Projet "Echo of Nyxalis"
Cahier des Charges Fonctionnel

PROMO 2029 Prepa Sup D1

Étudiants :
Aïzabel AUTECHAUD
Augustin BAZIN ROY
Julien CHARLES
Clément GOMEZ
Angéline TAING

22 octobre 2024

Sommaire

Introduction

Dans le cadre de notre projet informatique de second semestre, nous avons décidé de réaliser un jeu vidéo : Echo of Nyxalis. Echo of Nyxalis est un jeu de rôle de type RPG léger en 3D. Le joueur incarne un humain qui évolue dans un monde futuriste-fantastique en vue à la troisième personne. Ce monde est constitué de plusieurs zones, chacune avec ses propres caractéristiques et défis. La population vit sous le joug de forces mystérieuses et maléfiques qui imprègnent leurs mondes. Nous vous proposons ainsi de découvrir la réalisation du projet durant le premier temps de travail qui nous a été accordé.

Les joueurs incarnent des personnes qui se réveillent sur Nyxalis, une planète qui devrait leur être familière. Mais, un coup d'œil à travers le hublot abîmé leur suffit à dire qu'il y a une erreur. La planète verte est violette. Au milieu de la confusion, quelqu'un de l'équipage découvre que la capsule cryogénique a en fait dysfonctionné, et que leur sommeil de glace a duré plusieurs millénaires au lieu d'une centaine d'années. Ils sont sur la bonne planète. Seulement, ils se sont trompés d'ère.

Soudain, la porte du vaisseau cède. Une ombre tombe sur nos voyageurs. Une horreur sans nom, libre de toute définition, les regarde. Elle, qui n'a jamais eu de visage, ne peut s'empêcher d'envier ces êtres. Elle avance. Ils reculent. Elle ne comprend pas. Ils ne veulent pas la comprendre. Ou peut-être ne peuvent-ils pas la comprendre ? Pourront-ils transcender leurs peurs, le temps et leur humanité pour survivre ?

Tout au long du jeu, les joueurs seront invités à explorer et interagir avec ce monde aussi fascinant que terrifiant, avec un but : se battre pour survivre et comprendre ce qui s'est exactement passé sur cette planète hostile. Echo of Nyxalis saura séduire les joueurs grâce à ses énigmes, son univers et son intrigue entraînante. Il s'inspire d'ailleurs de plusieurs jeux à succès comme No Man's Sky et Outer Wilds.

L'aspect multijoueur permettra aussi aux utilisateurs de progresser aux côtés de leurs coéquipiers afin d'affronter les monstres les plus redoutables et débloquer les prochaines étapes de l'histoire. Que vous choisissiez de jouer seul ou en équipe, vivez une aventure immersive dans un autre monde en quête de percer les secrets qui vous entourent.

Un des systèmes les plus importants du jeu est le système de dialogue entre les joueurs et les PNJ. Ce système permet aux joueurs de découvrir petit à petit la langue des monstres qu'ils rencontrent à travers des énigmes, ajoutant une couche supplémentaire de profondeur et d'immersion à l'expérience de jeu. Les dialogues sont dynamiques et évoluent en fonction des actions et des choix des joueurs, offrant une narration riche et engageante.

0.1 Répartition et Organisation

Pour vous rappeler voici les taches qui nous était assigné.

Tâches	Responsable	Suppléant
3D / Assets	Aïzabel	Julien
Sound Design	Clément	Angeline
UX / UI Design	Julien	Aïzabel
Site Web	Clément	Julien
Multijoueur	Angeline	Augustin
Gameplay	Augustin	Aïzabel
IA	Augustin	Angeline
Qualité	Julien	Clément

1 Avancement du projet

1.1 site internet

Le site a été développé dans le but de promouvoir ce jeu mais aussi notre entreprise Prisme. Le site est organisé de manière à guider facilement les utilisateurs à travers les différentes sections. Sur la page d'accueil, vous trouverez un aperçu de ce que propose le site, comme télécharger le jeu. Chaque page a été construite de manière à offrir une expérience utilisateur optimale, tout en mettant en valeur le contenu principal. Sur la page d'accueil, par exemple, vous pourrez découvrir un gradient de couleur qui évolue, passant d'un assortiment de teintes plus claires à des nuances plus sombres en haut et en bas de la page, créant ainsi une ambiance plus interactive et moderne.

De plus, nous avons décidé d'intégrer plusieurs fonctionnalités essentielles afin de rendre le site web le plus accessible possible. C'est dans cette optique que nous avons ajouté un menu de navigation en haut de la page et présent sur chacune des pages afin que l'utilisateur puisse à sa guise, aller d'une page à l'autre sans avoir à trop chercher. Qui plus est, notre site web est responsive design, c'est-à-dire qu'il s'adapte aussi bien sur portable que sur ordinateur.

