Création d'un tutoriel Kinect V2

Rémi Jacquemard – Fin2
ETML - Lausanne
Du lundi 11 mai au lundi 8 juin 2015
Chef de projet : Patrick Chenaux

Table des matières

1 Spécifications

1.1 Spécifications de départ et documentation associée

Voici le cahier des charges dans sa version initiale (le cahier des charges original est disponible en annexe) :

1.1.1 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

1.1.2 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 un add-in pour Microsoft PowerPoint.
  - pour cela il faut intégrer des explications sur la création d'add-in sur Microsoft PowerPoint, des explications sur la détection de mouvements (logiciel KinectStudio, VisualGestureBuilder, gestion de mouvement dans une application).
  - l'add-in doit permettre de passer d'un slide à un autre et de pointer à l'aide du bras ou la main afin de faire apparaître un pointeur sur le 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.

1.1.3 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 doivent 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.

1.1.4 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.

1.1.5 Méthodes de validation des solutions

- A définir par le candidat.

1.1.6 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 un add-in pour Microsoft PowerPoint.
  - pour cela il faut intégrer des explications sur la création d'add-in sur Microsoft PowerPoint, des explications sur la détection de mouvements (logiciel KinectStudio, VisualGestureBuilder, gestion de mouvement dans une application).
  - l'add-in doit permettre de passer d'un slide à un autre et de pointer à l'aide du bras ou la main afin de faire apparaître un pointeur sur le slide, ce pointeur simule un pointeur laser.

Deux points ont étés 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 tout de même possible d'expliquer de quelle manière créer ce fichier.

D'autre part, malgré le fait que pointer à l'aide du bras ou la main sur le 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 via un add-in, et l'utilisation d'un *workaround* rend cette fonctionnalité compliquée à décrire (un objet en forme d'ellipse doit être 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, etc.). 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

2 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 les fonctionnalités réalisées dans le cadre des pré-TPI ne sont pas décrites dans le tutoriel.

2.1 Planning(s) initial

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

Module :	TPI
Prénom - nom :	Rémi Jacquemard
Classe :	FIN2
Lieu :	ETML - Lausanne

Date Début :	lundi 11 mai 2015
Date Fin:	lundi 8 juin 2015
Nombre de semaines:	5
Nombre de Périodes/Semaine :	36
Nombre de 1/4 heure/Période:	3

Voici un bref 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
Evaluation	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.

2.2 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 bref 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
Evaluation	24	6
Total planifié, jours fériés non compris	**384**	96

On remarque les changements majeurs suivants :

- Le nombre d'heure 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'heures 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.

3 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.

3.1 Faisabilité

La plupart des points concernant la faisabilité du projet ont été abordé 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 minimal 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-Core 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 notion spécifique au développement Kinect. Indirectement, cela a permi 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 crées avant le TPI seront incorporés au tutoriel. Cependant, les fonctionnalités présentes ne seront 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 le capteur 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.

3.2 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*

3.2.1 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'addin pour PowerPoint
  - Logiciel KinectStudio (permet 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-t-elle l'ancienne version, les fonctionnalités présentes, quelques jeux déjà développés pour Kinect, etc.). Ce chapitre servira aussi de mise-en-bouche.
   - **A qui se destine ce tutoriel ?**
     On indiquera dans ce sous-chapitre à qui se destine le tutoriel (élève de l'ETML en informatique en 3ème ou 4ème 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 fournis, 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 d'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 utilisable 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 simplifié 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**
   Dans ce chapitre, on développera en guidant l'utilisateur un addin pour PowerPoint qui permettra, tel que définis dans le cahier des charges, de passer d'un slide à un autre et de pointer à l'aide du bras ou la main afin de faire apparaître un pointeur sur le slide, ce pointeur simule un pointeur laser. Voici ce qu'il contiendra :
   - Présentation du projet
   - Présentation du logiciel KinectStudio, qui permet de se filmer. 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 d'un add-in pour PowerPoint
   - Présentation des classes *VisualGestureBuilderFrame*, qui permettent de détecter certains mouvements de personnes 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 seront nécessaires 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, tels 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 pré-TPI

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. Ceci a l'avantage que l'utilisateur comprendrait la grande majorité du code. Cependant, le développement d'application Windows Store est sensée être nouvelle pour lui, et est assez compliquée 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.

3.3 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 spécifiquement pour le tutoriel. Les concepts principaux de ces projets y sont explicités. De plus, étant des projets d'introduction, la difficulté du code n'est pas excessive.

