

DEVELOPPEMENT D’APPLICATIONS CREATIVES

Kinect avec Python

RAUDRANT – CLERET – JACQUIN – PERCHE | DAC | 10/01/2018

# Introduction

### Choix de la solution de programmation

Pour la réalisation, nous avons tout d'abord penser s'orienter sur l'utilisation du logiciel Processing avec une programmation en Java. Mais après avoir rencontré plusieurs problèmes notamment sur la conversion des positionnements des points dans les espaces des caméras, nous avons décidé de rechercher une alternative pour finir par concevoir le programme à l'aide de Python, en utilisant les librairies Pygames (pour l'affichage et l'interaction client) et PyKinect2 (pour l'utilisation de la Kinect V2).

### Composition de l’équipe

Pour mener à bien le projet qui nous a été demandé, nous avons décidé de répartir l'équipe comme suit :

* Sophie Raudrant - Théorie mathématique et librairies Utilitaires
* Laurine Cleret – Conception des Managers pour interactions Kinect
* Arthur Jacquin – Conception des Interfaces et éléments graphiques de débug
* Killian Perché - Conception des scripts d'interactions avec la Kinect

### Composants nécessaires

Pour exécuter le programme, il est nécessaire de disposer de de Python 3 (version 3.6+) en version 64 bits. Il faut aussi avoir installer la Kinect SDK 2 et avoir les configurations nécessaires pour pouvoir exécuter des programmes utilisant la Kinect V2.

Vous pouvez télécharger Python à l’adresse : <https://www.python.org/ftp/python/3.6.4/python-3.6.4-amd64.exe>

Configuration de Python:

* Pour que Python soit en mesure de pouvoir exécuter le programme, seul la présence de Python 3.6+ 64bits est nécessaire. Une fois installé, exécuter le script d'installation compris dans la solution. Celui-ci liera l'environnement virtuel « *venv*» à votre installation.
* Si l'utilitaire ne parviens pas à trouver votre installation de Python (généralement installer dans « *%LOCALAPPDATA%/Programs/Python/Python36* ») il faut éditer manuellement le fichier *venv/pyenv.cfg* et remplacer la valeur de home par le chemin d'accès de votre installation Python.

Attention : si votre installation se trouver dans un répertoire de type *Python3\*\_32*, vous n'avez pas installer la version 64 bits, et le package PyKinect2 ne fonctionnera pas.

### Exécution du programme

Pour exécuter le programme, il suffit d’exécuter le script « *DAC\_Executable.cmd* ».

# Prise en main du programme

### Prise en main

Espace : Changer le mode d’interaction  
Echappe : Fermer le programme

### Conception

Le premier script à s’exécuter est main.py. Celui-ci créer une instance de MainWindow contenant tous les éléments graphiques et logiques du programme. Ce dernier contient une instance de chaque méthode d’interaction (chute de neige ou jeu de balle) ainsi qu’une instance du Manager de KinectManager qui se charge de communiquer avec la Kinect.

#### Chute de neige

Pour générer la chute de neige, nous avons décidé d’afficher à l’écran des sphères blanches qui contiennent un vecteur de transformation vertical dirigées vers le bas. Ces sphères représentant les flocons sont générées avec une position et un vecteur de transformation aléatoire tronqué dans une limite de 803 éléments simultanément et sont automatiquement supprimés quand ils sortent de la zone de l’écran. Pour effectuer la détection nous avions tout d’abord implémenter un algorithme de détection de collision entre une sphère et une ligne. Seulement le nombre de flocons de neige à afficher cumuler au nombres de calcules vectoriels effectués avaient réduit drastiquement les performances du programme (une moyenne de 75% du temps d’exécution du code était dédié au calcule de collision – voire graphiques d’optimisations dans le dossier Resources). Nous avons donc décider de limiter les calculs aux flocons se trouvant à une distance arbitraire de 150 px d’un « os » Kinect. Ensuite, pour que la chute de neige s’arrête au niveau supérieure des membres supérieurs, nous avons implémenter une classe BodyParts, qui se charge de calculer le volume approximatif lié à l’os Kinect. Pour détecter la collision avec la partie tête, nous avons simplement implémenter une méthode de calcul de collision entre deux sphères.

#### Jeu de balle

Pour interagir avec la balle, nous n’avons malheureusement pas pu implémenter l’objet 3D comme demandé car PyGame ne prends pas en compte la 3D. Nous avons donc décider d’afficher une balle 2D sous forme d’une texture qui par défaut se trouve au centre de l’écran, et qui peut être étiré lorsque l’on ferme les poings au niveau de sa périphérie. Si nous approchons trop nos mains du centre de ce ballon, nous l’écrasons et il explosera. Aucune difficulté particulière n’a été rencontré lors du développement de cette interaction.

#### Debug

Pour activer les outils de développement tel que l’affichage du Kinect Skeleton, il suffit d’ajouter le string « Debug » dans la liste Globals disponible dans le script Utilities/globals.py