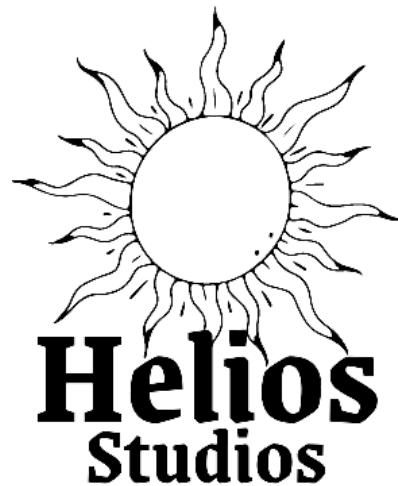


Rapport de soutenance

Par Helios Studios

Mars 2024



**KING SLAYER**

## Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Reprise du cahier des charges</b>	<b>3</b>
2.1	Identification de nos objectifs . . . . .	3
2.2	Contexte et justification . . . . .	3
2.3	Description générale du projet . . . . .	3
2.4	Le jeu . . . . .	3
2.5	Contraintes et limitations . . . . .	4
<b>3</b>	<b>Chronologie</b>	<b>4</b>
3.1	La chronologie de groupe . . . . .	4
3.2	Chronologie individuelle . . . . .	5
3.2.1	Gabriel . . . . .	5
3.2.2	Rania . . . . .	9
3.2.3	Jimmy . . . . .	12
3.2.4	Nicolas . . . . .	16
3.2.5	Jubair . . . . .	18
<b>4</b>	<b>Récits de réalisations</b>	<b>19</b>
4.1	Recit de réalisations du groupe . . . . .	19
4.2	Récit de réalisation personnel . . . . .	20
4.2.1	Gabriel . . . . .	20
4.2.2	Rania . . . . .	20
4.2.3	Jimmy . . . . .	20
4.2.4	Nicolas . . . . .	21
4.2.5	Jubair . . . . .	21
<b>5</b>	<b>Les annexes</b>	<b>22</b>

## 1 Introduction

C'est avec honneur et joie que nous, Hélios studio allons pouvoir vous présenter les avancements quant au développement de notre jeu dans les pages à venir. Hélios, faisant appel au Dieu de la mythologie grecque du soleil. Au-delà de l'esthétique du nom, il évoque avant tout la mythologie et la fantaisie, ce sur quoi est basé tout l'univers de notre jeu ! A cet univers qui nous réunit et qui nous passionne nous avons souhaité ajouter ce qu'il pouvait y avoir de plus en vogue en ce moment dans le jeu-vidéo: le battle royal !

KingSlayer, notre jeu-vidéo, embarque tout joueur assez fou pour se mêler à l'aventure, dans la peau d'une créature innocente, qui se voit malgré elle être mêlé aux jeux pervers imposés par leurs rois.

Durant votre lecture de notre rapport de soutenance, vous pourrez ainsi lire et voir notre implication dans le jeu, et donc en savoir plus non seulement sur notre avancée mais aussi sur notre ressenti durant la création du jeu. Vous verrez donc les premiers designs ainsi que des bouts de code que nous avons estimés importants.

## 2 Reprise du cahier des charges

### 2.1 Identification de nos objectifs

Dans le Cahier des Charges Fonctionnel (CdCF), notre objectif était de créer un jeu vidéo multijoueur “Battle Royale” avec pour style “Medieval Fantasy” à l'aide d'Unity. Pour cela, nous devons créer nous-mêmes certains assets comme les personnages. Nous cherchons à vouloir créer un monde magique, à l'apparence enfantine mais satiriquement dystopique, avec une histoire sombre. Nous souhaitons également exploiter le fait que le mode de jeu “Battle Royale” se popularise, avec des opportunités propices à la génération de revenus.

### 2.2 Contexte et justification

Ce projet n'est pas né par hasard, bien que les idées de nos coéquipiers divergeaient au début. Certains voulaient faire un jeu vidéo “FPS (First Person Shooter)” alors que d'autres souhaitaient faire un “RPG (RolePlay Game)” pour une aventure en coopération. Finalement, en amalgamant les idées et en faisant des compromis, nous nous sommes finalement mis d'accord et avons décidé de faire un “Battle Royale” dans un monde fantasy rempli d'ennemis donc non seulement du “PvP (Player Versus Player)” mais aussi du “PvE (Player Versus Environment)” comme dans un jeu RPG. Cette inspiration vient des films comme “Hunger Games” pour son genre “Battle Royale” avec des jeux comme “Super Animal Royale” et bien “PUBG”.

### 2.3 Description générale du projet

Pour réaliser ce jeu vidéo, nous utilisons “Unity”. C'est ce qui permet du prototypage rapide et une possibilité d'utiliser ses services pour le multijoueur. Nous codons sur C# pour gérer le multijoueur, les comportements d'objets et l'interface graphique. Nous essayons de produire nous-mêmes un maximum d'assets pour notre jeu afin qu'ils collent au jeu que nous imaginons et nous sommes en contact avec une personne pour produire une musique principale de notre jeu, cependant, en ce qui concerne les sons nous utiliserons ceux qui sont libres de droits.

Nous attendons de ce projet un jeu téléchargeable sur Windows, jouable avec clavier, souris avec une expérience de jeu “PvE” et “PvP” donc compétitif avec des aspects de survie et de compétition. Nous nous attendons à ce que chaque partie dure plus ou moins 20 minutes.

### 2.4 Le jeu

Ce jeu consiste alors en un monde post-apocalyptique rempli de magie mais aussi de corruption et d'antagonistes tels que des gobelins, morts-vivants et d'autres monstres. Avec un peu de chance, on aura la possibilité de rencontrer des boss qui sont très puissants et les vainqueurs de ces boss auront un avantage significatif. Ce jeu insistera aussi aux joueurs de pouvoir faire des alliances dynamiques,

c'est-à-dire des sortes de trêves temporaires pour remplir un certain objectif et toujours une possibilité de trahir les joueurs comme dans "Hunger Games". Afin de garantir un gameplay consistant, les joueurs dans l'œil de la tempête devront se déplacer pour rester dans l'œil de la tempête afin d'éviter les monstres qui apparaissent bien trop nombreux et puissants qui n'arrêteront pas de pourchasser les joueurs en dehors de l'œil.

## 2.5 Contraintes et limitations

Pour mener à bien notre projet, il était impératif à ce qu'on se renseigne sur des documentations non seulement sur C# d'en plus de nos cours d'Informatique Pratique, mais aussi celui d'Unity. Dans notre environnement de travail, il est nécessaire d'avoir un ordinateur sur lequel travailler avec des IDE comme Visual Studio (Code) et également Unity pour pouvoir éditer notre jeu vidéo et organiser nos assets. Cependant notre contrainte actuelle est que nous sommes une entreprise "start-up", donc un budget limité à utiliser avec parcimonie, une équipe de 5 personnes et un deadline vers l'été.

# 3 Chronologie

## 3.1 La chronologie de groupe

### Introduction

Dans notre aventure sur ce projet, nous avons suivi attentivement le plan que nous avions établi dans notre diagramme de Gantt initial. Nous avons respecté les délais fixés, ce qui démontre notre engagement et notre sérieux. Pour en arriver là, nous avons adopté une approche organisée, marquée par des réunions régulières les mardis et vendredis pour discuter de nos avancées et planifier la suite.

### Novembre-Décembre

Tout a commencé avec les idées visuelles brillantes de Jimmy, notre designer talentueux. Ces idées ont servi de base pour développer les différentes parties du projet. Un moment crucial a été la mise en place de la fonction multijoueur, dirigée de façon experte par Gabriel. Ce travail a renforcé notre infrastructure technique et a favorisé une collaboration harmonieuse entre tous les membres de l'équipe.

Pendant ce temps, Gabriel et Rania se concentrait sur la création du lobby, travaillant étroitement avec notre designer et le responsable du multijoueur. Leur objectif était d'intégrer les serveurs de manière transparente et de garantir une expérience fluide tout en mettant l'accent sur le design. Bien qu'il reste quelques détails techniques à peaufiner, cette partie essentielle est sur le point d'être finalisée.

