UNIVERSITÉ DE FRANCHE-COMTÉ

Membre de

UBFC UNIVERSITÉ BOURGOGNE FRANCHE-COMTÉ

Jeu de Football en 2D

ROLLET Tommy & VIPREY Pierre

Encadrant : BERNARD Julien

Université de Franche-Comté

Licence 3 Informatique

Année scolaire 2021 – 2022

Table des matières

Remerciements

Nous tenons à remercier sincèrement Monsieur J. BERNARD, qui, en tant que professeur encadrant, s'est toujours montré à l'écoute et très disponible tout au long de la réalisation de notre projet. Ainsi nous le remercions pour l'aide et tout le temps qu'il a bien voulu nous consacrer et sans qui le rapport et l'application n'auraient jamais vu le jour.

Nos remerciements s'étendent également à tous nos professeurs de l'université de Franche-Comté, lesquels nous ont tous donnés des enseignements intéressants et essentiels à notre culture informatique, voire même générale, ainsi qu'a nos techniques de développements.

Plus particulièrement, parmi nos enseignants et hormis Monsieur J. BERNARD déjà cité, nous remercions Monsieur A. HUGEAT qui nous a beaucoup appris et guidé sur les bonnes pratiques et les particularités du langage C++ lors de nos séances de travaux pratiques.

Introduction

Notre sujet porte sur l'implémentation d'un jeu de Football en deux dimensions avec la réflexion et la programmation d'intelligences artificielles (IA) de positionnement ainsi que d'un moteur physique simplifié à l'extrême. Nous avons choisis de faire plus particulièrement un jeu de Foot Salle avec deux équipes de cinq personnages.

Nous avons préféré le Foot Salle[1] à la place du Football, pour deux raisons :

- La première étant de changer les habitudes : lorsque que vous pensez à un jeu de foot, vous pensez au Football classique : un terrain vert avec des lignes blanches, des corners, des touches… Peu de personnes pensent au Foot Salle.
- La seconde étant de permettre à un jeu de foot en deux dimensions d'être plus dynamique, le ballon rebondira sur les murs, cela veut dire qu'il ne peut pas y avoir ni de touche, ni de corner. Le seul moment ou le ballon s'arrêtera, hormis le début de match, c'est lorsque qu'un but aura été marqué.

Une équipe se compose d'attaquant(s), de défenseur(s) et d'un gardien, il y a forcément un personnage contrôlé par le joueur attribué à cette équipe, les autres personnages sont contrôlés par des IA de positionnement. C'est à dire qu'une IA ne peut pas tiré dans le ballon, puisque le personnage possédant celui-ci sera forcément contrôlé par l'un des joueurs.

Lorsque l'une des deux équipes marque, le jeu s'arrête temporairement pendant un très cours instant, absolument tous les personnages sont remis à leurs position d'origine et le ballon revient lui aussi à son emplacement originel : le milieu du terrain.

Voici un petit historique des jeux de football :

- 1973 : Soccer par Tomohiro Nishikado, premier jeu vidéo de football, cependant il restera trop similaire au jeu : Pong.
- 1980 : NASL Soccer, le vrai premier jeu vidéo de football, deux équipes de six joueurs, avec vue sur le coté.
- 1985 : Tehkan World Cup, un jeu vidéo d'arcade, celui-ci affiche une mini-map (carte réduite) avec la positions des vingts-deux joueurs.
- 1988 : Kick Off par Dino Dini, le match est représenté en vue du dessus avec la physique de la balle travaillée pour que le ballon ne soit plus collant afin qu'il soit possible de frapper celui-ci dans la direction et avec la force choisie.
- 1993 : FIFA International Soccer, plus communément appelé FIFA, est le premier jeu de football d'EA Sports et probablement l'un des deux plus développés et commercialisé de nos jours avec PES.
- 1995 : Pro Evolution Soccer, plus communément appelé PES, de Konami.

C'est dans le cadre du projet semestriel de la Licence 3 Informatique que ce projet nous à été attribué.

