



# Rapport de soutenance

Soutenance du 4 mars 2021

---

Emre ULUSOY

Paul Hartmann

Guy-noé Djoufaing

Antoine Loeffler

Mohamed Elsonbaty

# Introduction

Depuis la création du groupe Paguet 18 en début d'année, nous avons cessé d'améliorer constamment notre jeu . La première soutenance arrive et notre bilan est plutôt positif , malgré les problèmes que nous avons rencontré nous pensons avoir respecté nos attentes et nos objectifs pour cette soutenance. A travers ce rapport de soutenance nous allons développer dans un premier temps les travaux effectués en groupe et individuellement de manière linéaire depuis la création de notre cahier des charges. Dans un second temps, il est évident de parler des problèmes que nous avons rencontré au niveau du projet mais également au sein du groupe.

À la création du cahier des charges nous avions plusieurs attentes et obligations pour la première soutenance. L'objectif étant de toucher un peu à tous les aspects du jeu pour voir si tout était faisable et surtout si on était sûr de réussir à le finir pour la dernière soutenance. Cependant nous avons quand même poser des obligations pour la première soutenance :

- Comprendre et réussir à utiliser Unity Collab, Scène Fusion et GIT (déjà vue à l'aide des TP)
- Trouver les bons assets, s'occuper de la création du terrain de la map et réussir l'assemblage
- Commencer à créer des interfaces et une minimap.
- S'occuper de tout ce qui touche au script de mouvement, le choix des animations.
- Le mode multijoueurs, le réseaux etc...
- Gérer les spawn des personnages.

# Unity

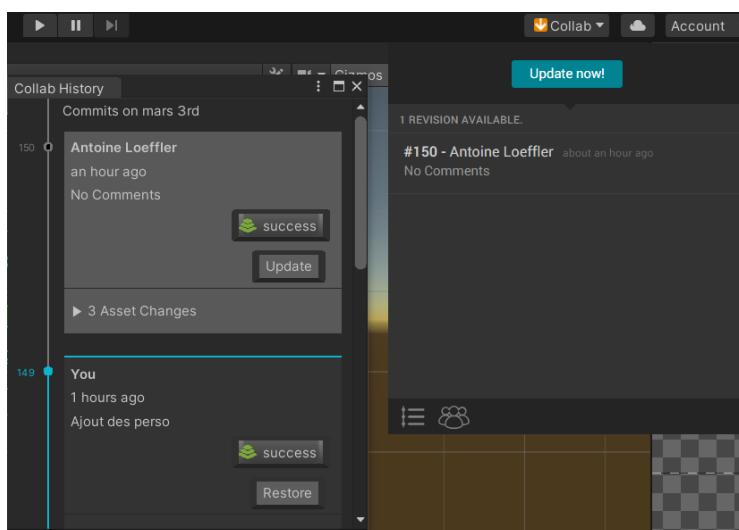
Comme indiqué dans notre cahier des charges , nous avons utilisé le moteur Unity pour le développement du jeu. Après avoir tester un peu ses fonctionnalités et comment il fonctionne à l'aide de cours en ligne (openclassroom) nous avons donc conclu que Unity aller correspondre parfaitement à la création de notre jeu.

## Système de collaboration

Etant donné le contexte actuel nous empêchant de faire des sessions de travail en présentiel, nous avons été contraint de trouver des outils nous permettant de continuer à avoir cet esprit collaboratif et évidemment ce sont des outils qui seront utiles si jamais nous devons passer à distance.

Après avoir vu les avantages et les inconvénients des deux options que nous avons citées dans le cahier des charges (Git, Unity Collab), nous avons décidé d'utiliser Unity Collab ce qui nous a permis de travailler en parallèle de façon très efficace. Cela nous permet de garder une grande liberté dans le projet, on a pu revenir sur des anciennes versions, faire des changements... et évidemment pour garder une sécurité au cas où il y aurait un crash et des bugs.

Image de l'interface de Unity Collaborate

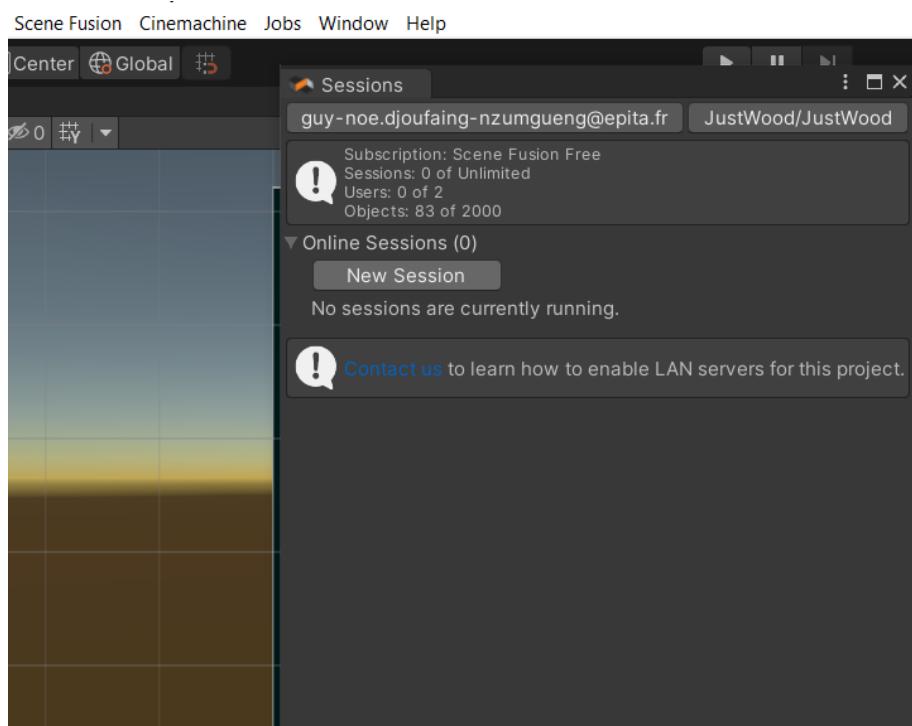


De plus, nous avons également utilisé le plugin Scène fusion, disponible pour Unity et Unreal. Cela nous a permis de travailler ensemble au même niveau, en temps réel. En effet ce plugin utilise le cloud pour partager automatiquement les modifications afin qu'il fonctionne pour les équipes locales et distantes. Il réplique tout le travail de l'environnement en temps réel, y compris le placement, la suppression et le déplacement d'objets, ainsi que la modification des propriétés. Cet asset s'est avéré très utile pour les travaux entre le responsable et son suppléant. En effet, nous observons que voir tous les changements en direct sans attendre le push des autres nous permet de travailler beaucoup plus efficacement et cet aspect visuel aide également. Enfin nous avons également utiliser l'outils GIT pour publier notre code et nos fichiers de préparations pour les rendu des soutenances