À l'avenir, nous prévoyons d'ajouter des images du jeu ainsi que des animations pour ajouter de la dynamique à l'ensemble du site et rendre notre jeu plus attractif.

Nous avons décidé de fragmenter notre site web en de multiples pages afin de permettre à l'utilisateur de trouver des informations plus facilement. En outre, nous avons ajouté de nombreuses pages dont voici un récapitulatif détaillé :

- **Accueil** : Cette page présente une vue d'ensemble du site, avec un aperçu des sections principales et des informations essentielles pour aider l'utilisateur à naviguer rapidement.
- **Présentation de l'histoire du jeu** : Cette section raconte l'intrigue du jeu et présente les différents personnages présents dans Echo of Nyxalis.
- **Chronologie** : Une ligne du temps détaillant les étapes clés du jeu depuis sa création. Cette page permet de suivre l'évolution du projet, les mises à jour majeures, ainsi que les moments importants qui ont influencé son parcours.
- **Outils utilisés** : Elle présente les logiciels utilisés pour créer le jeu. Ainsi, vous y trouverez des informations sur chacun des logiciels utilisés, ainsi qu'un petit descriptif expliquant la pertinence d'utiliser ces logiciels dans une logique de créer le meilleur jeu possible.
- **Sons** : Une section dédiée à la bande sonore du jeu, comprenant une classification des différents sons du jeu.
- **Membres** : Cette page met en avant les membres de l'équipe de développement. Chaque membre est présenté avec son rôle dans le projet, son parcours et sa contribution spécifique au jeu.
- **Entreprise** : Une page dédiée à l'entreprise qui a développé le jeu. Vous y trouverez des informations sur l'histoire de l'entreprise, sa mission, ses valeurs et ses projets à venir. Elle permet de mieux comprendre le contexte dans lequel le jeu a été créé et de découvrir d'autres projets de l'entreprise.
- **Télécharger** : Comme son nom l'indique, cette section permet le téléchargement du jeu ainsi qu'un aperçu détaillé des modifications apportées par chacun des membres de l'équipe à chaque soutenance technique.

1.2 Histoire

Les joueurs incarnent des personnes qui se r´eveillent sur Nyxalis, une plan‘ete qui devrait leur ˆetre famili‘ ere. Mais, un coup d’œil ‘a travers le hublot abˆ ım´ e leur suffit ‘a dire qu’il y a une erreur. La plan‘ete verte est violette. Aumilieu de la confusion, quelqu’un de l’´ equipage d´ecouvre que la capsule cryog´enique a en fait dysfonctionn´e, et que leur sommeil de glace a dur´e plusieurs mill´enaires au lieu d’une centaine d’ann´ ees. Ils sont sur la bonne plan‘ ete. Seulement, ils se sont tromp´ es d’‘ ere. Soudain, la porte du vaisseau c‘ ede. Une ombre tombe sur nos voyageurs. Une horreur sans nom, libre de toute d´ efinition, les regarde. Elle, qui n’a jamais eu de visage, ne peut s’empˆ echer d’envier ces ˆ etres. Elle avance. Ils reculent. Elle ne comprend pas. Ils ne veulent pas la comprendre. Ou peut-ˆ etre ne peuvent-ils pas la comprendre ? Pourront-ils transcender leurs peurs, le temps et leur humanit´ e pour survivre ?

1.3 Création sonore et bande-son

Pour la création des musiques et des effets sonores, j'ai utilisé Logic Pro X, un logiciel développé par Apple, reconnu pour sa puissance et son intuitivité dans la production musicale. Ce choix s'est imposé naturellement, car Logic Pro X offre une bibliothèque étendue de bandes-son, appelée Apple Loops, qui permet d'accéder à une grande variété de boucles sonores prêtes à l'emploi. Ces ressources m'ont été très utiles pour poser les bases des compositions et ajuster les morceaux à l'ambiance souhaitée pour le jeu.

Mon objectif principal était de créer une ambiance sonore immersive et cohérente avec l'univers futuriste de notre jeu. Les musiques de fond et les morceaux de combat devaient :

1. Évoquer un futur technologique et intrigant grâce à des sonorités modernes et électroniques.
2. Rester dynamiques et entraînantes, notamment pour accompagner les scènes d'action ou de combat.
3. Susciter le mystère, en intégrant des éléments sonores plus sombres et introspectifs pour les phases d'exploration et de réflexion.

Grâce à l'Apple Loops, j'ai pu explorer plusieurs palettes sonores, en mélangeant des instruments synthétiques, des rythmes futuristes et des textures complexes, pour traduire ces émotions de manière efficace.

