

S o u t e n a n c e 2

OUMEZIANE Mustapha, LOUP Thomas, RABARISON Tiana Mélanie

Avril 2023

REVENGE OF AKIRA



Plan:

1. Introduction
2. Rappel du sujet
 - 2.1 Reprise du cahier des charges
 - 2.2 Reprise de ce qui a été fait pour la première soutenance
3. Présentation chronologique de ce qui a été fait
 - 3.1. Finalisation du menu (Thomas)
 - 3.2. Création de la 2e map(Mustapha)
 - 3.3. Mise en place de la sauvegarde(Thomas)
 - 3.4. Implémentation des différentes intelligences artificielles
(Tiana Mélanie, Thomas)
 - 3.5. Implémentation du multijoueur(Thomas)
 - 3.6. Amélioration de l'inventaire et des items(Tiana Mélanie)
 - 3.7. Implémentation du menu Loading (Tiana Mélanie)
4. Industrialisation
 - 4.1. Début du Site Web (Tiana Mélanie, Mustapha)
 - 4.2. Installateur
5. Retards
6. Prévision pour la suite
7. Conclusion

1 - Introduction

Cette deuxième période après la première soutenance a permis de développer les bases que nous avons acquises pour la première soutenance de notre jeu vidéo et d'avoir une meilleur prise en main des différents outils que nous avons utilisé dans le cadre du projet : principalement Unity et Git, avec Github et GitKraken (qui peut remplacer GitHub).

Ce deuxième rapport de projet a pour objectif de présenter avec précision l'avancement de notre projet durant la période de mars à fin avril 2023. C'est-à-dire entre la première soutenance de projet et la deuxième, dont le but est de développer les bases fondées pour la première soutenance afin d'avoir un jeu qui ressemble plus à un RPG et qui soit jouable. Dans ce rapport nous allons détailler avec précision ce qui a été fait par les membres du groupe, comment les tâches ont été réparties et quels ont été les défis auxquels nous avons dû faire face.

2. Rappel du sujet

2.1 - Reprise du cahier de charge

Pour le projet du deuxième semestre, nous avons décidé de créer un jeu en Csharp avec l'outil Unity. Nous nous sommes mis d'accord pour créer un A-RPG immersif dans un univers médiéval en 3D.

Le joueur débutera dans un village pour découvrir l'environnement du jeu avec le premier monstre à battre, puis pourra accéder à une zone plus grande constituée d'une forêt et d'un royaume ou pour finir le jeu il faudra battre une bosse finale. Il aura aussi la possibilité de combattre contre d'autres personnes en multijoueur dans une arène lorsqu'on atteint la deuxième zone.

Répartition et avancement prévu:

Le groupe est aujourd'hui constitué de 3 personnes :

- OUMEZIANE Mustapha (chef de projet)
- LOUP Thomas
- RABARISON Tiana Mélanie

Bien que nous ayons changé une petite partie du cahier des charges suite au départ d'un membre du groupe pour la première soutenance, nous avons respecté les changements pour la nouvelle répartition des tâches et les différents éléments que nous avions choisie de mettre dans notre jeu.

On vous remet l'avancement de chacun sur les différentes tâches, nous avions fixés lors de la précédentes soutenance:

Taches & Soutenances	1 ère	2ème	3ème
décors et environnement (Mustapha)	25%	60%	100%
Modélisation des personnages (Mustapha)	20%	70%	100%
Physique(Thomas)	20%	60%	100%
Réseau (Tiana Mélanie, Thomas)	10%	50%	100%
Intelligence artificielle (Thomas)	20%	70%	100%
Son (Mustapha)	50%	80%	100%
Mécanique du jeu (Tiana Mélanie)	20%	80%	100%
Expérience utilisateur(Thomas)	30%	60%	100%
Site WEB (Tiana Mélanie)	30%	80%	100%

2.2. Reprise de ce qui a été fait pour la première soutenance

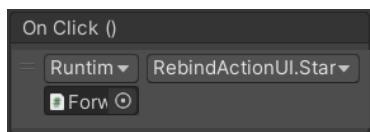
Pour la première soutenance, nous avons créé la première map avec le personnage principal, les menus, l'environnement sonore, l'inventaire avec quelques retards au niveau des IA et de l'équipement. La majorité des attentes ont été satisfaites : on arrivait à pouvoir déplacer le personnage dans la map avec ses différentes animations, à accéder au menu en pouvant modifier différents paramètres et un début d'inventaire. Nous avons également implémenté les scripts généraux tels que le PlayerMouvement ou le PlayerAttributes ainsi que les fonds sonores et la musique thème du jeu.

3- Présentation chronologique de ce qui a été fait

3.1 - Finalisation du menu (Thomas)

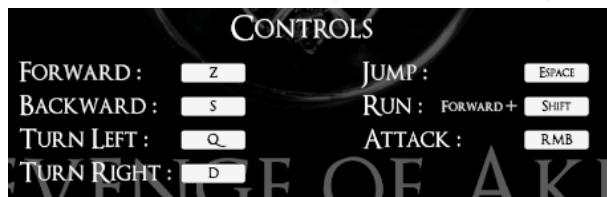
Pour permettre au joueur d'avoir un gameplay agréable nous avons mis en place un système de changement de touches. C'est-à-dire que le joueur pourra personnaliser comme il le souhaite les différentes touches (avancer, tourner, attaquer...).

