

RAPPORT PROJET INF443

JEU DE BASKET

Clément Barbier
Tzu-Yi Chiu

31 MAI 2019

Table des matières

1	Introduction	1
2	Mise en scène 3D	1
2.1	Création du terrain	1
2.2	Animation du joueur	2
2.3	Gestion de collisions	3
2.4	Interaction avec l'utilisateur	3
2.5	Suivi de la caméra	4
3	Bilan	4

1 Introduction

Le projet sur lequel nous avons travaillé consiste en la conception d'un jeu de basket dans lequel un joueur est capable de lancer un ballon pour marquer des paniers. La mise en scène 3D consiste en cinq parties : la création du terrain, l'animation du joueur tirant des ballons, la gestion de collisions ballon-sol et ballon-panier, l'interaction avec l'utilisateur et enfin le suivi de la caméra. L'objectif de créer un monde respectant le plus près possible à la physique du monde réel, ce qui oblige à bien prendre en compte les forces et les rebonds. Par ailleurs, nous avons souhaité que l'utilisateur puisse contrôler et faire tirer le joueur avec le clavier en intégrant notamment le suivi de la caméra pour une meilleure immersion.

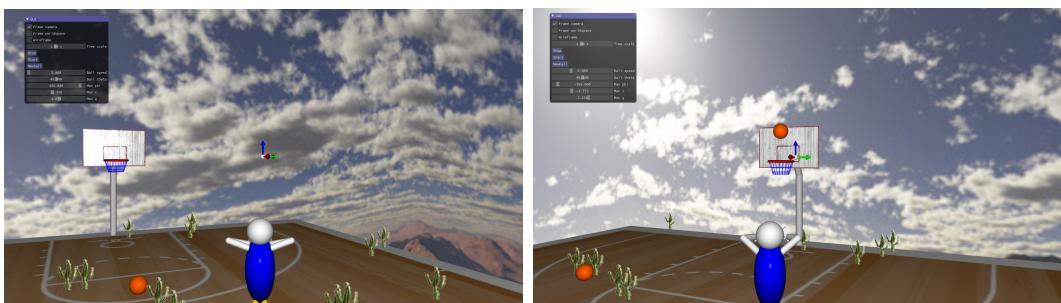


FIGURE 1 – Scène du jeu de basket

2 Mise en scène 3D

2.1 Crédation du terrain

Nous sommes partis d'un terrain plat dont la longueur et la largeur correspondent à ceux d'un terrain de basket auquel une texture adaptée a été ajoutée. De plus, nous avons souhaité que ce terrain se place dans un environnement désertique d'où l'ajout d'une skybox représentant un désert et de billboards correspondant à des cactus. Les paniers sont ensuite construits à partir d'une hiérarchie, dans laquelle nous avons créé 3 cylindres connectés par 2 balles de jonctions, tous du même rayon, un panneau sur lequel nous mettons une texture et un cerceau rouge.

Les cerceaux sont des meshs construits à partir du même principe que les autres meshs primitifs, possédant également deux champs qui nous permettent de récupérer la position et le vecteur normal de chaque sommet. Ceci est indispensable pour la gestion de collisions entre le ballon et toutes ces parties de la hiérarchie dans la suite.



FIGURE 2 – Panier sans les filets

Pour bien modéliser le comportement réel des filets, nous avons choisi le modèle masse-ressort comportant 20 fils verticaux. Chaque fil vertical possède quatre particules de petites masses, l'une reliée à l'autre par un ressort de petit raideur. Ensuite nous avons également relié les particules horizontalement. Chaque particule de masse subit ainsi 6 forces (sauf les extrémités) : les 4 forces de rappel des particules voisines, la gravité et le frottement dû à l'air. Entre chaque frame, la position et la vitesse sont ensuite mises à jour selon le PFD :

$$v = v + dt \cdot \frac{F}{m}$$

$$p = p + dt \cdot v$$

Les longueurs à vide sont à définir avec les valeurs appropriées. En effet, si la longueur à vide est trop petite, les particules qui se trouvent au même étage se poussent l'une à l'autre et forment une étoile.

De même si la force de frottement est prise trop petite, nous avons pu constater une agitation des particules qui entraîne la divergence de leur position et leur vitesse. Ceci est potentiellement dû aux erreurs informatiques qui s'accumulent à long termes.

Par souci de simplicité, comme la modélisation des filets est relativement compliquée, nous avons choisi de ne pas intégrer les filets dans la hiérarchie du panier. Par contre il n'est pas difficile de récupérer la position absolue du cerceau, dont nous nous sommes servis pour “accrocher” les filets.

2.2 Animation du joueur

Tout comme le panier, le joueur est modélisé à partir d'une hiérarchie. Le corps est modélisé à partir d'une sphère déformée sur laquelle on ajoute la tête, les bras puis les jambes. Initialement, les bras et avant-bras ont subi une rotation de manière à ce qu'au repos le joueur tienne le ballon entre ces deux mains.