Le processus de création sonore n'a pas été sans défis. Voici les principaux problèmes auxquels j'ai été confrontée, ainsi que les solutions que j'ai apportées :

1. **Limitation de la bibliothèque sonore**
   - **Problème** : Bien que riche, la bibliothèque Apple Loops ne correspondait pas toujours parfaitement à mes attentes. Certaines boucles manquaient de personnalité ou ne correspondaient pas à l'univers spécifique du jeu.
   - **Solution** : J'ai contourné cette limitation en combinant plusieurs boucles pour créer des textures sonores uniques. J'ai également utilisé les fonctionnalités de modification audio de Logic Pro X, comme l'ajustement des fréquences, des filtres et des effets, pour personnaliser les sons et les rendre plus adaptés.
2. **Transitions musicales complexes**
   - **Problème** : Intégrer des transitions fluides entre des morceaux de fond calmes et des musiques de combat plus dynamiques était un défi. Les ruptures brutales pouvaient casser l'immersion.

- **Solution** : J'ai travaillé sur des crossfades et utilisé des effets comme le reverb et le delay pour adoucir les passages d'un morceau à l'autre. J'ai également composé des segments intermédiaires qui servent de ponts musicaux entre deux ambiances.

3. **Optimisation des performances audio**

   - **Problème** : Certains morceaux, trop complexes ou contenant de nombreuses pistes, pouvaient augmenter le poids global des fichiers audio et impacter la performance du jeu.
   - **Solution** : J'ai utilisé des techniques de compression audio et optimisé les paramètres de rendu pour réduire la taille des fichiers sans compromettre leur qualité.

À ce stade, les morceaux de musique de fond et de combat sont finalisés et parfaitement intégrés à l'univers du jeu. Les retours de l'équipe sur ces compositions sont positifs, notamment en ce qui concerne :

- L'atmosphère futuriste et immersive des morceaux.
- La fluidité des transitions, qui contribuent à renforcer l'immersion du joueur.
- L'équilibre entre dynamisme et mystère, qui enrichit l'expérience globale.

—

Voici les principales sources et ressources qui m'ont aidée tout au long de ce processus :

- Logic Pro X User Manual : Documentation officielle d'Apple pour apprendre à tirer parti des fonctionnalités avancées du logiciel.
- Apple Loops Library : Bibliothèque intégrée de boucles sonores et d'effets, disponible sur Logic Pro X.
- Article : "The Role of Music in Video Games: Setting the Mood and Driving the Story" publié dans Game Developer Magazine.

—

Pour la suite, je prévois de :

1. Travailler sur les effets sonores (bruits d'environnement, actions spécifiques des personnages, etc.) pour compléter l'ambiance sonore.
2. Réaliser des tests in-game pour m'assurer que les musiques s'intègrent parfaitement au gameplay et ajuster si nécessaire.

Assets et 3D

Dans le cadre de ce projet, j'ai eu l'opportunité de m'occuper de la création du personnage principal en 3D pour notre jeu vidéo, un petit astronaute. Afin de réaliser cette tâche, j'ai utilisé le logiciel Blender, une plateforme puissante et polyvalente qui m'a permis de modéliser avec précision chaque détail du personnage. Ce dernier, placé sur une base de couleurs représentant la planète Nyxialis, devait non seulement refléter l'univers du jeu, mais aussi capter l'attention du joueur grâce à une conception soignée. Mon objectif était de créer un personnage non seulement visuellement convaincant mais aussi fonctionnel, capable de se déplacer et d'interagir de manière fluide à travers des animations réalistes.

Le processus de création a débuté par la modélisation de la silhouette du personnage. Grâce aux outils de modélisation 3D de Blender, j'ai pu travailler avec une grande précision sur la forme du corps de l'astronaute, ses équipements et ses vêtements. Chaque détail, du visage aux accessoires, a été minutieusement sculpté pour s'assurer que l'aspect visuel correspondait exactement à la vision créative que nous avions pour ce personnage. La combinaison de l'astronaute, par exemple, a été conçue avec des nuances de couleurs en harmonie avec celles de la planète Nyxialis, ce qui a nécessité une attention particulière. En effet, l'idée était de s'assurer que chaque élément visuel de l'astronaute évoque l'univers de la planète tout en étant adapté au gameplay et à l'environnement du jeu.

Une fois la modélisation achevée, j'ai procédé à l'ajout du squelette, ou rigging , qui est essentiel pour animer le personnage et lui donner vie. Le rigging consiste à insérer une structure interne composée de joints et d'os, permettant de contrôler les mouvements du personnage avec fluidité et de manière réaliste. Grâce à ce squelette, chaque articulation, du bras aux doigts, a pu être manipulée pour obtenir des animations plus naturelles, qu'il s'agisse de marches, de sauts ou d'interactions avec l'environnement. Ce processus a été un défi technique en soi, car il nécessitait une bonne compréhension des principes de l'animation, notamment en ce qui concerne les contraintes physiques du mouvement humain et la gestion des déformations de la géométrie du personnage pendant les animations.