Pour cela on a utilisé un module de la classe Input System (c'est avec cette classe que l'on a implémenter les différentes touches) qui s'appelle Rebinding UI. Ce module importe un script dans le projet et c'est avec celui-ci que nous avons pu modifier les touches. Nous avons seulement eu à appeler la fonction correspondante au niveau du bouton.



Dans le bouton on choisit ici quelle fonction exécuter lors du clic

Pour permettre au joueur de savoir quelles touches il a, nous avons placé un texte sur le bouton avec le nom de la touche correspondante (photo ci-dessous).



Menu des contrôles avec les différents couples (nom de l'action et bouton)

Lorsque le joueur va appuyer sur ce bouton, une fenêtre va apparaître pour nous dire de presser le bouton que l'on veut choisir, elle restera affichée tant que nous n'avons pas appuyer sur une touche ou sur un clic de la souris.

Nous avons également ajouté deux réglages pour le gameplay, le premier est la possibilité de changer la sensibilité de la caméra (qui bouge en fonction de la souris). Pour cela nous avons récupéré un composant de la cinémachine utilisé sur le personnage qui est la vitesse de déplacement de la caméra sur les deux axes que l'on va modifier dans le code sources en fonction de ce que le joueur demande.

Le deuxième réglage que nous avons ajouté sert à modifier l'axe Y (vertical) pour qu'il soit dans un sens ou dans un autre. Pour cela nous avons procédé à peu près de la même manière en récupérant un composant de la cinemachine mais ici avec une valeur booléenne qui nous dit si on inverse ou non.



on a la barre pour modifier la sensibilité et les boutons pour cocher si on inverse l'axe Y et accéder aux contrôles

Ces nouveaux réglages se trouvent dans le menu lorsqu'on lance le jeu mais aussi quand on met pause au jeu en pleine partie, ce qui permet au joueur de pouvoir modifier son environnement de jeu tout au long de sa partie.

Avec une première partie de sauvegarde effectuée (expliquer plus tard), nous avons ajouté la possibilité au joueur de charger une partie déjà commencée. Pour cela on a ajouté des boutons qui permettront au joueur de choisir entre commencer une nouvelle partie et charger une partie. Puis un autre bouton lorsque le jeu est en cours pour sauvegarder la progression.

3.2 - Crédit de la 2e map(Mustapha)

Pour la deuxième map, qui est au passage la map finale, nous avons fait en sorte qu'elle soit beaucoup plus grande que la map précédente pour montrer au joueur que l'aventure ne fait que commencer une fois la map du tutoriel fini.

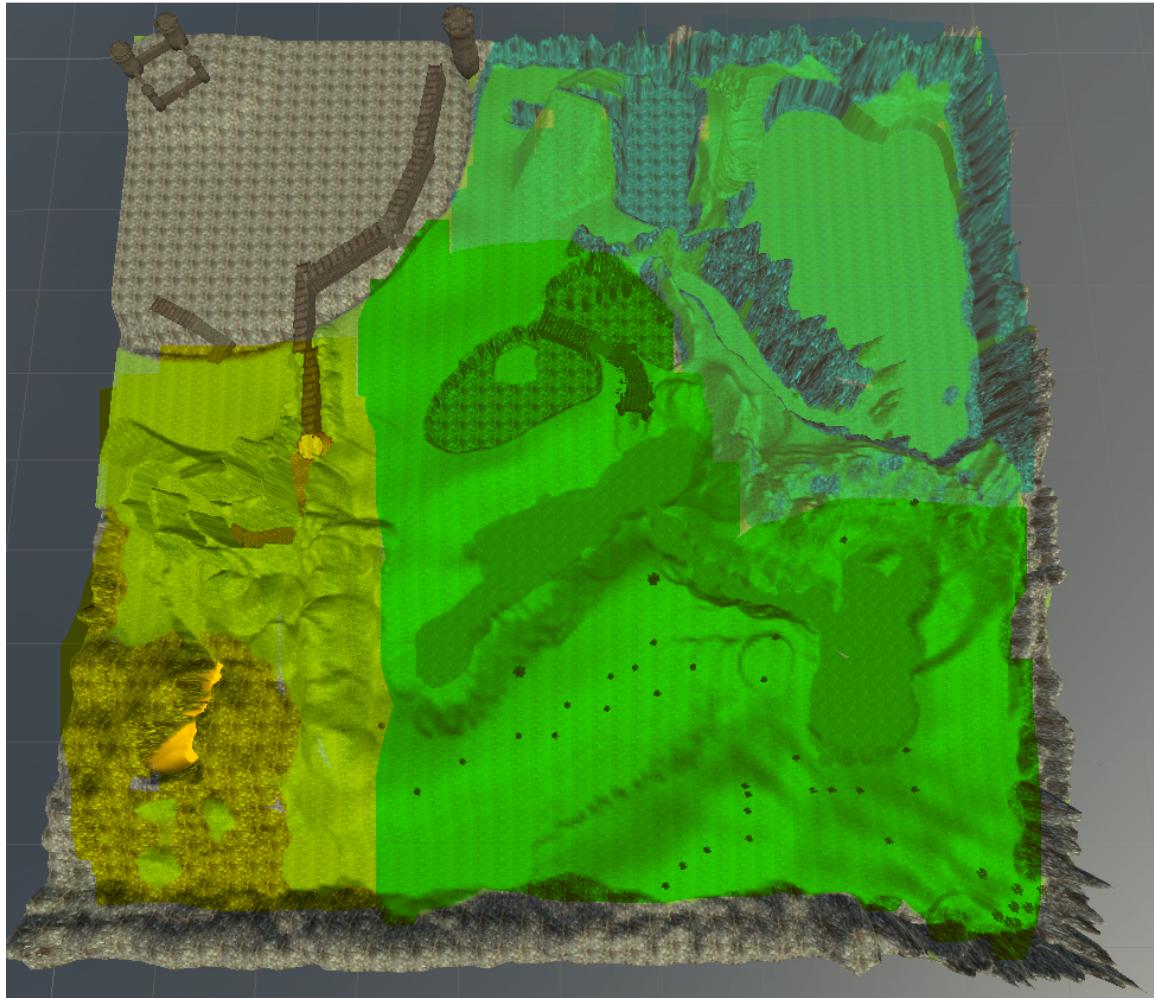
Le point fort de cette partie du jeu est sa capacité à être exploré car elle est très grande et parsemée de lieux divers à découvrir avec comme seul clé pour y accéder, la curiosité. Pour l'instant les textures ne sont pas incroyables mais nous les emploierons par la suite