3.4 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 parfois l'interaction)
- Tests unitaires : cependant, par manque de temps, il sera peu probable de pouvoir en créer

4 Réalisation

4.1 Dossier de Réalisation

4.1.1 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.

4.1.2 La structure du tutoriel

Le tutoriel a été réalisé en grande partie tel que l'analyse le demandait. En voici la structure des chapitres finaux :

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*. Cela 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.

4.1.3 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, entres 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).

4.1.4 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 plus utile pour des sources dont on veut insérer une citation.

4.1.5 L'archive fournie

Pour rappel, après analyse, il a été décidé qu'une archive contenant les projets de bases *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 réseau *k:*. Il aurait été possible d'indiquer ainsi le chemin de cet archive directement dans le tutoriel.

Après discussion avec le chef de projet, il a été décidé que l'archive sera sur un CD, et sera indiqué tel quel dans le tutoriel.

4.1.6 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 development guidelines*, disponible pour les élèves sur le réseau, ainsi qu'en annexe. 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** préfixées, mais restent explicites :
  Exemple : titleLabel (de type *Label*), name (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.

4.1.7 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 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 apparaissent dans le tutoriel :

Atteindre la méthode *initKinect()*

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

4.1.8 Gestion des versions

Information : l'outil décrit ici a été utilisé principalement dans le cadre des pré-TPI. 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 intéressant de cet outil est le tableau (*Board*), comme on le voit sur la figure ci-dessous:

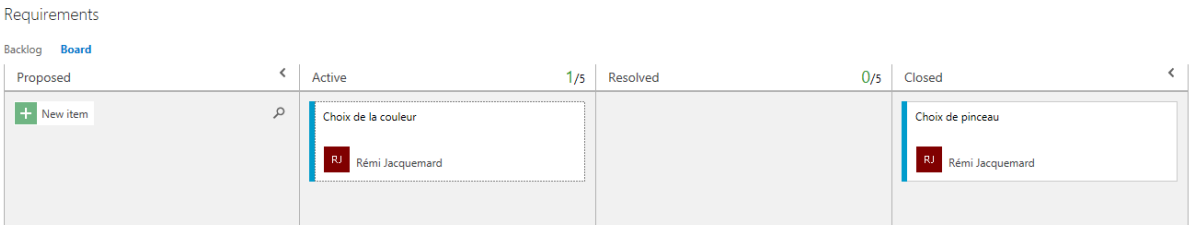

Figure 4.1 : Le Board de Visual Studio Online

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

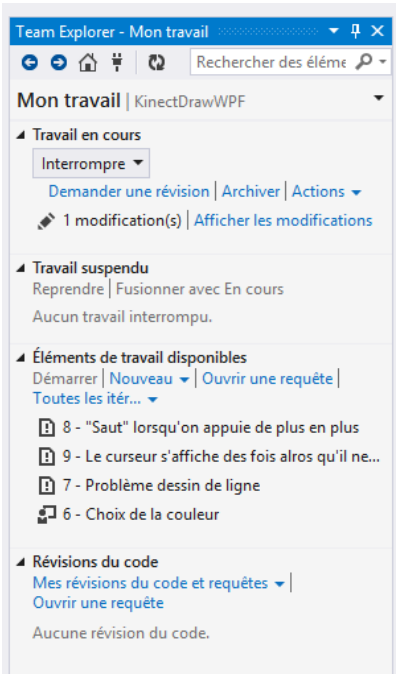

Figure 4.2 : Les tâches créées dans Visual Studio Online sont disponibles dans Visual Studio

4.1.9 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 leur nom, 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 traduit en français, et de façon claire, de tel 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 *Final* des projets ont été copiés et renommés en *Kinect Draw – Base* et *Kinect PowerPoint – Base*. Ces dossiers contiennent les projets de bases qui sont fournis à l'élève. Du code a donc été supprimé et remplacé parfois par des commentaires du type *A COMPLETER*. 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 code ont été incorporées au tutoriel lorsque cela était nécessaire et apportait une plus-value.

Kinect Draw

Très peu de difficultés sont apparues lors de la création et de la mise à jour de l'ancien code 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 *PosY* sont modifiées, tel 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 du tutoriel par l'élève.

Kinect PowerPoint

Quelques difficultés sont apparues avant tout lors de la création initiale de ce projet. Ces problèmes sont dus au fait que c'était un add-in PowerPoint ; ils ne seraient pas apparus si c'était, par exemple, un programme *WPF*.