### Décembre-Février

Nicolas, de son côté, s'est investi dans la conception de l'interface utilisateur, en particulier sur l'inventaire du joueur et son interface graphique. Sa collaboration étroite avec notre designer a assuré une expérience utilisateur cohérente et facile à prendre en main. Alors que la partie technique est presque terminée, quelques ajustements mineurs sont encore nécessaires pour parfaire le résultat final.

En parallèle, Jubair a accompli des avancées significatives dans l'implémentation des mécanismes de déplacement des joueurs et dans l'amélioration de leur système de caméra, ainsi que dans la conception des systèmes de détection des IA pour les ennemis et leurs animations. Bien que ces progrès soient prometteurs, des étapes cruciales telles que l'intégration des mécanismes d'attaque et l'amélioration des aspects graphiques demeurent à venir.

### Février-Mars

Enfin, Jimmy et Gabriel ont également développé un site web de qualité après avoir terminé la fonction multijoueur. Ce site, pensé pour offrir une expérience utilisateur optimale, intègre toutes les fonctionnalités nécessaires pour soutenir notre projet. Jimmy a apporté son expertise en design pour créer une interface attrayante et facile à utiliser, tandis que Gabriel s'est assuré du bon fonctionnement technique pour une expérience fluide.

### Prochainement

Pour les prochaines étapes du projet, nous prévoyons que Gabriel se chargera des tâches restantes, notamment la conception de la carte et du système de zones. Ces éléments viendront compléter notre vision d'un projet abouti et fonctionnel, prêt à être déployé avec succès. En résumé, notre avancée méthodique et notre engagement sans faille montrent notre détermination à mener ce projet à bien.

## 3.2 Chronologie individuelle

### 3.2.1 Gabriel

En tant que délégué sur le multijoueur et la programmation (Gabriel), l'objectif principal est alors la programmation des comportements des éléments du jeu, appelés “GameObjects”, tout en prenant compte des contraintes du “multijoueur”. Afin de rendre le jeu jouable en multijoueur, il est important de tout d'abord comprendre comment cela fonctionne.

Mi-novembre: Début des recherches sur le fonctionnement du multijoueur sur Unity.

Pour me renseigner, j'ai dû rejoindre un serveur discord officiel d'Unity qui s'appelle “Unity Multiplayer Networking” et j'ai pu alors me documenter sur les services “Relay” et “Lobby” qui sont très bien documentés et permet une bonne prise en main après la lecture.

On apprend aussi que le multijoueur peut se réaliser de plusieurs manières:

- En hébergeant un serveur. Ce serveur va pouvoir gérer les relations clients-serveur à lui seul. Le serveur va permettre d'actualiser le jeu des clients par rapport à ce que le serveur reçoit (les “events” par les clients ou par lui-même) et ce qu'il renvoie.

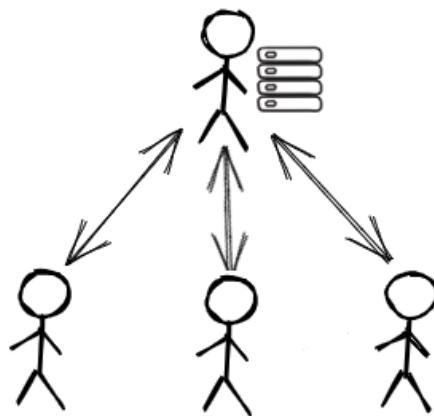


Figure 1: Représentation du système client-serveur, où les clients sont représentés par des personnages “stickmen”.

- En “Peer-to-Peer (P2P)”. Le P2P est une relation client-client et il existe plusieurs moyens pour y parvenir.

Le P2P direct consiste en des clients interconnectés et n'a besoin de serveur. En reliant tous les clients par eux-mêmes, il est alors possible de remplir le rôle de serveur, c'est-à-dire d'actualiser le jeu à tous les clients interconnectés.

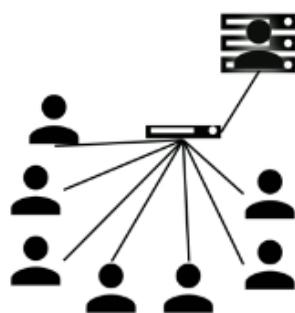


Figure 2: Représentation d'un P2P direct.

Le P2P client-hôte de serveur. Cette forme de P2P est un moyen où le client “hôte” va remplir le rôle de serveur et pouvoir gérer les connexions entre les clients.

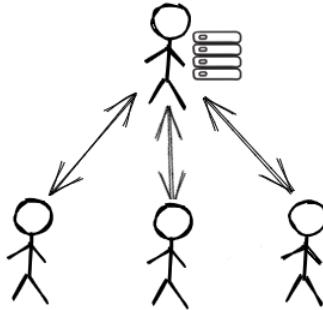


Figure 3: Représentation d'un P2P client-hôte de serveur.

Et dans notre cas, nous utilisons le service “Relay” d’Unity qui est du P2P client-hôte où les clients devront passer par un serveur intermédiaire d’Unity pour ensuite se joindre à l’hôte. C’est une façon sécurisée et pratique pour la programmation puisque nous pouvons utiliser ce système pour par exemple définir le nombre maximum de clients qui peuvent se connecter. C'est alors un moyen qui facilite les connexions entre les clients et le l'hôte.

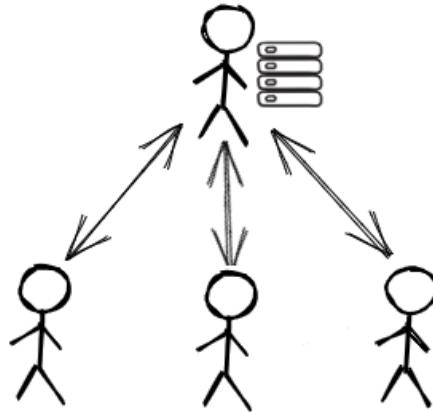


Figure 4: Représentation du P2P “Relay” d’Unity. Le serveur intermédiaire n'est alors qu'une sorte de pont pour permettre la connexion entre l'hôte et les clients.

Début décembre - début janvier: Début de l’implémentation du multijoueur dans l’éditeur Unity. Il était difficile de commencer à implémenter ce système de P2P pour permettre le multijoueur puisqu'il faut des connaissances non seulement de la bibliothèque d’Unity mais également de ses services en plus dont “Relay” et plus tard le “Lobby”.

Afin d’organiser les parties du jeu, il est alors nécessaire pour nous d’utiliser un autre service d’Unity qui s’appelle “Lobby”. C'est ce qui va permettre d’organiser des groupes auxquels les joueurs pourront rejoindre et commencer le jeu.

Pour utiliser les services d’Unity, il faut tout d’abord les initialiser:

```

// ----- Initialize unity services -----
private async Task Authenticate()
{
    await UnityServices.InitializeAsync();
    await AuthenticationService.Instance.SignInAnonymouslyAsync();
    // DisplayErrorMessage("Welcome to my game!");
}
  
```

Figure 5: Méthode Authenticate() pour l’authentification des services.

Ici “await” permet d’exécuter asynchroniquement la ligne de commande et se comporte comme une

“coroutine”, c'est-à-dire qu'il exécute pendant que le jeu tourne (sinon, le jeu stopperait) et en même temps, exécute la prochaine commande qui est le prochain “await”.

Le `DisplayErrorMessage()` mis en commentaire ici est une méthode qui permet d'afficher un message dans le jeu avec pour paramètre un string comme contenu.

Lorsqu'on crée un lobby, il est possible pour nous d'ajouter plus d'informations. Comme le code qui permet de se connecter à un hôte. Ce lobby peut également permettre d'être privé ou public. Un lobby privé nécessite au joueur de vérifier un code pour y entrer. Un lobby public lui peut être rejoint avec un code ou être rejoint en utilisant le “QuickJoin” du service Lobby, qui permet de rejoindre un lobby public aléatoirement.

```
public async void CreateLobby()
{
    try
    {
        // Check correct lobby name
        if (string.IsNullOrWhiteSpace(_lobbyNameInput.text))
        {
            DisplayErrorMessage("Invalid lobby name!");
            return;
        }
    }
}
```

Figure 6: Méthode `CreateLobby()`.

Dans cette méthode `CreateLobby`, nous vérifions d'abord si le nom du Lobby est correct. En utilisant `IsNullOrEmptySpace` du module `string` de C#, on vérifie pour l'instant si le nom donné par le joueur n'est pas vide ou rempli uniquement d'espaces. Dans ce cas là, on fait afficher au joueur que le nom de lobby est incorrect.