Au-delà de l'aspect technique, ce projet m'a confronté à plusieurs défis. L'un des premiers obstacles a été le matériel informatique. Le logiciel Blender, bien que très performant, demande des ressources matérielles importantes pour travailler de manière fluide, en particulier lorsque l'on manipule des modèles complexes en 3D. Mon matériel n'étant pas à la hauteur de ces exigences, cela a entraîné des ralentissements notables, qui ont parfois perturbé le processus de création.

Un autre défi majeur a été la recherche d'informations techniques. Le monde de la modélisation et de l'animation 3D est vaste, et trouver des tutoriels ou des conseils spécifiques à mes besoins s'est révélé difficile. J'ai dû consacrer un temps considérable à fouiller des forums en ligne, regarder des vidéos YouTube et lire des articles spécialisés pour combler mes lacunes et surmonter les obstacles rencontrés. Par exemple, l'aspect de la sélection des couleurs pour la combinaison de l'astronaute ne s'est pas fait sans heurts.

Les couleurs que j'avais en tête pour le personnage ne s'accordaient pas toujours comme je l'avais imaginé, ce qui nécessite des ajustements qui seront finis très prochainement afin de trouver la palette idéale.

Malgré ces difficultés, j'ai continué à avancer grâce aux ressources en ligne qui m'ont fourni des conseils et des astuces. L'entraide des communautés de développeurs et d'artistes 3D a été précieuse pour progresser dans la bonne direction. Par ailleurs, ces échanges m'ont permis d'approfondir ma compréhension du logiciel Blender, d'améliorer mes compétences en modélisation et d'acquérir des connaissances techniques que je pourrai appliquer à l'avenir.

À l'avenir, je devrai répéter ces étapes pour les autres personnages du jeu, notamment les ennemis qui peupleront les donjons. Cependant, contrairement au personnage principal, ces créatures auront un design différent, ce qui nécessitera des ajustements supplémentaires tant au niveau de la modélisation. Le processus sera similaire, mais chaque personnage aura ses propres spécificités qui devront être prises en compte pour garantir une cohérence visuelle et fonctionnelle avec l'ensemble de l'univers du jeu.

En conclusion, la création de ce personnage 3D a été un processus enrichissant à la fois sur le plan technique et créatif. Non seulement cette expérience m'a permis d'améliorer mes compétences en modélisation, animation et rigging, mais elle m'a également appris à surmonter les défis techniques et à chercher des solutions de manière autonome. Cette expérience constitue un pas important dans le développement de mes compétences en développement 3D et m'a donné les outils nécessaires pour continuer à travailler sur les autres éléments du projet avec confiance et efficacité.

1.4 Gameplay

Dans le cadre de notre projet de jeu vidéo, nous avons développé un système de mouvement et de gestion de la caméra pour notre personnage principal. Ce système est inspiré de jeux comme "The Legend of Zelda: Breath of the Wild" (Zelda: BoTW), où le mouvement fluide du personnage et la caméra dynamique jouent un rôle crucial dans l'immersion du joueur. Voici une explication détaillée du code utilisé pour implémenter ces fonctionnalités.

Le mouvement du personnage est géré par la classe `Mouvement`, qui hérite de `MonoBehaviour`. Cette classe contient des variables pour la vitesse de mouvement, la force de saut, et des références aux composants nécessaires comme le `Rigidbody` et la caméra.

```
1  public class Mouvement : MonoBehaviour
2  {
3      public float mouvementSpeed = 5f; // pour r gler le
           mouvement et le saut
4      public float jumpForce = 5f; // Permet de donner une
           puissance du saut
5
6      private Vector3 movement; // Obvious
7      public bool canJump = false; // pour savoir si on peut sauter
8
9      public Rigidbody rb;
10     public Transform cameraTransform; // Pour que la cam ra
           suive le joueur
11     public float mouseSensitivity = 400f; // la sensibilit  de
           la souris
12
13     // ... autres variables ...
14
15     private void Start()
16     {
17         // Initialiser la rotation cible    la rotation actuelle
18         targetRotation = transform.rotation;
19     }
20
21     private void Update()
22     {
23         // Pour bouger le personnage dans l'espace.
24         float horizontal = Input.GetAxisRaw("Horizontal");
25         float vertical = Input.GetAxisRaw("Vertical");
26         if ((horizontal != 0f || vertical != 0f) && Input.GetKey(
               KeyCode.LeftShift)) Mouv(horizontal, vertical,
               mouvementSpeed * 2);
27         else if (horizontal != 0f || vertical != 0f) Mouv(
               horizontal, vertical, mouvementSpeed);
28         else { movement.x = 0f; movement.z = 0f; }
```

```
29
30          // Le syst me de saut
31          if (canJump && Input.GetKey(KeyCode.Space))
32          {
33              rb.AddForce(Vector3.up * jumpForce, ForceMode.Impulse
                    );
34              canJump = false; // Emp che de sauter    nouveau
                    jusqu'   ce qu'on touche le sol
35              jumpTimer = 0f;
36          }
37
38          // Pour bouger    l'aide de la souris.
39          float mouseX = Input.GetAxis("Mouse X") *
                mouseSensitivity * Time.deltaTime;
40          targetRotation *= Quaternion.Euler(0, mouseX, 0);
41          transform.rotation = targetRotation;
42
43          if (!canJump)
44          {
45              jumpTimer += Time.deltaTime;
46              if (jumpTimer >= jumpTimeout)
47              {
48                  canJump = true; // Forcer le saut apr s 3
                        secondes
49                  jumpTimer = 0f; // R initialiser le timer de
                        saut
50              }
51          }
52      }
53
54      // pour avoir les mouvements de gauche    droite
55      private void Mouv(float horizontal, float vertical, float
            movementS)
56      {
57          Vector3 forward = cameraTransform.forward;
58          Vector3 right = cameraTransform.right;
59
60          forward.y = 0f;
61          right.y = 0f;
62
63          forward.Normalize();
64          right.Normalize();
65
66          movement = forward * vertical + right * horizontal;
67          movement = movement.normalized * movementS;
68      }
69
```

```
70      void FixedUpdate()
71      {
72          // pour rendre le mouvement en fonction des touches
73          Vector3 move = new Vector3(movement.x, rb.linearVelocity.
                y, movement.z);
74          rb.linearVelocity = move;
75      }
76
77      private void OnCollisionEnter(Collision collision)
78      {
79          // Si il touche un syst me sol
80          if (collision.gameObject.CompareTag("Sol"))
81          {
82              canJump = true;
83              jumpTimer = 0f;
84          }
85      }
86
87      private void OnCollisionExit(Collision collision)
88      {
89          // quand on quitte le sol
90          if (collision.gameObject.CompareTag("Sol"))
91          {
92              canJump = false;
93          }
94      }
95
96      private void OnCollisionStay(Collision collision)
97      {
98          // Aligner le mouvement avec la normale du sol
99          if (collision.gameObject.CompareTag("Sol"))
100         {
101             Vector3 groundNormal = collision.contacts[0].normal;
102             Vector3 groundForward = Vector3.Cross(transform.right
                    , groundNormal).normalized;
103             Vector3 groundRight = Vector3.Cross(groundNormal,
                    groundForward).normalized;
104
105             // Projeter le mouvement sur le plan du sol
106             Vector3 groundMovement = Vector3.ProjectOnPlane(
                    movement, groundNormal);
107             rb.linearVelocity = new Vector3(groundMovement.x, rb.
                    linearVelocity.y, groundMovement.z);
108         }
109     }
110 }
```

Le mouvement du personnage est géré en fonction des entrées de l'utilisateur. Les touches directionnelles (Horizontal et Vertical) sont utilisées pour déplacer le personnage dans l'espace. La vitesse de mouvement peut être augmentée en maintenant la touche Shift enfoncée. La méthode `Mouv` calcule le vecteur de mouvement en fonction des entrées horizontales et verticales et de la vitesse de mouvement.

Le saut est géré en ajoutant une force verticale au `Rigidbody` du personnage lorsque la touche Espace est pressée et que le personnage est en contact avec le sol. La variable `canJump` est utilisée pour vérifier si le personnage peut sauter.

La gestion de la caméra est également intégrée dans la classe `Mouvement`. La caméra suit le personnage et peut être tournée horizontalement et verticalement en utilisant la souris. La sensibilité de la souris est ajustable via la variable `mouseSensitivity`.

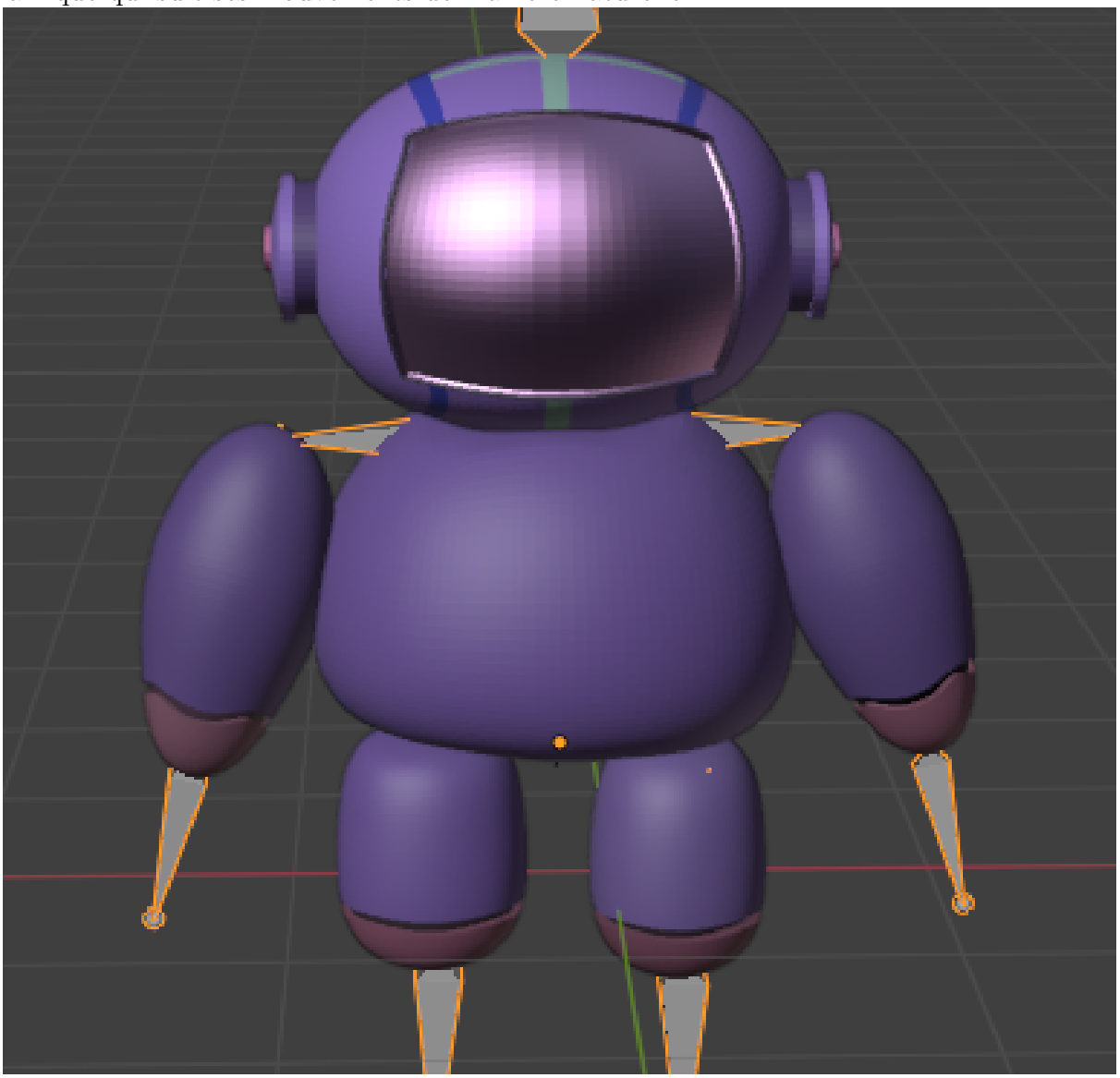
```
1  public CinemachineRotationComposer cam;
2  public CinemachineFollow cam2;
3  private float verticalRotation = -100f;
4
5  public CinemachineVirtualCameraBase virtualCamera; // R f rence
       la cam ra virtuelle
6  public Transform Cameravizual;
7
8  private float targetDistanceToCamera;
9  private float currentDistanceToCamera;
10 private float lerpSpeed = 10f; // Vitesse de lissage
11
12 private void CameraRotation(float mouseY)
13 {
14     verticalRotation -= mouseY;
15     verticalRotation = Mathf.Clamp(verticalRotation, -180f, 90f);
16     if (verticalRotation < 0) cam.Composition.ScreenPosition.y =
          verticalRotation / 70f;
17     else cam.Composition.ScreenPosition.y = verticalRotation / 90
          f;
18
19     Vector3 cameraPosition = Cameravizual.localPosition;
20     cameraPosition.y = (verticalRotation / -90f);
21
22     // Calculer la distance entre le joueur et la cam ra
23     targetDistanceToCamera = Vector3.Distance(virtualCamera.State
          .GetFinalPosition(), Cameravizual.position);
24     currentDistanceToCamera = Mathf.Lerp(currentDistanceToCamera,
           targetDistanceToCamera, Time.deltaTime * lerpSpeed);
25
26     // Ajuster les coordonn es de Cameravizual en fonction de la
           distance
```

```
27      if (cameraPosition.y > 0f)
28      {
29          cameraPosition.y *= 2.5f * currentDistanceToCamera / 8;
30          cam2.FollowOffset.z = -10;
31      }
32      else if (cameraPosition.y <= 0)
33      {
34          cameraPosition.y *= 0.3f * currentDistanceToCamera / 5;
35          cam2.FollowOffset.z = -10 / (1 - cameraPosition.y * 4.5f)
                ;
36      }
37      Cameravizual.localPosition = cameraPosition;
38  }
```

La méthode `CameraRotation` gère la rotation verticale de la caméra en fonction des entrées de la souris. La variable `verticalRotation` est utilisée pour ajuster la position verticale de la caméra. La distance entre le joueur et la caméra est calculée et lissée pour obtenir une transition fluide.

La caméra est également ajustée en fonction de la distance entre le joueur et la caméra, ce qui permet de créer une expérience visuelle immersive et dynamique, similaire à celle de Zelda: BoTW.

Le système de mouvement et de gestion de la caméra que nous avons développé pour notre jeu vidéo est inspiré de jeux comme Zelda: BoTW. En utilisant des techniques de programmation avancées et des outils puissants comme Unity et Cinemachine, nous avons réussi à créer une expérience de jeu fluide et immersive. Ce système permet au joueur de se déplacer librement dans l'environnement du jeu tout en profitant d'une caméra dynamique qui suit ses mouvements de manière naturelle.



1.5 Inventaire

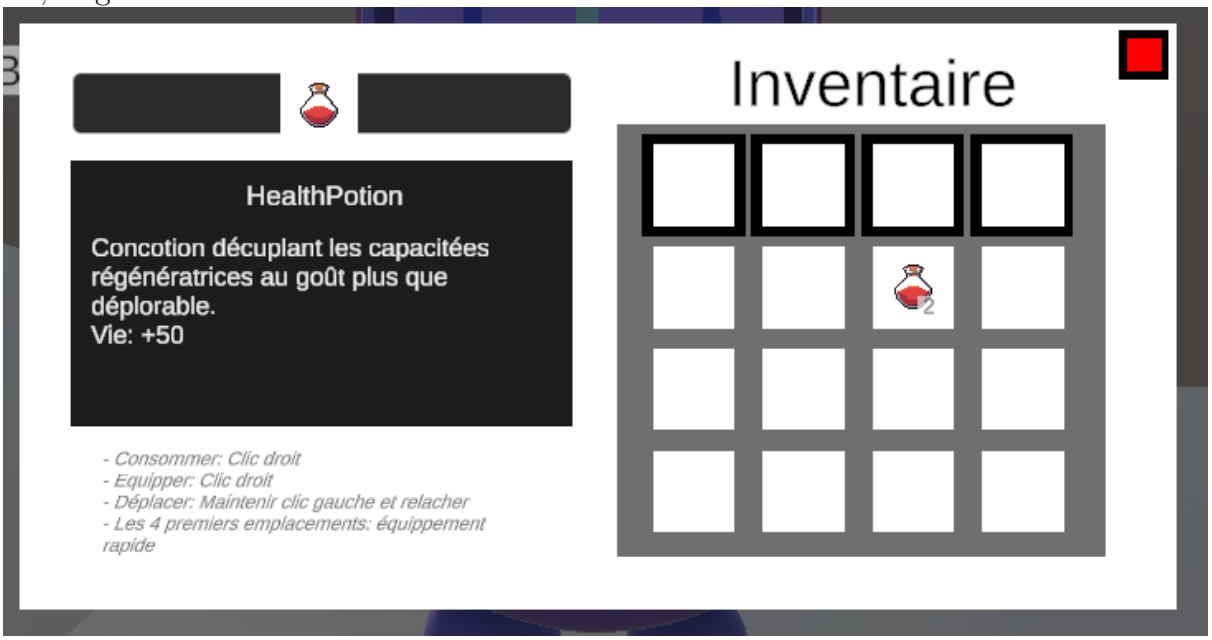
Pour l'invenataire je ne pourrais pas vous expliquer aussi précisément comment fonctionne l'inventaire.

Dans le cadre de notre projet de jeu vidéo, nous avons développé un système d'inventaire complet pour gérer les objets que le joueur peut collecter et utiliser.

L'inventaire est composé de deux parties principales : l'interface utilisateur (UI) et le contrôleur d'inventaire. L'interface utilisateur est responsable de l'affichage des objets et de la gestion des interactions de l'utilisateur, tandis que le contrôleur d'inventaire gère la

logique des objets et leur état.

L'interface utilisateur de l'inventaire est implémentée dans la classe `Invantaire`. Cette classe contient des références aux éléments UI nécessaires pour afficher les objets, ainsi que des méthodes pour initialiser l'inventaire, mettre à jour les données des objets, et gérer les interactions de l'utilisateur.



Initialisation de l'Inventaire UI

Lors de l'initialisation de l'inventaire UI, nous créons dynamiquement des éléments UI pour chaque emplacement d'objet dans l'inventaire. Chaque élément UI est associé à un emplacement spécifique dans l'inventaire et est configuré pour répondre aux interactions de l'utilisateur, telles que la sélection, le glisser-déposer, et les actions spécifiques aux objets.

Mise à Jour des Données de l'Inventaire

La méthode `UpdateData` est utilisée pour mettre à jour les données d'un emplacement d'objet spécifique dans l'inventaire. Cette méthode prend en entrée l'index de l'objet, l'image de l'objet, et la quantité de l'objet, et met à jour l'élément UI correspondant pour refléter ces données.

