RAPPORT DE SOUTENANCE 1

REMOVED FROM THE PAST

BY TIMESOFT DEV

EPITA SUP S2 – Promo 2026

Abel ROFFIAEN
Yusuf OZEN
Yann MESSE
Solène DINTINGER

Table des matières

1. Introduction

Deux mois après le début du projet, ce rapport de la première soutenance nous permet de décrire notre avancée, nos remarques et nos difficultés. Ce rapport prend en compte le CDC (Cahier des Charges) modifié après correction. Nous apporterons ainsi nos explications en se basant sur notre deuxième essai du CDC du projet.

Ce projet nous a permis de nous consacrer à une tâche précise et surtout un réel projet de programmation. Le travail d'équipe pour un projet de cette ampleur, et surtout de cette durée, nous est encore parfois difficile à gérer mais nous nous améliorons sans cesse.

N'étant pas encore habitués et très performants dans la programmation, il est plus compliqué de nous lancer sur la programmation en C# mais le nombre d'outils et l'aide à notre portée nous encourage à avancer.

Notre temps c'est premièrement consacré sur notre adaptation au moteur de jeu Unity. Nous avons également réussi à prendre de l'avance sur certaines parties du projet qui étaient plus facilement réalisable que ce que nous pensions.

Nous avons profité du début de semaine libre lors de la semaine de soutenance pour nous retrouver et nous mettre à jour sur ce qui a été fait et ce qu'il reste à faire pour notre projet.

2. Partie Commune

2.1. Répartition des tâches

Tâche	Abel	Solène	Yusuf	Yann
Menu principal	Suppléant	Principal		
Commandes			Suppléant	Principal
Options	Principal		Suppléant	
Graphismes	Suppléant	Principal		
Carte		Suppléant		Principal
Apparition	Suppléant		Principal	
Interface joueur		Suppléant	Principal	
Histoire principale	Suppléant			Principal
Système de déplacement		Suppléant	Principal	
Interaction PNJ		Suppléant	Principal	
Récompenses	Principal		Suppléant	
Système de combat		Suppléant	Principal	
Ennemis		Principal		Suppléant
Niveau du joueur			Suppléant	Principal
Système d'amélioration	Principal			Suppléant
Multijoueur		Principal	Suppléant	
Sauvegarde	Suppléant			Principal
Quitter	Principal			Suppléant
Site Web	Principal			Suppléant

Répartition des tâches

Pour la répartition, il n'y a pas du tout eu de changements sur la répartition des tâches. Mais en cas de difficultés, nous nous aidons tous si une autre personne s'y connait mieux que nous sur un point. Mais comme nous commençons tous Unity, cela n'arrive pas souvent.

Cependant, pour les tâches plus créatives comme les graphismes ou la carte, tout le monde apporte du sien, rien d'en donnant son avis sur une modèle de personnage ou de texture.

Le projet avance donc à son allure prévue car aucun changement majeur de la répartition des tâches n'a été apporté.

2.2. Avancement du projet

Pour commencer notre projet, nous avons premièrement pris connaissance avec Unity. En effet, celui-ci est un nouvel outil pour nous tous. S'y habituer nous a donc pris plusieurs semaines avant de l'utiliser de manière vraiment efficace. Nous découvrons cependant toujours de nouveaux outils pour nous aider et rendre notre programmation plus simple et même plus efficace.

Une des parties les plus dures est évidemment le lancement du projet. Afin de se mettre directement dans l'ambiance du jeu, nous avons créé les premières scènes principales : le menu principal, la scène de jeu et la scène de combat.

Chacun se lançait et s'habituait à Unity à son propre rythme mais nous sommes tous capables d'utiliser Unity correctement à présent.

En plus de Unity, nous ne sommes pas non plus tout à fait familiers avec le langage C#. Bien que nous connaissons à présent les bases avec les tps de cours d'IP, en situation réelle pour un projet, les différents fichiers et fonctions deviennent alors moins claires.

En parallèle à l'apprentissage de Unity, nous avons alors également dû apprendre les fonctions et commandes de C# qui pourraient nous être utiles : par exemple l'utilisation des fonction Start() et Update() dans nos scripts.

Bien que cet apprentissage nous ait fait ralentir sur le début de la conception du projet, nous n'avons pas oublié nos objectifs. Nous travaillons tous régulièrement en alternant les tâches afin de ne pas nous lasser. La motivation vient avec le temps et nous sommes maintenant tous prêt à y travailler plus souvent que nous le faisions auparavant.

Nous ne doutons alors pas que le projet avancera de la manière voulue et feront de notre mieux pour arriver à un résultat final qui nous plairait à tous.

2.3. Groupe du projet

Notre groupe est toujours composé de ses 4 membres originels : le chef de groupe Abel ROUFFIAEN, Yann MESSE, Yusuf OZEN et Solène DINTINGER.

Aucun changement majeur n'a été constaté dans notre groupe. Les 4 membres de notre groupe travaillent, pour la majorité de leur temps, sur Unity, où nous utilisons l'option Collaborate afin de n'avoir qu'un seul projet que nous pouvons modifier en temps réel. Cet outil nous a été d'une grande aide pour notre travail d'équipe et pour l'avancement du projet. Nous nous aidons également entre nous lorsque nous découvrons une option de Unity qui nous était inconnue mais qui se révèle très utile pour nous tous.

Pour un bon suivi de l'avancement du projet, nous organisons des réunions hebdomadaires en début de semaine. Lors de ces réunions, nous parlons de ce que chacun a fait la semaine précédente, les comparons avec nos prédictions et nous adaptons si une personne se retrouve bloqué ou a besoin d'aide.

Nous gardons constamment l'application de discussion Discord a portée de main dans lequel nous avons créé un serveur consacré au projet. Nous restons alors constamment connectés entre nous pour que tout le monde puisse avancer correctement et que cela reste un travail de groupe.

Nous profitons également des jours sans ou avec peu de cours pour nous retrouver et travailler plus efficacement sur le projet.

3. Parties Individuelles

3.1. Menu principal

3.1.1. Graphisme et boutons (Solène)

Avec l'utilisation du logiciel de dessin Krita, je me suis premièrement occupée des graphismes du menu principal, le point de départ. Le point de départ fut le fond d'écran du menu principal. J'ai alors créé une illustration qui mettrait le joueur directement dans l'atmosphère du jeu. Pour cela, j'ai pensé à une image représentant la ville du jeu, où le joueur passera la plupart de son temps. Cette image sera peut-être modifiée dans le futur.

En plus de ce fond, c'est également moi-même qui ai réalisé les textures pour les boutons. Pour ceux-ci, j'ai découvert une nouvelle option d'Unity que je ne connaissais pas. En mettant le mode Navigation des différents boutons en Explicit, il est possible de se balader de boutons en boutons grâce aux flèches du clavier, et donc sans l'utilisation de la souris.

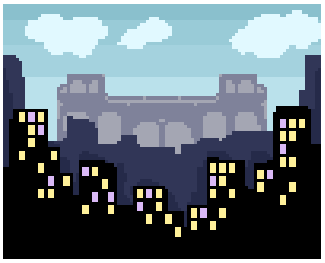
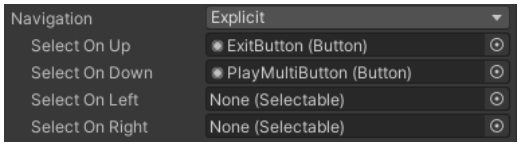


Image de fond

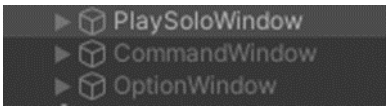
Boutons Menu Principal



Bouton Histoire

3.1.2. Transitions des boutons (Yann)

Je suis en charge de la mise en fonction des boutons du menu principal. J'ai utilisé des canvas pour que quand on appuie sur les boutons, on puisse ouvrir une fenêtre pour les autres boutons : histoire, commande et options.



Canvas

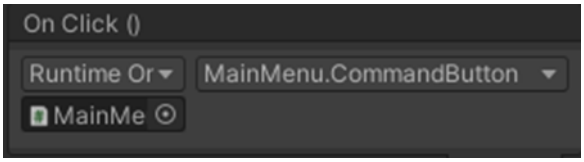
Pour que les canvas s'affichent lorsque l'on appuie sur le bouton, j'utilise la fonction suivante :

```
public GameObject commandWindow;
```

Ce qui permet que « commandWindow » soit le canvas souhaité.

```
public void CommandButton() //active le panel de command
{
    commandWindow.SetActive(true);
}
```

Cette fonction est activée lorsque l'on appuie sur le bouton grâce à la fonctionnalité OnClick qui active la fonction :



Bouton Commande

J'ai ensuite directement créé dans les panels, le bouton pour quitter le canvas. Ce bouton fonctionne de la même fonction que le bouton pour ouvrir le panel à l'exception que la fonction qui est :

```
public void CommandButtonLeave() //désactive le panel de command
{
    commandWindow.SetActive(false);
}
```

3.2. Commandes et Options

3.2.1. Musique (Abel)

Dans le but d'offrir une expérience unique aux joueurs, j'ai souhaité créer la bande son du jeu entièrement. A cette fin et avec l'aide d'un logiciel de MAO (Musique Assistée

Une capture de mon logiciel de MAO.

par Ordinateur) (FL Studio), j'ai donc créé deux des musiques servant l'ambiance du jeu.

Les scènes de combat misent à part, l'idée est de rester sur une ambiance calme. Ainsi pour une des musiques, j'ai choisi un tempo lent (90 bpm) et des accords longs sur un orgue. Par-dessus, une mélodie aigue et piquée apporte du mouvement. Ensuite, un pad et une basse réhaussent respectivement les aiguës et les graves des accords. J'ai aussi ajouté un ensemble de cordes frottées et une cloche pour de meilleures textures d'accords et de mélodie. Enfin, j'ai mixé le tout pour un rendu plus travaillé.

3.2.2. Interface et affichage (Yusuf)

Pour le menu commande, j'ai d'abord essayé d'afficher un clavier sur lequel sera afficher les différentes touches utilisées par le joueur. J'ai d'abord pris le fond d'écran que Solène à designer puis j'ai ajouté le clavier que j'ai importé d'internet. Après cela j'ai ajouté les textes pour préciser les touches que le joueur va utiliser.

Touches du clavier utilisées

J'ai également essayé de réaliser une transition entre chaque scène avec une animation sur fond noir. Mais cela n'a pas fonctionné car le code ne compilait pas. J'ai passé énormément de temps pour trouver l'erreur mais je ne l'ai pas trouvé donc je suis passé au design et à l'affichage de l'inventaire qui lui aussi prend beaucoup de temps.

Après avoir discuté avec Abel, nous allons trouver une alternative à la transition entre les scènes pour la prochaine soutenance.

3.2.3. Touches du clavier, volume et plein écran (Yann)

Pour l'onglet commande, nous avons affiché en fond d'écran une image sur le canvas du menu de commande qui montre sur un clavier AZERTY la disposition des touches et leurs fonctions.

Ensuite, dans l'onglet d'options, j'ai décidé de mettre le réglage du volume général du jeu et la mise en plein écran. Cependant, il est possible que de nouvelles options apparaissent au fur et à mesure que le jeu avance et devienne plus complet.

Ainsi pour régler le volume du jeu, j'ai mis le paramètre « Volume » en « modifiable depuis l'extérieur », puis j'ai changé le nom du paramètre du volume en « Son ». La fonction suivante permet de prendre la valeur d'un curseur pour la mettre en tant que valeur du volume :

```
public AudioMixer audioMixer;
```

Un audio mixer rentrée en paramètre ici, le volume du jeu.

```
public void SetVolume(float volume) //modifie le volume du son
{
    audioMixer.SetFloat("Son", volume);
}
```

J'ai donc créé un curseur qui varie entre –80 et 0, celles-ci étant les valeurs du volume par défaut pour les valeurs faible et élevé.

Pour le mode plein écran, la commande est dans un bouton donc et la fonction est :

```
public void PleinEcran()
{
    Screen.fullScreen = !Screen.fullScreen; //inverse l'état de fullscreen
}
```

Cependant j'ai rencontré un problème dû au changement de la taille de la fenêtre, pour remédier à cela j'ai mis la taille du canvas en modifiable en fonction de la taille de l'écran, ce qui a résolu le problème.

3.3. Graphismes et carte

3.3.1. Carte du jeu (Yann)

J'ai décidé de séparer la carte en plusieurs parties. La première, la principale où se déroule la majeure partie de l'histoire, et la seconde qui permet de rendre le monde du jeu plus complet et où se déroule une petite partie de l'histoire principale.

La première partie est l'intérieur d'une ville séparée en 5 zones et 2 axes principaux. Les zones sont :

- Une zone marchande
- Une zone de recherche (laboratoire)
- Deux quartiers résidentiels
- La mairie

Cette ville est dans une enclave de montagne et protéger de l'extérieur par une muraille.

La seconde partie est le monde en dehors des murs, qui est une zone de plaine, de forêt ainsi que de parois rocheuse. Près des parois rocheuses se situe un groupe de survivaliste qui ne souhaite pas rejoindre la ville.

Dans la zone commerciale, il n'y aura pas d'ennemis. La mairie ne sera accessible que vers la fin de l'histoire car c'est à cet endroit que la conclusion du jeu aura lieu.

3.3.2. Personnages, textures, animations et carte (Solène)

Pour commencer, je me suis concentrée sur le chara-design du personnage principal, celui-ci étant au centre de tout. N'ayant jamais dessinée en pixel art, sa conception a été la plus longue. Mais, par chance, nous avons tous été satisfait dès son premier design. Seul ses vêtements ont encore susceptibles de changer dans le futur.

Personnage principal

Ensuite, le tour des ennemis fut venu. Les gardes, chercheurs, directeurs de laboratoire, bandits, paranoïaques et le maire de la ville (dans cet ordre ci-dessous) sont les ennemis qui ont été créés pour le moment. Ceux-ci sont encore en cours de développement mais leur base ne devrait pas changer.

Garde

Chercheur

Directeur de Laboratoire

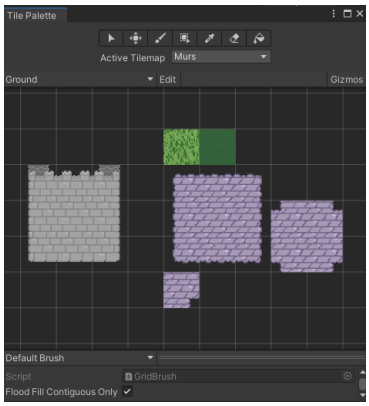
Bandit

Paranoïaque

Maire

Pour les textures, je me suis principalement inspirée des autres jeux en pixel art tel que Pokémon ou OneShot (cités dans le CDC). J'ai tendance à mettre trop de détails dans mes textures, ce qui ne donnait pas un rendu très compréhensible à cause de la petite taille de ces textures. Avec un peu d'entrainement, j'ai réussi à créer mes premières textures : des murs (qui entourent la ville), de l'herbe et des pavés (qui montrent les rues de la ville).

Afin d'appliquer ces textures dans Unity, j'ai utilisé l'outil de Unity appelé « Tilemap ». Il est alors possible de créer des « Palettes » qui contiennent les différentes images utilisées comme des pinceaux, une fois dans Unity. Il devient alors possible d'appliquer ces textures directement sur la scène de Unity. Cet outil est extrêmement pratique à utiliser et rend alors la création de carte très facile.



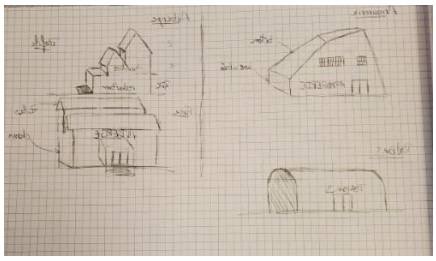
Palette de Tilemap

Pour finir, j'ai testé l'animation du personnage principal. Celle-ci est cependant encore incomplète. J'ai d'abord créé des dessins du personnage des 4 différents sens afin de les associés avec les flèches. Il suffira alors ensuite juste de créer d'autres dessins pour simuler la marche du personnage dans les différents sens et de les associer.

4 directions du personnage

3.3.3. Graphisme textures (Abel)

Dans un premier temps, j'ai fait les esquisses des graphismes sur papier. Afin d'aider Solène dans la réalisation des graphismes, j'ai d'abord dessiné les bâtiments de formes différentes afin qu'ils soient reconnaissables facilement. J'ai donc fait les dessins de l'armurerie, de l'auberge et du magasin de potions (ci-dessous) mais également du laboratoire.



Les esquisses des bâtiments cités.

Graphisme d'arbre

Graphisme sac poubelle

De plus, j'ai aussi aidé Solène à faire les graphismes de certains objets de décors comme les arbres ou les poubelles (ci- dessus) sur Krita. Cependant, mon principal rôle est de dresser la liste complète des objet et texture utile à la réalisation du projet, tant au niveau des bâtiments, des objets de décors et des objets interactifs qu'au niveau des ennemis et des personnages.

3.4. Histoire

3.4.1. Histoire principale (Yann)

L'histoire principale n'a pas énormément changé depuis sa première apparition dans le cahier des charges :

Synopsis : En sortant de l'hôpital, un portail s'ouvre devant toi. Tu distingues deux hommes armés qui s'approchent de plus en plus de toi avant que ta vue ne se trouble et que tes yeux ne se ferment. Entraîné dans un temps qui n'est pas le tien, là où règne le chaos, sauras-tu faire le chemin retour vers ta sœur alitée pour la sauver de sa maladie ?

Contexte : Dans le futur la population humaine est drastiquement plus faible à cause d'une pandémie qui ruina les contrés les unes après les autres tuant plusieurs milliards de gens, et ce malgré l'amélioration de la médecine. Le virus qui en est la cause fut surnommé le Brain Killer car il rend les contaminés complètement amnésiques en quelques mois. Pour pallier ce mal, un laboratoire se lança dans la poursuite d'un remède. En besoin de cobayes et profitant de la technologie avancée de voyage temporel vers le passé, le laboratoire décida de chercher ses patients tests dans

une temporalité différente. Le laboratoire se situe dans une des villes fortifiées construites après la crise internationale que le virus a causée. Dans ces villes se regroupent la technologie et les personnes de génies.

Histoire principale du jeu : Le joueur se retrouve donc dans ce nouveau monde et dans cette situation des plus compliqués. Le joueur doit, dans un premier temps, s'enfuir du laboratoire. Une fois sorti du laboratoire, le joueur découvre le monde et notamment les différents magasins du jeu. Après s'être familiarisé avec ce nouveau monde, il découvre qu'il existe un moyen de rentrer chez lui et cherche donc à obtenir ce moyen pour retourner auprès de sa sœur. Pour cela il sort de la ville et découvre le monde extérieur ou il apprend qu'il existe un remède pour sa sœur. Sachant cela, le joueur va chercher ce remède. Ainsi lorsque le joueur possède et le remède et le moyen de rentrer le joueur dois retourner dans son époque et soigné sa sœur.

3.4.2. Quêtes (Abel)

Afin que le joueur suive l'histoire qui lui est conté dans l'ordre chronologique, il est important de concevoir un système de quêtes et d'objectifs à atteindre. J'ai donc découpé l'histoire en chapitres contenant une ou plusieurs quêtes principales et des quêtes annexes (sorte de succès du joueur). Les quêtes se déclenchent en interagissant avec des PNJ (personnages non-joueurs) ou des objets spécifiques.

