Melody : Chronicle of the Knight

by Nou, soutenance 2.

2023-2024

Sommaire

1 Préambule

Depuis les prémices de ce projet, l'équipe a toujours eu une vision précise de la finalité et des axes à approfondir afin de l'atteindre. Ainsi, ce rapport agit en indicateur de notre avancement, mais aussi de ce qu'il nous reste à réaliser. Celui-ci reprend dans un premier temps rapidement les éléments du projet détaillés lors de la première soutenance en y apportant des précisions et éventuelles corrections. Dans un second temps, il explore les mécaniques de notre jeu, en explicitant nos difficultés, mais surtout nos solutions et choix de gameplay. Ensuite, il rentra en détail sur la direction artistique, avec en son coeur le processus créatif derrière nos boss et nos environnements, en exposant également nos inspirations. Le site internet est également mis en exergue, fonctionnel depuis la première soutenance, il est désormais complêtement prêt à accueillir des visiteurs et à susciter leur envie de découvrir notre jeu. Enfin le rapport reprend les points essentiels abordés et rasssure sur notre volonté de tenir nos engagements.

2 Le projet

2.1 Rappel du Scénario

Notre jeu, "Melody: Chronicle of the Knight", est un jeu d'aventure en 2D se déroulant dans un univers médiéval-fantastique. Le joueur y incarne un jeune chevalier tout juste adoubé, à qui le roi confie la mission de restaurer la paix dans le royaume. Cette quête implique de vaincre des créatures mystérieuses réparties dans différentes régions du royaume et de collecter les gemmes donnant a ces créatures de pouvoirs mysthique.

Image généré par IA

La particularité de notre jeu réside dans son approche narrative et son gameplay. Le joueur commence son aventure dans le château, naviguant dans un monde à première vue paisible. Il affrontera ensuite les boss dans des scènes de combat vue de côté, rappelant les classiques du jeu 2D. Chaque victoire contre un boss permet au joueur de récupérer une gemme spéciale, conférant des pouvoirs uniques et une note de musique essentielle.

Cependant, la véritable intrigue se dévoile après la première fin du jeu, où le roi, personnage de confiance, révèle sa véritable nature en utilisant les gemmes afin d'anéantir le monde. Cette révélation bouscule le joueur et l'invite à redémarrer le jeu avec une nouvelle perspective. Dans cette deuxième partie, le chevalier, conscient de la tromperie du roi, cherche à rallier les êtres mystérieux, qu'il avait autrefois combattus, à sa cause. Utilisant la flûte enchantée et les notes musicales acquises, le joueur peut apaiser ces créatures, les ralliant ainsi à sa cause pour la bataille finale contre le roi.

La mécanique de recommencer le jeu, après le virement de situation, ajoute une profondeur unique à l'expérience de jeu, incitant les joueurs à réévaluer leurs décisions et explorer des voies alternatives pour une fin plus satisfaisante.

2.2 Retour sur le cahier des charges techniques

Suite aux conseils reçus concernant nos objectifs de projet, nous avons pris la décision de réviser notre planning de soutenance afin de mieux refléter les étapes actuelles de développement. Voici donc le planning actualisé :

1 ère Soutenance	- Multijoueur : faire en sorte que deux joueurs puissent se rejoindre dans une même session. - Développement des idées de boss (leur design, leur mécaniques et paternes). - Scène du château à 25% faite (juste les dessin aucune interactions possible). - Développement du personnage jouable. - Site Web
2-ème Soutenance	- Développement de deux boss (IA et paternes). - Développement du système de combat (attaque + animation d'attaque + barre de vie + perte de PV si le boss est attaqué) - Avancé sur les dessins des boss (Deux boss dessiné). - Intégration de ces nouveauté au multijoueur. - Mise a jour du site Web
3-ème Soutenance	- Intégration des IA aux dernier boss. - Finalisation du système de combat (Boss peut faire perdre des dégâts au joueur). - Finalisation du design des scènes et des boss. - Mise en place de l'histoire. - Intégration des nouveautés au multijoueur - Finalisation du site Web.

Dans ce dernier, nous avons étalé le multijoueur sur l'ensemble des soutenances, apporté des précisions sur le développement du système de combat et sur la création du personnage.

