Air Instruments

Document d'architecture logicielle

Version 1.0

Historique des révisions

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Auteur** |
| 2014-02-12 | 1.0 | Ébauche, mise en ligne. | Félix G. Harvey |
| 2014-02-12 | 2.0 | Ajout des diagrammes. | François Doray |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table des matières

1. Introduction 4

2. Objectifs et contraintes architecturaux 4

3. Vue des cas d’utilisation 4

4. Vue logique 4

4.1 Diagramme de paquetages 4

4.2 Description des paquetages de «Air Instruments» 5

4.3 Description des paquetages de «SensorLib» 9

5. Vue des processus 11

6. Vue de déploiement 11

6.1 Diagramme de déploiement 11

6.2 Plan physique de l’installation 12

7. Taille et performance 12

Document d'architecture logicielle

# Introduction

Ce document offre un aperçu de l’architecture du système qui sera conçu. On décrit d’abord les objectifs et contraintes architecturaux. Puis, à l’aide de diagrammes, on présente une vue des cas d’utilisation, une vue logique, une vue de processus et une vue de déploiement du système.

# Objectifs et contraintes architecturaux

Le système développé doit être d’une très grande fiabilité puisque toute défaillance pourrait démolir nos chances de remporter la compétition. Le système doit aussi mettre de l’avant des technologies nouvelles et faire preuve d’une grande originalité.

Afin d’obtenir des résultats satisfaisants dans le temps alloué, nous maximiserons l’utilisation de bibliothèques logicielles existantes.

# Vue des cas d’utilisation



# Vue logique

## Diagramme de paquetages

L’application est divisée en 2 binaires. Le binaire «Air Instruments» est un exécutable créé avec le moteur de jeu Unity. Il gère la logique de jeu, l’affichage et les sons des instruments. La librairie dynamique «SensorLib» est codée en C++. Elle accède à l’interface C++ des SDKs associés aux divers capteurs utilisés et permet au projet Unity d’accéder à leurs données sous une forme simplifiée. La librairie peut appliquer divers filtres à ces données afin de les rendre plus adaptées aux besoins du projet.



## Description des paquetages de «Air Instruments»

|  |  |
| --- | --- |
| **GameLogic** | |
| Description: | Conserve l’état de l’application (mode de jeu et instrument) et gère les transitions entre états. Gère le choix des musiques d’accompagnement dans le mode assisté. |
| Classes incluses: | GameManager  *Conserve l’état de l’application.*  InstrumentChooser  *Surveille les gestes de changement d’instrument et instancie les bons instruments.*  ModeChooser  *Surveille les gestes de changement de mode et exécute le changement de mode.*  AssistedController  *Gère le mode assisté.*  FreeController  *Gère le mode libre.*  HackerKit  *Surveille les commandes du présentateur.* |
| Relations: | Utilise Rendering, GestureRecognizer, Drum, Guitare, Piano. |
| Sous-paquetages: | - |



|  |  |
| --- | --- |
| **Rendering** | |
| Description: | Insère dans la scène les objets propres à chaque mode et gère l’animation du décor. |
| Classes incluses: | InstrumentChooserRenderer  ModeChooserRenderer  AsssitedModeRenderer  FreeModeRenderer |
| Relations: | Utilisé par GameLogic. |
| Sous-paquetages: | - |

|  |  |
| --- | --- |
| **GestureRecognizer** | |
| Description: | Contient des classes permettant de reconnaître des gestes faits avec le squelette capté par une Kinect. Ces gestes sont surtout utilisés pour effectuer le choix d’un instrument. |
| Classes incluses: | GestureRecognizer (Interface)  HandsUpRecognizer  DrumRecognizer  GuitareRecognizer  PianoRecognizer |
| Relations: | Utilise Skeleton. Est utilisé par GameLogic. |
| Sous-paquetages: | - |



