RAPPORT DE SOUTENANCE 1

TREASHUNT

de IBOREN™

Paul VOGELEISEN

Cyril KOHLER

Léo SIBOUR

Baptiste MARX

Mars 2024

Table des matières

1 Introduction

Ce rapport vise à présenter de manière détaillée les progrès accomplis dans le développement de notre jeu vidéo TreasHunt, un hard-plateformer 3D au style graphique low poly. Ce projet incarne l'essence de l'aventure, du défi et de l'exploration dans un monde en 3D.

Nous débuterons par un retour sur l'état de l'art, mettant en lumière les références et les influences qui nous ont inspirées. Suite au départ d'un membre, nous présenterons brièvement la nouvelle équipe chargée du développement de TreasHunt.

Nous exposerons également les objectifs que nous nous sommes fixés pour cette soutenance. Enfin, nous détaillerons la conception du projet en suivant l'avancement des tâches attribuées à chaque membre de l'équipe.

1.1 Notre projet

TreasHunt a pour objectif principal de procurer une expérience de jeu captivante, adaptée à un large public. En combinant divers modes de difficultés, avec une immersion narrative modeste à travers un mode histoire, nous cherchons à susciter l'intérêt des joueurs de tous horizons.

Le joueur endosse le rôle d'un courageux aventurier en quête d'un trésor légendaire. Sa mission : réussir cette quête principale en revenant sain et sauf avec le trésor tant convoité. Mais son chemin sera périlleux et il devra faire face à des parcours insolites semés d'embûches, mettant ainsi à l'épreuve ses compétences...

Cependant, TreasHunt ne se limite pas qu'à l'expérience solo. En effet, à la demande du client, nous avons inclus un mode multijoueur : les joueurs s'affrontent sur les parcours des chapitres du mode histoire et le premier arrivé l'emporte. Ce mode ajoute une dimension compétitive et sociale au jeu, renforçant ainsi son attrait pour un public varié.

TreasHunt encourage le développement du sens logique des joueurs. Dans un univers riche de défis, ils devront constamment faire preuve de réflexion pour emprunter le meilleur chemin et surmonter des obstacles pour avancer.

1.2 Etat de l'art

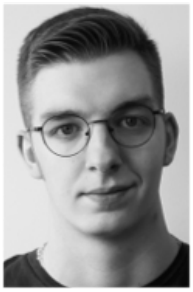
Le jeu que nous développons appartient au genre des jeux d'action-aventure, plus particulièrement à la catégorie des jeux de plateformes dits *hard-platformer*. L'émergence de ce genre remonte aux débuts des jeux vidéo, se concrétisant en 1990 avec le jeu *Alpha Waves*, pionnier en la matière. Ce titre a défini les bases de défis intellectuels et physiques complexes intégrés au gameplay.

Depuis lors, les *hard-platformer* ont évolué pour donner naissance à des titres renommés tels qu'AltF4 (2018), Only Up (2015) et Fall Flat (2016) dont nous nous sommes inspirés.

À l'heure où les jeux d'exploration 3D ont connu une évolution significative tel que la série Uncharted, notre jeu, TreasHunt, se distingue par sa fusion d'éléments d'action, d'énigmes et de diversités environnementales. Ses principes fondamentaux incluent une immersion totale du joueur dans l'exploration, une mécanique de plateforme exigeante (le joueur doit escalader et surmonter divers obstacles), des énigmes stimulantes, des combats occasionnels, et enfin, un style graphique *low poly*, qui confère au jeu un look distinctif tout en optimisant ses performances.

1.3 Notre équipe

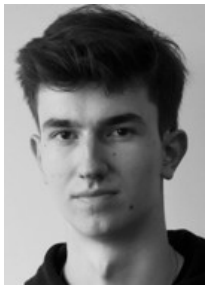
Suite au départ de Benjamin Mathieu, nous sommes désormais une équipe de quatre membres engagés dans le développement de TreasHunt. Face à cette nouvelle organisation, nous avons dû réajuster nos stratégies et répartir les tâches en conséquence. Nous détaillerons cette nouvelle répartition des responsabilités dans la partie qui suit.



Baptiste Marx
Directeur Artistique

Léo Sibour
Game/Web Designer

Cyril Kohler
Directeur Technique



Paul Vogeleisen
Chef de Projet

FIGURE 1 – Membres d'IBOREN

1.4 Attendus

	Paul	Cyril	Léo	Baptiste
Programmation				
Multijoueur	S	R		
Mécanique du jeu				
Déplacement du joueur	R	S		
Gestion inventaire Interface			R	S
Interractions environnement		R	S	
Intelligence Artificielle				
Pièges et défenses		R	S	
Apparition des ennemis	R	S		
Game Design				
Conception des maps				
Lobby : plage			S	R
Chapitre 1 : fôret			R	S
Chapitre 2 : montagnes			S	R
Chapitre 3 : volcan			S	R
Environnement				
Scénario et narration	S			R
Musiques et sons	R		S	
Gestion des assets			S	R
Site Web				
Page d'accueil			R	S
Section téléchargement			R	S
Section à propos			R	S

