LIFPROJET

Animation procédurale et jeu vidéo sous Unity.

ENNAOURA Riyad p1702710 |
LERAY Nicolas p1809852 |
CUZIN Kévin p1909458 |

Encadrant : MEYER Alexandre

UNIVERSITE CLAUDE BERNARD LYON 1 Automne 2021-2022

TABLE DES MATIÈRES

Abstract

Pour ce projet, nous devions, dans un premier temps, créer une animation procédurale, les mouvements d'un personnage puis l'intégrer dans un jeu. Le personnage choisi est une araignée à 8 pattes. L'objectif final étant d'intégrer cette araignée à notre jeu pour voir les mouvements et les interactions de celle-ci avec la scène. Le projet a été réalisé avec le logiciel Unity et les scripts ont été codés en C# sous Visual Studio Code.

Outils de développement

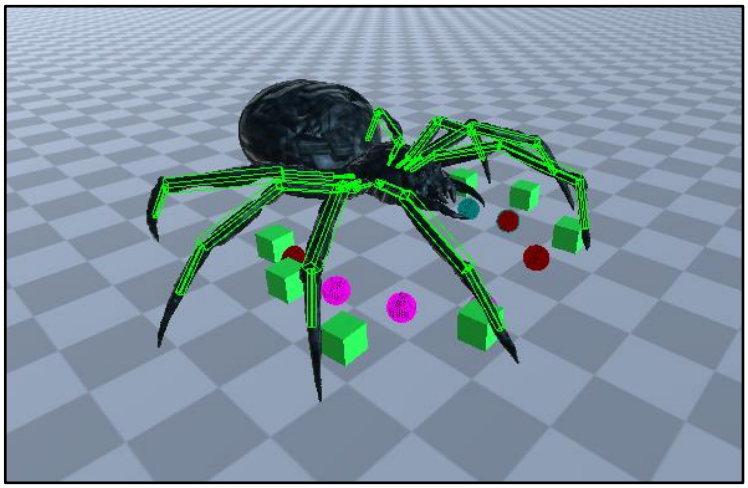
Unity est un moteur de jeu qui permet de créer des scènes mais aussi de pouvoir gérer des animations. Comme dit précédemment, nous avons utilisé Visual Studio Code qui est notre IDE de programmation courant. Également, Blender et Sculptrice nous ont permis de faire la modélisation et la mise en place d'un squelette pour notre personnage et notre araignée. Afin d'avoir un ensemble d'animations pour le personnage du jeu nous avons utilisé la banque de données d'Adobe Mixamo.

Vu d'ensemble du projet

L'araignée

Afin de faire notre araignée nous avons décidé de faire deux versions et de voir quelle était la version qui avait le meilleur rendu.

Au début du projet un package Unity (Animation Rigging) a été utilisé pour faire une première animation procédurale de notre araignée. Ce package nous permettait de lier les différents membres de notre araignée entre eux. On travaillait donc sur 3 membres d'une même patte, chaque bout de patte possédait un script pour pouvoir faire l'animation et donc pour gérer tous les membres de la patte relié par le script.



Après discussion par rapport à notre animation, nous avons convenu de changer de méthode pour laisser place à un peu plus d'algorithmes, donc nous avons codé l'algorithme de cinématique inverse, la Fabrik.

L'algorithme de cinématique inverse devait être mis sur chacune des pattes afin de faire bouger leur squelette. Nous avons fixé les pattes au sol qui est l'état initial de notre araignée, par la suite il a fallu vérifier la distance entre les pattes et le milieu du corps pour que les pattes puissent suivre le corps dans son déplacement. Pour cela, il a fallu faire un algorithme pour vérifier si la distance entre le corps et les pattes dépassait une certaine valeur. Les terrains pouvant avoir des aspects complexes il a donc fallu faire un

algorithme de Ray casting c'est-à-dire un script qui vérifie les rugosités de notre terrain.

Le terrain

Il a fallu réaliser un terrain sur lequel on pourrait faire évoluer notre araignée et pour cela nous avons utilisé des objets de différents aspects (sphérique, pentue, avec des marches…). Nous sommes partis sur un grand terrain que nous avons limité à une ville et pour finir un donjon. Pour le terrain de grande taille nous avons ajouté des « Landmark » c'est-à-dire des endroits qui vont attirer le regard du joueur.

