Air Instruments

Spécifications des requis du système (SRS)

Version 1.1

Historique des révisions

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Auteur** |
| 2014-01-15 | 1.0 | Ébauche du SRS | Alexandre Vanier |
| 2014-01-22 | 1.1 | Amélioration du SRS et ajouts de requis | Alexandre Vanier |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table des matières

1. Introduction 5

1.1 But 5

1.2 Définitions, acronymes et abréviations 5

1.3 Vue d’ensemble du document 5

2. Description globale 5

2.1 Caractéristiques des usagers 5

2.2 Interfaces 5

2.2.1 Interfaces usagers 5

2.2.2 Interfaces matérielles 5

2.2.3 Interfaces logicielles 5

2.2.4 Interfaces de communication 6

2.3 Contraintes générales 6

2.4 Hypothèses et dépendances 6

3. Exigences fonctionnelles 6

3.1 Instruments obligatoires 6

3.2 Exigences relatives au piano 6

3.2.1 Vue à la première personne 6

3.2.2 Positionnement du capteur 6

3.2.3 Notes jouables 6

3.2.4 Grosseur relative des modèles 7

3.2.5 Utilisation des pédales 7

3.3 Exigences relatives aux instruments à percussion 7

3.3.1 Vue à la première personne 7

3.3.2 Jouabilité de la batterie 7

3.3.3 Positionnement du capteur 7

3.3.4 Pédale de la batterie 7

3.4 Instruments optionnels 7

3.5 Mode libre 7

3.6 Mode assisté 8

3.7 Mode néophyte 8

3.8 Détection automatique des instruments 8

3.9 Affichage à la première personne 8

3.10 Affichage d’une scène 8

3.11 Nombre de joueurs 8

3.12 Deux pianos en simultané 8

4. Exigences non-fonctionnelles 9

4.1 Utilisabilité 9

4.1.1 Temps de formation 9

4.1.2 Temps de délai du son 9

4.1.3 Temps de délai de l’image 9

4.2 Fiabilité 9

4.2.1 Temps moyen entre pannes 9

4.2.2 Temps moyen jusqu’à la réparation 9

4.3 Performance 10

4.3.1 Utilisation de ressource 10

4.4 Maintenabilité 10

4.4.1 Calibrage facile 10

4.4.2 Normes de codage 10

4.5 Contraintes de conception 10

4.5.1 Langage de programmation 10

4.5.2 Librairies utilisées 10

4.5.3 Processus 10

4.5.4 Achat de composantes 11

4.6 Exigences de la documentation usager en ligne et du système d’assistance 11

ANNEXE A: Glossaire 12

Spécifications des requis du système (SRS)

# Introduction

## But

Le SRS décrit le comportement externe d’une application. Il décrit aussi les exigences non fonctionnelles, les contraintes de conception, ainsi que les autres facteurs nécessaires à la description complète des exigences du logiciel à développer.

## Définitions, acronymes et abréviations

## Vue d’ensemble du document

Le document ci-dessous présente tout d’abord une description globale du système puis présente ensuite les exigences fonctionnelles organisées par thèmes (modes de jeu et instruments). Il y a finalement les exigences non-fonctionnelles qui sont organisées par des thèmes prédéfinis.

# Description globale

Air Instruments est un logiciel à but récréatif permettant à l’utilisateur de jouer de la musique sans avoir besoin d’instruments physiques. Le logiciel détecte les mouvements faits par l’utilisateur à l’aide de capteurs et effectue un traitement qui permet de déterminer les notes joutées et de produire un son réaliste.

## Caractéristiques des usagers

Air Instruments est un projet réalisé dans le cadre de la compétition Laval Virtual, il doit donc être adapté au grand public tout en impressionnant les experts en réalité virtuelle qui jugeront le produit. Autrement dit, le logiciel doit se démarquer du point de vue technologique tout en demeurant simple à utiliser pour un utilisateur lambda.