Évidemment nous avons rencontré de nombreux problèmes dans cette partie de projet, en effet on a rencontré de nombreux bugs et des déconnexions avec Unity Collab. Par exemple, on a eu des soucis de version, on n'avait pas toujours la bonne version ou la même version que les autres. De plus, lorsque deux personnes changeaient le même Game Object en même temps il y avait un conflit sur le collab, une des deux versions était systématiquement écrasée et on était obligé de revenir sur une ancienne version pour récupérer notre travail.

De plus, on a dû arrêter d'utiliser Scene Fusion car avec la version gratuite nous ne pouvions pas dépasser 1000 assets et c'est maximum deux utilisateurs en même temps. Pas pratique pour la mise en commun étant donné que nous sommes cinq.

Image de l'interface de Scene fusion



# Multijoueur

Pour la partie multi-joueur, nous avons décidé d'utiliser notre propre "Dedicated Server". L'interaction Homme-Machine avec le serveur a été grandement facilité avec l'outil open source [Pterodactyl](#) que nous avions précédemment utilisé pour des serveurs de jeu sur Minecraft. Nous avons réussi à implémenter le projet C# "Server" de Just Wood pour avoir un contrôle complet sur le serveur depuis internet.

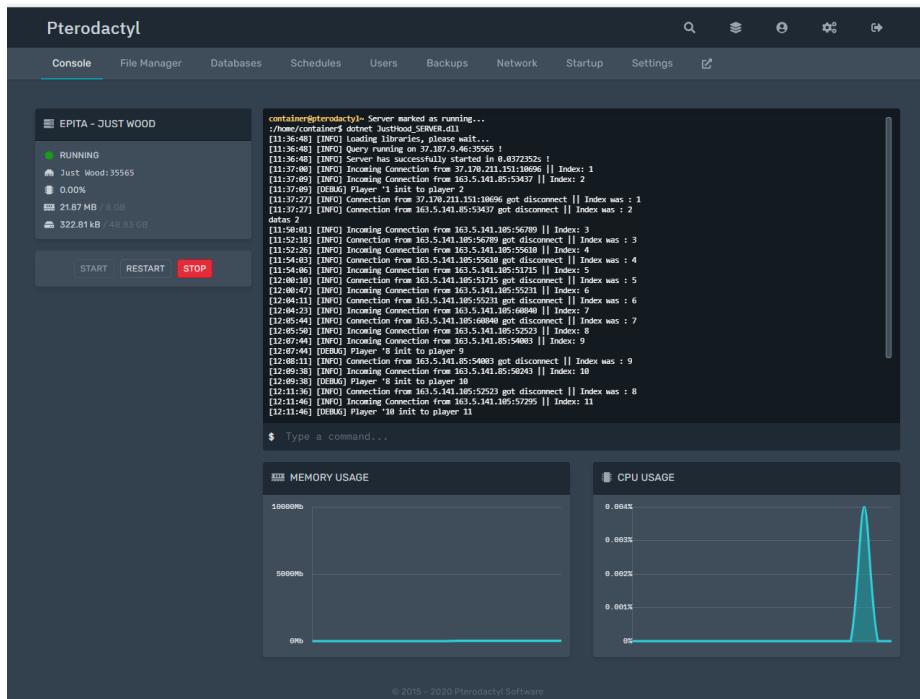


Figure 1 : Interface Pterodactyl

Pterodactyl nous offre ainsi la possibilité d'interagir avec la console et de voir l'utilisation notamment de la RAM et même du CPU.

## Quelques définitions

Pour la suite du rapport nous allons souvent utiliser plusieurs termes dont voici l'explication :

- “l’index d’un joueur” : l’index d’un joueur est la clé du joueur. En effet, si le joueur 4 envoie “bonjour” au serveur, sa clé est ici 4 et permet d’authentifier le client auprès du serveur et aussi localement auprès de tous les clients.
- “packet” : Un packet est un groupe d’information en octet envoyé du serveur au client et vice-versa.
- “client” : Un client est un objet ayant en attribut un Index, ses coordonnées XYZ et ainsi que ses informations concernant le réseau (Adresse IP etc...)
- “clé” : revient à l’index du joueur
- “GameObject Player”: un objet sur Unity représentant graphiquement un joueur ayant des attributs tel que son Index.

## Le principe de base

Le serveur ici est censé gérer l’attribution des “room publics” et les “room private” dès la connexion du joueur. Pour la première soutenance, nous avons fait une seule et unique “room” où les joueurs peuvent se connecter directement et où ils pourront se déplacer ensemble. En effet, le fonctionnement d’un client dès le lancement du jeu se résume à ces quelques étapes :

- Envoie d’une requête de connexion au serveur grâce à l’ip et ainsi du port du serveur.
- Le serveur, recevant la requête du client, instancie localement un nouveau Client avec une nouvelle clé et envoie un packet qui contient la clé du nouveau client avec ses coordonnées XYZ à tous les clients connectés sur le réseau (le nouveau client compris). Par défaut, les coordonnées sont (0,64,0) respectivement pour (X,Y,Z).
- Tous les clients recevant ce packet feront alors apparaître un nouveau GameObject Player dont l’index de ce GameObject est précisé dans le packet. Par défaut, l’index représentant le joueur qui joue sur Unity vaut 0 : si cet index vaut 0 alors le packet reçu est le groupe d’information le concernant. En d’autres termes il aura la vision et le contrôle de ce nouveau GameObject et redéfinira son index local au nouveau index.