point de vue du joueur au moment de l'arrivée

On peut y apercevoir au loin le château qui est entre autres la fin du jeu là où se trouve le boss final.

voici la map vu du dessus :



La map est divisée en 4 parties que nous avons mis en évidence avec des marqueurs de couleur sur la screenshot :

- La partie en vert est la partie principale avec des monstres et un pont à traverser, c'est le chemin le plus direct pour atteindre la zone non-colorée où se trouve le château de boss final. C'est aussi la zone la plus vaste et la plus simple à franchir.
- La zone en jaune est labyrinthe et cache de nombreux objets tel que des armes et des potions, elle contient moins d'ennemis et permet de contourner le châteaux et ainsi éviter les monstres les plus difficiles à combattre. Nous rappelons que les trois chemins permettent d'arriver à la zone de fin et totalement possible de revenir en arrière pour continuer l'exploration
- La zone en bleu quant à elle, est une zone très compliquée à trouver de part les reliefs qu'ils l'entourent. Mais une fois dans la zone en question, on va y trouver un boss caché qu'il aussi dur à voir plus dur que le boss final mais une fois ce boss vaincu, le joueur pourra détenir l'arme la plus puissante du jeu car c'est celle qui

délivre le plus de dégâts. Ce chemin converge également vers la zone finale qui n'est pas encore développée pour cette soutenance mais qui le sera par la suite.

La Map est munie de mur invisible aux limites des arêtes du carré pour empêcher le joueur de sortir de la map et tomber dans vide.

3.3 - Mise en place de la sauvegarde(Thomas)

La sauvegarde représente une part importante dans un rpg, car c'est des jeux qui sont souvent long à finir où le personnage augmente de niveau, c'est pourquoi tout recommencer a chaque fois n'est pas forcément ce que veut le joueur.

Unity propose différentes choses pour sauvegarder des données. Tout d'abord on a le *PlayerPrefs* *c'est une classe qui est relativement sécurisée, facile d'utilisation, et elle est* disponible d'office sur tous les supports. Mais elle possède un gros défaut c'est qu'elle est limitée dans le type de données qu'elle peut sauvegarder (entier, chaîne de caractères et float).

Nous utilisons cette classe seulement pour enregistrer des données du menu (pour garder les modifications que le joueur fait).

On avait ensuite le choix entre des sauvegardes sous des fichier JSON ou XML, les deux ont le même fonctionnement seul la manière de représenter les données va changer. Le XML structure les données avec des balises (comme un fichier HTML), tandis que le JSON structure les données avec des caractères particuliers { « key »: « value » } .

Comme Unity propose déjà des fonctions pour le JSON, nous avons choisi de l'utiliser.

Ensuite il a fallu réfléchir à ce que l'on veut sauvegarder dans une partie, de base on pensait que l'on pouvait tout sauvegarder mais le format JSON ne permet pas de tout enregistrer. On a dû donc réfléchir à ce qui était important et quels éléments pouvaient varier d'une partie à une autre. Les éléments à sauvegarder sont:

- les attributs du personnage avec ses coordonnées
- le nom de la scène dans lequel le joueur est
- l'attribut de chaque monstre avec sa position
- l'état de l'inventaire
- les items présent sur la map

Pour cette soutenance nous avons implémenté les trois premières.

Un problème rencontré assez rapidement avec JSON c'est lorsque l'on veut sauvegarder directement des données de type entier, chaîne de caractères ou float, lorsqu'on voudra enregistrer les données, JSON mettra seulement des accolades vides sans données. Après quelque recherche, on a choisi d'enregistrer les données en les mettant toutes dans une classe que l'on va sauvegarder.