TABLE 1 – Tableau récapitulatif : répartition des tâches

R : Responsable | **S** : Suppléant

Soutenance	***Mars***	***Juin***
Programmation		
Multijoueur	**100%**	**100%**
Mécanique du jeu		
Déplacement du joueur	**100%**	**100%**
Gestion inventaire Interface	25%	**100%**
Interractions environnement	25%	**100%**
Intelligence Artificielle		
Pièges et défenses	0%	**100%**
Apparition des enemis	0%	**100%**
Game Design		
Conception des maps		
Lobby : plage	25%	**100%**
Chapitre 1 : fôret	0%	**100%**
Chapitre 2 : montagnes	0%	**100%**
Chapitre 3 : volcan	0%	**100%**
Environnement		
Scénario et narration	50%	**100%**
Musiques et sons	0%	**100%**
Site Web		
Page d'accueil	60%	**100%**
Section téléchargement	60%	**100%**
Section à propos	60%	**100%**

TABLE 2 – Planning détaillé : avancement des tâches

X% : pourcentage d'avancement

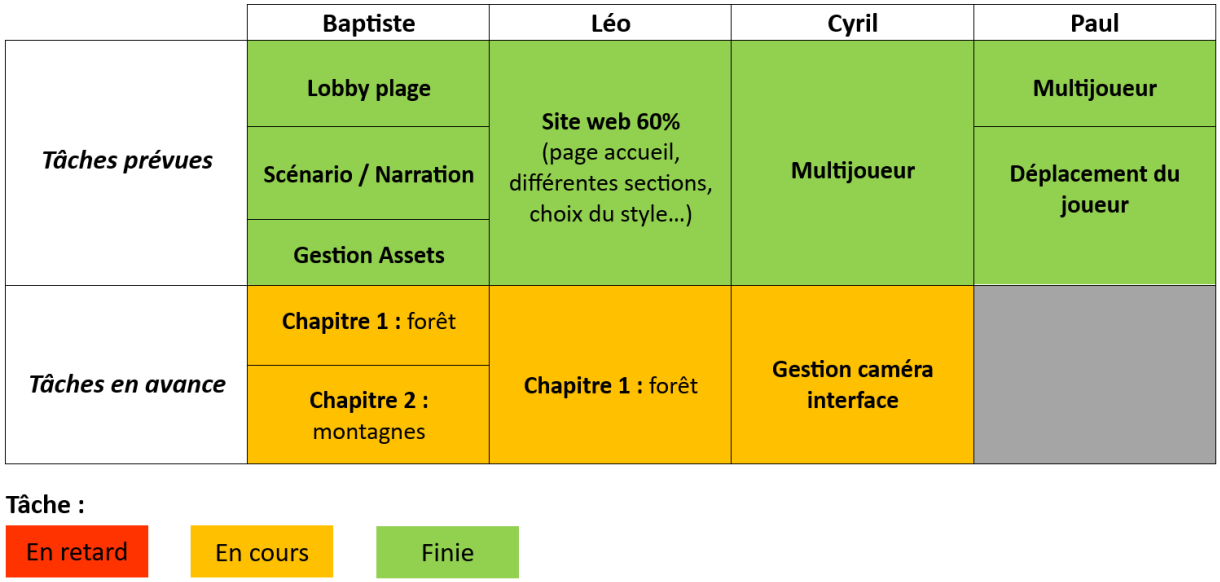


FIGURE 2 – Réalisation des tâches : soutenance technique 1

Pour cette première soutenance technique, notre objectif principal était de mettre en place un mode multijoueur fonctionnel. Nous avons choisi de prioriser ce mode car il constitue une base solide sur laquelle nous pouvons ensuite intégrer les différentes mécaniques de jeu ainsi que l'environnement tel que les parcours et les maps.

Cyril et Paul ont donc commencé par développer ce mode, travaillant chacun de leur côté. Cette approche nous a permis de comparer les différentes versions et de sélectionner celle qui convient le mieux tout en garantissant de meilleures performances. De plus, ce processus de développement conjoint nous a offert l'opportunité d'échanger nos connaissances et d'améliorer ainsi le produit final de manière significative.

En parallèle, Léo s'est concentré sur le développement de notre site web, tandis que Baptiste s'est chargé de la gestion des assets (dont nous avons acheté le pack sur l'Unity Store) en élaborant les différentes maps et parcours du jeu ainsi que la structuration narrative le tout en étroite collaboration avec Léo.

Suite au départ de Benjamin, nous avons dû revoir la répartition des tâches au sein de l'équipe, comme vous pouvez le constater.

Nous aborderons en détail la conception et l'avancement de ces tâches dans la partie suivante du rapport.

2 Conception du projet

Dans cette section, vous trouverez toutes les informations concernant la réalisation et l'état d'avancement des tâches de chacun du développement de TreasHunt.

2.1 Baptiste - Directeur Artistique

2.1.1 Gestion des assets

En tant que responsable de la gestion des assets, une de mes priorités était de m'assurer que chaque élément visuel utilisé dans notre jeu était approprié à l'environnement du chapitre correspondant. Cette tâche, bien que souvent négligée, était essentielle pour maintenir l'immersion des joueurs et la cohérence globale de notre création.

J'ai donc entrepris de veiller attentivement à ce que les assets sélectionnés s'intègrent harmonieusement dans chaque scène, évitant ainsi des incohérences flagrantes. Rien n'est plus déroutant pour un joueur que de se retrouver face à des palmiers qui poussent mystérieusement au sommet d'un rocher ou dans des environnements où les éléments semblent déplacés. Mon travail consistait donc à anticiper ces situations en choisissant judicieusement les assets et en veillant à leur agencement cohérent dans chaque contexte.

