**Création d’un tutoriel Kinect V2**



Rémi Jacquemard – Fin2

ETML - Lausanne

De lundi 11 mai au lundi 8 juin 2015

Chef de projet : Patrick Chenaux

**Table des matières**

[1 Spécifications 4](#_Toc421266565)

[1.1 Spécifications de départ et documentation associée 4](#_Toc421266566)

[1.1.1 Objectifs et portée du projet (objectifs SMART) 4](#_Toc421266567)

[1.1.2 Fonctionnalités requises (du point de vue de l’utilisateur) 4](#_Toc421266568)

[1.1.3 Caractéristiques des utilisateurs 4](#_Toc421266569)

[1.1.4 Travail à réaliser par l'apprenti dans le cadre de ce travail spécialisé 5](#_Toc421266570)

[1.1.5 Méthodes de validation des solutions 5](#_Toc421266571)

[1.1.6 Modification du CDC 5](#_Toc421266572)

[2 Planification 6](#_Toc421266573)

[2.1 Planning(s) initial 6](#_Toc421266574)

[2.2 Version 2 du planning 7](#_Toc421266575)

[3 Analyse initiale 8](#_Toc421266576)

[3.1 Faisabilité 8](#_Toc421266577)

[3.2 Document d’analyse et conception 9](#_Toc421266578)

[3.2.1 Structure du tutoriel 9](#_Toc421266579)

[3.3 Les projets à réaliser 12](#_Toc421266580)

[3.4 Conception des tests 12](#_Toc421266581)

[4 Réalisation 13](#_Toc421266582)

[4.1 Dossier de Réalisation 13](#_Toc421266583)

[4.1.1 Design général du tutoriel 13](#_Toc421266584)

[4.1.2 La structure du tutoriel 14](#_Toc421266585)

[4.1.3 Les images, schémas, légendes et références 14](#_Toc421266586)

[4.1.4 Les sources 15](#_Toc421266587)

[4.1.5 L’archive fournie 15](#_Toc421266588)

[4.1.6 Les normes de l’ETML 16](#_Toc421266589)

[4.1.7 Les styles 16](#_Toc421266590)

[4.1.8 Gestion des versions 17](#_Toc421266591)

[4.1.9 Points relatifs aux projets 18](#_Toc421266592)

[4.1.10 Les projets effectués avant les TPI 19](#_Toc421266593)

[4.2 Modifications 20](#_Toc421266594)

[5 Tests 20](#_Toc421266595)

[5.1 Dossier des tests 20](#_Toc421266596)

[6 Conclusion 21](#_Toc421266597)

[6.1 Bilan des fonctionnalités demandées 21](#_Toc421266598)

[6.2 Bilan de la planification 22](#_Toc421266599)

[6.3 Bilan personnel 24](#_Toc421266600)

[7 Divers 25](#_Toc421266601)

[7.1 Journal de travail de chaque participant 25](#_Toc421266602)

[7.2 Sources 25](#_Toc421266603)

[8 Annexes 26](#_Toc421266604)

# Spécifications

## Spécifications de départ et documentation associée

Voici le cahier des charges dans sa version initiale :

### Objectifs et portée du projet (objectifs SMART)

* Créer entièrement un tutoriel sur le périphérique Kinect version 2 de Microsoft.
* Réaliser une documentation complète sur le tutoriel, celle qui sera transmise aux élèves.
* Réaliser une documentation complète sur le projet.
* Informer régulièrement le chef de projet sur l’état du tutoriel
* Respecter le cahier des charges

### Fonctionnalités requises (du point de vue de l’utilisateur)

* L'utilisateur doit pouvoir en suivant le tutoriel:
  + obtenir un historique sur le périphérique Kinect.
  + Comprendre le principe de fonctionnement de Kinect.
  + Installer un poste de travail permettant de suivre le tutoriel.
* L'utilisateur doit pouvoir créer un projet (***Kinect Draw***) :
  + qui lui permet de dessiner de manière simplifiée sur un écran.
  + Pour cela il faut intégrer des explications sur l’utilisation de la classe KinectCore Window.
* L'utilisateur doit pouvoir créer un projet (***Kinect PowerPoint***) :
  + qui lui permet de créer 1 ADP in pour Microsoft PowerPoint.
  + Pour cela il faut intégrer des explications sur la création d’ADP in sur Microsoft PowerPoint, des explications sur la détection de mouvements (logiciel KinectStudio, VisualGestureBuilder, gestion de mouvement dans une application).
  + L’ADP in doit permettre de passer d’une slide à un autre et de pointer à l’aide du bras ou la main afin de faire apparaître un pointeur sur la slide, ce pointeur simule un pointeur laser.
* L'utilisateur doit pouvoir créer un projet:
  + qui lui permet de jongler avec un objet à l’écran (ballon par exemple), la personne peut être représentée de manière simplifiée à l’écran.
* Il convient de produire une documentation pour l’utilisateur qui lui permette de travailler de manière autonome avec le périphérique Kinect version 2 de Microsoft.

### Caractéristiques des utilisateurs

 Les utilisateurs sont des élèves de troisième ou quatrième année de la section informatique de l’ETML.

Contraintes (sécurité, système utilisé, interfaces avec autres logiciels, etc.)

* Si le temps le permet, ajouter un projet ou deux utilisateurs doit pouvoir:

 jongler avec un objet à l’écran (ballon par exemple), les deux personnes peuvent être représentées de manière simplifiée à l’écran.

* Si le temps le permet, le candidat crée un jeu ou une application de son choix.

### Travail à réaliser par l'apprenti dans le cadre de ce travail spécialisé

* Analyser et planifier les différentes tâches.
* L'analyse doit contenir une conception des fonctionnalités, un design des applications.
* Lister les tests qui devront être réalisés avec les résultats attendus.
* Développer l'application en respectant les normes, standards et méthodes apprises durant toute la formation à l'ETML
* tester les différentes applications.
* Produire les documentations nécessaires à la mise en œuvre et à d'éventuelles modifications ultérieures par une tierce personne.

### Méthodes de validation des solutions

* À définir par le candidat.

### Modification du CDC

Le 22.02.2015, le choix a été fait de modifier le cahier des charges avec le chef de projet. En effet, il a été remarqué que le temps manquerait et que le planning ne pourrait pas être suivi. C’est pourquoi certains points ont été modifiés :

|  |
| --- |
| * L'utilisateur doit pouvoir créer un projet:   + qui lui permet de jongler avec un objet à l’écran (ballon par exemple), la personne peut être représentée de manière simplifiée à l’écran. |

Ce point a été modifié. En effet, il est possible que ce projet prenne du temps à analyser, à réaliser et à documenter dans le tutoriel. C’est pourquoi il devient facultatif, et sera réalisé uniquement si le temps le permet.

|  |
| --- |
| * L'utilisateur doit pouvoir créer un projet:   + qui lui permet de créer 1 ADP in pour Microsoft PowerPoint.   + Pour cela il faut intégrer des explications sur la création d’ADP in sur Microsoft PowerPoint, des explications sur la détection de mouvements (logiciel KinectStudio, VisualGestureBuilder, gestion de mouvement dans une application).   + L’ADP in doit permettre de passer d’une slide à un autre et de pointer à l’aide du bras ou la main afin de faire apparaître un pointeur sur la slide, ce pointeur simule un pointeur laser. |

Deux points ont été modifiés.

D’une part, les logiciels *KinectStudio* et, avant tout, *VisualGestureBuilder*, sont difficiles et long à décrire, car chacun ayant leur propre concept et leurs nombreuses fonctionnalités.  
Dans ce projet, ces logiciels étaient utilisés pour créer un fichier de base de données contenant les gestes à détecter. Cependant, ce fichier peut être fourni directement à l’élève, ce qui permet de gagner du temps. Si le temps le permet, il sera possible de tout de même expliquer de quelle manière créer ce fichier!