3 Mécaniques

3.1 Déplacements

Dans tout jeu, qu'il s'agisse du gameplay basique d'un Mario ou de la complexité d'un Dark Souls, la capacité de se déplacer est essentielle. Cependant, même pour des déplacements de base, il ne suffisait pas de simplement faire avancer le personnage. Dans notre objectif de créer un jeu vivant et rythmé, les mouvements se devaient de suivre cette danse effrénée. La précision des sauts représentait un défi majeur. Nous avons cherché comment faire en sorte que lorsque le joueur lâche le bouton de saut, le personnage retombe plus rapidement. Deux options se sont présentées à nous, soulevant encore débat : réduire la vitesse sur l'axe Y ou augmenter la gravité. Pour l'instant, nous avons opté pour la réduction de la vitesse, mais cette solution n'est pas définitive.

```
if (Input.GetButtonUp("Jump") && rb.velocity.y > 0f)
{
    rb.velocity = new Vector2(rb.velocity.x, rb.velocity.y * 0.5f);
}
```

Cependant, afin d'éviter que le personnage ne s'envole vers l'infini et au-delà en sautant sans s'arrêter, nous avons créé une entité appelée "ground check" pour vérifier que le personnage est bien au sol avant de sauter.

```
[SerializeField] private Transform groundCheck;
[SerializeField] private LayerMask groundLayer;

private bool IsGrounded()
{
    return Physics2D.OverlapCircle(groundCheck.position, 0.2f, groundLayer);
}
```

La création d'une fonction flip pour permettre au personnage de se retourner et n'a pas été particulièrement problématique.

```
private void Flip()
{
    if (isFacingRight && horizontal < 0f && IsOwner || !isFacingRight && horizontal > 0f && IsOwner)
    {
        isFacingRight = !isFacingRight;
        Vector3 localScale = transform.localScale;
        localScale.x *= -1f;
        transform.localScale = localScale;
    }
}
```

fonction flip

Inspirés par les roulades de Dark Souls et les dashs de Hollow Knight, nous avons intégré un dash qui, lorsqu'il est utilisé, octroie des frames d'invincibilité pour esquiver les attaques des boss. Pour l'intégrer au multijoueur, nous avons mis en place une vérification pour s'assurer que la personne qui effectue les inputs est bien l'owner, en utilisant souvent la fonction isOwner, préprogrammée dans la classe NetworkBehaviour du package Network for GameObject.

```
private IEnumerator Dash()
{
    canDash = false;
    isDashing = true;
    float originalGrav = rb.gravityScale;
    rb.gravityScale = 0f;
    rb.velocity = new Vector2(transform.localScale.x * dashingPower * -1f, 0f);
    tr.emitting = true;
    yield return new WaitForSeconds(dashingTime);
    tr.emitting = false;
    rb.gravityScale = originalGrav;
    isDashing = false;
    yield return new WaitForSeconds(dashingCooldown);
    canDash = true;
}
```

fonction dash

Le code de déplacement a été notre moindre souci, réalisé rapidement et efficacement avec peu de difficultés, à l'exception de son intégration dans le multijoueur, qui a été notre principale préoccupation.

3.2 Système de combat

La conception du système de combat s'est avérée être un défi de taille, étant donné son importance centrale dans le jeu.

La gestion des points de vie a été relativement simple, avec l'implémentation d'une classe HP capable de retirer et de restaurer des points de vie, limitant ces derniers entre 0 et 100 pour le personnage jouables. En parallèle, une classe HP Display a été créée pour afficher la barre de points de vie du personnage.

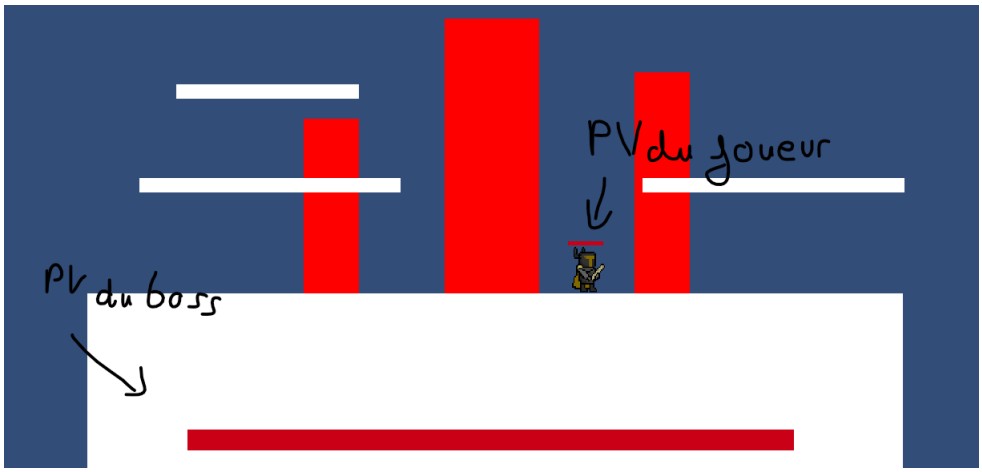


Image concept de boss

La véritable complexité est survenue lors du développement d'une mécanique d'attaque fonctionnant en multijoueur. Bien que de nombreuses vidéos présentent des systèmes d'attaque qui fonctionnent en solo, nous avons dû nous tourner vers l'utilisation de projectiles en réseau. Les attaques sont ainsi des entités distinctes du personnage, apparues au niveau du point de spawn attaché à ce dernier.

Lorsqu'un personnage attaque, deux entités apparaissent : un clone côté serveur et un autre côté client. Le clone côté serveur gère toute la logique de l'attaque, déclenchant la fonction de dégâts dans la classe HP du collider touché, tandis que le clone côté client est responsable des animations et des images associées.

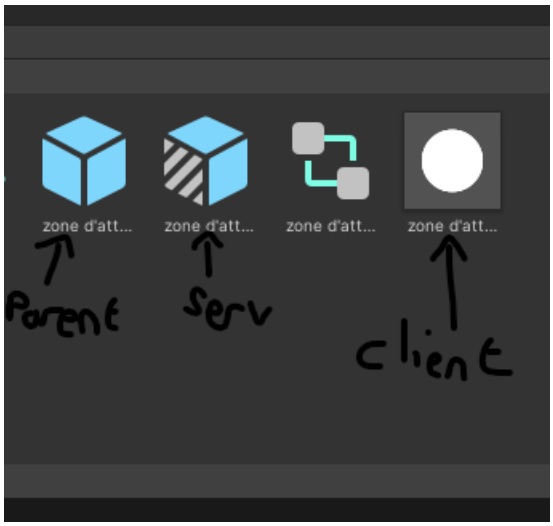


Image des attaques (le parents nous a permis d'initialiser les deux autres)

Pour prévenir les tensions et les amitiés brisées, nous avons délibérément évité les coups entre coéquipiers, créant ainsi une classe HPforboss mais ressemble en grande partie au code de la classe HP. Celle-ci sera détectée par les attaques des joueurs, empêchant ainsi les situations où le manque de coopération pourrait entraîner la mort d'un allié.

```
public class Attaque : MonoBehaviour
{
    [SerializeField] private int damage = 5;
    [SerializeField] private bool istouching = false;

    private ulong ownerClientId;

    private void OnTriggerEnter2D(Collider2D col)
    {
        if(col.attachedRigidbody == null)
        {
            istouching = false;
            return;
        }

        // cherche si le l'entité avec le collider attacher touché a le
        //component Healthforboss et si oui alors on appel la fonction take damage de
        //la class Healthforboss avec l'argument health.
        if(col.attachedRigidbody.TryGetComponent<Healthforboss>(out Healthforboss health) )
        {
            istouching = true;
            health.TakeDamage(damage);
        }
    }
```

Image du code Attaque(un des component de l'attaque serv)

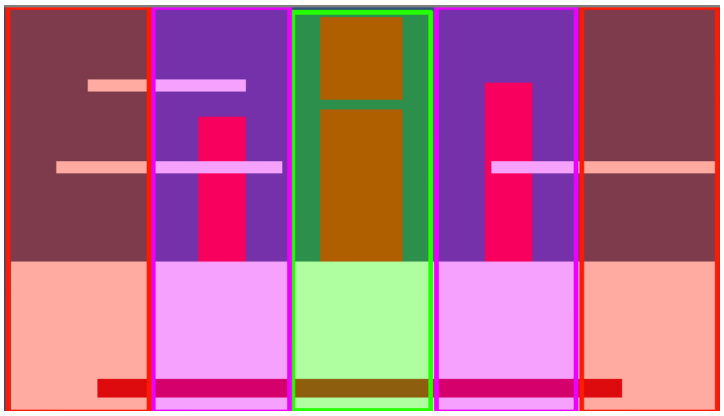
Bien que cela puisse sembler simple, l'utilisation des serveurs RPC et des clients RPC était nouvelle pour nous, tout comme le fait que les informations fournies ne produisaient pas les résultats attendus. Nous avons donc dû retravailler le code pour obtenir le résultat souhaité et être pleinement satisfaits du système en place.

3.3 Inteligence Artificielle des Boss

Lors de la conception de notre jeu "Melody: Chronicle of the Knight", une attention particulière a été portée sur l'intelligence artificielle (IA) des boss. Inspirés par divers jeux, nous avons opté pour une IA qui réagit et s'adapte en fonction des actions du joueur, à l'instar de ce que l'on peut observer dans "Dark Souls". Cette décision visait à offrir une expérience de jeu plus immersive et stratégique, contrastant avec le modèle d'attaques aléatoires employé dans des jeux comme "Cuphead".

Pour implémenter cette IA adaptative, nous avons imaginé une aire de combat séparé en trois zones spatiales distinctes, chacune dictant le comportement du boss en fonction de la position du joueur. Par exemple, dans notre scénario avec le boss Kraken, le style d'attaque du boss varie selon que le joueur se trouve dans la zone verte (proche), la zone rose (intermédiaire), ou la zone rouge (éloignée). Si le joueur se trouve dans la première zone, notre poulpe géant balayera la scène avec un tentacule. Si le joueur se trouve dans la deuxième zone, notre boss transpérsera la niveau avec plusieurs tentacules. Enfin, si le joueur se trouve dans la dernière zone, la créature fera pleuvoir des projectiles sur la scène. Cette approche permet non seulement de diversifier les tactiques de combat, mais aussi à annulé la prédictabilité du boss, incitant les joueurs à analyser et à réagir aux modèles d'attaque du boss.

Scène original



Scène séparé en trois zones

La complexité de cette tâche a été amplifiée lorsqu'il a fallu adapter cette IA au mode multijoueur. La première solution, centrée sur le premier joueur pénétrant dans la salle du boss, s'est avérée être une approche intéressante, mais limitée dans la création d'une expérience multijoueur dynamique. En conséquence, nous avons envisagé d'introduire un système d'alternance ciblée, où le boss changerait de cible aléatoirement pendant le combat. Cette modification demanderait aux joueurs de rester vigilants et de coopérer étroitement, partageant le rôle d'agresseur et de soutien selon les besoins.

3.4 Multijoueur

L'intégration du multijoueur dans notre jeu était véritablement un élément central de notre projet. Ces fonctionnalités a été conçue selon un modèle client-serveur auto-hébergé similaire à celles utilisées dans des jeux indépendants tels que "Gang Beasts" ou "Satisfactory", nous avons opté pour une structure où l'un des joueurs sert de serveur, comme illustré ci-dessous. Ce choix nous a permi d'évité le gestion d'un serveur dédié, économisant ainsi un temps précieux.

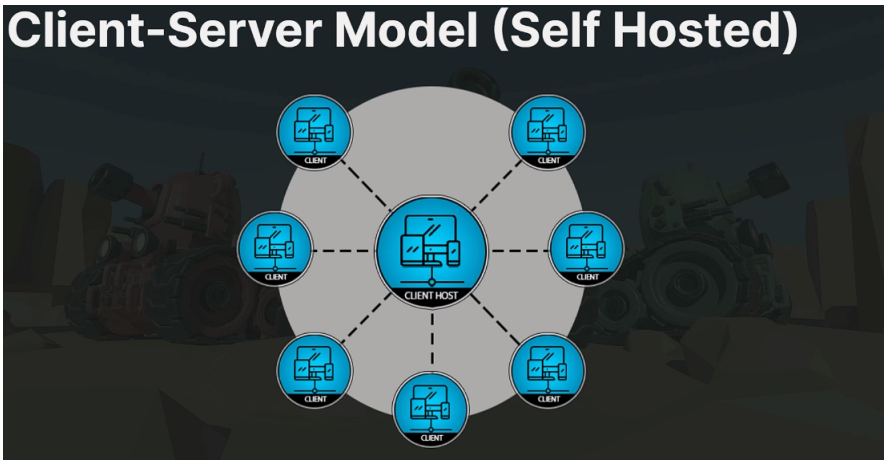


Image de Udemy

Pour développer cet aspect multijoueur, nous avons suivi une formation spécialisée dans le développement de jeux multijoueurs avec Unity 2D. Ce processus s'est révélé être un défi technique majeur, et nous avons rencontré plusieurs difficultés. Nous avons commencé par développer un système permettant la communication fluide entre un hôte et les clients, en utilisant des outils, tel que NetCode, de Unity. Cependant, aligner les mouvements des personnages sur différents appareils a été un obstacle de taille. La formation que nous avions suivie était plus adaptée aux jeux en vue du dessus et en mode compétitif, et ne correspondait pas parfaitement à notre concept.

