

Rapport de Projet

Psyhunt



Game Design Document réalisé dans le cadre du cours 8INF960 : “Principes de conception et de développement de jeux vidéo” pour le jeu “Psy Hunt”

Collaborateurs :

SHAN YAN Alexis - #SHAA31039905

BLOMME Thomas - #BLOT30049904

LACLAVERIE Pierre - #LACP03119904

PORTIER Barnabé - #PORB26070006

OUEDRAOGO Issouf - # OUEI04079506

Rappel du concept du jeu ainsi que ses mécaniques	3
Concept	3
Histoire	3
Présentation des interfaces	4
Menu principal	4
Menu pause	5
Menu Game Over	5
Menu niveau terminé	6
Choix du niveau	6
Contrôles	7
Organisation du code	7
Player :	8
Patroller :	9
GameManager :	9
Répartition des tâches	10
Division du projet	10
Répartition des tâches	11
Justification des choix de conception et réduction du scope initial	11
Choix de conception	11
Choix de la plateforme	11
Choix du gameplay	12
Choix des assets	12
Choix des technologies	12
Outil de gestion des fichiers	12
Outil de gestion de projet	13
Outil de communication	13
Réduction du scope initial du projet	13
Assurance qualité	13
Images du jeu et exemples de code	14
Problèmes rencontrés	16
Conclusion et les enseignements que nous avons tiré du projet	17
Assets utilisés	17

1. Rappel du concept du jeu ainsi que ses mécaniques

a) Concept

“Psyhunt” est un jeu d’infiltation qui nous met dans la peau d’un jeune adulte dans une cité cyberpunk dystopique. C’est donc un jeu à destination des 16 ans ou plus, jouable sur PC. Ce jeu s’appuie sur une utilisation intelligente des pouvoirs dont dispose le joueur afin de s’échapper de différents niveaux en essayant de ne pas se faire repérer par la milice de la ville.

Le jeu se joue en vue à la première personne, contrairement à plusieurs jeux du genre, qui eux, se déroulent à la 3e personne (Metal Gear Solid pour ne citer que lui). Nous avons fait ce choix car cela force le joueur à utiliser les pouvoirs du personnage principal, destinés principalement à la reconnaissance. Le joueur devra atteindre la fin de niveaux afin d’avancer dans l’histoire pour finalement, à la fin, confronter l’antagoniste du jeu. Seulement un niveau a été implémenté, ainsi que 3 pouvoirs sur 5. Si nous pouvions continuer le développement de notre jeu, nous développerions les 2 autres pouvoirs ainsi que d’autres niveaux afin de montrer tout le potentiel de notre histoire.

b) Histoire

Kadingirra la fastueuse, joyaux au milieu du grand désert de sel, est la dernière ville humaine connue depuis la Guerre, resplendissante de richesse et de puissance. Porté par la toute puissance technologique de Spark Industries (ou “SI”), le parti technocrate exerce un contrôle politique et militaire total sur la ville. La milice privée de “SI”, équipée des implants les plus avancés et des armes les plus mortelles, assure l’ordre et la sécurité au sein de la cité. Leur monopole total des technologies de grade militaire suffit à démotiver la population de tout désir de changement. Seuls les déphasés, rares individus dotés de pouvoir psychiques, pourraient menacer cette domination. Pour se prémunir de tout risque de rébellion et garder sous haute surveillance cette partie de la population, les technocrates mènent depuis des années une importante campagne de propagande. De cette dernière résulte un climat profondément hostile aux déphasés, qui se retrouvent alors totalement isolés, groupés dans les quartiers les plus insalubres de Kadingirra.

Nous sommes en 2198 à Kadingirra. Le personnage principal, un adolescent tout à fait ordinaire, découvre un jour qu’il a des pouvoirs psychiques. Alerté par une voix dans sa tête, il échappe in extremis à un raid de la milice appelée par ses parents ayant pour but de l’emprisonner.

Sonné et choqué, il se fait guider par la voix dans sa tête pour échapper à la milice. Il s'avère que la voix provient d'un autre détenteur de pouvoir psychique qui va progressivement faire comprendre au héros principal que tout est la faute de la maire de Kadingirra qui, pour se faire réélire, exacerbe les tensions entre la population et les déphasés, les accusant de tous les maux. En effet, c'est elle la figure de proue du mouvement anti-déphasé.