- Et finalement, le serveur envoie un packet contenant les informations de tous les autres clients connecté à ce nouveau client pour que lui aussi les instancier et les créer localement (sans le compter).

## Et en pratique ?

En théorie, la mise en place du serveur et de ce système de packets est un jeu d'enfant... En pratique, c'est toute une autre histoire. Pour commencer, nous devons créer un "egg" pour pouvoir rendre compatible le serveur de jeu Just Wood avec l'outil Pterodactyl. ([→ Voir figure 2 ci-contre](#) pour l'egg de Just Wood)

```

1  {
2      "exported_at": "2021-03-03T15:42:57+01:00",
3      "name": "JustWood",
4      "author": "emre.ulusoy@epita.fr",
5      "description": "Une plateforme pour host des serveurs Just Wood",
6      "features": null,
7      "images": [
8          "quay.io\parkervcp\pterodactyl-images:debian_dotnet"
9      ],
10     "file_denylist": "",
11     "startup": "dotnet {{SERVER_DLLFILE}}",
12     "config": {
13         "files": "{}",
14         "startup": "{\r\n    \\"done\\": \"Server has successfully started in\r\n\", \r\n    \"logs\": \"{}\", \r\n    \"stop\": \"stop\"\r\n},\r\n        \"scripts\": {\r\n            \"installation\": {\r\n                \"script\": "#!/bin\\ash\\r\\n# JustWood Installation Script\\r\\n#",\r\n                \"container\": \"alpine:3.4\", \r\n                \"entrypoint\": \"ash\"\r\n            }\r\n        },\r\n        \"variables\": [\r\n            {\r\n                \"name\": \"Server DLL File\", \r\n                \"description\": \"The name of the Exefile to use when running For\\r\\nthe server. It must end with .dll\", \r\n                \"env_variable\": \"SERVER_DLLFILE\", \r\n                \"default_value\": \"JustHood_SERVER.dll\", \r\n                \"user_viewable\": true,\r\n                \"user_editable\": true,\r\n                \"rules\": \"required|regex:^([\\w\\d.-]+)(\\.dll)$\\r\\n\""
15            }
16        ]
17    }
18 }

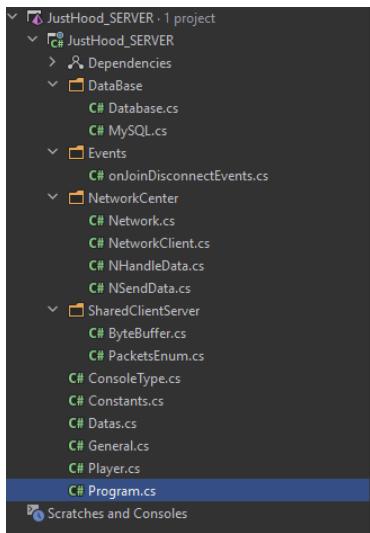
```

*Figure 2 : Code source de l'"egg" pour la mise en place du serveur Just Wood sous Pterodactyl*

Pour faire simple, un "egg" est un fichier json qui met en lien :

- le docker, qui lui permet d'exécuter les fichiers programmes du serveur dans un univers fermé et sécurisé
- le script d'installation (qui permet de télécharger les fichiers programmes)
- quelques variables pour avoir plus de facilités à écrire directement avec l'outil

Après des heures de recherches, nous avons finalement trouvé un docker capable d'exécuter correctement les fichiers .DLL (qui proviennent, eux, de la compilation des codes C# du serveur). Nous remercions au passage les développeurs de l'outil Pterodactyl car ils nous ont aidé à trouver le docker que nous utilisons.



## A. Le Thread

Nous pouvons enfin démarrer correctement le serveur.

L'architecture du serveur se présente sous cette forme.

→ [Voir figure 3 ci-contre](#)

Le fichier que nous utilisons pour lancer le serveur est le fichier Program.cs. Dans ce fichier, nous initialisons tout simplement un Thread qui fera office d'interpréteur Homme-Machine par l'outil Pterodactyl.

→ [Voir figure 4 ci-dessous](#)

Figure 3: Architecture du serveur

On commence d'abord à initialiser un Thread qui sera notre Thread principale où nous pouvons aussi agir directement depuis la console de Pterodactyl avec ce Thread représenté ci-dessous :

Figure 4 : Code source du Program.cs du Serveur

```

6
7
8
9
10
11  ►
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51

[5 usages]
class Program
{
    private static Thread threadConsole;
    private static bool consoleRunning;

    static void Main(string[] args)
    {

        DateTime now = DateTime.UtcNow;
        BroadcastConsole(ConsoleType.INFO, msg: "Loading libraries, please wait...");

        threadConsole = new Thread(ConsoleThread);
        threadConsole.Start();

        Datas.networkHandleData.InitMessages();
        Datas.general.InitServer();

        BroadcastConsole(ConsoleType.INFO, msg: "Server has successfully started in " + DateTime.UtcNow.Subtract(now));
    }

    [1 usage]
    private static void ConsoleThread()
    {
        string line;
        consoleRunning = true;

        while (consoleRunning)
        {
            line = Console.ReadLine();

            if (line != null)
            {
                if (line.Equals("stop"))
                {
                    consoleRunning = false;
                    return;
                }
                else if (line.StartsWith("datas"))
                {
                    string[] part = line.Split(separator: " ");
                    if (part.Length == 2)
                    {
                        Datas.networkSendData.SendAllDatas(index: Int16.Parse(part[1]));
                    }
                }
            }
        }
    }
}

```

## B.L'écoute du port

Comme dit précédemment, pour qu'un client se connecte, il doit envoyer une requête à l'IP de la machine avec le port où écoute le serveur Just Wood. Pour cela nous devons écouter un port spécifique et unique ("unique" car en toute logique le port reste le même du début à la fin). Nous avons alors décidé d'écouter le port 35565 concernant le serveur Just Wood. Pour se faire, au démarrage du serveur, nous initions un TcpListener sur le port 35565 où nous autorisons toutes les IP et où nous attribuons la toute première fonction qui intercepte les requêtes de connexion des nouveaux clients sur le réseau.

```
ServerSocket = new TcpListener(localaddr: IPAddress.Any, port: 35565);
ServerSocket.Start();
ServerSocket.BeginAcceptTcpClient(OnClientConnect, state: null);
```

Figure 5 : Code pour "écouter" un port

OnClientConnect est une fonction qui permet d'initialiser le joueur et crée la Clé spécifique du joueur et exécute la série d'envoie de packets cité ci-dessus.