D’autre part, malgré le fait que pointer à l’aide du bras ou la main sur la slide utilise le même concept que le projet *KinectDraw*, son intégration à PowerPoint est compliquée. En effet, l’utilisation de l’outil pointeur intégré aux présentations ne peut pas être manipulé directement par 1 ADP in, et l’utilisation d’un *workaround* (un objet en forme d’ellipse est créé dans chacune des slides au lancement du mode présentation, cet objet doit s’effacer à chaque changement de slide et lors du retour dans le mode édition, il ne doit pas apparaître, etc.) rend cette fonctionnalité compliquée à décrire. De plus, des problèmes de threading (accès simultané à la même ressource) sont apparus, ce qui a rendu le code complexe. Le concept du pointeur sera tout de même décrit en une page dans le tutoriel

# Planification

**Information importante**

Les deux projets abordés dans le tutoriel, soit *Kinect Draw* et *Kinect PowerPoint*, ont été réalisés en grande majorité dans le cadre des pré-TPI. Seules quelques retouches doivent être effectuées au code. Ces projets sont donc utilisés, et pratiquement aucune heure n’est à planifier pour leur réalisation.

De plus, toutes leurs fonctionnalités ne sont pas décrites dans le tutoriel.

## Planning(s) initial

Vous trouverez en annexe le fichier Excel contenant le planning initial dans son ensemble. Cependant, en voici les informations principales :





Voici un résumé des tâches et de leurs répartitions :

|  |  |  |
| --- | --- | --- |
| Tâches - objectifs | Nb 1/4 heure | Nb d’heure |
| Absence - Imprévus | 156 | 39 |
| Prise de connaissance du projet | 6 | 1,5 |
| Information | 9 | 2,25 |
| Planification du projet | 12 | 3 |
| Analyse - Tutoriel | | |
| Analyse - Tutoriel : Structure | 3 | 0,75 |
| Analyse - Tutoriel : Difficulté du code | 1 | 0,25 |
| Analyse - Projets | | |
| Analyse - Projets : Technologie de l'API à utiliser | 2 | 0,5 |
| Analyse - Projets : Diagramme de classe | 9 | 2,25 |
| Analyse - Projets : Fonctionnalités | 3 | 0,75 |
| Décision - Tutoriel | 1 | 0,25 |
| Décision - Projets | 1 | 0,25 |
| Réalisation/Documentation - Tutoriel | 9 | 2,25 |
| Réalisation/Documentation - Tutoriel : Design | 12 | 3 |
| Réalisation/Documentation - Tutoriel : Structure | 9 | 2,25 |
| Réalisation/Documentation - Tutoriel : Contenu | 171 | 42,75 |
| Réalisation - Développement : Kinect Draw | 6 | 1,5 |
| Réalisation - Développement : Kinect PowerPoint | 6 | 1,5 |
| Réalisation - Développement : Kinect Ball | 54 | 13,5 |
| Documentation | 64 | 16 |
| Évaluation | 6 | 1,5 |
| Total planifié | 540 | 135 |
| Total planifié, jours fériés non compris | **384** | 96 |

On pourra remarquer que près de 45% du temps (soit près de 43 heures sur les 96 planifiées) est utilisé pour réaliser le contenu du tutoriel. En effet, c’est le sujet principal du TPI, et faire un tutoriel clair, précis et reproductible peut prendre beaucoup de temps. C’est pourquoi, sur cette planification, cette tâche est privilégiée.

Dans le planning détaillé, en début de semaine 4, on peut trouver les deux tâches ci-dessous :

|  |  |  |
| --- | --- | --- |
| Documentation | 6 | Complétion/"finalisation" du rapport de projet des projets déjà écrit |
| Réalisation/Documentation - Tutoriel : Contenu | 6 | Complétion/"finalisation" du tutoriel pour que celui-ci soit déjà plus ou moins utilisable |

Ces deux tâches ont été ajoutées pour que le tutoriel et la documentation soient déjà transmissibles une semaine avant la fin du TPI. Ainsi, même s’il venait à manquer de temps pour terminer l’ensemble du TPI, le travail effectué sera tout de même utilisable. On peut les considérer comme des tâches *tampons*.

## Version 2 du planning

Le planning a été modifié le 22.05.2015 en réponse à la modification du cahier des charges (sous-chapitre : 1.1). En effet, il a été remarqué que le temps manquerait et que le planning ne pourrait pas être suivi.

Vous trouverez en annexe le fichier Excel du journal de travail V2, contenant le planning mis à jour. Cependant, voici un résumé des quelques tâches et de leur nouvelle répartition :

|  |  |  |
| --- | --- | --- |
| Tâches - objectifs | Nb 1/4 heure | Nb d’heure |
| Absence - Imprévus | 156 | 39 |
| Prise de connaissance du projet | 6 | 1,5 |
| Information | 3 | 0,75 |
| Planification du projet | 12 | 3 |
| Analyse - Tutoriel | | |
| Analyse - Tutoriel : Structure | 3 | 0,75 |
| Analyse - Tutoriel : Difficulté du code | 1 | 0,25 |
| Analyse - Projets | 0 | 0,0 |
| Analyse - Projets : Technologie de l'API à utiliser | 0 | 0,0 |
| Analyse - Projets : Diagramme de classe | 3 | 0,75 |
| Analyse - Projets : Fonctionnalités | 0 | 0,0 |
| Décision - Tutoriel | 1 | 0,25 |
| Décision - Projets | 0 | 0,0 |
| Réalisation/Documentation - Tutoriel | | |
| Réalisation/Documentation - Tutoriel : Design | 12 | 3 |
| Réalisation/Documentation - Tutoriel : Structure | 9 | 2,25 |
| Réalisation/Documentation - Tutoriel : Contenu | 201 | 50,25 |
| Réalisation - Développement : Kinect Draw | 6 | 1,5 |
| Réalisation - Développement : Kinect PowerPoint | 6 | 1,5 |
| Réalisation - Développement : Kinect Ball | 0 | 0,0 |
| Documentation | 97 | 24,25 |
| Évaluation | 24 | 6 |
| Total planifié, jours fériés non compris | **384** | 96 |

On remarque les changements majeurs suivants :

* Le nombre d’heures pour le contenu du tutoriel a été augmenté, afin que celui-ci soit finalisé et que les retards soient compensés
* Le développement du projet *KinectBall* n’est plus du tout prévu dans le planning. Cependant, si le temps le permet, il sera tout de même réalisé
* Le nombre de quarts d’heure concernant la documentation du projet a été revu à la hausse

Grâce à ces modifications, il est prévu que ce projet finisse dans les temps.

* Ce paragraphe présent le *planning d'origine* (date de début, date de fin, vacances et congés, liste hiérarchique des tâches ou GANTT, jalons, durée totale)
* toutes les mises à jour subies par le planning sont à reporter (avec date de mise à jour) et peuvent déboucher sur plusieurs versions de plannings.

# Analyse initiale

Cette analyse ne prend pas en compte la modification du cahier des charges. Ainsi, le projet *Kinect Ball* est abordé, alors qu’il ne sera finalement pas réalisé.

## Faisabilité

La plupart des points concernant la faisabilité du projet ont été abordés dans le chapitre *1. Spécifications*. Vous trouverez ci-dessous quelques informations supplémentaires.

Premièrement, il est bien entendu nécessaire de disposer du capteur *Kinect V2*, ainsi que la configuration minimale qu’il demande sur un PC. Ce point est abordé dans le tutoriel. En voici l’extrait concerné :

* Un PC
  + Un processeur 64-bit dual-Corée cadencé à 3.1 GHz
  + Un port USB3 disponible
  + 4Go de RAM
  + Une carte graphique supportant DirectX11
  + Windows 8 minimum, 8.1 pour pouvoir suivre ce tutoriel
* Le capteur Kinect pour Xbox One
* L’adaptateur Kinect (V2) pour PC
* Une version de Microsoft Visual Studio 2013
* Pour le projet Kinect PowerPoint : Le logiciel Microsoft Office PowerPoint 2013

Pour créer ce tutoriel, il est nécessaire d’avoir déjà touché à la programmation pour Kinect V2. C’est ce qui a été fait dans le cadre des projets de fin d’années et des pré-TPI. Deux petits projets ont été créés sans avoir de notions spécifiques au développement Kinect. Indirectement, cela a permis de réfléchir comment le tutoriel devrait être structuré, quelles informations y retrouver, etc. pour un apprentissage aisé. C’est pourquoi l’orientation à donner au tutoriel a été effectuée en connaissance de cause.

