Projet Informatique Casque VR

GAYET Constant GAIGNE Paul LEGAUD Pierre

0.Introduction

Tout d'abord, le casque de réalité virtuelle est une technologie d'avenir. Celle-ci connaîtra sûrement un essor dans les années à venir dans plusieurs domaines (médecine, jeux vidéos …). C'est pourquoi nous avons voulu travailler sur ce projet, en ajoutant à cela un aspect concret, physique de ce que nous manipulions, au travers de composants comme l'Arduino ou l'accéléromètre.

Nous avions pour objectif de programmer un casque VR à partir de données brutes sortant de l'accéléromètre, c'est-à-dire pouvoir se déplacer dans une image virtuelle à partir d'un mouvement réel.

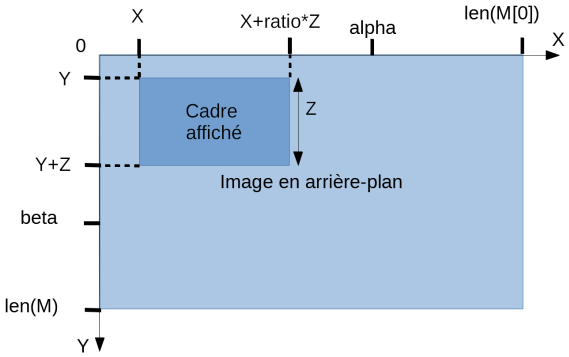
I. Affichage de l'image

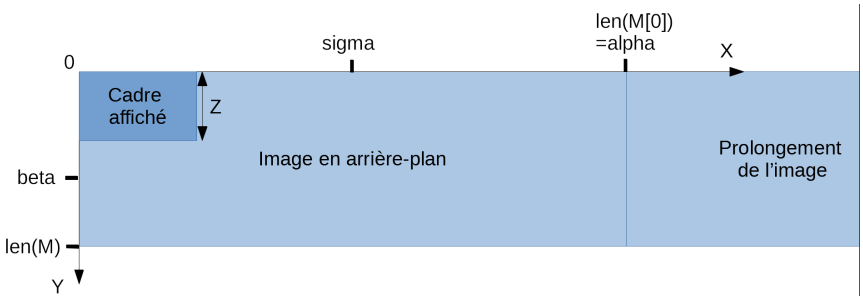
Deux possibilités s'offraient à nous : gérer l'image en tant que matrice ou l'afficher comme graphique grâce à matplotlib et se déplacer dans ce graphe. Nous avons retenu la seconde solution. La lecture de l'image s'effectue donc avec la fonction imread du module imageio (transformation en matrice) et l'affichage se fait avec la fonction imshow du même module (affichage de la matrice dans une fenêtre graphique).

II. Déplacement dans l'image

Pour gérer le déplacement dans l'image, il nous suffisait donc de gérer la redéfinition et la réactualisation des axes et de leurs limites. Dans un premier temps nous l'avons effectué avec des curseurs pour se déplacer et pour zoomer, en tenant compte des limites d'affichage, explicitées ci-dessous :

Limites d'affichage pour 2D et 360 :





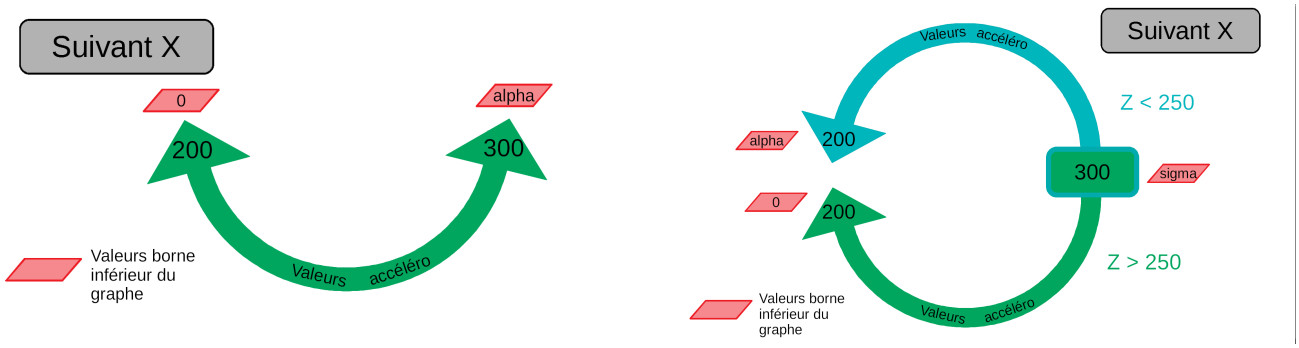
Dans la continuité du projet, il fallait désormais relier python avec l'acquisition Arduino.

