La Maison-Dieu

Soutenance 1.

Arthur Mallet arthur.mallet

Uriel Fellmann uriel.fellmann

Victor Blanchot victor.blanchot

Maxime Castres maxime.castres

Mars 2023

Table des matières

1 Introduction.

Bienvenue à la première soutenance de notre jeu vidéo ambitieux intitulé La Maison-Dieu. Cette création immersive plonge les joueurs dans un monde en déclin, où la magie et les légendes ont sombré dans l'oubli, devenant de simples souvenirs du passé. L'histoire se déroule dans un univers apocalyptique, où le bien et le mal s'affrontent sans relâche, à un tel point que la frontière entre les deux est devenue difficile a distinguer L'aventure épique débute brusquement avec l'apparition mystérieuse d'une tour massive au cœur du continent. L'objectif ultime de ce jeu passionnant est de parvenir au plus profond de La Maison-Dieu, où réside soi-disant le pouvoir d'un dieu. Pour accomplir cette quête, les joueurs devront non seulement maîtriser leurs sorts et invocations, mais aussi prendre des décisions stratégiques cruciales à chaque étage. Ils devront découvrir les mystères et les secrets de La Maison-Dieu. Notre ambition est de captiver les joueurs en leur offrant une expérience de jeu inoubliable, combinant une histoire riche en mythologie et en intrigue, des mécanismes de jeu complexes qui stimulent la réflexion stratégique et laissent leur place à l'expérimentation, et un défi constant pour les amateurs de jeux vidéo.

2 État de l'art.

2.1 Roguelike.

Les Rogue-likes sont un genre ancien datant des années 80. Le genre est passé par des nombreuses étapes d'évolution avant de devenir ce qu'il est actuellement.

Rogue est un jeu initialement sorti sur le système d'exploitation UNIX en 1980. Le jeu tient son inspiration des jeux d'aventure papier comme donjons et dragons, le jeu voit le personnage joueur se faire envoyer dans une mine pour récupérer une amulette. Ce qui sépare Rogue de ses inspirations c'est son système de *perma-death* (mort permanente) couplé à un système de génération procédural de niveau qui assure que le joueur aura une expérience différente à chaque run.

Rogue reste cependant limité en termes de mécanique système du a ses limitations techniques, la seule forme de progression se fait en augmentant ses stats soit en équipant de meilleurs objets ou en gagnant un niveau (tout de même impressionnant pour l'époque). Le jeu sera ressorti sur toutes sortes de micro-ordinateurs.

Le genre restera ensuite plus ou moins dormant pendant une longue période (malgré la curiosité occasionnelle comme Baroque sur PS1). Vint ensuite le renouveau du jeu indépendant pendant le début du 21e siècle porté par des noms créatifs comme Braid, Super Meat Boy ou Plants Vs Zombies ou les Exemples nous intéressant ici :

The Binding of Isaac et Spelunky.

The Binding of Isaac est Sorti en 2011 sur Steam ou il est très vite devenu populaire autant pour son univers macabre et fascinant que pour son gameplay novateur. Le jeu se déroule dans une maison ou une mère très catholique abuse violemment un enfant nommé Isaac en l'enfermant dans la cave, tout l'expérience de jeu est donc l'exploration de cette cave a travers la vision du monde troublé d'Isaac .

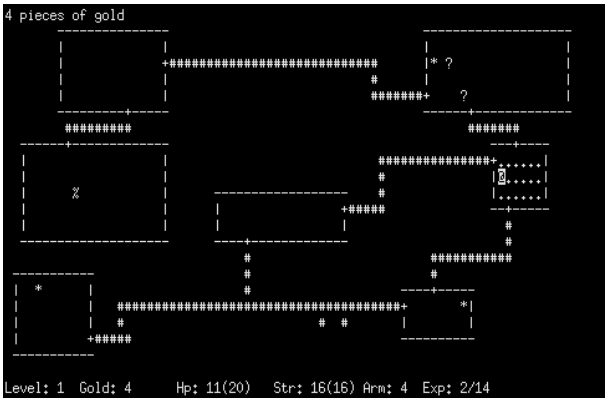

FIGURE 1 – Donjon généré procéduralement dans Rogue.