Les deux projets créés avant le TPI seront incorporés au tutoriel. Il est possible que les fonctionnalités présentes ne soient pas toutes intégrées au tutoriel. Ces deux projets sont nommés ainsi :

* **Kinect Draw**Projet qui permet de dessiner de manière simplifiée sur un écran
* **Kinect PowerPoint**Projet qui permet de passer d’une slide à l’autre sur PowerPoint à l’aide d’un geste de la main et de pointer à l’aide du bras afin de faire apparaître un pointeur sur la slide courante ; il simule un pointeur laser

Les principales compétences nécessaires au TPI ont été acquises. Néanmoins, le projet Kinect Ball comprend quelques points à approfondir :

* La classe CoordinateMapper, qui permet, par exemple, de faire correspondre un point dans l’espace vue depuis un certain capteur de Kinect à un point dans le plan correspondant à l’image en couleur
* L’interaction entre les squelettes des joueurs et une balle *virtuelle* (quelles fonctionnalités de l’API utiliser, etc.)
* Comment ajouter de la gravité à une balle, et la collision associée?

## Document d’analyse et conception

**Information importante** : ce chapitre traitera uniquement de l’analyse de la création du tutoriel, soit le but de ce TPI. Pour éviter les redondances inutiles, vous pourrez trouver l’analyse des projets (structure des applications, etc.) directement dans le tutoriel

### Structure du tutoriel

Du cahier des charges, on peut mettre en évidence les points résumés suivants qui doivent apparaître dans le tutoriel :

* Historique de Kinect
* Fonctionnement de Kinect
* Installation du poste de travail
* Projet de dessin Kinect
  + La classe KinectCoreWindow
* Projet d’Andin pour PowerPoint
  + Logiciel KinectStudio (permets d’enregistrer un clip vidéo)
  + Logiciel VisualGestureBuilder (permet de créer un fichier contenant les mouvements à détecter à partir d’un clip vidéo enregistré)
  + Intégrer la détection de mouvement dans le développement d’une application

Après analyse, voici les chapitres qui seront présents dans le tutoriel :

1. **Introduction**Ce chapitre décrira le rôle de ce tutoriel et contiendra un historique de Kinect (qui développe Kinect, en quoi la version V2 améliore l’ancienne version, les fonctionnalités présentes, quelques jeux déjà développés pour Kinect, etc.). Ce chapitre servira aussi de mise en bouquet.
   * **À qui se destine ce tutoriel ?**On trouvera dans ce sous-chapitre à qui se destine le tutoriel (élève de l’ETML en informatique en 3e ou 4e année, les modules ICH-103 et ICH-303 de programmation, notions de programmation orientée objet, etc.)
   * **Marche à suivre**Ce sous-chapitre indiquera ce que devront faire les élèves de l’ETML pour bien suivre le tutoriel (récupérer le code fourni, etc.)
2. **L’installation du poste de travail**Ce chapitre contiendra toutes les informations nécessaires pour que l’élève puisse commencer à coder sans problème
   * Posséder le capteur Kinect V2
   * Posséder l’adaptateur Kinect V2 pour le PC
   * PC : Windows 8 au minimum, USB3, etc.
   * Installation de Visual Studio
   * Installation du SDK de développement
   * Avoir PowerPoint installé pour le projet Kinect PowerPoint
   * Etc.
3. **Découverte de l’API**On découvrira dans ce chapitre la structure générale de l’API ainsi que certaines classes nécessaires au développement
   * KinectSensor (classe principale)
   * Les *frames* (les principales images reçues par le capteur Kinect et utilisables dans notre application)
4. **Premier projet : Kinect Draw**On commencera à toucher dans ce chapitre au développement sur Kinect, en créant un petit projet qui consistera à pouvoir dessiner de manière simplifiée sur un écran. Voici ce qu’il contiendra :
   * Présentation du projet
   * Une petite introduction à la structure d’une application Windows Store
   * Une introduction à la classe KinectCoreWindow, qui permet de créer un pointeur selon des axes X-Y de manière simplifiée, ainsi que le système de PHIZ utilisé par cette classe
   * Un code de base contenant le design et la structure de l’application sera fourni
   * Le développement en lui-même
5. **Deuxième projet : Kinect PowerPoint**On développera en guidant l’utilisateur dans ce chapitre 1 ADP in pour PowerPoint qui permettra, tel que défini dans le cahier des charges, de passer d’une slide à un autre et de pointer à l’aide du bras ou la main afin de faire apparaître un pointeur sur la slide, ce pointeur simule un pointeur laser. Voici ce qu’il contiendra :
   * Présentation du projet
   * Présentation du logiciel KinectStudio, qui permet de s’enregistrer. L’utilisateur s’enregistrera en train d’effectuer les mouvements qui permettront de changer de slide (vers l’avant, vers l’arrière, de la main gauche et de la main droite)
   * Présentation du logiciel VisualGestureBuilder, qui permet à partir d’un clip enregistré à partir de KinectStudio, de créer un fichier contenant les gestes à détecter dans une application. L’utilisateur créera le fichier de base de données contenant les gestes qu’il a précédemment enregistré avec KinectStudio
   * Une petite introduction à la structure de 1 ADP in pour PowerPoint
   * Présentation des classes *VisualGestureBuilderFrame*, qui permettent de détecter certains mouvements de personne à partir du fichier de base de données dans une application
   * Le développement en lui-même
6. **Troisième projet : Kinect Ball**L’utilisateur pourra ici développer un projet qui lui permettra de jongler avec un objet à l’écran
   * Présentation du projet
   * Description des classes pas encore présentée qui sera nécessaire au projet et de leurs utilisations
   * Un code de base contenant le design et la structure de l’application sera fourni
   * Le développement en lui-même
7. **Conclusion**Ce chapitre contiendra un rapide résumé de ce qui a été vu
8. **Pour aller plus loin…**Ce chapitre contiendra quelques pistes pour que l’utilisateur puisse continuer d’apprendre par lui-même le développement sur Kinect V2, tel que des classes qui restent à découvrir, ou des liens sur internet pour approfondir certains points.  
   Il peut aussi contenir des informations sur les projets effectués dans le cadre des préTPI
9. **Annexe (si nécessaire)**
10. **Sources**
11. **Remerciements**

Information concernant les projets

2 options s’offraient pour la création des projets dans le tutoriel :

1. Laisser coder l’utilisateur à partir de zéro, uniquement avec les modèles déjà créés. Ceci a l’avantage que l’utilisateur comprendrait la grande majorité du code. Cependant, le développement d’application Windows Store est censé être nouveau pour lui, et est assez compliqué à prendre en main
2. Fournir à l’utilisateur un code de base contenant d’ores et déjà le design et la structure de l’application, mais aucun code concernant Kinect. Une explication sur ce code serait fournie. L’utilisateur pourrait se lancer directement dans la programmation pour Kinect V2, et peut-être ainsi éviter une perte de motivation dès le début du tutoriel. De plus, il est possible qu’il ne soit aucunement intéressé par le développement d’application Windows Store

après discussion avec M. Chenaux, il a donc été décidé qu’un code basique serait fourni à l’utilisateur, et ceci pour tous les projets abordés dans le tutoriel.

## Les projets à réaliser

Pour rappel, l’analyse détaillée des projets est rédigée dans le tutoriel. Il est bon de préciser que ces analyses sont orientées pour le tutoriel spécifiquement. Les concepts principaux de ces projets y sont explicités. De plus, étant des projets d’introduction, la difficulté du code n’est pas excessive.