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

> **Le déploiement d'add-in Office**
>
> Ci-dessous, les principales étapes de l'ajout d'un add-in à un logiciel Office sont vulgairement décrits :
>
> 1. **Compilation de l'add-in** : on le trouve maintenant sous forme de fichiers *.dll*.
> 2. **Ajout d'une clé de registre** : une clé indiquant le chemin de l'add-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'add-in** : au démarrage de *PowerPoint*, cette clé est lue et charge l'add-in. Pour ce faire, chaque fichier *dll* est copié dans des sous-dossiers, qu'on peut trouver dans *%localappdata%*[1]*\assembly\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 innaccessible.
>
> 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'add-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.

4.1.10 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 pour les deux projets.

Concernant *Kinect Draw*, un projet similaire avait été effectué. Celui-ci utilisait une fenêtre WPF[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 balayement.

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 qui ont été réalisés dans le but de découvrir l'API de Kinect. On trouve aussi des projets d'exemple fournis par Microsoft, et disponible de le logiciel *BrowserSDK*.

---

[1] *%localappdata%* correspond, la plupart du temps, à *C:\Users\[username]\AppData\Local*
[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.

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 point a été discuté avec le chef de projet.

4.2 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.	

5 Tests

5.1 Dossier des tests

Voici quelques tests effectués en ce qui concerne le tutoriel :

19.05.15 + 22.05.15	Relecture du tutoriel
Tutoriel lu par Nicolas Salvadore jusqu'à « PHIZ » compris. En conséquence : Correction de fautes d'orthographe, mise à jour du design, modification de tournure de phrase.	
03.06.15	Relecture du développement des projets du tutoriel
Suivis point par point des étapes et instructions de la réalisation des projets *Kinect Draw* et *Kinect PowerPoint*. Les manipulations sont effectuées de la même manière que le demande le tutoriel. En conséquence : Correction de fautes d'orthographe, ajout de petites informations sur les instructions.	
03.06.15	Relecture du tutoriel
Le tutoriel a été relu par plusieurs personnes. En conséquence : Correction de fautes d'orthographe et de grammaire, mise à jour du design, modification de tournure de phrase.	
04.06.15	Relecture du rapport de projet
Le rapport de projet a été relu par plusieurs personnes. En conséquence : Correction de fautes d'orthographe et de grammaire, modification de tournure de phrase.	

Le tutoriel n'a pas pu être fourni à des élèves pour être testé, ceci dû au manque de temps et de matériel tel que décrit au chapitre *3.4 Conception des tests*.

Cependant, les divers documents ont été parcourus par plusieurs personnes. Quelques fautes d'orthographe et de grammaire ont pu être ainsi corrigées. De plus, la correction d'orthographe et de grammaire de Word a été utilisée.

5.1.1 Tests des projets

Rappel : les projets n'ont pas été réalisés dans le cadre du TPI

Les projets ont été testés tout au long de leur création. Ils ont été réalisés étape par étape, ainsi, chaque fonctionnalité a été testé. De plus, il ne sera décrit ici que les tests concernant les fonctionnalités des projets inclues dans le tutoriel.

Kinect Draw

Les points qui ont été testés sont les suivantes :

- Pouvoir démarrer l'application sans erreur
- Pouvoir déplacer le curseur de la main droite
- Pouvoir commencer à dessiner en poussant le bras vers l'avant, et arrêter lorsqu'on le ramène vers soi.
- Pouvoir dessiner avec un pinceau plus gros en poussant de plus en plus vers l'avant
- Aucune erreur ne doit apparaître

Chacun de ces points a été testé au fur et à mesure du développement.

Tout d'abord, le fait de déplacer le curseur a été codé, et les tests associés effectués :

- Lorsque la main est déplacée selon l'axe des X, le curseur se déplace de même. On peut atteindre le bord gauche et le bord droit de l'écran.
- Lorsque la main est déplacée selon l'axe des Y, le curseur se déplace de même. On peut atteindre le bord du haut et du bas de l'écran.

Puis, la fonctionnalité de pouvoir dessiner – avec un diamètre fixe – en avançant le bras vers l'avant a été développé, et les tests associés effectués :

- A partir d'une certaine distance de la main par rapport à l'épaule, le dessin s'effectue
- Des lignes se dessinent lorsqu'on bouge la main
- Lorsqu'on remmène le bras vers soi, on ne peint plus

La distance minimum de la main par rapport à l'épaule qui déclenche le dessin a été cherchée à tâtons. Au final, la distance idéale trouvée est de 0.5 mètres. (cf. tutoriel, chapitre *KinectDraw*)

Finalement, le fait de pouvoir dessiner de tailles différentes a été testé :

- Lorsqu'on commence à dessiner, le pinceau est petit. Plus on pousse le bras vers l'avant, plus le pinceau est gros

Lors des différents tests, aucune erreur n'a été détectée.

