Interface de visualisation 3D pour la Small-Size League de la RoboCup Soccer

Unity

Lucie MATHÉ

6 janvier 2020

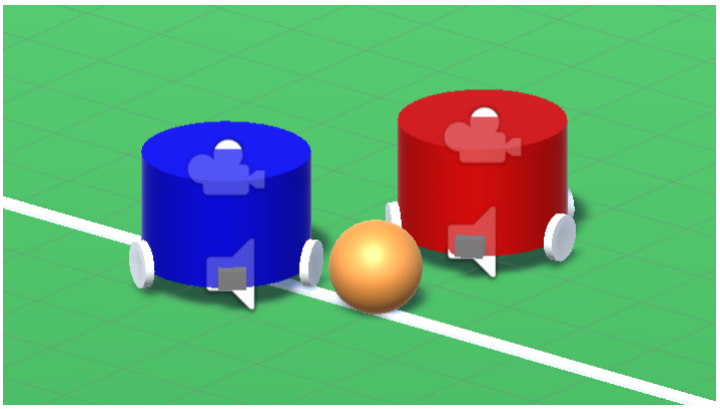

Table des matières

Chapitre 1

Contextualisation

La RoboCup est la plus grande compétition internationale de RoboCup. Cette année, la compétition se déroule à Bordeaux du 25 au 29 juin. Parmis les cinq ligues majeures, on retrouve entre autre la RoboCup Soccer.

À l'origine, cette ligue avait été créée suite à l'objectif lancé par H. Kitano en 1996 : "by the year 2050 develop a team of fully autonomous humanoid robot that can win against the human world soccer champions". Pour atteindre ce but on retrouve majoritairement la ligue humanoïde dont l'équipe bordelaise de Rhoban tiens le titre de champion en catégorie KidSize (robot de taille inférieure à 1m).

Parrallèlement à cela, on retrouve la Standart Platform League où ce sont des robots Nao qui jouent au foot et deux ligues de robots roulants la Midle-Size League et la Small-Size League.

Nous pouvons donc également retrouver à Bordeaux une équipe de Small-Size League appelée NaMeC. Pour ce projet, je trouvais intéressant d'essayer de créer un simulateur de match pour l'équipe de Rhoban dans laquelle je fais partie mais, simuler le déplacement d'un robot à 20 degrès de liberté me semblait un peu ambitieux pour ce projet. En revanche, la SSL possède des robots holonomes à 4 roues, ceci me semble davantage abordable. Je souhaite donc aider cette équipe débutante (créée en 2018) et leur fournir un outil de visualisation 3D.

Chapitre 2

Objectifs fixés

Ce projet serait donc une simulation 3D d'un robot de SSL dans son environnement de jeu (un terrain de football).

Le but à l'issu de ce projet est de faire marquer un but au robot. J'aimerai placer la balle au centre du terrain, et permettre à un robot de traverser le terrain et taper dans la balle pour marquer un but.

L'objectif se partageait en plusieurs tâches :

— Créer un terrain de foot
— Créer un robot de SSL
— Implémenter le déplacement d'une balle en fonction force exécutée par le robot
— Implémenter un léger autopilote

Lors de la compétition, les robots reçoivent le flux vidéo d'une caméra en TopView du terrain. Le traitement d'image est effectué par l'arbitrage, par un code commun à toutes les équipes. L'équipe de SSL reçoit des positions des différents robots et de la balle extraits de l'image.

Le seul capteur embarqué sur le robot est un capteur de proximité situé devant le kicker (partie du robot qui tir dans la balle) pour détecter si la balle est présente où pas. J'ai donc décidé de simuler un capteur de distance.

Le but ici était de créer deux modes, l'un où le joueur contrôle le robot, l'autre où il y a un autopilote.

Chapitre 3

Création des objets

3.1 Le terrain

Un terrain de foot est symétrique par l'axe horizontal et vertical. Pour créer le terrain, j'ai dupliqué quatre quarts de terrain. Chaque quart est composé d'un sol et des bordures. J'ai ensuite tracé les lignes et ajouter un bloc permettant de faire un des deux poteaux des cages.

Le terrain a été créé de taille réduite par rapport à un vrai terrain (1 pour 4 pour la taille totale, 1 pour 2 pour les cages et la zone du gardien). Une fois les quatre parties du terrain assemblée, j'ai rajouté une ligne représentant la ligne médiane du terrain. Je n'ai pas ajouté le rond central car il est peu utile et ne semblait pas évident à faire.