## C. Packet

Un packet est un groupe d'information en byte envoyé du serveur au client et vice-versa. Concrètement, un packet est composé d'un ID de packet et de Datas.

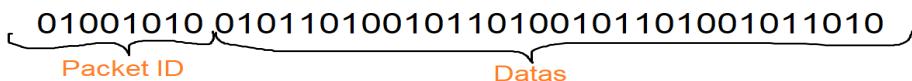


Figure 6 : Exemple du contenu d'un packet envoyé entre le serveur et un client

Donc le serveur et le client discutent en binaire : ils se partagent une table de Packets pour identifier quel est le thème de chaque packet. Par exemple, si le serveur envoie un packet de mouvement alors le Packet commencera par l'ID du packet de mouvement. Voici pour l'instant la table des packets → [Voir ci-dessous](#) :

```
public enum ServerPackets
{
    SAlertMsg = 1,
    SPlayerDatas = 2,
    SPlayerMovement = 3
}
```

```
public enum ClientPackets
{
    CHandleMovement = 3
}
```

# Choix et animations des personnages

Pour le choix des personnages, nous avons longuement hésité. Notre premier choix était, comme le prévoyait le cahier de charge, de choisir des personnages libres de droits et gratuits de l'asset store mais après plusieurs essais de personnages, nous nous sommes aperçu que ceux-ci ne correspondaient pas vraiment avec l'univers du jeu.

Premier choix de troll



Premier choix d'elfe



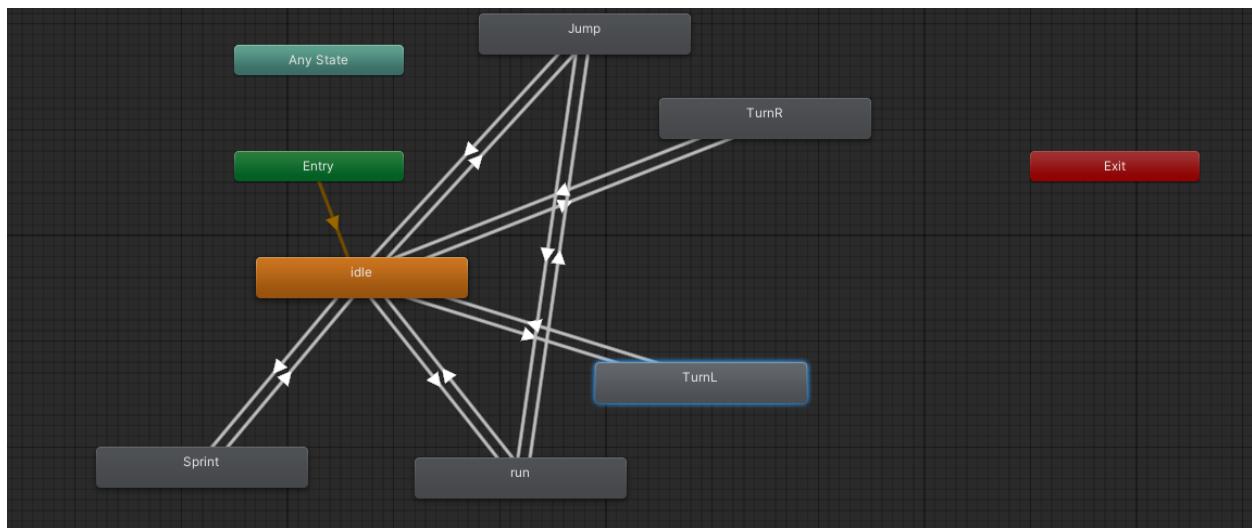
Nous avons donc cherché des personnages gratuits disponibles sur internet mais la encore sans véritable succès. Nous avons alors pensé à sélectionner des personnages du site mixamo. Les personnages choisis sur mixamo ont été pendant des semaines ceux que nous avons gardé mais après avoir commencé la création de la map, le rendu avec les personnages de mixamo n'était plus ce que l'on espérait. En effet, notre jeu est censé être en low poly mais mixamo ne propose aucun personnage dans ce style . Nous avons alors décidé de changer les personnages. Et après quelques recherches, nous avons trouvé un pack contenant différents personnages fantastiques, dont certains correspondaient à l'image que nous avions des personnages. Nous avons alors acheté ces personnages. Mais par manque de temps, nous n'avons pas encore remplacé les personnages dans la scène de jeu. Les

personnages qui sont donc présents dans le jeu au moment où j'écris ces lignes et qui seront présentés à la première soutenance seront donc changés dans les jours à venir.

Pour l'animation des personnages, nous avons globalement suivi ce que nous avions projeté de faire au travers du cahier de charge. Lorsque nous avons switché entre les différents personnages, nous avons alors eu deux options. soit le personnage n'était qu'un modèle 3D dans quel cas nous l'avons importé dans mixamo et nous lui avons assigné des animations, c'était notamment le cas pour les premiers personnages du troll et de l'elfe ou encore des personnages de pack que l'on a acheté, soit le personnage possédait déjà un système complet d'animation c'était notamment le cas pour la deuxième version du troll (voir image annexe).

Une fois le choix des animations fait, il fallait créer un système d'animations pour les différents personnages. Concrètement, nous avons mis en place un animateur qui déclenche une animation sous certaines conditions.

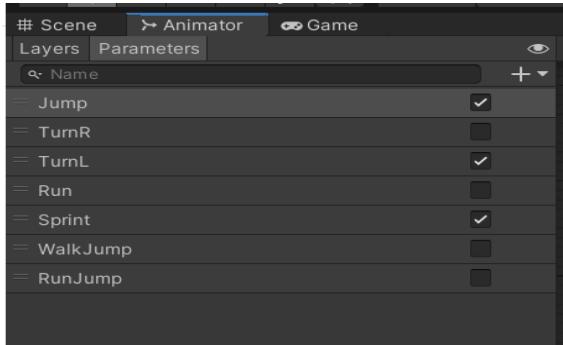
Animator d'un personnage



Par exemple sur cet animator d'un elfe, ce qui nous intéresse est :

- Le bloc vert représente le point d'entrée dans l'animateur au lancement de la partie et n'est joué qu'une fois.
- Les blocs gris et oranges sont les animations des personnages.
- Le fil blanc représente les transitions. Ils permettent de définir sous quel condition on change d'animation grâce à des variables.

