Écran fin tuto (2)	la barre de progression est pleine et l'étape 1 du tutoriel s'affiche		
Écran fin tuto (3)	l'application lance l'affichage de l'écran d'accueil		

Figure 39 – Résultat Test Sprint 2 (3)

Écran Hors Zone (1)	les éléments graphiques (textes, audio, l'image des empreintes de pas et la barre de progression) se lancent correctement. Le décompte de la barre de progression de vingt secondes se lance et la barre de progression se remplit chaque seconde.		Si on vient de l'étape 2 alors l'affichage du curseur de la souris est toujours présent. De même l'audio venant de l'étape en cours ne s'arrête pas.
Écran Hors Zone (2)	l'application lance l'affichage de l'écran d'accueil		Pour le moment au bout de 3 secondes au lieu de 20
Écran Hors Zone (3)	l'application lance l'affichage de l'étape en cours du tutoriel dans l'état exact où elle était avant que l'utilisateur ne sorte de la zone.		Les modifications sur les positions et tailles des images ne sont pas prises en compte. Tout est réinitialisé.

Figure 40 – Résultat Test Sprint 2 (4)

Test	Résultat Attendu	Résultat Obtenu	Commentaires
Lancement Application	Les éléments graphiques (textes et audio) de l'écran d'accueil se lancent correctement.		
Écran d'accueil	L'étape 1 du tutoriel se lance		
Tutoriel Étape 1 (1)	les éléments graphiques (squelette, textes et audio) se lancent correctement		
Tutoriel Étape 1 (2)	l'application lance l'épreuve à effectuer (texte et audio)		
Tutoriel Étape 1 (3)	l'application comprend le geste et lance l'étape 2		
Tutoriel Étape 1 (4)	l'application lance l'affichage de sortie de zone		
Tutoriel Étape 2 (1)	les éléments graphiques (squelettes, textes, audio, affichage de la main dominante et barre de progression) se lancent correctement		Barre de progression au sprint 4
Tutoriel Étape 2 (2)	l'application lance l'épreuve à effectuer (texte, audio et cibles)		Les cibles s'affichent, mais lors d'un deuxième passage sur le tutoriel ne sont plus à la bonne taille
Tutoriel Étape 2 (3)	l'application comprend le geste et déplace le curseur de la souris en conséquence		
Tutoriel Étape 2 (4)	la cible touchée devient verte signifiant sa validation		
Tutoriel Étape 2 (5)	l'application affiche l'étape 3 du tutoriel		
Tutoriel Étape 2 (6)	l'application lance l'affichage de sortie de zone		
Tutoriel Étape 3 (1)	les éléments graphiques (squelettes, textes, audio) se lancent correctement		

Figure 41 – Résultat Test Sprint 3 (1)

Tutoriel Étape 3 (2)	l'application lance l'épreuve à effectuer (texte, audio et compteur de fermeture des poings) et le compteur de fermeture des poings a été réinitialisé		
Tutoriel Étape 3 (3)	l'application comprend le geste, affiche un cercle rouge et incrémente le compteur		
Tutoriel Étape 3 (4)	l'application affiche l'étape 4 du tutoriel		
Tutoriel Étape 3 (5)	l'application lance l'affichage de sortie de zone		
Tutoriel Étape 4 (1)	les éléments graphiques (squelettes, textes, audio et l'image de la statue) se lancent correctement		La statue change de position lors d'un 2e passage sur le tuto
Tutoriel Étape 4 (2)	l'application lance l'épreuve à effectuer (texte, audio et socle)		
Tutoriel Étape 4 (3)	l'application comprend le geste, et déplace la statue dans l'espace		Le déplacement est plus fluide et mieux interpréter. Information sur ce point dans le manuel utilisateur
Tutoriel Étape 4 (4)	la zone de réception est au bon endroit, l'application comprend la validation de l'étape et affiche l'étape 5 du tutoriel		La zone de réception est décalée par rapport au socle
Tutoriel Étape 4 (5)	l'application lance l'affichage de sortie de zone		
Tutoriel Étape 5 (1)	les éléments graphiques (squelettes, textes, audio, l'image de la statue et le cadre) se lancent correctement		
Tutoriel Étape 5 (2)	l'application lance l'épreuve à effectuer (texte, audio)		
Tutoriel Étape 5 (3)	la taille de l'image de la statue s'agrandit		La vitesse de changement de taille est trop lente