## Interfaces

### Interfaces usagers

Le jeu ne comportera pas d’interface usager traditionnelle dans laquelle il faut faire des sélections à l’aide d’une souris. Dans les rares cas où l’utilisateur doit faire un choix (choix du mode de jeu ou choix de l’instrument), un geste sera associé à chaque choix et l’utilisateur devra effectuer ceci. Un bonhomme affiché à l’écran guidera l’utilisateur en lui montrant les gestes associés à chaque choix. Les choix seront décrits à l’aide d’image afin de permettre aux utilisateurs de les comprendre rapidement.

### Interfaces matérielles

Le logiciel utilisera deux capteurs « Kinect for Xbox 360 » pour collecter de l’information sur les mouvements des joueurs. Un projecteur multimédia sera utilisé projeter les images générées sur une toile. Il est aussi possible que des écrans supplémentaires soient utilisés pour séparer l’interface du joueur de ce qui est vu par les spectateurs. Enfin, il faudra des haut-parleurs pour faire jouer le son des instruments.

### Interfaces logicielles

Le moteur Unity sera utilisé pour la partie qui sera affichée à l’écran (les modèles 3D et leur logique d’affichage). Le « Kinect SDK » sera utilisé pour gérer la capture d’images et pour avoir des fonctions de traitement de base. Nous utiliserons aussi les librairies OpenCV et OpenCL afin de faire du traitement d’images et des calculs en parallèle sur le GPU.

### Interfaces de communication

Les Kinect seront branchées via USB à un ordinateur responsable du traitement des images. La scène à afficher sera ensuite projetée à l’aide d’un projecteur. Si nous décidons d’avoir une vue distincte pour les joueurs et pour les spectateurs, la scène des spectateurs sera projetée à l’aide d’un projecteur multimédia et la scène «première personne» des joueurs sera affichée sur un écran d’ordinateur portable.

## Contraintes générales

L’espace disponible en compétition sera de 2m x 2m. Le logiciel doit fonctionner bien dans une pièce de ces dimensions. Le logiciel doit aussi fonctionner avec un niveau de bruit élevé et avec un éclairage variable.

## Hypothèses et dépendances

Nous émettons l’hypothèse que la Kinect calculera de façon précise et rapide les coordonnées de chaque articulation de 2 joueurs et fournira à une fréquence de 30 Hz des images que nous pourrons traiter.

# Exigences fonctionnelles

## Choix d’instrument

### Détection du geste correspondant à chaque instrument

À l’entrée dans un mode de jeu (les modes de jeu sont décrits à la section 3.6), chaque joueur pourra choisir l’instrument qu’il souhaite utiliser en mimant une personne en train de jouer cet instrument.

**Essentiel**

### Guidage pour le choix de l’instrument

Afin d’indiquer à l’utilisateur quels instruments sont disponibles et quel mime doit être fait pour que le logiciel reconnaisse le choix d’instrument, l’interface graphique guidera l’utilisateur. Ce guidage sera fait en affichant une image de chaque instrument disponible accompagnée d’un personnage effectuant le mime correspondant.

**Essentiel**

### Emplacement de l’instrument

Une fois le choix d’instrument reconnu, l’instrument virtuel sera positionné à l’endroit précis où le mime a été effectué.

**Souhaitable**

## Exigences relatives au piano

Le piano devra obligatoirement être disponible dans le jeu.

### Vue à la première personne

La vue à la première personne pour le piano sera une représentation comprenant les mains modélisées du joueur avec un piano modélisé en dessous. La caméra (Unity) devra être placée de façon à ce que le joueur puisse bien voir la hauteur de ses mains par rapport au piano afin de faciliter l’utilisation de celui-ci.

**Essentiel**

### Positionnement du capteur

Le capteur devra être placé au sol de façon à ce que celui-ci puisse détecter plus facilement la profondeur des mains.

**Essentiel**

### Notes jouables

Au moins 5 octaves de notes d’un piano classique devront être jouables, incluant les noires et les blanches.

**Essentiel**

### Grosseur relative des modèles

Le modèle du piano devra contenir des notes qui sont d’une taille un peu plus grande que la largeur des doigts afin de faciliter un fonctionnement précis du capteur et du traitement d’images.