Figure 2 Ville

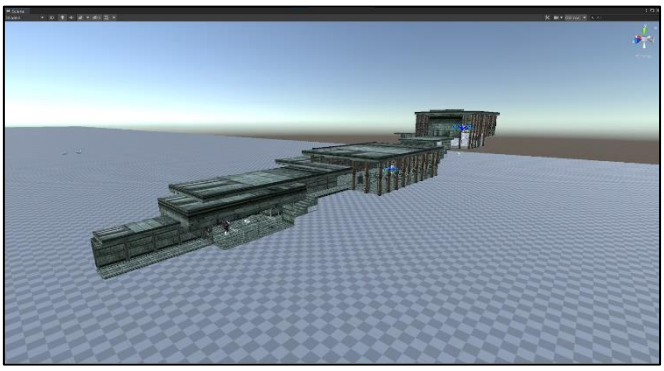


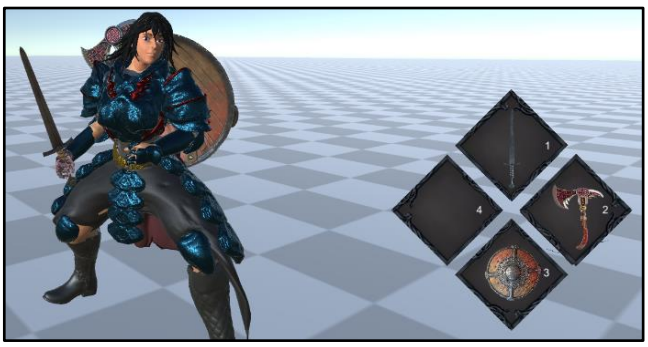
Figure 1 Donjon

L'interface graphique et les objets

Les interfaces de notre jeu sont gérées par des canevas qui sont des zones où l'utilisateur va pouvoir interagir à l'aide de sa souris. Plusieurs de ces interfaces sont implémentées, une pour mettre en pause le jeu avec une possibilité de revenir au menu principal ou de continuer à jouer, une pour l'inventaire où l'utilisateur va pouvoir cliquer sur les objets qui y sont présents et une pour le menu principal du jeu. Ces interfaces contiennent des boutons cliquables. Donc quand l'utilisateur va cliquer dessus le script relié au canevas va lancer la fonction liée au bouton.

De plus différents objets (Épée, hache, potion…) sont disposés sur le terrain. L'utilisateur à la possibilité de les ramasser, cela va les ajouter à l'inventaire du joueur et il pourra par la suite les utiliser en ouvrant l'inventaire et en cliquant sur les objets.

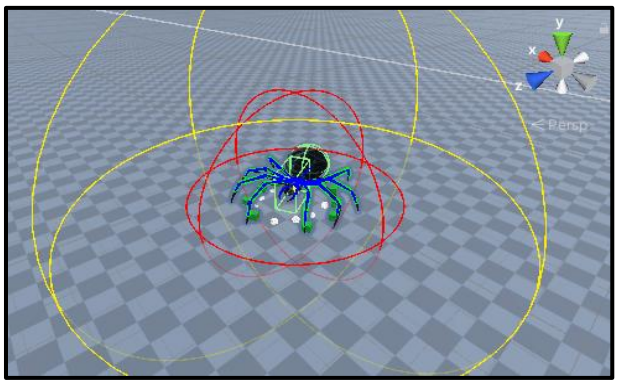
Les armes du terrain peuvent être équipées par le personnage, si les armes sont présentes dans l'inventaire du joueur elles seront visibles sur le joueur.



L'IA

L'IA est utilisée par les ennemies araignées de notre Jeu. Nous procédons à la mise en place des étapes suivantes :

Tout d'abord nous devons créer un « NavigationMesh ». Il s'agit d'un objet qui va permettre de quadriller notre terrain en définissant des zones accessibles ou non par notre agent et de sauvegarder ces données. Ensuite, nous mettons un « Nav Mesh Agent » qui va simuler notre ennemie sur le terrain, certains paramètres comme la vitesse peuvent être modifiés pour chacun de nos ennemis, cela nous permet d'ajouter du dynamisme à la scène.