Pour avancer dans l'histoire le joueur devra compléter les quêtes principales de chaque chapitre a minima. Le jeu à jouer se termine donc à la fin de la dernière quête principale du jeu. Après cela, la partie se fini sur une cinématique. Il est à noter que tout ceci n'existe encore que sur le papier et reste encore à être implémenté.

3.5. Déplacements

3.5.1. Marche (Solène)

Afin de tester mes premières animations du personnage principal, j'ai implémenté le système de déplacements de base, soit la marche. Pour cela, j'ai créé un nouveau fichier script en C# nommé PlayerMouvement.

Script avec commentaire :

```
public float moveSpeed = 5f;
```

Création d'une variation correspondant à la vitesse de base du personnage.

```
public Rigidbody2D rb;
```

Un Rigidbody2D est créé afin que le personnage principal respecte les lois de la physique sur un plan 2D.

```
Vector2 movement;
```

La variable “movement” permet de stocker les valeurs x et y du joueur selon l’endroit où il se trouve sur la carte.

```
void Update()
{
    movement.x = Input.GetAxisRaw("HorizontalAzerty");
    movement.y = Input.GetAxisRaw("VerticalAzerty");
}
```

La fonction Update est constamment appelée et permet de mettre le jeu à jour continuellement. La variable “movement” est alors modifiée en -1 pour la flèche gauche et 1 pour la flèche droite. Le système est identique pour les flèches du haut et du bas.

```
void FixedUpdate()
{
    rb.MovePosition(rb.position + movement * moveSpeed * Time.fixedDeltaTime);
}
```

C’est cependant cette fonction qui permet au personnage de se déplacer en modifiant la position du personnage principal selon sa position actuelle, la valeur de la touche sélectionnée (fonction Update) et la vitesse du personnage. La commande Time.fixedDeltaTime permet de régler les bogues qui peuvent apparaitre avec la fonction Update.

3.5.2. Course (Yann)

Pour la course, j’ai repris le système de marche où j’ai ajouté une vérification d’activation de touches, ici la touche pour sprint qui est “right shift”. J’ai donc récupéré si elle était active ou non, puis j’ai créé une variable multiplicatrice de la vitesse. Si la touche est enfoncée alors ce facteur est de 2 (il est donc modifiable). Sinon il vaut 1 (facteur neutre de la multiplication). Voici le code :

```
if (Input.GetAxisRaw("Sprint")!=0)
{
    sprint = 2f;
}
else
{
    sprint = 1f;
}
```

Et la nouvelle ligne de code pour la vitesse où l’on multiplie la vitesse par le sprint :

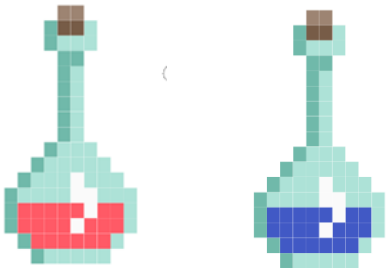
```
void FixedUpdate()
{
    rb.MovePosition(rb.position + movement * (moveSpeed*sprint) * Time.fixedDeltaTime);
}
```

3.6. Inventaire

Dans l'inventaire, il y aura donc des potions qui permettront de donner des vies ou de l'endurance, des objets trouvés dans la carte ou encore des armes.

Exemple inventaire

Ici, à la place de « use item », il y aura les objets que le joueur possède et il pourra changer avec les flèches pour voir les autres objets. Cet inventaire va donc apparaître quand nous allons appuyer sur la touche E puis disparaître quand nous allons rappuyer sur la touche E.