2. Présentation des interfaces

a. Menu principal



Nouvelle partie: permet de démarrer le jeu depuis le premier niveau (par défaut le level 1) et de pouvoir faire des sauvegardes par la suite.

Continuer partie: permet au joueur de continuer sa partie depuis le dernier point de sauvegarde.

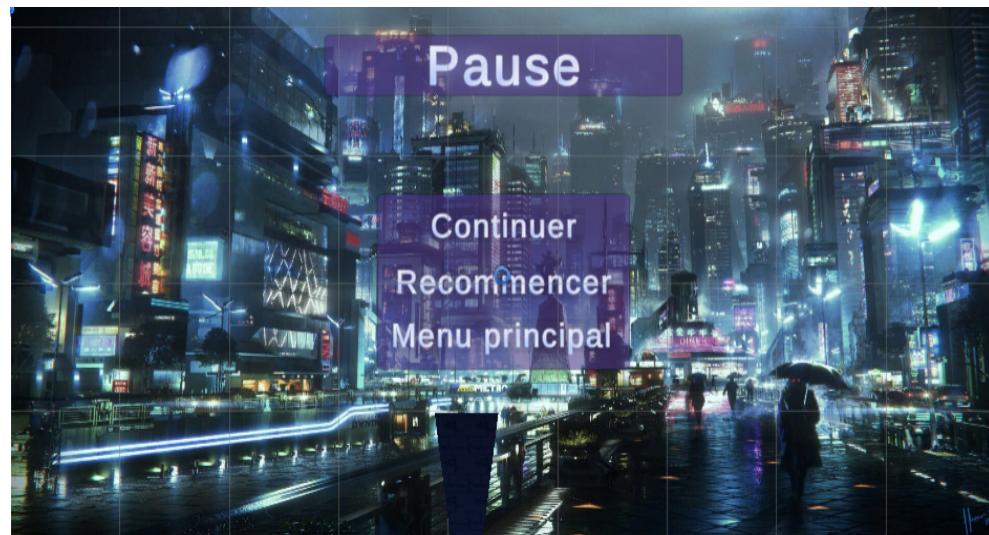
Niveaux: permet de visualiser les niveaux qui sont disponibles

Contrôles: liste les caractères du clavier qui permet de manipuler le joueur

Crédits: Affiche la liste des membres de l'équipe

Quitter: permet de quitter le jeu et mettre fin à l'exécution du programme

b. Menu pause



Continuer: Permet de reprendre une partie après l'avoir mis sur pause de manière volontaire (appuyer sur la touche Echap). Le joueur peut aussi continuer en appuyant sur la touche Echap

Recommencer : permet au joueur de reprendre une partie depuis le début du niveau. Il a pour but de reprendre le jeu dans de bonnes conditions, remet tout à zéro.

Menu principal : permet de quitter une partie et de repartir sur l'écran du menu principal

c. Menu Game Over



Recommencer le niveau : permet au joueur de rejouer le niveau après avoir perdu.

Retourner au menu principal : permet au joueur de quitter l'écran de game over et d'atteindre le menu principal.

d. Menu niveau terminé



L'écran s'affiche pour faire comprendre à l'utilisateur que le niveau est terminé avec succès. Il peut retourner au menu principal ou choisir le niveau suivant (dans une future version).

e. Choix du niveau



Permet de présenter au joueur les niveaux disponibles. Il peut sélectionner uniquement les niveaux débloqués. A chaque fois qu'un niveau est terminé le

suivant est débloqué automatiquement. Les numéros sur les niveaux permettent d'avoir une idée sur l'ordre de déblocage des niveaux. Le niveau 1 est toujours activé et permet de débloquer le niveau 2 (suivant) et le niveau 2 permet de débloquer le niveau 3 et ainsi de suite. A cause de problèmes de temps, seulement le niveau 1 est disponible pour le moment.

f. Contrôles



Donne une idée des touches à utiliser dans le jeu.