Pour les monstres plusieurs possibilité ont était possible, on a d'abord pensé a utilisé une liste où on met tout les monstres du jeu. La liste composée des attributs des monstres avec leurs

coordonnées nous a amené à quelques difficultés. Bien que l'on ai essayé de différentes manières on n'arrivait pas à enregistrer correctement la liste alors que lorsqu'on a essayé de sauvegarder une liste de nombre cela marche bien. Pour résoudre ce problème, on a choisi d'attacher une sauvegarde sur chaque modèle de monstre, on a créé une classe mère *Monstre* où l'on a implémenter deux fonctions que toutes les classes qui héritent de *Monstre* définiront. C'est-à-dire que sur chaque modèle de monstre (deux actuellement) on définit les fonctions pour sauvegarder en JSON et pour lire du JSON.

```
public abstract void SaveToJson();
public abstract void LoadFromJson();
```

Fonctions de la classe mère Monstre que l'on implémente dans chaque classe fille

```
↳ Frequently called 1 usage new *
public override void SaveToJson()
{
    ...
}

↳ Frequently called 1 usage new *
public override void LoadFromJson()
{
    ...
}
```

Dans chaque monstre on définit les deux fonctions de cette manière

Ensuite, lorsqu'on a voulu créer un bouton pour que le joueur puisse sauvegarder sa partie un problème arrive, c'est comment faire en sorte de lancer la fonction sur tous les monstres car on doit sauvegarder chaque monstre. Tout d'abord lorsque nous avons testé, nous utilisions une touche pour lancer la fonction, mais au passage au bouton il fallait trouver quelque chose pour que toutes les fonctions se fassent à peu près en même temps. Pour cela on a pensé à utiliser une valeur booléenne que l'on désactive une fois la sauvegarde faite pour ne pas la répéter, mais une fois le premier monstre sauvegarder ça s'arrêtait là car le valeur est passé à faux. Nous avons finalement utilisé une coroutines qui permet de répartir des tâches sur des images, dans notre cas elle a servi à attendre que toutes les fonctions se finissent (qu'une image soit finie) puis de mettre la valeur booléenne à faux.

```
public void ToSave()
{
    IsSave = true;
    StartCoroutine(routine: TimeSave());
}

↳ Frequently called 1 usage new *
private IEnumerator TimeSave()
{
    yield return new WaitForEndOfFrame();
    IsSave = false;
}
```

code source où on met le booléen correspondant à sauvegarder à vrai puis on lance la coroutine

On va utiliser ce système pour charger les données lorsqu'on va lancer le jeu et choisir de charger une partie ou quand on sauvegarde dans le menu.

3.4 - Implémentation des différentes intelligences artificielles(Tiana Mélanie, Thomas)

Dans notre jeu les intelligences artificielles auront différents rôles, on aura des villageois et des monstres. Les villageois pourront se déplacer aléatoirement dans la map et les monstres nous attaqueront lorsqu'on se rapprochera d'eux. Pour cette soutenance nous avons choisi d'implémenter les monstres car ils nous permettront d'augmenter de niveau.

Unity va encore nous aider pour les implémenter, en effet Unity possède un module **NavMesh**. Il va nous permettre de créer une zone sur laquelle l'intelligence artificielle pourra se déplacer.

Tout d'abord, nous avons ajouté au monstre un animator comme pour le personnage qui comporte les différentes animations reliées entre elles par des transitions. Ensuite on va calculer la distance entre le monstre et le joueur, en fonction de celle-ci on effectue différentes actions. La première c'est quand le joueur est trop loin, le monstre va alors retourner à sa place d'origine si il n'y est pas et sinon va pouvoir faire divers actions (le monstre nommé "Golem" va dormir une fois à sa position initial et le monstre "Drake" va jouer une animation de temps en temps). La deuxième c'est quand le joueur est proche mais pas assez pour l'attaquer, le monstre va donc le poursuivre. Pour mettre une cible à l'intelligence artificielle, il faut donner une destination au monstre.

```
agent.destination = TargetReturn.position;
```

ici "agent" correspond au monstre et on définit sa destination par la position de "TargetReturn" qui correspond au joueur

Une fois que le monstre est arrivé à une certaine distance définie dans ses attributs, on lance l'action d'attaque. Pour effectuer celle-ci on crée un raycast qui permet de voir si on est proche de quelque chose, et si c'est le joueur qui est proche on effectue les dommages correspondants.

```
if (hit.transform.CompareTag("Player"))
{
    _PlayerActions.ReceiveDamage(_drakeAttribut.Dommage);
}
```

condition qui vérifie si "hit" l'objet proche du monstre est le joueur dans ce cas on applique les dégâts

Pour donner un peu plus de naturel au combat, le monstre "Drake" est implémenté avec deux attaques différentes qui se joue de manière aléatoire pour éviter d'avoir toujours la même.

Il ne manquera plus qu'à ajouter d'autres monstres et des villageois pour diversifier les différents éléments dans le jeu.

3.5- Implémentation du multijoueur(Thomas)

Pour le multijoueur nous avions prévu de faire une arène ou deux personnes pourront combattre et obtenir des récompenses.

Pour cela Unity propose principalement deux manières d'intégrer du multijoueur dans un projet, la première est avec Mirror, c'est une bibliothèque de mise en réseau de haut niveau pour Unity, optimisée pour la facilité d'utilisation et la probabilité de succès. Et la deuxième bibliothèque est PUN (Photon Unity Networking) de Photon c'est la base solide pour tout type de jeu multijoueur en salle dans Unity 3D.

Après quelque recherche nous avons décidé d'utiliser Photon car c'est celui qui correspond le mieux à ce que l'on veut faire par rapport à la création de nouvelle salle, la gestion des salles déjà existantes. Et il fait un travail derrière pour nous faciliter la gestion du serveur.

Le retard majeur de projet se situe ici, nous n'avons pas encore commencé le code source de la partie multijoueur mais nous avons déjà réfléchi à ce que l'on voulait faire exactement. Une fois arrivé sur la deuxième map, le joueur verra apparaître un bouton dans le menu pour pouvoir jouer en multijoueur. Ensuite, une nouvelle fenêtre apparaît et différents choix sont possibles: on pourra créer une nouvelle salle, rejoindre une salle déjà créée par un autre joueur ou revenir en arrière.

Lorsque deux joueurs auront rejoint une salle, le combat pourra commencer. Pour tenir informer le joueur sur leurs états nous ferons en sorte que chaque joueur voit la barre de vie de l'autre mais aussi un pseudo que l'on demandera de définir avant de rentrer dans la salle. Pour éviter les grande différence de niveau entre les joueurs on bloquera l'inventaire pour éviter que certain ai plein de potions et d'autres non.

Voici quelque fonction de PUN:

```
PhotonNetwork.ConnectUsingSettings("1.0");
```

la principal fonction, elle nous permet de nous connecter au serveur

```
public void OnConnectedToMaster()
{
    PhotonNetwork.CreateRoom("Room42", true, true, 4);
```

celle-ci permet de créer une nouvelle salle, les paramètre sont le nom, si d'autre peuvent la trouver, si on peut la rejoindre et son nombre max de joueur

Avec cela on utilisera les fonctions pour rejoindre dès parties soit en aléatoire soit en choisissant.

3.6 - Amélioration de l'inventaire et des Items (Tiana Mélanie)

L'inventaire est une partie essentielle d'un jeu vidéo. Grâce à l'implémentation de la physique, l'implémentation de l'inventaire a pu se mettre en place pour la première soutenance.

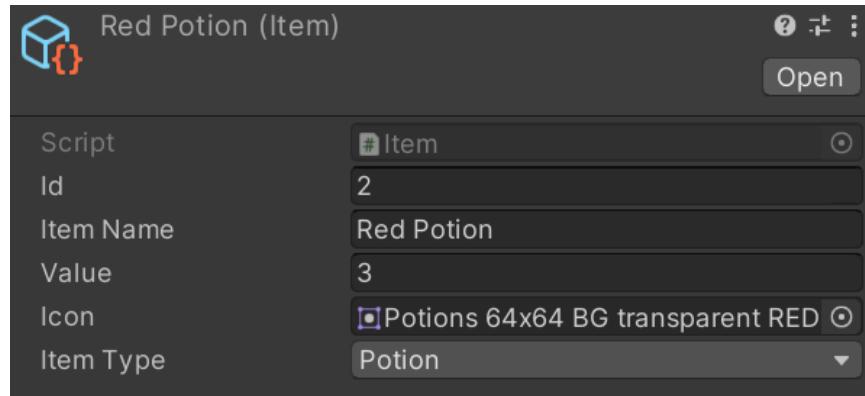
L'inventeur du jeu se fait pas la création d'un nouveau Canva et ensuite d'un scrollView où l'on insère des content pour la place de chaque item.

Ci-dessous vous pouvez voir l'affichage de l'inventaire.



Tout d'abord on a dû améliorer le principe de ramassage des objets. Pour la première soutenance, pour reprendre un objet c'était grâce à la touche "E", maintenant il se fait en cliquant sur l'objet à l'aide de la souris.

Pour permettre l'affichage de chaque items dans l'inventaire , nous avons créé une fonction qui prend en paramètre une liste et qui ajoute à chaque fois les items en fonction des attributs qu'on leur a donnés notamment le nom, l'icon et la valeur.



un exemple sur le “Red potion”

```
public void ListItems()
{
    //pour éviter les clones.

    foreach (Transform item in ItemContent)
    {
        Destroy(item.gameObject);
    }

    foreach (var item in Items)
    {
        GameObject obj = Instantiate(InventoryItem, ItemContent);
        var itemName = obj.transform.GetComponentInChildren<TMP_Text>();
        var itemIcon = obj.transform.GetComponentInChildren<Image>();
        var removeButton = obj.transform.Find("RemoveButton").GetComponent<Button>();

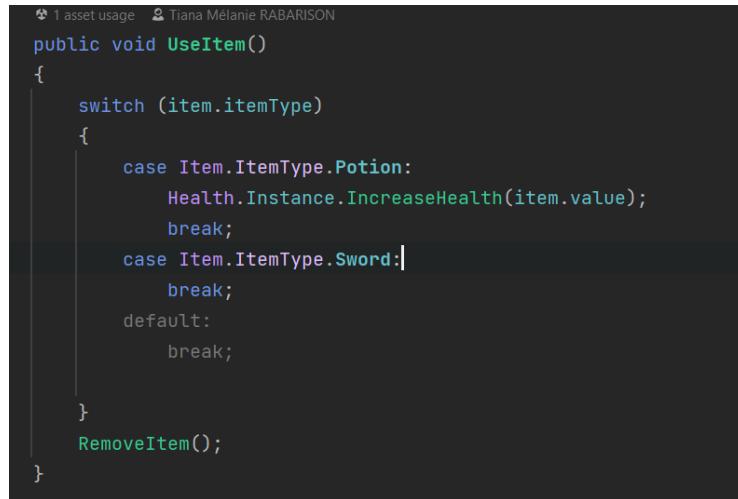
        itemName.text = item.itemName;
        itemIcon.sprite = item.icon;
    }
}
```

Cette fonction a donc permis l'affichage de chaque item, en prenant leur nom et leur icon respective.