Tout le sel qui fait le succès d'Isaac repose dans la personnalisation qui existe dans le gameplay, en effet les niveaux ne changent pas vraiment. Le vrai triomphe du jeu est dans ses objets, en effet si au début d'une run Isaac se bat uniquement avec ses larmes (macabre j'ai dit) on peut très vite obtenir des *power-up* farfelus qui changent radicalement l'expérience de jeu allant des bombes aux rayons lasers en passant par les morceaux de verre.

FIGURE 2 – La sortie originale de TBOI (D'autres suivront).

Vint ensuite Splunk qui prit une route différente de The Binding of Isaac, au lieu de se concentrer sur les items et synergies (même si ceux-ci sont au rendez-vous) le level design est au centre de l'attention. Ici on joue un explorateur a la recherche d'un trésor au fond d'une mine qui change à chaque mort. Or le jeu est intéressant car le défi n'est pas mis sur le combat mais sur l'exploration de la mine et des dangers de celle-ci, Le joueur aura donc à se battre contre des pics et autres pièges. Sous le capot un des algorithmes du jeu est chargé de s'assurer que là la sortie du niveau est atteignable par le joueur sans faire sauter de bombe laissant toutes les parties inaccessibles pour les courageux en quête de richesse.
Il est intéressant de noter que ces 2 jeux appartiennent au genre du "Rogue-lite" plutôt que like, en effet pour compter comme un "vrai" Rogue-like le tour par tour est nécessaire.

Suite à la popularité des jeux mentionnées précédemment et de l'explosion du jeu vidéo indépendant une quantité immense de Roguelike firent leur apparition comme (liste non exhaus-

FIGURE 3 – La version classique de Spelunky dans toute sa splendeur 8-bit.

tive) : Rogue Legacy, Risk of Rain, Dead Cells, Enter the Gungeon, Hadès et Crypta of the Necrodancer. Ces jeux appartenant donc au sous genre du Rogue-lite. Toutefois certains restent fidèles aux racine tour par tour du genre comme les 2 jeux nous intéressant ici : Slay the Spire et Inscryption.

Dans Slay the Spire on joue un personnage envoyé tuer le mal au sommet d'une tour (2018). Au cours de cette ascension le joueur est soumis à des évènement choisis aléatoirement (par exemple un combat ou un coffre). La progression ici n'est pas réellement faite a partir d'objets mais de cartes (on peut en trouver dans des coffres) ces cartes peuvent faire toutes sortes de choses comme simplement attaquer ou s'améliorer.

Tout l'intérêt de ce système repose dans la synergie qui peut être créer au sein d'un deck comme dans un jeu de carte traditionnel. Mais couplé avec le hasard et les différents personnages jouables chacun avec leurs statistiques chaque run est quasiment unique.

FIGURE 4 – Une combat normal de Slay The Spire.

Inscryption est un des exemples les plus récent du genre, il est sorti en 2021. Dans celui-ci on y joue un personnage non nommé enfermé dans une cabine avec une vieil homme nous proposant de jouer à un jeu de cartes.

Chaque tentative est donc une nouvelle partie contre le vieillard. Si Inscryption partage un système de Deck avec Slay the Spire la manière d'utiliser ses cartes est radicalement différente. Ici à la place des cartes servant comme éventail des capacités du personnage, chaque carte

fonctionne plutôt comme une créature qui va attaquer indépendamment du joueur à la fin de chaque tour transformant le joueur en une sorte d'invocateur stratège plutôt que de guerrier.
Des nouvelles cartes sont donc obtenues et peuvent être modifiées pour les rendre plus puissantes, assurant que le joueur a la liberté de choisir comment construire son deck pour se préparer au défi l'attendant à la fin de la partie.
Notre jeu est donc un jeu de carte qui vise à s'inscrire dans la longue ligné des rogue-like.

FIGURE 5 – Un duel dans Inscryption.