**Essentiel**

### Utilisation des pédales

Il sera possible d’utiliser les pédales du piano en plaçant les pieds au sol à un endroit défini.

**Optionnel**

## Exigences relatives à la batterie

La batterie devra obligatoirement être disponible dans le jeu.

### Vue à la première personne

Une vue à la première personne sera affichée lorsque le capteur détectera que l’utilisateur effectue un mouvement semblable à un joueur de batterie (mouvement des bras). La vue présentera une vue des baguettes qui s’agitent sur un modèle de batterie.

**Essentiel**

### Jouabilité de la batterie

Tous les instruments à percussion dans une batterie seront jouables. Les mouvements pour jouer chaque instrument devront être conformes à la réalité.

**Essentiel**

### Positionnement du capteur

Le capteur pour la batterie sera positionné en face du joueur (à une distance d’environ 2 mètres et à la hauteur de ses épaules). Il détectera les mouvements des bras afin de savoir quels instruments sont joués à chaque instant.

**Essentiel**

### Pédale de la batterie

La pédale de la batterie sera jouable.

**Essentiel**

## Exigences relatives à la guitare

Il est souhaitable que le jeu permette de jouer de la guitare.

### Vue à la troisième personne

Une vue à la troisième personne sera affichée lorsque le capteur détectera que l’utilisateur effectue un mouvement semblable à un joueur de guitare. La vue présentera le joueur de face, avec une guitare dans les mains. Les mouvements de bras du joueur sur les cordes et sur le manche de la guitare seront reproduits fidèlement.

**Souhaitable**

### Jouabilité de la guitare

Des notes seront produites lorsque le joueur passera sa main sur les cordes de la guitare. Les notes produites seront choisies en fonction de 2 facteurs :

* La position de l’autre main sur le manche de la guitare déterminera si le son produit doit être plus aigu ou plus grave.
* Les notes seront choisies dans une tonalité déterminée à l’entrée dans le mode. Dans le cas du mode assisté, la tonalité dépendra de la musique jouée.

**Souhaitable**

### Positionnement du capteur

Le capteur pour la guitare sera le même que pour la batterie et sera donc positionné au même endroit.

**Souhaitable**

## Instruments souhaitables

Il est souhaitable d’ajouter divers instruments à percussion, tels que des tam-tams, un triangle, un gong et des maracas.

**Souhaitable**

## Instruments optionnels

Divers instruments à vents pourront être jouables.

**Optionnel**

## Modes de jeu

### Mode libre

Un mode libre sera disponible où l’utilisateur pourra jouer de l’instrument choisi comme il le désire. Dans ce mode, seul le son des instruments sera joué par les haut-parleurs.

**Essentiel**

### Mode assisté

Un mode assisté sera disponible. Dans ce mode, une musique présélectionnée sera jouée. Lorsque l’utilisateur choisira son instrument, cet instrument deviendra silencieux dans la musique jouée automatiquement. L’utilisateur pourra alors jouer l’instrument choisi, idéalement en suivant la mélodie qui était jouée par l’instrument devenu silencieux. Lorsque l’utilisateur arrête de jouer, l’instrument sélectionné recommence à être joué automatiquement.

**Essentiel**

### Mode néophyte

Un mode néophyte sera disponible. Dans ce mode, l’utilisateur sera assisté par le logiciel dans la pratique de l’instrument choisi. Pour un piano, par exemple, les notes à jouer pourraient être mis en surbrillance à l’avance ou alors on pourrait fournir l’enchainement de notes à suivre dans une sorte de « tableau » à la Guitar Hero/Rock Band.

**Optionnel**

## Affichage à la première personne

Un affichage à la première personne sera affiché pour tous les instruments afin d’aider l’utilisateur à percevoir sa position par rapport à l’instrument. Cette vue dépendra de l’instrument détecté au départ et pourra changer si l’utilisateur change d’instrument en cours de route.

**Essentielle**

## Affichage d’une scène

Une scène sera affichée où chaque joueur sera montré avec l’instrument joué près de lui (comme s’il jouait d’un vrai instrument). Un décor pourra être affiché en arrière.