3.2 Le robot

J'ai basé les tailles de mon robot sur les règles du jeu. Le cylindre de base du robot mesure donc 18 cm de diamètre. Pour la hauteur, je suis passée des 15 cm obligatoire à 18 car je trouvais ça plus homogène.

Le robot dispose d'un kicker lui permettant de tirer dans la balle. Ici, il est juste représenté par un cube.

J'ai ensuite ajouté les 4 roues. Les roues de SSL ne sont pas placées avec un écart de 90 degrès. Les deux roues à l'avant sont légèrement écartées dans le but de laisser la place au kicker. J'ai donc dupliqué le cylindre de base. J'ai décalé l'orientation de 25 degrès ce qui m'a permis de décaler le positionnement de la roue que j'ai placé depuis ce cylindre.

J'ai ajouté une caméra embarquée dans le robot afin de pouvoir le manipuler comme un jeu FPS (First Person Shooter). L'équipe ne possède pas de caméra mais c'est une évolution possible dans la ligue.

3.3 La balle

La balle est simplement une sphère, comme pendant la RoboCup, celle-ci est orange.

Je voulais la rendre plus légère que le robot (dont la masse est égale à 1) mais après quelques essais, j'ai préféré augmenter la masse du robot (maintenant de 20).

Également, j'ai dû trouvé une solution pour ajouter de la friction entre la balle et le sol. En effet, comme Unity est un monde parfait, lorsqu'un robot pousse la balle, celle-ci roulait ensuite jusqu'à l'inifini. J'ai donc changé le *drag* et le *angularDrag* afin de les mettre tous les deux à 1 pour augmenter le frottement de l'air sur la balle.

3.4 Agencement des caméras

Le jeu se compose de quatre caméras. Une est embarquée dans chacune des robots, une en TopView et la dernière pour le score.

J'ai agencé l'écran de sorte que les deux caméras des robots se partagent l'intégralité de l'écran et la caméra de TopView est centrée en bas (comme dans les jeux de sport courants) et le score est ajouté tout en haut, aligné avec la TopView.

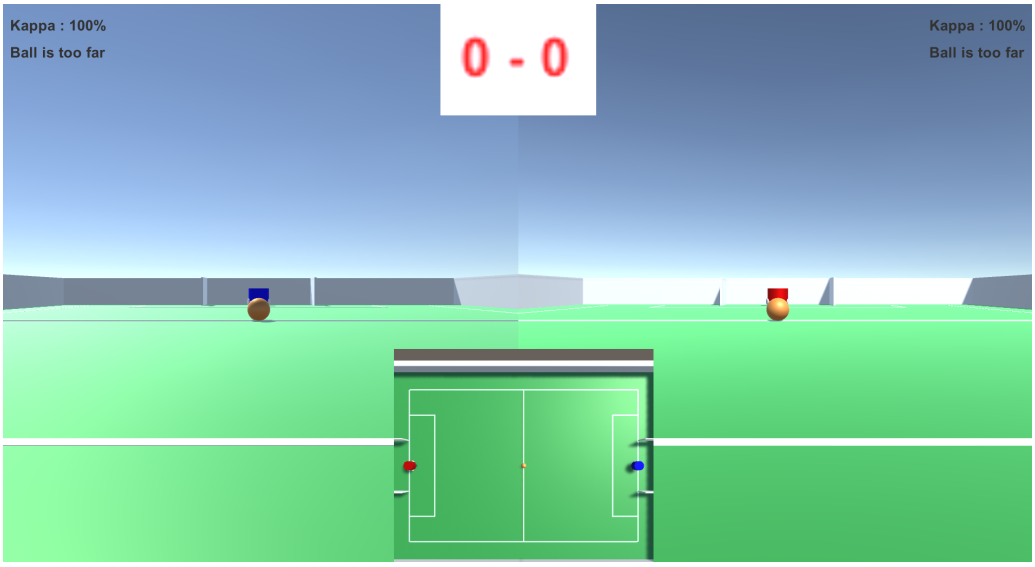

FIGURE 3.1 – Vue du jeu

Chapitre 4

Fontionnalités implémentées

4.1 Déplacement manuel du robot

Pour implémenter le déplacement j'ai décidé d'utiliser les forces afin d'avoir un déplacement très fluide du robot.