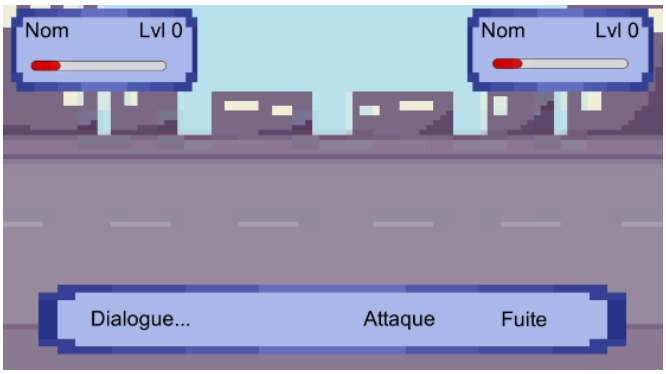
Exemples de potions

3.7. Système de combat (Solène)

Pour le système de combat, nous nous sommes basés sur du combat en tour par tour. En premier lieu, j'ai créé 2 GameObjects nommé « PlayerBattleStation » et « EnemyBattleStation » qui représentent les points d'apparition sur l'écran du joueur et de l'ennemi rencontré.

L'affichage des bulles de dialogue et de caractéristiques des personnages a été créé sur un seul canvas. On y rencontre les 2 bulles des caractéristiques des personnages contenants leur nom, leur niveau et leur barre de vie.

En bas de l'écran, on trouve la barre de dialogue avec les boutons d'attaque et de fuite sur la gauche.



Fond de combat

Pour la partie codage, c'est celle qui m'a pris le plus de temps à réaliser. Elle se découpe en 3 fichiers différents afin de rendre le tout plus clair.

Le premier fichier nommé « Unit » défini les caractéristiques d'un personnage ou d'un ennemi.

```
public string unitName;
public int unitLevel;

public int damage;

public int maxHP;
public int currentHP;
```

Dans l'ordre, on définit ci-dessus les variables du nom, du niveau, du nombre de dégâts qu'il cause, de sa vie maximale et de sa vie actuelle d'un personnage.

```
public bool TakeDamage(int dmg)
{
    currentHP -= dmg;
    if (currentHP <= 0)
        return true;
    else
        return false;
}
```

Cette fonction permet de mettre à jour la vie actuelle d'un personnage selon le nombre de dégâts de son adversaire. Elle est appelée plus tard dans le fichier « BattleSystem ».

Le second fichier nommé « BattleHUD » sert d'initialisation pour l'affichage des différents textes et de la barre de vie.

```
public Text nameText;
public Text levelText;
public Slider hpSlider;

public void SetHUD(Unit unit)
{
    nameText.text = unit.unitName;
    levelText.text = "Lvl " + unit.unitLevel;
    hpSlider.maxValue = unit.maxHP;
    hpSlider.value = unit.currentHP;
```

```
}
```

Cette fonction initialise donc le nom, le niveau, et les vies maximales et actuelles.

```
public void SetHP(int hp)
{
    hpSlider.value = hp;
}
```

Cette fonction permet simplement de varier la barre de vie selon le nombre de points de vie que possède le personnage.

Le dernier fichier se nomme « BattleSystem » et regroupe toutes les informations des précédents fichiers.

Les différents textes sont alors changés lors du combat en cours. Parmi les fonctions de ce fichier, on peut retrouver :

- Start : première fonction appelée par Unity qui lance la fonction suivante
- SetupBattle : place les personnages sur la scène, affiche le premier dialogue de combat, affiche les variables du personnage et lance le tour du joueur.
- PlayerAttack : touche en premier l'ennemi avec une attaque puis regarde si l'adversaire est mort. Si c'est le cas, il lance la fonction EndBattle, sinon c'est au tour de l'ennemi.
- EnemyTurn : fonction à améliorer. Pour le moment, l'ennemi attaque le joueur. Si le joueur est mort, elle lance la fonction EndBattle, sinon c'est tour du joueur.
- EndBattle : si le joueur est mort, le texte affiche une défaite. Si l'ennemi est mort, il affiche un texte de victoire.
- PlayerTurn : Laisse le temps au joueur de choisir un bouton
- OnAttackButton : Lance PlayerAttack si le bouton Attaque a été sélectionné.