**Optionnel**

## Nombre de joueurs

Le nombre maximal de joueurs simultanés sera de 2. Chaque joueur pourra choisir l’instrument de son choix, mais il pourra y avoir au maximum un piano à la fois.

**Essentiel**

### Deux pianos en simultané

Il sera possible pour les deux joueurs de jouer du piano de façon simultanée.

**Optionnel**

# Exigences non-fonctionnelles

## Utilisabilité

### Temps de formation

Un utilisateur normal doit pouvoir utiliser le logiciel facilement, avec une période d’apprentissage d’au plus 1 minute. Cela est nécessaire puisque le logiciel sera présenté dans un salon dans lequel les utilisateurs ont peu de temps à consacrer à chaque présentation.

### Latence du son

Le son de chaque note pour chaque instrument devra être joué avec un délai maximal de 25 millisecondes après que l’action ait été effectuée par l’utilisateur.

### Latence de l’image

Les actions de l’utilisateur seront répercutées dans la vue affichée avec un délai maximal de 50 millisecondes après que l’exécution de l’action.

## Fiabilité

### Temps moyen entre pannes

Le temps moyen entre panne devra être grand afin d’assurer le bon déroulement des démonstrations, avec 1 panne par 9 heures.

### Temps moyen jusqu’à la réparation

Le système devra être facilement réparable et calibrable. Ainsi, le temps pour redémarrer le logiciel sera d’au maximum 5 minutes. Le temps pour installer tous les capteurs et les calibrer sera d’au maximum 20 minutes.

## Performance

### Utilisation de ressource

Le système devra pouvoir s’exécuter sur un ordinateur portable d’un membre de l’équipe, doté de la configuration suivante :

* Processeur : Intel Core i7 – 4 coeurs @ 2.70 GHz
* Mémoire : 8 Go
* Carte graphique : NVIDIA Quadro K1000M et Intel HD Graphics 4000
* Espace disque : 500 Go

Nous n’avons pas de contraintes d’économie de ressources matérielles ou de courant électrique. Nous pouvons donc nous permettre d’utiliser 100% des ressources décrites ci-dessous dans le but d’avoir un logiciel permettant une immersion dans le monde virtuelle qui semble aussi réelle que possible.

## Maintenabilité

### Calibrage facile

Le système doit être facile à calibrer afin que le système puisse être réutilisé par le département de génie logiciel pour maximiser le rayonnement de l’École Polytechnique.

### Normes de codage

Les conventions de codage utilisées seront le Google style guide pour C++ (<http://google-styleguide.googlecode.com/svn/trunk/cppguide.xml>) et les conventions de Microsoft pour le C#.

## Contraintes de conception

### Langage de programmation

Les langages utilisés seront le C# pour tous les scripts Unity et le C++ pour les algorithmes de traitement d’image et la communication avec les capteurs.

### Librairies utilisées

Le moteur 3D Unity sera utilisé pour contrôler le jeu. Les libraires OpenCV et OpenCL seront aussi utilisées pour le traitement d’images et les calculs en parallèles sur GPU. Le Kinect SDK de Microsoft sera utilisé pour faire des traitements de base sur les données reçues de la Kinect.

### Processus

Le processus utilisé sera une variation des processus agiles. Ce processus est défini plus en détails dans le schéma déjà remis.

### Achat de composantes

Il faudra se procurer deux capteurs Kinect ainsi qu’un projecteur auprès de l’école. Il faudra aussi se procurer 2 trépieds pour supporter les Kinect ainsi que quelques matériaux de constructions pour construire une protection pour les capteurs.

## Exigences de la documentation usager en ligne et du système d’assistance

Il n’y aura pas de documentation usager excepté une aide en jeu indiquant les mouvements nécessaires pour simuler chaque instrument.

# ANNEXE A: Glossaire

[Si vous n’utilisez pas cette annexe, veuillez la supprimer complètement.]

|  |  |
| --- | --- |
| **Terme** | **Description** |
| SDK | *Software Development Kit* |
| FPS | Vue à la première personne (*First person shooter*) |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |