

# Rapport de Projet

Journey Towards Dawn

Home Studio

Charles Delahousse

Maxence Jenn

Lino Develotte

Matthias Couturier

Benjamin Dubois



## Table des matières

<b>1</b>	<b>Introduction</b>	<b>4</b>
<b>2</b>	<b>Vue générale du projet</b>	<b>5</b>
2.1	Présentation des membres . . . . .	5
2.2	Outils utilisés . . . . .	7
2.3	Répartition des tâches . . . . .	7
<b>3</b>	<b>Réalisation du projet</b>	<b>8</b>
3.1	Menu . . . . .	8
3.1.1	Menu principal . . . . .	8
3.1.2	Menu des paramètres . . . . .	9
3.1.3	Menu InGame . . . . .	10
3.2	Graphismes . . . . .	11
3.2.1	Arrière plan . . . . .	11
3.2.2	Sprites . . . . .	12
3.2.3	Interface . . . . .	14
3.3	Mécaniques de jeu . . . . .	18
3.3.1	Niveaux de difficultés . . . . .	18
3.3.2	Sélection de personnages . . . . .	18
3.3.3	Buff et Débuff . . . . .	21
3.3.4	Salles . . . . .	21
3.3.5	Drag and Drop . . . . .	23
3.3.6	Aléas . . . . .	25
3.3.7	Journal de bord . . . . .	26
3.4	Écran de fin . . . . .	27
3.5	Intelligence Artificielle . . . . .	28
3.6	Musiques et Effets sonores . . . . .	30
3.7	Multijoueur . . . . .	31

3.8 Tutoriel . . . . .	35
3.9 Site Web . . . . .	37
<b>4 Conclusion</b>	<b>44</b>

## 1 Introduction

Dans le cadre de notre projet informatique, nous, l'équipe Home Studio, avons décidé de réaliser un jeu vidéo : Journey Towards Dawn.

Le monde humain stagne. Les ressources s'affaiblissent et le monde scientifique peine à faire des découvertes révolutionnaires. Mais un jour, des scientifiques découvrent la présence d'un artefact au fin fond de l'espace. Cet artefact est un mystère et beaucoup théorise sur son origine. Certains chercheurs prétendent que cet objet pourrait être la clé pour sauver l'humanité et le surnomment "Dawn Peak", traduit en "Pic de L'Aube".

C'est pourquoi une expédition spatiale est maintenue pour retrouver cet artefact. Le joueur se verra incarner une IA, du nom de "Wanderer", installée dans le vaisseau qui amènera une équipe de quatre scientifiques vers l'objet mystérieux.

Journey Towards Dawn est donc un jeu de type survie et gestion tour par tour. En effet, après avoir consolidé une équipe de quatre astronautes, le joueur se voit gérer un vaisseau consisté de 6 salles qui abrite son équipe. Des aléas vont menacer son vaisseau et son équipage. Il en est au joueur de prendre les précautions et actions nécessaires pour faire face à ces menaces.

Nous vous proposons ainsi de découvrir la réalisation du projet, nos joies et nos peines.

## 2 Vue générale du projet

### 2.1 Présentation des membres

#### — Maxence Jenn :

Depuis mon enfance, je suis passionné par les jeux vidéo, qu'ils soient sur console ou sur PC. Récemment, mon intérêt s'est orienté vers les aspects techniques de l'informatique, notamment la conception de jeux vidéo et de sites web. C'est pourquoi j'ai choisi d'intégrer l'EPITA, une école spécialisée dans le domaine de l'informatique, afin d'atteindre mes objectifs.

Lors de la réalisation de ce projet, j'ai participé à l'implémentation des musiques, des effets sonores, de l'intelligence artificielle et des différents menus de notre jeu.

#### — Matthias Couturier :

De tous mes hobbies, la programmation a toujours piqué ma curiosité. Pouvant s'appliquer dans d'innombrables domaines, c'est pour moi un champ d'étude indispensable à comprendre. C'est il y a de cela 3 ans que je m'y suis vraiment intéressé, notamment avec la spécialité NSI. Cela m'a permis d'obtenir certaines bases en apprenant le Python et en effectuant déjà quelques petits projets personnels ou pour l'école. Mais cela n'était pas suffisant, voulant une expérience plus complète dans ce domaine, l'EPITA m'est paru comme un choix évident.

Ayant toujours adoré dessiner et voulant développer mes compétences en programmation, j'étais responsable de la réalisation des différents éléments graphiques de notre jeu et j'ai participé à l'implémentation des différents menu et du multijoueur.

#### — Benjamin Dubois :

J'ai toujours été un grand fan de jeu vidéo, mais c'est à partir de la première que j'ai commencé à vraiment m'intéresser à la programmation de ces derniers, et après être partie dans le domaine de la science, je me suis rendu compte que cela ne m'intéressait pas et je me suis tourné vers le domaine de la programmation et c'est à ce moment que j'ai rejoint EPITA.

Durant ce projet informatique, j'étais responsable de l'implémentation du drag and drop, des fonctionnalités des personnages et des salles.

— **Charles Delahousse :**

Mon intérêt pour les jeux-vidéos a commencé tôt dans mon enfance, et s'est poursuivi jusqu'à ma participation à un tournoi e-sport pour le jeu Naraka : Bladepoint. Au long de ce cheminement, je n'ai cessé de me poser la question de la manière dont les jeux sont créés. C'est de là que ma curiosité pour la programmation a commencé et c'est pourquoi j'ai d'abord suivi un stage d'initiation à la programmation, qui a confirmé mon intérêt, avant d'intégrer l'EPITA. Grâce à mon expérience d'e-sport j'ai compris que l'effort solitaire ne suffit pas. Un projet en groupe nécessite de se faire confiance au sein de l'équipe constituée et de consolider stratégiquement nos points forts et faibles. Au travers d'un vote unanime, j'ai été désigné en tant que chef de groupe de l'entreprise Home Studio parce que je me sentais capable de mener ce leadership, j'en ai l'appétence, l'enthousiasme et je veux me frotter à cette responsabilité.

Lors de ce projet informatique, j'ai été responsable du GameDesign et j'ai participé à l'implémentation du tutoriel et des différents aléas.

— **Lino Develotte :**

J'ai découvert la programmation à l'âge de 9 ans avec l'outil scratch. Ma relation avec la programmation a cependant cessé à l'entrée au collège et au lycée où les matières liées à l'informatique étaient quasi inexistantes. J'ai par la suite découvert l'EPITA qui m'a tout de suite intéressé et redonné l'envie d'apprendre à coder. J'ai ainsi, pendant les dernières grandes vacances d'été, appris les bases de python pour rattraper un certain retard. J'ai pu, à l'aide de ma personnalité, de ma capacité à travailler en équipe et de mes connaissances en programmation, apporter ma vision à ce projet.

Durant ce projet, j'ai été responsable de la réalisation du site web et j'ai participé à l'implémentation du multijoueur.

## 2.2 Outils utilisés

Pour la réalisation de notre projet, nous avons mis en place plusieurs moyens de collaboration et de suivi.

Tout d'abord, nous avons utilisé GitHub afin de travailler en collaboration sur les mêmes fichiers et de résoudre les conflits éventuels.

Nous avons également utilisé Discord pour communiquer, échanger des idées et travailler. C'est aussi sur cette plateforme que nous mettons tous les avancements, les problèmes rencontrés et les choses à faire.

Pour l'aspect graphique de notre jeu, nous avons utilisé le logiciel Asperite permettant de réaliser nos propres graphismes.

Enfin pour l'aspect technique du jeu ( le code ), nous avons utilisé le langage de programmation C# sur l'éditeur Rider.

## 2.3 Répartition des tâches

	Matthias	Charles	Benjamin	Lino	Maxence
Aléas		R	S		
Actions Générales		S	R		
Buff et Débuff		R		S	
Menu	S				R
Santé des Personnages			R		S
Sprite Vaisseau	R	S			
Sprites Personnages	S	R			
Arrière Plan	R			S	
HUD	R	S			
Musiques			S		R
Effets Sonores				R	S
Site Web			S	R	
Intelligence Artificielle	S				R
Multijoueur Actions			R	S	
Multijoueur Connectivité				R	S

## 3 Réalisation du projet

### 3.1 Menu

#### 3.1.1 Menu principal

Dans un premier temps, nous avions commencé les différents menus qui composent notre jeu. Tout d'abord, nous avons fait un premier concept de menu principal comportant :

- Un bouton “Jouer” qui affiche le menu jouer pour commencer une nouvelle partie, reprendre la partie ou jouer en multijoueur
- Un bouton “Paramètres” qui affiche le menu Paramètres pour les changer
- Un bouton “Documentation” qui affiche le site web de notre jeu
- Un bouton “Credits” qui affiche une scène de credits pour remercier tous les participants, directs ou indirects, au projet
- Un bouton “Quitter” qui ferme le jeu



FIGURE 1 – Menu Principal

Puis lors de la 2ème soutenance, nous avions ajouté deux sections supplémentaires au menu principal du jeu, nécessaires pour accéder à certaines de ses fonctionnalités.

Nous avions deux approches possibles pour la gestion du menu : soit changer de scène à



chaque changement entre ses sections, soit regrouper l'ensemble des sections dans une seule scène et activer/désactiver les GameObjects en fonction des besoins. Pour une meilleure lisibilité et une navigation plus fluide lors du développement, nous avons choisi la seconde option.

De plus, lors de la première soutenance, nous n'avions pas implémenté certaines fonctionnalités dans le Menu. Donc pour ce qui est du Menu, nous avions rajouté quelques fonctionnalités qui n'étaient pas implémenté lors de la première soutenance :

- Le bouton “Documentation” dans le Menu Principal a été implémenté et ouvre le site web de notre jeu.