Par la suite nous avons également mis le bouton “Enable Remove” qui permet d'afficher les boutons “close” lorsque le joueur le souhaite et ainsi faire disparaître l'objet de l'inventaire grâce à la fonction `EnableItemsRemove()`.



Après avoir récupéré les items, le joueur peut utiliser évidemment les items. Pour cela on a dû implémenter chaque items. Pour le moment on a défini deux types d'items : "Potion" et "sword" . Ainsi, on a pu écrire la fonction qui permet d'utiliser les items en fonction de leur type. Évidemment après avoir utilisé l'item, l'item disparaît de l'inventaire étant donné qu'il peut être utilisé qu'une fois.



Exemple pour les potions il permet de faire augmenter la barre de vie selon la "value" de l'item.

3. 7 - Implémentation du menu Loading (Tiana Mélanie)

Les interfaces utilisateurs sont des éléments importants du jeu. Elles permettent une personnalisation du jeu, et guident le joueur sur sa prise en main et son gameplay. Pour ce faire, des menus simples permettant de réaliser des interactions basiques ont été déjà implémentés pour la première soutenance.

Au lancement du jeu, le menu principal s'affiche, dans lequel le joueur peut choisir de lancer une partie, d'accéder aux options pour modifier ses préférences, telles que les touches, le volume du jeu, ou encore ses options graphiques. Il y a également un bouton pour quitter le jeu.

Pour effectuer les interfaces, nous avons créé des "canevas" et des objets graphiques d'Unity qui permettent l'affichage au niveau de l'écran. Un canva peut être complété par d'autres objets graphiques (bouton, text, menu déroulant ...) qui pourraient être liés avec le code source des options à l'interface graphique.

Au démarrage, le joueur arrive sur le canva du menu principal puis peut accéder aux autres *canvas grâce à des boutons qui activent ou désactivent les canevas (les rendre visibles ou non)*.

Mais on a remarqué qu'en voulant lancer la gameplay, ça prenait du temps. C'est pourquoi on a décidé de faire une interface de chargement pour le jeu.

Alors pour ce faire, on a dû créer un nouveau Canevas qui s'appelle "LoadingScreen", et on a remis les textes, les logos, ainsi que le même fond. Mais pour faire la barre de chargement on a ajouté un "Panel". Qui s'affiche lorsque le boutons qui activent ou désactivent les canevas s'activent.

L'image ci-dessous correspond à l'interface :

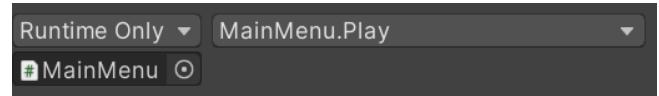


Canvas du Menu avec chargement



Canvas du Menu lorsque la bar chargement se charge

Pour cela on a utilisé un module de la classe MainMenu (c'est avec cette classe que l'on a implémenter la fonction permettant de passer au menu au LoadingMenu en attendant que le gamePlay se charge) qui s'appelle Play. Ce module importe un script dans le projet et c'est avec celui-ci que nous avons pu passer d'une interface à une autre.. Nous avons seulement eu a appelé la fonction correspondante au niveau du bouton.



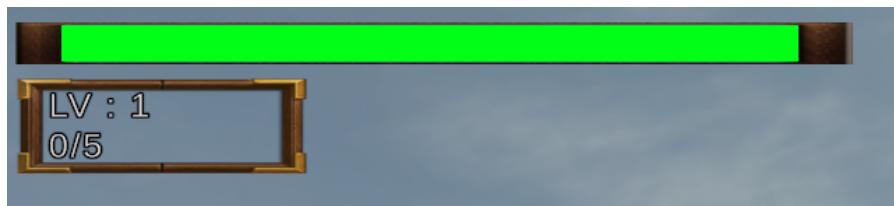
Le script ci dessous correspond à deux fonction qui a pour but d'afficher le menu de loading en attendant que la gameplay charge et de faire scroller le “Fill Amount” en fonction du chargement de la GamePlay.

```
1 usage  Tiana Mélanie RABARISON
public void LoadScenes(string scene)
{
    loadingScreen.enabled = true;
    bar.fillAmount = 0;
    List<AsyncOperation> _ops = new List<AsyncOperation>();
    _ops.Add(item: SceneManager.LoadSceneAsync(scene));

    StartCoroutine(routine: Loading(_ops));
}

Frequently called  1 usage  Tiana Mélanie RABARISON
private IEnumerator Loading(List<AsyncOperation> _ops)
{
    for (int i = 0; i < _ops.Count; i++)
    {
        while (!_ops[i].isDone)
        {
            bar.fillAmount = _ops[i].progress;
            yield return null;
        }
    }
    loadingScreen.enabled = false;
}
```

Mais si on devait parler des interfaces, on a également améliorer l'interface de la barre de vie ainsi que celle du niveau.