Ma première intention a été de changer la vitesse du robot en ajoutant de l'accélération. Ensuite j'ai voulu faire un bouton d'arrêt d'urgence afin de pouvoir stopper net le robot. Or, si l'on met la vitesse du robot à 0, alors rajouter de l'accélération ne le permettra pas de le redémarrer. J'ai donc préféré effectuer le changement de vitesse grâce à velocityChange. J'ai ensuite tuner à la main la force à effectuer pour que le jeu soit jouable par un joueur.

Pour la rotation, j'utilise simplement la fonction rotate d'un pas petit pour que la rotation soit fluide et depuis le repère du robot.

Le déplacement se fait exclusivement à l'aide des touches du clavier (voir Section 6).

4.2 Tirs dans la balle et buts

4.2.1 Chargement d'un tir

Sur les robots de SSL, le tir est effectué par des Kappas qui chargent de l'électricité et qui déchargent une partie de cette électrité lors d'un tir. Après avoir discuté avec des membres de l'équipe de NaMeC, j'ai reproduit un schéma similaire : le tir est disponible lorsque la kappa est à plus de 60% de charge et chaque tir fait baisser la kappa de 30%.

Le tir ne peut être effectué que si le kicker du robot est en contact avec la balle. Pour simpifier le fichier, j'ai fait un script GETKICKED que j'ai ajouté

à la balle. Ce fichier prend en compte le tir pour les deux robots en même temps. La balle est ensuite "tirée" par l'exertion d'une force proportionnelle au chargement de la kappa et dans le sens donné par le robot.

4.2.2 Affichage distance et puissance

La distance est simplement calculée en utilisant la fonction de distance entre deux vecteurs fournie par Unity.

L'affichage de la distance se compose de 4 état :

— Ball is too far : le robot est loin de la balle
— Getting close : le robot se rapproche mais ne peut pas tirer
— Kick the ball : le joueur peut tirer
— Wait for power : le joueur est assez proche mais la kappa n'as pas assez de puissance pour tirer.

L'affichage est effectué grâce à deux canvas (un pour chaque robot) contenant tous les deux zones de textes, l'un pour la distance et l'autre pour la puissance. Les deux canvas sont changés dynamiquement depuis le script GETKICKED à chaque Update de la frame.

4.2.3 But

Le but est géré par un autre script GOAL qui est également affecté à la balle. J'ai créé deux cubes totalement transparents et non solides, chacun représentant la surface d'un but.

On verifie si la balle est en contact avec un des deux cubes. Dans ce cas là, on incrémente la variable de score approprié et on replace tous les éléments du jeu à leur position initiale, les vitesses sont également remises à zéro.

Pour l'affichage du score, c'est exactement la même méthode que pour les informations des joueurs.

4.3 Autopilote

Contrairement à ce que j'espérais, l'équipe de NaMec n'utilise pas vraiment d'autopilote pour 1 seul robot. En effet, le jeu se joue à 6 contre 6, ils n'ont pas implémenté de comportement spécifique pour un seul joueur. J'ai donc tout recréer par moi-même, suivant les idées de déplacement qu'ils m'ont données.

Mon autopilote n'est pas parfait, s'il n'en existe pas dans l'équipe, c'est bien parce que c'est peu évident. J'ai fait le choix d'éviter à tout prix que le jeu se bloque si la balle est dans des endroits peu convénients. J'ai fait

en sorte que l'autopilote puisse jouer tout seul dans des situations difficiles, même si parfois son déplacement est lent.

4.3.1 Déplacement

Je ne pouvais plus me contenter d'ajouter des forces ou de la rotation. J'ai donc du re-implémenter tout le système de mouvement.

Pour les déplacements, j'ai utilisé un moyen simple, donnant comme vitesse la différence entre la position du robot et la position souhaitée. Le déplacement se fait ensuite tout seul car le robot bouge exactement vers la cible de façon autonome.

Pour la rotation, la méthode est à peu près similaire, ajoutant moi-même la rotation calculant l'angle signé représentant la différence entre la direction du robot et celle souhaitée.

J'utilise une fonction simple pour me positionner derrière la balle. Je calcule la droite donnée par ces deux points et place le robot dans l'alignement à une distance de 1 derrière la balle.

Enfin, je fais attention à ce que toutes les positions souhaitées du robot se situe bien sur le terrain.

Pour éviter les poteaux, j'ai utilisé une astuce très simple, m'inspirant de la version manuelle. En effet, j'applique une force permettant au robot de s'éloigner du poteaux (vers le centre du terrain) et vers son objectif (en x).