### 3.1.2 Menu des paramètres

Dans un premier temps, nous avions implémenté le menu des paramètres, obscurcissant un peu le menu principal, qui comporte :

- Un menu déroulant “Résolution” pour changer la résolution
- Une case à cocher “Plein Ecran” pour mettre en plein écran
- Un slider “Musique” pour changer le volume de la musique
- Un slider “Effets Sonores” pour changer le volume des effets sonores
- Un bouton pour fermer le menu des paramètres représenté par une croix rouge

Par la suite, nous avons décidé de retirer le menu déroulant ”Résoltuion”.

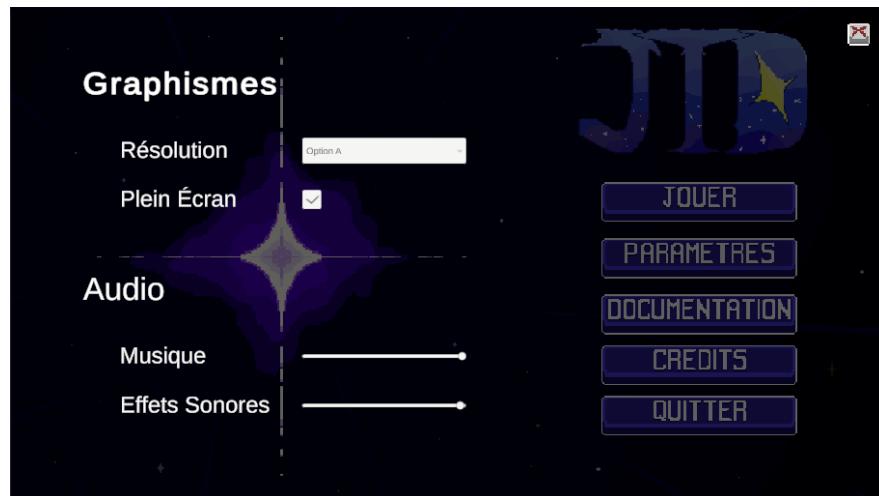


FIGURE 2 – Menu paramètres

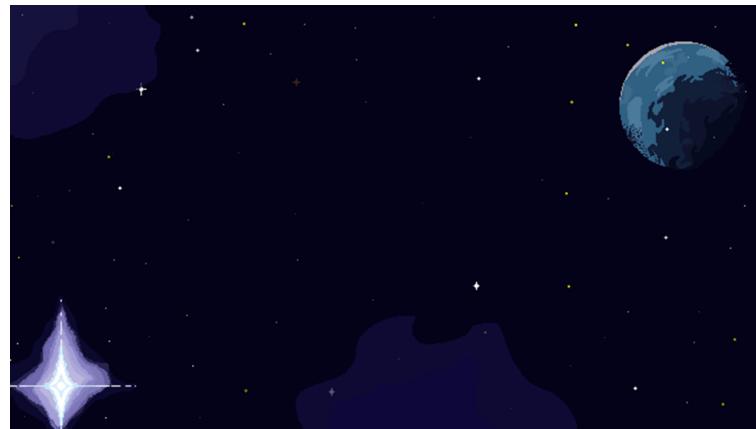
### 3.1.3 Menu InGame

Nous avons également ajouté un menu de pause nous permettant nous permettant d'accéder directement au paramètre depuis le jeu ou encore de revenir à l'écran titre si on le souhaite.



## 3.2 Graphismes

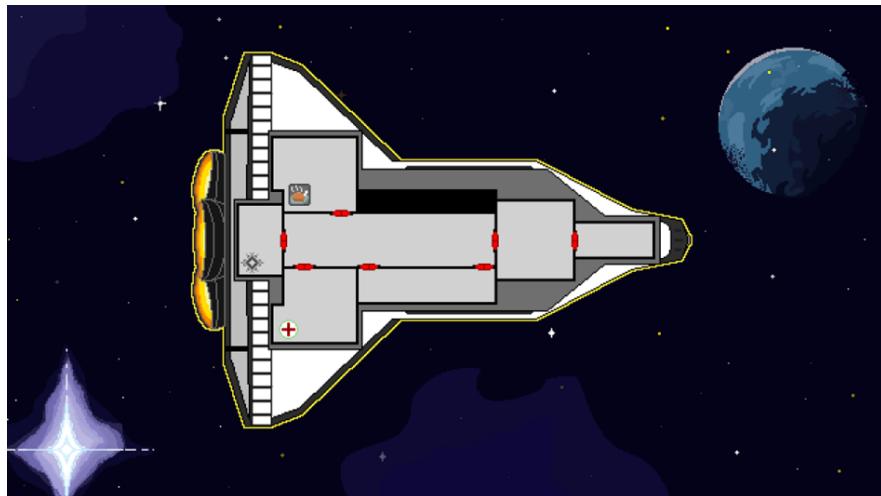
### 3.2.1 Arrière plan



En ce qui concerne les arrière-plans, nous voulions appuyer sur la nostalgie des joueurs vétérans et avons donc décidé un retour aux sources avec cet aspect 2D et en pixel art rappelant le genre des jeux rétro. Les couleurs utilisées pour ces arrières plans ne sont pas anodines étant toutes centrées autour du bleu. De ce fait, cela renforce l'ambiance de l'espace mais surtout fait en sorte que le joueur se concentre sur l'élément centrale du jeu qu'est le vaisseau qui lui possède des couleurs vives permettant de le distinguer du reste. Au départ, nous avions créé un arrière-plan contenant davantage d'étoiles mais cela rendait l'écran moins lisible, d'où la décision d'en réduire le nombre.

### 3.2.2 Sprites

Pour le vaisseau, nous avions le choix entre deux concepts. Le premier propose une coupe transversale du vaisseau avec un affichage clair de ses différentes salles. En revanche, cet affichage rend les déplacements des personnages et certains événements que nous voudrions inclure moins lisibles. Nous avons donc opté pour le second concept qui lui propose une vue de dessus, plus réaliste, où l'on distingue plus aisément l'état global du vaisseau, mais faisant perdre quelque peu en clarté notamment pour distinguer les objets et personnages. Grâce à cet aspect, le joueur peut facilement regarder les informations du personnage sélectionné tout en observant l'état global du vaisseau. Ainsi, cela rend la lecture du jeu plus simple et permet ainsi de dynamiser le gameplay.



Chaque personnage possède quatre sprites distincts :

- Un pour l'écran de sélection en début de partie



- Un pour sa représentation dans le vaisseau



- Un pour afficher son portrait dans le HUD



- Un pour la surbrillance lorsqu'il est sélectionné



Une question s'est notamment posée quant à la vue à adopter pour les personnages dans le vaisseau : de dessus ou de face. Afin de mieux les distinguer, nous avons opté pour une vue de face. Nous avons d'ailleurs dû nous y reprendre à deux fois pour créer celui dans le vaisseau, initialement légèrement trop grand.



FIGURE 3 – Avant

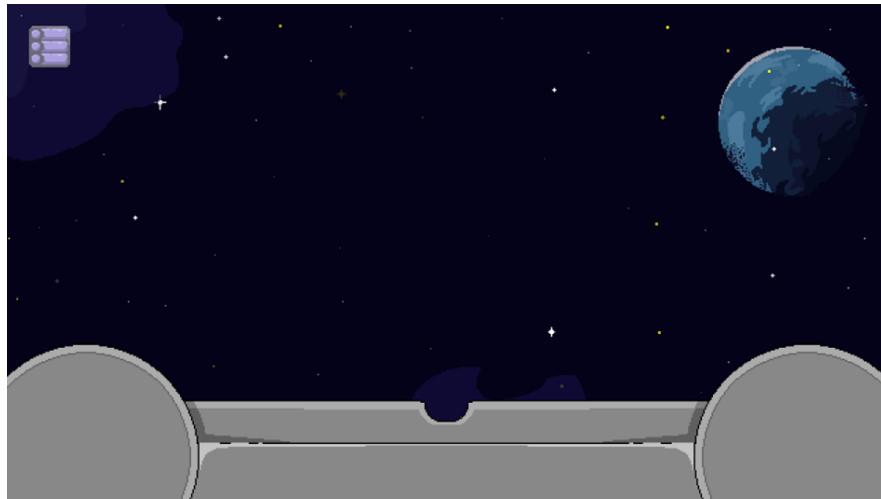


FIGURE 4 – Après

Enfin, nous avions initialement prévu d'animer les déplacements des personnages dans le vaisseau avec plusieurs sprites. Faute de temps, cette idée a été mise de côté, et nous avons opté pour un système de glisser-déposer à la place.

### 3.2.3 Interface

Dans un premier temps, après avoir créé des concepts arts de l'HUD, nous l'avions réalisé sur Aseprite. Il n'était cependant pas complètement terminé. En effet, elle ne comportait, en haut à gauche, qu'un unique bouton permettant d'ouvrir le menu de pause du jeu.



Nous avions constaté que le HUD occupait trop d'espace à l'écran. Sa taille a donc été réduite et son apparence ajustée en conséquence pour améliorer sa lisibilité.

Nous avions également ajouté les informations relatives au personnage, à savoir sa barre de santé physique, mentale et de faim. Il était initialement prévu d'utiliser des formes différentes du rectangle classique pour représenter ces barres de statistiques. Cependant, nous avons rencontré des difficultés techniques : la jauge de remplissage se déformait lorsque sa valeur était modifiée. Ne souhaitant pas investir trop de temps sur cet aspect, nous sommes finalement revenus à une forme rectangulaire. Enfin, un bouton permettant de mettre fin à la journée en cours a été ajouté.

Tous ces éléments ont été organisés en plusieurs couches de sprites afin de faciliter l'implémentation de l'UI.



De nombreux ajouts d'interface utilisateur (UI) et d'éléments graphiques ont été réalisés au cours du projet. Tout d'abord, le bouton permettant de mettre fin à la journée a été refait pour mieux s'intégrer à la forme de l'interface. De même, les sprites complets des personnages présent dans l'écran de sélection, qui étaient auparavant peu soignés, ont été améliorés pour gagner en qualité visuelle.



Des éléments d'interface utilisateur ont été ajoutés afin de permettre la sélection directe des personnages en cliquant dessus, ainsi que pour afficher leurs malus et bonus. Chaque personnage dispose désormais d'un bouton permettant d'afficher ses compétences, fonctionnalité auparavant accessible uniquement depuis l'écran de sélection des personnages. De plus,

lorsque la souris survole l'icône d'un malus ou d'un bonus, la description de son effet apparaît, offrant ainsi plus de clarté au joueur.



Un sprite a également été conçu pour la nouvelle fonctionnalité de notre jeu, le journal de bord, nous permettant d'afficher les données des événements de manière claire et organisée.



Les salles possèdent désormais leur propre HUD, qui affiche leur état actuel lorsque l'utilisateur survole leur icône avec la souris.



Enfin, le HUD du joueur affiche désormais deux informations supplémentaires : le nom de l'astronaute et son taux de réparation, données qui étaient auparavant accessibles uniquement depuis l'écran de sélection des personnages. Des sprites spécifiques ont également été créés pour représenter la mort d'un personnage, remplaçant son sprite par une tombe.

De plus, lorsqu'une salle est détruite, tous les personnages qui s'y trouvent sont expulsés vers la salle centrale du vaisseau, et il n'est plus possible de faire de glisser-déposer sur cette salle.

### 3.3 Mécaniques de jeu

#### 3.3.1 Niveaux de difficultés

Le jeu se différencie en trois niveaux de difficultés :

- **Facile** : partie en 25 tours avec seulement des aléas faciles. L'idée est l'immersion du joueur et la prise en main du jeu.
- **Normale** : partie en 50 tours avec des aléas faciles et normaux. L'expérience classique de Journey Towards Dawn.
- **Difficile** : partie en 50 tours avec des aléas faciles, normaux et difficiles. Expérience réservée aux vétérans et joueurs qui souhaitent un défi.

#### 3.3.2 Sélection de personnages

Chaque personnage possède ces caractéristiques suivantes :

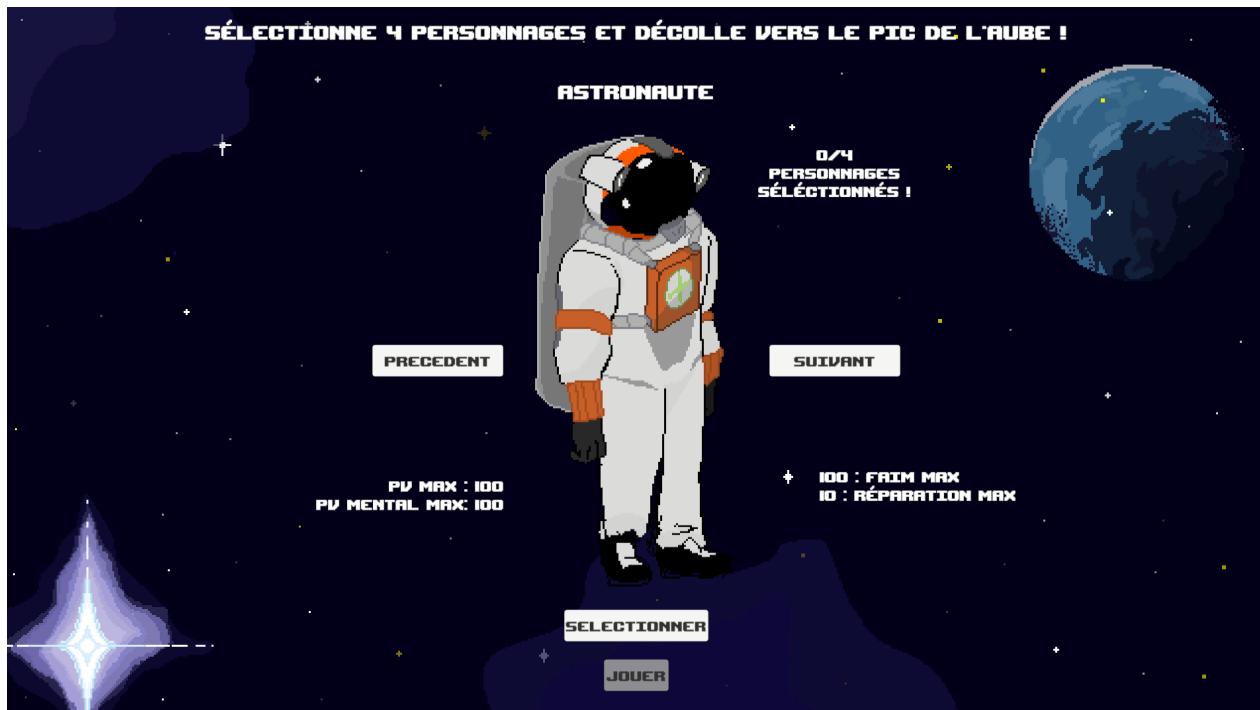
- Une barre de points de vie
- Une barre de points de santé mentale
- Une barre de faim
- Un effet spécial unique à chaque personnage

Voici la liste de tous les personnages qui sont disponibles :

- **Médecin** : si il se trouve dans l'infirmerie, il soigne tous les membres d'équipage
- **Ingénieur** : si il se trouve dans la salle des machines, il répare toutes les salles avec son taux de réparation
- **Pilote** : si il se trouve dans le poste de pilotage, il réduit les chances d'apparition de l'évènement "Débris"
- **Chef cuisinier** : si il se trouve dans la cantine, il annule la perte de satiété de tout l'équipage
- **Capitaine** : si il se trouve dans la salle de contrôle, il révèle les chances d'apparition des aléas à venir

- **Chill Guy** : si il se trouve dans le dortoir, il régénère la santé mentale de l'équipage
- **Survivor** : il possède des points de vie et de santé mentale augmentés

Après avoir choisi la difficulté du jeu, le joueur accède à la sélection des personnages qu'il devra guider lors de la mission vers le Pic de l'Aube. Le joueur doit choisir un total de quatre personnages parmi une liste de huit astronautes, chacun possédant des statistiques spécifiques.



Cette scène est composée de plusieurs boutons interactifs et de textes qui s'adaptent aux interactions du joueur.

Grâce aux boutons “Suivant” et “Précédent”, le joueur peut parcourir la liste des astronautes disponibles. Chaque astronaute est identifié par un nom et un sprite uniques, accompagnés de ses statistiques spécifiques :

- **Santé Physique**
- **Santé Mental**

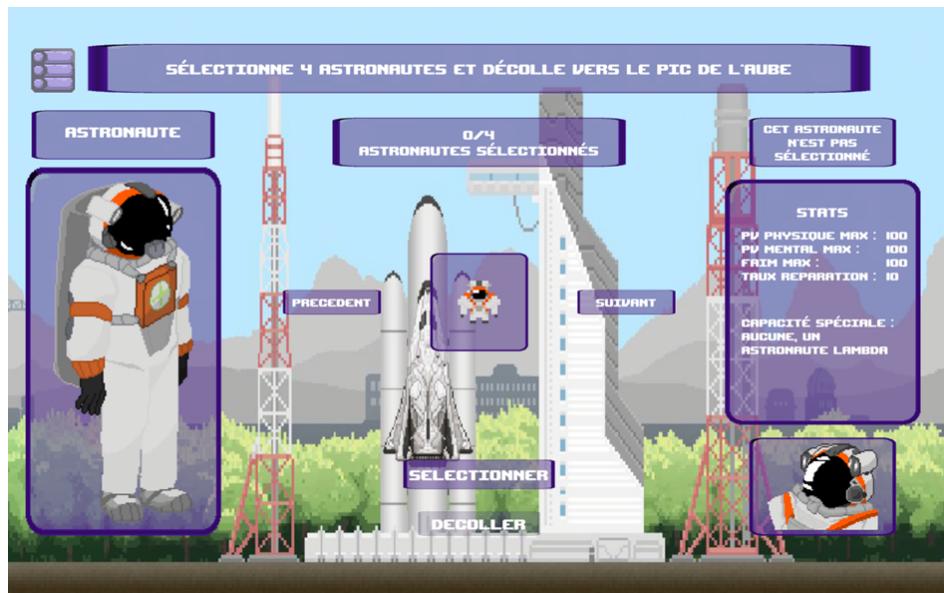
- Faim
- Points de Réparation

Une liste initialement vide est prévue pour enregistrer les personnages sélectionnés. Cette liste constitue l'équipe principale du joueur. Un compteur affiche le nombre de personnages sélectionnés.

Lorsqu'un personnage est ajouté, le compteur augmente d'une unité, le bouton "Sélectionner" est remplacé par le bouton "Retirer", et un texte de confirmation s'affiche en haut à gauche du sprite du personnage. Si le joueur choisit de retirer un personnage, celui-ci est supprimé de la liste, le compteur diminue d'une unité, le texte de confirmation disparaît et le bouton "Sélectionner" réapparaît.

Une fois que quatre personnages sont sélectionnés, le bouton "Jouer" devient actif. En cliquant dessus, la liste des personnages choisis est enregistrée, et le joueur est redirigé vers la scène principale du jeu où son équipage devra faire face aux dangers de l'espace.

La scène des sélections des personnages a eu un coup de renouveau pour présenter un départ réel à partir de la surface de la Terre.



Dorénavant nous pouvons voir le portrait ainsi que le mini-sprite qui représenteront l'as-

tronaute sur le vaisseau et l'HUD du joueur. De plus, les statistiques du personnage sont illustrées proprement sur la droite de l'écran avec en plus la capacité spéciale, et unique, de celui-ci spécifiée. Enfin, nous avons rajouter l'accès à un menu de pause placé en haut à gauche de l'écran comme pour la scène principale du jeu. Ce menu permet de revenir au menu principal en cas d'indécision et aussi d'accéder aux paramètres du menu principal.