|  |  |
| --- | --- |
| **Drum** | |
| Description: | Gère le fonctionnement du drum. |
| Classes incluses: | TipController  *Contrôle les mouvements d’une baguette de drum.*  DrumComponent  *Script associé à un objet de la scène. Joue un son lorsqu’une collision avec une baguette est détectée.* |
| Relations: | Utilise Skeleton. Est utilisé par GameLogic. |
| Sous-paquetages: | - |



|  |  |
| --- | --- |
| **Guitare** | |
| Description: | Gère le fonctionnement d’une guitare. |
| Classes incluses: | GuitarController  *Script associé à un modèle de guitare. Gère son positionnement sur un squelette.*  StringComponent  *Joue des sons de guitare en fonction des mouvements d’un squelette.* |
| Relations: | Utilise Skeleton. Est utilisé par GameLogic. |
| Sous-paquetages: | - |



|  |  |
| --- | --- |
| **Piano** | |
| Description: | Gère le fonctionnement du Drum. |
| Classes incluses: | HandController  *Script associé à un modèle de main. Gère son positionnement.*  NoteComponent  *Script associé à un modèle 3D de note de piano. Joue un son lorsqu’une collision est détectée.* |
| Relations: | Utilise Skeleton. Est utilisé par GameLogic. |
| Sous-paquetages: | - |



|  |  |
| --- | --- |
| **Skeleton** | |
| Description: | Extrait les informations des squelettes détectés par la Kinect et les utilise pour contrôler un modèle 3D de joueur. |
| Classes incluses: | Skeleton  *Point d’accès aux données brutes de la Kinect pour un squelette particulier.*  JointMover  *Script associé à un modèle 3D d’un membre du joueur. Gère son positionnement à partir des données obtenues d’un objet Skeleton.* |
| Relations: | Utilise SensorLibInterop. Est utilisé par GestureRecognizer, Drum, Guitare, Piano. |
| Sous-paquetages: | - |



|  |  |
| --- | --- |
| **SensorLibInterop** | |
| Description: | Point d’accès à la librairie SensorLib. |
| Classes incluses: | SensorLibInterop |
| Relations: | Est utilisé par Skeleton. |
| Sous-paquetages: | - |



## Description des paquetages de «SensorLib»

|  |  |
| --- | --- |
| **SensorLib** | |
| Description: | SensorLib |
| Classes incluses: | Expose une interface à l’extérieur de la librairie. |
| Relations: | Utilise HandTracker et KinectWrapper. |
| Sous-paquetages: | - |



|  |  |
| --- | --- |
| **HandTracker** | |
| Description: | Applique des filtres aux données du «Intel Hand Tracker» et effectue la calibration des nouvelles mains apparaissant dans le champ de vision. |
| Classes incluses: | HandTracker  HandData  HandCalibrator  KalmanFilter |
| Relations: | Est utilisé par SensorLib. Utilise IntelWrapper. |
| Sous-paquetages: | - |



|  |  |
| --- | --- |
| **IntelWrapper** | |
| Description: | Interface au Intel Perceptual SDK. |
| Classes incluses: | HandTrackerInterface  *Cette classe sera fortement inspirée d’un* sample *fourni par Intel.* |
| Relations: | Est utilisé par HandTracker. |
| Sous-paquetages: | - |



|  |  |
| --- | --- |
| **KinectWrapper** | |
| Description: | Interface au Kinect SDK, offrant des fonctionnalités supplémentaires comme la possibilité de rejouer des séquences enregistrées. |
| Classes incluses: | KinectWrapper  KinectSensor  KinectData  KinectReplay |
| Relations: | Est utilisé par SensorLib et HandTracker. |
| Sous-paquetages: | - |