## Conception des tests

L’idéal pour tester un tutoriel serait de le fournir à quelques élèves afin d’avoir un retour. Certains points peuvent être clairs à l’écriture, mais peu compréhensible pour le lecteur. Cependant, plusieurs points ne permettent pas de le faire dans le cadre de ce TPI :

* Le manque de temps : En effet, le TPI ne durant que quelques semaines, il n’est que peu envisageable de produire le tutoriel et de le faire tester.
* Le matériel : Ce projet utilise Kinect, et son adaptateur pour PC. Ce n’est pas un matériel que l’on possède tous. Il n’est donc pas possible de développer pour Kinect dans le cadre du TPI et, dans le même temps, fournir Kinect et son adaptateur.

Cependant, il reste possible de vérifier la compréhension des concepts, de la bonne tournure de phrase, du design du tutoriel, de l’orthographe et de la grammaire en fournissant uniquement le dossier écrit du tutoriel.

Quant aux projets, voici quelques tests qui peuvent être effectués :

* Tests tout au long du développement : les projets sont réalisés par étapes, en ajoutant petit à petit des fonctionnalités. Ceci permet que le projet reste fonctionnel.
* Essai des projets par des tiers : ils peuvent ainsi rapporter un bug, ou une manipulation avec Kinect trop compliquée (le fait de ne pas utiliser ni clavier ni souris complique souvent l’interaction)
* Tests unitaires : cependant, par manque de temps, il sera peu probable de pouvoir en créer

# Réalisation

## Dossier de Réalisation

### Design général du tutoriel

Le design du tutoriel est basé sur le modèle de l’ETML. Les en-têtes et pieds de pages ont été mis à jour, ainsi que la page de garde. Pour faciliter la lecture, la taille de la police a été modifiée à 11 points.

### La structure du tutoriel

Le tutoriel a été réalisé en grande partie telle que l’analyse le demandait. En voici la structure des chapitres finale :

1 Introduction

1.1 Le rôle de ce tutoriel

1.2 Kinect

1.3 À qui se destine ce tutoriel ?

1.4 Marche à suivre

2 Mise en place du poste de travail

2.1 Composants et logiciels nécessaires

2.2 Installation

3 Découverte de Kinect et son API

3.1 Les data sources

3.2 Les frames

4 Premier projet : Kinect Draw

4.1 Description du projet

4.2 Mise en place et prérequis

4.3 Structure et design du projet

4.4 Petite introduction à la structure de ce projet Windows 8.1

4.5 La classe KinectCoreWindow

4.6 Développement et conseils

4.7 Correction

5 Deuxième projet : Kinect PowerPoint

5.1 Description du projet

5.2 Mise en place et prérequis

5.3 Structure et principe du projet

5.4 La détection de mouvement avec Visual Gesture Builder

5.5 Principe de la détection de mouvement dans le projet

5.6 Introduction au développement de cet ADP in PowerPoint

5.7 Développement et conseils

5.8 Correction

5.9 Le concept de l’ajout du pointeur

6 Pour aller plus loin…

6.1 Ajouter des fonctionnalités aux projets

7 Sources

8 Remerciements

Information : pour en simplifier la lecture et en améliorer la compréhension, les sous-chapitres de niveau 3 et plus ont été omis de cette structure

La différence la plus frappante entre la structure finale et la structure analysée initialement est la disparition du chapitre *Troisième projet : Kinect Ball*. Ceci est dû à la deuxième version du cahier des charges (voir chapitre 1.1.6).

Le chapitre *Conclusion* a été retiré par rapport à ce que demandait l’analyse. En effet, le chapitre *pour aller plus loin…* permet de conclure le tutoriel.

Le reste de la structure est très proche de celle analysée. Quelques chapitres ont été ajoutés pour que la structure soit plus claire pour l’élève.

### Les images, schémas, légendes et références

Le tutoriel a été fortement illustré. En effet, pour aider à la compréhension de l’élève, des images et des schémas ont été insérés. Parfois, une explication écrite peut être beaucoup moins claire qu’un schéma explicatif.   
De plus, développer pour Kinect nécessite la compréhension de beaucoup de concept, et ceux-ci n’en sont que mieux décrits à l’aide d’une image ou d’un schéma. Une fois les concepts bien assimilés, sa transcription en instruction C# en devient facilitée.

Sous chacune de ces images, une légende a été ajoutée. En voici un exemple :

Figure 4.10 : La position de la main dans la PHIZ est retournée en %

La fonctionnalité *insérer une légende* de Word a été utilisée. Celle-ci permet, entre autres, de numéroter les figures automatiquement.   
Le choix a été fait d’ajouter le numéro de chapitre devant le numéro de figure, pour une meilleure compréhension de la linéarité et de la structure du tutoriel par l’élève.

Les notes de bas de page ont été utilisées lorsque des précisions sur certains points étaient nécessaires.   
De plus, la fonctionnalité *référence* de Word a souvent été utilisée, permettant ainsi, au désir de l’élève, d’accéder rapidement à un chapitre à l’aide d’un clic (lors d’une lecture numérique).

### Les sources

Les sources utilisées dans le tutoriel ont été indiquées, notamment pour les images tirées d’internet ou de capture d’écran de logiciel.

Il avait été prévu que la fonctionnalité d’ajout de source de Word (onglet *références*, *citations et bibliographie*) soit utilisée dans le tutoriel. Cependant, après l’avoir testée et utilisée pendant une semaine, elle a été abandonnée. En effet, cette fonctionnalité est plutôt utile pour des sources dont on utilise le contenu mot pour mot.

### L’archive fournie

Pour rappel, après analyse, il a été décidé qu’une archive contenant les projets de base *Kinect Draw* et *Kinect Powerpoint* serait fournie à l’élève.

La question se posait de savoir quelle devra être la marche à suivre pour les élèves de l’ETML pour s’approprier cette archive. L’utilisation d’un stockage sur le cloud tel que *Dropbox* a été pensée, ou encore une mise à disposition sur le réseau, via le lecteur *k:*. Il aurait été possible d’indiquer ainsi le chemin de cette archive directement dans le tutoriel.

Après discussion avec le chef de projet, il sera considéré dans le tutoriel que l’archive est sur un CD.

### Les normes de l’ETML

L’ETML a ses normes de codage. La dernière version (au 31 mai 2015) est la 1.1.0. La norme a été décrite dans le fichier *ETML développent guidelines*, disponibles pour les élèves sur le réseau. Cependant, en voici quelques spécificités (liste tirée du tutoriel) :

* Chaque fichier créé contient un entête contenant l’auteur, la date, l’entreprise (ETML) et un bref résumé :   
  ///ETML   
  ///Author : Luke Skywalker   
  ///date : 19.01.2014   
  ///Summary : Algorithm which transforms the dark force into light...
* Pour chaque méthode, l’utilisation des commentaires XML de Visual Studio est utilisée et les éventuelles sources sont indiquées
* Des commentaires de fin d’accolade sont utilisés seulement si ceux-ci améliorent la visibilité
* Les noms de variable utilisent le « *camelCase »*, **ne** sont **pas** préfixés, mais restent explicites :   
  Exemple : titleLabel (de type *Label*), nase (de type *string*), *isCursorMoving* (de type booléen), etc.

Le choix a été fait de rédiger les normes de codages en français, et non pas en anglais. En effet, dans le cadre d’un tutoriel, des explications claires et précises sont nécessaires. Des commentaires en français, pour un francophone, aident à la compréhension des instructions, ainsi qu’aux concepts dont elles découlent. Ce point a été discuté avec le chef de projet, et validé.

### Les styles

Dans le cadre de ce tutoriel, deux styles personnalisés ont été créés, dans le but d’en simplifier la compréhension.

