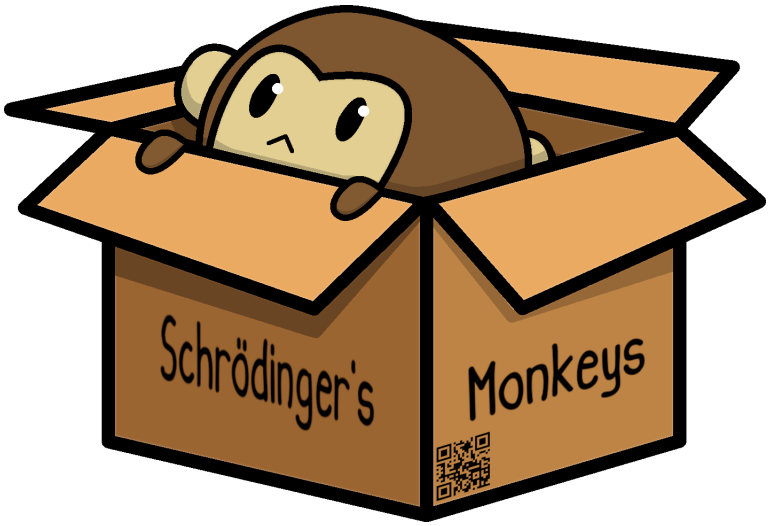
Rapport de soutenance 1

Projet Fracture



Groupe Schrödinger's Monkeys

- Toan Gaucher-Kriete
- Rapahël Leroy
- Adam Nessaibia
- Maëlys Rimbert

Mars 2021

Table des matières

1 Introduction

Fracture est un jeu vidéo de plateformes en deux dimensions, axé sur la coopération de deux joueurs évoluant dans des univers parallèles, en écran scindé. Le jeu prend place dans la Grèce antique et permet uniquement à 2 joueurs de jouer en multi-joueur.

Notre équipe est soudée et arbore un nom singulier : les Singes de Schrödinger. Le logo de notre équipe se trouve sur la page de couverture de ce rapport. Vous pourriez demander ce qu'est un singe de Schrödinger ; eh bien en réalité, c'est ce que nous sommes tous. Des singes qui, dès lors qu'ils réalisent leur condition simiesque, changent d'état pour devenir humains. Nous remercions M. Ternier, grâce à qui nous avons enfin pu devenir non-singes.

Nous nous sommes tous mis d'accord sur le principe du jeu, dès le début : à terme, la mécanique principale du jeu consistera, pour les joueurs, à échanger de monde afin de surmonter les épreuves mises en place dans les niveaux. Le but est que les joueurs aient besoin l'un de l'autre pour avancer dans deux mondes distincts, d'où le nom *Fracture*.

Nous avons travaillé dur pour cette première soutenance, et nous sommes fiers de vous présenter le premier livrable de *Fracture*. Dans le rapport qui suit, vous en apprendrez plus sur l'avancement du projet, les difficultés que nous avons pu rencontrer jusque-là, mais aussi nos avances, nos retards, et enfin nos prévisions pour la soutenance n°2.

2 Méthodes de travail

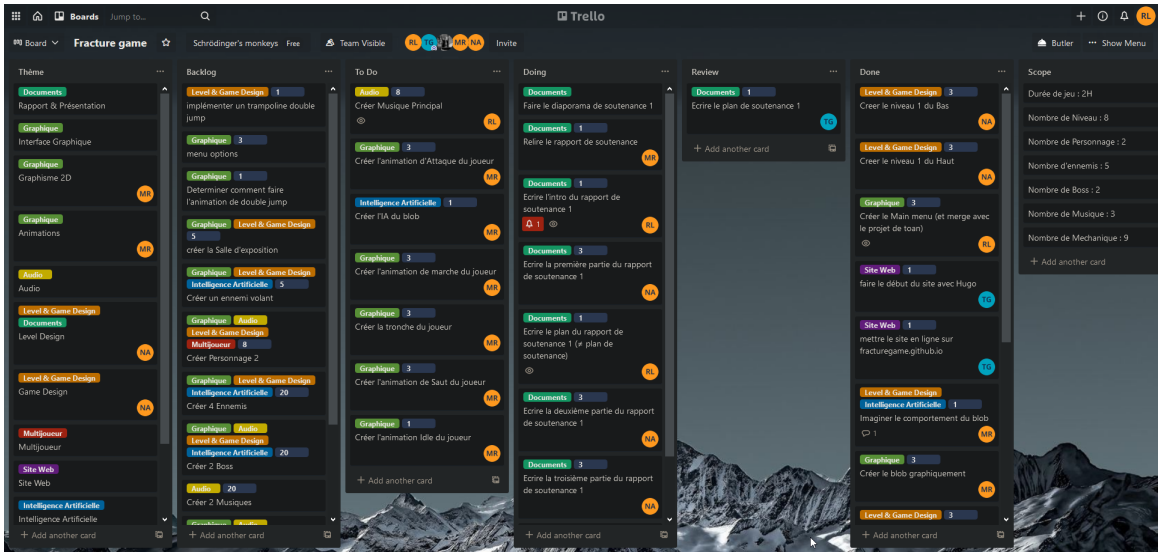
2.1 Travail en équipe

Ce projet est pour nous le premier avec une très grande importance des méthodes de travail de groupe, contrairement à ceux que nous avons pu avoir au lycée où les outils et méthodes que nous utilisions passaient au "second plan". Nous nous sommes rapidement rendus compte que sans organisation et bons outils pour mettre en commun notre travail, nous allions avoir des difficultés pour réaliser le projet.

Nos méthodes de collaboration au début du projet étaient relativement peu efficaces : en effet, nous avions commencé à travailler chacun sur des projets Unity différents, sans trop penser à comment nous allions fusionner nos résultats avant la soutenance. Cela nous a côuté un temps non négligeable passé à mettre en commun notre travail à la main, scène par scène tout en adaptant les scripts. Nous avons cependant réussi à prendre ce moment relativement en avance, afin de pouvoir repartir sur une base viable pour la collaboration avec git et GitHub.

Une fois le dépôt git créé il fut plutôt facile de travailler chacun de notre côté, nous n'avions pour la plupart du temps pas de conflits (plus de détails dans la section **Difficultés communes**). Nous avons pu réaliser différentes fonctionnalités chacun de notre côté comme par exemple l'interface utilisateur, les niveaux, le réseau et les graphismes puis les réunir dans notre branche principale sans soucis.