### 3.3.3 Buff et Débuff

Un personnage peut être affecté par des effets temporaires appelés buffs ou debuffs, en fonction de leur impact. Dans un premier temps, nous avions fait une liste des différents bonus et malus que l'équipage pouvait recevoir. Ces effets sont également représentés sous forme d'une énumération :

- Inspiré
- Anxieux
- Gourmand
- Affamé

Un personnage peut cumuler plusieurs buffs et debuffs simultanément grâce à une liste de paires (type d'effet, durée). Ces effets sont attribués via certains événements aléatoires. Cependant, un personnage ne peut pas recevoir deux fois le même effet consécutivement.

### 3.3.4 Salles

Lorsqu'une salle est occupée, elle active un effet à l'équipage. Plusieurs équipiers peuvent être mis dans une même salle pour réparer plus rapidement mais ceci ne rendra pas plus efficace l'effet de la salle.

Chaque salle possède une barre de points de vie allant de 0 à 100. Si la barre atteint 0, la salle est détruite et est inaccessible pour le reste de la partie.

Voici la liste des différentes salles ainsi que leurs effets :

- **Poste de pilotage** : baisse les chances de l'aléa "Heurté des débris"

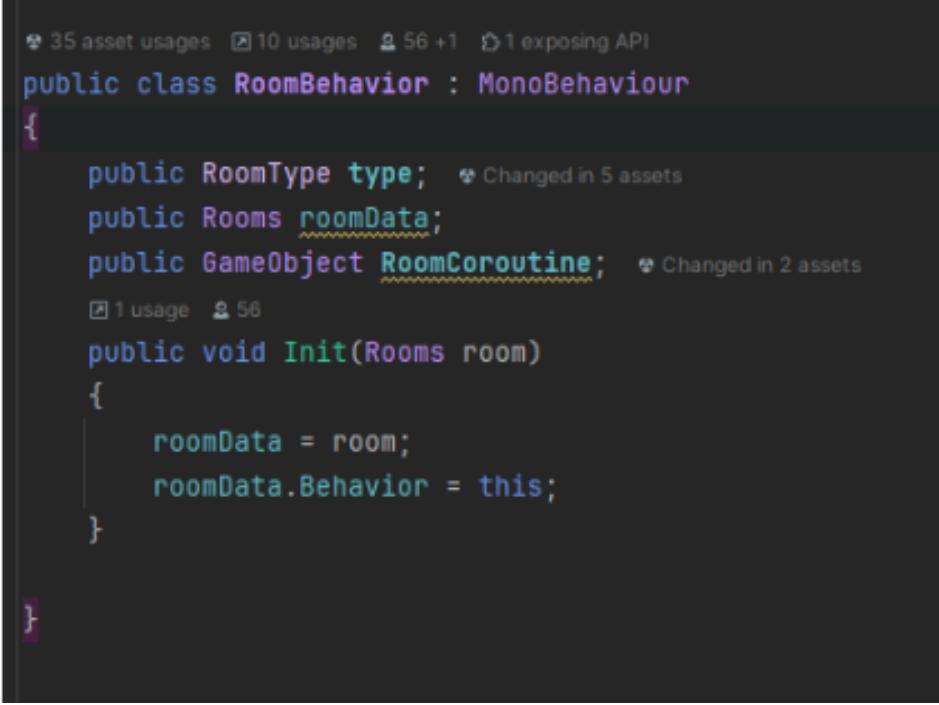
- **Poste de contrôle** : révèle l'aléa avec la plus grande chance d'apparition
- **Cantine** : nourrit les personnages assignés à cette salle
- **Infirmerie** : soigne les points de vie des personnages assignés à cette salle
- **Dortoir** : soigne les points de santé mentale des personnages assignés à cette salle
- **Salle des machines** : permet de réduire les dégâts pris par toutes les salles

Si toutes les salles du joueur sont détruites, le joueur perdra.

Durant la dernière phase de développement du jeu, nous avons implémenté les effets des salles. Un problème majeur a été rencontré lors de l'implémentation du système de réparation des salles : malgré la présence d'un astronaute affecté à cette tâche, aucune salle n'était réparée. Après analyse, il est apparu que le script logique Rooms, qui contient les informations métier de chaque pièce (solidité, occupants, etc.), n'était pas correctement relié aux objets présents dans la scène Unity. Avant de pouvoir développer les fonctionnalités propres à chaque salle, il a été nécessaire de stabiliser l'infrastructure logicielle qui les gère. Pour corriger cela, un composant Unity nommé RoomBehavior a été conçu. Ce script, attaché à chaque GameObject représentant une salle, permet d'établir une connexion bidirectionnelle entre l'objet Unity et son équivalent logique. Sa méthode Init(Rooms room) permet d'initialiser cette liaison dès le chargement du niveau.

Grâce à cette structure, les différentes fonctions du jeu peuvent désormais accéder aux données des salles via leurs objets Unity, ce qui a permis de faire fonctionner la gestion des réparations, et a ouvert la voie à l'implémentation des effets passifs spécifiques à chaque salle.

Une fois ces fondations stabilisées, les effets passifs propres à chaque salle ont pu être intégrés. Ces effets s'exécutent automatiquement à la fin de chaque journée, et agissent sur tous les personnages vivants se trouvant dans la salle en question. Chacune de ces fonctionnalités est encapsulée dans une fonction dédiée, appelée depuis le gestionnaire de fin de journée.



```
⌘ 35 asset usages ⌘ 10 usages ↳ 56 +1 ⌘ 1 exposing API
public class RoomBehavior : MonoBehaviour
{
    public RoomType type; ⌘ Changed in 5 assets
    public Rooms roomData;
    public GameObject RoomCoroutine; ⌘ Changed in 2 assets
    ↳ 1 usage ↳ 56
    public void Init(Rooms room)
    {
        roomData = room;
        roomData.Behavior = this;
    }
}
```

### 3.3.5 Drag and Drop

Dans le développement de Journey Towards Dawn, nous avions d'abord envisagé un système permettant aux joueurs d'assigner les astronautes aux salles du vaisseau à travers une simple sélection. L'idée initiale était de permettre au joueur de cliquer sur une salle, puis sur un astronaute, afin de l'y affecter automatiquement. Ce choix se basait sur une logique de gestion simplifiée et s'appuyait sur l'interaction classique d'un jeu de stratégie où les unités sont déplacées par des commandes directes plutôt que par un glissement manuel.

Cependant, au fil des tests, cette approche s'est révélée peu intuitive et trop rigide. L'interaction manquait de fluidité et obligeait le joueur à effectuer plusieurs actions successives sans véritable feedback visuel immédiat. Nous avons alors opté pour une approche plus naturelle en implémentant un système de Drag and Drop. Ce dernier permet au joueur de glisser directement les astronautes d'un endroit à un autre, offrant une meilleure ergonomie et un meilleur ressenti d'interactivité.

Le système repose sur deux scripts principaux : DragDrop, qui gère le comportement des astronautes en tant qu'objets déplaçables, et DropZone, qui régule la réception et l'assигnation

des astronautes aux emplacements dédiés dans chaque salle. L'objectif est de permettre le déplacement libre des astronautes tout en garantissant qu'un slot ne puisse contenir qu'un seul astronaute à la fois.

L'objectif du script DragDrop est de permettre à un astronaute d'être déplacé librement à l'aide de la souris et de s'intégrer correctement à une salle lorsqu'il est relâché. Ce script repose sur les interfaces IBeginDragHandler, IDragHandler et IEndDragHandler, fournies par Unity, qui permettent de détecter respectivement le début du glissement, le déplacement et le relâchement de l'objet.

Dès le lancement du jeu, chaque astronaute enregistre sa position d'origine et son parent dans la hiérarchie. Cette information est cruciale car elle permet de replacer l'astronaute à sa position initiale s'il n'est pas déposé dans une salle valide.

Le script DropZone est responsable de la gestion des salles et de la validation des emplacements disponibles pour chaque astronaute. Chaque salle contient plusieurs slots, qui sont des positions fixes où les astronautes peuvent être assignés. Si aucun slot valide n'a été détecté, l'astronaute est simplement replacé à sa position initiale ce qui empêche les erreurs de placement.

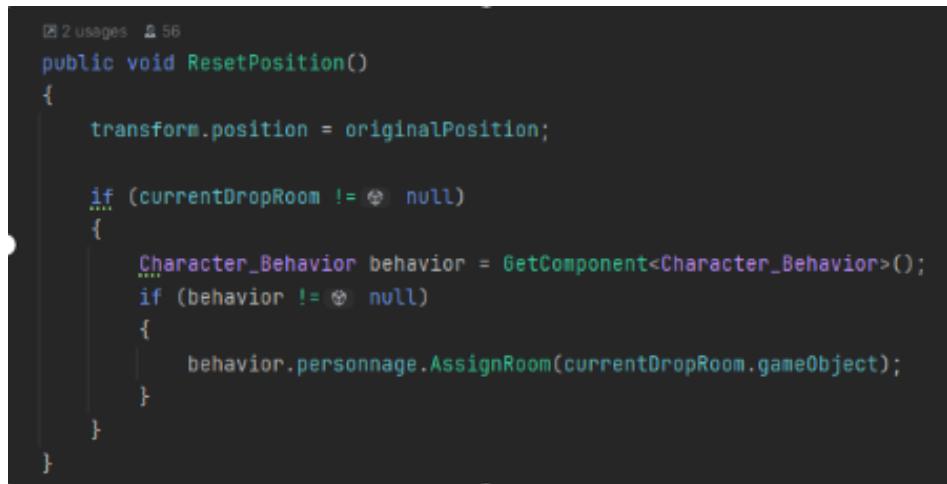
Lors des premiers tests, nous avions remarqué qu'un slot devenait inutilisable après avoir été occupé une première fois. Même lorsqu'un astronaute était retiré de son emplacement, le slot était toujours marqué comme occupé, empêchant d'y placer un autre astronaute.

