Université de Neuchâtel

Master in Computer Science – BeNeFri

MULTIMODAL INTERFACE

Rapport de projet

HelpCopter

Professeurs : Omar Abou Khaled, Jacques Bapst, Denis Lalanne, Elena Mugellini

Etudiants : Patrick Brunner, Simon Brunner, Mahamat Abakoura

Table des matières

1 Introduction

Dans le cadre du cours "Multimodal Interfaces", un projet de fin de semestre doit être réalisé. L'application doit être de type divertissement ou de type jeu sérieux. C'est donc dans cette optique que le projet HelpCopter a vu le jour.

En effet, HelpCopter est un jeu qui consiste à aider des sinistrés en leur jetant des vivres et des médicaments. Le défit du jeu est notamment l'utilisation de deux modalités différentes : les gestes (avec la Wiimote) et la reconnaissance vocale (avec Speech). La Wiimote va servir principalement à guider l'hélicoptère. Quant à la voix, sa fonction première est d'effectuer des commandes. Il est également possible de jouer le clavier de l'ordinateur. Le but principal de l'application est d'immerger le jouer dans le jeu à l'aide des différentes modalités utilisées.

2 Description du projet

Le projet HelpCopter utilise deux modalités : les gestes et la reconnaissance vocale.

Le joueur est aux commandes d'un hélicoptère, c'est à l'aide la modalité gestuelle qui est représentée par *Wiimote* de Microsoft que l'utilisateur fait évoluer son engin au sein d'un monde virtuel. Les axes de rotation de la Wiimote (X, Y) permettent de faire bouger l'hélicoptère vers l'avant et vers l'arrière, ainsi que de glisser latéralement vers la gauche et vers la droite. Seuls les axes X et Y seront utilisés.

Pour la reconnaissance vocale, *Speech SDK version 5.1* est utilisé; il sert à effectuer certaines commandes vocales en utilisant les mots :

- Soins : pour larguer des caisses de médicaments.
- Vivres : pour larguer des caisses de vivres.

Le jeu consiste à larguer des vivres et des soins aux populations sinistrées qui sont représentées sous forme de villes. Durant la partie les villes apparaissent aléatoirement sur une carte. Si l'on ne largue pas des vivres à ces dernières dans le temps imparti, la ville disparait et le nombre de mort est comptabilisé. Dès que le nombre de morts maximal est atteint, le jeu se termine. Plus le joueur arrive à sauver des villes et plus long dure la partie. Pour gagner des points, il faut larguer les bonnes caisses sur les bonnes villes. Pour ce faire il faut ouvrir la soute au préalable. Une indication de l'état de la soute est affichée en haut à gauche de l'écran. Attention elle se referme automatiquement si le bouton est relâché. Les caisses de soins doivent tomber sur les villes claires, les caisses de vivres doivent tomber sur les villes foncées pour faire des points. Le nombre de mort augmente aussi si l'hélicoptère vole trop bas ou trop. Une indication s'affiche s'il faut remonter ou redescendre.

Attention La jouabilité du jeu demande pas mal de doigté car il est très sensible. Il est donc recommandé d'essayé d'effectuer de petit geste afin de ne pas perdre le contrôle. En effet l'hélicoptère possède une certaine inertie qui est à maitriser tout au long de la partie.

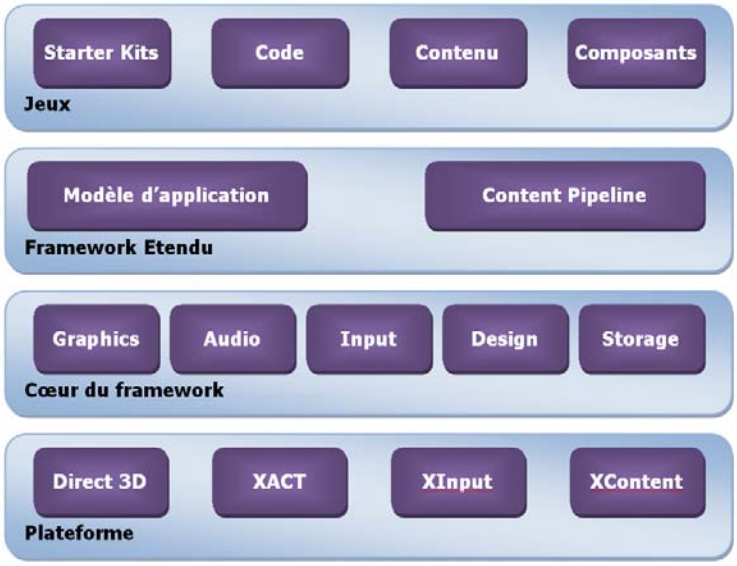
3 Technologies et outils de développement

Ce chapitre donne une explication concernant les différentes technologies utilisées au sein de l'application. Si le projet ne compile pas sous Visual Studio veuillez ne pas oubliez d'ajouter les libriaries se trouvant sous "CD\Project\lib\" qui sont les libriaries de WiimoteLib et de Speech.