2.1.2 Level design : lobby - chapitre 1 - chapitre 2

Le lobby du jeu s'est vu agencé comme une plage délimitée par des récifs de part et d'autre de ses extrémités. Cela permet de proposer aux joueurs un espace vaste mais limité dès leurs premiers pas dans l'aventure. Le bateau avec lequel l'aventurier a débarqué sur l'île est bloqué sur des rochers, n'offrant au joueur que la possibilité de continuer à travers la forêt.

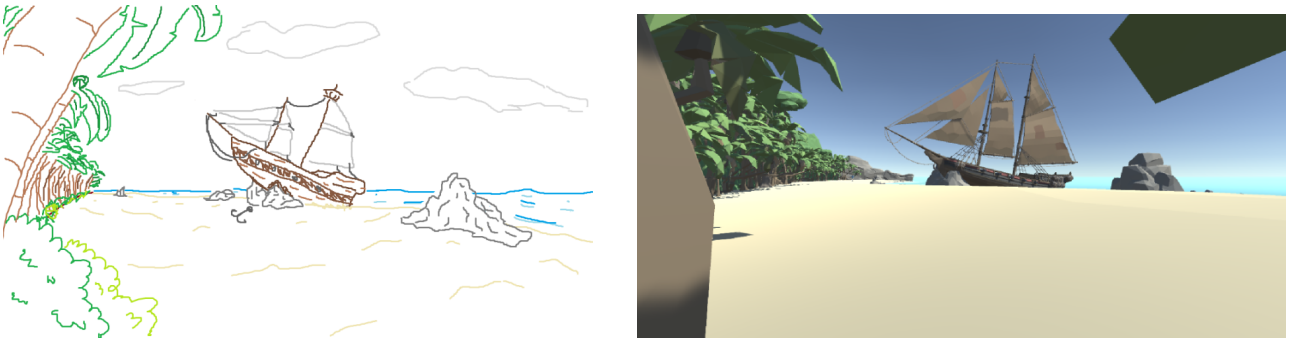


FIGURE 3 – croquis et design du lobby

Pour guider les joueurs dans le premier chapitre, un sentier en terre battue se dessine, ce qui permet de suivre un fil conducteur et une progression fluide en évitant ainsi toute confusion.

Cependant, comme toute exploration en milieu sauvage, le sentier n'est pas sans ses défis. Les embûches naturelles de la forêt viennent ponctuer le parcours, testant ainsi les compétences du joueur, tout en lui offrant une chance de s'améliorer avant de se lancer dans les niveaux suivants.

Après avoir bravé les épreuves de la forêt, le joueur passe au chapitre suivant, symboliquement plus haut dans son périple. Le décor change radicalement, laissant place à des montagnes majestueuses. Ce changement de paysage marque une évolution dans l'aventure du joueur, signifiant qu'il a déjà parcouru une distance considérable.

Dans ce chapitre, le sentier devient plus escarpé, plus difficile à gravir. Les fragments de chemin se succèdent, défiant l'équilibre et la ténacité du joueur, lui offrant ainsi une expérience de jeu toujours plus immersive et stimulante.

Chaque pas dans cette montagne représente un défi supplémentaire, mais aussi une occasion pour le joueur de repousser ses propres limites et de se rapprocher un peu plus de son objectif.

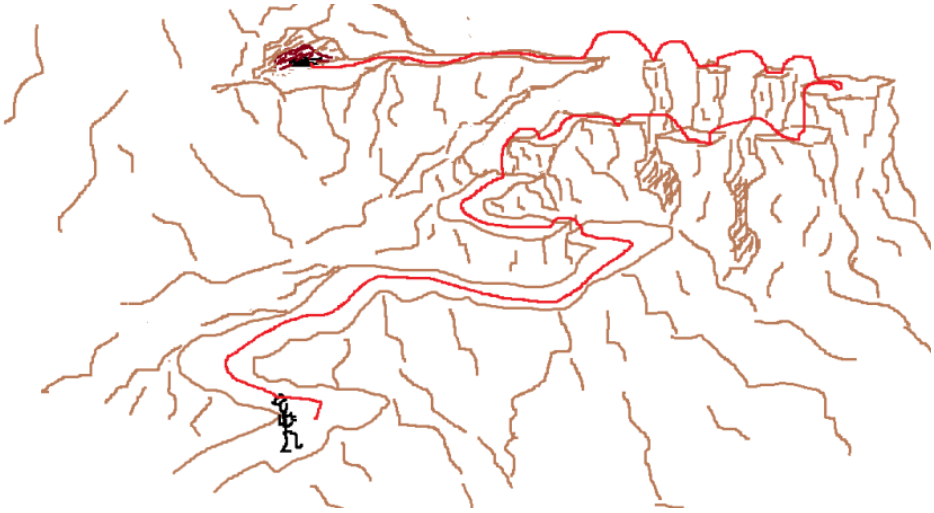


FIGURE 4 – croquis non exhaustif parcours chapitre 2

2.1.3 Scénario / narration

Dans TreasHunt, le joueur incarne un explorateur intrépide se lançant dans une quête à travers trois environnements distincts : la forêt, la chaîne montagneuse et le volcan. Chaque chapitre représente une étape de son périple, avec des défis uniques à surmonter et des découvertes à faire.