Ce problème venait du fait que notre gestion des slots ne mettait pas à jour leur état lorsqu'un astronaute était retiré. Nous avions bien une structure permettant de stocker quel astronaute était dans quel slot, mais nous ne la vidions pas correctement lorsqu'un astronaute était déplacé ailleurs. Nous avons corrigé ce problème en introduisant une méthode dans le script DropZone. Cette méthode permet de libérer un slot lorsqu'un astronaute quitte la salle.

Durant la dernière phase de développement, nous avons corrigé quelques problèmes. L'un des premiers problèmes rencontrés résidait dans le système de glisser-déposer des astronautes au sein des salles. Au lancement d'une partie, aucun personnage n'était initialement affecté à un emplacement (ou "slot"), ce qui permettait aux joueurs de superposer plusieurs personnages dans une même case. De plus, lorsque la méthode ResetPosition était appelée, notamment après une interaction invalide, les personnages n'étaient plus correctement associés à leur emplacement d'origine.

Ce comportement a été corrigé de deux manières :

D'une part, en initialisant explicitement les positions de départ des personnages dans leur slot respectif. D'autre part, en s'assurant que la méthode ResetPosition réaffecte bien les liens entre les personnages et les emplacements.



### 3.3.6 Aléas

Dans Journey Towards Dawn, les dangers de l'espace prennent la forme d'événements aléatoires. C'est la mécanique principale de notre jeu. À chaque fin de tour, l'équipage peut être confronté à un événement positif ou négatif.

Dans un premier temps, nous avions fait une liste de tous les évènements aléatoires qui sont susceptibles d'apparaître durant la partie :

- Heurté des débris : les salles perdent toutes des points de vie

- Fuite d'oxygène : les personnages perdent tous des points de vie
- Rumeur négative : applique le debuff « Anxieux » à un nombre aléatoire d'équipiers qui dure un certain nombre de tours.
- Planète Food : applique le debuff « Gourmand » à un nombre aléatoire d'équipiers qui dure un certain nombre de tours.
- Sonde spatiale : Choix à risque. Une chance sur deux. Peut être ignorer.

Effet positif : découverte de données utiles. Attribue le buff « Inspiré » à toute l'équipe.

Effet négatif : Découverte de l'explosion d'une planète. Applique le debuff « Anxieux ».

- Découverte station spatiale : Choix à risque. Une chance sur deux. Peut être ignorer.

Effet positif : l'équipage de la station est en vie. L'équipe est nourrie à max, guérie à max et est attribuée le buff « Inspiré ».

Effet négatif : l'équipage de la station a disparu. L'équipe gagne le Debuff « Anxieux »

- Découverte Lola : Le joueur peut adopter un chat extraterrestre. Peut être ignorer.

Effet positif : néglige la baisse de l'HP mental

Effet négatif : endommage la cantine. Si la cantine est détruite, Lola mange un par un les membres de l'équipage.

- Mini astéroïde : vise et endommage sévèrement une salle

De plus, les effets des aléas sont influencés par la difficulté choisie par le joueur. Lorsque le joueur sélectionne une difficulté, des fonctions appropriées sont exécutées. Elles initialisent la difficulté, le nombre de tours à survivre ainsi qu'une liste d'événements.

### 3.3.7 Journal de bord

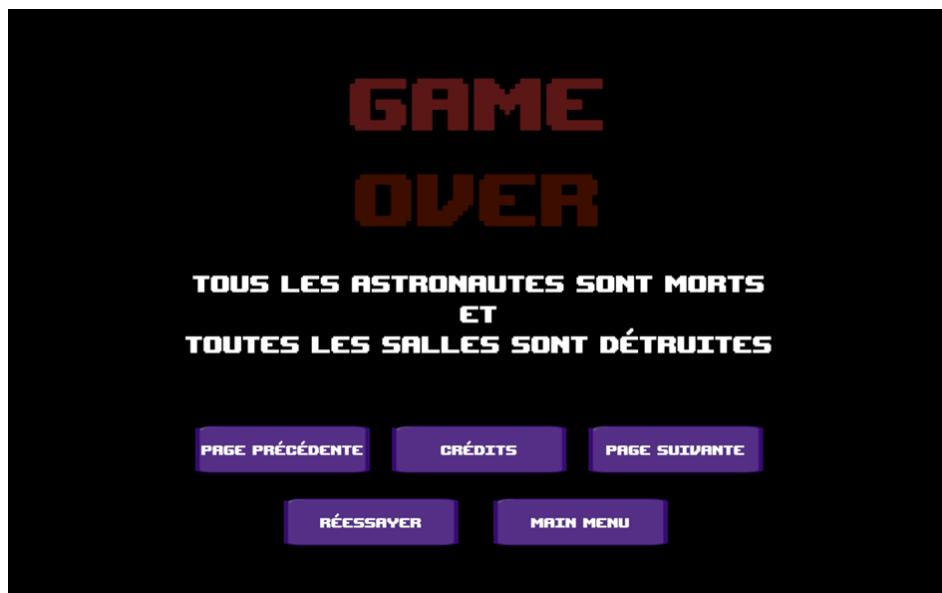
Nous avons créé une nouvelle interface, le journal de bord. Il permet au joueur d'obtenir de nombreuses informations que ce soit à propos des événements à venir, ou encore de ce qu'il s'est passé le jour précédent. L'ayant ajouté tardivement, ce fut fastidieux de revenir

sur chaque ligne de codes qui à un effet concret dans la partie pour enregistrer l'information et l'ajouter dans le journal. Chaque astronaute possède ainsi le résumé de ce qu'il a fait ou subit le jour précédent (perte de vie, réparation, etc.) ce qui permet de toujours comprendre pourquoi ses statistiques ont évolué de tel ou tel manière. Ce journal liste aussi l'événement s'étant produit et son effet sur les astronautes et salles, c'est donc un élément important du gameplay à toujours consulter. C'est notamment en codant ce journal de bord que nous avons réalisé de nombreuses erreurs présente dans la priorité d'exécution des codes, car nous pouvions enfin avoir sous les yeux ce qu'il se passait. Par exemple, on s'est rendu compte que des personnages morts ou des salles détruites pouvaient encore recevoir des événements, il a donc fallu réorganiser notre code.

### 3.4 Écran de fin

Après un certain nombre de jours de survie, ou en raison de conditions spécifiques, le joueur est conduit à la scène EndGame. Cette scène affiche si le joueur a atteint le Pic de l'Aube ou s'il a perdu. La victoire est obtenue en survivant le nombre de jours requis, selon la difficulté choisie. En revanche, si tous les membres de l'équipage du joueur sont morts, ou si toutes les salles du vaisseau sont détruites, celui-ci perd et se retrouve face à un écran de Game Over.

Le joueur peut par la suite, défiler la page pour observer les survivants ou victimes de l'expédition vers le Pic de l'Aube. En cas de défaite, le joueur peut réessayer sa partie à nouveau en se faisant ramener vers la scène de sélection de personnages, avec une chance de refaire son équipage. Sinon il peut se diriger vers les crédits ou le menu principal.



### 3.5 Intelligence Artificielle

Dans Journey Towards Dawn, nous voulions que l'intelligence artificielle soit en charge des aléas qui influenceront la partie. Selon la difficulté choisie, elle analyse le progrès du joueur et sa facilité ou non à progresser. Si le joueur s'en sort facilement : l'IA devait choisir des aléas plus sévères qui le mettra en proie à une plus grande réflexion et l'amènera à prendre des décisions plus importantes. Si au contraire, le joueur semble avoir du mal : l'IA aura plus tendance à l'aider en réduisant la dangerosité des événements.

Au fil des discussions entre membre de l'équipe, nous avions modifié légèrement ce système. Nous avons gardé le fait qu'elle analyse la situations du joueur (statistiques des personnages, points de vie des salles et progression dans la partie) mais au lieu de modifier la difficulté des évènements, nous avons décidé d'augmenter plus ou moins les chances d'apparition des aléas selon la situation du joueur.

Donc, dans Journey Towards Dawn, l'intelligence artificielle s'occupe des chances d'apparition des évènements. Pour ce faire, Elle analyse toutes les statistiques des personnages choisis par les joueurs (les points de vie, la faim et les points de santé mentale) ainsi que les points de vie de toutes les salles pour pouvoir augmenter les chances d'apparition des évènements en fonction de la situation du joueur. Chaque statistique possède une plage de valeurs divisé en quelques intervalles. Selon l'intervalle dans lequel se trouve la somme des valeurs de la statistique, cela va augmenter plus ou moins la valeur IA établie précédemment.

Nous sommes revenus sur notre système d'aléatoire pour l'exécution des événements, qui n'était pas optimal. Les mêmes événements revenaient trop souvent en boucle, et il nous était impossible d'afficher leur taux d'apparition, car le système initial ne le permettait pas. Nous avons donc opté pour un système d'aléatoire pondéré, offrant une approche à la fois plus simple et plus efficace.

Désormais, notre intelligence artificielle ajuste dynamiquement les « poids » attribués à chaque événement en fonction des statistiques de la partie et du niveau de difficulté sélectionné. Cela permet de moduler leur fréquence d'apparition de manière cohérente et équilibrée. Et cette fois ci, c'est un problème inhabituel auquel nous avons dû faire face, non pas de la programmation, mais du game design. Nous avons constaté qu'il était particulièrement difficile d'équilibrer l'apparition des événements, en raison du grand nombre de paramètres à prendre en compte. Cette complexité nous a conduits à affiner nos réglages à travers de nombreux tests, dans le but d'obtenir les meilleurs résultats possibles. Ce travail d'ajustement nous a même amenés à revoir l'équilibrage de certains malus, bonus, et compétences de personnages, afin d'obtenir un équilibre cohérent.

```
// Analyse des HP totaux des personnages

if (CharactersHP >= 100 && CharactersHP < 200)
{
    IA += 1;
}
else if (CharactersHP >= 200 && CharactersHP < 300)
{
    IA += 1.75;
}
else if (CharactersHP >= 300 && CharactersHP <= MaxCharactersHP)
{
    IA += 2.5;
}
```

FIGURE 5 – Exemple d'analyse d'une statistique

### 3.6 Musiques et Effets sonores

Pour l'implémentation des musiques et des effets sonores, nous voulions attendre que notre jeu prenne forme afin d'adapter le choix des musiques et des effets sonores en conséquence.