2.2 Gameplay.

Le gameplay de notre jeu peut paraître simple mais cache en réalité un système complexe de décision et de choix telle la gestion des sortilèges, le choix de classe ou encore du système de ressource. Pour commencer, après avoir choisie votre classe parmi les 5 disponibles (pour le moment), vous débutez votre aventure et commencez à combattre les monstres que renfermes la tour.
Vous êtes un aventurier d'une certaine classe (au choix au début du jeu), vous possédez 2 choses importantes votre grimoire et votre sac de ressource. Votre grimoire renferme des sortilèges, sort qui peuvent invoquer des éclaires ou réanimer les morts, chaque sort possède un certain besoin en ressource. Vous devez donc vous débrouiller pour que votre deck ressource concorde avec les coûts de votre grimoire.

FIGURE 6 – L'écran de création de personnage.

Au début du partie votre classe vous alloue des sorts et entités de départ. Vous possédez aussi votre deck de ressource qui contient tout le nécessaire pour lancer les sorts de votre grimoire.

Chaque étage de la tour correspond avec une succession de rencontres dont la plus part vont être des combats.

Chaque niveau est un combat, il se déroule de la manière suivante :

- Vous commencez le combat, avec les ressources de votre sac de ressource et les sorts de votre grimoire
- Selon vos ressources vous invoquez vos entités grâce à vos sorts qui combattront contre les créatures du niveau.
- C'est maintenant aux créatures de vous attaquer et ainsi de suite jusque ce que vous n'ayez plus de vie ou que le camp adverse n'est plus de créatures.
- Si vous avez gagné le combat, vous devez faire un choix parmi des nouveaux sorts ou invocations qui vous sont proposées, pour renforcer votre partie.

FIGURE 7 – Un écran de combat (susceptible de changer).

2.3 Multijoueur.

2.3.1 Vu d'ensemble.

Notre jeu utilise un multijoueur de type peer-to-peer, c'est-à-dire qu'un joueur est désigner l'hôte : il est à la fois client et serveur, les autres joueurs qui le rejoigne sont alors simple client. Ce qui nous enlève le besoin d'avoir à trouver et configurer un serveur distant. Pour faire cela, nous utilisons la library **Mirror** qui se repose sur l'ancien *Netcode* d'Unity afin de sucrer quelques étapes et inconveniences du networking.

Notre jeu est un jeu coopératif qui se joue à un maximum de 2 joueurs :

- Les deux joueurs choisissent leur classe, ressources et sorts en même temps et se concertant.
- Durant le combat ils joueront un tour sur deux, et devront alors planifier le fait que leurs sorts n'utilise probablement pas les mêmes ressources.
- Leur inventaire et ce qu'ils trouveront dans La Tour est partagé.

2.3.2 Vu technique.

Nous prenons avantage de la classe `NetworkManager` donné par Mirror qui propose de nombreux callback pour le serveur et pour le client afin de réagir adéquatement aux actions des joueurs.

NetworkRoomManager.cs exemple de callback

```
1  public class NetworkRoomManager : NetworkManager
2  {
3    // Appele quand un client se connecte au serveur (ici l'hote)
4    public override void OnRoomServerConnect(NetworkConnectionToClient conn)
5    {
6        Debug.Log("New client connected.");
7    }
8
9    // Appele quand tous les joueurs sont prets
10   public override void OnRoomServerPlayersReady()
11   {
12       //La partie commence une fois le compte a rebours termine.
13       StartCoroutine(StartCountdown(() =>
14       {
15           base.OnRoomServerPlayersReady();
16       }));
17   }
18 }
```

Ici nous utilisons le `NetworkRoomManager` qui hérite du `NetworkManager` et qui s'occupe du lobby où les joueurs se connectent pour commencer une partie.

Le plus pratique avec Mirror c'est de pouvoir utiliser les propriétés comme `isClient` ou bien `isServeur` ou encore `isLocalPlayer` en héritant de la classe `NetworkBehaviour` pour pouvoir implémenter différents comportements aux `GameObject` selon s'ils sont propriétaire de l'objet ou non.