3.1 XNA

XNA est un ensemble d'outils permettant de faciliter le développement de jeux vidéo pour : La XBOX 360, Windows Phone, Zune et PC. Il contient plusieurs outils tels qu'un framework spécialement conçu pour la création de jeux-vidéo. Il a aussi des outils d'intégration de contenus, qui permettent d'intégrer des médias tels que des sons, images, musiques etc. XNA possède une documentation très bien détaillé dans sa documentation.

Il faut savoir que le framework est disponible uniquement pour le langage de programmation C#. Il faut donc aussi utiliser un IDE supportant C# tel que Visual C# (express) ou Visual Studio.
L'architecture de XNA est en model de couches, une représentation de ces couches figure ci-dessous :



La couche " Plateforme " est la couche de bas niveau de XNA. Elle constitue l'un des points forts de XNA, puisque le développeur n'a pas besoin d'écrire du code "bas niveau". Cette couche permet d'accéder aux ressources de la machine notamment :

- La carte vidéo via le composant Direct 3D.
- La carte son via le composant XACT.
- Les contrôles (clavier, manettes) via le composant XInput.
- La gestion des ressources via le composant Xcontent.

La couche "Cœur du framework" représente, comme son nom l'indique, le noyau central de XNA. Elle contient l'ensemble des classes qui seront utilisées par le développeur. Ces classes constituent donc un intermédiaire entre le développeur et la couche " bas niveau " :

- Microsoft.Xna.Framework.Graphics permet l'accès aux graphismes 2D ou 3D.
- Microsoft.Xna.Framework.Audio permet la manipulation des sons.
- Microsoft.Xna.Framework.Input permet la gestion des divers contrôles du jeu (manette, clavier, etc.).
- Microsoft.Xna.Framework.Designer contient des classes "mathématiques" qui seront surtout utiles pour la 3D.
- Microsoft.Xna.Framework.Storage permet l'accès aux périphériques de stockage (disque dur d'un PC ou d'une Xbox 360 ou une carte mémoire)

La couche framework étendu est constituée de deux composants importants :

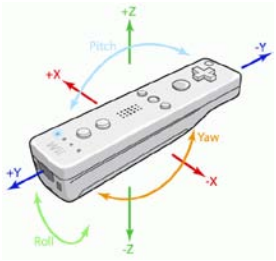
- Le modèle d'application permet de gérer l'architecture globale du jeu. Elle permet d'automatiser un certain nombre d'opérations comme la gestion des boucles du jeu et les évènements bas niveau.

- Le content pipeline permet d'importer facilement les contenus multimédia au sein du jeu. Il se charge notamment de reconvertir ces média pour les rendre utilisables par les plateformes visées.

La couche de plus haut niveau, "Jeux", contient le code et les ressources multimédia (contenu). Elle contient aussi les différentes librairies fournies par la communauté et des starters kits. Les starters kits sont des mini-jeux que les débutants peuvent manipuler afin de comprendre le fonctionnement de XNA.

3.2 WiimoteLib

Wiimote :



La grande originalité de la télécommande de la Wii, appelée Wiimote (contraction de Wii+Remote) réside dans l'utilisation d'un système de détection et d'analyse de mouvements et de positionnement.

Pour ce faire, la Wiimote embarque non pas un gyroscope (!) mais un accéléromètre 3D et un capteur infrarouge 1024*768 pixels. L'envoi direct de toutes les données vidéo vers la console demanderait une liaison haut débit ; aussi la Wiimote contient un calculateur de triangulation (tracking) analysant les données vidéo, pour fournir seulement à la console des données de positionnement et les accélérations dans les trois dimensions.

Les informations sont transmises par Bluetooth par un contrôleur Broadcom-2042 conforme au standard USB-HID. C'est ce qui va permettre la communication avec un contrôleur Bluetooth de PC; la télécommande apparaîtra alors dans la classe des périphériques HID (Human Interface Device) standards, au même titre qu'une souris ou une manette de jeux Bluetooth.

Pour effectuer le tracking, la manette repère la lumière IR envoyée par une rampe de leds IR (appelée sensor bar) placée sous l'écran ou sur l'écran. Seules une partie de toutes les fonctionnalités proposées sont utilisées.