Gestion des Interactions de l'Utilisateur

L'interface utilisateur de l'inventaire gère plusieurs types d'interactions de l'utilisateur, notamment la sélection d'objets, le glisser-déposer, et les actions spécifiques aux objets. Les méthodes `HandleItemSelection`, `HandleBeginDrag`, `HandleSwap`, et `HandleShowItemActions` sont utilisées pour gérer ces interactions et mettre à jour l'état de l'inventaire en conséquence.

Le contrôleur d'inventaire est implémenté dans la classe `InventoryController`. Cette classe est responsable de la gestion de la logique des objets et de leur état. Elle contient des références à l'interface utilisateur de l'inventaire et aux données d'inventaire, ainsi

que des méthodes pour initialiser l'inventaire, mettre à jour l'interface utilisateur, et gérer les interactions de l'utilisateur.

Initialisation du Contrôleur d'Inventaire

Lors de l'initialisation du contrôleur d'inventaire, nous préparons l'interface utilisateur de l'inventaire et les données d'inventaire. Les objets initiaux sont ajoutés aux données d'inventaire, et l'interface utilisateur de l'inventaire est configurée pour répondre aux interactions de l'utilisateur.

Mise à Jour de l'Interface Utilisateur de l'Inventaire

La méthode `UpdateInventoryUI` est utilisée pour mettre à jour l'interface utilisateur de l'inventaire en fonction de l'état actuel des données d'inventaire. Cette méthode prend en entrée un dictionnaire représentant l'état actuel de l'inventaire et met à jour les éléments UI correspondants pour refléter ces données.

Gestion des Interactions de l'Utilisateur

Le contrôleur d'inventaire gère plusieurs types d'interactions de l'utilisateur, notamment la sélection d'objets, le glisser-déposer, et les actions spécifiques aux objets. Les méthodes `HandleDescriptionRequest`, `HandleSwapItems`, `HandleDraggind`, et `HandleItemActionRequest` sont utilisées pour gérer ces interactions et mettre à jour l'état de l'inventaire en conséquence.

2 Bilan

Tâches / Période	Prévu	Réaliser
3D / Assets	40 %	10 %
Sound design	5 %	80 %
UI	20 %	10 %
Site Web	40 %	90 %
Multijoueur	10 %	10 %
Gameplay	40 %	30 %
VFX	50 %	0 %
IA	30 %	10 %
Qualité	0 %	0 %

2.1 UI Futur

Dans le futur, nous prévoyons de développer une interface utilisateur (UI) complète pour notre jeu vidéo. Voici les principales étapes et fonctionnalités que nous envisageons

d'implémenter :

L'écran d'accueil sera la première interface que le joueur rencontrera. Il devra susciter l'enthousiasme tout en étant fonctionnel. Nous prévoyons d'inclure le nom du jeu, des options de menu telles que "Jouer", "Paramètres" et "Quitter". Le design visuel inclura une animation subtile ou une image statique représentant le thème du jeu.

Structure Recommandée

- **Logo/Titre** : Positionné au centre ou en haut de l'écran.
- **Menu Principal** : Une liste verticale ou horizontale claire avec des boutons bien espacés.
- **Design Visuel** : Animation subtile ou image statique représentant le thème du jeu.

La section des paramètres doit être facile à naviguer et organiser les options en catégories telles que "Audio", "Vidéo", et "Commandes". Chaque paramètre doit être accessible sans devoir parcourir de longues listes. Nous prévoyons d'utiliser des onglets ou des sections claires pour une navigation facile, des sliders pour ajuster les niveaux sonores, et des cases à cocher pour activer/désactiver des fonctionnalités.

Organisation

- **Onglets ou Sections Claires** : Pour une navigation facile.
- **Slider pour Volume** : Pour ajuster les niveaux sonores.
- **Case à Cohcer** : Pour activer/désactiver des fonctionnalités.

Ces éléments sont essentiels à l'écran de jeu. La barre de vie doit être visible sans distraire. Nous prévoyons d'utiliser des couleurs intuitives (épuisement en rouge, vie pleine en vert) avec des animations pour signifier la perte ou le gain. La position de la barre de vie sera en haut à gauche ou en bas au centre, selon les conventions du genre.

Position et Design

- **Position** : Haut gauche ou bas central, selon les conventions du genre.
- **Design** : Utiliser des couleurs intuitives (épuisement en rouge, vie pleine en vert) avec des animations pour signifier la perte ou le gain.

2.2 Combat et IA dans le futur

Le combat va être fait avec un système comme celui de monster hunter et pokemon donc ce sera un tour par tour. Nous utiliserons plusieurs IA une pour les combats contre les monstres durant le combat en tour par tour l'ia pour le mouvement des monstres et l'ia de PNJ qui seront présent dans les villes et villages.