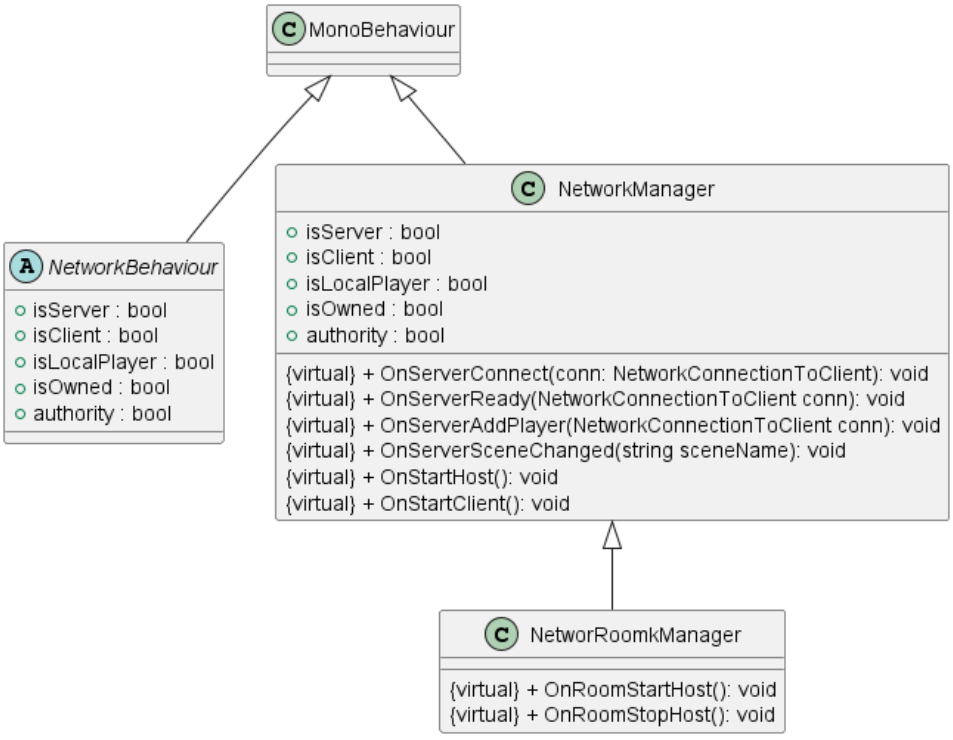

FIGURE 8 – Schéma des classes de base de Mirror (non-exhaustif).

Enfin la philosophie de Mirror repose sur les `[Command]` et `[ClientRPC]` qui permettent d'exécuter des méthodes depuis les client vers l'hôte et depuis l'hôte vers les clients respectivement.

PlayerMirror.cs exemple de Command

```
1   [Command]
2   private void CmdChangePlayerName(string newName)
3   {
4       PlayerNameText = newName;
5   }
```

Ici donc une méthode invoquée depuis un client pour demander à l'hôte de mettre à jour son pseudonyme. L'hôte pourra à son tour mettre à jour chaque autre client avec une RPC.

2.4 Direction artistique.

Notre direction artistique s'inspire de Inscryption et de son désigne old school qui peut faire pensez à du parchemin ou a de vielles images.
Toute notre palette graphique est centrée autour d'une composition monochromatique de 2 fois 6 couleurs. la première avec des teintes grises pour donner du corps, et la deuxième, en teintes rouges pour accentuer certains détails.

FIGURE 9 – La palette définitive du jeu.

Ce choix ce justifie par notre envie de transmettre au joueur un environnement et une ambiance a la fois morbide, morne et froid de notre monde. Le jeu entité suit donc cette palette monochromatique. Durant la conception nous avons grandement hésite a ce choix à double tranchant et clivant, en effet il est difficile de rendre des graphismes digestes et compréhensible par l'œil humain mais aussi cela peut aussi amener le joueur a ce lasser visuellement du jeu et de sa direction artistique avec une palette monochromatique.

FIGURE 10 – Un exemple de sprite créé avec la palette.