Donc pendant la dernière phase de travail, nous avons ajouté les éléments sonores à notre jeu. Pour ce qui est du menu principal, nous avons sélectionné deux musiques qui correspondent avec le côté rétro de notre jeu. Nous avons fait un système qui permet de lire en boucle ces musiques avec un léger intervalle de temps entre les musiques pour que le changement de musique ne soit pas trop brutal.

Lorsque le joueur sera amené à choisir ses personnages pour sa partie, une musique calme et mystérieuse sera joué en boucle mettant directement le joueur dans l'ambiance de notre jeu.

Enfin lorsque le joueur sera dans sa partie, deux musiques calmes et mystérieuses seront joués en boucle avec le même système que le menu principal. Ces musiques permettent d'immerger totalement le joueur durant sa partie.

Pour les effets sonores, ils seront principalement joués lorsque le joueur appuiera sur les différents boutons qui composent notre jeu.

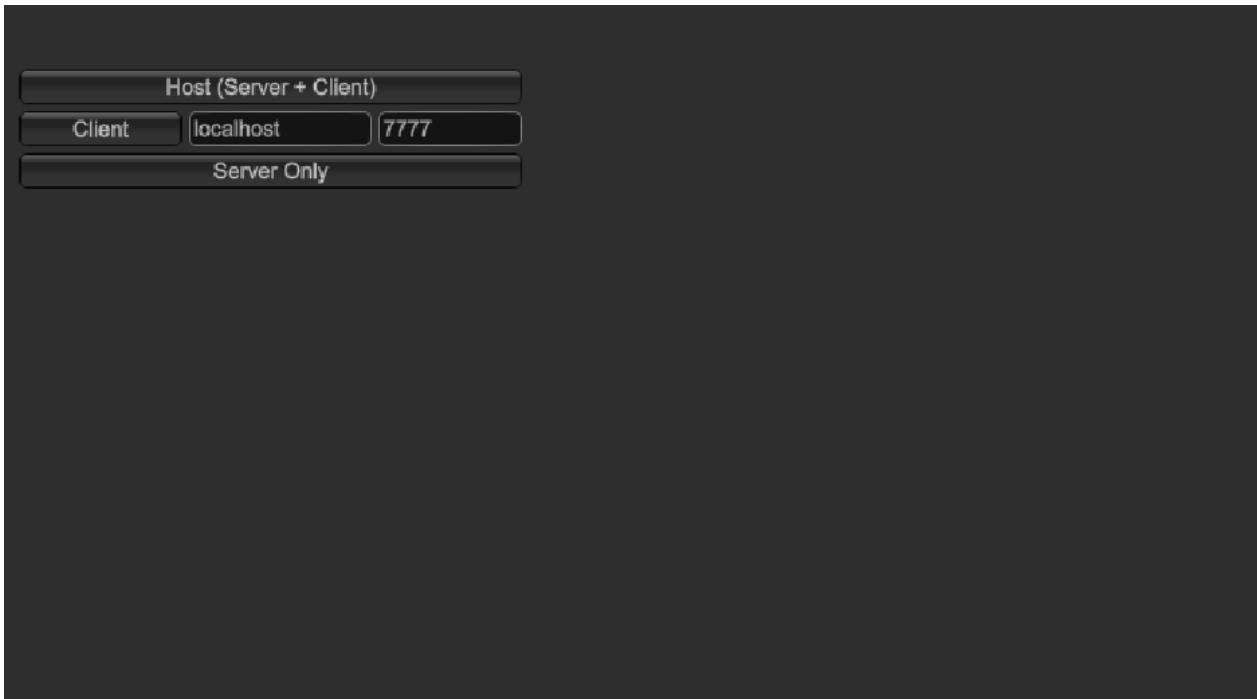
### 3.7 Multijoueur

Journey Towards Dawn propose un mode multijoueur qui se présente sous la forme d'une course compétitive dans laquelle le premier joueur à atteindre l'artefact l'emporte. Dans ce mode, l'histoire est réitérée. Chacun des joueurs assemblera son équipe et devra, tout comme dans le mode solo, garder son équipage en vie jusqu'à la fin du voyage ou lorsqu'il n'y a plus d'autres équipages adverses en vie. Pour rendre la partie plus intéressante et captivante, nous avons décidé d'incorporer dans leur vaisseau des moyens pour saboter l'adversaire. Un système sera donc mis à disposition pour le joueur lui permettant de choisir un aléa parmi une liste qui affectera l'adversaire de son choix.

Dans un premier temps, pour implémenter le multijoueur, nous avions regarder des tutos sur YouTube, notamment celle de « TUTO UNITY FR » qui a opté pour l'utilisation d'un asset multijoueur pas présent de base sur Unity, Mirror. Ainsi, nous avons décidé de faire de même car l'asset nous paraissait d'apparence être plutôt simple d'utilisation et qu'il correspondait à nos critères pour le multijoueur.

Ainsi nous avons tout d'abord créé un « Network manager » qui est l'élément le plus important pour pouvoir joueur à plusieurs avec Mirror. Par la suite, nous avons créé une prefab d'un « game object » avec le component « Network Identity » et une caméra qui sera le joueur et un « game object » avec le component « Network Start Position » qui permettra au « Network Manager » de le reconnaître comme un point d'apparition de la prefab joueur. Par la suite, nous avons paramétré le « Network Manager » en mettant par exemple le nombres de joueurs max pouvant rejoindre la partie à 4, en choisissant le mode de transport multijoueur (KCP) ou encore en remplissant le compartiment « Player Prefab » par notre joueur.

Finalement, nous avons créé une scène qui est le lobby dans lequel les joueurs choisiront de soit rejoindre un serveur déjà existant ou de créer un serveur à l'aide d'un HUD qui est un component de « Network Manager ».



Cependant lors de la réalisation du multijoueur nous avions rencontré de nombreux problèmes, principalement dû aux canvas que nous avons utilisé pour les sprites du vaisseau ce qui nous a fait perdre énormément de temps sur l'avancement de celui-ci.

Par exemple, dans les canvas sur Unity, il y a une section « event mode » et une section « render mode » où il faut insérer une caméra et choisir un mode de rendu pour celle-ci. Cependant lorsque qu'une caméra est insérée dans cette emplacement, cela rend impossible l'utilisation de toute autre caméra. Ainsi, lorsqu'un joueur apparaissait, l'écran devenait tout bleu. Malheureusement, nous n'avons pu découvrir la cause de ce problème (le canva) qu'après plusieurs jours de recherche.

Finalement, durant la dernière phase de travail, le mode multijoueur a été difficile à réaliser et plusieurs ajouts et suppression ont dû être réalisé.

Dans un premier temps, nous avons remplacé le GameObject instancié par Mirror par le Canvas utilisé dans le mode solo.

Ensuite, nous avons modifié le flux de navigation après avoir cliqué sur le bouton multijoueur. Initialement, les joueurs étaient directement envoyés vers la scène de création ou de

connexion à un serveur. Nous avons ensuite tenté de faire transiter les joueurs par la scène de sélection des astronautes avant d'atteindre cette étape. Cependant, cela s'est révélé très compliqué à gérer : lorsqu'un joueur interagissait avec un bouton, l'action était déclenchée pour tous les autres joueurs connectés. Pour éviter ce problème, nous avons revu l'ordre des scènes. Désormais, les joueurs accèdent d'abord à la sélection des astronautes, chacun de manière indépendante, puis sont dirigés vers la scène permettant de rejoindre ou de créer un lobby multijoueur. Ce nouvel ordre permet une expérience plus fluide et évite les conflits d'interactions.

Mais malgré ces changements, trop nombreux problèmes ont persisté, nous forçant à repartir de zéro. Nous avons donc opté pour une nouvelle architecture en séparant notre jeu en deux scènes distinctes : une première pour le lobby, où les joueurs peuvent se connecter ou héberger la partie, et une seconde où se déroule la partie. Mais même en suivant un tutoriel, le codage s'est révélé particulièrement déroutant : Il a fallu override de nombreuses fonctions du NetworkManager de Mirror afin de concevoir notre propre gestionnaire réseau personnalisé, ce qui représentait une quantité conséquente de concepts techniques à assimiler d'un coup. Nous avons ensuite créé un prefab représentant le joueur dans le lobby, et un autre utilisé une fois la partie lancée. L'autorité du joueur est alors transférée entre ces deux prefabs lors du changement de scène. Mais une fois ce système de lobby mis en place, un autre défi est apparu : réussir à synchroniser manuellement les éléments côté client et serveur, ce qui fut vraiment difficile.

À l'origine, nous avions prévu d'implémenter un système permettant aux joueurs d'envoyer des aléas à leurs adversaires. Cependant, par manque de temps, nous avons dû renoncer à cette fonctionnalité. Nous nous sommes donc concentrés sur l'implémentation des deux conditions de fin de partie. La première survient lorsqu'un joueur atteint le "pic de l'aube" en premier : dans ce cas, la partie se termine immédiatement pour tous les joueurs, et chacun se voit afficher un écran de fin personnalisé, selon qu'il ait gagné ou non. La deuxième condition se déclenche lorsqu'un joueur perd la partie en cours de route. Initialement, nous avions prévu de créer un mode « spectateur », permettant au joueur éliminé de changer de caméra pour

observer la partie des autres. Cependant, nous nous sommes rendu compte que cela aurait nécessité la synchronisation d'un grand nombre de variables avec le serveur, complexifiant fortement le développement.

Nous avons donc finalement décidé que, si un joueur perd, la partie s'arrête pour tous les joueurs, et que celui ayant le plus grand nombre de jours est déclaré vainqueur.