1 Aussi appelé Futsal

Analyse

Comme indiqué dans le sujet, notre projet sera développé en C++ à l'aide de la bibliothèque GF[2] qui nous permettra de gagner beaucoup de temps de développement car celle-ci est spécialisé dans la création de jeu en 2D.

Règles du jeu

Le jeu se joue en un joueur contre un joueur pendant cinq minutes. Chaque équipe est constituée de cinq footballeurs avec obligatoirement un seul gardien par équipe et au moins un défenseur et un attaquant.

L'équipe qui marque le plus de but dans le temps imparti remporte le match. En cas d'égalité aucun joueur n'est désigné vainqueur.

Un but est marqué lorsque le ballon rentre en contact avec la ligne des cages.

Au premier engagement le ballon est au centre du terrain et les footballeurs sont positionnés à leur emplacement initial, chaque joueur possède le contrôle du footballeur le plus proche du ballon. Après un but c'est l'équipe qui a concédé le but qui engage avec le 1er attaquant positionné juste à côté du ballon.

Le ballon et les footballeurs ne peuvent pas sortir du terrain et ils ne peuvent pas se superposer.

Le joueur de gauche joue avec les touches ZQSD pour le déplacement, il marche avec la touche Shift enfoncée et change de footballeur avec Tab. Le joueur de droite lui joue avec les flèches directionnelles pour se déplacer, il marche avec la touche Alt droit enfoncée et change de joueur avec Espace.

À tout moment le joueur peut changer de footballeur à contrôler, le jeu lui proposera le footballer le plus proche du ballon, il se peut que le footballeur le plus proche soit celui déjà contrôlé par le joueur, dans ce cas aucun changement n'est effectué.

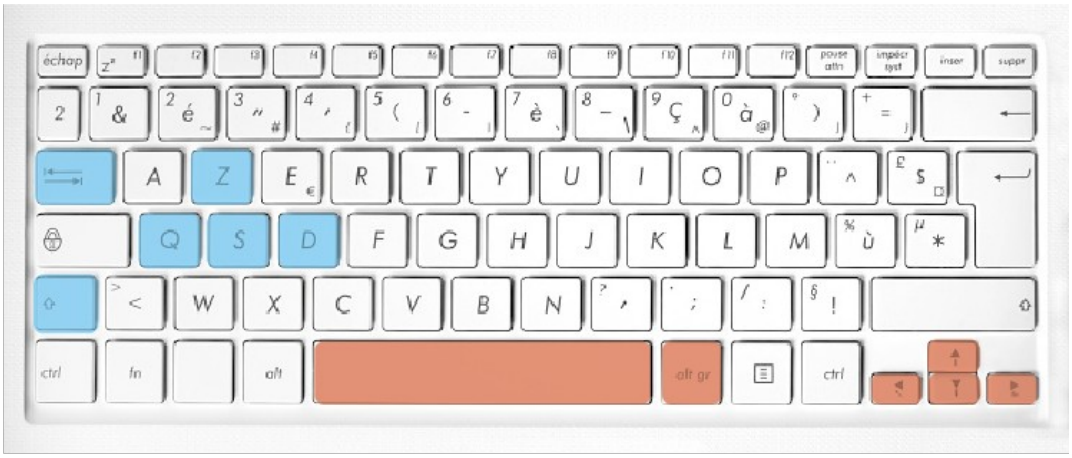

Figure 1: Touches utilisées pour jouer

2 Gamedev Framework

Modélisation

Notre projet mêlant dix footballeurs, un ballon et un terrain il nous a fallu dès le début bien délimiter ces classes pour nous permettre de travailler le plus sainement possible tout au long du de l'implémentation. Nous avons donc développé nous même les classes suivantes :

Game

La classe mère de notre architecture, c'est elle qui gère l'initiation de toutes les classes ainsi que les éléments extérieurs au Gameplay ou à la physique comme l'affichage de la fenêtre du jeu ainsi que sa gestion, l'affichage du score pendant la partie ou le chargement de la bibliothèque de textures.Terrain

Le terrain est constitué de 377 tuiles soit 13 de hauteur et 17 de longueur. Chacune de ces tuiles sont des Entités[3], il n'est pas nécessaire de stocker la texture du terrain en entier, nous avons simplement besoin de stocker les textures qui le compose. Le terrain est entouré de Rectangle[4] pour simuler les murs autour et également les deux cages de chaque côté.