Enfin, notre script « EnemyAI » va récupérer l'agent afin de savoir où nos araignées peuvent se déplacer et va aussi récupérer la position de notre joueur. Possédant une zone de détection (cercle jaune ci-dessus), l'araignée commencera à poursuivre le joueur s'il rentre dans sa zone de détection et elle s'arrêtera si le joueur en sort. L'objectif de l'ennemi sera alors de se rapprocher du joueur jusqu'à ce que celui-ci entre dans sa zone d'attaque (cercle rouge) et à partir de ce moment, un projectile est lancé en direction du joueur et lui inflige des dégâts. Cependant si l'ennemi n'a plus assez de « mana », il n'y aura plus de tir de projectiles

Animation et contrôle du joueur

Pour faire le joueur nous avons créé le script « TristaScriptMoveBasic » qui va permettre de contrôler le personnage et de le faire se déplacer. Ce script va faire appel aux différentes animations de Mixamos que nous avons calibrées avec notre modèle. Les animations sont gérées via un arbre d'animation qui sert à relier ces dernières. On a appliqué différents paramètres aux liens entre nos animations. Pour que les animations de marche et de course soient plus fluides pour le joueur nous avons rajouté un arbre

d’animation avec mixage qui permet de combiner plusieurs animations. Par exemple pour passer de la marche à la course nous avons utilisé un entier qui en fonction de la vitesse du joueur dans le script va faire une transition de la marche à la course. Nous avons également ajouté une animation de sauts, d’attaques légères, fortes et de magie

Vie et niveau

Le joueur dispose d’une barre de vie (graphique) qui est reliée à la vie du personnage. A chaque fois qu’il subira des dégâts, sa vie diminue et elle peut descendre jusqu'à zéro, et passer ce cap, il disparaîtra de l’écran pour réapparaître quelques secondes après.

Le personnage a aussi la capacité de prendre plusieurs niveaux. A chaque ennemi tué, il gagne des points d'expérience et à un certain nombre de points il passera au niveau supérieur. Cela lui rajoute des points de vie et des dommages à chaque niveau gagné.

Check point

Des points de réapparitions sont présents sur le terrain afin que le joueur puisse y apparaître en cas de défaite face à un ennemi.

Communauté Unity

Contribution à la communauté

À la suite des nombreux bugs survenus pendant l’utilisation de Unity, un report de bug a été opéré car le logiciel se fermait s’en raison à cause d’un problème dont nous n’avons pas connaissance.

Documentation

La communauté Unity étant vaste nous avons trouvé beaucoup d’informations sur internet. Par exemple, beaucoup de forums nous ont permis de trouver des solutions à nos problèmes. Également, Unity dispose d’un manuel complet regroupant toutes les explications des fonctions proposées par l’API de Unity. Ce manuel nous a permis de comprendre le comportement de certaines fonctions et donc de bien les appliquer dans notre projet.

API Unity

Unity dispose d’une interface de programmation d’application (API) très complète et nous permet d’agir sur la majorité des propriétés de nos objets sur une scène. Nous avons la possibilité de désactiver la collision de notre objet en jeu par exemple dans le cas de notre épée, si et seulement si une collision entre nos ennemis et l’épée est détectée par la méthode « *OnCollisionEnter* » fournit par l’API de Unity. Nous avons également pu exploiter les caractéristiques du moteur physique de ce dernier. Afin de garder un œil sur les performances nous avons utilisé les différents systèmes de gestion des FPS et du nombre de polygones.

Bilan

Mauvais points

Certaines erreurs nous donnent peu d'informations sur la nature du problème, telle qu'une texture qui dit être supprimée alors que nous la voyons toujours afficher sur nos objets sur la scène. Ce genre de bug empêche de lancer le jeu. Pour vérifier alors si c'est vraiment le cas nous devons revenir en arrière dans notre projet (cela peut aller de quelques heures de travail à quelques jours…). En allant sur le forum de Unity, nous avons constaté que cela arrive à beaucoup de monde et nous espérons que dans quelque temps ce problème trouvera une explication. A cela s'ajoutent les crashs qui ne sont certes pas courants mais qui peuvent ruiner des heures de travail, nous forçant à réaliser de nombreuses sauvegardes car Unity ne possède pas de système de sauvegarde automatique. Nous n'avons pu utiliser git car le fichier « Asset » contient la majorité des textures, d'autres scripts et peut très rapidement dépasser les 5 gigas. Il y a également un nombre important d'éléments dans les dossiers implémentés par Unity qui ne sont pas utiles. Nous avons également fait la connaissance des problèmes de corruption de fichier et de back up nous faisant perdre également des heures de travail.