Nous avons surmonté se problème en nous inspirant d'autres formations ne proposant pas de multijoueur. Par la suite nous avons adapté ce code orienté sur des jeux solo, en un code compatible avec le multijoueur.

De plus, l'ajout des boss dans le jeu fu une étape importante dans notre projet. Nous avons pris soin de les synchroniser pour tous les joueurs. Heureusement, cette intégration s'est avérée plus simple que prévu. En effet l'implémentation des boss diffère très peu de celle des joueur. Ceci nous permis de les intégrer avec aisance dans le jeu multijoueur.

3.5 Scènes

Nous avons choisi de faire un jeu comprenant plusieurs scènes à travers lesquelles le personnage pourrais naviguer. Cette approche est courante dans les jeux 2D, tel que "Cuphead" ou "The Legend of Zelda". Le joueur, en entrant dans une zone au préalable définit, est téléporté dans une nouvelle scène. Nous avons donc décidé de créer cinq zones distinctes. Une zone d'apparition pour les joueurs et quatre autres représentant les différentes salles des boss.

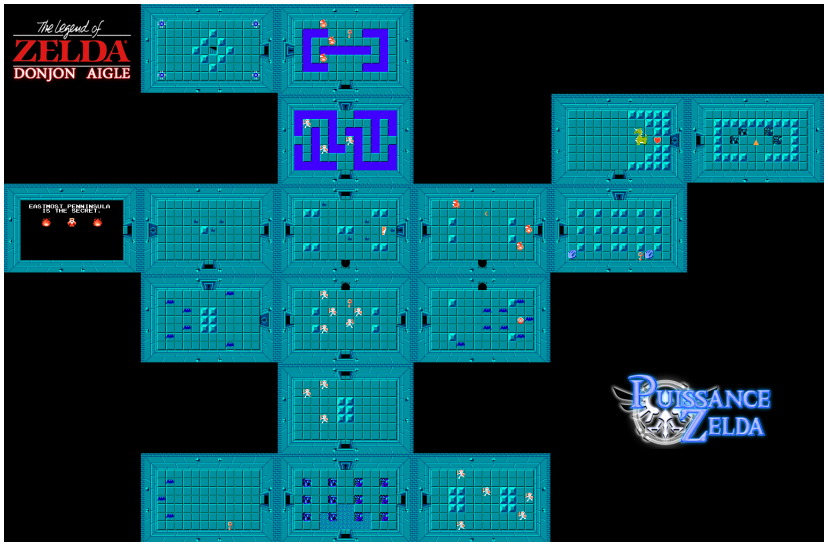


Image illustrant les différente scène de Zelda

L'implémentation de cette fonctionnalité fu également un défi, principalement en raison des complexités liées à la gestion du multijoueur. L'un des problèmes rencontrés était que, lors du changement d'une scène, les personnages des joueurs étaient détruits. En effet, notre programme de gestion du multijoueur ne conservait pas les informations des joueurs d'une scène à l'autre.

Après d'intenses recherches sur divers forums, nous avons finalement trouvé une solution. Nous avons développé une méthode permettant au programme de conserver les informations du joueur lors d'un changement de scène, assurant ainsi une transition fluide et cohérente pour les joueurs dans l'univers de notre jeu.

4 Direction Artistique

4.1 Visuel des Boss

Alup Vorthe, Rédempteur des dragons

Un dragon de feu dégageant une aura d'agilité et de puissance. Les nuances de rouge sont subtilement accentuées sur différentes parties de son corps, tandis que ses yeux rougeoyants brillent d'une lueur fluorescente, lui donnant une expression agressive et menaçante. Sa queue, forgée dans une lame d'acier, renforce davantage son aspect agile et tranchant, lui permettant ainsi d'effectuer des attaques redoutables avec celle-ci.