L'écran de fin affiche ensuite le nombre de jours du joueur par rapport au gagnant. Le principal défi pour ces deux conditions a été de synchroniser plusieurs variables, telles que « est-ce que le joueur a gagné ? », « nombre de jours », ainsi que d'autres booléens. Certaines informations importantes de la partie étaient stockées par défaut dans une classe statique, ce qui rendait leur utilisation impossible dans un contexte multijoueur. Cela a rendu la tâche bien plus difficile car nous avons donc dû adapter le fonctionnement de certaines fonctions selon que le joueur soit en multijoueur ou non. À cela s'est ajouté le défi d'assimiler les différentes méthodes propres à Mirror pour gérer les interactions serveur-client, telles que Command, Server et TargetRpc, afin de comprendre quand les utiliser et sur quel script les placer, client ou serveur. De plus, un problème qui nous a également pris beaucoup de temps à résoudre concernait la duplication du GameObject serveur. Lorsqu'on rechargeait la scène où ce GameObject était initialisé par défaut alors qu'un serveur était déjà actif, deux instances se retrouvaient actives simultanément, ce qui causait des erreurs. Nous avons donc décidé de déplacer ce GameObject dans la scène de démarrage (bootstrap), ce qui empêche sa duplication, car il est impossible de revenir à cette scène sans relancer complètement le jeu.

Enfin il fut très difficile de comprendre nos erreurs lorsqu'il y en avait car, par exemple, lorsque l'on souhaitait débugger en étant l'hôte, il y avait un timeout qui déconnectait le client, rendant donc le débogage moins pertinent. Il fallait refaire des builds à chaque modification dans le code côté serveur, certaines fois le code marchait quand nous prenions l'hôte mais pas quand nous étions client... Le multijoueur a vraiment constitué une étape ardue de la conception de notre jeu, mais a également enrichi nos connaissances en nous permettant d'apprendre les bases et comment fonctionne une structure en réseau.

Enfin, il a été très difficile de comprendre nos erreurs lorsqu'elles survenaient. Par exemple, lors du débogage en tant qu'hôte, un timeout finissait souvent par déconnecter le client, ce qui rendait le débogage moins pertinent. Il fallait de plus refaire des builds à chaque modification côté serveur, et parfois le code fonctionnait correctement en tant qu'hôte, mais pas en tant que client. Le multijoueur a vraiment représenté une étape difficile dans la conception de notre jeu, mais il a aussi grandement enrichi nos connaissances en nous permettant d'apprendre les bases et le fonctionnement d'une architecture réseau.

### 3.8 Tutoriel

Afin de guider le joueur à comprendre toute la richesse de Journey Towards Dawn, nous avons implémenté un système de tutoriel linéaire, accessible dès le menu, avant la sélection de la difficulté.



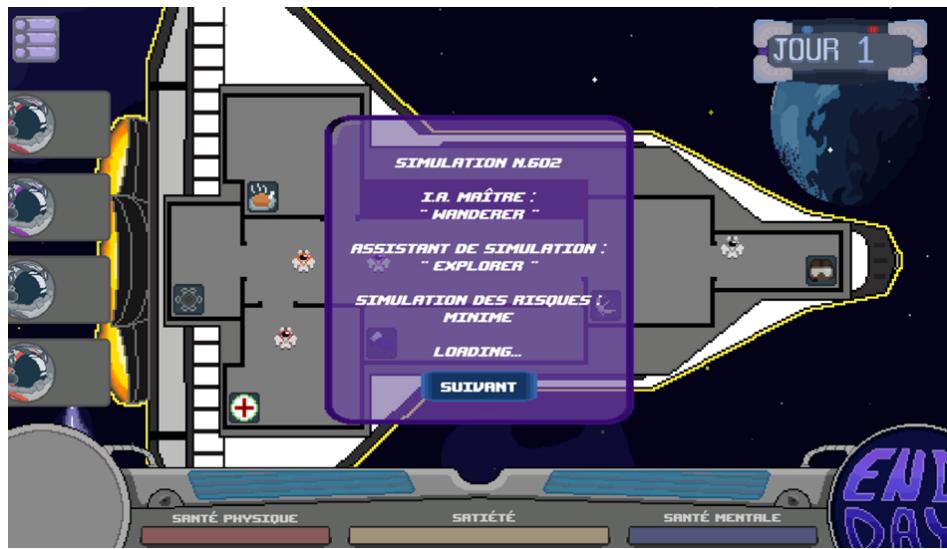
Si le joueur accepte, il est envoyé vers la scène de sélection des personnages. Sinon, la fenêtre de choix de la difficulté s'affiche. Après avoir choisi une difficulté, il est également redirigé vers la scène de sélection des personnages. Au début de cette scène, une cinématique d'introduction présente l'histoire de

Journey Towards Dawn. Si le tutoriel a été choisi, la partie qui suit n'est qu'une simulation,

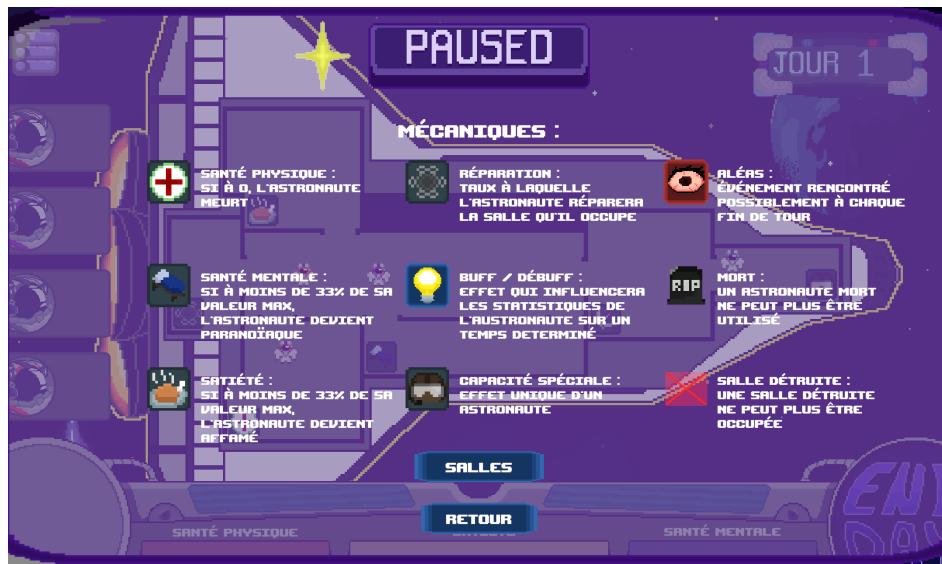


et non l'expédition réelle vers le Pic de l'Aube.

Le tutoriel se présente sous la forme d'un enchaînement de fenêtres, expliquant en détail les mécaniques du jeu, enrichies par l'histoire de Journey Towards Dawn.



Si le joueur ne se souvient plus de certaines fonctionnalités du jeu, il peut accéder à des informations utiles via le menu pause, en appuyant sur le bouton "Information". Cette page contient la généralité des mécaniques de Journey Towards Dawn, ainsi que les fonctionnalités des salles du vaisseau. Tout n'est pas spécifié, afin de garder un aspect mystérieux et de découverte pour le joueur



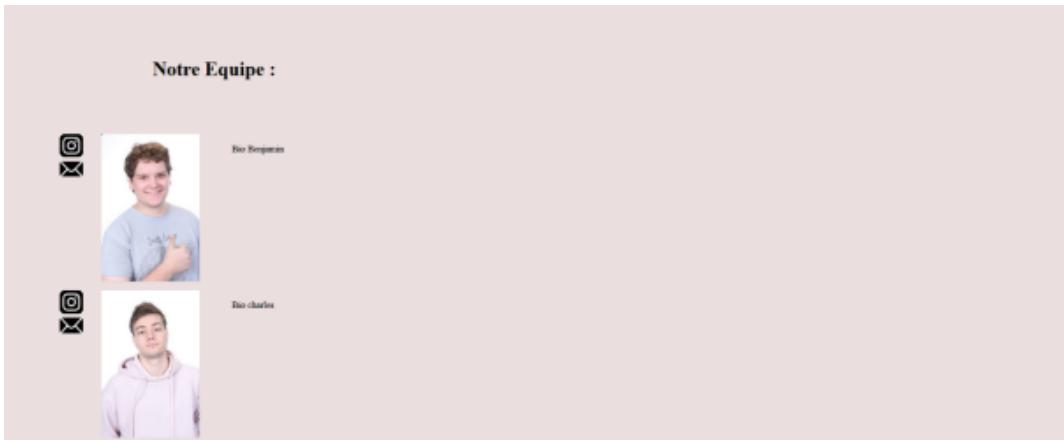
### 3.9 Site Web

Afin de réaliser le site web, il nous a fallu dans un premier temps regarder des tutoriels sur Youtube car nous ne savions pas coder avec les différents langages de développement web. Ainsi, nous nous sommes documentés en consultant des sites qui nous ont permis d'expliquer le fonctionnement des balises en HTML et d'expliquer leurs différentes variantes. Aussi, à l'aide de vidéos telles que celles du vidéaste Graven, nous avons pu créer la page d'accueil suivante :



Cette page d'accueil permet d'accueillir nos potentiels joueurs et leur donner une première

impression sur notre jeu. Elle permet également de rediriger les utilisateurs vers nos différentes ressources. Par exemple, un clique sur le bouton “L'équipe” redirigera vers les informations des différents membres du groupe avec la photo et les réseaux sociaux de chacun suivi d'une courte description.



De plus, la page d'accueil permet également de rediriger les utilisateurs vers une autre page qui contiendra un manuel d'installation du jeu, ainsi que les fichiers nécessaires à son installation. Enfin, la page d'accueil présentera un bouton "Ressources" qui redirigera les utilisateurs vers une page qui listera l'ensemble des documents relatifs au projet tels que les outils utilisés, les rapports et plans de soutenance et le cahier des charges. Nous n'avions cependant pas encore réalisé ces deux dernières pages.