Pour nous organiser, nous avons commencé à utiliser Trello pour se définir des objectifs à atteindre par périodes de 2 semaines. On nous a conseillé pour travailler la méthode "agile", avec un système où chaque mise à jour / fonctionnalité doit être validée par au moins un autre membre du groupe pour la confirmer.



Capture d'écran de notre Trello

Cela nous a aussi permis de nous concentrer sur des fonctionnalités *requises* pour la première soutenance afin d'avoir un produit présentable avec lequel on pourrait effectuer une démonstration relativement complète pour le travail que l'on a fourni.

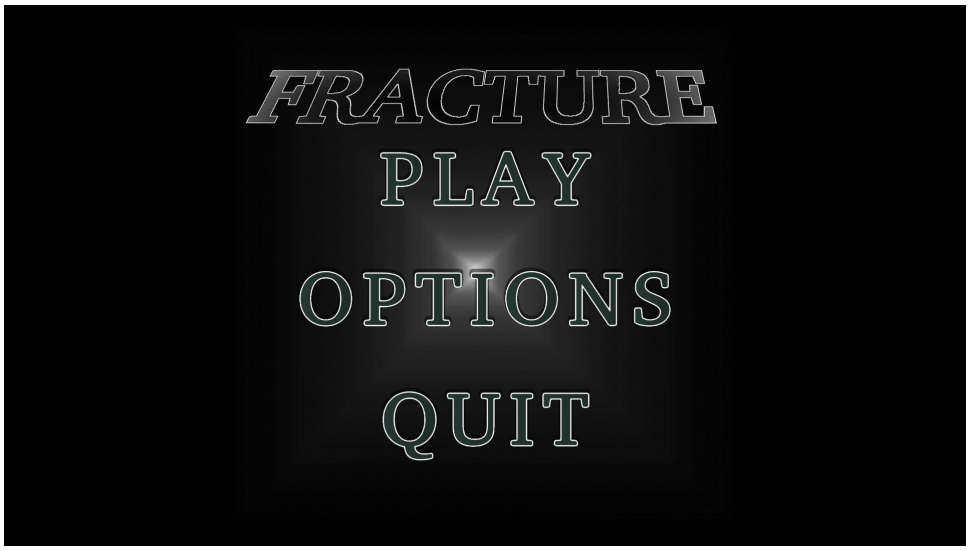
2.2 Recherches

Unity est une première pour tout le groupe, et nous avons naturellement dû faire beaucoup de recherches sur Internet pour nous documenter. Nous avons beaucoup utilisé de tutoriels sur YouTube et les forums Unity pour résoudre nos problèmes et apprendre comment réaliser certaines tâches comme une interface, un multijoueur, des animations et autres. Nous avons notamment utilisé la chaîne YouTube Brackeys qui a énormément de ressources pour Unity mais aussi sur le CSharp, qui, couplé avec Unity, possède beaucoup de syntaxes qui nous sont actuellement inconnues.

3 Avancement du projet

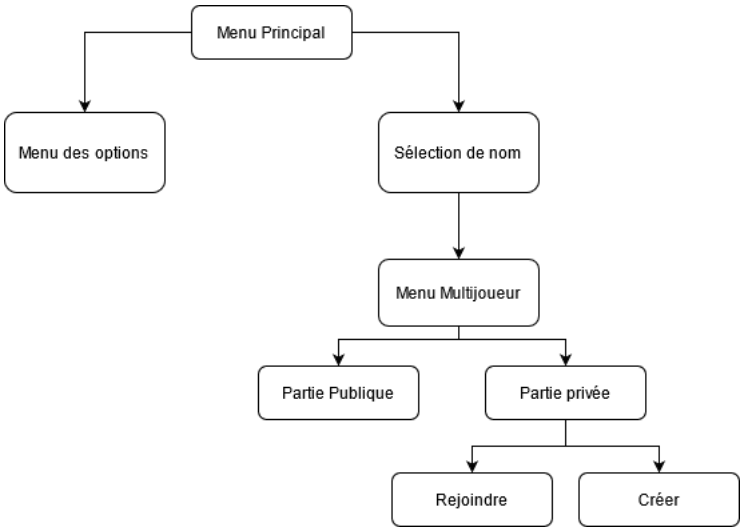
3.1 Interface graphique (Raphaël)

L'interface graphique a été faite en parallèle de la partie multijoueur, dans un projet Unity différent. Celle-ci comporte un menu d'options qui pour l'instant permet de changer la résolution du jeu, un bouton pour quitter le jeu ainsi que différents menus qui sont directement liés au multijoueur. Vous pourrez trouver plus de détails sur la liste des menus dans l'organigramme en page suivante.



Menu principal

Le menu principal, le menu des options et le menu de sélection de nom sont sur la même scène et sont affichés / effacés à l'aide de la méthode d'Unity *gameObject.SetActive(bool)* qui est appelée lorsqu'on clique sur l'un des boutons. Chaque élément d'un menu est en effet "enfant" d'un gameObject vide (sauf dans certains cas où il aura un script CSharp attribué), qui est activé ou désactivé en fonction du bouton cliqué par le joueur (par exemple, cliquer sur le bouton "Play" désactivera le menu principal et activera la sélection de nom, et le bouton "Back" désactivera la sélection et activera le menu principal) .



Cheminement des menus

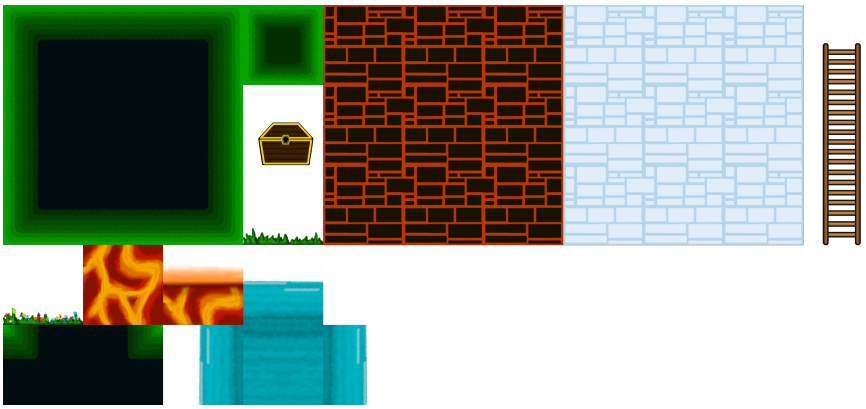
Menu du multijoueur

Ensuite, lorsque le nom a été choisi, on change de scène pour arriver sur celle des menus de jeu en ligne. Ici aussi, les différents sous-menus sont des gameObject parents des éléments de chaque menu. Nous ne détaillerons pas beaucoup le fonctionnement du menu réseau car il relève plus de la partie multijoueur, cependant il peut être intéressant de parler de celui des options. Plus d'options s'y ajouteront dans le futur mais actuellement il est possible de changer la résolution du jeu ainsi que de choisir s'il s'affichera en plein écran ou en mode fenêtré. Ceci est effectué à l'aide d'une classe Resolutions avec 2 attributs de taille d'écran, ainsi que d'un tableau de résolutions et de la méthode *Screen.SetResolution* de Unity. Le code sera ajusté plus tard pour éviter la séparation entre la liste déroulante et la liste des résolutions.

