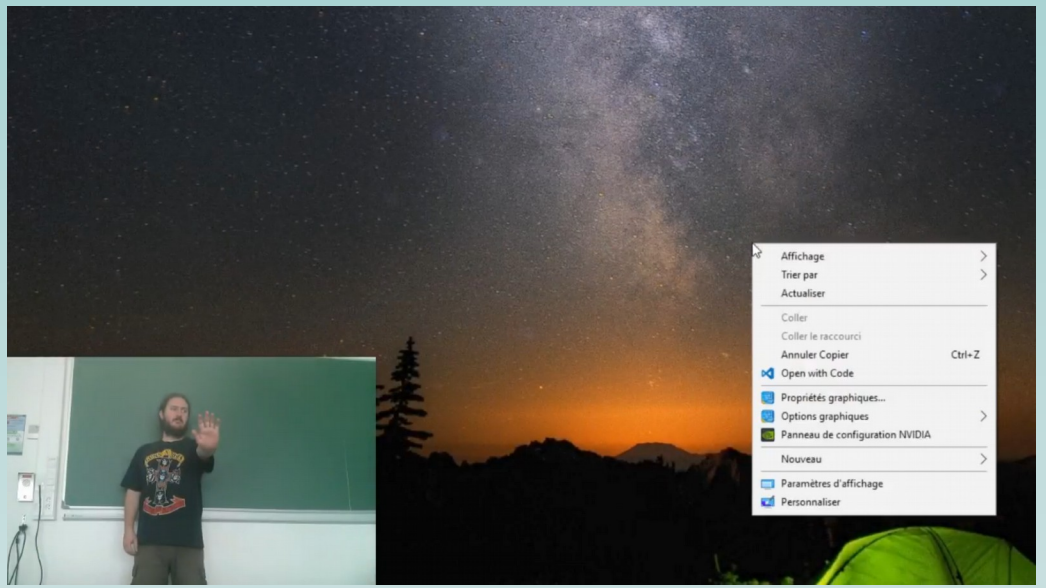
Utilisation de la main pour remplacer la souris avec Kinect

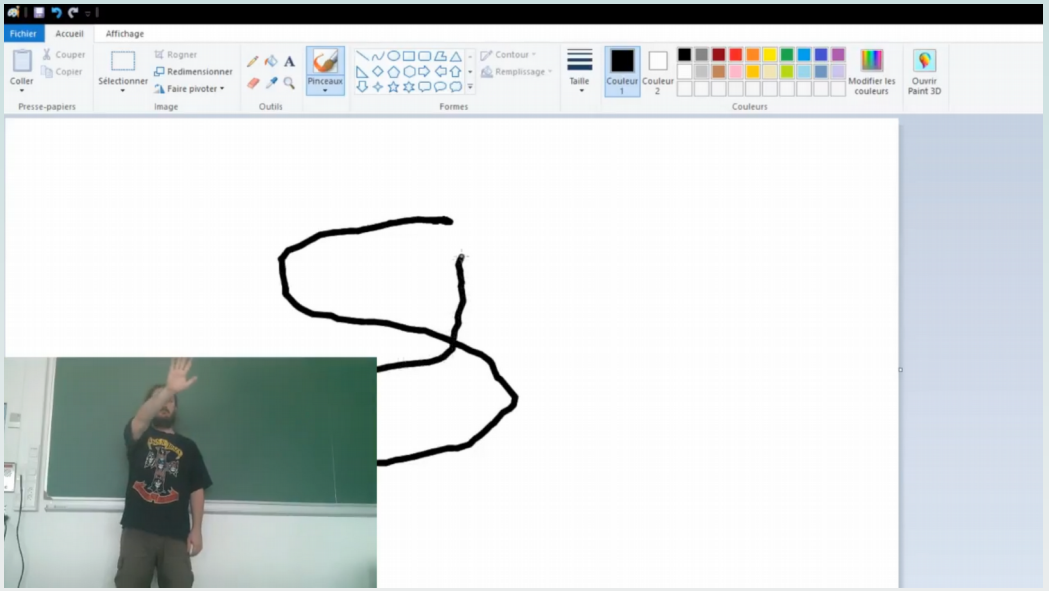
HAUTIER Thomas
FLÉCHEUX Joan

Pourquoi ce projet ?

- But : se passer de souris lors d’une présentation avec vidéoprojecteur
- Permet plus de souplesse lors de la présentation : pas besoin de surface pour utiliser la souris
- Pourquoi avoir choisi Kinect ?

Fonctionnalités du projet





Méthode de travail

- Découpe du projet en étapes clairement définies : captation du mouvement, lissage, clic gauche, clic droit, glisser-déposer.

- Approfondissement de nos connaissances en C++, appréhension d’une nouvelle API et d’un nouvel environnement de développement.

Déroulement du développement

- Sept mois et demi au total, soit un mois et demi de recherche et conceptualisation et six mois de développement.

- Répartition des tâches à peu près équitable

Architecture du projet

- Une classe principale contenant la boucle principale.

- Chaque autre classe correspond à un élément du logiciel.

Difficultés techniques

- Compréhension du fonctionnement d’un Kinect
- Lissage imprévu des données
- Absence de modularité du code = temps perdu

Aperçu du code

```
void Calibration::calibrationMouvement(NUI_SKELETON_FRAME* sframe, double **positions, Infos* env) {

    sf::SoundBuffer buffer;
    sf::Sound notif;
    NuiSkeletonGetNextFrame(0, sframe);

    buffer.loadFromFile("son.wav");
    notif.setBuffer(buffer);
    notif.play();
    notif.play();
    double tmpX, tmpY, t
    int check = 0;
    bool cal1 = false, cal2 = false;
    cout << "Debut calibration" << endl;
    Sleep(2000);
    for (int i = 0; i < 2; i++) {
        NuiSkeletonGetNextFrame(0, sframe);

        positions[i] = new double[3];
        for (int k = 0; k < 6; k++)
        {
            if (sframe->SkeletonData[k].eTrackingState == NUI_SKELETON_TRACKED) {
                tmpX = sframe->SkeletonData[k].SkeletonPositions[NUI_SKELETON_POSITION_HAND_RIGHT].x;
                tmpY = sframe->SkeletonData[k].SkeletonPositions[NUI_SKELETON_POSITION_HAND_RIGHT].y;
                tmpZ = sframe->SkeletonData[k].SkeletonPositions[NUI_SKELETON_POSITION_HAND_RIGHT].z;
                cout << "Calibration " << i << " : " << tmpX << tmpY << endl;
                positions[i][0] = tmpX;
                positions[i][1] = tmpY;
                positions[i][2] = tmpZ;
                if (i == 0) {
                    cal1 = true;
                }
                else if (i == 1) {
                    cal2 = true;
                }
            }
            else {
                cout << "skeleton not found" << endl;
            }

        }
        notif.play();
        Sleep(2000);
    }
    if (!cal1 || !cal2) { // Si la calibration échoue on la refait
        calibrationMouvement(sframe, positions, env);
    }
    else {
        notif.play();
        notif.play();
        env->setCalibration(true);
    }
};
```

- Bien plus simple qu’avant le refactor.
- Utilise des méthodes de l’API Kinect.

Conclusion

- Appréhension d’une API inconnue et d ‘un langage peu maîtrisé
- Structure du code primordiale
- Améliorations possibles