Le style *code* permet d’intégrer du code brut depuis Visual Studio dans le fichier Word :

* La taille de police est de 8 points, ce qui permet un gain de place et de lisibilité.
* Les couleurs contextuelles de Visual Studio sont conservées
* Une bordure à gauche du code a été ajoutée, ce qui permet de comprendre de suite que la partie concernée est du code

Ci-dessous, un exemple de code tel qu’il apparait dans le tutoriel :

void kinectCore\_PointerMoved(KinectCoreWindow sender, KinectPointerEventArgs args)

{

KinectPointerPoint point = args.CurrentPoint;

}

Le style *instruction* indique à l’utilisateur une manipulation à effectuer :

* Des bordures ont été ajoutées, ce qui permet de comprendre de suite que la partie concernée est une instruction

Ci-dessous, un exemple d’instructions telles qu’elles apparaît dans le tutoriel :

Atteindre la méthode *initKinect()*

Compléter la section « création d'un GestureDetector» de cette méthode

### Gestion des versions

Information : l’outil décrit ici a été utilisé principalement dans le cadre des préTPI. Si le projet Kinect Ball avait été réalisé, il aurait été utilisé.

Pour gérer les différentes versions des programmes développés, l’outil de gestion *Visual Studio Online* a été testé et utilisé. Il utilise *Team Foundation*

Un de ses avantages et d’être bien implémenté dans *Visual Studio*. Il dispose aussi d’un outil en ligne, disponible sur <https://www.visualstudio.com>

Il permet de créer des tâches depuis *Visual Studio*, ou depuis l’interface en ligne. On peut les assigner à une personne spécifique, indiquer le temps prévu, etc. Il peut être paramétré pour utiliser des méthodes agiles telles que *Scrum*. Un des points intéressants de cet outil est le tableau (*Board*), comme on le voit sur la figure ci-dessous:

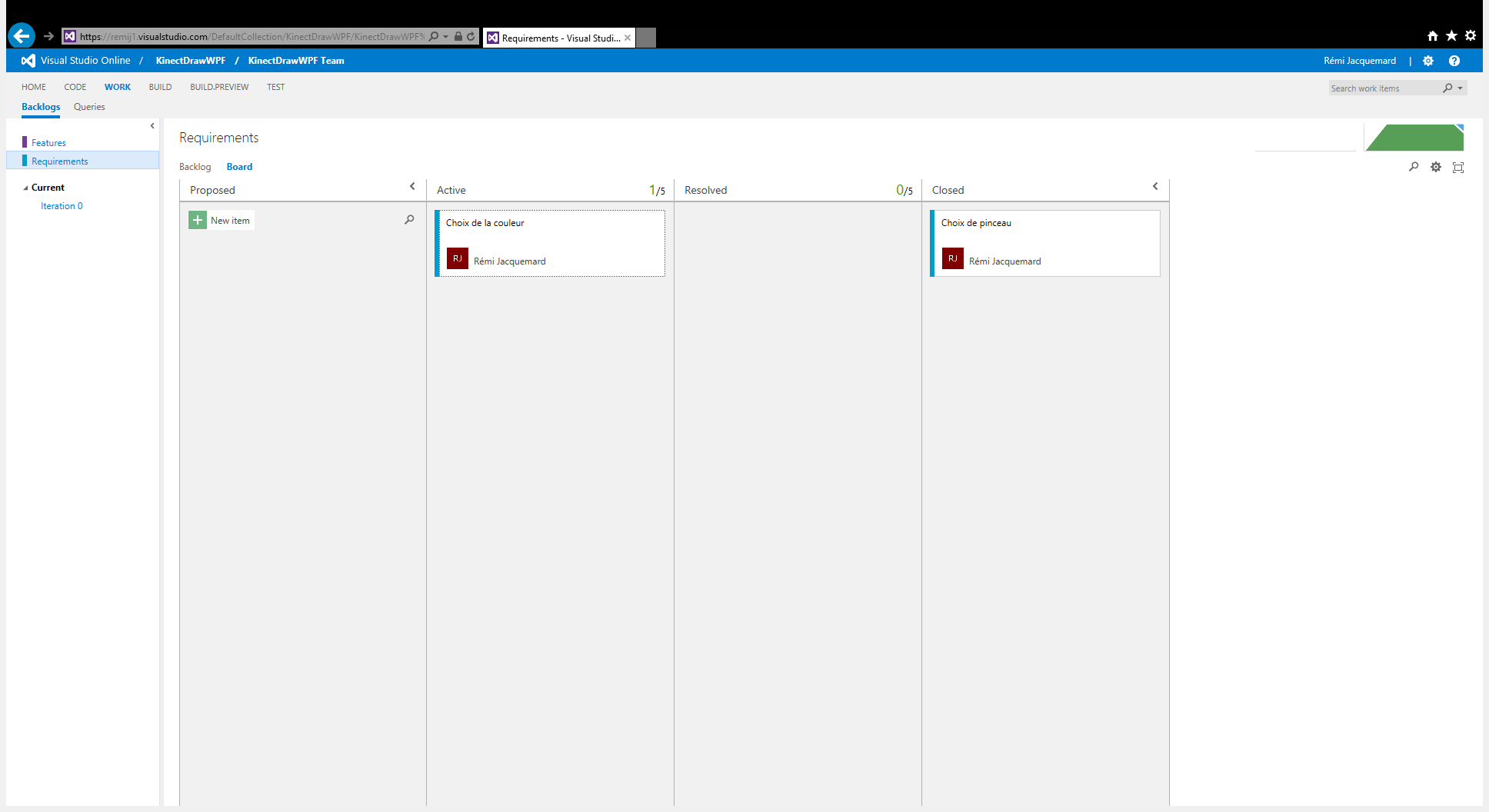


Figure . : Le Board de Visual Studio Online

Il permet de créer des tâches à la volée, et de les déplacer dans la bonne colonne en les glissant. Elles sont directement disponibles dans *Visual Studio*:

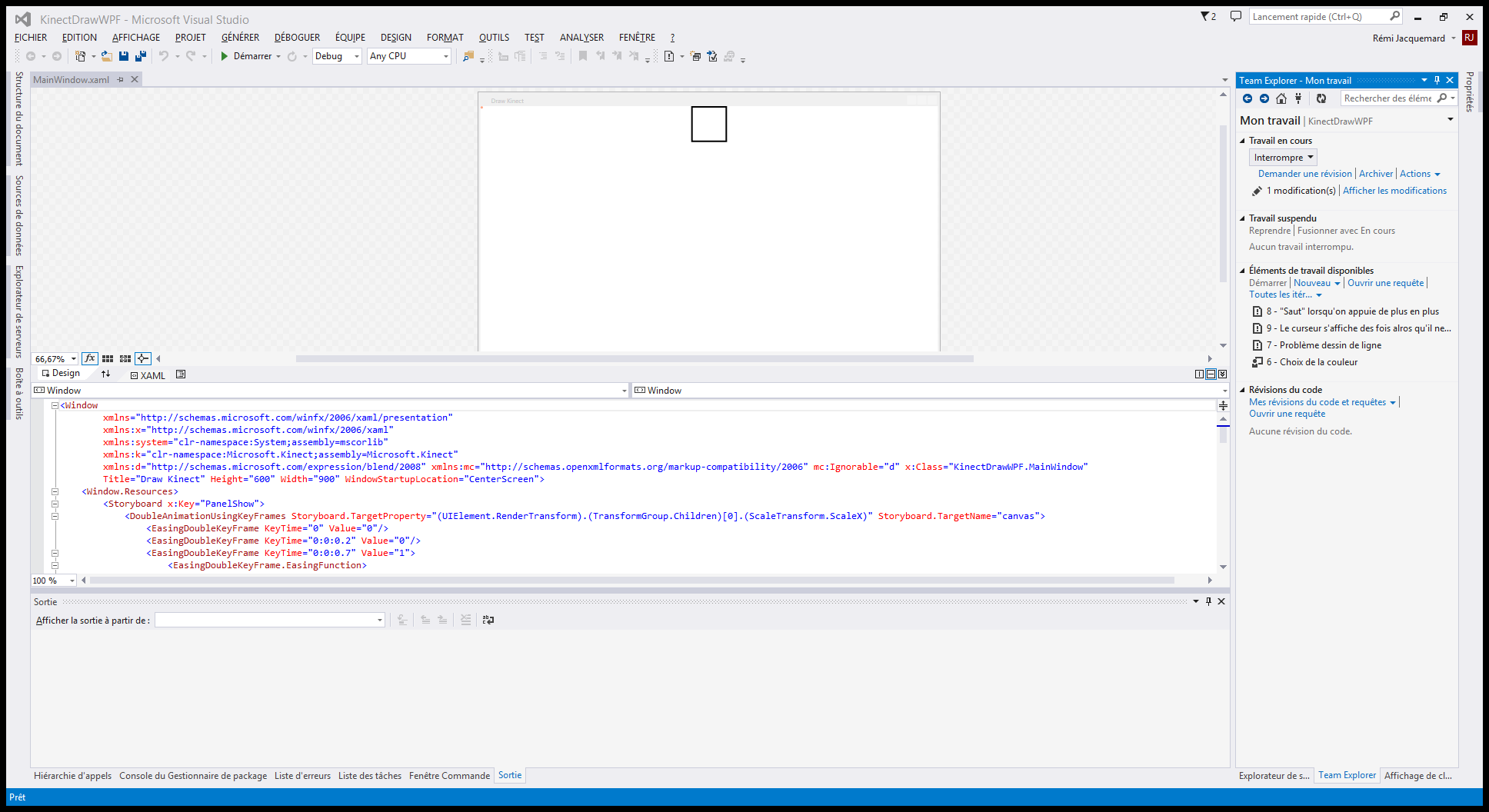


Figure . : Les tâches créées dans Visual Studio Online sont disponibles dans Visual Studio

### Points relatifs aux projets

Information : les projets Kinect Draw et Kinect PowerPoint ont été réalisés dans le cadre des pré-TPI

Le code effectué avant le TPI a été repris et mis à jour pour le tutoriel.

Dans un premier temps, les projets effectués ont été copiés dans un dossier portant leurs noms, et finissant par *– final*. Ce code est une proposition de correction du projet du tutoriel, qui contient les fonctionnalités demandées.  
Les en-têtes de fichiers ont été créés. Les commentaires écrits en anglais ont été traduits en français, et de façon claire, de telle sorte que la compréhension soit meilleure par l’élève.   
Le code des fonctionnalités non décrites dans le tutoriel a été supprimé.

Une fois le code final réalisé, les dossiers *finaux* des projets ont été copiés et renommés en *Kinect Draw – base* et *Kinect PowerPoint – Base*. Ces dossiers contiennent les projets de base qui sont fournis à l’élève. Du code a donc été supprimé et remplacé parfois par des commentaires du type *À COMPLÉTER*. Cela permet à l’élève de connaitre l’emplacement des lignes de code qu’il aura à écrire tout en respectant la structure du projet.

Certaines lignes de codes ont été intégrées au tutoriel lorsque cela était nécessaire et apportait une plus-value.

#### Kinect Draw

Très peu de difficultés sont apparues lors de la création et mise à jour de l’ancien code de ce projet pour le tutoriel.

La classe CursorPosition

Il ne sera pas expliqué ici en profondeur comment fonctionne les applications Windows store. Cependant, la création d’une classe contenant la position du curseur, la classe *CursorPosition*, a été nécessaire. Elle implémente l’interface *INotifyPropertyChanged*, ce qui permet que le curseur bouge à l’écran lorsque ses valeurs *PosX* et *Poissy* sont modifiées, telle qu’il est décrit dans le tutoriel, au chapitre *petite introduction à la structure de ce projet Windows 8.1*.

Le choix a été fait que cette classe soit interne. Cela simplifie la compréhension de l’élève du tutoriel.

Kinect PowerPoint

Quelques difficultés sont apparues avant tout lors de la création initiale de ce projet. Ces problèmes sont apparus, car c’était 1 ADP in PowerPoint ; ils ne seraient pas apparus si c’était un programme *WPF*.

Comme expliqué quelque peu dans le chapitre *Introduction au développement de cet ADP in PowerPoint* dans le tutoriel, un problème d’accès à des dll est apparu. Voici l’extrait du tutoriel le relatant :

|  |
| --- |
| **Le déploiement d’ADP in Office**  Ci-dessous, les principales étapes de l’ajout de 1 ADP in à un logiciel Office sont vulgairement décrites :   1. **Compilation de l’ADP in**: on le trouve maintenant sous forme de fichiers *.dll*. 2. **Ajout d’une clé de registre**: une clé indiquant le chemin de l’ADP in Office est ajoutée. Par exemple, dans ce projet, on trouve la clé : *HKEY\_CURRENT\_USER\Software\Microsoft\Office\PowerPoint\Addins\PowerPointKinect*, dont la valeur *Manifest* pointe vers les fichiers compilés. 3. **Chargement de l’ADP in**: au démarrage de *PowerPoint*, cette clé est lue et charge l’ADP in. Pour ce faire, chaque fichier *dll* est copié dans des sous-dossiers, qu’on peut trouver dans *%localappdata%[[1]](#footnote-1)\assemblée\dl3* et charge ceux-ci en mémoire.   Les fichiers *.dll* dans le dossier *dl3* sont isolés les uns des autres, ce qui pose problème pour l’implémentation de *Kinect* dans PowerPoint. En effet, le dossier *vbtechs* en devient inaccessible.  Ouvrir le fichier *KinectPowerPoint.cs*  Observer la méthode *initProjectFiles()*  Cette méthode permet de résoudre cette difficulté. Elle est appelée au démarrage de l’application et copie les fichiers nécessaires au bon emplacement pour que l’ADP in puisse détecter des mouvements. Elle doit être laissée telle quelle. Elle définit aussi la chaine de caractère *this.originalPath*, qui détermine le chemin du dossier contenant les fichiers originaux, soit l’emplacement des fichiers non situé dans *%localappdata%*. On y trouve donc tous les fichiers qui sont inclus dans le projet. |

### Les projets effectués avant les TPI

Avant les TPI, des projets de tests ont été créés. C’est deux de ceux-ci qui ont été repris dans le tutoriel. Comme dit précédemment, des fonctionnalités ont été supprimées. C’est le cas des deux projets.

Concernant *Kinect Draw*, un projet similaire avait été effectué. Celui-ci utilisait une fenêtre WPF[[2]](#footnote-2), et possède des fonctionnalités supplémentaires telles que le choix de couleur.

En ce qui concerne *Kinect PowerPoint*, un projet possédant des fonctionnalités non décrites dans le tutoriel a aussi été créé. Celui-ci apporte en plus un ruban s’intégrant dans PowerPoint, l’affichage d’un pointeur, et la possibilité de modifier les transitions entre les slides pour que ce soit plus proche du geste de balayèrent.

On parle rapidement de ces fonctionnalités dans le tutoriel, qui sont présentées comme des idées pour continuer le développement.

Beaucoup d’autres petits projets ont eux aussi été développés. Ce sont avant tout des projets de tests avec très peu de fonctionnalités et a été réalisé dans le but de découvrir l’API de Kinect. On trouve aussi des projets d’exemple fournis par Microsoft, et disponibles du logiciel *BrowserSDK*.

Tous ces codes sont inclus en annexe, afin que, si le projet était repris par un tiers, il possède tous les fichiers produits, et ce même si certains points sont mal commentés ou mal documentés. Ce sujet spécifique a été discuté avec le chef de projet.

## Modifications

|  |  |
| --- | --- |
| 19.05.2015 | Rédaction : Fonctionnalité source de Word |
| Au début de la rédaction du tutoriel, la fonctionnalité source de Word a été testée, dans le but d’augmenter la productivité, cependant, cette solution posait des problèmes de mises en pages et une perte de temps, c’est pourquoi elle a été abandonnée | |
| 22.05.2015 | CDC et planning : Replanification |
| Après discussion du jour avec M. Chenaux, il a été remarqué que le temps manquerait et que le planning ne pourrait pas être suivi. C’est pourquoi certains points du cahier des charges ont été modifiés. cf. : CdC, spécification et planning V2  En conséquence, certains points de l’analyse n’apparaîtront pas dans le tutoriel | |

* Historique des modifications demandées (ou nécessaires) aux spécifications détaillées.  
  – Date, raison, description, etc.