Kinect PowerPoint

Les points qui ont été testés sont les suivantes :

- Pouvoir démarrer l'application sans erreur
- Pouvoir changer de slide avec chacune des mains, et dans les deux sens
- Aucune erreur ne doit apparaître

L'application se lance sans erreur. On peut aussi changer de slide :

- Avec le bras droit, on peut passer au slide suivant
- Avec le bras droit, on peut passer au slide précédent
- Avec le bras gauche, on peut passer au slide suivant
- Avec le bras gauche, on peut passer au slide précédent

Globalement, cette fonctionnalité fonctionne bien. Cependant, lors de ces tests, il a été remarqué que certains gestes sont parfois mal détectés. C'est par exemple le cas pour passer au slide suivant avec la main gauche, qui est moins bien détecté que les autres. Cependant, ce geste n'est pas souvent utilisé, car peu intuitif.
En effet, pour passer au slide suivant, le bras droit est utilisé dans la plupart des cas. De la même manière, le bras gauche est utilisé pour revenir au slide précédent. Ainsi, que ce soit pour un gaucher ou un droitier, l'add-in fonctionne la plupart du temps.

6 Conclusion

6.1 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 implémentées. Ceci est principalement dû 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 un add-in pour Microsoft PowerPoint.
  - pour cela il faut intégrer des explications sur la création d'add-in sur Microsoft PowerPoint, des explications sur la détection de mouvements (logiciel KinectStudio, VisualGestureBuilder, gestion de mouvement dans une application).
  - l'add-in doit permettre de passer d'un slide à un autre et de pointer à l'aide du bras ou la main afin de faire apparaître un pointeur sur le 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 rédigés. 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 d'un add-in pour Microsoft PowerPoint. Ce point du CdC a été traité. Il aurait été possible d'expliquer plus profondément le fonctionnement d'un add-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 profondément le sujet du développement d'add-in aurait pu nuire à sa motivation.
La détection de mouvement 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 décrite dans son ensemble, et 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'un slide à l'autre avec le bras. Cependant, comme vu 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'add-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 demandais le CdC. Cependant, 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 suivis 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.

6.2 Bilan de la planification

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

Tâches	# ¼ d'heures	# d'heures
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**
Evaluation	*24 (0)*	*6 (0)*
	0	**0,0**

A 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 réel du nombre de quart d'heure planifiée 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 au jour de la rédaciton.

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 était prévue 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 heure 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 permi 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 étés 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 compliquée, 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.

6.3 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 pris dans l'ordre de rédaction du tutoriel ?
- Est-ce dû au tutoriel contenant trop de chapitres ou une mauvaise structure ? La direction 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 d'extrait de code. La mise en page et la structure du tutoriel en lui-même est aussi importante, et ce sont des points auxquels j'ai pu réfléchir grâce à ce TPI.

J'ai aussi pu tester un 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 quelques fois, 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 réalisée avant les TP, soit la version dont le pointeur a été implémenté, j'en suis globalement satisfait. Après l'avoir testé quelque fois, 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.

Bien entendu, ce tutoriel peut être amélioré sur de nombreux points. Beaucoup de concepts présents dans l'API de Kinect n'ont pas été expliqués. Cependant, et comme l'indique le bilan des fonctionnalités en page 22, je pense que l'élève, après avoir suivis ce tutoriel, est capable d'apprendre par lui-même de nouvelles fonctionnalités, les principaux concepts étant acquis.

7 Divers

7.1 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 numérique a été discutée avec M. Chenaux.

7.2 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

8 Annexes

Annexe, au format papier :

- Tutoriel : « Apprendre à développer pour Kinect V2 »

Annexes, au format numérique :

- Journal de travail V1 au format *xlsm*. L'extension *.old* a été ajoutée de telle sorte que la bonne version du journal de travail soit utilisée par défaut. (*X-TPI-jacquemare-JDT-V1.xlsm.old*)
- Journal de travail V2, contenant la dernière planification (*X-TPI-jacquemare-JDT-V2.xlsm*)
- La planification initiale
- Tutoriel : « Apprendre à développer pour Kinect V2 »
- Le dossier qui sera fourni à l'élève, disponible dans le dossier *TutorialFolder*
- Le dossier *ProjectFolder*, qui peut être utilisé comme tel pour une reprise du projet
- Le fichier *I-DevelopmentGuidelinesV1.1.0.pdf*, contenant les normes de l'ETML
- Le cahier des charges original (*H-TPI-PCX01-JaquemardRemi.pdf*)