#### Variable de transition de l'animator précédent



#### Gestion des variables via le script

```
//Idle
if (verticalPress == 0)
{
    anim.SetBool(name: "Run", value:false);
    anim.SetBool(name: "Sprint", value:false);
}
//course
if (verticalPress > 0)
    anim.SetBool(name: "Run", value:true);
```

Dans le processus de mise en place des animations, la difficulté venait du fait qu'aucun d'entre nous n'avait des connaissances sur le sujet et on a donc dû essayer de comprendre tout cela en partant de rien, et nous avons souvent bloqué pendant des heures sur des petits détails qui aujourd'hui semblent ridicules.

# Minimap

Après avoir mis en place notre map, on a voulu faciliter l'expérience de nos joueurs et les aider à s'orienter.

Les mini-maps sont devenues une norme parmi les **RTS** (Real Time Strategy), dans lesquels avoir une vision globale du jeu est nécessaire et c'est pour ça qu'on s'était senti obligé d'implémenter celle-ci dans le jeu Just-Wood.

Placée dans le coin bas droit de l'écran, elle affichera généralement les types de terrain, les alliés, les prochaines zones rouges, les lieux ou objets autour du joueur.

L'adaptation de celle-ci n'était pas très complexe puisqu'il suffisait de créer une caméra (vu d'en haut) de chaque déplacement du joueur et en faire la conception de nos choix dans un canvas qui allait servir de visuel pour le joueur.

Mais bien qu'elle paraisse simple, la mini-map n'est fonctionnelle qu'avec un code qui permettra de faire suivre la caméra (de la mini-map) au joueur et ses rotations. La difficulté s'imposait donc dans ce code qui présentait souvent des défauts suivant les déplacements du joueur.

# Spawns

Bien évidemment, les joueurs de just wood ont besoin d'apparaître pour jouer à Just Wood. Concernant cette partie, on savait que les elfes allaient apparaître dans le village (au milieu de la map), il fallait donc se focaliser et ajouter les localisations ou ajouter les trolls en début de partie. En effet, on comptait faire en sorte que les trolls apparaissent partout dans la map.

Pour faire cela, on a ajouté des “Spawn points” (GameObject vide) dans chaque coin de la map et on a écrit un script permettant d'ajouter les trolls à ces positions données.

Nous avons réfléchis à plusieurs problématiques:

- Comment peut-on faire un code qui pourra gérer plusieurs spawn points à la fois?
- Comment est-ce qu'on fera pour appliquer un même script à nos deux personnages?
- Comment faire un code qui va pouvoir gérer un personnage même si on lui apporte des modifications?

Nous avons donc réussi à coder une fonction qui prendra le Prefab indiqué du dossier “resources” et l'insérer dans une des positions: “Spawn Points” auxquels on a ajouté des tags plus tard choisis aléatoirement par le code source.

# Map

Pour la création de la map, nous nous étions donné des objectifs conséquents. Nous voulions faire une très grande surface de jeu pour dynamiser notre jeu et permettre aux joueurs de vivre cette expérience dans cinq décors distincts.

Comme prévu, nous avons mis en place 5 environnements différents: la zone été, la zone hiver, la zone désertique, la zone asiatique et la zone village.

## Description de la mise en place de la “Zone Village” par Guy-Noé

Lors de la conception du projet, l'idée était de faire un château, lorsque nous avons commencé à poser les premières pierres du projet, nous nous sommes rendu compte que la mise en place du château nous demanderait de repenser entièrement certains aspects du jeu. Étant donné que le château devait être la zone de fin de partie, et que l'on compte jusqu'à 24 joueurs en multijoueur, le château se devait d'être gigantesque pour pouvoir contenir tous ce monde. C'est alors que nous avons décidé d'opter pour un village, ce qui a entraîné quelques modifications dans le cahier de charge. Concernant la mise en place du village, J'ai fait le choix de ne prendre que des assets gratuits du unity store. Je n'ai pas eu de réelle difficulté dans la conception du village puisque contrairement aux environnements (désert, hiver...) le village n'a pas de reliefs et du coup j'ai aisément créé un terrain vide où j'ai pu arranger les assets. La difficulté en soi n'était donc pas de créer le terrain mais l'établissement et l'ajustement des assets que j'ai dû recommencer à plusieurs reprises, jusqu'à être bien heureux du rendu final.