Dans le premier chapitre, notre héros pénètre dans l'épaisse forêt, où il doit naviguer à travers des sentiers sinueux et éviter des pièges naturels. Le deuxième chapitre le conduit à travers une chaîne montagneuse escarpée, où il affronte des crevasses glacées et des chemins escarpés. Grâce à son expérience acquise dans la forêt, il réussit à surmonter les obstacles montagneux.

Enfin, le héros atteint le volcan, un territoire hostile où la lave bouillonnante et les geysers explosifs représentent des dangers ultimes. Utilisant ses compétences, il progresse à travers ce paysage chaotique. Là, il trouve un grand trésor enfoui dans le cœur du volcan, la récompense tant convoitée.

2.1.4 Prochaine soutenance

Pour la prochaine soutenance, je peaufinerai la map du lobby en y ajoutant des détails comme des rochers supplémentaires, des îles à l'horizon, etc. Également, j'achèverai la map de la forêt en y ajustant l'emplacement des obstacles et éléments de l'environnement, sans oublier de réaliser les terrains des chapitres 2 et 3. Je veillerai également à la bonne gestion des assets tout au long des prochains mois.

2.2 Léo - Game/Web Designer

2.2.1 Site Web : agencement et style

Pour débuter, je me suis plongé dans des ressources en ligne pour comprendre en détail le fonctionnement des classes et des ID en HTML, ainsi que celui de Sublime Text, mon éditeur de code. Heureusement, la prise en main de l'outil s'est avérée rapide.

Une fois ces bases assimilées pendant les vacances de janvier, j'ai immédiatement entrepris de coder la barre de navigation du site composé de la page d'accueil, téléchargement et à propos, telle que je l'avais imaginée lors de la rédaction du cahier des charges.

Dans l'ordre des choses je me suis focalisé sur la page d'accueil, la section « à propos » et la section téléchargement.

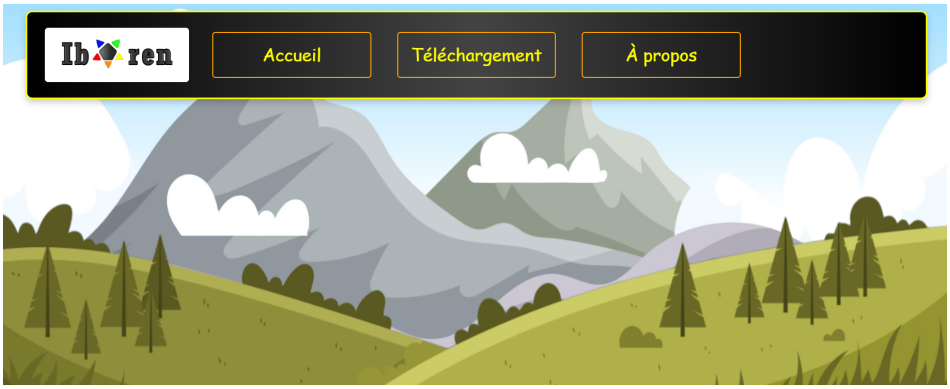


FIGURE 5 – Page d'acccueil

Une fois le premier jalon atteint, je me suis tourné vers YouTube et le site officiel de GitHub pour apprendre comment héberger mon site web gratuitement avec GitHub. Après avoir créé une clé SSH, je l'ai ajoutée au dépôt "iboren" (celui de mon site web) et j'ai ensuite mis en ligne le site. Chaque push effectué actualise ainsi la version en ligne.

Ensuite, j'ai approfondi mes recherches pour intégrer des animations, des GIF et d'autres propriétés visant à rendre le site plus dynamique. J'ai également exploré le JavaScript pour créer un curseur de souris personnalisé et implémenter des effets de parallaxe durant les week-ends lorsque j'avais du temps libre.

C'est un effet optique où les objets proches semblent se déplacer plus rapidement que les objets éloignés, créant ainsi une sensation de profondeur sur un site web.

```
@keyframes race-bulle {
    from {
        left: 9%;
    }
    50% {
        left: 55%;
    }
    51% {
    }
    to {
        left: 9%;
    }
}
```

FIGURE 6 – code HTML/CSS animation avec keyframes

J'ai consulté différentes vidéos pour comprendre la logique sous-jacente au parallaxe, ce qui m'a grandement intéressé.

Je n'ai pas particulièrement eu recours à des bibliothèques pour la création d'effets dynamiques ; la plupart ont été conçus en CSS. Pour mes tests, j'ai souvent travaillé sur une page blanche nommée "test", ou encore sur le site JSFiddle : https ://jsfiddle.net.

2.2.2 Level design : chapitre 1

Pour commencer, j'ai importé les assets dans le logiciel Unity3D en utilisant Git Bash. Ensuite, j'ai créé les premiers éléments du parcours forestier peu de temps après l'AFIT, puis j'ai ajusté leur taille pour façonner la forme du parcours.

Cette forme slalomera entre des montages tout en gagnant en altitude petit à petit afin de faire transition au prochain chapitre montagne.