Lorsqu'on crée des méthodes en utilisant les services d'Unity, il est important de toujours faire un try-catch. Lorsqu'un service devient indisponible, au lieu de stopper le processus et crasher le jeu, on envoie simplement un message d'erreur par un catch.

```
// defining lobby options from player input
CreateLobbyOptions createLobbyOptions = new CreateLobbyOptions()
{
    Data = new Dictionary<string, DataObject> { { "JoinCodeKey", new DataObject(DataObject.VisibilityOptions.Public, joinCode) } },
    IsPrivate = _private.isOn,
    Player = GetPlayer() // the host's
};
```

Figure 7: Crédit de la création du serveur relais.

Dans la figure 6.b, on crée un relais avec un nombre défini par le joueur (qui est bel et bien limité de 2 à 8). Ensuite, nous récupérons le code qui va être utilisé pour rejoindre l'hôte par le serveur intermédiaire.

```
Allocation a = await RelayService.Instance.CreateAllocationAsync((int)_maxPlayersInput.value);
string joinCode = await RelayService.Instance.GetJoinCodeAsync(a.AllocationId);
```

Figure 8: `CreateLobbyOptions`.

La création de ce lobby peut prendre en paramètre un “`CreateLobbyOption`” qui seront alors les options que nous pouvons choisir. Ici par exemple, en fonction des entrées du joueur, on va pouvoir rendre ce lobby privé ou public. On met aussi le nom d'utilisateur de l'hôte du lobby dans “`Player`”.

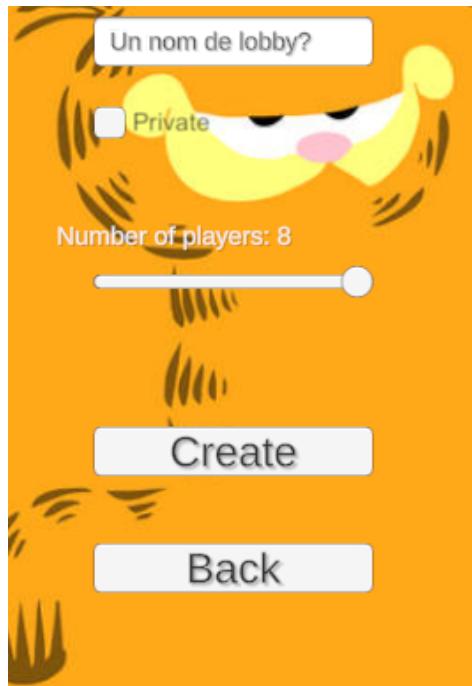


Figure 9: Création du lobby.

Le lobby va alors être créée en appliquant les options dans les paramètres dans cet ordre: le nom du lobby, le nombre maximum de joueurs, et les options choisies. Tout cela en fonction du choix du joueur tel que “\_lobbyNameInput” ou bien “\_maxPlayersInput”.

Après cela, on manipule l’interface pour faire afficher le lobby avec la liste des joueurs en appelant une méthode qui permet de rafraîchir le lobby actuel si des joueurs on rejoint celui-ci.



Figure 10: Un extrait du prototype lorsqu'on crée un lobby.

Cet extrait est sujet à énormément de modifications comme ce n'est qu'un prototype et ne sert qu'à implémenter le multijoueur.

Début janvier:

Après cela, nous avons une dizaine de méthodes à dispositions en se servant des services d’Unity comme pour expulser un joueur du lobby, quitter le lobby (ensuite la rendre hors-service si c'est l'hôte qui quitte), rendre ce lobby privé ou public en live et enfin commencer le jeu avec les joueurs présents dans le lobby.

En dehors de la création du lobby, nous avons également le choix de rejoindre un lobby d'un autre joueur.

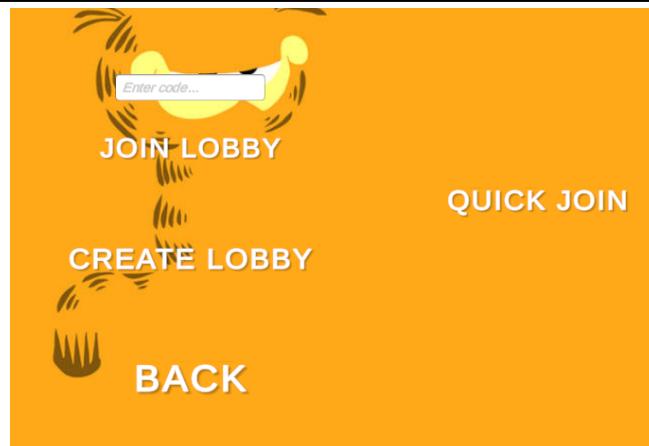


Figure 11: Page de sélection de lobby.

Ici le “JOIN LOBBY” permet de rejoindre un lobby privé/public avec un code.  
“CREATE LOBBY” permet de faire afficher les paramètres du lobby pour ensuite la créer.

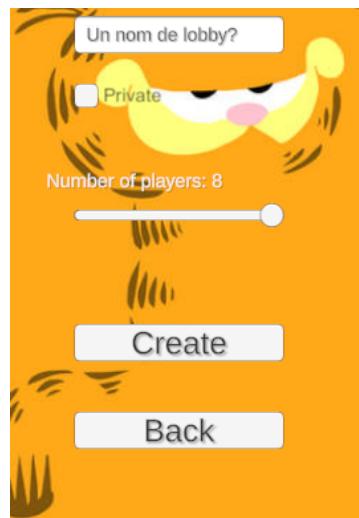


Figure 12: Paramétrage du lobby.

Et enfin “QUICK JOIN” pour rejoindre un lobby public au hasard ou bien dans un ordre particulier comme par nombre décroissant de joueurs pour remplir aussi vite les lobby.

Sources pour les illustrations: <https://blog.hathora.dev/peer-to-peer-vs-client-server-architecture/>  
<https://docs.unity.com/ugc/en-us/manual/relay/manual/integration>

### 3.2.2 Rania

Durant le mois de novembre :

Nous avons commencé à travailler concrètement sur le projet après avoir fait plusieurs réunions pour se mettre d'accord dessus, j'ai ainsi la charge du menu principal, seulement nous avons eu un malentendu ce qui m'a empêché de travailler sur le projet durant une semaine. Après cela je commence à travailler sur le menu principal après avoir eu des problèmes pour l'installation de Visual Studio Code. Le menu principal ne fut pas compliqué à faire au vu du fait que tous les menus sont faits de la même manière, n'importe quel tutoriel sur YouTube aurait servi, cependant ce fut très frustrant de perdre ce qu'on faisait au début. Les fonctions pour animer le menu principal n'ont pas été difficile non plus à comprendre et à faire. Voici le script qui commande le bouton “Start” qui sert à signaler qu'on est prêt à jouer et pour ainsi nous emmener vers le Lobby :

```

public class main : MonoBehaviour
{
}

0 references
public void Play()
{
    SceneManager.LoadSceneAsync(1);
}

```

Figure 13: La fonction qui implemente le bouton Play

La fonction Play ici sert à charger une nouvelle scène en arrière plan sans bloquer l'exécution du programme.