3 Avancés du projet.

3.1 Comparaison avec le cahier des charges techniques.

Tâches	Pourcentage actuelle	Pourcentage visée
Brainstorm	100%	100%
Design des items (Gameplay)	100%	100%
Création de l'histoire	100%	100%
Scriptables objects	100%	100%
Architecture des classes	95%	100%
Event Manager	100%	100%
Création des assets graphiques	85%	100%
Création de l'UI	90%	100%
Système de sauvegarde	10%	50%
Sound Design	25%	100%
Génération procédurale	20%	100%
Game Loop	60%	80%
Mise en place de Mirror	100%	100%
Création du coopératif	40%	80%
Effets spéciaux	10%	0%
Post Process	30%	0%
Équilibrage	0%	0%

Comme on peux le voir sur le tableau les buts visés n'ont pas tous étés atteints et d'autres au contraire excédés.
Une majorité de nos efforts on étés centrés sur la mise en place de systèmes sur lesquels tout le jeu dépend et sans qui il est impossible que le jeu fonctionne (comme Mirror ou l'Event Manager), étant donné que rien ne peux réellement fonctionner sans eux une majorité du temps a été passé a s'assurer de leur bon fonctionnement pour s'assurer que l'implémentation du reste du code se passerait sans encombre.
Le revers de la médaille ici étant que d'autres parties que nous avions supposés comme implémentable avant la soutenance ont du être délaissées (par exemple le système de sauvegarde ou la génération procédurale). Laissant donc un variété d'idées plus bas que le pourcentage attendu.

3.2 Rapport du travail des membres.

- **Arthur Mallet** : J'ai commencé à travailler sur le projet en novembre, ayant déjà utiliser Unity et le C# auparavant, j'ai pu concevoir l'architecture du code du jeu, les classes des objets utilisé dans le jeu ainsi que le multijoueur. Je m'occupe donc principalement d'implémenter les mécaniques du jeu et de créer une base solide pour pouvoir avancer rapidement sur la création des effets de cartes sur le long terme.

- **Victor Blanchot** : En début d'année mon rapport avec l'informatique a été assez compliqué et j'ai eu beaucoup de mal avec les TP. Après avoir constaté mon manque de connaissance en informatique je me suis chargé de tout ce qui, dans mes tâches, ne demandait pas de compétences informatiques. J'ai donc commencé à faire toute la partie visuelle et la D.A. Après avoir fait tout le plateau, les cartes et les spirites de monstres, sorts et ressources, j'ai en parallèle commencer à m'améliorer en programmation. Une fois mon niveau suffisant j'ai pu commencer à comprendre les bases de codes écrites par Arthur. Je me suis donc mis à partciper à l'écriture des DéfinitionPools et les prefabs de entités et sortilèges, finalement mes compétences en design et dessin mon permis d'être à l'aise sur l'UI du jeu.

- **Uriel Fellmann** : En début d'année me s capacités informatiques n'étaient malheureusement pas assez développées pour réellement avoir un impact, Mon travail lors du commencement du projet se limitait donc plutôt a la rédaction de l'histoire et le brainstorming des idées de gameplay et des ennemis. J'ai donc eu un rôle plutôt fondamental pour la mise en place de l'univers et la création des idées du jeu ainsi que des classes des joueurs qui sont au cœur de La Maison-Dieu. Des responsabilités ne nécessitant pas réellement de code donc. Ensuite une fois mes compétences en code suffisamment développées au fil de l'année j'ai pu commencer a ajouter a la base de code posée par Arthur en faisant par exemple le créateur de personnage.

4 Choix des techniques mise en place.

4.1 Logiciels externes.

Obsidian, Trello, GitHub, Piskel Concernant la gestion du projet, nous nous sommes servis de nombreux logiciel nous permettant d'être plus performant dans notre approche, structuration, répartition du projet.
Pour commencer, toute notre base de données, autrement dit, nos recherche, l'histoire de notre jeu, toutes les créatures sorts et entités, ou encore tous ce qui concerne la structure de notre jeu. . .

En bref, toute information en relation avec notre jeu est stockée sur un logiciel nomme Obsidienne. Ce logiciel polyvalent de prise de notes et de gestion de connaissances, basée sur des fichiers Markdown. Sa principale utilité réside dans son système unique de liens entre les notes, qui permet aux utilisateurs de créer, organiser et naviguer leur base de connaissances de manière visuelle et intuitive. En résume il nous permet d'avoir une vue d'ensemble et de créer un une toile de lien entre toutes nos notes.