Déplacements: effectués avec les touches W-A-S-D (W= aller en avant, A = aller à gauche, S = reculer , D = aller à droite)

Attaque: permet d'attaquer un ennemi (Patrouilleur, Sentinelle) en faisant un clic gauche de la souris

Utiliser le pouvoir: permet de mettre en exécution un pouvoir sélectionner (par exemple prendre le contrôle d'un rat ou de l'aigle)

Pour des pouvoirs : permet de voir les pouvoirs disponibles en fonction du niveau actuel. La touche tabulation permet de voir les pouvoirs et faire un choix avec un clic gauche.

accroupissement: permet au joueur de s'accroupir en cliquant sur la touche Ctrl. Il lui permet de se cacher à certains types d'ennemis.

Courir : permet de faire courir le joueur en appuyant sur la touche Shift. La touche shift doit être enfoncée pour maintenir la course.

Sauter: permet au joueur de décoller du sol pour une courte durée en appuyant sur la touche espace.

3. Organisation du code

Nous nous sommes principalement inspirés du modèle de conception ECS (entité-Composant-Système) dans notre projet. Nous n'avons donc pas de diagramme de classe à proposer, mais plusieurs entités sur lesquelles nous avons attaché des composants

appelés par le système afin d'appliquer des actions/modifications sur ces entités. Toutes nos entités travaillent ainsi de concert afin de donner le jeu Psyhunt.

Quelques exemples d'entités comportant plusieurs composants :

Player :



The screenshot shows the Unity Inspector window for a 'Player' Prefab. The top bar includes a blue cube icon, a checked 'Player' checkbox, a 'Static' dropdown, a 'Tag' dropdown set to 'Player', a 'Layer' dropdown set to 'Player', and tabs for 'Prefab', 'Open', 'Select', and 'Overrides'. Below this, a list of components is displayed:

- Transform
- Character Controller (checked)
- Animator (checked)
- Box Collider (checked)
- Player (Script) (checked)
- Movement Controller (Script) (checked)
- Field Of View Player (Script) (checked)
- Animal Control Power (Script) (checked)
- Attack (Script) (checked)
- Invisibility (Script) (checked)

Patroller :



The screenshot shows the Unity Inspector window for a Prefab named "Patroller".

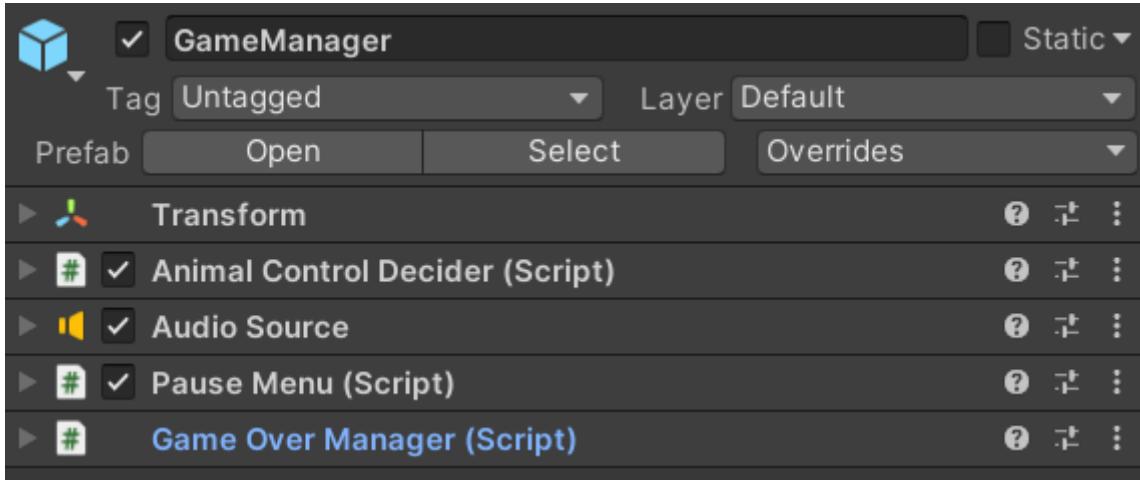
General:

- Checkmark next to "Patroller"
- Static
- Tag: Untagged
- Layer: ennemy
- Prefab button
- Open button
- Select button
- Overrides dropdown

Components:

- Transform
- Animator
- Box Collider
- Patroller Behaviour (Script)
- Nav Mesh Agent
- Field Of View Enemy (Script)

GameManager :



Comme nous avons pu le voir, nous avons plusieurs entités possédant plusieurs script chargés de contrôler son comportement. Il existe deux types d'entités : les entités physiques (Player, Patroller par exemple) et les entités logiques (GameManager).