Voici le script pour le bouton “Exit” qui sert à sortir de l'application:

```

0 references
public void Quit()
{
    Application.Quit();
}

```

Tout ceci réuni, cela m'a permis d'avoir un menu principal de ce type:

Grace au bouton de paramètres, nous pourrons accéder aux différentes options pour régler le son:

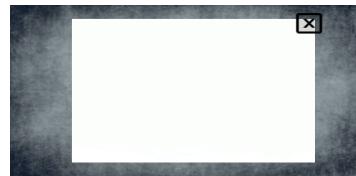


Figure 14: Le carré blanc sera là où se trouveront les options pour le son

J'ai à ce moment aussi pu m'occuper du son du Menu Principal voici mon code:

```

5  public class audiomang : MonoBehaviour
6  {
7      [Header("---Audio Source ---")]
8      [SerializeField] AudioSource musicSource;
9
10     [Header("----Audio Clip-----")]
11     public AudioClip background;
12
13     public AudioClip touch;
14
15     private void start()
16     {
17         musicSource.clip = background;
18         musicSource.loop = true;
19         musicSource.Play();
20     }
21
22 }
23

```

A partir de décembre:

Ma deuxième tâche fut plus compliquée, j'eus donc la charge du lobby, mon coéquipier, Gabriel était lui en charge du multijoueur, nos tâches se ressemblent donc puisque le lobby doit être fait en multijoueur. Le lobby était réellement très long et très frustrant à faire car il y avait beaucoup moins de tutoriels avec multijoueur. Il a donc fallu tout faire en étant beaucoup plus autonome, il fallait

donc se rappeler de ce qu'on a appris durant la création du menu principal et s'aider de différents tutoriels trouvés sur internet afin d'avoir le rendu voulu. Ce fut donc laborieux. Du a un second manque d'attention, j'ai malheureusement empiété sur le travail de mon camarade Gabriel mais voici tout de même des scripts de mon lobby :

Je vais donc présenter et expliquer certains scripts de mon Lobby, en commençant par la fonction qui crée de Lobby :

La fonction createlobby crée un lobby nommé My Lobby qui pourra accueillir jusqu'à 8 joueurs. Les options de création du lobby sont définies dans la partie createLobbyOptions et elles incluent le public, les données de jeu, et le joueur créateur. Une fois crée grâce à CreateLobbyAsync(), les détails du lobby pourront être affichés dans la console via Debug.Log. En cas d'erreur, le bloc try-catch gère toute l'exception de type LobbyServiceException en affichant le message correspondant.

```

13  public class lobby : MonoBehaviour
14  {
15      [System.Serializable]
16      public class CreateLobbyOptions
17      {
18          public string Name;
19          public int MaxPlayers;
20          public bool IsPrivate;
21          public Player Creator;
22          public Dictionary<string, DataObject> Data;
23          public string GameMode;
24          public VisibilityOptions Visibility;
25      }
26
27      private void CreateLobby()
28      {
29          try
30          {
31              string lobbyname = "My Lobby";
32              int maxPlayers = 5;
33              CreateLobbyOptions createLobbyOptions = new CreateLobbyOptions{
34                  IsPrivate = false,
35                  Creator = GetPlayer(),
36                  Data = new Dictionary<string, DataObject>{
37                      {"GameMode", new DataObject(DataObject.VisibilityOptions.Public, "CaptureTheFlag")},
38                  };
39              };
40              Lobby lobby = await LobbyService.Instance.CreateLobbyAsync(lobbyname, maxPlayers, createLobbyOptions);
41
42              hostlobby = lobby;
43
44              Debug.Log("Created Lobby ! " + lobby.Name + " " + lobby.MaxPlayers + " " + lobby.Id + " " + lobby.LobbyCode);
45
46              Printplayers(hostlobby);
47          }
48          catch (LobbyServiceException e)
49          {
50              Debug.Log(e);
51          }
52      }
53  }

```

La seconde fonction que je vais présenter est celle qui liste les Lobby présents si on décide de jouer grâce au QuickJoin. Elle utilise la fonction QueryLobbiesAsync() pour rechercher les lobbies en utilisant des options de requête comme le nombre maximal de lobbies à récupérer, les filtres (par exemple, en recherchant les lobbies avec des places disponibles), et l'ordre de tri des résultats. Si la requête réussit, les détails de chaque lobby trouvé, tels que le nom, le nombre maximal de joueurs et le mode de jeu, sont affichés dans la console de débogage.

```

13  public class lobby : MonoBehaviour
14  {
15      [System.Serializable]
16      public class QueryLobbiesOptions
17      {
18          public int Count;
19          public List<QueryFilter> Filters;
20          public OrderOptions Order;
21      }
22
23      private void ListLobby()
24      {
25          try
26          {
27              QueryLobbiesOptions querylobbiesoptions = new QueryLobbiesOptions{
28                  Count = 25,
29                  Filters = new List<QueryFilter>{
30                      new QueryFilter(QueryFilter.FieldOptions.AvailableSlots, "0", QueryFilter.OpOptions.GT)
31                  },
32              };
33
34              Order = new List<QueryOrder>{
35                  new QueryOrder(false, QueryOrder.FieldOptions.Created)
36              };
37          }
38
39          catch (LobbyServiceException e)
40          {
41              Debug.Log(e);
42          }
43
44          QueryResponse queryResponse = await Lobbies.Instance.QueryLobbiesAsync();
45
46          Debug.Log("Lobbies found: " + queryResponse.Results.Count);
47          foreach (Lobby lobby in queryResponse.Results)
48          {
49              Debug.Log(lobby.Name + " " + lobby.MaxPlayers + " " + lobby.Data["GameMode"].Value);
50          }
51
52      }
53  }

```

Notre Lobby est désigné par Jimmy, voici l'image de fond de notre Lobby:



### 3.2.3 Jimmy

L'un des meilleurs moyens pour faire passer un message, une émotion au travers d'un jeu, c'est de fournir une direction artistique immersive. Chargé de cette dernière, j'ai l'honneur d'avoir le rôle de transporter les joueurs dans l'univers que propose notre jeu Kingslayer.

#### Mi-Novembre

Après quelques débats d'une courte durée sur le thème de notre jeu et son style nous en sommes venu à la conclusion rapide qu'un battle royale, style fantaisie serait idéal bien qu'ambitieux. L'univers de la fantaisie bien que très à part en comparaison à d'autres comme le réalisme. C'est un style qui permet de grande chose (au moins autant que le réalisme) mais avec une plus grande liberté sur des concepts rigides comme les lois de la physique. Plus liberté, engendre alors plus de simplicité pour les dessins. S'il y a des aspects que je peux simplifier dans le design des certains objets sans que pour cela tout ne perde en cohérence, c'est ravi que je me mette au travail.

Enthousiaste, (peut-être un peu euphorique) je commence les première recherche quant au style du jeu. Il a été décidé que le jeu serait en 2D avec un style cartoon sans passer par du pixel art.

#### Sketchbook

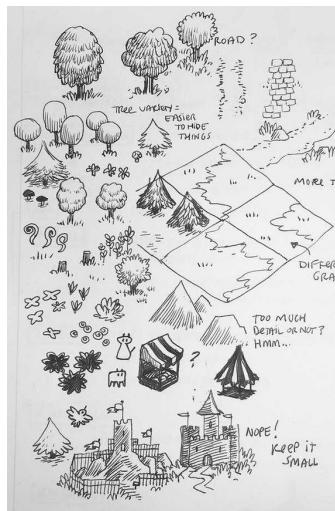


Figure 15: <https://www.pinterest.fr/pin/639300109632854943/>

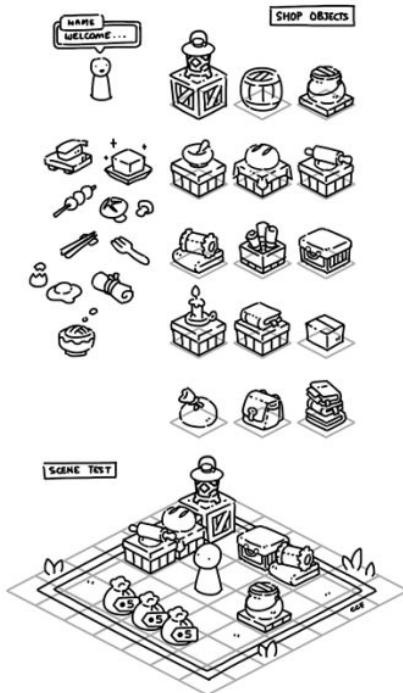


Figure 16: <https://www.pinterest.fr/pin/639300109630208130/>

Ma première et deuxième semaine, sont toutes les deux dédiées à la recherche à la recherche d'une D.A (Direction Artistique) adéquate. J'entends par là, quelque chose de simpliste et "cartoonesque". "Simpliste", car le temps nous a compté. Je ne peux pas tous les jours passer des heures à dessiner et même si je le pouvais d'un point de vue technique mentalement cela pourrait être très éprouvant. "Cartoonesque" car il rejoint ce que j'ai dit avant par sa dimension simple et parce qu'il évoque aussi l'atmosphère imaginaire qui entoure KingSlayer.