Pour enrichir le début du parcours, j'ai intégré davantage de détails, tel qu'un sentier agrémenté d'objets représentatifs de l'environnement. En parallèle, j'ai configuré un personnage capable de se déplacer et une caméra sans recourir à la gestion de paquets.

Ce personnage, bien que temporaire, me permet d'évaluer si la disposition des objets entrave le parcours, notamment en raison de leurs boîtes de collision. De plus, il me permet de tester la faisabilité de certains sauts.

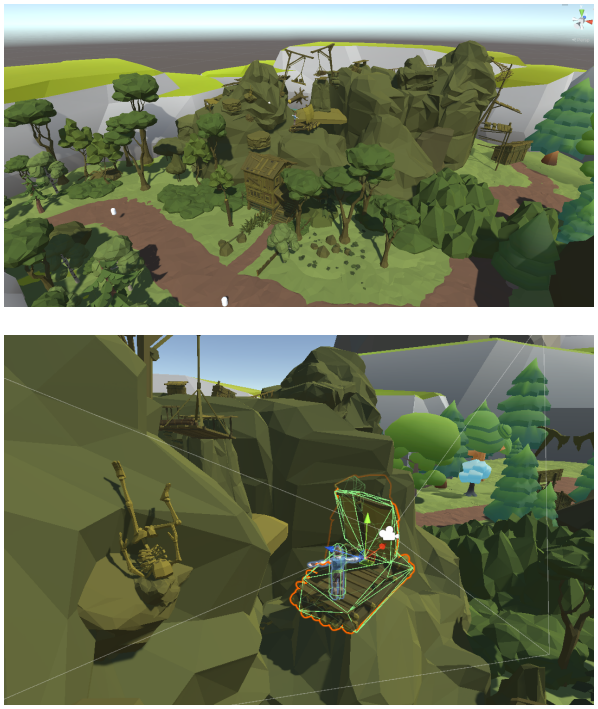


FIGURE 7 – extrait environnement chapitre 1 - test personnage

En termes de conception, je me suis largement inspiré de la dynamique de jeu d'Only-Up pour agencer les objets du quotidien formant le parcours. J'ai exploité toutes les formes d'objets disponibles pour faciliter la création. J'ai également optimisé l'utilisation de l'espace de la carte afin que le personnage puisse explorer tous les recoins de Treashunt sans en manquer.

J'ai ajusté les échelles, les boîtes de collision et la position/orientation des objets à l'aide d'Unity pour diversifier l'environnement (certains objets sont similaires mais de tailles et d'inclinaisons différentes). Les retours des membres de l'équipe m'ont également aidé à stimuler ma créativité.

2.2.3 Prochaine soutenance

Pour les prochains mois, je prévois d'ajouter de petits détails aux platesformes du parcours. Il pourrait s'agir de vases, de livres ou d'autres éléments décoratifs qui enrichissent l'environnement sans impacter le parcours. Je compte également achever les trois quarts restants du parcours en forêt en suivant le modèle établi. Enfin, je terminerai le développement du site web ainsi que l'élaboration de l'inventaire du joueur.

2.3 Cyril - Directeur Technique

2.3.1 Multijoueur : test, problèmes rencontrés et résolution

Ma principale responsabilité dans le projet a été de mettre en œuvre le mode multijoueur, en utilisant la librairie NetCode de Unity. Cette librairie s'est avérée extrêmement pratique à utiliser, nécessitant peu de code mais une configuration substantielle. Par exemple, elle fournit des scripts C# qui gèrent la synchronisation des positions et des rotations des joueurs.

Cependant, plusieurs défis se sont présentés lors du développement de cette fonctionnalité, principalement liés à la connexion entre deux ordinateurs distincts sur un même réseau. Deux problèmes notables sont survenus.

Tout d'abord, la décision récente de NetCode de ne pas autoriser les connexions provenant d'autres machines par défaut a rendu de nombreuses ressources en ligne obsolètes.

Après avoir correctement configuré NetCode pour accepter ces connexions entrantes, un autre problème majeur est survenu. Chaque ordinateur est équipé d'un pare-feu qui bloque la plupart des connexions entrantes, y compris la connexion nécessaire pour notre jeu. Pour résoudre ce problème, il a été nécessaire d'ouvrir manuellement le port utilisé par notre jeu (le port 7777 par défaut pour NetCode).

```
public class NetworkManagerUI : MonoBehaviour
{
    [SerializeField] private Button serverBtn;
    [SerializeField] private Button hostBtn;
    [SerializeField] private Button clientBtn;

    private void Awake() {
        serverBtn.onClick.AddListener(() => { NetworkManager.Singleton.StartServer(); });
        hostBtn.onClick.AddListener(() => { NetworkManager.Singleton.StartHost(); });
        clientBtn.onClick.AddListener(() => { NetworkManager.Singleton.StartClient(); });
    }
}
```

FIGURE 8 – Code C# serveur-client

2.3.2 Gestion caméra

Après avoir fini la partie multijoueur, j'ai concentré mes efforts sur le développement du personnage jouable, en commençant par la mise en place d'un système de caméra à la troisième personne qui suit le joueur.

La création d'une telle solution peut présenter des défis techniques, c'est pourquoi j'ai décidé d'utiliser le module Cinemachine fourni par Unity, reconnu pour son système de caméra avancé et hautement configurable.

