Rapport Lab3_COUCHE INTERPRÉTATION 《RECONNAISSANCE DE GESTES》

ZOU Zixin　　LI Runcong　　SIP Freder

Le thème de ce laboratoire est l'identification de gestes spécifiques via le capteur Kinect et le serveur FASST. Le programme termine la commande correspondante en reconnaissant un geste spécifique.

Dans ce programme, la version du logiciel et du serveur que nous utilisons est C ++, Kinect SDK_1.8, FAAST_1.2, Visual Studio2017.

Ce programme est divisé en deux parties:

- La première partie est le contrôle du programme "*paint*" par des gestes spécifiques.
- La deuxième partie consiste à modifier la forme, la couleur et la texture des graphiques 2D et des modèles 3D avec des gestes spécifiques à l'aide de l'interface interactive SFML.

Reconnaissance de geste

Nous avons utilisé deux structures pour définir les articulations du corps.

- Chaque articulation a ses propres valeurs de coordonnées inférieures x, y, z

```
struct JOINT3D {
    double x;
    double y;
    double z;
};
```

- Le squelette humain est composé de 25 articulations

```
struct Squelette3D {
    JOINT3D SpineBase; //0
    JOINT3D SpineMid;  //1
    JOINT3D Neck;      //2
    JOINT3D Head;      //3
    JOINT3D ShoulderLeft; //4
    JOINT3D ElbowLeft; //5
    JOINT3D WristLeft; //6
    JOINT3D HandLeft;  //7
    JOINT3D ShoulderRight;  //8
    JOINT3D ElbowRight;  //9
    JOINT3D WristRight;  //10
    JOINT3D HandRight;  //11
    JOINT3D HipLeft;   //12
    JOINT3D KneeLeft;  //13
    JOINT3D AnkleLeft;  //14
    JOINT3D FootLeft;   //15
    JOINT3D HipRight;  //16
    JOINT3D KneeRight;  //17
    JOINT3D AnkleRight;  //18
    JOINT3D FootRight;  //19
    JOINT3D SpineShoulder;  //20
    JOINT3D HandTipLeft;  //21
    JOINT3D ThumbLeft;  //22
    JOINT3D HandTipRight;  //23
    JOINT3D ThumbRight;  //24
};
```

Au cours du processus d'exécution, un total de deux squelettes humains sont stockés, qui sont «squelette actuel» et «squelette précédent». Nous déterminons quelle action l'utilisateur a faite par la relation de position des deux squelettes.

```
Squelette3D currentSqlt3DTrk0;
Squelette3D prviousSqlt3DTrk0;
```

Paint

Ouvrir & Fermer paint

Nous utilisons le geste "clap" pour contrôler l'ouverture et la fermeture du programme Paint.

Afin de détecter l'action clapping, nous définissons les variables suivantes :

- ***px*** : La distance entre les mains gauche et droite sur l'axe des X dans le squelette précédent.
- ***cx*** : La distance entre les mains gauche et droite sur l'axe des X dans le

squelette actuel.

- ***cy*** : La distance entre les mains gauche et droite sur l'axe des Y dans le squelette actuel.
- ***cz*** : La distance entre les mains gauche et droite sur l'axe des Z dans le squelette actuel.

Quand le programme détecte: ***px*** *> 0.1 et* ***cx*** *< 0.1 et* ***cy*** *< 0.1 et* ***cz*** *< 0.1,* Nous pensons que l'utilisateur a une action de frappe.

Ordre de peinture

Nous utilisons une fonction appelée ***geste3()*** pour contrôler que la souris se déplace avec la main droite et dessine avec la main droite en avant.

Le mouvement de la souris change en fonction de la position de la main droite du squelette actuel. Lorsque la main droite de l'utilisateur avance d'une certaine distance et la maintient, le programme effectue un appui gauche sur le bouton de la souris et déplace la main droite pour tracer une trace sur la toile.

Dans la fonction ***geste3 ()***, le paramètre formel ***mainDZ*** représente la position de l'axe Z de la main droite du squelette en cours.