WiimoteLib :

Cette bibliothèque permet d'utiliser une ou plusieurs Wiimote dans un environement .NET. Une fois la Wiimote connectée à l'ordinateur via Bluetooth, WiimoteLib peut connecter la télécommande dans l'application.

```
mWC = new WiimoteCollection();
ws = new WiimoteState();
        try
        {
                mWC.FindAllWiimotes();
        }
        int i = 0;
        foreach(Wiimote wm in mWC)
        {
      Wiimote[i] = wm;
      Wiimote[i].Connect(); // se connecte a la Wiimote
      WiimoteConnected[i] = true;
         i++;
        }
```

Pour récupérer les valeurs provenant de la Wiimote il faut utiliser l'objet WiimoteState. Avec cet objet il est possible de récupérer l'état courant des accéléromètres, des boutons et du tracking.

```
ws = Wiimote1.WiimoteState;
ws.AccelState.Values.ToString();

ws.ButtonState.A;
ws.ButtonState.B;
ws.ButtonState.One;
ws.ButtonState.Two;
ws.ButtonState.Up;
ws.ButtonState.Down;
ws.ButtonState.Left;
ws.ButtonState.Right;
```

Il faut savoir que mise à part les boutons qui renvoient un booléen pour donner leurs états. Les autres valeurs sont retournées sous forme de string. Il faut donc parser ces strings afin d'en ressortir les valeurs numériques.

```
private void setAccelerometers(string data)
{
    string[] accl = data.Split(',');
    string x = accl[0];
```

```
        x = x.Substring(3);
        string y = accl[1];
        y = y.Substring(3);
        string z = accl[2];
        z = z.Substring(3);
        z = z.Remove(z.Length - 1);
        xAxis = Double.Parse(x);
        yAxis = Double.Parse(y);
        zAxis = Double.Parse(z);

}
```

3.3 Speech SDK

Speech SDK est une API développée par Microsoft qui permet de faire de la synthèse vocale (Text to Speech) et de la reconnaissance vocale (Speech Recognition) dans les applications Windows. Son module de reconnaissance vocale utilise des algorithmes puissants tels que Hidden Markov Model (HMM) et Dynamic time warping (DTW). La version actuelle du framework est 5.3, le projet utilise la version 5.1.

Il faut noter que le framework Speech SDK est supporté par la plupart des langages de .NET. Par contre il est plus utilisé en C# et C++. Pour cela, un IDE supportant ces langages est nécessaire (Ex : Visual C# (express) ou Visual Studio).

Les systèmes d'exploitation tels que Windows Vista et Windows 7 intègrent des fonctionnalités de reconnaissance vocale.

Plusieurs applications utilisent déjà la reconnaissance vocale telle que dans la téléphonie, la domotique, le traitement de texte et d'autres applications d'authentification.

C'est la classe SpeechRecognizer qui est chargée de faire la reconnaissance vocale :

```
// Une isnstance de SpeechRecognizer
sr = new SpeechRecognizer();
// Une grammaraire définissant les mots à reconnaitre
Choices words = new Choices();

words.Add("soin");
words.Add("soins");

GrammarBuilder gb = new GrammarBuilder();
gb.Append(words);
```

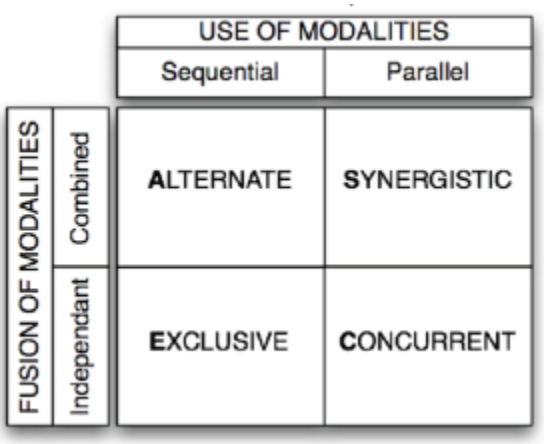
```
// Chargement de la grammaire dans le SpeechRecognizer
Grammar g = new Grammar(gb);
sr.LoadGrammar(g);

// Capture d'un événement pour SpeechRecognized
sr.SpeechRecognized += new
EventHandler<SpeechRecognizedEventArgs>(sr_SpeechRecognized);
```

4 Modalités

Pour normaliser les interactions multimodales Homme-Machine et faire des associations entre les entrées et les sorties multimodales, deux modèles conceptuels sont importants. Ils permettent de conceptualiser les relations entre les entrées et les sorties des modalités :

- Côté machine : CASE.
- Côté homme : CARE.