Depuis la première soutenance, nous avions amélioré son design en remplaçant l'ancien fond beige qui n'était pas en accord avec le thème de notre jeu par un fond bleu foncé qui tend vers le bleu plus clair (dégradé) rappelant ainsi le thème de l'espace. De plus, nous avions complètement remplacé l'ancienne page de présentation des membres en rajoutant notamment le synopsis de notre jeu et les biographies des membres du groupe.

Dans ce monde fictif, l'humanité stagne. Les ressources s'affaiblissent et la technologie ne fait plus de progrès. Mais un jour, des scientifiques découvrent la présence d'un artefact au fin fond de l'espace. Cet artefact est un mystère et beaucoup théorisent sur son origine. Énormément de chercheurs pensent que cet objet pourrait être la clé pour sauver l'humanité et le surnomment "Dawn Peak" ou "Pic de L'Aube". C'est pourquoi une expédition spatiale est mise en place pour atteindre cet artefact afin de l'étudier. Le joueur se verra incarner une IA, du nom de "Wanderer", installée dans le vaisseau qui amènera une équipe de quatre scientifiques vers l'objet mystérieux.

Passionné de jeux vidéo, j'ai découvert la programmation en première et choisi cette voie après une brève incursion en sciences. À EPITA, j'ai concrétisé mon projet de jeu, dont je gère une grande partie du développement et la conception.

Passionné de jeux vidéo et ancien compétiteur e-sport, j'ai découvert la programmation par curiosité avant d'intégrer EPITA. En tant que chef de groupe et Game Designer, je mets mon leadership et mon imagination au service de notre projet innovant.

Passionné de programmation depuis mon enfance, j'ai appris Python et me spécialise dans le développement web. Je contribue activement à notre projet de jeu vidéo en mettant à profit mes compétences techniques et mon esprit d'équipe pour assurer son succès.

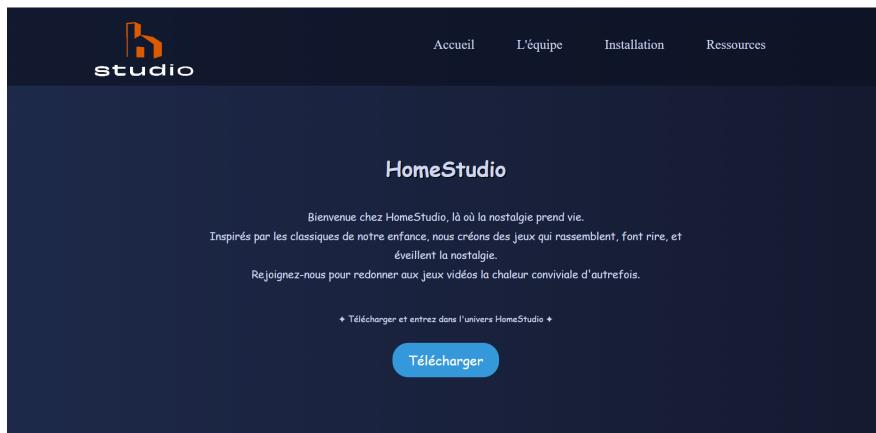
Passionné de programmation et de dessin, j'ai approfondi mes compétences en NSI avant de rejoindre EPITA. Dans ce projet, je gère la partie artistique, alliant code et design, tout en développant mes compétences en travail d'équipe et en organisation.

Sur la page Installation, nous avions également changé le bouton d'installation et implanté un carrousel faisant défiler différentes images de notre jeu, donnant ainsi un premier aperçu de notre jeu tout en leur donnant envie de l'installer.

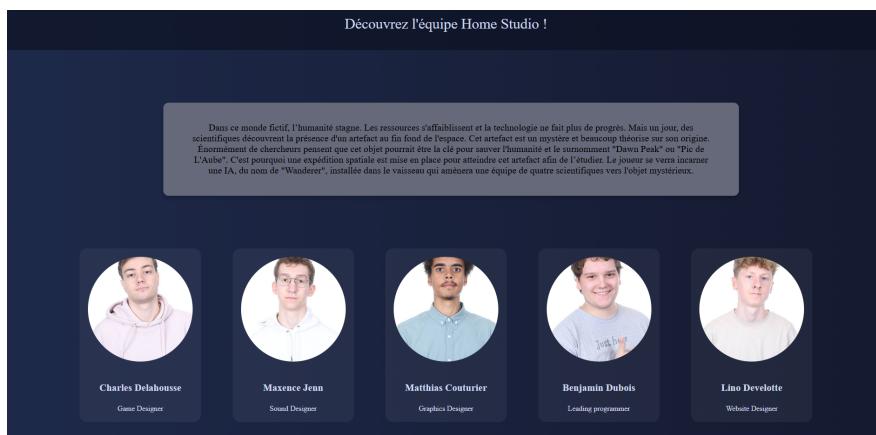


Durant la dernière phase de travail, pour la page d'accueil, nous sommes repartis de zéro afin de proposer un design à la fois sobre, élégant et accueillant. L'objectif était de mieux représenter notre entreprise tout en gardant une touche moderne et interactive. La barre de navigation a été revue avec une couleur plus foncée, apportant plus de contraste et de lisibilité. Nous avons également remplacé l'ancien fond sur le thème de l'espace, plus lié à l'univers du jeu qu'à celui de l'entreprise, par un dégradé allant du bleu clair au bleu foncé, plus cohérent avec l'identité visuelle globale. Le nom de l'entreprise s'affiche en grand au centre de la page, accompagné d'une phrase qui présente le style de nos jeux, ainsi qu'un slogan animé qui attire l'œil. Juste en dessous, un bouton « Télécharger » permet aux visiteurs de facilement télécharger notre jeu.

Pour la page "Équipe", nous avons également repensé le design en optant pour un fond en dégradé de bleu clair à bleu foncé, dans la continuité visuelle de la page d'accueil. Les biographies des membres ont été retravaillées pour les rendre plus modernes et dynamiques. Désormais, en passant la souris sur la photo de chaque personne, une version plus détaillée de sa bio s'affiche, avec notamment une présentation plus complète de son rôle dans le développement du jeu. C'est une manière simple et interactive de mettre en valeur le travail



de chacun, tout en rendant la page plus vivante et agréable à explorer.



Troisièmement, la page "Installation" a elle aussi été mise à jour pour s'aligner visuellement avec le reste du site. Le fond a été remplacé par le même dégradé de bleu clair à bleu foncé, créant une cohérence graphique entre les différentes pages.

Nous avons également enrichi cette section avec plusieurs images accompagnées de textes courts qui mettent en valeur notre jeu vidéo. Ces éléments visent à susciter l'intérêt des visiteurs et à leur donner un aperçu de l'univers du jeu.

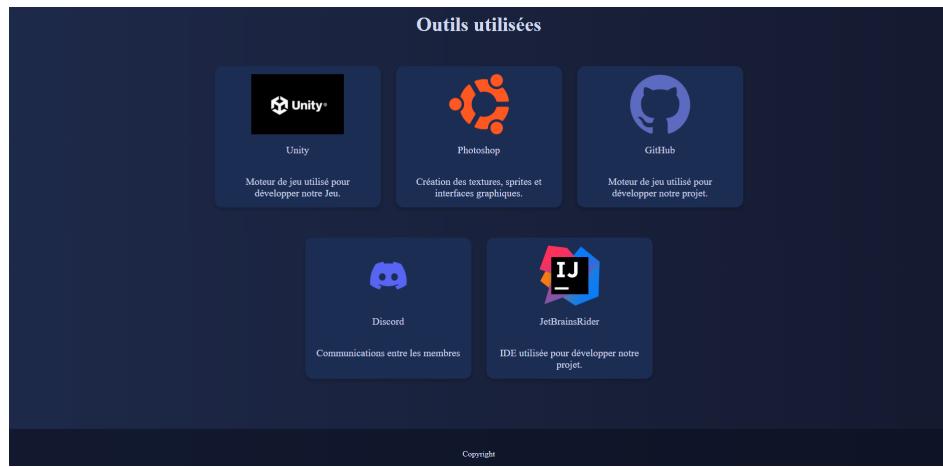
Enfin, juste au-dessus du bouton « Télécharger », nous avons ajouté un message encourageant les visiteurs télécharger le jeu.

Dernièrement, la page "Ressources" a été modernisée tout en conservant l'esprit du design général du site. Le visuel a été légèrement retravaillé pour gagner en clarté, tout en gardant une présentation simple et cohérente avec les autres pages. Nous avons également amélioré



l'organisation des documents en ajoutant des titres clairs, afin de faciliter la navigation et permettre aux visiteurs de trouver rapidement ce qu'ils cherchent.

Lors de la réalisation du site web, nous avons rencontrés de nombreux problèmes liés à notre absence de connaissances et d'expérience que nous avons su résoudre. Par exemple, lorsque nous devions utiliser du CSS pour styliser le site, nous avions au début du mal à



cibler un objet du fichier HTML, ce qui nous a fait perdre beaucoup de temps notamment car les tutoriels n'expliquaient pas vraiment le principe derrière cette fonctionnalité. Nous avons ainsi dû apprendre par le biais d'échecs et de réussite. Par ailleurs, bien que nous ayons pu comprendre les bases du HTML et du CSS, il reste difficile de bien agencer les éléments du site afin qu'ils soient esthétiques. Enfin, ne connaissant pas JavaScript pour le moment, cela réduit grandement les choix d'ajouts potentielles de fonctionnalités poussées.

## 4 Conclusion

Ce projet nous a permis d'acquérir de solides connaissances sur le fonctionnement d'Unity, ses différentes fonctionnalités ainsi que ses nombreux plugins. Avant de commencer, nous n'avions aucune expérience avec le développement multijoueur, ce qui a rendu cet aspect particulièrement formateur. Au-delà des compétences techniques, ce travail a renforcé notre capacité à collaborer efficacement en équipe, à mieux communiquer et à organiser la répartition des tâches. Enfin, cette expérience nous a aussi montré l'importance de la persévérance face aux défis techniques.