Au courant de ces mêmes semaines, je me suis dit qu'un style de vue intéressant à exploiter serait la 2D isométrique. En jeu vidéo, il y a plusieurs styles de vue que l'on peut plus ou moins diviser en 3 grandes parties :

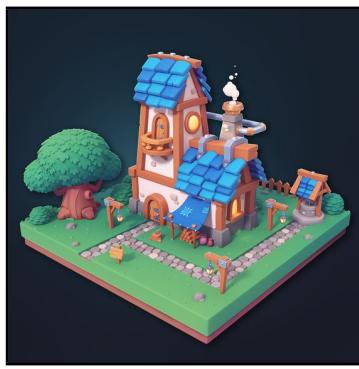


Figure 17: 3D image

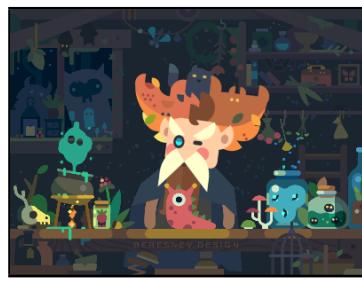


Figure 18: 2D image



Figure 19: isometric 2D image

Concevez ici, la vue isométrique comme une caméra en position plongeante sur le un plan 2D. En d'autres termes on regarde le jeu par le dessus. La meilleure option encore une fois. Un jeu en 3D aurait été trop compliqué à gérer et une un seul plan 2D manquerait de dynamisme. Dynamisme essentiel dans un jeu de combat.

Dans la vue isométrique, on donne une impression de 3D grâce à la perspective obtenue par notre

point de vue. La grille qui nous permet un tel rendu est composé de losange (et parfois de transversale verticale) dont les trois arêtes principales (qui correspondent aux trois dimensions de l'objet) forment des angles égaux de 120°.

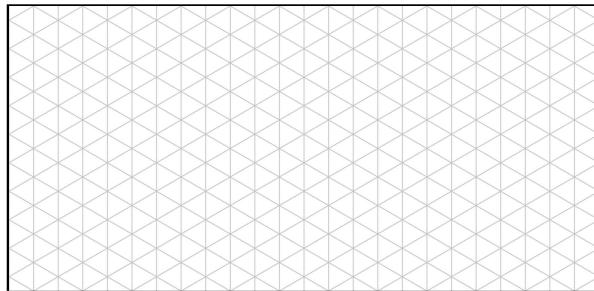


Figure 20: simple grille isométrique

Au cours de ma troisième semaine de travail je me suis mis à faire les premiers croquis. J'ai pour se faire utiliser le logiciel de dessin procreate qui est une exclusivité IPad. Les fonctionnalités complexe et poussées spécialement pour les designs professionnels m'ont bien aidé notamment au niveau de la grille isométrique. Avec quelque calcules pas trop complique j'ai pu importer les mesures de ma grille de l'iPad vers l'ordinateur, sur Unity:

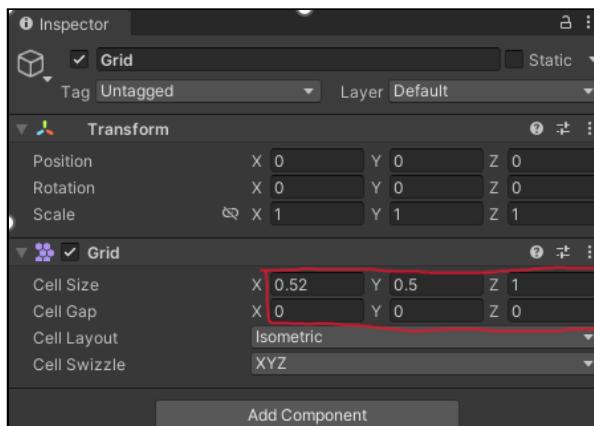


Figure 21: modification du ratio des celulles

### Fin décembre - Fin janvier

De fin décembre jusqu'à janvier je me suis concentré sur la création d'assets. Les assets sont des éléments qui vont servir au jeu. Voyons les ici comme des petits fichiers qui contiennent chacun des éléments précis. Dans le cas d'un personnage et de ses animations, un asset sera plutôt appelé Sprite. Notez aussi que des morceaux de codes peuvent aussi être considéré comment des assets, mais dans mon cas nous n'avons pas besoins de faire attention à cela.

### Février

Pendant cette période, j'ai focalisé mes efforts sur la création de l'inventaire et sur l'aspect esthétique du lobby du jeu. Le but était de rester dans un contexte médiéval avec la notion de roi évoqué dans "KingSlayer".

Pour l'inventaire, j'ai travaillé sur la base de croquis et dessin déjà présent sur Pinterest pour gagner du temps sur l'idée finale que je voulais avoir. Encore une fois une SketchBook a fait l'affaire. L'inventaire fu esthétiquement simple à réaliser. Un layer (couche principale) avec d'autre empiler par-dessus représentant les différents emplacements d'item du jeu.

Quant à l'esthétique du lobby, j'ai avant tout pris le temps de réfléchir au Lore du jeu, l'histoire qui l'entoure. Qui sont les rois ? Quels sont leurs objectifs et quelle place occupe le joueur au cœur de tout cela ? Cette question fut très importante car, je voulais que lorsque les joueurs rejoignent pour la première fois le lobby ils puissent tout de suite avec une idée de ce à quoi ils vont jouer. Que

l'attente du démarrage d'une partie ne se fasse dans une place vide mais réactive et complexe. S'ils en ont le temps ils pourront regarder la pancarte en font, ici comme une sorte de prophétie et tenter de la comprendre.

## Sketchbook

---

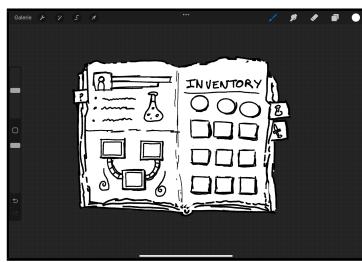


Figure 22: 3D image

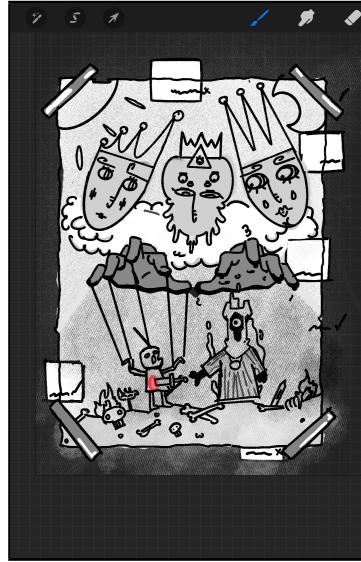


Figure 23: 2D image



Figure 24: isometric 2D image

De plus, la quatrième semaine de janvier, j'ai également commencé à travailler sur les premières animations des personnages pour donner vie à cet environnement. Bien que ce soit un début, ces animations reflètent bien l'aspect cartoon recherché avec très peu d'image par seconde un style un peu plus brouillon qui est voulu.

Dans l'ensemble, mon travail a été guidé par la volonté de créer une expérience de jeu immersive et mémorable, où chaque détail, de l'inventaire à l'esthétique du lobby, contribue à l'ensemble cohérent et captivant du jeu.

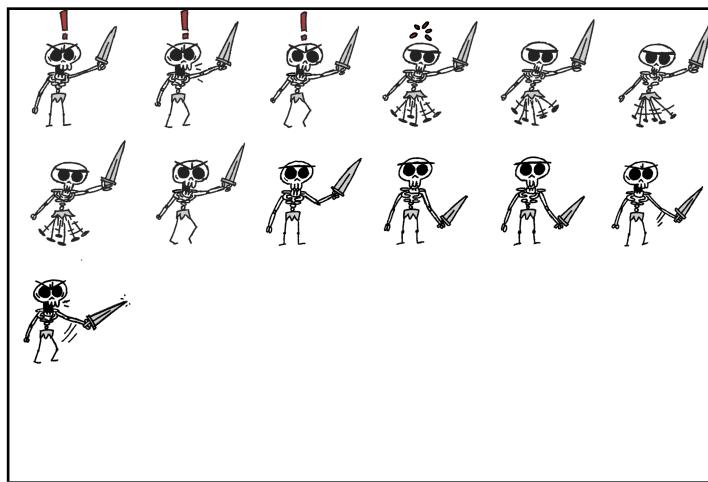


Figure 25: skeleton sprite

### 3.2.4 Nicolas

L'avancement du Game design fut sans problèmes et régulier, notamment en ce qui concerne les différents crafts que les joueurs pourront créer ainsi que leurs ingrédients. Les crafts dans le jeu reposent sur le principe que le joueur doit faire un choix c'est à dire qu'un ingrédient peut être utilisé pour plusieurs crafts, le joueur doit alors choisir s'il veut utiliser son item X pour craft soit l'item A ou l'item B en fonction de la situation dans laquelle il se trouve. Étant donné le gameplay assez nerveux du battle royal les crafts restent simple avec une combinaison simple de 2 ingrédients donnant un item. C'est au joueur de faire appel à son bon sens et à son imagination pour trouver les recettes de craft.