3.2 Graphismes 2D (Maëlys)

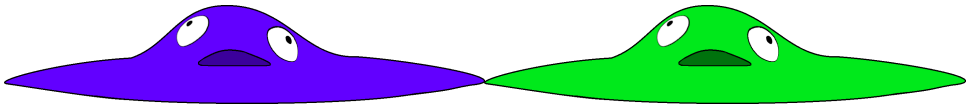
Les graphismes du jeu actuel sont tous originaux. Ils ont été réalisés par Maëlys, avec le logiciel gratuit Gimp. A ce jour, nous comptons un ennemi (le blob) ainsi

que les graphismes du sol, des murs, de l'eau, de la lave, des coffres, des échelles, de l'herbe et enfin, des fleurs.



La version actuelle de la "tilesheet", ou "ensemble de tuiles"

De plus, le blob existe un deux couleurs : vert et violet. Seul le violet est actuellement présent dans le jeu, bien que le vert soit censé être le futur ennemi du "niveau du dessus".

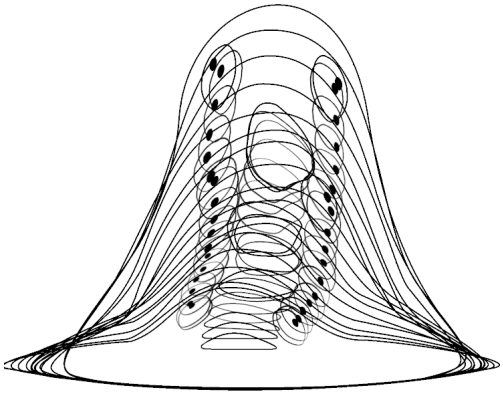


Les blobs violet et vert

Pour le moment, les joueurs sont remplacés par des carrés réalisés avec soin par Toan. Une "base" pour le dessin des joueurs a déjà été réalisée, mais elle n'est pas dans le jeu pour le moment. Vous trouverez l'art conceptuel du premier joueur en annexe.

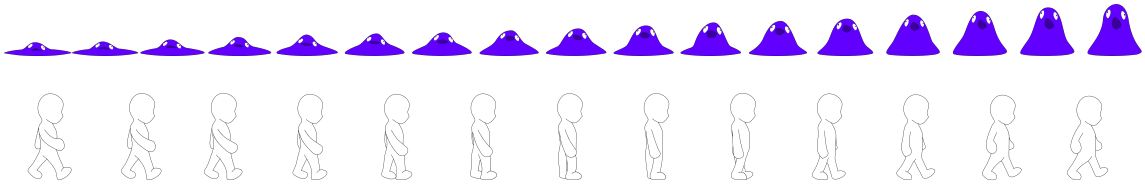
3.3 Animations (Maëlys)

L'animation du blob a été faite en dessin "image par image", pour un total de 17 images. Cette animation a été implémentée grâce à l'outil d'animation d'Unity, qui permet d'éditer une animation à partir de plusieurs images. Le blob ne bénéficiera pas d'animations supplémentaires.



L'animation image par image du blob

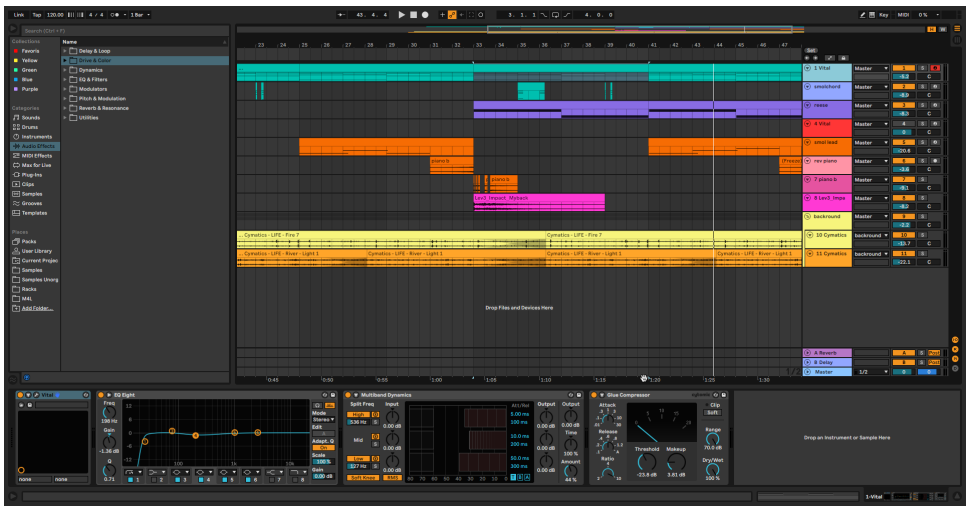
Pour ce qui est des joueurs, une animation de marche existe déjà, mais elle n'est pas encore implémentée. Celle-ci est présente sur le powerpoint de présentation de la soutenance.



L'animation image par image du blob et du personnage, utilisées sur Unity

3.4 Audio (Raphaël)

Nous avons pour l'instant un thème de menu principal qui sera bientôt adapté afin de pouvoir boucler sans "coupure". Il n'y a pas encore de musique pour les niveaux, elles seront réalisées au fur et à mesure que les niveaux avancent.



Projet du menu principal

Pour implémenter la musique (et futurs effets sonores du jeu), nous nous sommes inspirés d'une vidéo de Brackeys afin de créer un petit utilitaire qui sert simplement à jouer / arrêter des sons grâce à une simple commande qui passe par un gameObject présent dans une scène qui ne se décharge jamais.