4.1 Modèle CASE



CASE est défini de la façon suivante :

- Séquentiel : Une modalité à la fois.
- Parallèle : Plusieurs modalités utilisées de façon simultanées.
- Combiné : Fusion nécessaire.
- Indépendant : Pas de fusion absence de coréférence

CASE est donc formé de quatre groupes nommés : Concurrent – Alterné – Synergique – Exclusive.

L'application se place de la façon suivante dans le CASE :

- Concurrent : L'utilisateur a la possibilité de faire de tâches distinctes en parallèle tel que monter avec la Wiimote et ouvrir la soute avec le clavier.
- Synergique : Une tâche utilisant plusieurs modalités. Un exemple est le largage de vivres, l'utilisateur doit presser un bouton de la Wiimote et tout en restant pressé effectuer le largage à l'aide d'une commande vocale.

4.2 Modèle CARE

Quatre propriétés sont disponibles : Complémentarité – Assignement – Redondance – Equivalence.

L'application HelpCopter se positionne de la façon suivante dans le système CARE :

- Complémentarité : La Wiimote ainsi que la voix doivent être utilisées afin d'atteindre le but qui est de larguer une caisse.
- Assignement : Une modalité suffit pour atteindre un état. Absence de choix. Pour toutes les actions le choix entre deux modalités est possible dans l'application.
- Equivalence : L'utilisateur a la possibilité d'utiliser le clavier à la place de la Wiimote.

4.3 Fusion

Le moteur de fusion utilise les deux modalités, trois si l'on compte le clavier : les mouvements et les signaux envoyés par les boutons de la Wiimote sont transmis au moteur de fusion et en même temps les mots reconnus par le reconnaisseur vocal sont transmis au moteur de fusion. Le niveau de fusion pour les gestes est au niveau décisionnel, il en va de même pour le traitement de la voix. Ce choix a été pris car dans les deux cas car les décisions à prendre par les modalités sont claires. Par exemple pour la voix; soit le mot est reconnu soit pas. Même chose pour les événements de la Wiimote, si le bouton est pressé la décision ne peut pas être contestée. Dans l'hypothèse de l'utilisation d'un deuxième microphone une telle fusion ne serait pas adaptée, en effet il serait plus judicieux de l'effectuer au niveau *Features* ou *Data* afin de comparer les signaux reçus par les deux micros afin de confirmer mathématique ou par une autre méthode que tel ou tel mot a bien été reconnu.

4.4 Fission

La restitution est directement faite sur l'écran. On peut facilement percevoir le largage des médicaments et des vivres. Quand le bouton A de la Wiimote est appuyé, la soute de l'hélicoptère s'ouvre et c'est à ce moment où la reconnaissance vocale intervient pour choisir le type d'objet à larguer (vivres ou médicaments).

5 Architecture logicielle

La phase Initialize permet d'initialiser les paramètres du jeu, de créer les composants du jeu et de récupérer des informations sur la plateforme d'exécution.

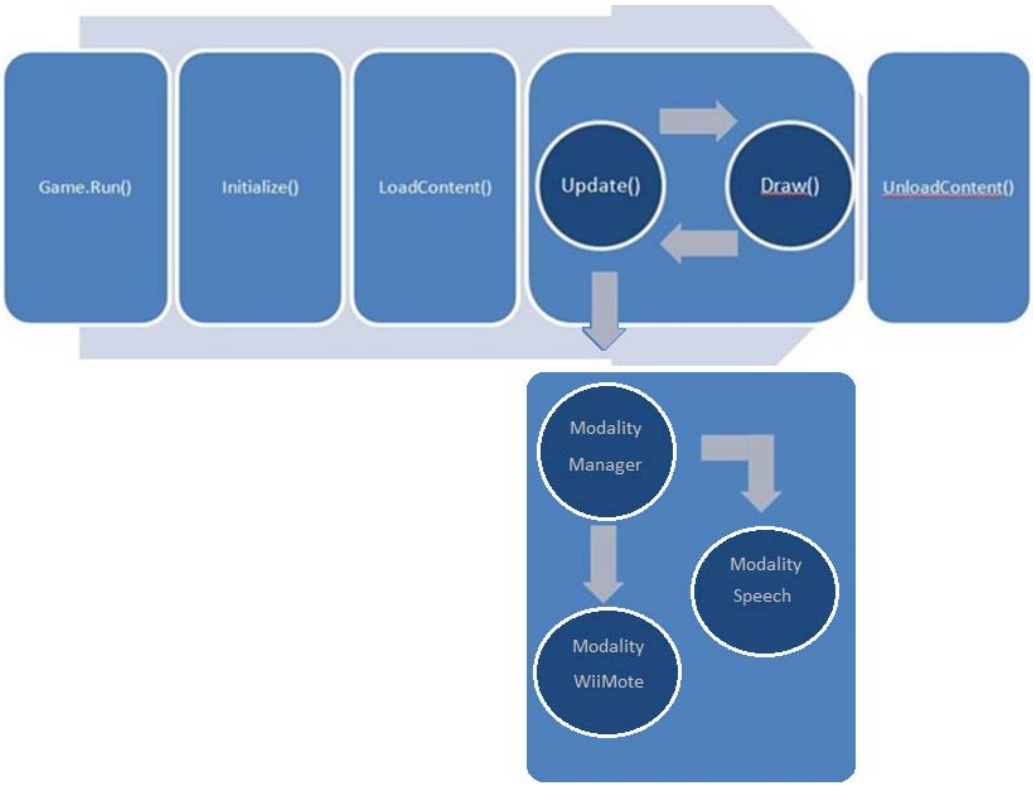
La phase LoadContent permet de charger les différentes ressources de votre jeu (textures, fichiers son etc.).