3.8. Quitter

3.8.1. Quitter la fenêtre de jeu (Solène)

La fonction pour quitter la fenêtre de jeu est beaucoup plus simple que je ne le pensais. En effet, elle ne consiste que d'une seule ligne, se trouvant dans le script du menu principal nommé MainMenu.

Application.Quit();

La fonction Quit() permet de fermer complètement ce qui lui est associé (ici l'Application).

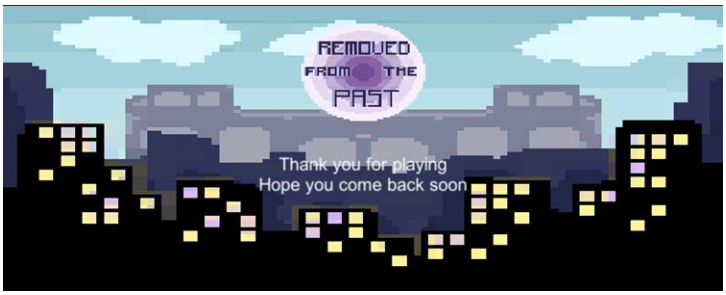
Pour effectuer les tests, la commande Debug.Log("Game closed"); est utilisée. La Console de Unity retourne alors le texte « Game closed » si le jeu s'est fermé correctement.

3.8.2. Scène de fermeture (Yann)

J'ai lié le bouton pour quitter à une nouvelle scène grâce au sceneManager. La fonction suivante me permet de passer à la scène de déconnexion.

```
public void ExitGame() //fait quitter le jeu
{
    SceneManager.LoadScene("Scene de déconnexion");
}
```

Cette scène affiche le logo un message de remerciement et le fond d'écran du jeu :



Déconnexion

Cette scène reste affichée 2 secondes puis ferme l'application avec la fonction suivante :

```
void Start()
{
    StartCoroutine(ExampleCoroutine());
}
```

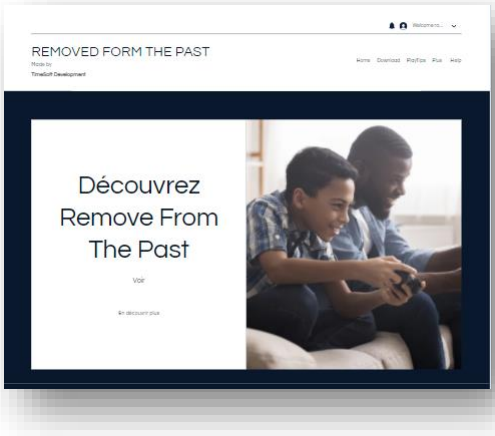
On lance la routine qui attend 2s puis on quitte l'application

```
IEnumerator ExampleCoroutine()
{
    yield return new WaitForSeconds(2);
    Application.Quit();
}
```

3.9. Site web (Abel)

Afin de garder une trace de nos travaux au plus tôt et à l'aide du site Wix.com, j'ai commencé la conception du site internet qui a maintenant sa structure finale. Nous pouvons dès à présent le remplir au fur et à mesure de notre avancé sur la réalisation du jeu.

Il se découpe en six pages principales que sont la page d'accueil, la page de téléchargement, la page d'aide et conseils pour jouer, la page recensant les musiques et sons, la page recensant le code et la page de crédits. Enfin un forum sera disponible pour les joueurs qui souhaite partager leurs expériences et leurs idées sur le jeu.



Site web

4. Conclusion

Pour cette première soutenance, nous nous sommes d'abord concentrés sur notre adaptation à l'outil Unity. De plus, dû à la nature du projet, nous avons alors plus avancé sur le côté graphisme et théorique que du côté de la programmation, même si nous n'avons pas délaissé ce point.

Le projet avance à un bon rythme malgré cela. Nous ne cessons d'avoir de nouvelles idées et de découvrir des facettes de Unity que nous ne connaissions pas. Travailler sur ce projet nous aide également en programmation en apprenant des outils utilisés par Unity.

Aucun grand retard n'a été constaté dans notre cas donc le planning pour la prochaine soutenance devrait être respecté sans problème.