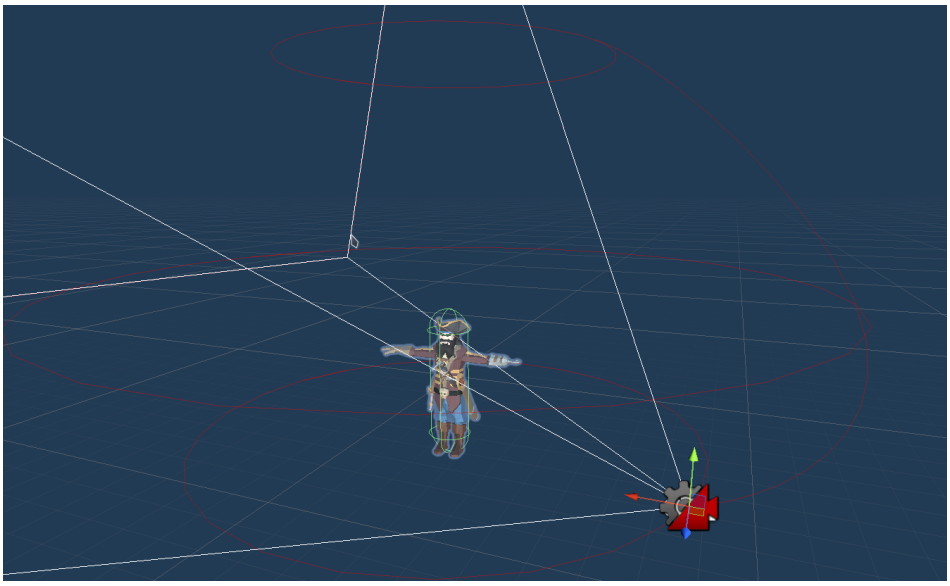


FIGURE 9 – Réglages caméra avec Cinemachine

Cinemachine est un outil puissant inclus dans Unity qui simplifie grandement le processus de mise en place de caméras dynamiques dans les jeux. Il offre une variété d'options pour contrôler le mouvement, l'orientation et la composition de la caméra, permettant aux développeurs de créer des expériences visuelles immersives et cinématographiques.

En utilisant Cinemachine, j'ai pu configurer facilement une caméra à la troisième personne qui suit le personnage jouable, offrant ainsi une perspective dynamique et fluide tout en évitant les obstacles et en s'ajustant automatiquement aux mouvements du joueur. Cela a grandement amélioré la qualité visuelle et l'expérience de jeu globale.

2.3.3 Déplacement et animation

Dans la dernière phase de mon travail, j'ai concentré mes efforts sur le développement des déplacements du joueur ainsi que sur son animation afin de rendre l'expérience de jeu plus immersive et captivante.

L'objectif était de permettre au personnage de réagir de manière réaliste aux actions du joueur, ce qui contribue à renforcer l'immersion dans le jeu. Pour gérer les déplacements du joueur, j'ai opté pour une approche simple mais efficace en utilisant le composant Rigidbody de Unity, qui permet d'interagir avec la physique du monde du jeu. La logique principale du code repose sur quelques étapes clés :

```
private void Update() {
    if (!IsOwner) return;

    // ground check
    grounded = Physics.Raycast(transform.position, Vector3.down, playerHeight * 0.5f + 0.2f, groundLayer);

    MyInput();
    SpeedControl();
    StateHandler();
    AnimeHandler();

    // handle drag
    if (state == MovementState.walking || state == MovementState.sprinting) rb.drag = groundDrag;
    else rb.drag = 0;
}

private void FixedUpdate() {
    if (!IsOwner) return;

    MovePlayer();
}
```

FIGURE 10 – Logique de déplacements du joueur

Tout d'abord, nous effectuons un test pour déterminer si le joueur est au sol, ce qui est essentiel pour savoir s'il peut sauter ou non. Ensuite, nous récupérons les entrées du joueur (les touches qu'il presse) pour calculer les forces qui doivent être appliquées au joueur.

Les deux prochaines fonctions se concentrent sur la mise à jour de l'état du joueur. La fonction StateHandler détermine l'état de déplacement du joueur, qui peut être soit "marche", "course" ou "en l'air". Ensuite, la fonction AnimeHandler applique la bonne animation au joueur en fonction de son état.

Cependant, une grande partie de la logique derrière les animations est réalisée grâce au diagramme d'animation de Unity, ce qui simplifie considérablement le processus de gestion des animations et permet une transition fluide entre les différentes actions du joueur.