Data

Cette classe très succincte permet à la classe Terrain d'interpréter notre tableau de caractères (voir Figure 4) symbolisant le terrain pour le placement des tuiles.

Equipe

La classe structure les actions de chacune des équipes a et des footballeurs d'une même équipe, c'est elle qui va créer le joueur et s'assurer que les règles sont bien spécifiées.

Joueur

Cette classe a pour objectif de définir un joueur comme une Entité avec son poste ainsi que son style de jeu et de le modéliser sur le terrain. Cette classe va permettre de déplacer les footballeurs pour le joueur mais également pour l'IA.

Ballon

La classe créé l'Entité du ballon, lui assigne une texture et modélise celui-ci sur le terrain.

Physics

Pièce angulaire du projet, cette classe va gérer les collisions entre tous les éléments du jeu. Elle gère les collisions entre joueurs, avec le ballon et les bords du terrain et indique lorsque qu'un but a été marqué.

3 gf::Entity
4 gf::RectF

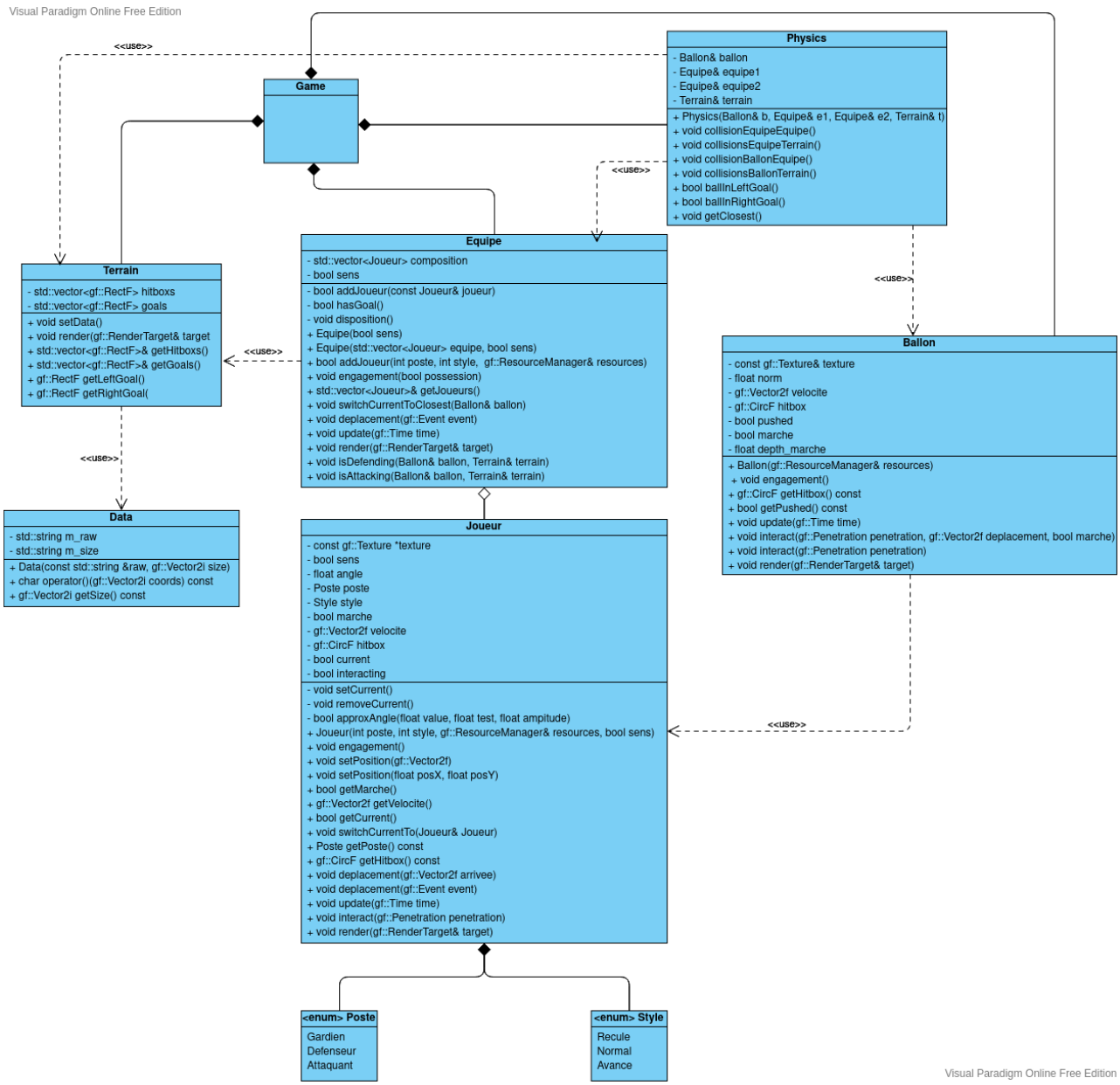

Figure 2: Diagramme UML du projet

Une chose qui nous a marqué pendant le développement du jeu et qui apparaît bien dans le diagramme (voir Figure 2) est la très forte dépendance des classes entre elles et ce malgré une architecture sobre et bien construite.

Intelligence Artificielle

Pour accompagner le joueur dans sa partie il était prévu l'implémentation d'une IA capable de proposer des solutions simples de positionnement permettant au joueur de faire des passes à ses coéquipiers en attaque et de couper les lignes de passes ou de bloquer le porteur du ballon en défense.

Un footballeur choisirait sa tactique en fonction de si l'équipe possède ou non le ballon, son Poste ainsi que son Style.

Poste

Le Gardien possède un style de jeu très simple, celui-ci doit en défense suivre les mouvements du ballon pour pouvoir se positionner entre ses cages et le ballon. En attaque il restera proche de ses cages en se plaçant dans sa position initiale.