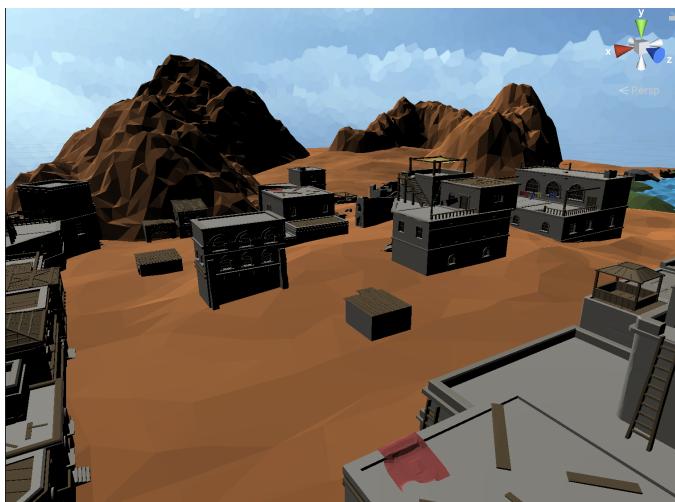
Image de la dernière version du village



## Description de la mise en place de la “Zone Désert” par Mohamed

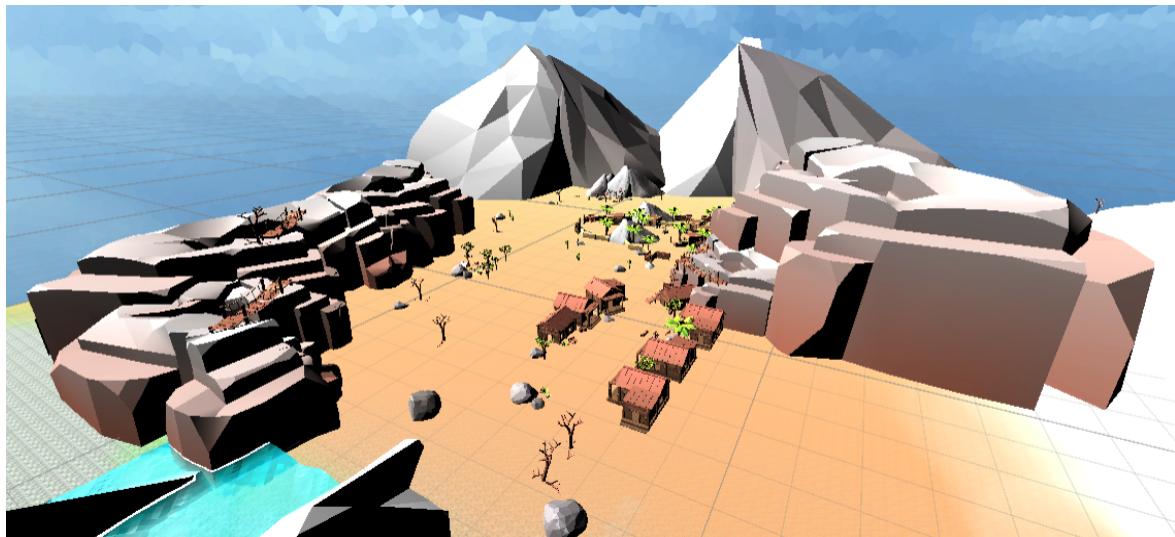
Concernant le désert, plusieurs problèmes ont été rencontrés qui m'ont obligé à tout reprendre à plusieurs reprises. Au début, j'étais parti sur une base de dunes que j'ai créé sur Blender avec des montagnes (que j'avais aussi créées sur Blender) et des assets d'un village médiéval abandonné.

Image de la précédente version du désert



Cependant, j'ai vite remarqué que mon terrain ne correspondait pas au terrain de jeu qu'on voulait mettre en place pour plusieurs raisons: j'étais le seul qui a décidé de créer sur Blender et du coup les terrains étaient très différents et les assets que j'avais choisis ne concordait pas avec le reste du groupe. Donc j'ai été obligé de refaire ma partie mais cette fois-ci avec des idées plus claires. Pour ma part, j'ai été contraint, parce que je ne savais pas du tout comment faire un terrain sur Unity et j'ai alors dû tout recommencer de zéro. Et c'est en regardant des tutoriels et des idées de terrains que j'ai pu mettre en place le désert. Pourtant, il n'est toujours pas fini puisque je compte y mettre des modifications; plus précisément, je vais reprendre l'idée de dunes pour le terrain et du village abandonné, enterré.

Image de la dernière version du désert



## Description de la mise en place de la “Zone Asiatique” par Paul

Pour le terrain de jeu, nous avions donc décidé de faire une partie de la map chacun de notre côté, j'ai dû tout apprendre depuis le début étant donné que je n'avais jamais utilisé Unity pour ce genre de chose, et encore moins pour faire du graphisme. Après avoir regardé des cours, des tutoriels... J'ai commencé par la création de la map en essayant de garder cet univers de LOW POLY, j'ai dû placer les assets, trouver des bonnes textures... Une fois la map terminée et lors de la mise en commun, les premiers problèmes ont commencé. En effet tout d'abord les assets étaient trop différents des autres, de plus les dimensions étaient beaucoup trop petites par rapport aux autres, les colliders ne marchait pas, rien n'était bon. J'ai dû repartir de zéro, et tout refaire en accord avec les autres et cette fois-ci avec de la communication en plus, en effet le manque de communication a été flagrant, et ça a affecté tout le monde. La création de la map a donc pris beaucoup plus de temps que prévu. J'ai donc abouti à un premier croquis de ma zone de jeu, mais je pense encore qu'il va y avoir des changements au cours du temps pour améliorer et changer des aspects de la map.