Hippiel Cerbal, La Momie du Crépuscule Eternel

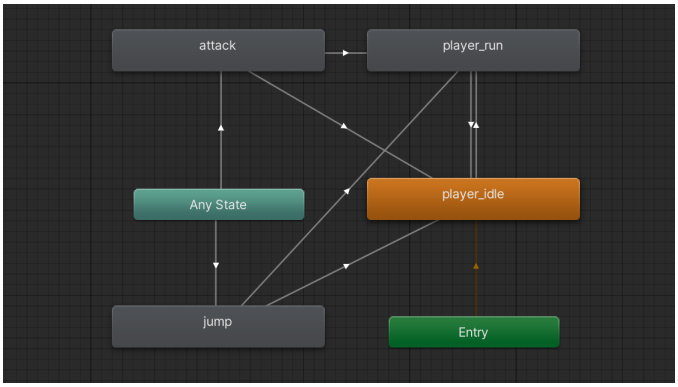
Dans les sables mouvants du village désertique, où les pyramides se dressent comme des sentinelles silencieuses, se trouve un adversaire redoutable. La Momie du Crépuscule Eternel, une créature enveloppée de mystère et de bandages. Elle est ornée de bijoux évoquant la chauve-souris, avec un œil vert inquiétant qui brille d'une lueur sinistre.

Olirevi Ephelo, Le maître des profondeurs

Des profondeurs insondables de la mer Thyrennia surgit le redoutable Kraken, dont la peau rugueuse et impénétrable défie toute tentative d'attaque. Ses tentacules massifs et adhérents servent de bouclier protecteur, repoussant toute proie potentielle loin des habitants des océans. Doté d'une agilité incomparable dans les eaux tumultueuses, il évite avec grâce les projectiles et se déplace à une vitesse prodigieuse, semblant défier les lois de la nature elle-même.

4.2 Animation

Dans le cadre de notre projet de développement de jeu, l'équipe a mis en œuvre des techniques avancées d'animation à l'aide de l'environnement de développement Unity. Le personnage principal, représenté par l'objet *player*, a été doté de plusieurs animations distinctes : *Attack*, *Jump*, *Idle* et *Run*. Les sprites ont été élaborés sur une *tilemap* et ensuite découpés à l'aide de l'outil approprié. Ils ont ensuite été intégrés dans une *timeline* bouclée pour générer des animations fluides et harmonieuses. Ainsi, le personnage principal se compose de quatre composants, prêts à être exploités ultérieurement.



L'Animator dans Unity

L'outil *Animator* offre la possibilité d'établir des liens entre chaque animation. Par exemple, l'état *player_idle* constitue l'état de base du personnage, manifesté par cette animation. À partir de cet état, l'animation de course du joueur peut être activée. Pour ce faire, des conditions ont été ajoutées : lorsque la vitesse est supérieure à 0.01, l'animation de course est déclenchée, et vice versa.

Il convient de noter l'existence de l'état *Any state*, permettant de déclencher un autre état à partir de n'importe quel point. Ainsi, il est possible de sauter même en courant ou d'attaquer sans bouger, grâce au paramètre *isjumping* vérifié en continu. Après le saut, deux possibilités se présentent : rester immobile ou continuer à courir.

Par conséquent, deux liens distincts ont été établis pour chaque éventualité, avec les conditions appropriées. Si la vitesse est inférieure à 0.01 et que le personnage ne saute pas, il passe à l'état d'immobilité, et ainsi de suite pour l'attaque.

Les conditions

Pour modifier la valeur de ces paramètres, il est nécessaire d'intervenir directement dans le code. Dans le script de déplacement du personnage, une variable *animator* de type *Animator* a été ajoutée, faisant référence à l'*animator* où les liens sont effectués. Les valeurs ou états peuvent être attribués à cet *animator* à l'aide des fonctions *SetBool* ou *SetFloat* ; dans ce cas, seules ces deux fonctions ont été utilisées. Ainsi, lors de la détection de la collision avec le sol, la ligne `animator.SetBool("isjumping", false)` a été ajoutée, affectant la valeur *false* au paramètre *isjumping* et déclenchant le changement d'état ainsi que l'animation correspondante. De plus, la négation de *isgrounded* a été ajoutée afin que tant que le personnage n'est pas en contact avec le sol, il reste en animation de saut et sa vitesse est mise à zéro pour éviter une animation de saut simultanée. L'animation de l'attaque est intégrée dans la *zone d'attaque*. Bien qu'il ne s'agisse pas d'un état à proprement parler, il s'agit d'un objet distinct qui détecte les collisions avec d'autres objets pour infliger des dégâts.

```
if (!IsGrounded()) {
    animator.SetBool("isjumping",true);
    animator.SetFloat("speed",0);
}
if (IsGrounded()) animator.SetBool("isjumping",false);
if (Input.GetKeyDown(KeyCode.LeftShift) && canDash)
{
    StartCoroutine(Dash());
}

if (Input.GetKey(KeyCode.J)) animator.SetBool("isattacking",true);
else {animator.SetBool("isattacking",false);}
```