Le Défenseur possède un rôle très important, en défense il doit empêcher la progression du ballon vers ses cages et également aider le joueur à progresser dans le camp adversaire en attaque.

L'Attaquant en défense doit se tenir prêt à réceptionner le ballon et donc se positionner plutôt haut sur le terrain. En attaque il essayera également d'aider le joueur mais sera positionné encore plus haut que les défenseurs.

Style

Enfin à propos des styles, un footballeur avec un style 'Avancé' aura tendance à jouer plus pour l'attaque que la normale ainsi que de maintenir une position avancée sur le terrain pour soutenir le joueur en attaque. À l'inverse un joueur avec un style 'Reculé' restera plus en arrière que la normale afin de garantir la défense.

Vitesse

À cela s'ajoute si l'IA souhaite courir ou marcher. Dans le premier cas il est capable de parcourir une plus grande distance mais il prend le risque de perdre le ballon et ne pourra changer la trajectoire du ballon que lorsqu'il sera à son contact. Dans le second cas le joueur est plus précautionneux, gardant le ballon proche de lui et facilitant sa navigation au détriment de sa vitesse.

Implémentation

Boucle de Jeu

La boucle de jeu est l'élément primordial à tout jeu. C'est elle qui fait avancer et vivre le jeu, elle doit être conçu pour pouvoir tourner à l'infini en recevant ou non des données des utilisateurs et de mettre à jour des comportements puis de les afficher.

Pour donner cette impression de fluidité au joueur il est primordial que la boucle se réalise le plus de fois possible, le minimum confort sur les jeux est généralement placé à 60 FPS[5], cela signifie que pour atteindre ce confort la boucle de jeu doit réaliser 60 tours dans une seconde, cela peut paraître beaucoup mais dans notre projet cette valeur est bloquée à 60 FPS.

Il existe trois grandes parties dans une boucle de jeu.