Voici un exemple pour mieux illustrer cela. Les coffres répartis aux quatre coins de la carte contiennent une grande variété d'items générés aléatoirement. Parmi ces items on trouve notamment des fioles vides, des bâtons, des fleurs ou du papier pour en nommer quelques-uns. Les monstres lâches aussi des ingrédients après avoir été vaincu avec les boss lâchant des items plus rares. Les orcs peuvent parfois lâcher des cornes une fois vaincu ainsi que de la bave et le boss Roi Mage lâche son âme et l'artefact. Avec tous ces items une grande variété de crafts s'offre au joueur, il peut utiliser la fiole pour créer une potion de poison avec la bave d'orc ou bien utiliser la fiole pour créer une potion de soin avec la fleur. De même la fleur peut aussi être utilisée avec l'âme du Roi Mage pour créer un collier de fleur magique béni par le dieu Hélios protégeant le joueur contre les dégâts de poison. Le bâton peut être utilisé comme arme assez faible mais peut être combiné avec la corne d'orc pour créer une lance. Cette lance peut être combinée avec la potion de poison rendant la lance empoisonnée. Le papier quant à lui peut être utilisé pour créer un parchemin magique avec l'âme du Roi Mage ou bien peut être combiné avec la lance empoisonnée pour l'essuyer et enlever l'effet de poison si le joueur souhaite rajouter un autre effet. Ainsi chaque ingrédient peut être utilisé dans 2 crafts au maximum et c'est au joueur de décider ce qui est le plus optimal selon la situation.



Figure 26: Potion vide

Finalement le Game design du jeu fut assez fluides lors de sa conception jusqu'à présent vu qu'il s'agissait majoritairement de Brainstorming et de voir ce qui fonctionnerait le mieux dans l'esprit du jeu et si tout cela reste réalisable.



Figure 27: Potion de Vie

Concernant la réalisation de l'inventaire la tâche fut plus ardue. La prise en main d'unity fut assez difficile et l'est toujours même après avoir lu un grand nombre de documentation et avoir visionné plusieurs vidéos tutoriels. La recherche pour trouver un tutoriel expliquant comment réaliser un inventaire qui me conviendrait à moi et à notre groupe n'était pas très facile non plus. La plupart des tutoriels étaient sous forme de let's play de 20-40 épisodes où il était expliqué comment créer un jeu vidéo de A à Z et même les vidéos dédiées uniquement à la réalisation de l'inventaire avaient parfois des fonctionnalités en trop ou des fonctionnalités manquantes. Afin d'obtenir un inventaire qui

nous conviendrait j'ai suivi les instructions de plusieurs tutoriels tout en ignorant les fonctionnalités en trop et en implémentant celles d'autres tutoriels.

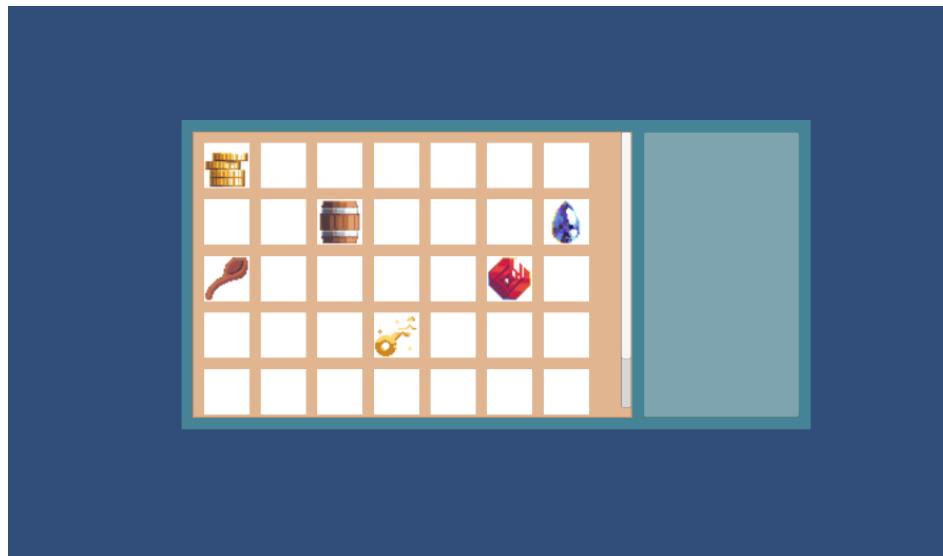


Figure 28: Version 1 de l'inventaire

Pour ce qui est du script C# celui-ci fut assez simples car tous les inventaires fonctionnent plus ou moins selon le même script qui est facilement implantable en regardant une vidéo et les classes de bases d'Unity permettent de rendre l'inventaire fonctionnel en quelques lignes de codes seulement.

Le plus dur fut de ne pas faire tout s'effondrer et de garder l'inventaire fonctionnel après avoir importé des modifications dans Unity. Il m'est arrivé un grand nombre de fois en expérimentant avec Unity ou en essayant de suivre un tutoriel à la lettre de rendre l'inventaire non fonctionnel d'une seconde à l'autre. Ensuite pour ce qui est de retrouver les erreurs et de les corriger cela dure parfois quelques minutes, mais parfois aussi assez longtemps allant des assets qui n'apparaissaient pas parce qu'ils ne sont pas du bon format à l'inventaire qui ne fonctionne pas parce qu'une petite case n'était pas cochée. Les assets temporairement utilisés pour l'inventaire sont des assets libres de droit disponibles gratuitement sur internet sur des sites comme Craftpix.net

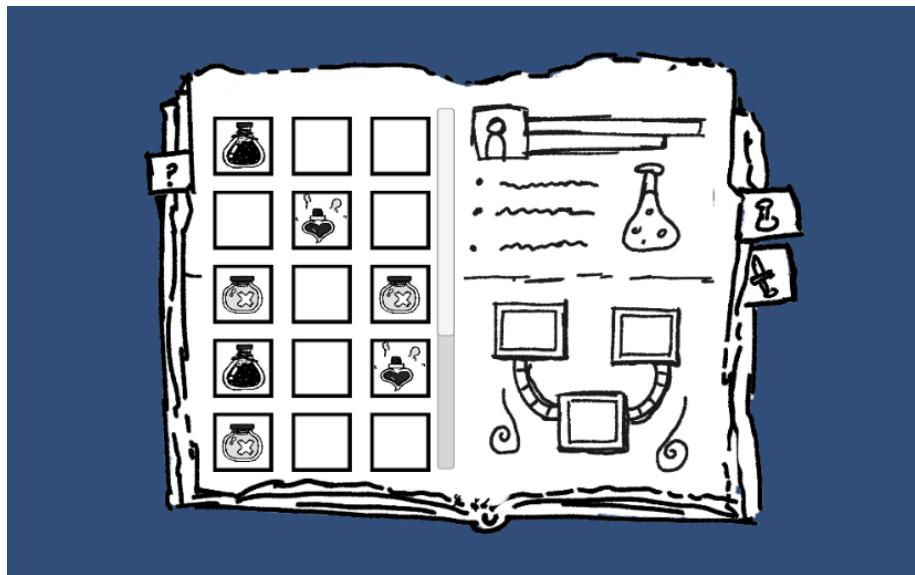


Figure 29: Version 2 de l'inventaire

Pour conclure l'inventaire et le Game design avancent en accord le planning avec la prochaine étape étant d'implémenter un système de craft fonctionnel dans le menu de l'inventaire ainsi que de travailler sur la génération aléatoire du contenu des coffres tout en créant des nouvelles recettes de crafts et idées pour le jeu.

