Rapport de Projet

Modélisation Géométrique : Création et suivi de trajectoire de caméras

Parcours : Multimédia

Année universitaire 2022-2023

Clémentine Grethen et **Tom Bonetto**
Groupe M1

0.1 Introduction

L'objectif de ce projet est de construire, à partir des connaissances acquises durant le cours d'interpolation et d'approximation, un module sous Unity3D qui permet de réaliser un suivi de trajectoire de caméra en ne fournissant qu'un nombre restreint d'informations, à savoir quelques points.Ce projet comporte deux composantes : le suivi de position et le suivi de rotation.

0.2 Notre modèle : Présentation, avantages et limites

Pour notre scène nous avons choisi un modèle 3D représentant une ville afin de mieux visualiser le mouvement de caméra et son parcours.

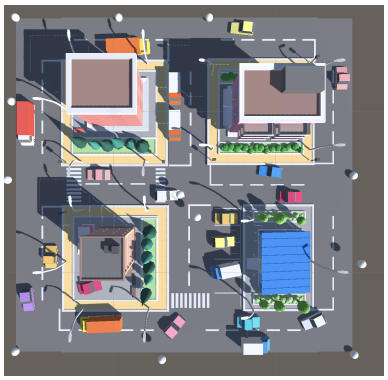

FIGURE 1 – Modèle d'un mini-ville qui constituera notre scène

Pour définir notre trajectoire nous avons placé des points sur la scène (Fig1). Nous avons crée deux configurations de points pour visualiser deux parcours : il suffit de cocher l'une des deux "Sphere" dans l'architecture du projet(cf Fig.2).
Enfin, pour le suivi de position nous avons fait le choix de traiter deux méthodes en laissant le choix à l'utilisateur (en cliquant sur l'objet "projet" puis type "Neuville" ou "Splines" cf Fig.2) : interpolation de Neuville et les Splines

L'intérêt de notre modèle est qu'il est visuel ainsi on peut facilement observer le bon fonctionnement de notre parcours de caméra qui simule une "balade" dans une ville. De plus, nous pouvons comparer l'efficacité des deux méthodes pour la position grâce à la flexibilité des méthodes (avec un contrôle utilisateur).

Cependant, nous avons fait le choix de placer en amont nos sphère ce qui ne permet pas à l'utilisateur de pouvoir choisir la trajectoire qu'il souhaite (à cause d'un problème que nous avons rencontré sur Unity). Il faudrait créer une scène contenant le chemin souhaité avant chaque Play.

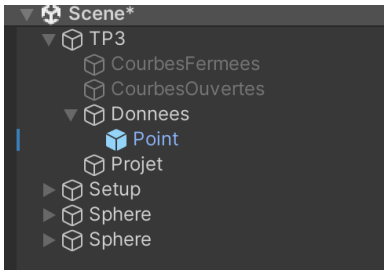

FIGURE 2 – Architecture de notre projet

0.3 Notre algorithme

Nous avons notre unique code contenu dans le fichier "projet.cs" qui réalise le suivi de position et le suivi de rotation.Pour réaliser un suivi de rotation on réalise une interpolation sphérique d'où l'utilisation des quaternions (CF CTD2).On va alors définir des listes pour stocker les points de la courbe de parcours de la caméra(ListePoints,ListePointsSub,ListeQuaternions,ListeQuaternionsSub) ainsi que les points du polygone de contrôle (Px,Py,Pz) et d'autres valeurs classiques (pas,..).

Nous avons, comme expliqué précédemment, réalisées deux stratégies pour le parcours. La première concerne Neuville : nous avons construit une paramétrisation de Tchebycheff pour l'échantillonnage (nous avions vu dans le TP1 que c'était la plus précise), puis la fonction Neuville qui effectue l'interpolation de neuville (private (Vector3, Quaternion) neville(List<float> X, List<float> Y, List<float> Z, List<Quaternion> Q, List<float> T, float t)).
Cette fonction est presque la même que celle du TP1, cependant nous devons prendre en compte les Quaternions : une interpolation sphérique (slerp) est effectuée entre Qi[1] et Qi[k+1] en fonction de t et on retourne un tuple constitué d'un Vector3 (représentant le point interpolé) et d'un Quaternion (représentant la rotation interpolée).

On peut alors créer la fonction applyNevilleParametrisation qui applique l'algorithme de Neville à une série de points et de Quaternions pour générer une courbe interpolée à des instants de temps spécifiques. Pour chaque instant dans la liste tToEval, elle produit un point et un Quaternion interpolés. Ces points et Quaternions interpolés sont ensuite renvoyés sous forme de listes.

Pour les splines : on réalise la fonction subdivise qui prend en entrée une liste de points (en 3 dimensions X, Y, Z) et une liste de Quaternions. Elle effectue alors un certain nombre d'itérations pour subdiviser les points et les quaternions, créant ainsi une courbe plus lisse. À chaque itération, chaque point (et quaternion) est dupliqué, puis chaque pair de points consécutifs (et quaternions) est moyenné pour créer un nouveau point (et quaternion) intermédiaire.
Neuville et splines permettent de construire la trajectoire de la caméra.

Le script récupère tous les objets de type "Player", trie par ordre alphabétique et calcule les positions et quaternions pour chaque point de contrôle. Il utilise ensuite soit la méthode de Neville soit la méthode Spline pour créer une courbe lisse à partir des points de contrôle. FixedUpdate est appelé à intervalles de temps réguliers et s'occupe de bouger la caméra le long de la trajectoire en plus de l'orienter. Enfin, OnDrawGizmosSelected dessine ces courbes dans l'éditeur Unity.

0.4 Résultats

Nous avons implémenté deux types de méthodes : interpolation et splines, ce qui nous permet d'observer deux aspects différents dans le suivi de la position. Avec Neville et la paramétrisation de Tchebychev, on force notre objet à passer par les différents points de contrôle, on obtient ainsi des trajectoires prévisibles et régulières. Cependant, la forme de la trajectoire est très peu maitrisée car celle-ci oscille lorsque les directions changent brutalement entre les points de contrôle. Vu que l'objectif de notre projet est de faire passer notre caméra dans les rues de notre modèle de ville, la trajectoire a besoin d'être maîtrisée et fluide pour donner une expérience réaliste et agréable. Les splines remplissent ce rôle parfaitement car elles sont très flexibles, en jouant sur le degré des polynômes on peut forcer notre trajectoire à suivre de près le polygone de contrôle tout en restant à l'intérieur de son enveloppe, il n'y a pas d'oscillation. Au final, c'est avec les splines que les résultats sont les plus satisfaisants en ajustant le degré on obtient une meilleure trajectoire qu'avec Neville sur les deux configurations de points, d'autant plus que nous avons un problème sur l'orientation de la caméra dans le cas de Neville.

LIEN DU PROJET : https ://drive.google.com/file/d/17HzUYll6B2FqytJYeXDPiRPlJ9XseUig/view ?usp=sharing