Réception des actions du joueur

La réception consiste à enregistrer toutes les actions du joueur, dans notre cas le jeu ne se joue qu'avec un clavier, nous n'avons donc que à nous préoccuper des appuis sur des touches mais également des retours de touche[6]. Cette partie sert à nous indiquer que le joueur souhaite déplacer les footballeurs, marcher ou courir, changer de footballeurs ou quitter le jeu.

Les mises à jour

Dans cette partie nous allons réévaluer toutes les données et ce que cela peut impacter depuis le précédent tour de boucle. C'est à cet instant que l'on intègre les actions du joueur, si le joueur voulait aller vers le haut on dira au footballeur de se déplacer dans cette direction. L'IA « joue » également à cet instant, si elle estime devoir se déplacer elle fera la même chose que le joueur, à la différence qu'elle n'a pas besoin d'appuyer sur une touche. Les règles du jeu sont également vérifiées à cet instant, comme les collisions ou le marquage d'un but.

Dessiner les entités

Nous avons calculer les modifications, encore faut-il les répercuter à l'écran. C'est à cet instant que nous le faisons, les footballeurs, le ballon et le terrain seront redessinés sur l'écran pour permettre au joueur de visualiser ses actions et celles du jeu.

Et on réitère cette boucle d'actions (Réception/ Mise à jour/ Redessiner) jusqu'à la fin de la partie, dans notre cas jusqu'à l'extinction du jeu par un joueur ou après les cinq minutes.

5 Frame Per Second (Image Par Seconde)
6 i.e. lorsqu'une touche est relâchée

```
// Début de la boucle de jeu
while (window.isOpen()){

    // 1. reception des actions du joueur
    while (window.pollEvent(event)){   //reception action sur clavier
        //...
    }

    // 2. Mise à jour des données
    update();

    // 3. Afficher les entités
    render();
}
```

Figure 3: Exemple d'une structure de boucle de jeu

La physique du jeu

Pour pouvoir déplacer les footballeurs sans qu'ils ne se superposent, permettre aux footballeurs de déplacer le ballon collé à eux, ou encore pour pousser et délimiter les bords du terrain de foot salle il était nécessaire de mettre en place une physique dans le jeu.

Toutes les interactions entre les Entités sont gérées dans la classe Physics. La fonction prédominante pour permettre la gestion est gf::collides() de la bibliothèque GF, malgré cette fonction centrale très utile il nous est apparu rapidement qu'elle ne serait pas suffisante pour modéliser l'intégralité des collisions dans notre jeu.

Collisions Footballeur/Mur

Le cas le plus simple de toutes les collisions concerne celles entre un footballeur et un mur du terrain qui est un élément fixe. La fonction va initialiser par argument une variable pénétration contenant un vecteur pour la direction de la force et un float pour mesurer l'enfoncement. Il ne nous reste plus qu'à appliquer au joueur cette variable pour obtenir un footballeur qui avance le long du mur.

Collisions Footballeur/Footballeur

Dans le cas des collisions entre deux footballeurs d'équipe différentes le problème peut être plus complexe à gérer, une collision entre ces deux entités se réalisant dans la même tour de boucle avec notre technique il n'est pas possible d'obtenir un résultat cohérent, une entité a toujours la priorité de déplacement sur l'autre. Ce phénomène s'explique car notre système de test de collision ne gère qu'un cas à la fois et écrasera le résultat du second. Des techniques pour pallier à ce problème ont été cherché et il semblerait que stocker les variables à modifier avant d'appliquer les changements serait une solution mais ils n'ont pas pu être mis en place faute de temps.

Collisions Footballeur/Ballon

Le cas d'un footballeur déplaçant le ballon est aussi intéressant. Nos footballeurs et le ballon étant modélisés par des cercles[7] si nous n'utilisions que la variable de pénétration le ballon glisserait à côté d'un joueur au lieu d'être poussé en avant. Pour obtenir un résultat souhaité il était nécessaire de récupérer le vecteur de déplacement du joueur et si le footballeur courait ou marchait. Le vecteur nous permettait de connaître la direction dans laquelle devait aller le ballon tandis que la marche nous indique si le ballon doit resté collé ou doit être poussé.