# Tests

## Dossier des tests

|  |  |
| --- | --- |
| 19.05.15 + 22.05.15 | Relecture du tutoriel |
| Tutoriel lu par Nicolas Salvador jusqu’à « PHIZ » compris En conséquence : Correction de fautes d’orthographe, mise à jour de design, modification de tournure de phrase | |

PARLER DES TESTS NON EFFECTUES

Plus de tests !!!!

* On dresse le bilan des tests effectués (qui, quand, avec quelles données?) sous forme de procédure (tableau).
* Si des tests prévus dans la stratégie n'ont pas pu être effectués:   
  – raison, décisions, etc.
* Liste des bugs répertoriés avec la date de découverte et leur état:  
  – Corrigé, date de correction, corrigé par, etc.

# Conclusion

## Bilan des fonctionnalités demandées

Comme on l’a vu dans le chapitre *Planification*, et le chapitre *1.1.6 Modification du CDC*, certaines fonctionnalités initialement prévues non pas pu être implémentée. Ceci est dû principalement à un manque de temps.

Voici un extrait du cahier des charges initiales indiquant les fonctionnalités recommandées :

* L'utilisateur doit pouvoir en suivant le tutoriel:
  + obtenir un historique sur le périphérique Kinect.
  + Comprendre le principe de fonctionnement de Kinect.
  + Installer un poste de travail permettant de suivre le tutoriel.
* L'utilisateur doit pouvoir créer un projet :
  + qui lui permet de dessiner de manière simplifiée sur un écran.
  + Pour cela il faut intégrer des explications sur l’utilisation de la classe KinectCore Window.
* L'utilisateur doit pouvoir créer un projet :
  + qui lui permet de créer 1 ADP in pour Microsoft PowerPoint.
  + Pour cela il faut intégrer des explications sur la création d’ADP in sur Microsoft PowerPoint, des explications sur la détection de mouvements (logiciel KinectStudio, VisualGestureBuilder, gestion de mouvement dans une application).
  + L’ADP in doit permettre de passer d’une slide à un autre et de pointer à l’aide du bras ou la main afin de faire apparaître un pointeur sur la slide, ce pointeur simule un pointeur laser.
* L'utilisateur doit pouvoir créer un projet:
  + qui lui permet de jongler avec un objet à l’écran (ballon par exemple), la personne peut être représentée de manière simplifiée à l’écran.
* Il convient de produire une documentation pour l’utilisateur qui lui permette de travailler de manière autonome avec le périphérique Kinect version 2 de Microsoft.

Les chapitres *Introduction, mise en place du poste de travail* et *Découverte de Kinect et son API* du tutoriel ont été rédigé. Ces points du cahier des charges ont donc été réalisés.

Le projet *Kinect Draw* a été inséré dans le tutoriel. Celui-ci permet de dessiner de manière simplifiée sur un écran. La classe KinectCoreWindow a elle aussi été décrite. Les points du CdC correspondants ont été traités.

Le projet *Kinect PowerPoint* a été inséré dans le tutoriel. Il décrit notamment brièvement la structure de 1 ADP in pour Microsoft PowerPoint. Ce point du CdC a été traité. Il aurait été possible d’expliquer plus profondément le fonctionnement de 1 ADP in pour PowerPoint. Mais le temps a manqué. De plus, l’élève suivant le tutoriel s’attend à apprendre comment développer pour Kinect, et aborder ce sujet profondément aurait pu nuire à sa motivation.  
La détection de mouvements a été abordée dans le tutoriel. L’utilité des logiciels *KinectStudio* et *Visual Gesture Builder* a été expliquée. Le processus complet à effectuer pour la détection de mouvement a été décrit dans son ensemble, son concept a été abordé. Cependant, la partie *Réalisation* correspondante a été mise de côté, cela étant dû à la modification du CdC (*1.1.6)*. L’élève sait donc ce qu’on *peut* faire avec les logiciels *Kinect Studio* et *Visual Gesture Builder*, mais il n’a pas été expliqué *comment* le faire. Le logiciel *Visual Gesture Builder* n’étant pas facile à aborder, un temps supplémentaire total d’environ 20 heures devrait être probablement effectué pour terminer ce point  
Ce projet permet de passer d’une slide à l’autre avec le bras. Cependant, comme vue dans le CdC modifié, la fonctionnalité du pointeur n’a pas été intégrée au tutoriel. En effet, le temps manquait, et même si le principe de base du pointeur est basé sur le projet *Kinect Draw*, son implémentation dans PowerPoint est laborieuse, à cause des limitations de l’API de l’ADP in Office. Pour expliquer tous les points consciencieusement, un temps approximatif d’une semaine en plus aurait sans doute été nécessaire.

Un projet *Kinect Ball* avait été prévu, qui aurait permis de jongler avec un objet à l’écran, comme le demandait le CdC. Cependant, par manque de temps, ce projet a été abandonné, comme vu dans le chapitre *1.1.6 Modification du CDC*. Pour analyser, réaliser et documenter ce projet dans le tutoriel, un temps supplémentaire de 3 semaines aurait peut-être été nécessaire.

Globalement, le tutoriel a été rédigé dans son ensemble. Même si certains points ont dû être raccourcis ou supprimés, l’élève ayant suivi le tutoriel devrait être capable d’avoir une bonne vue d’ensemble de l’API de Kinect, et de continuer son apprentissage par lui-même. L’un des derniers chapitres du tutoriel, *pour aller plus loin…*, indique notamment quelques liens et idées pour continuer son apprentissage.

* Il s’agit de reprendre point par point les fonctionnalités décrites dans les spécifications de départ et de définir si elles sont atteintes ou pas, et pourquoi.
* Si ce n’est pas le cas, mesurer en « % » ou en « temps supplémentaire » le travail qu’il reste à accomplir pour terminer le tout.

## Bilan de la planification

Ci-dessous, voici l’état du projet au 04.06.2015 présentant les heures planifiées et réalisées :

|  |  |  |
| --- | --- | --- |
| Tâches | # ¼ d’heure | # d’heure |
| Prise de connaissance du projet | *6* | *1,5* |
| **6** | **1,5** |
| Information | *3* | *0,75* |
| **2** | **0,5** |
| Planification du projet | *12* | *3* |
| **22** | **5,5** |
| Analyse - Tutoriel | *0* | *0,0* |
| **0** | **0,0** |
| Analyse - Tutoriel : Structure | *3* | *0,75* |
| **12** | **3** |
| Analyse - Tutoriel : Difficulté du code | *1* | *0,25* |
| **0** | **0,0** |
| Analyse - Projets | *0* | *0,0* |
| **0** | **0,0** |
| Analyse - Projets : Technologie de l'API à utiliser | *0* | *0,0* |
| **0** | **0,0** |
| Analyse - Projets : Diagramme de classe | *3* | *0,75* |
| **0** | **0,0** |
| Analyse - Projets : Fonctionnalités | *0* | *0,0* |
| **0** | **0,0** |
| Décision - Tutoriel | *1* | *0,25* |
| **0** | **0,0** |
| Décision - Projets | *0* | *0,0* |
| **0** | **0,0** |
| Réalisation/Documentation - Tutoriel | *0* | *0,0* |
| **0** | **0,0** |
| Réalisation/Documentation - Tutoriel : Design | *12* | *3* |
| **9** | **2,25** |
| Réalisation/Documentation - Tutoriel : Structure | *9* | *2,25* |
| **0** | **0,0** |
| Réalisation/Documentation - Tutoriel : Contenu | *201* | *50,25* |
| **206** | **51,5** |
| Réalisation - Développement : Kinect Draw | *6* | *1,5* |
| **9** | **2,25** |
| Réalisation - Développement : Kinect PowerPoint | *6* | *1,5* |
| **12** | **3** |
| Réalisation - Développement : Kinect Ball | *0* | *0,0* |
| **0** | **0,0** |
| Documentation | *97 (73)* | *24 (18.5)* |
| **58** | **14,5** |
| Évaluation | *24 (0)* | *6 (0)* |
| **0** | **0,0** |

À gauche du tableau, on voit les catégories de tâches. Dans les deux colonnes de droite, on trouve le nombre de quart d’heure et d’heure entière.   
Les valeurs en italique correspondent au temps planifié, et celles en gras à la réalisation effective.