4.3.2 Contournement de la balle

Pour savoir s'il est nécessaire d'effectuer un contournement de la balle, je vérifie dès de la départ si la balle est situé derrière le robot par rapport aux cages. Ensuite, je calcule l'équation de droite entre la position actuelle du robot et celle souhaitée. Puis je vérifie si la droite passe par une position proche de celle de la balle en calculant une coordonées potentielle ayant le même z que la balle.

S'il est nécessaire d'éviter la balle, je définis une cible provisoire pour le robot, plus éloignée de la balle, décalée sur le côté. Cette position donne un placement avantageux pour le robot qui reprend ensuite un objectif "normal" qui lui permettra de tirer en se ré-alignant simplement sur l'axe balle-cages.

Chapitre 5

Extensions

5.1 Écartement de la balle

Dans mon jeu, malgré que les lignes du terrains soient tracées, celles-ci n'ont pas d'effet sur le jeu. Dans les futures extensions de la ligue, le comité prévoit de supprimer les lignes au profit de barrière (en s'inspirant du futsall par exemple). En effet, les robots de SSL se déplacent vite et tapent fort dans la balle, les trop nombreuses sorties de jeu sont souvent critiquées par les équipes. C'est une évolution qui semble donc naturelle.

En revanche, ces lignes que j'ai tout de même tracées ne sont pas inutiles. En effet, utiliser les bordures pour limites sont bien pratiques pour moi mais malgré des rebonds contre les parois, il est possible que la balle se coince contre les murs.

J'ai donc ajouté un script AVOID_WALLS permettant à la balle de s'écarter des murs et de revenir sur la ligne pour que le jeu reprenne. Si la balle se trouve à l'extérieur de la ligne, elle réapparait sur une ligne à la position la plus proche possible sur la ligne. J'ai mis un timer de 7 secondes derrière la ligne, il est donc facilement possible de mettre ce timer à un délai extrêmement réduit et cela peut simuler les règles de sorties.

5.2 Bruitage

Par curiosité, j'ai voulu ajouté des bruitages lors des tirs des robots. J'ai donc mis des sources audios sur le kicker de chaque robot pour emettre les sons. Ensuite, les AudioClips sont joués depuis le script GETKICKED. Lorsqu'un joueur tire (et vide sa kappa), la musique est jouée une fois.

Cela a été difficile de trouver une méthode qui marche mais au final la méthode est simple (il faut également configurer les sons sur les réglages).

5.3 Menu

Comme il me restait du temps j'ai décidé de faire un menu, la base pour un jeu.

J'ai pris beaucoup de temps à faire le layout, je n'arrivais pas à configurer le canvas. Une fois cela fait, ajouter les boutons a été très simple.

Ensuite, j'ai voulu pouvoir changer l'utilisation ou non de l'autopilote. Cela a été compliqué parce que je n'arrivais pas à accéder aux objets de l'autre scène et à activer/désactiver le script sur les robots.

J'ai réussi à changer de façon globale le booléen d'autopilote pour le script GetKicked. Pour résoudre mes problèmes d'activation du script d'autopilote j'ai donc assez facilement rajouter un booléen dans le script (dont j'avais l'accès depuis le menu) qui récupérait la valeur du booléen de GetKicked et activer ou non l'autopilote en fonction de ça.

J'ai ensuite ajouté un DropDown pour choisir la couleur que le joueur voulait jouer pour avoir le choix.

Chapitre 6

How to use it

Le robot rouge (Player 2) est sur l'écran de gauche et le robot bleu (Robot) est à droite pour correspondre aux touches de clavier.

Le robot 1 se joue sur le pad numérique.

Touche Robot 1	Touche Robot 2	Commande
8	Z	avant
5	S	arrière
4	Q	gauche
6	D	droite
7	A	rotation vers la gauche
9	E	rotation vers la droite
2	X	arrêt immédiat
Entrée (du pad)	Barre d'espace	Tir

— MOVE, ROTATE pour changer les touches de déplacement du robot bleu.

— MOVE_PLAYER2, ROTATE_PLAYER2 pour changer les touches de déplacement du robot rouge.

Pour activer/désactiver les autopilotes sans passer par le menu de jeu, il suffit de cocher la case Autopilote correspondante dans le script GETKICKED (contenu dans la balle). Pour rappel le Robot est bleu et le Player 2 est rouge.