Bons points

Sur les 3 mois de projet, 1 mois a été consacré à l'implémentation de l'animation procédurale des mouvements de notre araignée, nous trouvons ce temps raisonnable par rapport à l'objectif que nous nous étions fixés car il nous restait un peu plus de 2 mois pour faire le jeu et ses diverses implémentations.

De plus, nous avons pu implémenter la majorité des fonctionnalités que nous voulions. Et ce projet nous a permis de découvrir de nouveaux outils ce qui sera un plus sur notre CV pour, par exemple, trouver un stage.

De ce qui est de l'équipe enseignante, notre référent était là toutes les semaines ce qui nous a permis d'échanger régulièrement avec lui à propos de notre projet et des différents problèmes rencontrés. Il a pu nous donner des conseils pour réaliser au mieux notre projet, nous rediriger vers des forums et également vers des tutoriels pour comprendre les bases de Unity au début du projet.

Pistes d'améliorations

Lors du brainstorming nous avons implémenté les éléments qui nous paraissaient essentiels mais également intéressants pour la démo. Nous voulons également ajouter d'autres fonctionnalités que nous avons trouvé lors du brainstorming :

Animation

- Tirer à l'arc ;
- Combat à deux mains ;
- Lancer un objet ;
- Des animations procédurales sur des ennemis bipèdes et insectoïdes avec un nombre de pattes varié ;
- Des animations de marche en fonction des dégâts subis (marche plus vite avec 100% de sa barre de vie et au contraire avec 20% se barre de vie on a une marche plus lente) ;
- Parcours des objets (Monter une échelle, escalader un rebord…).

Physique du jeu

- De la destruction procédurale d'objets, éclats d'objet lors de la destruction.

Environnement / Graphique User interface

- L'intégration de notre terrain complet (voir Annexe « ImageTerrain ») ;
- L'ajout de nouveaux ennemis.

Comportement

- Un système sur la caméra pour qu'elle verrouille un ennemi ;
- Lorsqu'un ennemi est tué il lâche des objets au sol ;
- Un système d'esquives et de parade des coups avec des animations intégrées ;
- Des dégâts de chutes ;
- L'intégration de dégâts magiques par le joueur (l'animation étant déjà mise en place dans le projet) ;
- Un mécanisme de lancer d'objets ;
- Un système de combat à deux mains ou avec un arc.

Performance

Nous avions créé un terrain mais ce dernier avait un LOD (niveau de détail) trop important. Nous allons le calibrer pour que le terrain, avec nos fonctionnalités, puisse s'ouvrir même sur un ordinateur sans carte graphique.

Avis du groupe

Nous sommes plutôt satisfaits des différentes implémentations faites dans le projet mais nous avons été plutôt déçus lors de la phase finale car malgré notre répartition du travail la mise en commun des implémentations a été complexe. Cependant nous avons réussi tout de même à implémenter l'ensemble de nos parties respectives. De plus, le projet ne va pas s'arrêter là car nous avons comme ambition de le continuer sur notre temps libre. Pour beaucoup d'entre nous le projet a été un plaisir car cela nous a permis d'apprendre à maîtriser un logiciel de développement de jeu qui est utilisé autant par des entreprises de jeux indépendantes que des grosses entreprises. A cela s'ajoute qu'Unity nous a permis de nous entraîner à la programmation en C#, la modélisation, l'infographie, le Level design, le texturing, l'implémentation d'IA et même la réalisation d'interfaces graphiques. Pour certains ce projet a fait naître l'envie de réaliser un stage dans des entreprises spécialisées dans ce secteur, mais également de développer d'autres types de jeux voire même d'en faire un métier à plein temps.