Création et suivi de trajectoire de caméras

Omar NAIM
Saad MDAA
Younes EL BOUZEKRAOUI

Département Sciences du Numérique - Deuxième année
2020-2021

1 Introduction

A partir de positions des points définies et en utilisant les quaternions, nous avons programmé sous Unity la trajectoire d'une caméra dans une scène. Le but de ce projet est de réaliser un module sous Unity 3D qui permet de réaliser un suivi de trajectoire de caméra à partir de quelques points.

2 Suivi de la rotation

Afin de définir la rotation de la caméra, nous utilisons l'interpolation sphérique entre les points trajectoires tout en orientent chacun d'entre-eux vers le sujet focalisé.
Pour faciliter l'orientation du point nous avons défini le script PointTrajectoire.cs qui permet de dessiner la droite de direction indiquant l'angle visé par nôtre point.

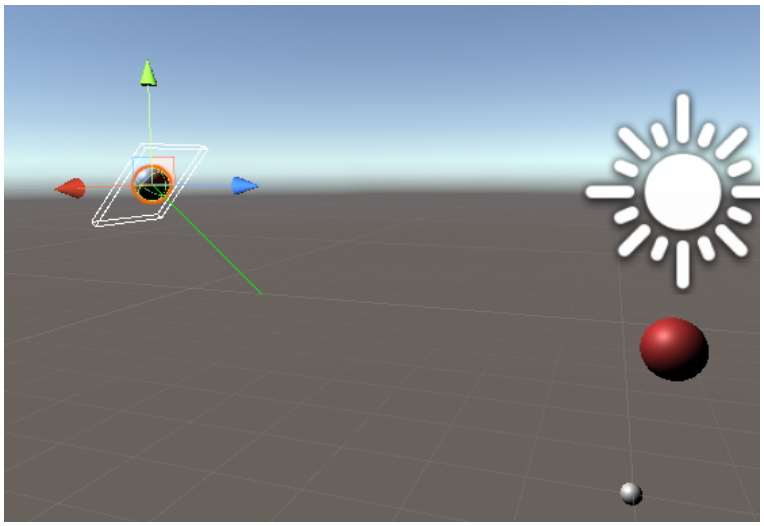
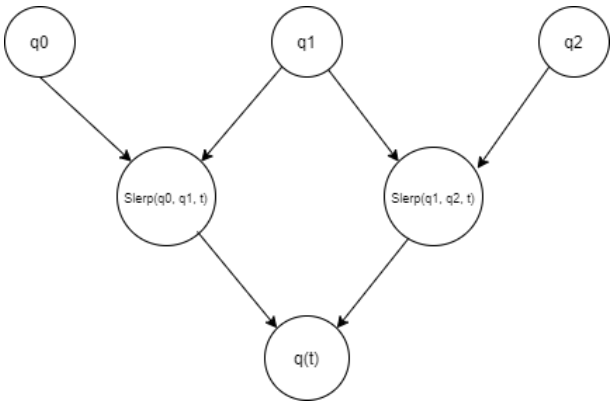


Figure 1: Visualisation de l'orientation d'un point trajectoire

Le module Quaternion nous fournit la fonction $Slerp(q_1, q_2, t)$ qui réalise l'interpolation sphérique entre q_1 et q_2 au temps $t \in [0, 1]$.
Afin de le généraliser pour n quaternions nous utilisons un algorithme similaire à celui de Neville :



3 Suivi de la position

Pour réaliser le suivi de la position, nous avons défini deux méthodes d'approximation : approximation par courbe de Bézier (DeCasteljau) et approximation par B-splines uniformes (Subdivise)

3.1 Approximation par courbe de Bézier

Nous avons choisi d'implémenter l'algorithme de DeCasteljau par évaluation en utilisant la formule suivante :