Les premières sont chargées de conditionner le comportement des objets en jeu et ainsi leur donner plusieurs fonctionnalités, comme pouvoir marcher, tirer, utiliser un pouvoir, etc... Les deuxièmes sont principalement utilisées pour contrôler des options plus globales du jeu, comme la musique, les caméras à activer ou désactiver, etc...

Nous avons regroupé toutes ces entités sous la forme de prefabs que nous avons enregistrés dans le projet afin de pouvoir les réutiliser le plus efficacement possible.

4. Répartition des tâches

a) Division du projet

Pour la répartition des tâches, nous avions au départ un échéancier comportant les différentes tâches assignées à chaque sprint (voir le GDD pour plus d'informations). Cependant, au fur et à mesure de l'avancée du projet, nous nous sommes rendus compte de l'importance de certaines fonctionnalités que l'on avait référencé dans un sprint ultérieur. Nous avons donc remanié notre Backlog Project afin de mieux organiser le projet. Ce Backlog Project a ensuite été divisé en Backlog Sprints, qui contenaient chacun les tâches que chaque membre du groupe devait réaliser (il est à noter que nos sprints duraient une semaine afin d'éviter un "rush" final lors des dernières semaines de conception de notre jeu). La version finale de nos Backlog Sprints est la suivante :

Sprint 1	Sprint 2	Sprint 3	Sprint 4
Personnage Principal	mort du personnage principal	IA Ennemis	Sauvegarde
Patrouilleur	Création banque de son	Sentinelle	Niveau 1
Menu Principal	Création HUD	Invisibilité	Bandes sons
Déplacements	Contrôle animal	Sauvegarde	Game Over
Detection du joueur par l'ennemis	Menu Game Over	Menu pause	Roue des pouvoirs
	Spyder sens	Effets visuels	Cinématique
		Attaque mélée	
		Ebauche premier niveau	

b) Répartition des tâches

Au début de chaque sprint, nous nous rencontrions en personne pour discuter de l'avancement du projet et de l'assignement des prochaines tâches. De cela s'est dégagé quelques tendances de préférence au codage de certaines fonctionnalités par plusieurs membres du groupe. Nous regardions alors le contenu du Backlog Project et essayions d'être cohérent avec le GDD en choisissant les fonctionnalités à implémenter. C'est notamment lors de cette phase que nous nous sommes plusieurs fois aperçus de l'importance de certaines fonctionnalités réservées jusqu'alors à un sprint ultérieur (voir plus haut). Les principales fonctionnalités implémentées par chaque membre du groupe sont présentées dans le tableau ci-dessous :

Alexis	Les ennemis / Control du rat / Invisibilité / Détections des ennemis
Pierre	Recherche des assets / Control de l'aigle / Mock test / Level1 / Montage vidéo
Issouf	Interfaces / Animation de mort / Pistes audio / Musiques / Effets sonores
Barnabé	Spider sense / Attaque / Level1 / Debug
Thomas	Personnage / HUD / Cinématique / Création de clips

5. Justification des choix de conception et réduction du scope initial

a) Choix de conception

I. Choix de la plateforme

Nous avons choisi d'implémenter notre jeu pour PC, car nous sommes convaincus qu'aujourd'hui, c'est la façon la plus simple de distribuer un petit jeu indépendant comme le nôtre, via notamment des plateformes d'achats de jeux en ligne tel que Steam. L'idée d'un jeu mobile a été rapidement écartée, car bien évidemment, un jeu d'infiltration ne se prête

pas beaucoup à la maniabilité sur smartphone, ce qui nous a confortés dans l'idée de développer notre jeu uniquement pour des joueurs PC.