### 3.2.5 Jubair

Dans le cadre de notre projet, chaque membre de l'équipe se voit confier des responsabilités spécifiques pour concrétiser notre vision commune. En tant que responsable technique du joueur et du système d'ennemis avec son intelligence artificielle (IA), ma mission est de créer des mécaniques fluides et captivantes pour garantir une expérience de jeu immersive. Cependant, ce parcours n'a pas été exempt de défis. Dans ce compte rendu, je partagerai mes réalisations jusqu'à présent, les défis auxquels j'ai été confronté et les solutions que nous avons trouvées pour maintenir notre avancée sur la voie du succès.

#### Décembre

J'ai débuté mon implication en mettant en œuvre les mécaniques de déplacement pour le joueur. Cela impliquait de concevoir des commandes intuitives et réactives, permettant au joueur d'explorer l'environnement du jeu de manière fluide et agréable. Cette étape initiale a posé les fondations nécessaires pour la suite du développement.

J'ai également mis en place la caméra du joueur de manière à ce qu'il reste constamment centré à l'écran, garantissant ainsi une expérience fluide et immersive.

Toutefois, mon attention s'est surtout portée sur les ennemis, qui occupent une place centrale dans notre jeu.

#### Janvier-Février

Pour les ennemis, j'ai développé un système complet incluant leur détection par rapport au joueur, leurs comportements de poursuite, leurs mouvements en fonction du joueur, le calcul du trajet le plus rapide et efficace, ainsi que leur stratégie d'encerclement. De plus, j'ai mis en place le système d'attaque des ennemis et leur gestion dans certaine situation. À cet égard, le responsable du design m'a fourni les éléments nécessaires pour les ennemis, ce qui m'a permis de réaliser les animations de déplacements, d'attaque et autre.

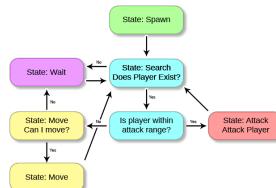


Figure 30: Diagramme des ennemis



Figure 31: Assets des ennemis

Pour progresser dans ces tâches, j'ai régulièrement consulté la documentation Unity ainsi que des tutoriels en ligne sur YouTube. Ces ressources m'ont fourni les connaissances et les techniques nécessaires pour surmonter les obstacles techniques rencontrés et pour optimiser l'efficacité de mes développements.



Pourtant, malgré les avancées réalisées, mon parcours dans ce projet n'a pas été sans défis. La transition vers Unity, bien que stimulante, a nécessité un temps d'adaptation pour maîtriser pleinement les fonctionnalités de cette plateforme de développement. De plus, des problèmes de cohésion au sein de l'équipe ont parfois entravé notre progression, notamment lorsqu'il s'agissait de la fourniture des ressources visuelles par le designer. Cependant, ces défis ont été surmontés grâce à une communication ouverte et à une collaboration étroite entre les membres de l'équipe.

À ce stade, ce projet m'a beaucoup appris et m'a permis de m'épanouir. J'ai eu l'occasion de plonger dans l'univers de Unity, d'explorer ses fonctionnalités et de réaliser un projet excitant. En parallèle, j'ai pu peaufiner mes compétences en C++, en me concentrant sur des aspects spécifiques du langage liés au développement de jeux vidéo. Tout cela m'a vraiment boosté et m'a donné une confiance supplémentaire dans mes capacités.

En somme, mon implication dans ce projet a été marquée par un engagement constant à créer des mécaniques de jeu innovantes et captivantes. Malgré les défis rencontrés en cours de route, je reste déterminé à contribuer au succès de notre projet et à offrir aux joueurs une expérience de jeu mémorable et immersive.

## 4 Récits de réalisations

### 4.1 Recit de réalisations du groupe

Le projet avance en accord avec le diagramme de gant a l'exception de l'interface qui n'est pas encore complètement terminée. Lors de la réalisation de nos tâches nous nous sommes rencontrées a des intervalles d'une à deux fois en présentiel ou en ligne par semaine. Lors de ces réunions nous avons pu progresser sur nos tâches ou bien discuter d'implémentation possible ainsi que de problèmes rencontrés. La plupart de la réalisation s'est déroulé sans ambiguïté a l'exception de 2 cas.

Le premier est dû à un désaccord entre les membres de notre groupe concernant l'implémentation des Boss. Un parti souhaitait enlever l'aspect des Boss obligatoires et revenir sur un battle royal plus traditionnel avec des éléments PVE. Cela rendrait ainsi le côté PVE moins dominant et permettrait aux joueurs de se concentrer sur l'aspect PVP du jeu et de gagner une partie sans être obligé de tuer de Boss. Le parti opposé était en faveur de l'implémentation de Boss obligatoires étant donné que l'idée de Boss avait été discutée plusieurs fois auparavant et que cela donnerait un aspect original au battle royal, qui serait de gagner non en tuant tous les joueurs comme dans un battle royale classique, mais en récoltant tous les artefacts récupéré après avoir éliminé les boss. Après avoir longuement discuté en présentiel afin d'éviter tout malentendu nous sommes parvenus à un compromis. Celui-ci serait de remplacer les boss obligatoires par des totems. Afin de gagner la partie tous les totems devront être détruit et celui qui aura récupéré tous les artefacts lâchées lors de la destruction de ces totems. Ces totems sont gardés par les boss et mettent longtemps à être détruit. 3 choix s'offrent alors aux joueurs. Celui-ci peut vaincre les boss et détruire le totem en sécurité, il peut essayer de détruire le totem sans vaincre les Boss ou bien il peut aller chasser d'autre joueurs qui ont déjà ramassé des artefacts de

totems. Finalement l'implémentation de totem destructible obligatoire a permis de résoudre ce conflit en acceptant la plupart des conditions des deux partis.

Le deuxième grand problème rencontré lors de la réalisation du projet fut un malentendu au niveau des répartitions des tâches. Ce malentendu dû à un manque de communication créa une certaine ambiguïté concernant la charge de l'inventaire. Ce malentendu nous a causé un retard de deux semaines mais fut résolu rapidement. A part ces 2 cas nous n'avons pas rencontré d'autres problèmes majeurs en tant que groupe, le reste fut des problèmes mineurs résolu rapidement tel que les directions artistiques concernant le site ou bien des petits concepts de jeu.

Afin d'éviter de recréer des problèmes similaires nous allons mettre d'autant plus l'accent sur la communication entre les différents membres du groupe. Cela a déjà été mis en place d'une partie notamment grâce la création de canal texte spécifiquement dédiée à la demande d'aide. Finalement le projet a avancé à un stade satisfaisant le diagramme de Gant et nous continuerons à travailler afin de respecter notre emploi du temps donné.

## 4.2 Récit de réalisation personnel

### 4.2.1 Gabriel

Au tout début j'étais enthousiaste à l'idée de pouvoir créer un jeu vidéo. En fait c'était quelque chose auquel je rêvais d'en faire, c'est comme si l'on crée notre propre univers avec des legos. Mais d'un autre côté, cette liberté de pouvoir créer demande un travail soutenu en plus des études, donc l'inquiétude était aussi présent. Cependant, lors de la création d'une d'un univers, il y a aussi création d'histoire. Et il était alors frustrant d'en apprendre que, pendant la concrétisation d'idées, mes camarades avaient des points de vue différentes. Malgré l'insistance, je décidais alors de faire un compromis assez difficile comme tous les autres de mon équipe.

Concernant le développement, j'ai eu peur. Peur de ne jamais pouvoir le finir, surtout lorsque j'ai dû abandonner mes idées, mes motivations n'étaient plus les mêmes. Heureusement que les jours sont passés et vers début décembre on commençait alors à se mettre d'accord sur le contenu du jeu. Et c'est ainsi que je commençais alors à parfois travailler sur Unity pour faire passer de mon temps libre. Chaque découverte sur Unity me fait plus rêver et me donne plus envie à continuer et me voilà déjà réaliser notre multijoueur grâce à Unity.

### 4.2.2 Rania

Le commencement de ce projet n'était pas des plus simple au vu du fait que nous devions utiliser des outils dont nous n'avons jamais entendu parler et que nous n'avons donc jamais utilisé. Les premières semaines du projet étaient donc assez compliquées pour moi car je croyais que je n'allais pas réussir à me servir correctement des outils que nous devions utiliser, seulement je n'ai jamais été quelqu'un qui abandonnait facilement cela a énormément servi durant ce projet pour l'instant. Ce fut aussi très frustrant durant le début lorsque les programmes plantaient lorsqu'on changeait de tutoriel, il a fallu tout apprendre mais c'était assez amusant.