Le programme intégrant les animator

4.3 Environnement

Le thème de notre jeu est le médiévale fantaisiste c'est pourquoi comme tout bon jeu médiévale le cœur de notre carte est un château. Autour de ce château 4 petit village très différent mais qui recouvre un point commun le chaos.

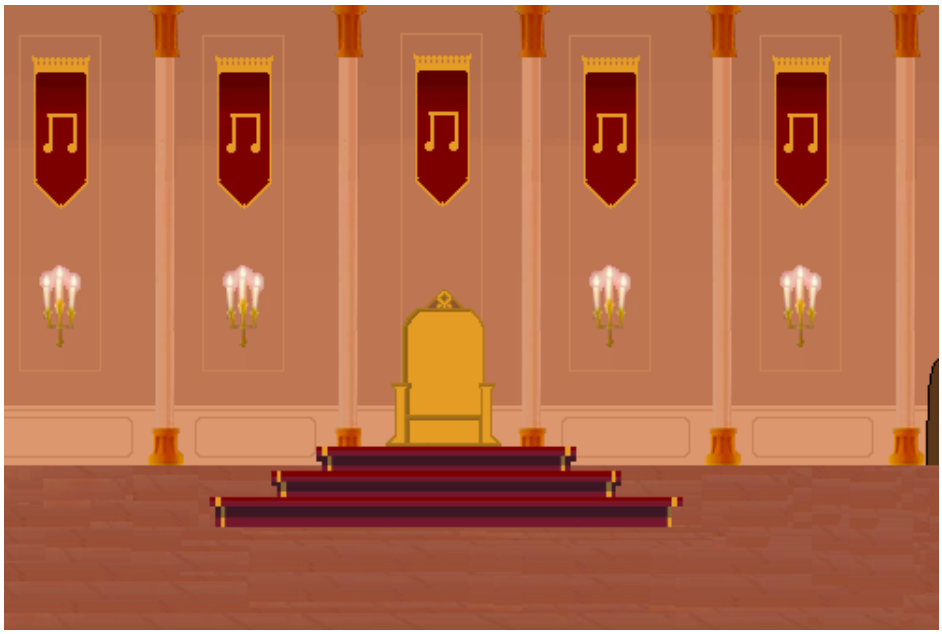
Chacun de ces villages ont une créature mystérieuse qu'ils vénèrent , l'un de ces villages est un village sur pilotis lors de l'arrivé dans ce village vous comprenez rapidement que les villageois vénèrent cette créature, vous trouvez ça étrange mais les ordres du rois doivent être exécuté, vous vous rendez donc au milieu de la mer et provoqué le kraken. Une fois le combat achevé lors de votre retour au village l'ambiance à totalement changé, il n'y a plus aucun signe de vie la mer fait rage et surtout le ciel c'est assombrit.

Au Sud de ce village se trouve une région montagneuse ou ce trouve au pied de la plus grande montagne du royaume un village encore une fois rempli de joie, le roi vous ordonne cependant d'aller tuer la divinité qui si trouve donc vous vous mettez en route. Une fois arriver au sommet un dragon apparait le dragon ne semble pas menaçant mais vous décidez de lui porter un coup il décide de répondre en retour. Votre combat terminer vous descendez de la montagne et soudain une brume noir et épaisse vous obstrue la vue , de retour au village certain endroit sont en feu et plus aucune vie n'anime le village.