Collisions Ballon/Mur

Le dernier cas intéressant à traiter est le rebond du ballon contre un mûr, actuellement celui-ci suit la même démarche que le joueur en glissant le long. À terme il aurait été souhaitable de vraiment le faire rebondir en utilisant une autre méthode propre à lui même.

Détection de but

Il est apparu que gérer le cas du ballon rentrant en contact avec les cages ressemblait fortement au calcul de pénétration. Le ballon et les cages étant deux objets possédants une hitbox[8] il était possible d'utiliser la fonction gf::collides() pour savoir quand un but a été marqué. Dans ce cas il n'était pas nécessaire de récupérer la variable pénétration.

Affichage du terrain

Le terrain est initialisé à l'aide d'un tableau de caractères (voir Figure 4), chaque caractère correspond à une tuile du terrain et chaque caractère unique à une texture. Pour récupérer les textures on utilise un spriteSheet[9], à l'aide d'un fichier XML (voir Figure 5) on délimite chaque texture du spriteSheet en fonction de ses coordonnées et de la taille de ses tuiles, enfin on lui donne un nom pour permettre au système de la retrouver plus tard.

L'étape suivante requière de reconnaître chacun des caractères et de l'interpréter comme une texture, une fois la texture chargée on la place sur la tuile à l'écran suivant la position du caractère dans son tableau.

Cette technique permet des chargements plus rapide, au lieu de charger l'image du terrain complet il est plus rapide de demander au système de construire le terrain en direct à l'aide des tuiles qui lui sont fournies.

De plus l'utilisation de spriteSheet pour le stockage de textures est plus avantageux que de posséder un fichier de chacune des textures, il permet de réduire l'usage de la mémoire et accélère les phases de chargement.

Enfin le spriteSheet permet une certaine polyvalence, si on dispose d'un autre fichier construit de la même façon mais avec un style graphique différent il est très facile à utiliser car nous disposons déjà du fichier XML du fichier précédent.

7 Gf::CircF
8 Terme pour désigner l'espace qu'occupe l'objet
9 Une image regroupant toutes les textures mises à disposition

```
const char TERRAIN[] =
    " O------------Y------------P "
    " |            '            ! "
    " G            '            D "
    " |            '            ! "
    " |            '            ! "
    "W|           1A2           !X"
    "#|    .      (.)      .    !="
    "V|           4a3           !C"
    " |            '            ! "
    " |            '            ! "
    " G            '            D "
    " |            '            ! "
    " L____________y____________M "
;
```

Figure 4: Notre terrain en caractère

```
1  <TextureAtlas imagePath="groundTarmac.png">
2
3      <SubTexture name="Ground.png"                    x="0" y="0" width="64" height="64"/>
4      <SubTexture name="TopLeft_LShape.png"            x="64" y="0" width="64" height="64"/>
5      <SubTexture name="Top_Line.png"                  x="128" y="0" width="64" height="64"/>
6      <SubTexture name="Top_TShape.png"                x="192" y="0" width="64" height="64"/>
```