La phase Update permet de mettre à jour le modèle. Cela signifie que c'est ici que nous effectuons les opérations logiques du jeu. Cette phase permet par exemple de :

- Prendre en compte les différentes actions du joueur.
- Vérifier l'état des modalités dans le manager.
- Recalculer les positions des éléments du jeu.

La phase Draw permet d'afficher les éléments graphiques à l'écran.

La phase UnloadContent permet de décharger les ressources qui ont été préalablement chargé par LoadContent, cette phase est appelée lorsque l'utilisateur ferme l'application.



Afin de gérer au mieux les modalités dans l'application plusieurs processus sont créés. En utilisant une architecture avec plusieurs processus nous somme certains que les modalités sont gérées de façon totalement indépendantes et que certains traitements ne vont pas ralentir ou en pénaliser d'autres. Trois processus sont actifs :

1. Le processus père comprend la boucle principale de l'application. Les méthodes Update() et Draw() sont appelées indéfiniment tout au long du jeu. Update appel le ModalityManager qui a récupéré les événements des modalités. Dans un deuxième temps les calculs de jeux sont effectués.
2. Le processus WiimoteManager récupère les différentes actions de la Wiimote et les envois de façon asynchrone au ModalityManager.
3. Le processus SpeechManager fonctionne quant à lui de la même façon que le WiimoteManager.

Un système d'événement de C# est utilisé afin d'effectuer la communication inter-processus et ainsi permettre aux modalités d'envoyer leurs messages. Le même processus est utilisé par le processus père lors de la fermeture de l'application afin de transmettre le message de fin d'activité des modalités.

Le ModalityManager est créé afin qu'il puisse supporter plusieurs événements de même type venant de modalités différentes. Chaque modalité envoi également son identifiant afin que le ModalityManager connaisse la modalité ayant généré l'événement. Ce type d'implémentation permet d'être flexible et de traiter des cas complexes dont voici un exemple : Imaginons que pour larguer des vivres il nous faille l'événement "vivres" du SpeechManager mais également le bouton "vivres" de la Wiimote. Dans ce cas là le ModalityManager est capable de gérer ce type d'événement puisque chaque modalité est identifiée. Le ModalityManager attendra donc deux fois la même action mais de modalités différentes.

6 Lancement de l'application

Voici les différentes étapes à suivre et les différents problèmes qui peuvent apparaître.

Tout d'abord il faut lancer l' assistant Bluetooth avec de commencer la détection de la Wiimote. Veuillez ensuite la faire reconnaitre par votre système comme un composant Bluetooth normal. Une fois que la détection est réussie il se peut que l'application HelpCopter ne puisse directement reconnaitre la Wiimote. Veuillez donc au préalable cliquer sur l'exécutable « CD\WiimotLib_1.7\WiimoteTest.exe », il vous est alors possible de visualiser si la Wiimote est reconnues correctement.

Il est enfin possible d'exécuter « CD\Exécutable\HelpCopter.exe » et ainsi commencer le jeu.

Après avoir terminé une partie et avoir atteint l'écran « Game over », si l'utilisateur désire faire une nouvelle partie celui-ci doit redémarrer « HelpCopter.exe ».

Il est à noter qu'une Wiimote et que l'activation de Speech ne sont pas nécessaire au fonctionnement de l'application.