La barre de vie et le niveau

4- Industrialisation

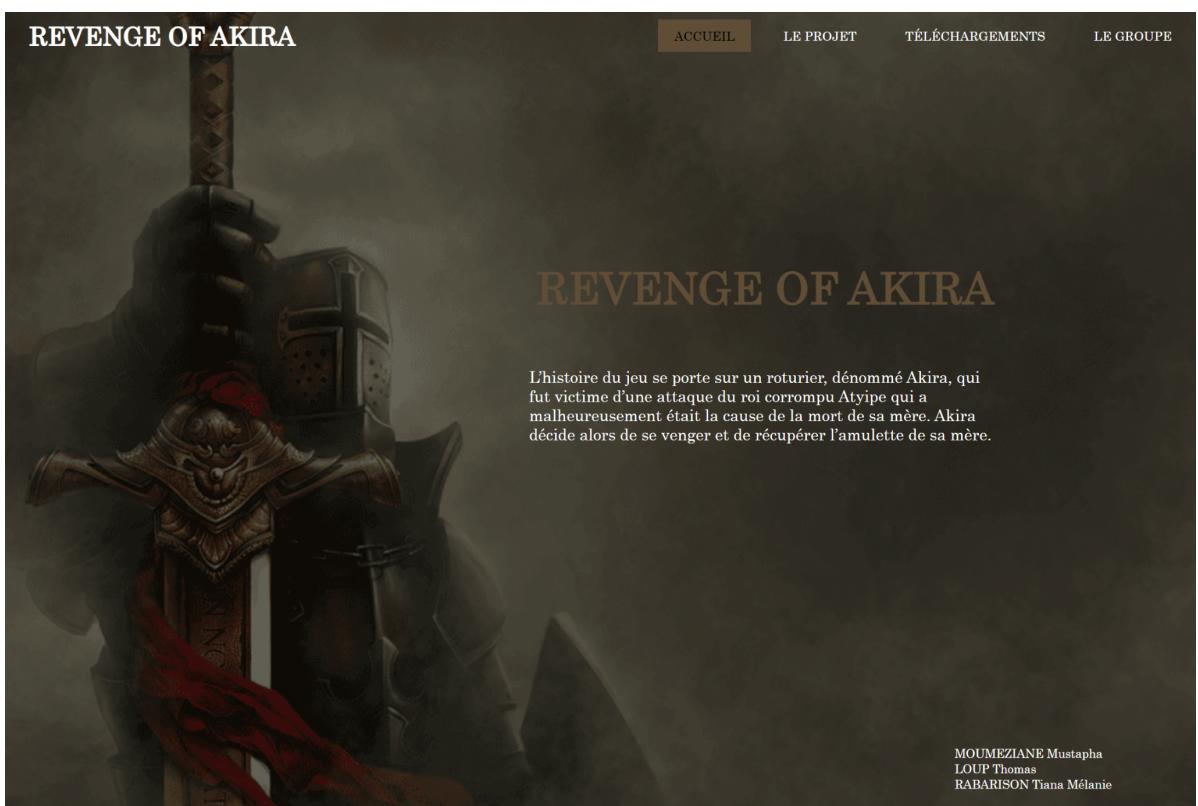
Pour que notre jeu devienne visible il faut trouver un moyen de le rendre publique et accessible. Pour se faire, nous avons décidé de créer un site web pour faire découvrir ce projet, attiser la l'attention et surtout promouvoir notre jeu. Nous avons au préalable fait l'installateur du jeu pour le compresser et le distribuer de manière plus efficace et plus accessible.

4.1 Site web

Le site web a également été travaillé pour cette soutenance. En effet il contient :

- La page d'accueil : où l'on retrouve le nom du jeu, le synopsis de l'histoire et également le nom des membres du groupe
- Le projet, qui est une page qui va expliquer les raisons de notre projet
- La page de téléchargement contient des liens pour télécharger le jeu et les fichiers en rapport avec le projet avec notamment le cahier des charges et les rapports de soutenance.
- La page de présentation du groupe, qui parle de ce dernier et de ses membres.

L'image ci-dessous correspond au menu tel qu'il est affiché sur le site web que nous avons développé.



REVENGE OF AKIRA

[ACCUEIL](#)[LE PROJET](#)[TÉLÉCHARGEMENTS](#)[LE GROUPE](#)

Dans le cadre de notre première année à l'EPITA, tous les étudiants ont pour objectif de réaliser un projet. L'intérêt principal de ce projet est d'appliquer les différentes notions vues en cours afin de créer un projet dans son intégralité. A la fois stressant et intriguant, il s'agit d'un processus tout nouveau pour la plupart des étudiants. Nous avons eu de longues discussions et ce qui en est ressorti était l'envie commune de créer un jeu vidéo. C'est alors que de nombreuses idées tantôt bien trop ambitieuses tantôt peu motivantes en ressortaient. Revenge of Akira naît petit à petit en recueillant les différentes idées issues des membres du groupe. Ce site est destiné à voir les avancées de notre jeu tout au long de sa préparation. Vous pourrez aussi découvrir les membres du groupe, les rapports du projet, nos avancements ainsi qu'un lien de téléchargement du jeu.