$$P(t) = \sum_{k=0}^{n} B_k^n(t) P_k$$

où

$$B_k^n(t) = \binom{k}{n} (1-t)^{n-k} t^k$$

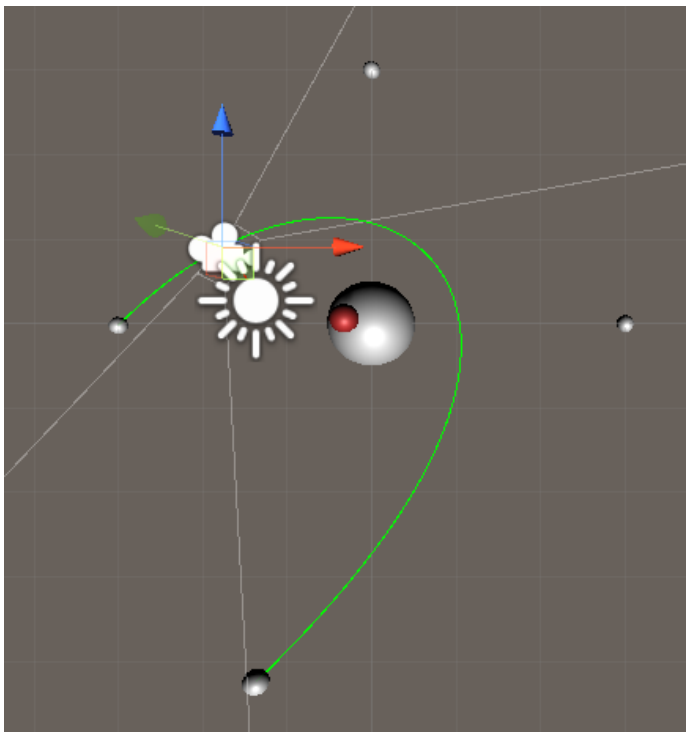


Figure 2: Approximation par courbe de Bézier

L'algorithme de DeCasteljau peut être implémenté récursivement avec une compléxité de l'ordre $O(2^n)$ ou itérativement avec une complexité $O(n^2)$, du coup il est plus intéressant d'opter pour la version itérative.

3.2 Approximation par courbes B-splines

En ce qui concerne l'approximation par courbes B-splines, nous utilisons l'algorithme simple de subdivision pour tracer des B-splines uniformes :

- Dupliquer les points de contrôle.
- Prendre le milieu de deux points consécutifs dans le polynôme de contrôle. (degres fois).
- répéter les deux étapes précédentes nombreIteration fois.

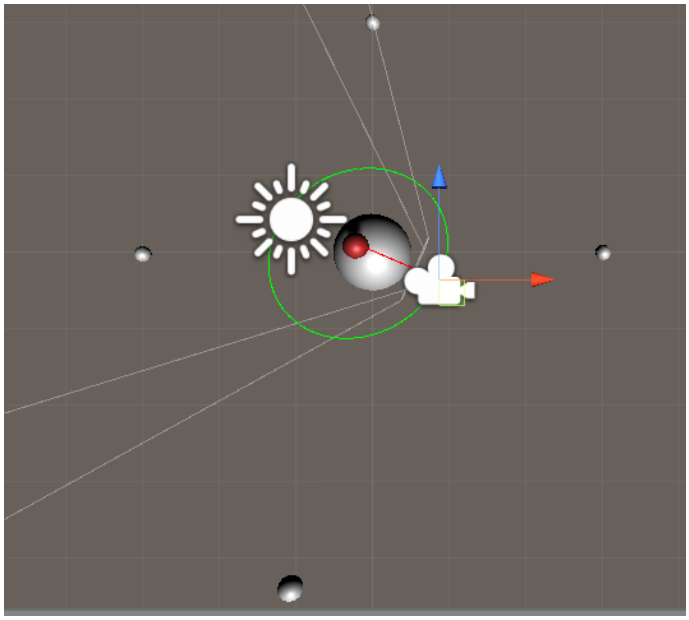


Figure 3: Approximation par courbes B-splines

4 Conclusion

Pour conclure, ce projet nous a permis de mettre en pratique les connaissances acquises sur les approximations via les courbes de Bézier (Approximation DeCastalejau dans notre cas) et sur les approximations via les B-splines, que nous nous sommes amuser à coder de façon itératives ainsi que récursives.
Nous avons grâce à Unity réussi à modéliser puis à visualiser la trajectoire d'une caméra qui suit sa cible et cela en ne fournissant qu'un nombre limité d'informations, 4 points dans notre cas. Pour mieux visualiser la trajectoire, nous avons introduit une droite qui règle l'orientation de la caméra comme cela nous pouvons mieux initialiser la vision des caméras et voir l'évolution des points au cours du temps.
Nonobstant, le fait de régler les points manuellement devient de plus en plus fastidieux plus le nombre de points augmente, vu qu'il faut pour chaque point régler sa rotation sur chaque dimension. Il serait donc mieux, si nous pouvions voir directement ce que voit le point et contrôler sa rotation comme dans le cas du logiciel Blender.