Figure 5: extrait de notre fichier XML

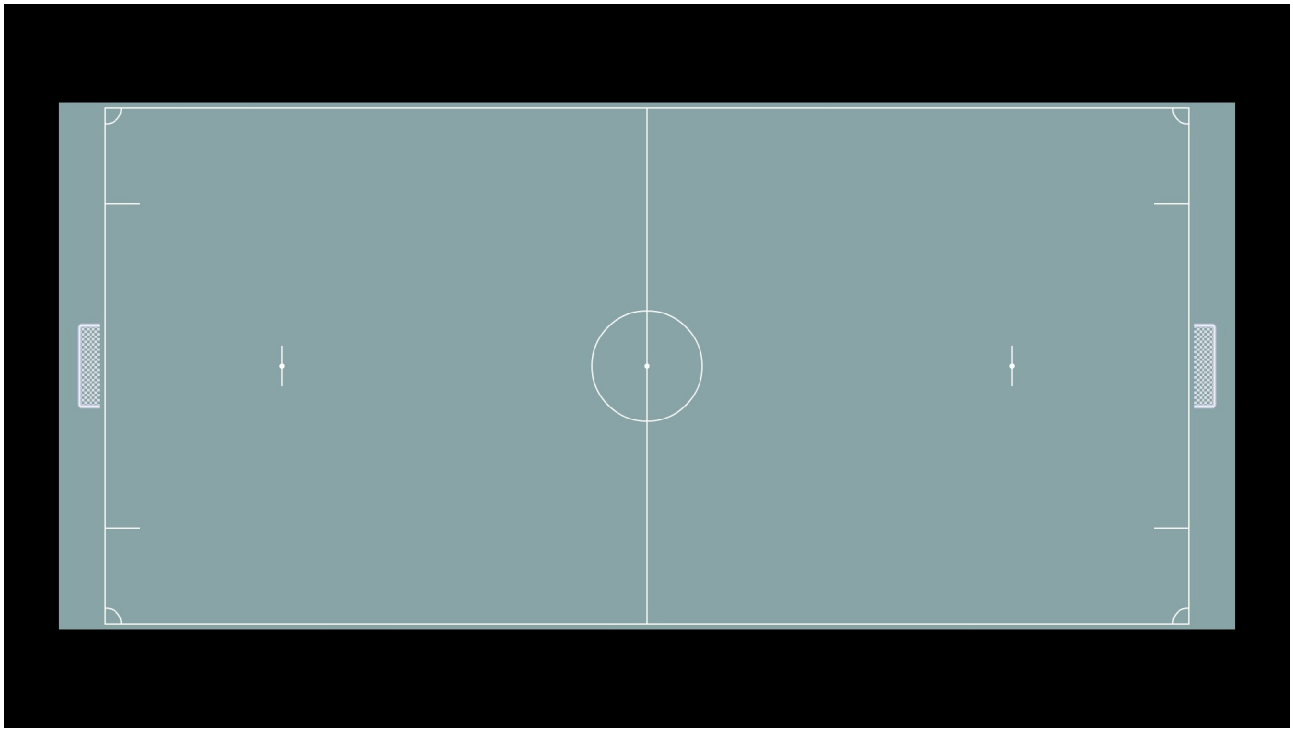

Figure 6: Rendu final du terrain

Organisation

Planning

Même si une première représentation sous forme de diagramme UML a été réalisée au début du projet, le développement n'a commencé qu'au début du mois de Février. Pour nous permettre de proposer le meilleur jeu possible un calendrier de développement a été mis en place.

Semaine 05 : Afficher le terrain

Semaine 06 : Afficher un joueur et le déplacer

Semaine 07 : Afficher deux joueurs et alterner entre les deux

Semaine 08 : Ballon (Affichage, Collision avec les joueurs & but), afficher les deux équipes

Semaine 09 : IA Joueurs

Semaine 10 : IA Joueurs

Semaine 11 : Rapport, Match & Score

Semaine 12 : Rapport & Préparation de la soutenance

Avec notre tuteur M. BERNARD Julien, nous avions convenu à une séance hebdomadaire pour discuter de l'avancement de notre projet.

Le planning n'a pas connu de retard avant la semaine 08 suite à des difficultés avec les collisions, le déplacement du ballon nous a demandé pas mal de temps pour arriver à un résultat satisfaisant et nous avons essayé de régler les problèmes de collisions entre deux footballeurs. Ce retard a entraîné un décalage du travail attendu sur le développement des IA et nous n'avons eu qu'une semaine pour nous attaquer à cette partie.

Git, GitHub & Méthode de travail

Très tôt dans le projet, a été mis en place un VCS[10] pour nous permettre de travailler au mieux et sans risquer de perdre notre travail. Notre choix s'est naturellement tourné vers Git et le service web de GitHub pour héberger notre travail.

Pour compenser une partie de notre retard nous avons opté pour travailler en Pair Programming[11]. Ce choix se motive par le fait que nous vivons tout les deux ensemble en colocation ce qui nous a permis de toujours pouvoir travailler à deux sur le projet. Cela nous a permis de réduire au maximum les pertes de temps sur les merge[12] et de gagner en efficacité de développement.