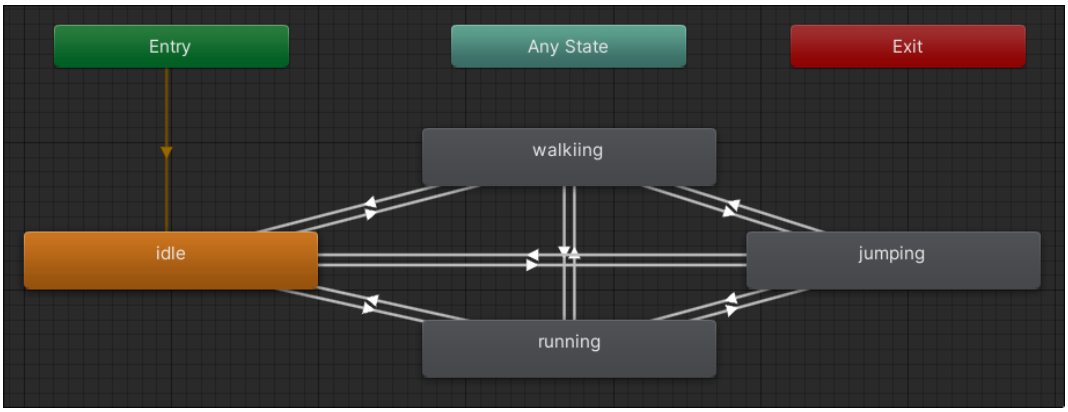


FIGURE 11 – Diagramme d'animation

Enfin, la force de mouvement calculée précédemment est appliquée au joueur, lui permettant de se déplacer de manière réaliste dans le monde du jeu.

Dans le cadre du développement des animations, nous avons opté pour l'utilisation du site Mixamo, qui offre une vaste bibliothèque d'animations gratuites et libres de droits. Cette décision s'est révélée être un choix judicieux pour notre projet, surtout compte tenu du fait qu'aucun membre du groupe n'avait d'expérience préalable en animation.

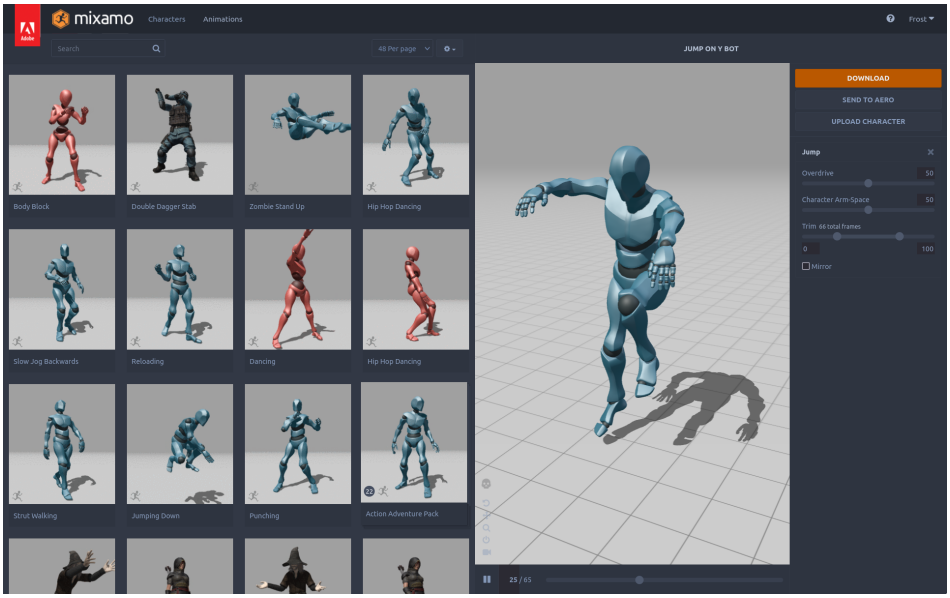


FIGURE 12 – Bibliothèque d'animation Mixamo

Mixamo propose une gamme diversifiée d'animations prêtes à l'emploi, couvrant diverses actions telles que la marche, la course, le saut, les attaques, et bien plus encore. La plateforme offre également la possibilité de personnaliser les animations en ajustant les paramètres tels que la vitesse, la direction, et le style.

L'utilisation de Mixamo nous a permis de gagner un temps précieux et d'éviter la complexité associée à la création d'animations à partir de zéro. En quelques clics, nous avons pu trouver et télécharger les animations appropriées pour notre personnage joueur, et les intégrer facilement dans notre jeux.

2.3.4 Prochaine soutenance

Pour la prochaine soutenance, mes efforts seront principalement concentrés sur l'intégration de l'intelligence artificielle (IA) dans notre jeu, notamment sous forme d'ennemis et de pièges conçus pour entraver la progression du joueur. L'objectif est de créer des défis dynamiques et stimulants, tout en ajoutant une dimension stratégique au gameplay.

2.4 Paul - Chef de Projet

2.4.1 Déplacement du joueur

La mise en place des déplacements du joueur dans notre jeu s'est déroulée sans difficultés majeures. J'ai rapidement implémenté les contrôles classiques au clavier, me permettant de me déplacer dans l'environnement en utilisant les touches ZQSD pour avancer, reculer et me déplacer latéralement, ainsi que la touche Espace pour sauter. L'un des principaux défis a été de gérer les collisions avec l'environnement et d'appliquer correctement la gravité pour assurer un mouvement fluide et réaliste. Heureusement, après de nombreux essais, ça a fini par fonctionner !