Les entiers entre parenthèses indiquent la valeur réelle du nombre de quarts d’heure planifiés jusqu’au 04.06.2015 compris. Les jours du 05.06.2015 et 08.06.2015 ont été omis dans ces valeurs, pour un meilleur bilan de la planification.

Dans un premier temps, il avait été prévu près de 43 heures de travail sur le tutoriel en lui-même. Dans la deuxième version du planning, environ 50 heures étaient prévues au total, dont 6 heures utilisées comme *tampon*, via la *tâche Finalisation du tutoriel*. Au final, environ 53 heures auront été nécessaires à la rédaction du tutoriel. On remarque de suite la quantité de travail qui a été nécessaire de fournir pour le tutoriel. Heureusement que les 6 heures de tampon ont été prévues, celles-ci ayant été totalement utilisées. On peut dire que la deuxième version de la planification a été suivie, et les jalons principaux respectés.

Créer une deuxième version du planning a été nécessaire. Toutefois, cette planification a pris du temps. Une journée entière de travail (6 heures le 22.05.2015) a été nécessaire pour réfléchir aux points du CdC à modifier, pour discuter avec le chef de projet, modifier le rapport de projet et créer un nouveau planning en conséquence. Malgré tout, la nouvelle planification a permis de finir dans les temps.

Selon le journal de travail, le rapport de projet n’aurait pris environ qu’une quinzaine d’heures à être réalisé. Cependant, ce rapport a souvent été écrit par morceau de 5 minutes. On estime le temps destiné à sa rédaction à une vingtaine d’heures.

5 heures supplémentaires ont été effectuées. Elles sont indiquées dans le journal de travail. En effet, il a parfois fallu travailler à domicile, ou du travail a été effectué lorsque du temps libre était disponible en classe. Ces heures supplémentaires ont été utilisées pour la rédaction du rapport de projet, ou celle du tutoriel.

On remarque que certaines tâches n’ont finalement pas été utilisées. Cependant, à cause du modèle de planification fourni par l’ETML, l’ajout d’une catégorie de tâche ultérieurement à la première planification est compliqué, et il a été préféré qu’elles apparaissent déjà dans le planning.

Également, la tâche *Réalisation/Documentation - Tutoriel : contenu* contient une quinzaine d’heures travaillées. Il aurait été utile de créer des sous-tâches pour chaque partie du tutoriel pour une meilleure analyse finale.

## Bilan personnel

Deux versions du planning et du cahier des charges ont été nécessaires à ce TPI. Mais la cause en est un peu floue :

* Est-ce dû à une mauvaise planification initiale, à des mauvais choix prix dans l’ordre de rédaction du tutoriel ?
* Est-ce dû au tutoriel contenant trop de chapitres ou une mauvaise structure ? La direction décidée que prenait le tutoriel était-elle mal pensée et nécessitait-elle une modification ?
* Est-ce dû à trop de fonctionnalités dans le cahier des charges et une sous-estimation du temps qu’allait prendre chacune de celle-ci ?

Il est sans doute possible que tous ces points en aient été la cause, et je planifierai les prochains projets que je réaliserai en en prenant compte.

Ce point mis à part, ce TPI m’a beaucoup apporté personnellement.

Premièrement, j’ai eu la chance de pouvoir toucher à la programmation pour Kinect V2 dans le cadre des projets de fin d’années et des pré-TPI. Je n’avais aucune connaissance en programmation pour Kinect, et n’avais aucune idée de quoi cela en avait l’air. Maintenant, j’ai acquis une bonne compréhension globale des concepts qui se cachent derrière ce capteur. J’ai pu développer des projets de tests qui m’ont permis d’en apprendre beaucoup, même si ceux-ci n’ont pas été réalisés pendant le TPI en lui-même.   
De plus, l’ETML m’a apporté les moyens physiques pour commencer à développer. En effet, coder pour Kinect V2 nécessite le capteur et l’adaptateur pour PC, que je n’avais pas.

J’ai aussi rédigé un tutoriel complet pour la première fois, m’apportant beaucoup d’expérience à ce niveau. Se mettre à la place de quelqu’un d’autre (ici, un élève de l’ETML) et faire en sorte que celui-ci comprenne les concepts n’a pas été aisé. J’ai pu tester différents moyens pour y arriver, notamment l’utilisation de schéma ou d’insertion de morceau de code. La mise en page et la structure du tutoriel en lui-même sont aussi importantes, et ce sont des points auxquels j’ai pu réfléchir grâce à ce TPI.

J’ai pu aussi tester un nouvel outil de gestion de version : *Team Foundation*, via les serveurs de *Visual Studio Online*. J’en ai été globalement convaincu, et l’utiliserai sans doute pour mes prochains projets développés au sein de *Visual Studio*.

J’ai pu remarquer qu’on connait mal le capteur Kinect, ses possibilités et ses inconvénients. Par exemple, pouvoir dessiner à l’écran avec Kinect (Projet *Kinect Draw*) à l’air au premier abord d’être une idée à exploiter. Cependant, après avoir testé le projet quelquefois, on remarque que son utilisation peut devenir très vite fastidieuse : la détection de la main peut être imprécise, des erreurs peuvent survenir, et surtout, il devient difficile de garder son bras levé au bout de quelques secondes. Ce projet est donc amusant, mais en devient inutilisable.

Concernant le projet *Kinect PowerPoint* dans sa version complète, c’est-à-dire que le pointeur peut être utilisé, j’en suis globalement satisfait. Après l’avoir testé quelquefois, je pense que Kinect, et ce type de technologie en général, est idéal pour des interactions qui doivent être rapides et efficaces. Kinect devient ici une aide, un outil, et n’est pas trop présent dans l’interaction. Dans le cadre du projet Kinect PowerPoint, si la détection fonctionne mal, il est toujours possible d’utiliser un autre moyen pour changer de slide, comme les touches du clavier. Kinect devient facultatif : on ne l’utilise que s’il facilite l’interaction.

* Si c’était à refaire:  
  – Qu’est-ce qu’il faudrait garder ? Les plus et les moins ?  
  – Qu’est-ce qu’il faudrait gérer, réaliser ou traiter différemment ?
* Qui est que ce projet m’a appris ?
* Suite à donner, améliorations souhaitables …
* Remerciements, signature, etc.

# Divers

## Journal de travail de chaque participant

cf. : journal de travail annexé au format numérique pour l’économie de papier. La mise en annexe de manière digitale a été discutée avec M. Chenaux. Webographie

## Sources

Concernant les sources utilisées pour le tutoriel, se référer au chapitre « source » de celui-ci.

Image

**Page de garde** : <http://www.gizmos.es/accesorios/kinect-for-windows-v2-ya-esta-en-proceso-de-reserva.html>

# Annexes

Annexes, au format papier :

* Tutoriel : « Apprendre à développer pour Kinect V2 »

Annexes, au format numérique :

* Journal de travail
* Le code source complet,

À COMPLÉTER EN FONCTION DU CD

* Listing du code source (partiel ou, plus rarement complet)
* Guide(s) d’utilisation et/ou guide de l’administrateur
* État ou « dump » de la configuration des équipements (routeur, switch, robot, etc.).
* Extraits de catalogue, documentation de fabricant, etc.
* Photocopies diverses, etc.

1. *%localappdata%* correspond, la plupart du temps, à *C:\Users\[username]\AppData\Local* [↑](#footnote-ref-1)
2. *WPF*: *Windows Presentation Foundation*. C’est un type et une structure d’application disponible via *.net*. Son fonctionnement est proche des applications Windows Store, utilisant le XAML. [↑](#footnote-ref-2)