Nous avons aussi créé une classe Sound qui nous permettra de mieux traiter nos différents fichiers sonores et de leur attribuer une liste définie de paramètres basiques pour les manipuler (volume, ton et capacité à boucler). Ainsi, si on veut jouer un son, on aura simplement à faire

```
FindObjectOfType<AudioManager>().PlaySound("Nom du son");
```

Et à l'inverse pour arrêter la lecture d'un son

```
FindObjectOfType<AudioManager>().StopSound("Nom du son");
```

3.5 Construction des mécaniques (Adam)

La construction des mécaniques regroupe plusieurs choses : les mécaniques de jeu, le timing avec lequel nous les donnons au joueur et enfin l'équilibrage de ces mécaniques afin qu'aucun des deux joueurs ne soit laissé pour compte.

Sur ces sujets, après mure réflexion, nous avons déterminé que les joueurs n'auront pas les mêmes capacités excepté pour la mécanique principale du jeu : échanger de position pour pouvoir passer certains obstacles. Au début du jeu, un des joueurs aura une capacité d'accélération lui permettant de se projeter vers l'avant à une grande vitesse. Cela lui permettra de passer de gros gouffres. L'autre joueur aura une

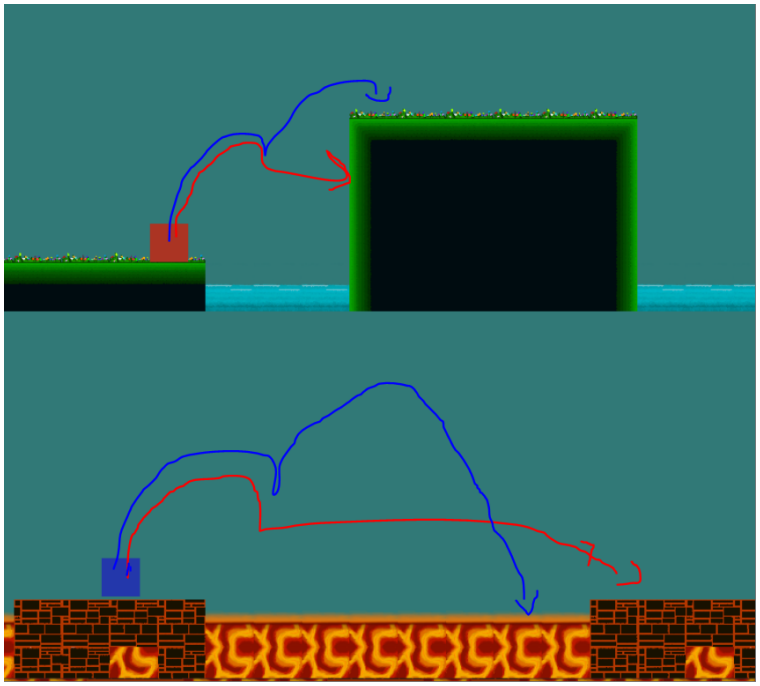
capacité de double saut lui permettant d'atteindre des positions élevées sans avoir besoin de plateformes. Avec ces capacités en main, les joueurs devront atteindre la fin du niveau. Après avoir vaincu des boss, les joueurs se verraient octroyer de nouvelles capacités pouvant être spécifiques à chacun ou communes. Cela permettrait de donner un vent de fraîcheur et de nouveaux outils aux joueurs après qu'ils aient exploré les précédentes capacités au maximum.

Les niveaux seront agencés par groupes de 2 niveaux. Chaque groupe aura des concepts particuliers. Nous pensons faire 8 niveaux et 2 boss qui devront tous tourner autour de la mécanique principale. Il faudra donc proposer des niveaux asymétriques permettant aux deux joueurs de s'amuser et de s'entraider.

Cela sera à la fois un défi technique et un défi créatif. Nous devrons réussir à être ambitieux et originaux sans pour autant nous perdre dans un torrent d'idées.

3.6 Conception de niveau (Adam)

La conception du niveau a été particulièrement intéressante et amusante tout en étant réellement difficile. Contrairement à n'importe quel jeu, *Fracture* repose sur un écran scindé en permanence. De ce fait, nous ne pouvons pas utiliser pleinement l'axe Y . De plus, nous sommes obligés de placer des éléments bloquants pour les joueurs, les obligeant à échanger leurs positions. Cette restriction se fera sous la forme de deux niveaux asymétriques ne proposant pas les mêmes obstacles à passer. C'est ce qui permet d'avoir une rejouabilité mais surtout, c'est ce qui rend notre jeu unique.



Vous pouvez voir sur l'image ci-dessus l'un des exemples de différence ; en rouge, on peut voir le premier joueur et la trajectoire qu'il peut prendre avec ses capacités

actuelles. Ensuite, on peut voir la même chose pour le second joueur, en bleu. Sur la partie du haut, le joueur doit effectuer un double saut pour passer l'obstacle tandis qu'en bas, le joueur doit effectuer une accélération. Cet exemple est rudimentaire car ce niveau est un niveau d'introduction au jeu et à la mécanique du jeu. Dans les prochains niveaux, les possibilités seront variées et demanderont de plus grandes capacités d'analyse. La mécanique d'échange peut sembler être une restriction cependant il est intéressant de les placer et la création de ce premier niveau qui est encore a l'état d'un niveau de démonstration nous a permis d'entrevoir toutes les utilisations de cette mécanique pour les prochains niveaux du jeu.