# Vue des processus

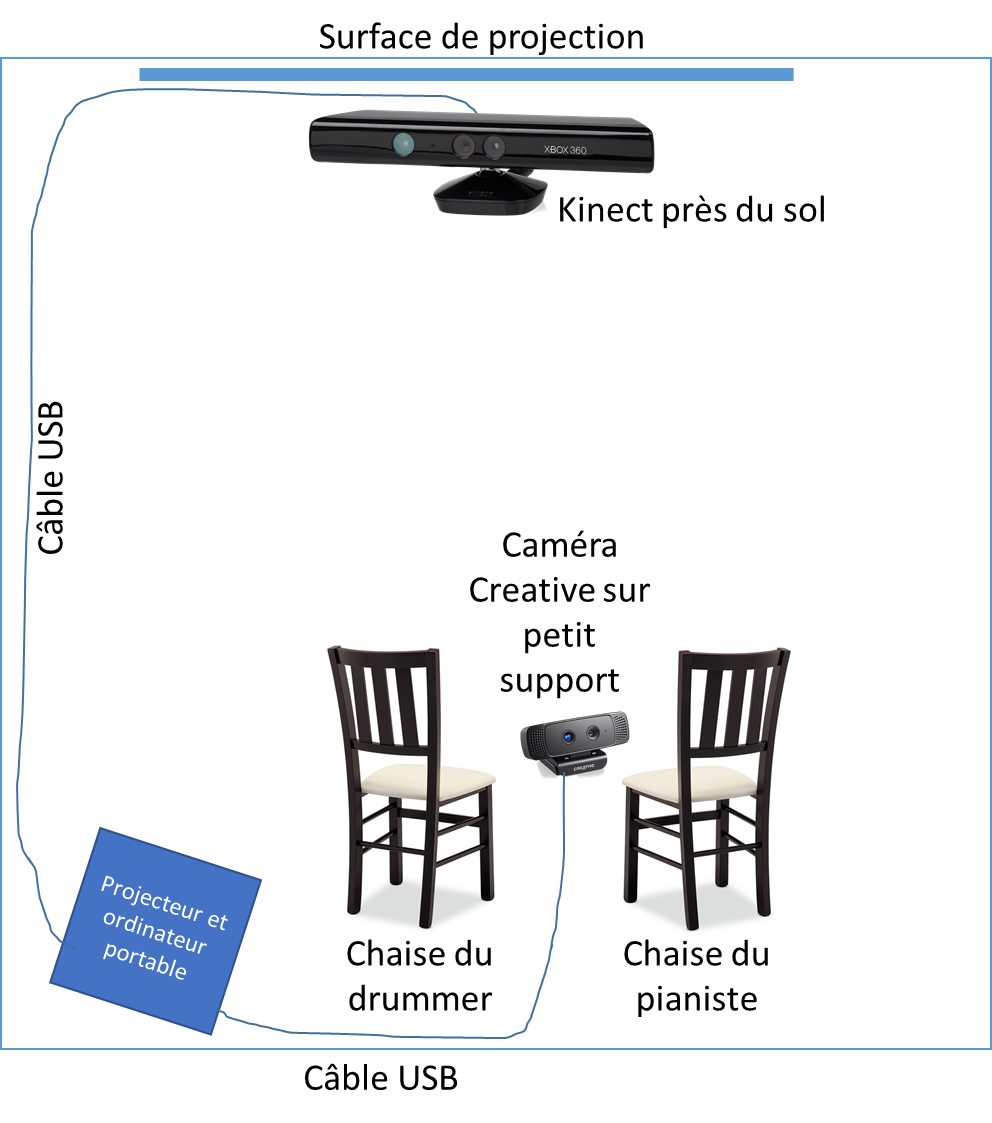
[Cette section décrit le système en termes d’interactions entre les différents processus (« lourds » ou légers). Typiquement, on utilise un diagramme des composantes pour ce faire. On peut aussi utiliser tout diagramme jugé pertinent, comme le diagramme de séquence.]

# Vue de déploiement

## Diagramme de déploiement



## Plan physique de l’installation



# Taille et performance

L’application doit pouvoir afficher 30 images par seconde et avoir une latence d’au plus 80 secondes (temps entre l’exécution d’un mouvement et la réponse de l’application) sur 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

Il n’y a pas de taille limite pour l’exécutable puisqu’elle sera exécutée sur un ordinateur disposant de beaucoup d’espace disque.