Figure 42 – Résultat Test Sprint 3 (2)

Tutoriel Étape 5 (4)	la taille de l'image de la statue se réduit		La vitesse de changement de taille est trop lente
Tutoriel Étape 5 (5)	l'application affiche l'étape 6 du tutoriel		L'image de la statue sort du cadre alors qu'elle devrait être bloquée dedans
Tutoriel Étape 5 (6)	l'application lance l'affichage de sortie de zone		
Tutoriel Étape 6 (1)	les éléments graphiques (squelettes, textes, audio, l'image de la statue et barre de progression) se lancent correctement		
Tutoriel Étape 6 (2)	l'application lance l'épreuve à effectuer (texte, audio)		
Tutoriel Étape 6 (3)	la barre de progression se remplit et la statue se tourne en fonction de la rotation du mouvement		
Tutoriel Étape 6 (4)	l'application affiche l'écran de fin du tutoriel		L'application plante. À résoudre au sprint 4
Tutoriel Étape 6 (5)	l'application lance l'affichage de sortie de zone		
Écran fin tuto (1)	les éléments graphiques (textes, audio, l'image de félicitation et la barre de progression) se lancent correctement. Le décompte de la barre de progression de vingt secondes se lance et la barre de progression se remplit chaque seconde.		Barre de progression au Sprint 4
Écran fin tuto (2)	la barre de progression est pleine et l'étape 1 du tutoriel s'affiche		
Écran fin tuto (3)	l'application lance l'affichage de l'écran d'accueil		

Figure 43 – Résultat Test Sprint 3 (3)

Écran Hors Zone (1)	les éléments graphiques (textes, audio, l'image des empreintes de pas et la barre de progression) se lancent correctement. Le décompte de la barre de progression de vingt secondes se lance et la barre de progression se remplit chaque seconde.		Si on vient de l'étape 2 alors l'affichage du curseur de la souris est toujours présent. De même l'audio venant de l'étape en cours ne s'arrête pas.
Écran Hors Zone (2)	l'application lance l'affichage de l'écran d'accueil		Pour le moment au bout de 3 secondes au lieu de 20
Écran Hors Zone (3)	l'application lance l'affichage de l'étape en cours du tutoriel dans l'état exact où elle était avant que l'utilisateur ne sorte de la zone.		Les modifications sur les positions et tailles des images ne sont pas prises en compte. Tout est réinitialisé.

Figure 44 – Résultat Test Sprint 3 (4)

Test	Résultat Attendu	Résultat Obtenu	Commentaires
Lancement Application	Les éléments graphiques (textes et audio) de l'écran d'accueil se lancent correctement.		
Écran d'accueil	L'étape 1 du tutoriel se lance		
Tutoriel Étape 1 (1)	les éléments graphiques (squelette, textes et audio) se lancent correctement		
Tutoriel Étape 1 (2)	l'application lance l'épreuve à effectuer (texte et audio)		
Tutoriel Étape 1 (3)	l'application comprend le geste et lance l'étape 2		
Tutoriel Étape 1 (4)	l'application lance l'affichage de sortie de zone		
Tutoriel Étape 2 (1)	les éléments graphiques (squelettes, textes, audio, affichage de la main dominante et barre de progression) se lancent correctement		
Tutoriel Étape 2 (2)	l'application lance l'épreuve à effectuer (texte, audio et cibles)		
Tutoriel Étape 2 (3)	l'application comprend le geste et déplace le curseur de la souris en conséquence		
Tutoriel Étape 2 (4)	la cible touchée devient verte signifiant sa validation		
Tutoriel Étape 2 (5)	l'application affiche l'étape 3 du tutoriel		
Tutoriel Étape 2 (6)	l'application lance l'affichage de sortie de zone		
Tutoriel Étape 3 (1)	les éléments graphiques (squelettes, textes, audio) se lancent correctement		

Figure 45 – Résultat Test Sprint 4 (1)