3.7 Multijoueur (Toan)

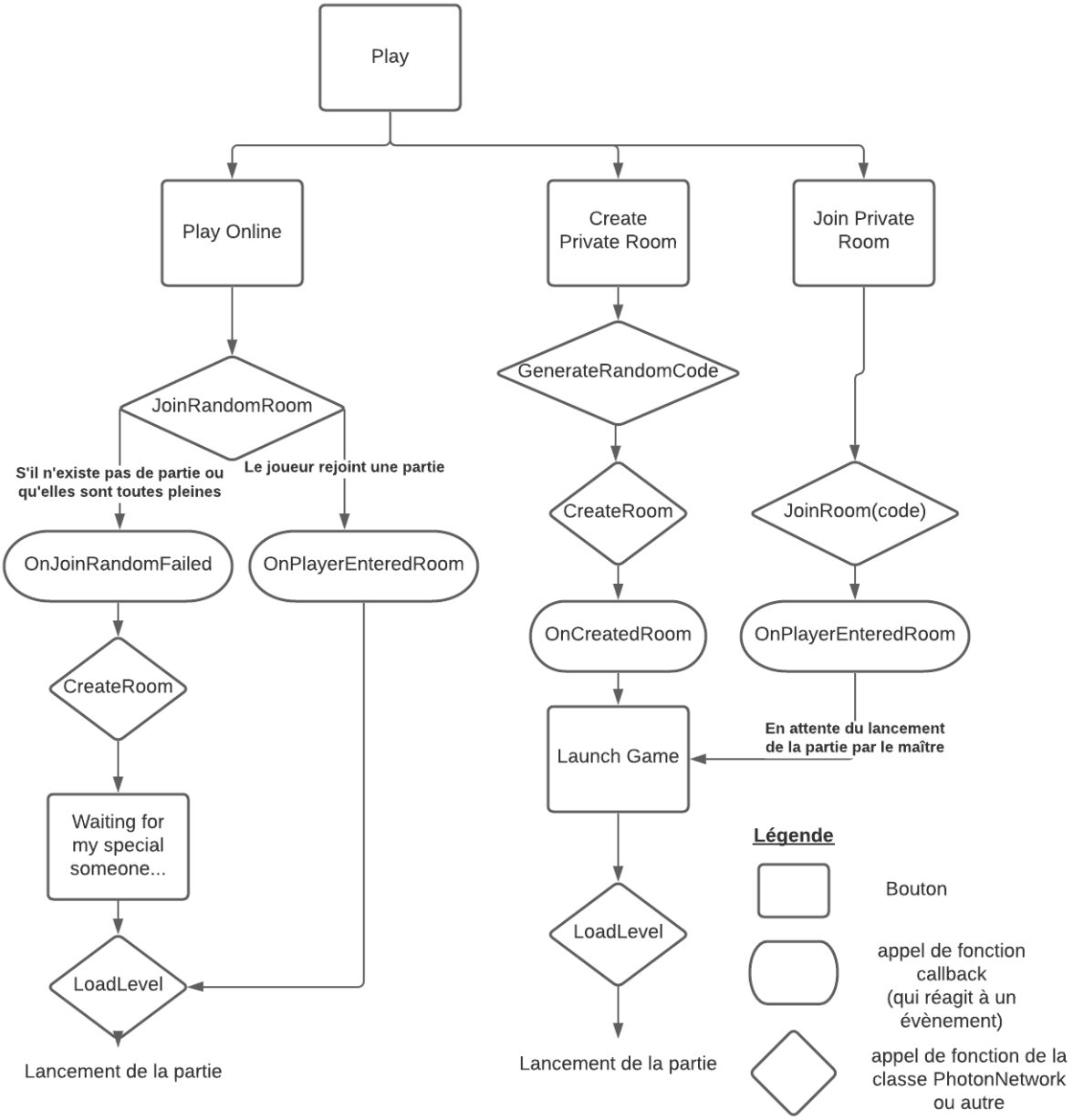
3.7.1 Photon Unity Networking 2

C'est Toan qui s'est chargé de l'implémentation du multijoueur du jeu. Afin d'y parvenir, nous avons utilisé la version gratuite de l'asset Photon Unity Networking 2 (PUN2) de Exit Games, présent sur l'Asset Store de Unity. Ce dernier permet de connecter jusqu'à 20 joueurs dans une même partie.

Après avoir installé PUN2, il a fallu lier notre projet à Photon à l'aide d'un identifiant (AppId) pour pouvoir utiliser le Photon Cloud. Il s'agit des serveurs de Photon répartis aux quatre coins du globe. Son fonctionnement est assez simple, les clients se connectent à un premier serveur qui prend connaissance de l'AppId du client, de sa version et de la région à laquelle il souhaite se connecter. Le joueur est alors redirigé à un deuxième serveur qui se charge de lister toutes les parties en cours du jeu pour une région. Ainsi, à chaque fois qu'une partie est créée ou rejointe, le client peut être redirigé vers un serveur de jeu dans lequel il pourra finalement accéder au jeu.

Enfin, Photon est particulièrement intéressant quand il s'agit de créer un jeu avec des parties comprenant un nombre limité de joueurs, ce qui est notre cas puisque nous souhaitons réaliser un jeu jouable à 2 joueurs. Cet asset nous laisse la liberté de choisir un système de partie publique ou de parties privées joignables avec un code. Nous avons décidé d'implémenter les deux !

3.7.2 Systèmes de matchmaking



Flow chart des systèmes de matchmaking

Il existe deux systèmes de matchmaking :

— les parties publiques

— les parties privées

Partie publique Dans une partie publique, s'il existe déjà une partie avec une place de libre, le joueur la rejoint et la partie se lance immédiatement. Au contraire, s'il n'existe aucun partie ou si toutes les parties sont déjà pleines, le joueur crée sa partie et attend son coéquipier.

Partie privée Afin de rejoindre une partie privée, il faut connaître le code généré par le créateur de la partie.

3.7.3 Instanciation des joueurs et ennemis

Pour instancier les joueurs et ennemis par le réseau avec Photon il faut tout d'abord créer leur prefab. Une prefab est un type particulier de composant qui permet de sauvegarder des GameObjects déjà configurés pour les réutiliser.

Puis, dans notre script GameManager.cs, attaché à la scène de jeu, nous instancions les différents objets selon qu'ils se trouvent dans la partie haute ou basse du niveau. Dans notre jeu, le joueur à l'origine de la création de la partie, aussi appelé le maître, débute toujours la partie sur la partie haute du niveau. Ainsi nous pouvons vérifier à l'aide d'une condition si le joueur est bien le maître avec

```
PhotonNetwork.isMasterClient;
```

et l'instancier avec les ennemis du haut. On fait de même pour l'autre joueur avec les ennemis du bas.

Pour instancier le joueur du client maître à la position (0,0,0) il suffit d'appeler l'instruction suivante après avoir vérifié qu'il s'agit bien du maître dans la fonction Start() de notre script :

```
PhotonNetwork.Instanciate(playerTopPrefab.name,
    new Vector3(0f,0f,0f), Quaternion.identity,
    0);
```

Ici playerTopPrefab est la prefab du joueur se trouvant sur la partie haute du niveau.

Photon va ensuite se charger d'activer la prefab à travers le réseau. Chaque prefab doit contenir un composant PhotonView qui observe plusieurs autres composants comme Photon Transform View Classic et Photon Rigidbody 2D View que nous étudierons un peu plus tard. Les prefab doivent être placées dans le dossier Ressources.

Enfin, lorsqu'un ennemi est vaincu, sa prefab est supprimée afin qu'il disparaisse chez les deux joueurs.

3.7.4 Déplacement et synchronisation

Pour pouvoir déplacer son personnage, le joueur doit donner des informations au jeu sur les mouvements qu'il souhaite effectuer à l'aide de son clavier. Cependant, si ces informations ne sont pas correctement filtrées, les joueurs risquent de pouvoir interagir avec les deux personnages. Il faut donc vérifier que le joueur interagit bien avec son personnage. Pour cela, nous pouvons utiliser le booléen PhotonView.IsMine.

```
if (PhotonView.IsMine)
{
// Deplacement du joueur
}
```

En ce qui concerne la synchronisation des personnages et des ennemis entre les deux joueurs, chacun d'eux possède les composants Photon Transform View Classic et Photon Rigidbody 2D View.