Lorsque ***mainDZ*** <1,3, le système détermine que l'utilisateur a effectué un mouvement de la main droite et commence à peindre.

SFML INETRACTION INTERFACE

Contrôle graphique 2D

Le programme gère deux formes au total: lorsque la variable ***shapeInt*** = 0, cela signifie d' opérer sur le cercle et lorsque ***shapeInt*** = 1, cela signifie d' opérer sur le rectangle.

La forme initiale est un cercle et la couleur est verte :

- Lorsque la main droite est levée, la couleur de la forme actuelle devient rouge.
- lorsque la main droite est abaissée, elle devient bleue.
- Lorsque la main gauche est levée, la forme devient un rectangle.
- Lorsque la main gauche est abaissée, la forme devient un cercle
- La main droite est légèrement levée vers l'avant pour changer la texture de la forme actuelle

1. Changer de couleur

Nous définissons une fonction appelée ***drawShape(sf :: Color color)*** pour changer la couleur de la forme actuelle.

Levez la main droite:

prviousSqlt_mainD_Y < prviousSqlt_ shoulderD_Y et

currentSqlt_mainD_Y > currentsSqlt_ shoulderD_Y

la main droite vers le bas :

prviousSqlt_mainD_Y > prviousSqlt_ shoulderD_Y et

currentSqlt_mainD_Y < currentsSqlt_ shoulderD_Y

2. Changer de forme

- S'il n'y a actuellement aucune texture, utilisez la fonction drawShape (sf :: Color

color) pour modifier la forme et prendre la couleur de la forme précédente.

- Si la forme actuelle a une texture, utilisez la fonction drawShape () pour modifier la forme et utiliser la texture de la forme précédente.

Levez la main gouche + la main gouche vers le bas :

Le principe est identique à celui de la main droite, il suffit de changer le nom de la variable pour les épaules gauche et la main gauche.

3. Changer la texture

Fonction ***changeTexture()*** pour changer les données de texture actuellement.

Nous définissons également une fonction surchargée ***drawShape()*** pour modifier la texture de la forme actuelle.

La main droite est légèrement levée vers l'avant :

prviousSqlt_mainD_Z > 1.3 et currentSqlt_mainD_Z < 1.3

L'interaction d'objet en 3D avec OpenGL

1. La rotation et couleurs

En ce qui concerne l' interaction d' objet 3D, nous utilisons opengl pour créer l' objet en 3D dans un premier temps. Pour cela, nous devons créer quatre vertex ayant des coordonnées (x, y, z) pour chaque face du cube. Ce qui fait au total 24 vertexs pour fabriquer un cube. Pour colorier une face d' un cube il faut utiliser une fonction de couleur appartenant à opengl. Il est important de l' utiliser à chaque début d' un vertex. En ce qui concerne la rotation, on utilise une fonction de rotation dans lequel on initialise un int rotation de 1.

Le cube peut changer de rotation et de couleur une fois que notre main droite est en l' air.

2. La Texture

Dans notre programme nous pouvons aussi charger une texture dans notre cube. Plus précisément, on peut charger une image de format bmp. Pour cela on a utilisé une fonction load_bmp qui permet de charger les données de l' image Bmp et son en-tête pour l' utiliser dans le cube. Pour l' afficher dans les faces, on utilise une function appelé TextCoord2i où on doit le placer avant les vertexs et indiquer par où commence la texture.

De même pour un cube texturé on peut changer sa rotation et sa couleur avec les mêmes fonctions dites précédemment.

Rapport Project

RA&RV application:

Jeu de basket

Ce projet est faire une application VR de basket-ball tir la balle. Nous utilisons Oculus comme dispositif d'affichage. Pour ce projet, nous nous sommes concentrés sur l'interaction des scènes virtuelles avec le monde réel. Et nous avons essayé de résoudre certains des problèmes de la motion sickness.