Le joueur peut se déplacer sur tout le terrain en modifiant ses coordonnées x et y et en modifiant l'angle φ désignant la direction regardée par le joueur. Enfin, lorsque l'utilisateur ordonne au joueur de lancer le ballon, les bras du joueur vont s'animer pour rendre plus réaliste ce lancer. Par ailleurs, le mouvement des bras est

réglé pour que l'angle entre les bras et le sol corresponde bien à celui du lancer de la balle.

2.3 Gestion de collisions

Une fois la balle lancée, sa trajectoire est actualisée à chaque frame selon les lois de la physique en prenant en compte la gravité et les frottements dus à l'air. Mais au cours de sa trajectoire, la balle peut rencontrer des éléments extérieurs comme le sol, la planche du panier ou encore le cerceau. Pour modéliser les rebonds sur la planche et le sol, il suffit tout simplement d'inverser respectivement les composantes horizontales et verticales de la balle. Pour modéliser la collision avec des objets plus complexes comme le cerceau, il suffit de déterminer avec quel sommet de l'objet, la balle entre en collision puis en connaissant la normale de la surface de l'objet en ce sommet, on en déduit le changement de trajectoire en calculant une matrice de rotation liée au rebond.

Les interactions entre la balle et le filets ont également été modélisées. Si la balle rentre en contact avec une des masses du filet, celle-ci va se retrouver projeter dans la direction du vecteur vitesse de la balle. Par souci de simplicité puisque les masses du filets ont une masse très inférieure à celle de la balle, l'action réciproque du filet sur la balle a été négligée. Enfin nous avons pris en compte des phénomènes dissipatifs lors du rebond, en effet à chaque collision de la balle avec un objet extérieur, en plus de voir changer la direction de son vecteur vitesse, la norme de sa vitesse est également atténuee pour davantage de réalisme.

2.4 Interaction avec l'utilisateur

Nous avons déjà pu modifier tous les paramètres du joueur (position et orientation) et du ballon (inclinaison et vitesse initiale) depuis le gui d'où nous avons stocké ces informations. Néanmoins pour davantage d'ergonomie si nous pouvons également contrôler et faire tirer le joueur depuis le clavier. Une nouvelle fonction `key_press` est appelée depuis le `main.cpp`. Au lieu de déplacer le joueur seulement selon les axes *x* et *y*, les flèches vers la gauche et vers la droite permettent de changer l'orientation du joueur et les flèches vers le haut et vers le bas permettent d'avancer et de reculer **selon son orientation** tout en restant sur le terrain. Pour lancer un ballon, nous avons créé une variable globale booléenne `new_ball` qui s'active depuis le gui ou en appuyant sur Entrée. Une fois `new_ball` activée, le joueur lance un ballon selon les paramètres choisis au départ. À ce stade, l'utilisateur possède tous les outils pour contrôler le joueur. Or pour que le joueur soit toujours visible dans la fenêtre, le suivi de la caméra nécessite la souris. Nous avons donc besoin d'automatiser la position et l'orientation de la caméra pour garder le joueur en vue pour simuler un vrai jeu de vidéo. Ceci introduit donc notre dernière partie du travail.

2.5 Suivi de la caméra

Le suivi de la caméra devrait être réglé de manière à ce qu'elle suive en permanence le joueur. Ses paramètres comme l'orientation et la translation par rapport à sa position initiale sont entièrement déterminés par ceux du joueur. Pour ce faire, nous nous sommes servis des méthodes déjà implémentées dans *camera.scene* comme *apply_rotation* et *apply_translation*. Ensuite, pour que le suivi soit cohérent, il nous a fallu bien initialiser la caméra. Ceci se fait beaucoup plus facilement avec les fonctions que nous avons implémentées nous-même qui permettent de choisir l'orientation et la position absolues de la caméra : *set_rotation* et *set_translation*. Ainsi, lors du déplacement, l'utilisateur n'a plus besoin de toucher à la souris pour régler correctement la caméra pour plus de confort pour l'utilisateur.

3 Bilan

L'étude de ce projet nous a permis de nous familiariser avec OpenGL et C++ et de se confronter aux difficultés que rencontrent les développeurs de jeux vidéo. Certaines interactions telles que les collisions entre le ballon et les objets du terrain ont été implémentées. Une suite possible serait de programmer le dribble du joueur par exemple. Ce dernier est parfaitement faisable en modélisant la collision entre le ballon et le corps du joueur : il faudrait juste connaître la position absolue des mains lorsque le joueur est en déplacement ; ensuite pour que le jeu se rapproche au réalisme, il est un peu plus subtil de gérer le mouvement du corps du joueur.

En conclusion, ce projet fut très intéressant dans la mesure où, en plus des outils que présentent les tutoriels qui nous permettaient de créer un monde que nous désirons, nous avons aussi réussi à bien modéliser les rebonds, à faire intervenir l'utilisateur et à gérer les paramètres de la caméra tout en restant autonomes dans la recherche et l'implémentation des fonctions qui ne sont pas nécessairement évidentes.



FIGURE 3 – Terrain de basket

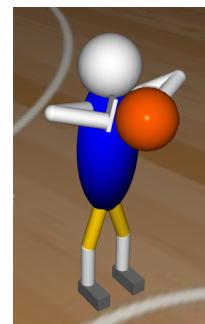


FIGURE 4 – Petit bonhomme