Le premier sert à synchroniser les informations de position des applications utilisant Photon. Nous avons choisi de synchroniser la position et d'activer la téléportation si jamais la différence de distance séparant le même objet chez les deux joueurs est supérieure à 3. L'option la plus importante est l'option Interpolate. Lorsqu'elle est configurée avec Lerp, elle permet de déplacer l'objet (personnage ou ennemis) avec une vitesse fixe vers sa nouvelle position.

Enfin, le composant Photon Rigidbody 2D View est également observée par Photon View et permet de synchroniser la vitesse d'un Rigidbody 2D.

3.8 Site Web (Toan)

Toan s'est vu attribuer la réalisation du site web. Notre site est consultable à l'adresse : https://fracturegame.github.io. Il a été réalisé à l'aide du générateur de sites statiques HUGO et du thème min_night.

Pour le moment, le site possède 5 pages :

1. L'accueil, qui présente le projet
2. La présentation des membres du groupe
3. L'historique de notre projet avec les liens permettant de télécharger le cahier des charges et les rapports à venir
4. La chronologie de réalisation du projet avec les liens des outils utilisés
5. Une version web de notre jeu

Le site possède également un en-tête qui sert de barre de navigation et un bas de page. Il s'adapte à la taille de l'écran sur lequel il s'affiche afin que tout le monde puisse le consulter sans gêne.

Le gain de temps apporté par l'utilisation d'un thème a pu profiter au reste du site puisque davantage de soin a pu être apporté à la création des pages en HTML et CSS. Il a tout de même fallu adapter le thème qui est pensé pour être utilisé en tant que blog.

3.9 Intelligence artificielle (Maëlys)

Le blob, premier ennemi, et ses points de cheminement en rouge

Le blob est pour l'instant le seul être du jeu doté d'intelligence artificielle. Celle-ci est simple : le blob suit un chemin défini par des points de cheminement, visibles en rouge sur l'image. Il n'est pas capable de sauter, de sortir de sa routine, et il ne peut pas attaquer le joueur. Si le joueur se met sur sa route, il est poussé par le blob. Les dégâts des ennemis seront implémentés plus tard. Cependant, le blob peut se faire tuer par le joueur : il possède 100 points de vie au départ, et chaque coup lui en retire 40. On peut frapper le blob en s'approchant de lui et en appuyant sur la touche A du clavier.

Le blob possède une boîte de collision visible en vert sur l'image. Cette boîte représente la partie solide de l'ennemi : le joueur ne peut pas passer à travers cette boîte, mais il peut marcher dessus et infliger des dégâts à l'ennemi en l'attaquant d'une distance inférieure à la distance d'attaque par rapport à la boîte de collision. Le blob peut aussi être tué si l'on saute sur sa tête. Il est donc composé d'une partie "Graphics", qui comporte son visuel ainsi que les propriétés qui y sont rattachées, et de ses deux "Waypoints" (points de cheminement).

Il a aussi fallu faire en sorte que le comportement de l'ennemi soit visible et égal pour les deux joueurs ; cela est rendu possible par Photon, qui synchronise la vue des deux joueurs.

La partie de l'intelligence artificielle a été principalement réalisée par Maëlys. Nous espérons implémenter plusieurs ennemis avec des comportements différents, par exemple un ennemi capable de suivre le joueur, ou des boss qui seraient plus imprévisibles.

4 Difficultés rencontrées et leur résolution

4.1 Difficultés individuelles

4.1.1 Maëlys

J'ai d'abord eu des difficultés d'adaptation aux outils comme Gimp ou Unity, ainsi que Git (cf. difficultés collectives). Ce problème était facile à résoudre, car il existe de nombreux tutoriels et aides sur Internet. Ensuite, le temps que prennent certaines tâches, notamment d'animation, est énorme ; c'est pourquoi j'ai décidé de changer d'approche à partir de cette soutenance, et d'utiliser davantage les outils d'animation d'Unity. Enfin, il a fallu adapter l'implémentation de l'intelligence artificielle avec le multijoueur ; avec l'aide de Toan et de quelques scripts Photon, nous avons finalement réussi à rendre l'ennemi visible et synchronisé pour les deux joueurs.

De plus, j'ai eu des problèmes techniques au niveau des logiciels ; j'ai dû télécharger et apprendre à utiliser Visual Studio car Rider ne fonctionnait pas avec Unity.

4.1.2 Raphaël

Je n'avais jamais utilisé Unity avant et il a été difficile pour moi de comprendre comment il fonctionnait, et je n'ai pas encore tout compris même si j'ai pu assimiler quelques concepts basiques tels que les objets, et des interactions basiques possibles avec les scripts. Le premier problème que j'ai rencontré en travaillant sur l'interface utilisateur était les méthodes de travail et de fonctionnement d'Unity (par exemple ce qu'était un gameObject, comment les scripts s'y appliquaient, etc...). J'ai également eu du mal à décider entre faire plusieurs scènes ou avoir tous les menus dans une seule scène, ce qui a généré d'autres problèmes quand nous avions dû mélanger ma partie avec celle de Toan qui était dans un projet Unity différent. J'ai également eu des problèmes pour changer la résolution, et je n'ai pas encore trouvé de moyen efficace d'avoir une liste de résolutions prédéfinies, de pouvoir choisir entre elles et d'avoir la même résolution lorsque l'on redémarre le jeu. J'ai personnellement eu beaucoup de mal avec git, principalement lorsque nous avons dû régler des conflits.

4.1.3 Adam

Je n'avais jamais travaillé avec tous les logiciels que nous avons utilisés, il a donc été très dur pour moi d'apprendre leur fonctionnement et je n'ai pas fini d'en apprendre à leur sujet. Cependant, cela a été enrichissant. De plus, ayant été chargé de la construction du niveau et de la création des mécaniques, j'ai eu besoin de faire des recherches pour comprendre comment réaliser mes idées. Enfin, la plus grosse problématique fut la conception du niveau ; j'ai dû tout d'abord me renseigner sur comment faire un bon niveau de plateformes, sur ce qui rend un jeu de plateformes bon. Puis j'ai commencé à produire le niveau. J'ai dû m'y reprendre à plusieurs fois en de scématisant sur papier avant de passer sur Unity. Là encore, j'ai fait plusieurs essais. Comprendre les subtilités de git a également été un défi. Enfin, l'apparence du personnage a été sympathique à imaginer.