II. Choix du gameplay

Psyhunt est un jeu d'infiltration avec une vue à la première personne. Cette décision, allant à l'encontre de la "méta" de beaucoup de jeux d'infiltration (très souvent en troisième personne) a un double objectif : celui d'apprécier au mieux les pouvoirs, et ainsi mieux les contrôler, et aussi forcer le joueur à utiliser ses pouvoirs pour réunir des informations sur son environnement et ses ennemis.

En effet, la principale raison de l'utilisation de la troisième personne dans les jeux d'infiltration est de fournir beaucoup d'informations au joueur, afin que ce dernier puisse planifier ses prochaines actions. Ici, pour faire cette récolte d'informations, le joueur aura accès à plusieurs pouvoirs dédiés : marquer les ennemis, prendre le contrôle d'un petit animal pour faire de la reconnaissance, etc...

III. Choix des assets

Nous avons choisi des assets ayant un design se rapprochant du style Steampunk afin de coller au mieux à notre univers. Nous avons utilisé des assets de toutes sortes : bâtiments, personnages, animaux, etc... Nous essayons le plus possible de prendre des assets gratuits, mais quelques-uns ont tout de même dû être achetés, soit par le département, soit par nous.

b) Choix des technologies

I. Outil de gestion des fichiers

Comme outil de gestion de nos fichiers, nous avons fait le choix d'utiliser Perforce, un outil déjà pris en main plus tôt dans l'année lors du mini-projet 2D. Pour un petit projet comme le nôtre, le choix de Perforce peut sembler quelque peu étrange, mais nous voulions surtout l'utiliser afin de gagner en compétences pour notre futur métier. Nous pouvons imaginer que si nous continuons le développement de Psyhunt, l'intérêt de Perforce augmentera au fur et à mesure de l'ajout de nouvelles fonctionnalités et niveaux. Il est à noter cependant, que nous avons tout de même rencontré quelques problèmes de synchronisation de fichiers avec Perforce et que de ce fait, pour un petit projet comme le nôtre, ce n'était pas l'outil de plus optimal à utiliser. Peut-être qu'un Unity Teams ou un simple git aurait très bien fonctionné.

II. Outil de gestion de projet

Comme outil de gestion de projet, nous avons choisi d'utiliser Trello, un des outils les plus populaires dans le cadre de projets informatiques. Il nous permettait de nous répartir les tâches et d'assigner, pour chaque tâche, des conditions de complétion, rendant le travail que nous devions réaliser clair et précis. La répartition des tâches via le trello nous permettait de voir qui implémentait quoi, et ainsi, nous permettait de suivre l'avancement du projet.

III. Outil de communication

En tant qu'outil de communication, nous utilisions discord. Nous avions un groupe discord sur lequel nous pouvions nous organiser. Plusieurs channels étaient présentes, chacune avec un rôle particulier (sur le channel "*Session-planning*", nous nous mettions d'accord tous ensemble pour savoir quand est-ce que l'on se verrait, sur le channel "*Devoirs*", nous discutions de l'avancée du projet en général, etc...). Nous nous retrouvions aussi en présentiel chaque lundi afin de discuter de l'avancée du projet et s'assigner les tâches sur le Trello.

c) Réduction du scope initial du projet

Le projet de départ (voir le GDD) était bien trop ambitieux par rapport à ce que nous pouvions réaliser en un mois (débutants avec Unity, quatre autres cours à côté pour la plupart, etc...). Il a donc fallu réduire le scope du projet afin d'avoir un jeu viable pour la date de rendu du projet. Nous avons donc enlevé deux pouvoirs : le choc psychique et la création d'illusions, qui étaient des pouvoirs optionnels. Par conséquent, nous avons aussi dû enlever deux types d'ennemis : les psychiques et les androïdes, qui devaient au départ contrer le pouvoir de choc psychique. Il était aussi prévu de réaliser plusieurs niveaux avec un niveau final comportant un boss à la fin, mais au vu de la grande difficulté du level design d'un jeu d'infiltration que nous avions sous-estimé, il a finalement été décidé qu'un seul niveau serait implémenté pour la fin du projet. Cependant, il est évident que la version finale de notre jeu comportera plusieurs niveaux entrecoupés de cinématiques afin d'offrir la meilleure expérience possible au joueur.