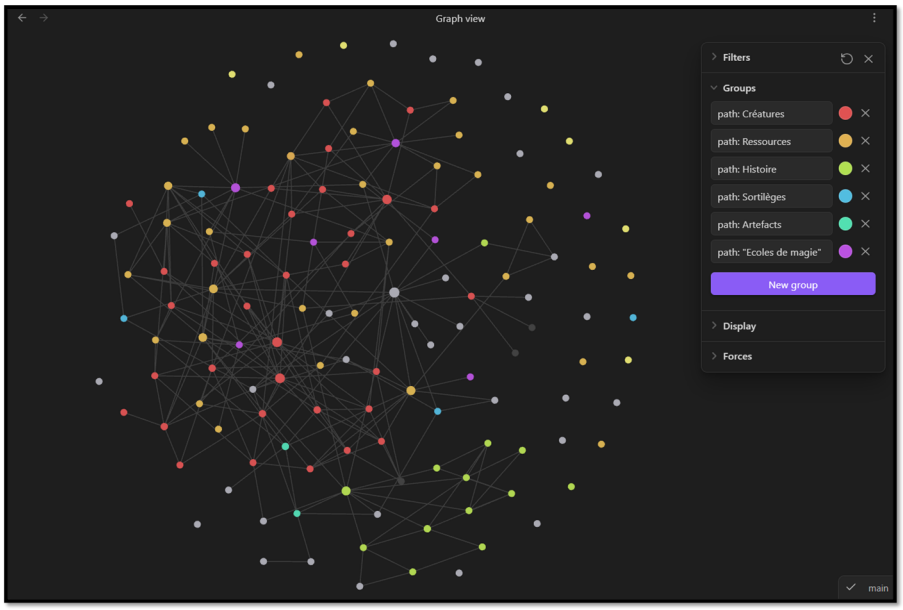

FIGURE 11 – Notre graph Obsidian.

Cela a permis à chacun de nous d'avoir en permanence et à tout moment toutes les informations nécessaires pour bosser de notre côté, sans devoir en permanence demande les fichiers et informations importantes (ce qui peut être lourd).

Mais nous avions aussi besoins de logiciel pour s'organiser. Pour bien s'organiser rien de mieux que Trello qui est un logiciel d'organisation entre membre. Trello est un outil de gestion de projets basé sur le cloud, conçu pour faciliter l'organisation et la collaboration au sein des équipes. Il emploie une interface intuitive composée de tableaux, listes, et cartes pour permettre aux utilisateurs de visualiser et de suivre la progression des tâches de manière dynamique.
Les fonctionnalités clés incluent la gestion de tâches avec des descriptions détaillées, des check-

lists, des échéances, et des pièces jointes, ainsi que la personnalisation via des étiquettes pour catégoriser les tâches par compétence ou domaine. Trello favorise la collaboration en permettant l'assignation de tâches, les commentaires, et les notifications, assurant que tous les membres de l'équipe restent informés.

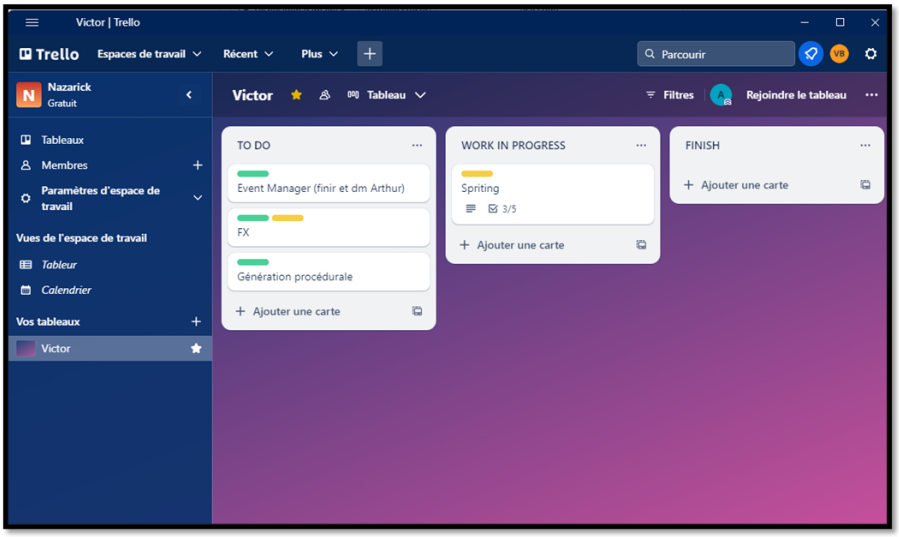

FIGURE 12 – Tableau Trello de Victor.