6.1 Fonctionnement

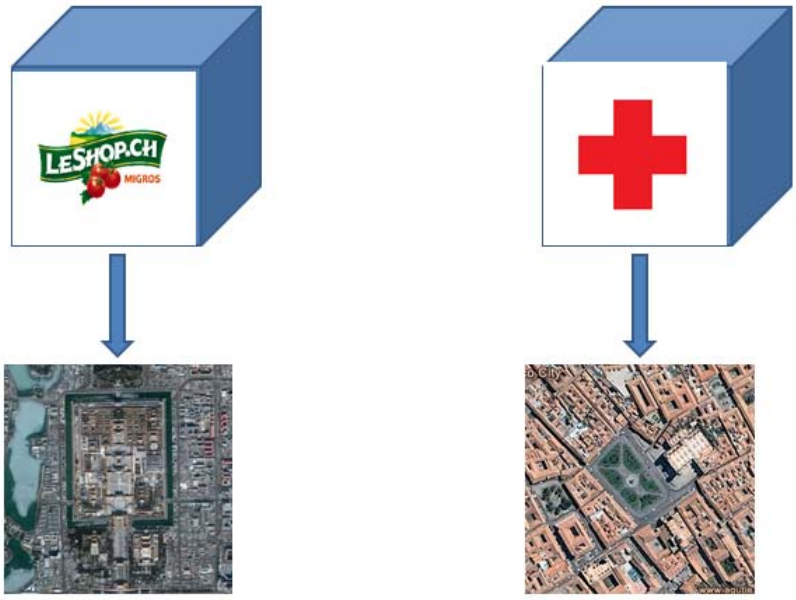
Voici quelques points importants concernant le comportement du jeu.

Lorsque l'utilisateur arrive à une extrémité de la carte celui-ci réapparait automatiquement de l'autre coté.

L'utilisateur ne doit pas non plus monter trop haut ou arriver en contacte avec le terrain, si cela lui arrive le nombre de mort augmentera ce qui peut provoquer une fin prématurée de la partie.

L'information en cas de trop haute altitude est « PULL DOWN » et en cas de contacte avec le terrain est « PULL UP ».

Il est nécessaire de larguer les bonnes caisses sur les bonnes villes, l'image ci-dessous en montre la correspondance.



Une seule caisse est nécessaire afin de sauver la ville. N'oubliez pas d'ouvrir au préalable la soute afin de pouvoir larguer une caisse. Il faut rester sur la touche correspondante(Q: clavier, A:Wiimote) afin de garder la soute ouverte. En lâchant la touche la soute se referme automatiquement.

Les commandes sont les suivantes afin d'effectuer les actions dans le jeu.

	accélérer	ralentir	monter	descendre	Tourne à gauche	Tourne à droite
Clavier	W	S	Flèche haut	Flèche bas	Flèche gauche	Flèche droite
Wiimote	Axe X contre l'avant	Axe X contre l'arrière	Croix directionnelle haut	Croix directionnelle bas	Croix directionnelle gauche	Croix directionnelle droite

	Glisser vers la gauche	Glisser vers la droite	Ouvrir soute	Larguer vivre	Larguer soins
Clavier	A	D	Q	Espace	alt
Wiimote	Axe Y sens contre-horaire	Axe Y sens horaire	Presser A	Dire ''vivre''	Dire ''soins''

6.2 Problème rencontré

Le processus de reconnaissance vocale étant géré par le système d'exploitation il se peut que celui-ci effectue des actions involontaires si celui-ci croit avoir détecté un mot tel que "bureau" ou "aide".

7 Evaluation

Pour l'évaluation nous avons testé le jeu sur plusieurs personnes. Nous avons essayé d'avoir différents type de personnes avec des profiles bien distincts. Nous avons catégorisé chaque testeur en 4 groupes; l'âge du participant, si c'est un homme ou une femme, si le joueur connait ou pas les jeux-vidéos et finalement s'il est habitué à jouer de temps en temps. Chaque participant à testé 3 fois le jeu. Une première fois pour apprendre et découvrir l'environnement les commandes du jeu avec la Wiimote et la reconnaissance vocale. Un second essai, mais avec les commandes au clavier afin de pouvoir comparer. Une fois que les mécanismes sont appris le testeur peut recommencer une nouvelle et dernière session afin d'améliorer son score ou affiner son jugement sur le produit. Lors de cette dernière session le participant peut choisir quelle modalité il veut utiliser. Cela permet de voir les préférences des joueurs. Afin de contre balancer l'effet d'apprentissage, les sessions 1 et 2 sont interverties à chaque nouveau testeur. Cela pour éviter que les testeurs choisissent toujours les modalités de la session 2(deuxième expérience généralement plus facile et donc préférée).