4.1.4 Toan

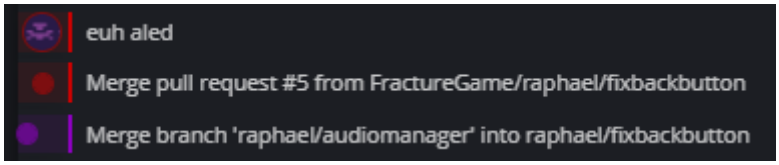
Durant cette première période de développement j'ai rencontré plusieurs difficultés. C'est tout d'abord Photon qui m'a posé problème. En effet, le fait que PUN2 soit récent (sorti en 2018) explique qu'il y n'y ait pas autant de discussions à son sujet qu'on le désirerait. J'ai aussi trouvé qu'il avait quelques fois tendances à rester dans l'ombre de son prédécesseur, PUN1, ce qui peut prêter à confusion si on commence à mélanger les deux versions. A part ça, une fois avoir lu la documentation et compris le système de callbacks et le fonctionnement des composants Photon, j'ai rapidement pu développer le multijoueur de notre jeu.

Concernant le site web, j'avais tout d'abord l'intention de le réaliser entièrement sans l'aide d'un générateur de site statique. J'ai finalement changé d'avis puisque c'était justement pour moi l'occasion d'apprendre à en utiliser un et j'ai choisi HUGO car j'en avais déjà entendu parler. La difficulté principale rencontrée en construisant le site est le fait que chaque thème de HUGO est différent au niveau structurel. Il n'y a donc pas de tutoriels universels et seuls les thèmes les plus populaires sont couverts. Ayant pris un thème peu connu, j'ai dû comprendre son fonctionnement pas à pas, fichier après fichier. Finalement, les pièces du puzzle se sont assemblées et j'ai réalisé le site en moins de 24h ce qui n'aurait pas été le cas si j'avais gardé mon idée initiale.

4.2 Difficultés communes

4.2.1 Mise en commun du travail et Git

Une des premières erreurs que nous avons faites était de commencer avec plusieurs projets Unity différents. Il a été fastidieux de "mélanger" nos différents travaux à la main, comme par exemple pour appliquer l'interface utilisateur de Raphaël à la partie jeu en réseau de Toan. Une fois ce travail effectué, nous sommes passé à git, qui a bien fonctionné hormis quelques conflits lorsque nous avons tenté de fusionner 2 branches : certains fichiers "prefabs" d'Unity avaient des changements de référence alors qu'une des branches ne leur avait pas apporté de modification, ce qui a créé de nombreux conflits de fusion. Hormis ce problème la majorité de la gestion des branches s'est plutôt bien passée, cependant nous n'avons pas trouvé de vraie solution pour les prefabs qui se changeaient "tout seuls" et nous devons donc être attentifs aux fichiers qui changent lors des futurs *commits*.



4.2.2 Compréhension des parties des autres membres du groupe

Avant même le début du projet, nous avons séparé les tâches pour chaque membre du groupe. Si cela nous a permis d'avancer plus vite sur nos parties respectives, nous avons également eu du mal à comprendre les parties de nos camarades, et pour

certains nous en avons encore (notamment concernant la partie réseau, qui est sans doute la plus complexe de ce que nous avons produit actuellement).

4.2.3 Distanciel

Nous nous en doutions, mais le distanciel n'a pas facilité le travail en groupe. Même si nous avons réussi à rester relativement organisés avec des réunions régulières, il manquait cette atmosphère de travail en équipe en face à face. De plus, même si nous avions pu nous réunir à l'école nous n'aurions pas pu tous travailler étant donné que nous ne sommes pas tous équipés d'un PC portable assez puissant pour Unity, et qu'Unity n'est à priori pas installé sur les machines de l'école.

5 Avances et retards

5.1 Produit d'amusement minimal

Note importante : Afin d'être plus conscient des avancées et retards de notre projet, nous avons choisi de ne plus les représenter sous forme de pourcentage, une présentation qui n'était pas assez concrète et qui pourrait, selon nous, se révéler sournoise. Nous avons donc décidé de nous baser sur notre produit d'amusement minimal.

Tâches	1ère soutenance	Produit d'amusement minimal
Interface graphique	Oui	Menu basique
Graphismes 2D	ensemble de tuiles, 1 ennemi (blob), 2 personnages manquant	ensemble de tuiles, 2 personnages, 1 ennemi
Animations	Animation d'un ennemi	Animation d'un ennemi et des deux personnages
Audio	1 musique de menu, pas de musique de jeu	1 musique de jeu
Construction de mécaniques	mécanique de changement de monde manquante	4 mécaniques avec le changement de monde
Conception de niveau	1 niveau (tutoriel)	1 niveau (tutoriel)
Multijoueur	Oui	Multijoueur fonctionnel
Site Web	Oui	Pas nécessaire
Intelligence artificielle	Oui, l'intelligence artificielle du blob	Intelligence articfielle d'un l'ennemi

5.2 Avancement des objectifs fixés par le cahier des charges

Interface graphique Premièrement, en ce qui concerne l'interface graphique, nous pensons être en accord avec notre cahier des charges. En effet, notre projet possède un menu dans lequel nous avons les fonctionnalités de base, à savoir le fait de pouvoir quitter le jeu, la modification des dimensions de la fenêtre, entrer son nom de joueur et choisir parmi les différents systèmes de jumelage de joueurs (partie publique partie privée). A chaque nouvelle fenêtre, il y a toujours un bouton 'Back' nous permettant de revenir en arrière et il est également possible de quitter une partie via l'interface graphique de notre jeu en informant le joueur qui reste seul de cette action.

Graphismes 2D Pour ce qui est des graphismes, Maëlys a réalisé un ensemble de tuiles complet qui compte différents sols et murs, de l'eau, de la lave, des coffres, des échelles, de l'herbe et des fleurs. Cela va permettre la création de tous les niveaux du jeu. De plus, les graphismes d'un premier ennemi ont aussi été réalisé. Pas de retard à signaler à ce niveau là non plus.

Animations Les animations, quant à elles, sont aussi suffisamment avancées pour cette première soutenance. En effet, cette première étape de développement a vu apparaître deux animations de déplacement, celle du blob et d'un personnage. La quantité de travail nécessaire à la création de ces animations a su nous mettre sur la bonne voie pour la suite. De plus, cette première période était surtout l'occasion d'apprivoiser les nouveaux outils nécessaires à la réalisation de nos tâches respectives. Ainsi, nous sommes convaincus que les animations se feront de plus en plus rapidement et efficacement dans le futur pour atteindre les objectifs des prochaines soutenances.