Actuellement, mes mouvements du joueur sont assez basiques, car ma priorité était de mettre en place le mode multijoueur. Cependant, nous allons opter pour les déplacements développés par Cyril car ils sont meilleurs. Nous devons toutefois les améliorer en les testant sur les différents parcours du jeu afin de déterminer les paramètres qui conviennent le mieux, assurant ainsi un gameplay fluide et engageant pour les joueurs. Je fais notamment référence à la sensibilité, la vitesse et fluidité des mouvements.

```
void Update()
{
    // deplacement
    if (!IsOwner) return; // evite que les joueurs clients non owner de l'objet modifie les données
    float horizontalInput = Input.GetAxis("Horizontal");
    float verticalInput = Input.GetAxis("Vertical");

    Vector3 movementDirection = new Vector3(horizontalInput, 0, verticalInput);
    movementDirection.Normalize();

    transform.Translate(movementDirection * (movementSpeed * Time.deltaTime), Space.World);

    // rotation orientation joueur
    if (movementDirection != Vector3.zero)
    {
        Quaternion toRotation = Quaternion.LookRotation(movementDirection, Vector3.up);
        transform.rotation =
            Quaternion.RotateTowards(transform.rotation, toRotation, rotationSpeed * Time.deltaTime);
    }

    // Saut
    if (Input.GetButtonDown("Jump"))
    {
        Jump();
    }
}

private void Jump() // fonction saut
{
    if (rb != null)
    {
        rb.AddForce(Vector3.up * jumpForce, ForceMode.Impulse);
    }
}
```

FIGURE 13 – Code C# Déplacement basique du joueur

2.4.2 Multijoueur : NetCode et rôles serveur client

Nous voila maintenant à une des parties cruciales du projet : le multijoueur. Pour développer cette partie de notre jeu, j'ai consacré environ 6 à 8 heures de travail. En effet, certaines choses ne se sont pas exactement passées comme prévu...

Dans la nuit du 8 au 9 février 2024, j'ai pu pour la première fois, faire connaissance avec le git merge conflict. Pour faire simple, j'avais mal géré mes commits et je me suis donc retrouvé avec de sacrés conflits. Voulant revenir au dernier commit que j'avais push, j'ai perdu tout mon travail. J'ai donc tout repris à 0 sur de bonnes bases et je serai plus vigilant pour éviter de telles erreurs à l'avenir.

Pour apprendre à mettre en place le multijoueur, j'ai suivi un tutoriel d'une heure sur YouTube en anglais. J'ai utilisé Netcode for GameObject, une technologie développée par Unity qui simplifie la création de jeux multijoueurs en réseau. Cette solution permet de synchroniser efficacement les objets de jeu entre les différents clients et le serveur, garantissant ainsi une expérience de jeu fluide et sans latence.

Concernant l'infrastructure réseau, le multijoueur utilise un réseau LAN (Local Area Network), ce qui signifie qu'il fonctionne sur un réseau local plutôt que sur Internet. Dans notre configuration, le joueur principal assume le rôle de serveur tandis que les autres joueurs qui le rejoignent agissent en tant que clients.

Voici la partie de code correspondante qui permet de lancer un joueur ayant le rôle du serveur (host) et un autre ayant le rôle du client à partir de 2 boutons.

```
16      private void Awake() // buttons actions
17      {
18          hostButton.onClick.AddListener(() =>
19          {
20              NetworkManager.Singleton.StartHost();
21          });
22
23          clientButton.onClick.AddListener(() =>
24          {
25              NetworkManager.Singleton.StartClient();
26          });
27      }
```

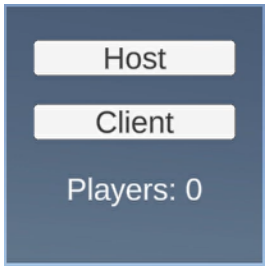


FIGURE 14 – Code C# serveur-client et rendu graphique

2.4.3 Prochaine soutenance

En vue de la prochaine et dernière soutenance en juin, c'est assez simple : il faut finir le jeu ! Pour ma part, je devrai donc améliorer les déplacements du joueur comme évoqué avant, développer avec Cyril les ennemis qui s'opposeront au joueur lors de son aventure (donc IA) et également m'occuper de tout le Sound Design du jeu puisque Benjamin n'est plus parmi nous. Le tout en continuant d'assurer mon rôle en tant que chef de projet dans la réussite de l'élaboration de TreasHunt.

3 Conclusion

En conclusion, nous sommes satisfaits de l'avancement de TreasHunt. Malgré les défis rencontrés, tels que le départ inattendu d'un membre de notre équipe, nous avons réussi à maintenir le cap et même à dépasser certains des objectifs que nous nous étions fixés en termes de délais pour cette étape importante du projet.

En effet, nous avons un mode multijoueur opérationel, le game et level design du lobby et du chapitre 1, une implémentation des déplacements et animations des mouvements du joueur, une bonne gestion de la caméra lorsque le joueur se déplace, une sélection d'assets précise, le scénario et la narration du mode histoire ainsi qu'un site web.

Pour la prochaine soutenance, nous améliorerons les fonctionnalités du mode multijoueur, développerons les maps et parcours du chapitre 2 et 3 (montagnes et volcan), nous optimiserons les mécaniques de mouvements du joueur afin de garantir une expérience de jeu la plus agréable possible, nous implémenterons de l'IA à travers des ennemis et des pièges, intégrerons le Sound Design unique de TreasHunt, nous développerons également l'inventaire du joueur ainsi qu'un HUD incluant une barre de vie et d'endurance et enfin, nous finirons le développement du site web.

Ayant chacun trouvé notre rythme, nous sommes déterminés à mettre en œuvre tous les efforts nécessaires pour mener à bien la finalisation de notre jeu.