A la fin de la démo, nous avons fourni un questionnaire aux participants qu'ils devaient remplir, ou nous posions les questions directement à la personne et remplissait le questionnaire à leur place. Les personnes interrogées proviennent de notre entourage. Leurs avis pourraient être un peu biaisés due à cela. Nous leurs avons expliqué qu'un avis franc était préférable à un avis amical.

Nous avons effectué l'évaluation sur 8 personnes. La répartition est la suivante, nous avions 4 femmes et 4 hommes. Leurs âges étaient répartis de 9 à 90 ans, un jeune de 9 ans en 4ème année primaire habitué à jouer à la Wii, deux demoiselles entre 20-30 ans qui ne jouent jamais aux jeux-vidéos, deux jeunes de 26 et 29 ans qui jouent très régulièrement aux jeux-vidéos, une dame de plus de 50 ans qui n'y connait absolument rien et deux retraité de 87 et 90 ans qui n'y connaisse rien non plus. Nous n'avons malheureusement pas trouvé de retraité sachant jouer à la Wii…Il parait qu'ils existent…

Le questionnaire est un QCM où il est possible de quantifier la qualité du produit avec 6 degrés d'appréciation. Le participant peut donc répondre à une question et juger la qualité en disant si de son point de vue ça lui plaît beaucoup, si ça lui plaît, si ça lui plaît un peu, si ça ne lui plaît pas ou vraiment pas. Un espace est libre en fin du questionnaire permet de laisser un commentaire. Le score des 3 sessions est noté ainsi que la durée totale passée sur le jeu et la modalité choisie pour la 3ème session d'essai.

MULTIMODAL INTERFACES | UNIVERSITE DE FRIBOURG

HelpCopter:

Ce formulaire est destiné à évaluer le projet HelpCopter d'un point de vue d'utilisateurs provenant de différents environnements. Ce document reste anonyme afin de préserver la vie privée des testeurs. Il est dûment rempli après la

Date : 24.05.2011 Candidat :

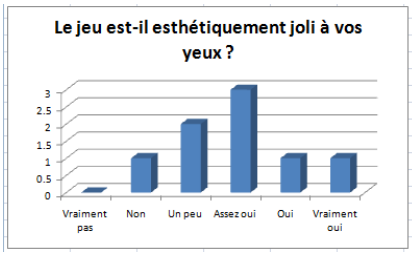
Age: Sexe : Occupation :

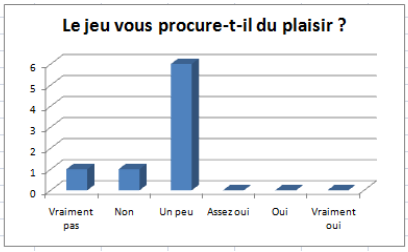
Attitude de joueur: Connaissances en jeux-vidéos :

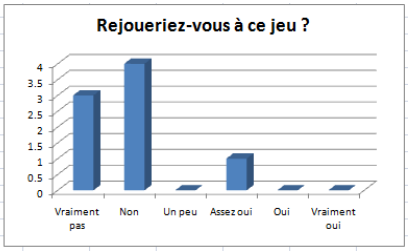
Questions	QUALITE					
	Vraiment pas	Non	Un peu	Assez oui	Oui	Vraiment oui
ASPECT GENERAL						
Le jeu est-il esthétiquement joli à vos yeux ?						
Le jeu vous procure-t-il du plaisir ?						
Rejoueriez-vous à ce jeu ?						
Seriez-vous prêt à acheter ce jeu ?						
Trouvez-vous un certain intérêt à ce jeu ?						
Est-ce que le fait que le jeu soit plutôt humanitaire vous plait ?						
Trouvez-vous le jeu original ?						
Y-a-t-il à votre avis assez de contenu ?						
COMMANDES						
Aimez-vous la jouabilité en général ?						
L'utilisation de la Wiimote est correctement utilisée et exploitée?						
Commentaires :						
L'utilisation de la reconnaissance vocale est-elle efficace ?						
Commentaires :						
Préférez-vous l'utilisation du clavier plutôt que de la Wiimote ?						
Préférez-vous l'utilisation d'un Game pad plutôt que de la Wiimote ?						
COMMENTAIRES						
	Scores: Session 1: Session 2: Session 3: Temps passé sur la démo : Choix de la modalité à la 3ème session :					

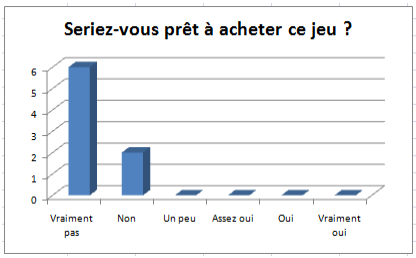
Les données sont ensuite insérées dans un fichier Excel a fin de créer des graphes permettant de voir les points forts et les points faibles du jeu.