Audio L'audio, réalisé par Raphaël est en bonne voie sachant que nous possédons une musique de menu et qu'il y a déjà eu de nombreuses tentatives de musiques de niveau.

Conception de niveau Concernant la création des niveaux, nous sommes en mesure de montrer un premier niveau pour cette soutenance et cela répond parfaitement à nos attentes. De plus, maintenant que la feuille de tuiles a été réalisée et que nous nous sommes fait la main sur le logiciel Unity, les niveaux vont voir le jour à une cadence toujours plus rapide.

Construction des mécaniques La construction des mécaniques a pu prendre un léger retard au profit des autres tâches. En effet, la mécanique qui permet au joueur de pouvoir changer de monde n'est pas encore implémentée dans le projet. Nous avions déjà beaucoup à faire avec les autres mécaniques telles que le déplacement horizontal, le saut et double saut, la propulsion en avant (l'accélération) qui ont toutes été implémentées. De plus, nous désirions surtout présenter un projet fonctionnel et soigné à cette soutenance et la mécanique principale de notre jeu est trop importante pour être bâclée.

Multijoueur La partie multijoueur du jeu est quant à elle complètement fonctionnelle. En effet, les joueurs sont capables de se connecter par différents systèmes de jumelage (partie publique, partie privée), et les personnages et ennemis sont synchronisés par le réseau avec un temps de latence minimal. L'objectif est respecté pour cette soutenance. Il était dans nos objectifs de terminer cette tâche au plus vite et elle est en bonne voie.

Site internet Le site n'était pas une priorité majeure pour notre équipe mais nous en avons tout de même réalisé un. Il possède même un petit plus, une version web de notre jeu dans la section WebGl ! Petite avance de ce côté-là.

Intelligence artificielle Enfin, le développement de l'intelligence artificielle de nos ennemis connaît un léger retard même si notre projet comporte tout de même celle du blob. Nous avons revu à la baisse nos objectifs pour cette tâche qui étaient peut-être un peu trop ambitieux. Nous sommes désormais plus à même d'évaluer la charge de travail que représente l'intelligence artificielle et voulons être certain qu'elle sera de qualité.

6 Prévision de l'avancement à la deuxième soutenance

Interface graphique

Nous avons réussi à avoir un menu fonctionnel. A présent, nous allons essayer de l'embellir avec un écran d'accueil plus beau et sophistiqué ; vous trouverez en annexe un concept graphique pour ce menu. Nous tenterons également d'implémenter une interface en jeu, par exemple avec une barre de vie et de pouvoir.

Graphismes 2D et Animations

Maintenant que les tuiles sont effectuées, nous allons pouvoir faire les personnages et leurs animations. Nous espérons finir l'animation des personnages et celles de deux autres ennemis : un ennemi volant et un ennemi plus robuste.

Construction des mécaniques

Pour la prochaine soutenance nous allons nous concentrer sur la mécanique d'échange. Avec cela, nous allons faire en sorte de rendre les joueurs dépendants l'un de l'autre. Nous espérons réussir à implémenter le saut mural et la nage avant la prochaine soutenance.

Construction de niveau

Nous espérons réussir à produire au moins 5 niveaux d'ici la prochaine soutenance. Nous allons varier les niveaux : on pourrait avoir un/des niveaux verticaux où les joueurs devront user de sauts pour atteindre la fin du niveau (exemple : escalader une tour).

Site internet

Nous devons mettre un lien de téléchargement du jeu.

Audio

Nous espérons faire au moins 3 musiques de niveaux différentes d'ici à la prochaine soutenance.

Intelligence artificielle

Nous espérons créer une intelligence artificielle pour deux nouveaux ennemis.

7 Conclusion

Arrivés au terme de cette première période de développement, nous sommes fiers du travail accompli jusqu'à aujourd'hui et confiants quant à l'avenir du projet pour les soutenances à venir. Nous, les Singes de Schrödinger, avons acquis les connaissances nécessaires afin de réaliser nos tâches respectives et avons pris la main sur les logiciels qui nous aideront à atteindre ce but. C'est donc avec fierté que nous présentons la première démo de notre jeu *Fracture*.

Cette démo comporte déjà de nombreuses composantes du jeu final attendu pour juin. En effet, elle est composée d'une interface graphique sobre mais élégante représentant différents menus. Elle est également munie d'une musique principale et permet aux joueurs d'effectuer les actions de base que l'on attend d'un jeu vidéo. Le joueur a le choix entre plusieurs systèmes de jumelage pour rejoindre un coéquipier dans une partie. Le multijoueur est donc fonctionnel. Une fois entré dans le niveau, le joueur peut constater le travail fourni au niveau des graphismes 2D faits maison mais aussi de la réflexion nécessaire à la conception du niveau et des mécaniques de jeu. Le jeu possède aussi ses premières animations et intelligences artificielles. Il nous semble important de préciser que la partie créative de notre jeu est entièrement conçue par nous-même.

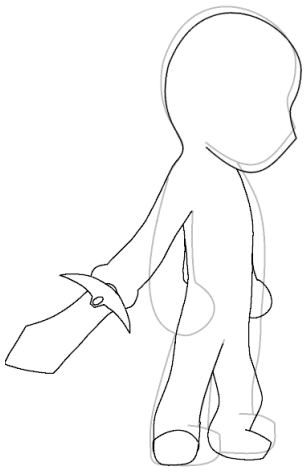
Afin d'aboutir à ce résultat, il a fallu améliorer nos méthodes de travail et notre organisation en tant que groupe. Cela s'est notamment fait à l'aide de l'application Trello et des réunions de groupe qui ont lieu tous les lundis soir.

Le chemin pour arriver jusqu'ici a été semé d'embûches mais les difficultés rencontrées ont toujours pu être surmontées, telles que la gestion de la mise en commun du travail à l'aide de Git ou encore le fait de toujours devoir travailler en distanciel.

Les tâches sont dans leur globalité en bonne voie même si certaines auraient pu bénéficier d'un peu plus d'attention. Il faudra redoubler d'efforts pour atteindre les objectifs que nous nous sommes fixés pour la soutenance intermédiaire.

8 Annexes

Concept graphique des personnages jouables



Début d'animation d'attaque, elle sera probablement abandonnée au profit d'une animation Unity

Concept graphique du menu du jeu