III. Lien avec l'Arduino

Le programme Arduino nous était fourni. Il fallait cependant exploiter les données en sortie de celui-ci, en fonction de la position de l'accéléromètre.

Nous avons remarqué que les valeurs (suivant les 3 axes) variaient entre 200 et 300 en valeur brute pour notre matériel. Grâce à la fonction map (annexe 1), nous pouvons récupérer une valeur comprise dans les bornes de l'image, proportionnellement à celle de l'Arduino(cf ci-dessous suivant X et fonctionnement analogue selon Y).

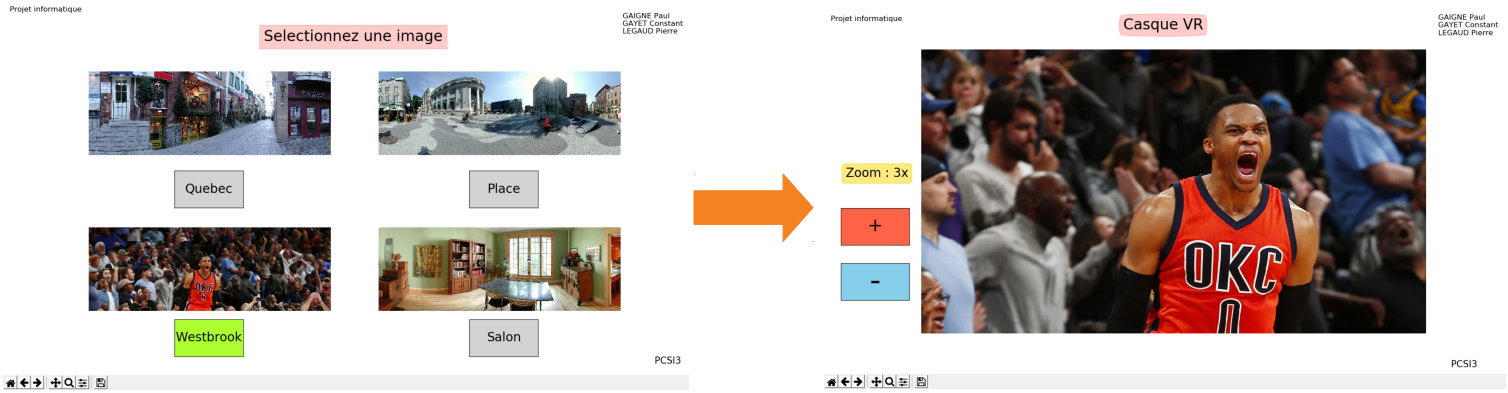
Rapport entre les valeurs de l'accéléromètre et les bornes du graphique en 2D et 360:



La fonction animate nous permet désormais d'afficher la partie de l'image correspondante en prenant en compte le zoom. Celle-ci est exécutée avec un intervalle de temps régulier ce qui permet de rafraîchir le graphe. Pour une meilleure fluidité dans le déplacement, nous avons finalement créé les fonctions de lissage suivant les deux axes de translation (annexe 2).

IV. Développement de l'interface

Une interface graphique était nécessaire dans le cadre de notre projet. En effet, il fallait un bel aspect graphique dans notre contexte. Nous avons donc développé une première interface dans laquelle il est possible de choisir son image (4 choix possibles en 2D ou 360°) et une seconde permettant de gérer le zoom de l'image, une fois celle-ci affichée.



V. Ouverture et conclusion

La principale limite du système reste la fluidité ou plutôt l'instabilité. Cela pourrait être corrigé en changeant le capteur pour un plus stable. Par ailleurs, nous aurions pu imaginer de se déplacer dans une vidéo et non plus dans une image fixe, mais la fluidité n'aurait pas été au rendez-vous ou encore une véritable photo 360°.

Annexe 1

```
def map(X,Y,V,Z):           #renvoie une valeur entre 0 et Z, proportionellement a V qui est entre X et Y
    P=int(((V-X)/(Y-X)*Z))
    return P
```

Annexe 2

```
def lissagex():
    X=[]
    for i in range (0,10):              # Création d'une liste avec 10 valeurs de l'accéléromètre
        X.append(donneex.read())
    return mean(X)                      # Renvoie la valeur moyenne de la liste

def lissagey():
    Y=[]
    for i in range (0,10):              # Création d'une liste avec 10 valeurs de l'accéléromètre
        Y.append(donneey.read())
    return mean(Y)                      # Renvoie la valeur moyenne de la liste
```