Etant donné que je suis en charge de la communication dans le groupe ça a été réellement frustrant pour moi lorsqu'il y avait des périodes trop chargées pour pouvoir prévoir des réunions car cela nous a causé deux malentendus durant lesquels le projet n'avancait pas beaucoup car les tâches étaient mal repartis au vu du fait que nous n'arrivions jamais à faire des réunions avec le groupe au complet, j'ai dû donc commencer à faire une partie du projet à deux reprises alors que cette même partie était déjà commencé par d'autres personnes du groupe. Heureusement, cette période est à présent terminée et j'espère bien que nous continuons de communiquer comme cela.

En dehors de cela l'entente du groupe est très bonne, je trouve que ce sont des personnes avec qui c'est agréable de travailler et tout le monde est assez sérieux. Je n'ai pas de doutes sur le fait que le rendu final du jeu sera à couper le souffle.

### 4.2.3 Jimmy

Peu de temps Après que le groupe soit formé nous avons vite décider de ce à quoi le jeu va ressembler et quelle direction artistique utiliser j'avais déjà une petite expérience dans le design que ce soit au niveau du développement du front end d'un site web ou plus simplement le dessin de cartoon. C'est tout

naturellement que j'ai décidé de proposer une direction artistique qui suivent mes skills. Après discussion et quelques micro-meetings nous avons décidés de faire un jeu 2D isométrique, soit exactement (a quelques détails près ce que j'avais en tête. C'est quasi-euphorique que j'ai commencer a travailler sur le jeu, effectuer les recherches et les premiers croquis. Ce n'est que bien après dans le développement de l'ethétisme que j'ai encore des difficultés...

Pour comprendre ou le problème est survenu il est important de se rappeler que je travail sur procreate sur iPad. Le problème c'est que lorsque j'utilise la grille fournit par procreate pour construire mes bâtiments qui doivent être en perspective et que je les importe sur la grille de unity il y a un problème de compatibilité. Ce fût un premier gros coup à prendre. J'au eu beau chercher tout les côtés une solution, rien de très concrets n'était proposer. Aujourd'hui, j'ai trouver une solution alternative dans que cela ne me demande tout referaire.

Autre soucis aussi que j'ai rencontrer à avec le groupe cette fois-ci c'est la différence de productivité entre les membres. Je suis chargé de tout dessiner, donc quand quelqu'un souhaite avancer il sur quelques choses en particulier il peut me demander de dessiner ce dont kl besoins. Le problème c'edt que lorsque plusieurs personne me demande en même temps du travail je suis incapable de répondre à toute les attentes dans les délais qui me sont demandés. Certains vont alors mettre en pause total leur travaille et attendre. Or cette démarche est d'être idéal avancer. Il a donc fallu que je m'impose, pour faire comprendre à tout qu'il ne pouvais pas de reposer sur moi.

#### 4.2.4 Nicolas

Créer un jeux-vidéo était un projet que j'envisageait de faire depuis un certain temps. Etant donné que j'ai toujours jouée au jeux vidéo depuis mon plus jeune âge, le processus d'en créer un m'a toujours intéressé. En commençant à travailler sur le projet j'ai vite remarqué que la tâche de créer un jeu-vidéo allait s'avérer être beaucoup plus dur que ce que j'envisageait. Rien que pour prendre en main Unity ou essayer de trouver les bonnes ressources s'est avérée assez frustrant, car cela prenait beaucoup de temps pour peu d'avancement. Cela était aussi le cas quand il s'agissait de résoudre les problèmes sur Unity. Malgré cela le fait de voir son programme fonctionner après avoir travaillé pendant des heures procure une grande satisfaction ainsi que de la motivation pour continuer à travailler, car cela donne une impression que le projet se concrétise. Le groupe me semble être plutôt passionnée sur l'idée de créer un jeu avec l'univers que nous avons créé de A à Z, le seul problème reste parfois le manque de communication entre les membres, conduisant à des malentendus ou rendant le savoir concernant la progression des membres assez vague. J'espère que nous pourrons éviter de répéter cette erreur étant donné que nous en avons tous pris conscience que ce soit individuellement ou en tant que groupe.

#### 4.2.5 Jubair

Ce projet représente pour moi bien plus qu'une simple opportunité d'améliorer mes compétences. C'est une aventure passionnante où je découvre non seulement comment gérer un projet en équipe, mais aussi comment travailler avec des personnes que je ne connaissais pas auparavant. C'est un peu comme plonger dans le monde professionnel, où la collaboration et le soutien mutuel sont essentiels.

Jusqu'à présent, cette expérience m'a enseigné énormément de choses. J'ai pu mettre en pratique ce que je savais déjà, mais j'ai également réalisé qu'il est parfois judicieux de demander de l'aide lorsque les choses deviennent complexes. Au départ, j'avais quelques appréhensions, mais finalement, je me retrouve à la fois confronté à des défis stimulants et à des moments de pur plaisir. C'est cette combinaison d'efforts intenses et de gratifications qui rend mon investissement dans ce projet si gratifiant.

## 5 Les annexes

Concepts art de la map (disposés dans l'ordre chronologique de leur réalisation)

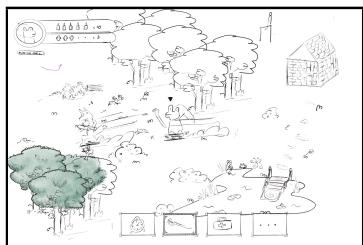


Figure 32: première idée visuelle du jeu

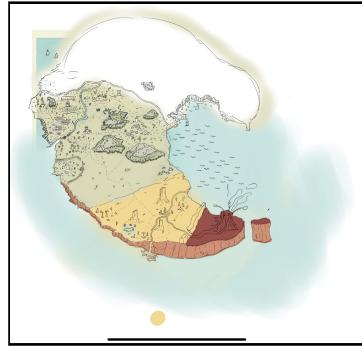


Figure 33: répartition de zone de la map

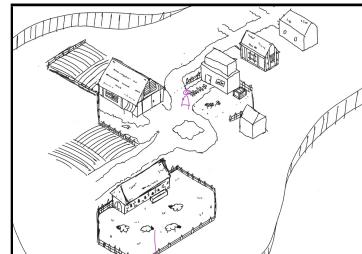


Figure 34: premier brouillon de la création d'un village

### Développement des personnages et de leur fonctionnalité

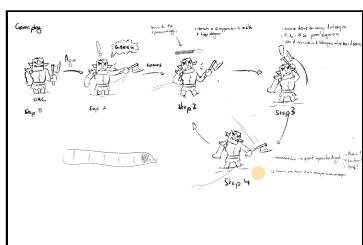


Figure 35: pattern de base de l'orc et caractéristiques

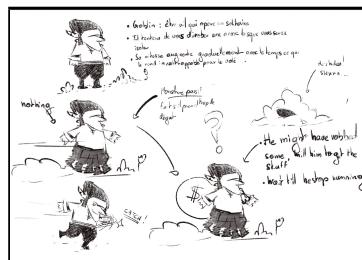


Figure 36: pattern de base du globelin et fonctionnalité

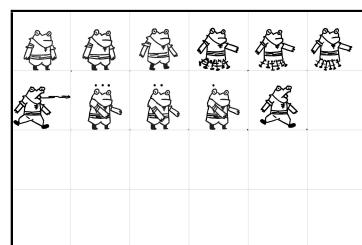


Figure 37: sprite du personnage grenouille

### assets de bases

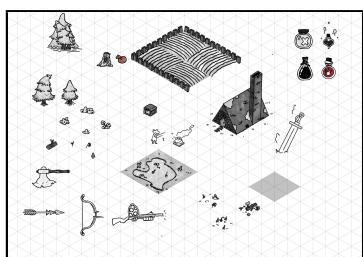


Figure 38: toile de création des assets

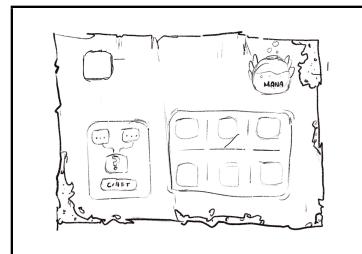


Figure 39: asset d'inventaire