Tutoriel Étape 3 (2)	l'application lance l'épreuve à effectuer (texte, audio et compteur de fermeture des poings) et le compteur de fermeture des poings a été réinitialisé		
Tutoriel Étape 3 (3)	l'application comprend le geste, affiche un cercle rouge et incrémente le compteur		
Tutoriel Étape 3 (4)	l'application affiche l'étape 4 du tutoriel		
Tutoriel Étape 3 (5)	l'application lance l'affichage de sortie de zone		
Tutoriel Étape 4 (1)	les éléments graphiques (squelettes, textes, audio et l'image de la statue) se lancent correctement		Il arrive que la statue change de position lorsque l'on fait plusieurs fois le tuto de suite
Tutoriel Étape 4 (2)	l'application lance l'épreuve à effectuer (texte, audio et socle)		
Tutoriel Étape 4 (3)	l'application comprend le geste, et déplace la statue dans l'espace		
Tutoriel Étape 4 (4)	la zone de réception est au bon endroit, l'application comprend la validation de l'étape et affiche l'étape 5 du tutoriel		
Tutoriel Étape 4 (5)	l'application lance l'affichage de sortie de zone		
Tutoriel Étape 5 (1)	les éléments graphiques (squelettes, textes, audio, l'image de la statue et le cadre) se lancent correctement		
Tutoriel Étape 5 (2)	l'application lance l'épreuve à effectuer (texte, audio)		
Tutoriel Étape 5 (3)	la taille de l'image de la statue s'agrandit		
Tutoriel Étape 5 (4)	la taille de l'image de la statue se réduit		

Figure 46 – Résultat Test Sprint 4 (2)

Tutoriel Étape 5 (5)	l'application affiche l'étape 6 du tutoriel		Il arrive par moment pour une raison inexplicable que l'image sort du cadre
Tutoriel Étape 5 (6)	l'application lance l'affichage de sortie de zone		
Tutoriel Étape 6 (1)	les éléments graphiques (squelettes, textes, audio, l'image de la statue et barre de progression) se lancent correctement		
Tutoriel Étape 6 (2)	l'application lance l'épreuve à effectuer (texte, audio)		
Tutoriel Étape 6 (3)	la barre de progression se remplit et la statue se tourne en fonction de la rotation du mouvement		La compréhension du mouvement reste difficile. Il faudra sans doute une aide humaine à côté pour aide à sa réalisation.
Tutoriel Étape 6 (4)	l'application affiche l'écran de fin du tutoriel		
Tutoriel Étape 6 (5)	l'application lance l'affichage de sortie de zone		
Écran fin tuto (1)	les éléments graphiques (textes, audio, l'image de félicitation et la barre de progression) se lancent correctement. Le décompte de la barre de progression de vingt secondes se lance et la barre de progression se remplit chaque seconde.		
Écran fin tuto (2)	la barre de progression est pleine et l'étape 1 du tutoriel s'affiche		
Écran fin tuto (3)	l'application lance l'affichage de l'écran d'accueil		

Figure 47 – Résultat Test Sprint 4 (3)



Webographie

- [WWW0] UC DAVIS. *Augmented Reality Sandbox*. Sous la dir. d'UC DAVIS. 2016. URL : <https://arsandbox.ucdavis.edu/>.
- [WWW0] Arnaud DEVILLARD. *Un "T. rex" passé à la kinect*. 11 juil. 2017. URL : https://www.sciencesetavenir.fr/high-tech/un-t-rex-passe-a-la-kinect_114635.
- [WWW0] Eric LARSON. *Fly Like a Giant Prehistoric Lizard With Kinect-Powered Simulator*. 3 avr. 2014. URL : <http://mashable.com/2014/04/02/pterosaur-flight-simulator/#qZSS.7HPP5qq>.
- [WWW0] SEGD. *<https://segd.org/dinostomp>*. Sous la dir. de DINOSTOMP. 2017. URL : <https://segd.org/dinostomp>.



Bibliographie

- [0] Ding-Jung Chiang CHING-SHENG WANG et Yu-Chia WEI. « Intuitional 3D Museum Navigation System Using Kinect ». In : *Information Technology Convergence* (14 juil. 2013).

Applications de médiation pour l'utilisation de périphériques sans contact en musée

Logan Verecque

Encadrement : Gilles Venturini et Barthelemy Serres

Objectifs

Conception de deux tutoriels pour apprentissage des technologies :

- Microsoft Kinect
- Leap Motion

Les visiteurs du Musée des beaux-arts de Tours devront, à terme, pouvoir utiliser ces technologies sans difficulté.



Kinect

Mise en œuvre

Kinect et Leap Motion utilisent la reconnaissance de mouvements par le corps ou les mains respectivement. Les fonctionnalités suivantes seront proposées :

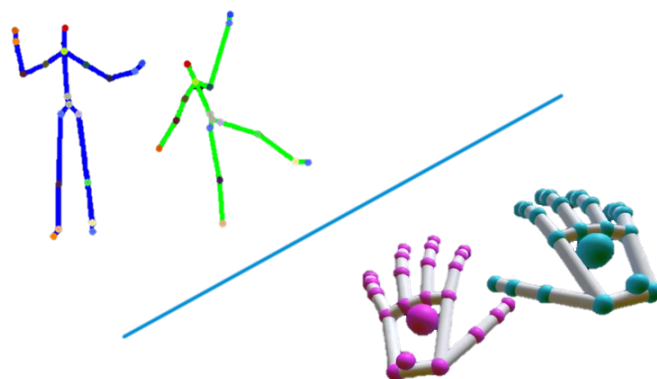
- Contrôle du curseur de la souris à distance
- Manipulation d'images 2D et 3D
- Dessiner



Leap Motion

Résultats attendus

- Utilisation intuitive
- Applications attractives et ergonomiques
- Robustesse : les applications devront pouvoir tourner sans interruption tout au long de la journée
- Application pour tout public



Reconnaissance du corps avec Kinect et des mains avec Leap Motion

Applications de médiation pour l'utilisation de périphériques sans contact en musée

Logan Verecque

Encadrement : Gilles Venturini et Barthelemy Serres

Objectifs

Conception de deux tutoriels pour apprentissage des technologies :

- Microsoft Kinect
 - Leap Motion
- Les visiteurs du Musée des beaux-arts de Tours devront, à terme, pouvoir utiliser ces technologies sans difficulté.

Mise en œuvre

- Kinect et Leap Motion utilisent la reconnaissance de mouvements par le corps ou les mains respectivement. Les fonctionnalités suivantes seront proposées :
 - Contrôle du curseur de la souris à distance
 - Manipulation d'images 2D et 3D
 - Dessiner

Résultats attendus

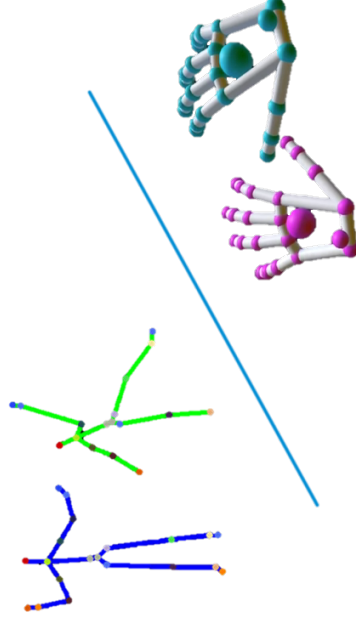
- Utilisation intuitive
- Applications attractives et ergonomiques
- Robustesse : les applications devront pouvoir tourner sans interruption tout au long de la journée
- Application pour tout public



Kinect



Leap Motion



Reconnaissance du corps avec Kinect et des mains avec Leap Motion

Applications de médiation pour l'utilisation de périphériques sans contact en musée

Résumé

Ce projet propose la création de deux applications de médiations pour apprendre la technologie Kinect et Leap Motion. La première est un tutoriel complet permettant d'apprendre à utiliser Kinect pas à pas en suivant une série d'étape lors d'un scénario complet. La deuxième, quant à elle consiste en une série de petites applications que l'utilisateur peut utiliser comme il le souhaite et donc apprendre par lui-même et voir quelles sont les possibilités que lui offre la technologie Leap Motion

Mots-clés

Kinect, Leap Motion, Musée, Reconnaissance de mouvements, Applications

Abstract

This project proposes the creation of two mediation applications to learn Kinect and Leap Motion technologies. The first is a complete tutorial to learn how to use Kinect step by step by following a series of steps in a complete scenario. The second is a series of small applications that the user can use as he wishes and thus learn for himself and see what the possibilities are with the Leap Motion technology.

Keywords

Kinect, Leap Motion, Museum, Applications

Tuteurs académiques

Gilles VENTURINI
Barthelemy SERRES

Étudiant

Logan VERECQUE (DI5)