Dans ce programme, la version du logiciel et du serveur que nous utilisons est Oculus Integration_1.32, Unity3D_5.6.5, Visual Studio2017.

Ceci est un simulateur de shooting. Dans ce application, on peut tirer comme dans le monde réel. Nous rejoignons le système de notation. Et la difficulté du jeu augmentera après l'obtention d'un certain score.

Notre réalisation:

- **Scène 3D**:

Tous les objets 3D requis ont été créés. Terrain de basket, panier de basket, basket, modèle de main, petit monstre, tableau de bord

- Oculus Rift:

Les joueurs peuvent tirer à travers le casque et la poignée Oculus Rift.

1. Les joueurs peuvent se déplacer avec le joystick de la poignée gauche

2. Le joueur saisit le ballon de basket avec l'action agrippante de la poignée gauche / droite (quand il est assez proche du ballon)

3.Le joueur effectue une action de tir et relâche la poignée qui saisit le ballon de basket. Le basket-ball tire

4. Le joueur évite l'approche du monstre et tire pour l'empêcher de bouger.

5. Lorsque le joueur est touché 3 fois par le monstre, le jeu redémarre

- Le jeu a deux modes: le mode normal et le mode fantôme.

1. En «mode normal», les joueurs peuvent s'exercer librement au tir. Quand un certain score est atteint, le jeu entre automatiquement en mode fantôme (3 buts)

2. Dans le "mode Ghost", le joueur doit éviter l'approche du monstre et lancer un but pour s'empêcher de "mourir" et la partie redémarre (HP initial du joueur = 3).

Reduce Motion Sickness:

1. La caméra suit toujours sans délai le mouvement de la position du joueur: Laisser le sens de rotation de l'utilisateur correspondre à la vision.
2. La main virtuelle est une référence fixe

Les concepts que nous avons étudiés:

- Production de la scène de jeu Unity3D:

Nous avons créé des scènes 3D de matchs de basketball à l'aide de didacticiels en ligne et de modèles Unity3D existants.

- Connexions Unity5 et Oculus

Unity5 et Oculus Rift développent conjointement des jeux de réalité virtuelle

Le pack d'intégration Oculus est désormais disponible au téléchargement directement à partir du magasin de ressources Unity, qui inclut les modèles SDK et VR Oculus pouvant être utilisés directement (comme les mains virtuelles). Grâce à ceux-ci, nous pouvons créer rapidement et facilement une application de réalité virtuelle dans Unity.

- Réduire Motion Sickness généré par les jeux de réalité virtuelle

Au début du projet de production, nous espérons utiliser le capteur Kinect pour créer un jeu de réalité virtuelle somatosensoriel afin de réduire le sentiment de vertige, mais Kinect ne joue pas un rôle important dans ce jeu. Il réduit considérablement l'expérience du jeu. nous avons choisi d'utiliser le manche Oculus. Nous ajoutons une référence statique au joueur et nous nous assurons que le délai de jeu est faible et que le joueur se déplace dans la direction de la caméra. Nous n'avons pas choisi de réduire le FOV pour réduire les vertiges, car le FOV a un meilleur effet sur les jeux rapides et à plus de textures, et cela n'a pas d'importance dans notre jeu.

Interaction dans le jeu:

Notre interaction est étroitement liée à l'oculus.

Le mouvement de l'utilisateur est obtenu par le levier sur la poignée. Dans le script OVRPlayerController.cs, nous définissons les actions de l'utilisateur telles que le mouvement, la direction et le saut etc.

L'utilisateur se déplace vers la balle et tient la poignée pour faire une prise (appuyez sur le bouton). La balle a été saisie. Ce processus est écrit dans le OVRGrabbable.cs et OVRGrabber.cs script.

Ensuite, l’utilisateur peut lancer une action dans n’importe quelle direction et la balle partira à une certaine vitesse et accélération. La vitesse est déterminée par la force du shooting de l'utilisateur. Ce processus est écrit dans le OVRGrabbable.cs et OVRGrabber.cs script aussi.