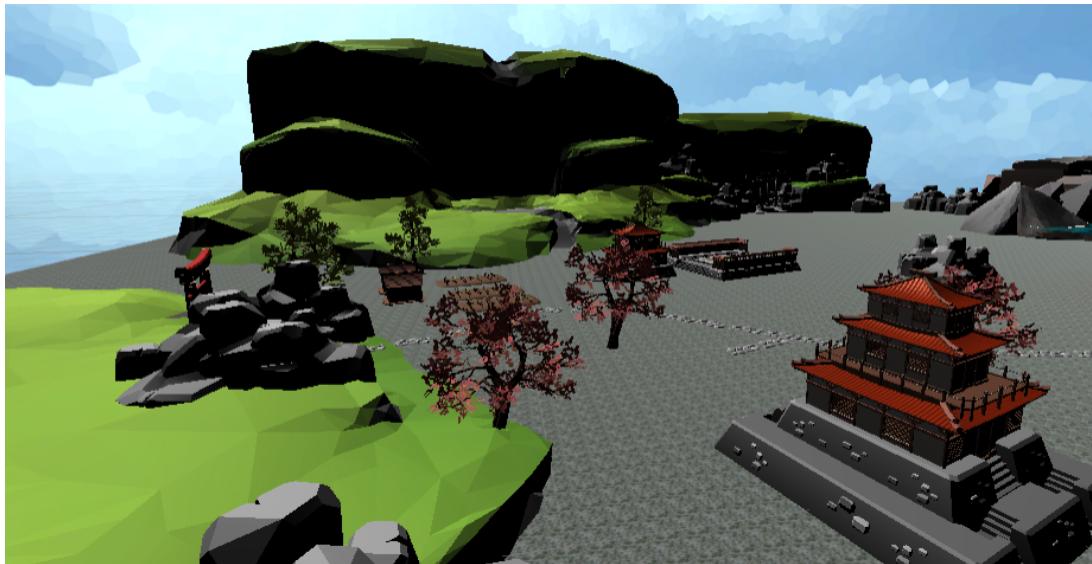
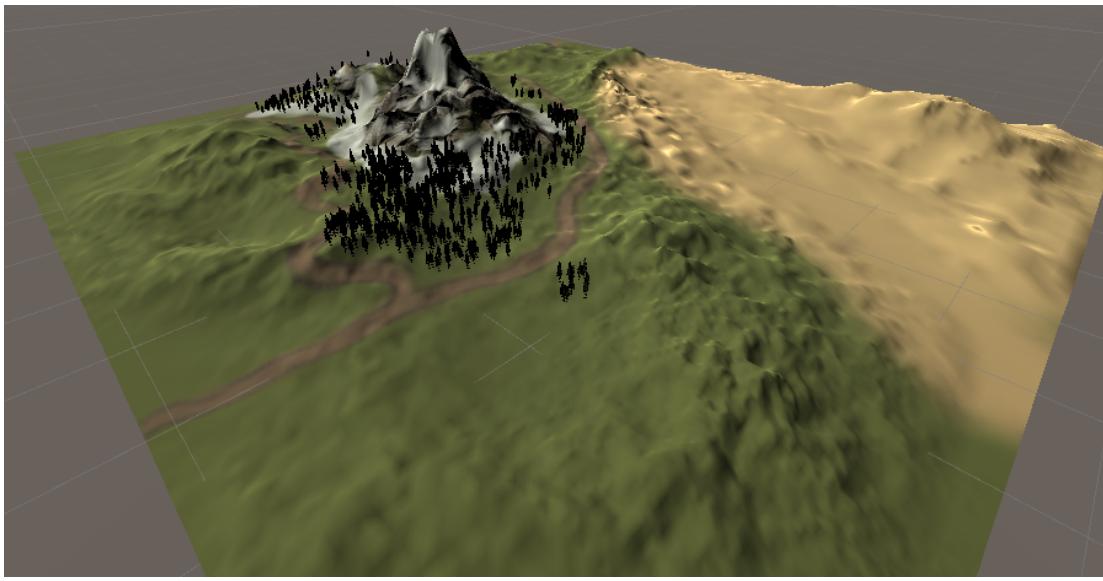


Image de la dernière version de la zone asiatique

## Description de la mise en place de la “Zone Été” par Antoine

Pour la réalisation ainsi que la mise en place de ma zone été, j'ai fait tout un tas de croquis et d'essais pour parvenir à un rendu qui me convient. J'ai essayé de procéder de façon méthodique. J'ai tout d'abord commencé par regardé de nombreux tutoriels sur Youtube pour réaliser et comprendre les bases de construction d'un terrain 3D sur Unity. Je suis alors arrivé à un rendu plutôt réaliste mais cependant qui ne correspondait pas aux attendus mis en place dans notre cahier des charges.

Image de la précédente version de la zone été



Malgré le fait que cela ne m'ait pas fait avancer dans le projet, cela m'a permis de mieux gérer l'utilisation des outils d'Unity ainsi que ces fonctionnalités. Pour continuer, j'ai donc trouvé un pack entier d'assets en LowPoly qui plus tard à été repris par le groupe entier. L'ensemble du groupe a beaucoup apprécié ce package et nous nous en sommes tous servis.

Pour ma part, j'ai donc dû réaliser une zone été, j'ai directement pensé à créer un endroit ressemblant à une île, puis, au fur et à mesure je suis parvenu à un créer une zone avec des parties distinctes. La première chose que j'ai fait pour mon terrain c'était donc d'installer un Package Manager nommé "Terrain Tools" qui m'a permis de créer un terrain avec de nombreux reliefs et qui m'a facilité le travail puisque ce Package contient de nombreux outils pour manipuler au mieux ma zone.

Une fois les reliefs et les textures appliquées, j'ai créé un lac au centre de ma zone avec une cascade. Cette cascade m'a pris pas mal de temps puisque j'ai dû l'animer de façon à ce que l'eau tombe de la bonne manière et j'ai dû créer des jeux de couleurs pour que la cascade semble suffisamment réaliste malgré qu'elle soit en low poly.

Ensuite, j'ai animé la zone autour du lac avec les assets que nous avons sélectionnées. Après la partie du lac, j'ai ensuite créé deux autres petites parties. L'une étant une petite zone de trois maisons et l'autre étant un campement dans une forêt. Les deux petites parties que j'ai créées ont été faites à partir des Prefabs des assets que nous avions importé.

Une fois que j'avais un beau rendu pour ma zone été, j'ai voulu la mettre en commun avec les autres membres du groupe, mais j'avais fait de nombreuses erreurs. En effet, je n'avais pas la même taille de terrain que les autres, j'avais modifié la "scale" de chacun des assets et j'avais donc chaque GameObject avec une "scale" différentes et j'ai eu des problèmes de colliders avec mes arbres. J'ai alors dû refaire l'intégralité de ma zone en l'intégrant avec le terrain de chacun. Ceci à été un peu plus rapide que lors de la première conception de ma zone été puisque j'avais déjà un modèle et je maniais bien mieux Unity.

J'ai donc abouti à cette zone là qui nécessitera encore quelques modifications pour la rendre plus esthétique et plus agréable à jouer mais également pour mettre en place d'avantages d'endroits pour se cacher.

*Image de la dernière version de la zone été*



## Description de la mise en place du “Zone Hiver” par Emre

Dans un jeu où des trolls pourchassent des elfes, il nous manquait un univers : celui du froid. La froideur des trolls, l'univers glacé des icebergs et des montagnes. C'est alors que nous avons eu l'idée de créer une partie où l'Hiver Éternel règne en maître. Sur cette zone, nous retrouvons l'univers glacial où seul l'Hiver est le seul vainqueur.

Sur cette zone, nous retrouvons aussi des particules. Pas n'importe quelle particule mais deux types de particules :

- Un épais brouillard : assez pour faire disparaître les elfs au loin.
- Une pluie de neige qui ne finit pas.