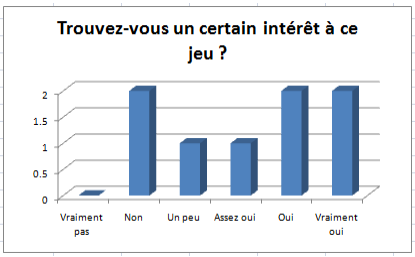
Voici les résultats que nous avons obtenus.

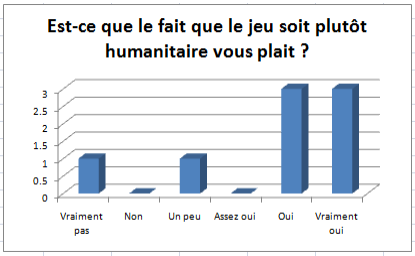


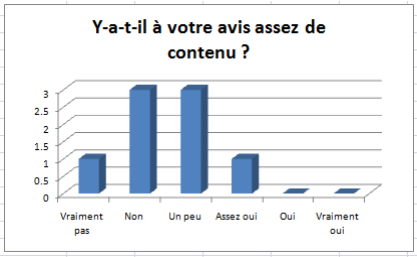


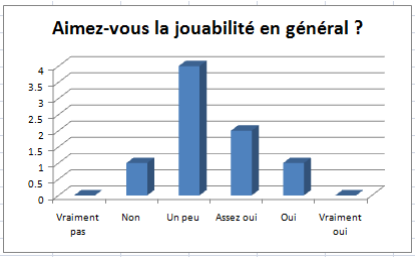


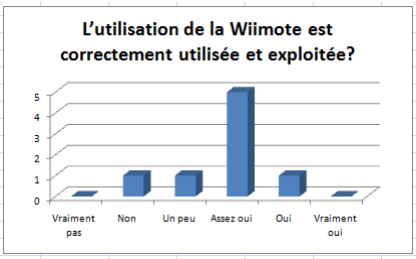


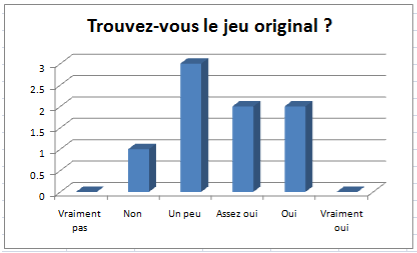


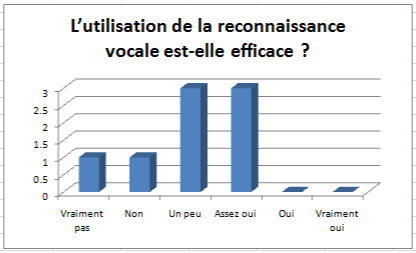


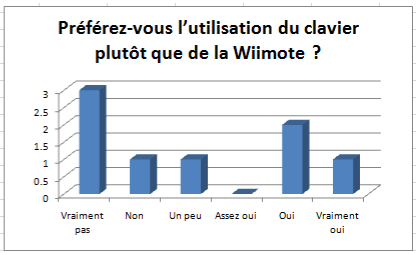












Les graphes nous montrent que les gens ayant testé le jeu ont eu moyennement du plaisir a priori du manque de variété dans le gameplay et à la difficulté trop élevée. Par contre ils ont l'air d'apprécier le fait que le jeu soit plutôt humanitaire que destructif. En ce qui concerne les modalités la Wiimote furent appréciées et plutôt bien utilisée, la reconnaissance vocale un peu moins dû à la difficulté d'utilisation.

8 Conclusion

Le projet HelpCopter nous a enfin permis de bien assimiler les différents concepts théoriques relatifs aux interfaces multimodales, notamment ceux de CASE/CARE, Fusion/Fission. Par contre, le développement d'une application 3D avec XNA nous a pris énormément de temps que nous avons malheureusement privilégié la partie XNA au lieu de la partie multimodale. Suite aux évaluations effectuées, il semble que les gens semblent plutôt vouloir rester sur Wiimote que sur clavier. Cela pourrait venir du fait que cela rende le jeu un peu plus ludique.

Dans une toute autre perspective, ce projet nous a également appris a savoir gérer notre emploi du temps, en effet nous sommes tous trois énormément chargés et souvent en déplacement en raison des différents lieux de cours.