A l'Est de cette montagne ce trouve le village des ouvriers c'est là qu'aura lieu notre 3ème affrontement lorsque l'on arrive dans cette région aucun village à l'horizon seulement une grotte mais on se rend compte rapidement que si l'on doit ce nom de village des ouvriers c'est parce que les habitant vivent dans cette grotte, une fois arriver au village encore une fois rien d'étrange les mines sont activent les villageois heureux, mais vous décidez de continuer votre quête. Plus profond dans la grotte un gouffre ce présente à vous, une villageois vous dit que le chevalier abyssal si trouve vous sautez donc sans hésiter, une fois arriver lorsque vous vous relevez des torches s'allument autour de vous et le chevalier abyssal apparaît. En retrouvant la sortie de ce gouffre le village s'est transformé plus aucune machine n'est active les mines ne tournent plus.

Enfin la fin de votre quête ce trouvera au Nord de cette région dans le village des sables. Ce village, avec en arrière plan des pyramides et une légère brise, est habité par des bédoins qui semblent heureux et épanouis. Le joueur distance ensuite celui-ci et traverse de mystérieuses ruines, où le vent devient de plus en plus puissant et ralentit l'avancée du joueur, jusqu'à arriver devant un temple avec une vision très réduite due au sable. L'intérieur sombre est constitué d'une grande salle éclairée par des torches et avec divers ornements au mur, ainsi qu'un sarcophage au bout de la pièce. Ainsi dès que le joueur se rapproche du sarcophage, celui-ci s'ouvre pour libérer une momie déchainée.

L'intérêt des environnements est que ceux-ci sont tous très paisibles et accueillants avant les affrontements contre les boss et deviennent entièrement dévastés suite à cela, ce qui contraste donc avec la mission du roi qui est supposément d'apporter la paix.



salle du trône

On a volontairement crée une salle du trône luxueuse car le roi est une personne qui a soif de pouvoir c'est pourquoi on crée dès le début du jeu un contraste élever entre le désespoir du village et la richesse du roi qui peut aguiller certain à penser que le roi cache quelque chose...

5 Site Internet

Le site internet s'est inspiré du style rétro, évoquant avec nostalgie les jeux classiques tels que Terraria, ainsi que leurs sites web emblématiques. Des éléments de design ont également été empruntés au célèbre site de Minecraft et à l'univers visuel envoûtant de Final Fantasy. Pour assurer une expérience utilisateur fluide et stable, le fond du site a été minutieusement conçu et figé.

La barre latérale, arborant un design moderne et épuré, facilite la navigation entre les différentes pages. Bien que la section de téléchargement soit d'ores et déjà accessible, le téléchargement effectif du jeu reste en attente. Une barre de progression interactive a été implantée pour permettre aux visiteurs de suivre l'avancement du développement, une tâche qui est actualisée manuellement pour tenir les utilisateurs informés de nos progrès.

Pour plonger davantage les joueurs dans l'univers du jeu, des descriptions détaillées des boss sont mises à disposition sur le site, les incitant ainsi à rejoindre l'aventure. En outre, l'engagement est pris d'actualiser régulièrement le contenu du site pour partager nos dernières avancées et réalisations. Des fonctionnalités de contact ont également été intégrées, permettant aux utilisateurs de nous joindre aisément via nos adresses e-mail Epita.

Une section spéciale est dédiée à la présentation de l'équipe talentueuse qui se cache derrière le développement du jeu. Cette rubrique offre également un lien vers le site de NOU, l'entreprise responsable non seulement du jeu, mais également de divers autres projets. Enfin, les rapports de soutenance sont soigneusement répertoriés dans un onglet dédié, offrant ainsi une consultation aisée aux visiteurs intéressés.

6 Conclusion

Notre projet avance à un rythme raisonnable malgré quelques retards. Voici un récapitulatif de ce qui a été accompli et de ce qui reste à faire :

Ce qui a été fait :

-Implémentation du mode multijoueur avec plusieurs instances

-Déplacement du personnage

-Système de combat du personnage

-Conception de la scène du château sans les interactions

-Avancement du site web en parallèle avec le projet

-Développement de l'IA du boss Kraken

-Dessin complet du boss Kraken

-Avancement significatif dans la conception des boss Dragon et Momie

Ce qui reste à faire :

-Finalisation des IA des derniers boss (l'un d'eux accuse un retard)

-Achever la mise au point du système de combat, notamment en ce qui concerne les boss

-Intégration de l'histoire dans le jeu

-Finalisation des dessins des boss (certains accusent un retard)"