10 Version Control System (Logiciel de gestion de versions)
11 Méthode de travail dans laquelle deux développeurs travaillent sur le même poste en même temps
12 Terme pour désigner la fusion de deux travaux portant sur le même code

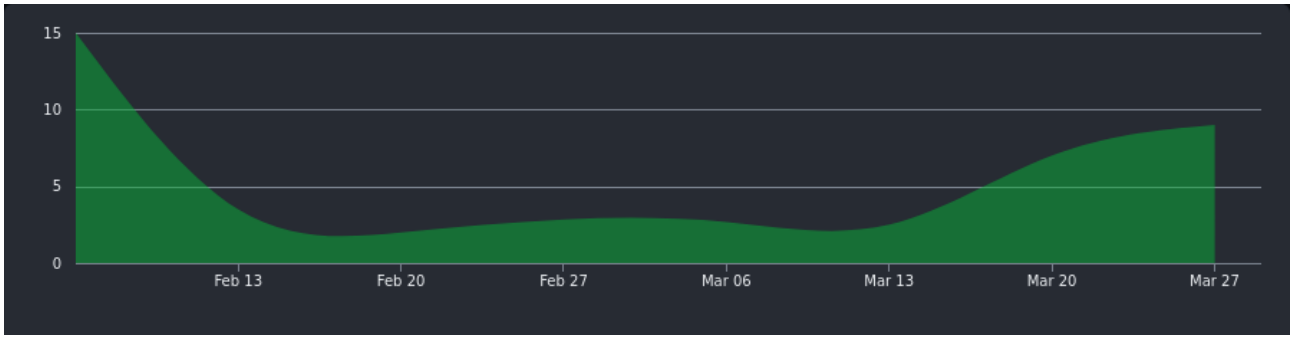

Figure 7: Nombre de commits pendant la période de projet

Conclusion

Finalement nous n’avons pas fini le développement du jeu, celui-ci manquant d’une IA, il est impossible de profiter pleinement du jeu. Certains phénomènes de collisions reste améliorable, tel que le rebond du ballon sur un mur ou les collisions entre deux footballeurs. L’affichage du score et du temps déjà écoulé dans la fenêtre de jeu aurait été un plus mais nous avons préféré continuer de nous concentrer sur le développement des IA. Enfin pour des raisons de confort il aurait été plus agréable de mettre en place une petite pause à chaque engagement pour permettre aux joueurs de se préparer. Dans son état actuel le jeu est proche d’un état de bac à sable. On peut s’essayer aux contrôles du jeu et il reste possible de marquer des buts.

Il est évident à posteriori qu’il aurait fallu commencer le développement du projet plus tôt que le mois de Février, nous pensons que début Janvier aurait été suffisant pour intégrer et développer toutes les fonctionnalités attendus du jeu.

Cette unité d’enseignement nous a permis de découvrir le fonctionnement et le développement d’un jeu vidéo avec sa boucle de jeu. Il est également le plus gros projet que nous avons eu à travailler depuis le début de notre formation, si nous n’avions pas suffisamment structurer notre code avec des classes il apparaît pour sur que nous nous aurions créé beaucoup de problèmes de développement et aurions perdu un temps précieux. Enfin ce projet nous a permis de forger et de consolider nos acquis et nos compétences en C++.

Résumé

Football est un projet de jeu vidéo de futsal en deux dimensions. Vous jouez contre un autre joueur dans un match de foot d’une durée de cinq minutes. Dans notre jeu vous pourrez contrôler cinq footballeurs avec un but très simple, marquer plus de but que votre adversaire.

Mot-clé : C++, Gamedev Framework, Football, Futsal, Jeu 2D

Abstract

Football is a futsal video game project in two dimensions. You are playing against an other player in a five minutes match. In our game you will be able to play with five footballers with a simple goal, score more goals than your opponent.

Key words : C++, Gamedev Framework, Football, Futsal, 2D game