REVENGE OF AKIRA

[ACCUEIL](#)[LE PROJET](#)[TÉLÉCHARGEMENTS](#)[LE GROUPE](#)

Inno Setup
Compiler

4.2 L'installateur

Afin de créer un installateur facile d'utilisation, nous avons décidé d'utiliser Inno Setup, qui permet de créer, à partir de leur propre langage de programmation (Inno Setup Script) des installateurs facilement utilisables. En indiquant l'emplacement de chaque fichier et dossier nécessaires à l'installation, il crée un exécutable qui fabriquera ensuite les fichiers dans l'ordinateur de l'utilisateur. Le principal avantage de cette méthode est la réduction de place : Revenge of Akira fait environ 1 gigaoctets, mais son installateur ne fait que 250 megaoctets, ce qui permet donc de l'héberger plus facilement sur le site internet du jeu.

Un script InnoSetupScript est principalement constitué de définition de variables et d'indication de chemins vers les fichiers locaux. La tête du script définit d'abord les caractéristiques de l'application

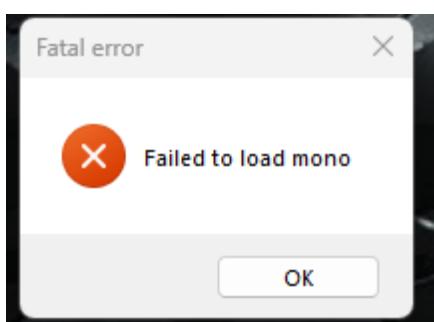
```
#define MyAppName "My Program"
#define MyAppVersion "1.5"
#define MyAppPublisher "My Company, Inc."
#define MyAppURL "https://www.example.com/"
#define MyAppExeName "Ptojet S2 2.0.exe"
#define MyAppAssocName MyAppName + " File"
#define MyAppAssocExt ".myp"
#define MyAppAssocKey StringChange(MyAppAssocName, " ", "") + MyAppAssocExt
```

Définition temporaire des caractéristiques de VOA

Il est ensuite nécessaire d'indiquer l'emplacement où l'installateur doit se construire. Pour cela on créer un dossier build avec tous les éléments assemblé de notre jeu depuis Unity

📁 .idea	19/04/2023 13:07	Dossier de fichiers
📁 Assets	20/04/2023 01:00	Dossier de fichiers
📁 build	20/04/2023 01:01	Dossier de fichiers
📁 Documents Projet	19/04/2023 13:07	Dossier de fichiers

cependant, nous avons dû faire face à une erreur jusqu'alors non corrigé et par manque de temps, nous la corrigerons pour la soutenance finales



5- Retard

Pour cette soutenance nous avons un principal retard sur la partie en réseau (multijoueur). Du au différents problème rencontré lors de l'implémentation de la sauvegarde nous n'avons pas eu le temps de commencer le multijoueur.

Commencer le multijoueur sans avoir fait la sauvegarde allait être compliqué car lorsque le joueur va arriver dans l'arène (partie réseau) on va devoir charger ses attributs de sa partie. Pour compenser le retard nous avons fait des recherches sur l'implémentation du multijoueur dans Unity avec Photon et nous avons réfléchi sur quels éléments nous allions mettre (nouveau menu avec différent choix comme créer une salle, rejoindre une salle) pour pouvoir implémenter directement dans le jeu.

Nous avons aussi un retard au niveau de l'implémentation du son mais nous allons y remédier pour la prochaine soutenance.

Nous sommes conscient des difficultés que l'on peut rencontrer lors de l'implémentation du multijoueur (problème de gestion du personnage, d' animation, les différentes données lié au personnage) mais cela n'empêchera pas de l'implémenter correctement pour la prochaine soutenance.

Pour les autres tâches, nous sommes dans l'avancement indiqué sur le cahier des charges, au niveau de l'expérience utilisateur nous sommes un peu en avance avec tous les menus implémentés. Il ne manquera plus que la partie correspondant au multijoueur.

6 - Prévision pour la suite

Les prévisions majeures:

- Implémentation de mode multijoueur
- Finalisation de la map finale
- Implémentation des boss
- Finaliser le système d'équipement
- Finaliser le système du sons
- Fixer l'erreur du mono file dans l'installateur
- Finaliser le site web
- Ajout des derniers éléments à la sauvegarde

Les prévisions mineures:

- fixer les bug de tremblement de caméra
- faire une porte de passage de la première zone à la deuxième zone
- disposer tous les items
- améliorer le visuel des éléments informatifs comme la barre de vie

7- Conclusion

Revenge Of Akira a bien avancé malgré quelques petits retards après la première soutenance. Nous avons bien rattrapé ce retard mais il en reste encore un avec le multijoueur, mais celui-ci n'impactera pas le rendu de la dernière soutenance et sera fait dans les temps. Bien que nous ayons rencontré un retard à cause de ceci, nous ferons en sorte que tout se passe bien pour la prochaine soutenance. Chaque membre du groupe à avancer sur ces tâches et à suivi de nombreux tutoriels ainsi que nos recherches personnelles ont permis d'aboutir à une nouvelle version du jeu.

Nous sommes conscients du travail déjà fourni et de celui qu'il reste à accomplir, et restons motivés pour la suite de notre projet.