Tout ceci mélangé à un terrain enneigé où est posé un village vide englouti à moitié avalé par la neige. Pour faire le terrain enneigé, nous avons, pour commencer, appliquer une hauteur aléatoire sur cette partie de la zone et ensuite nous avons utilisé l'outil "Smooth" qui permet d'atténuer en moyennant les hauteurs. Le résultat nous donne un univers assez montagneux. Ensuite nous avons utilisé les assets que les différentes zones partagent en ajoutant des arbres sans feuilles, des troncs d'arbres et des sapins enneigés.

*Figure : Version bêta de la Zone Hiver*



# Sons, bruitages et musique

En ce qui concerne la partie son, nous avons utilisé de nombreuses librairie :

- [universal-soundbank.com/mouvements.htm](http://universal-soundbank.com/mouvements.htm)
- <https://lasonotheque.org/>

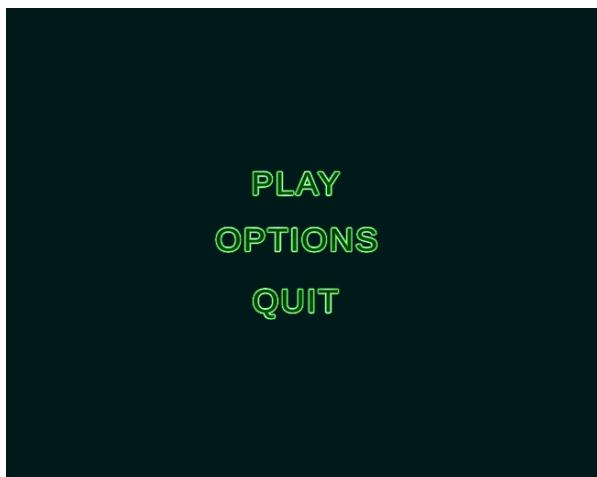
- <https://www.sound-fishing.net/sons/bruit-pas-terre>
- <https://www.soundsnap.com/>
- <http://dig.ccmixter.org/>

Avec une énorme variété de sons et de bruits, nous avons pu choisir des musiques dynamiques qui conviennent bien à notre jeu, nous voulons utiliser le logiciel FL studio si jamais nous avons des modifications à faire. L'objectif pour nous et de réussir à intégrer les sons au script des personnages pour la prochaine soutenance !

# Interface

Pour ce qui est des menus, nous avons commencé à mettre l'interface en place, mais actuellement tous les boutons à l'exception du “PLAY”, “OPTIONS”, “QUIT”, “Return” sont non opérationnels.

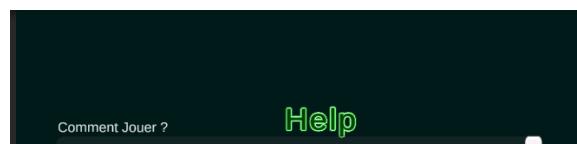
Menu D'accueil du jeu



Menu Options



Menu Help



# Site WEB

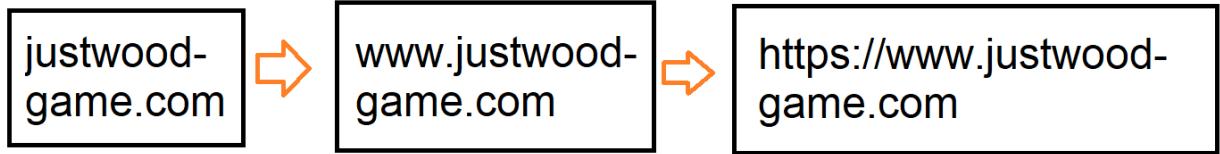
Le lien: <https://www.justwood-game.com/>

Pour la présentation de notre projet, et pour appuyer notre soutenance, nous avons décidé de créer un site web, qui va nous permettre d'avoir accès aux différentes versions du jeu. Il y aura également les explications et les motivations de ce projet, le mélange entre le scolaire, un projet imposé par EPITA mais également le loisir de faire un jeu et d'apprendre de nouvelles choses.

Nous allons continuellement améliorer notre site pour qu'il soit complet pour la dernière soutenance.

Le site web possède un certificat SSL et est hébergé sur le même serveur dédié que le serveur de jeu. Nous l'avons mis en place grâce à Apache2 et nous avons écrit un fichier de configuration pour être interprété par Apache2.

Pour donner une illustration de la configuration :



Si un utilisateur se dirige vers <http://justwood-game.com>, il sera alors redirigé vers une bonne URL : c'est-à-dire <https://www.justwood-game.com>

Nous utilisons aussi ce moyen pour faire les différentes mise à jour concernant le serveur du jeu : c'est-à-dire que nous communiquons le lien du site directement à "l'egg" du serveur de jeu sur Pterodactyl.

## Nos objectifs :

Pour cette soutenance nous avions deux grands objectifs :

- Se mettre au clair sur les attentes de notre jeu en déterminant précisément le cadre du jeu vidéo et la répartition des tâches au sein de notre groupe en fonction des qualités et des envies de chacun (évidemment tout le monde va toucher un peu à tout !) .
- Nous avions encore quelques doutes quant au mode multijoueur, réussir à créer notre propre serveur ou réussir à faire apparaître les "zones rouges" en mode multijoueur. Après recherche et réflexion nous avons maintenant la certitude que nous pouvons relever ce challenge et intégrer un mode multijoueur sans réel problème.

Pour la prochaine soutenance les attentes et les objectifs seront différents :

- Continuer à avancer sur la partie réseaux et surtout commencer et terminer la création de zones rouges qui représentent un élément essentiel pour notre jeu vidéo.

- Quant à l'avancée globale du jeu, nous aimerais réussir à avoir un jeu quasiment complet d'environ 80% et fonctionnel. Pour être prêt et en avance avant la dernière soutenance.
- Concernant la partie réseaux, il nous faudra mettre en place les “rooms” public et privé pour que les joueurs puissent rejoindre et jouer avec leurs amis. Nous prévoyons aussi d'implémenter une base de données pour enregistrer les informations de chaque joueur.

## Images Annexes :

Version 2 du troll



Version actuelle de l'elfe (mixamo)



Dernière version des personnages troll



Dernière version des personnages troll