6. Assurance qualité

Au cours du projet, tout le monde était tenu de tester le code qu'il venait d'implémenter afin d'assurer une certaine stabilité lorsqu'était venu le moment de fusionner cette fonctionnalité avec le projet existant. Cependant, à cause d'une erreur d'inattention ou de l'outil Perforce, il arrivait que des bugs soient découverts bien plus tard après leur implémentation, auquel cas il était nécessaire de se replonger dans le codage de la fonctionnalité en question afin de la débugger.

Il est cependant vrai que, même si plusieurs tests unitaires manuels ont été effectués sur les fonctionnalités indépendamment de chacune, peu de tests d'intégration ont été réalisés. En effet, nous n'avions pas eu d'organisation particulière concernant la détection des bugs, ce qui nous a fortement porté préjudice plus le projet avançait, car personne n'avait le rôle de "debugger" pour vérifier sur la scène de test / le premier niveau implémenté que tout fonctionnait bien. En dehors des tests unitaires, nous n'avons pas non plus fait de tests "limites" pour tester comment notre jeu réagissait à certaines conditions limites qui peuvent éventuellement arriver lorsque quelqu'un joue à notre jeu.

Il est à noter que lorsqu'un bug majeur était découvert, les efforts d'au moins 2 membres du groupe étaient dirigés vers la résolution de ce dernier. Lorsqu'un bug mineur était découvert (le joueur peut sauter par-dessus certains murs normalement conçus comme inatteignables par exemple), on le notait sur notre trello dans la section Améliorations / Bugs afin de le corriger plus tard.

7. Images du jeu et exemples de code

I. Images du jeu



niveau 1 du jeu

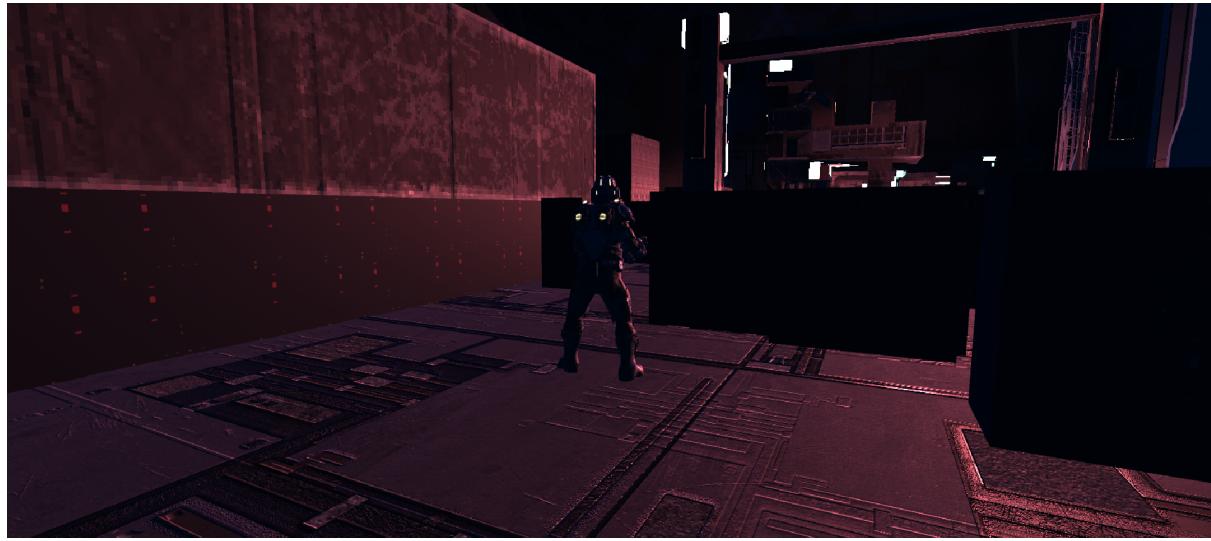


image in-game avec une sentinelle



image in-game du joueur sélectionnant un pouvoir

II. Exemple de code

```
if (Input.GetKey(KeyCode.UpArrow) || Input.GetKey(KeyCode.W))
{
    isWalking = true;
    animator.SetFloat("speed", 1);
} else
{
    if (Input.GetKey(KeyCode.DownArrow) || Input.GetKey(KeyCode.S))
    {
        isWalking = true;
        animator.SetFloat("speed", -1);
    } else
    {
        isWalking = false;
    }
}

if (Input.GetKey(KeyCode.RightArrow) || Input.GetKey(KeyCode.D))
{
    right = true;
} else
{
    right = false;
    if (Input.GetKey(KeyCode.LeftArrow) || Input.GetKey(KeyCode.A))
    {
        left = true;
    } else
    {
        left = false;
    }
}
```

Cet exemple de code montre comment nous avons géré les animations du personnage lors de ses mouvements avant, arrière, droite et gauche.