Pour créer tous nos sprites nous avons utilises Piskel art. Piskel Art est un éditeur de pixel art en ligne gratuit et convivial, destiné à la création d'animations et de graphiques en pixel art. Conçu pour être accessible aux artistes de tous niveaux, des débutants aux professionnels, Piskel offre une interface intuitive qui facilite la création de dessins pixel par pixel.

FIGURE 13 – Piskel.

Les fonctionnalités clés incluent divers outils de dessin comme le pinceau, le pot de peinture, la palette de couleurs, et plus encore. Piskel supporte l'exportation des créations en différents formats, y compris GIF, PNG, et même des feuilles de sprite, rendant cet outil particulièrement utile pour les développeurs de jeux, les artistes numériques, et les amateurs de retro gaming désirant créer ou éditer des pixel arts et des animations simples mais expressives.

4.2 Architecture du code.

4.2.1 Les permanents.

Dans notre jeu, il existe 5 type de *permanent* (terme emprunté à Magic The Gathering) :

- Le joueur : `Player.cs` (ou les joueurs si le jeu est lancer à 2 joueurs).
- Les sortilèges : `Spell.cs`, qui représente tous les sorts du jeu.
- Les entités : `Entity.cs`, les créatures qui peuplent le champ de bataille, invoqué par le jeu ou par le grimoire.
- Les ressources : `SpellResource.cs`, les ressources utilisé pour lancer les sortilèges.
- Les rituels : `Ritual.cs`, représente les effets de cartes qui reste un certains nombre de tour en jeu.

4.2.2 ScriptableObjects.

Tous les types de permanent du jeu (les sorts, cartes de monstres et objets) sont stockés dans des assets `ScriptableObject` et associé à une ID unique. Faire ceci nous donne une base solide qui rend chaque création et manipulation de permanent beaucoup plus simple car nous pouvons facilement retrouver un permanent grâce à son ID mais aussi le synchroniser sur le réseau juste avec l'ID.

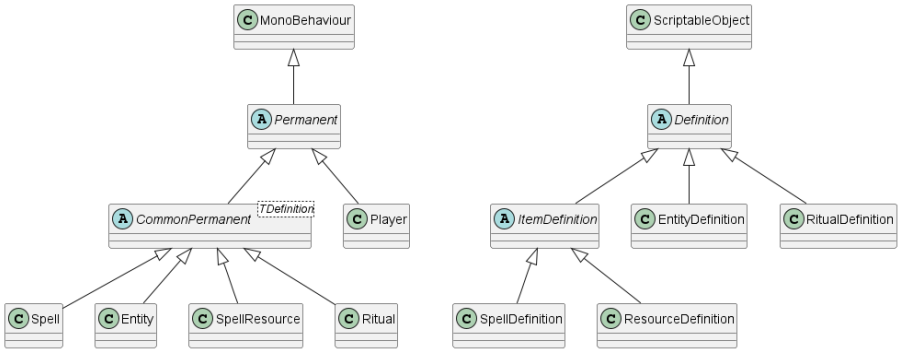

FIGURE 14 – L'architecture de base des classes items.

5 Obstacles rencontrés et solutions trouvées.

Durant toute la durée du projet, nous avons rencontré de nombreux obstacles.
Ces derniers nous ont ralenti dans notre progression. Pour commencer, le niveau hétérogène de notre groupe a créé de nombreux problèmes. Maxime et Victor ont dû attendre le B3 en programmation pour comprendre et pour commencer à écrire et implémenter le jeu. De plus, la synchronisation entre clients a posé problème dû à notre incompréhension de son fonctionnement. Nous l'avons réglé quand nous avons compris les commandes et les RPC.