Le joueur pourra jouer soit avec les touches W-S-D-A ou avec les flèches directionnelles nous avons donc pris en compte les deux cas. Nous gérons donc les animations à l'aide de variables booléennes qui seront mises à jour à chaque input du joueur. Nous articulons ensuite toutes ses variables grâce à l'Animator de Unity.

8. Problèmes rencontrés

Les principaux problèmes rencontrés pendant le projet viennent avant tout de notre utilisation de l'outil perforce. On peut notamment noter un problème au niveau des check-out oubliés par les membres de l'équipe. On se retrouvait donc avec des versions de fichiers différentes. De plus, les layers que nous avions créé sur unity ne pouvaient pas être submit sur le dépôt ce qui nous a ralenti sur la conception du projet. Pour finir sur perforce, la dernière semaine a mis la conception du jeu à l'arrêt à cause des problèmes de serveur de l'école.

Le projet était assez conséquent au vu du temps disponible, nous avons dû réduire le scope du projet en enlevant notamment des ennemis, des pouvoirs et des niveaux.

On a aussi rencontré quelques problèmes de communication au début du projet que l'on a pu régler par la suite.

9. Conclusion et les enseignements que nous avons tiré du projet

Nous avons pu continuer notre formation sur Unity grâce à ce projet de jeu vidéo. nous nous sommes rendus compte de la facilité d'utilisation de Unity, et en même temps, du vaste monde du développement du jeu vidéo. En effet, la panoplie immense de fonctionnalités que nous présente Unity est en même temps un avantage qu'un inconvénient, car quelques fois, un problème peut surgir, sans que nous ne sachions d'où cela pourrait provenir.

Aussi, pour certains, nous avons pu nous former à d'autres outils, tels que Perforce et Trello, ce qui nous sera très pratique dans notre vie professionnelle. Nous pouvons clairement dire que nous avons appris énormément de choses avec ce projet.

En revanche, nous avons aussi pu voir quelques problèmes pouvant arriver dans le cadre de la gestion de projet, comme des problèmes de communications, ou encore des synchronisations de fichiers parfois dures à résoudre. Cependant, cela fait aussi partie de notre formation, et nous avons appris de nos erreurs, ce qui nous empêchera de les répéter par la suite.

Finalement, nous avons aussi pu constater ce qu'apporte un projet trop ambitieux, dont le scope initial est beaucoup trop important par rapport au temps qui nous était alloué.

10. Assets utilisés

Ce projet n'aurait pas été possible sans plusieurs assets que nous avons trouvés sur internet. Voici la liste des assets utilisés dans le cadre du développement du jeu vidéo Psyhunt :

- [Patrouilleur, sentinelle]
<https://assetstore.unity.com/packages/3d/characters/humanoids/sci-fi/sci-fi-space-soldier-polygonr-66384>

- [Personnage Principal]
<https://assetstore.unity.com/packages/3d/characters/human-characters-free-sample-pack-181554>
- [Aigle]
<https://assetstore.unity.com/packages/3d/characters/animal-monster-eagle-202286>
- [Rat]
<https://assetstore.unity.com/packages/3d/characters/animals/mammals/rats-30264>
- [Outline Effect]
https://assetstore.unity.com/packages/vfx/shaders/fullscreen-camera-effects/outline-effect-78608?_qa=2.251252228.362176383.1639971457-460708442.1630707035
- [CyberPunk Citykit] <https://assetstore.unity.com/publishers/37980>
- [Greeble city kit]
<https://assetstore.unity.com/packages/3d/environments/sci-fi/greeble-city-kit-180500>
- [skybox]
<https://assetstore.unity.com/packages/2d/textures-materials/sky/skybox-series-free-103633>