MCR - Projet Médiateur

Introduction

Notre projet consiste en la création d'un jeu RPG (role playing game) en 2D, dénommé "Dream Team".

Le but de ce jeu est de parcourir trois mondes différents, représentant des niveaux, avant d'affronter le boss final.

Pour cela, le joueur pourra prendre le contrôle d'un des trois personnages, qui font partie de la "Dream Team" que sont : le **guerrier**, le **mage** ou l'**archer**. Mais pour parvenir à cette dernière étape, il devra affronter tous les monstres qui se dresseront sur son chemin. Pour cela, il aura à disposition des armes ou des sorts qu'il acquerra au fil du jeu pour les vaincre.

Quelques explications concernant le framework d'Unity

Comme on peut le constater, la grande majorité des classes de notre projet sont dépourvues de constructeur. C'est du fait que la plupart des classes héritent de la classe MonoBehaviour qui est issue de framework Unity. Il n'est pas permis de construire des objets héritant de cette classe.

Changement de scènes, niveaux

Chaque scène, donc niveau, possède son propre médiateur, les médiateurs des niveaux ne communiquent pas entre eux. À la fin d'un niveau, le médiateur va appeler une classe propre à Unity, PlayerPrefs, pour stocker deux variables, concernant le niveau du personnage et sa classe. Ces deux variables stockées sont les seules informations qui peuvent transiter entre les scènes. À la fin d'une scène, tous ces objets et scripts sont détruits pour faire place à ceux de la scène suivante.

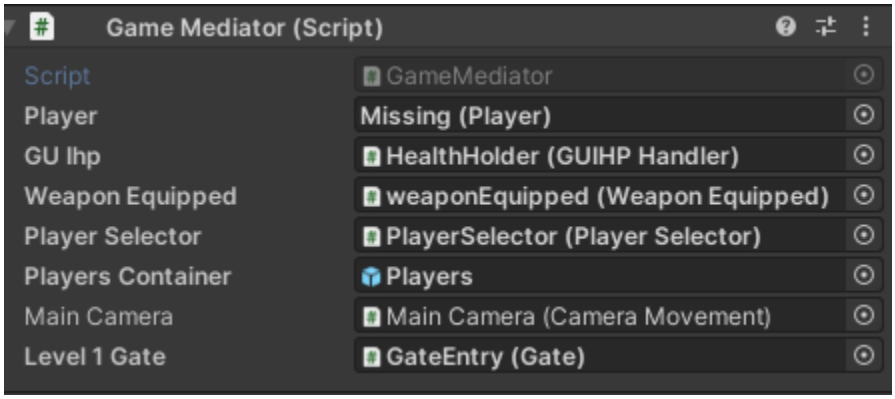
Classe MonoBehaviour

C'est la classe dont tous les games objects doivent hériter. Elle offre des méthodes telles que:

- Start - appelée au démarrage d'une scène une fois
- Awake - appelée quand une instance de gameobject est créée : une sorte de constructeur
- Update - appelée une fois par frame, permet de gérer la logique pour chaque game object frame par frame.
- ….

Champs [SerializeField]

Certains attributs des classes sont marqués par ce tag. Ce tag donne accès à Unity à des attributs des game objects.




```
[SerializeField] public Player player;
[SerializeField] private GUIHPHandler GUIhp;
[SerializeField] private WeaponEquipped weaponEquipped;
[SerializeField] private PlayerSelector PlayerSelector;
[SerializeField] private GameObject PlayersContainer;
[SerializeField] private CameraMovement MainCamera;
[SerializeField] private Gate level1Gate;
```

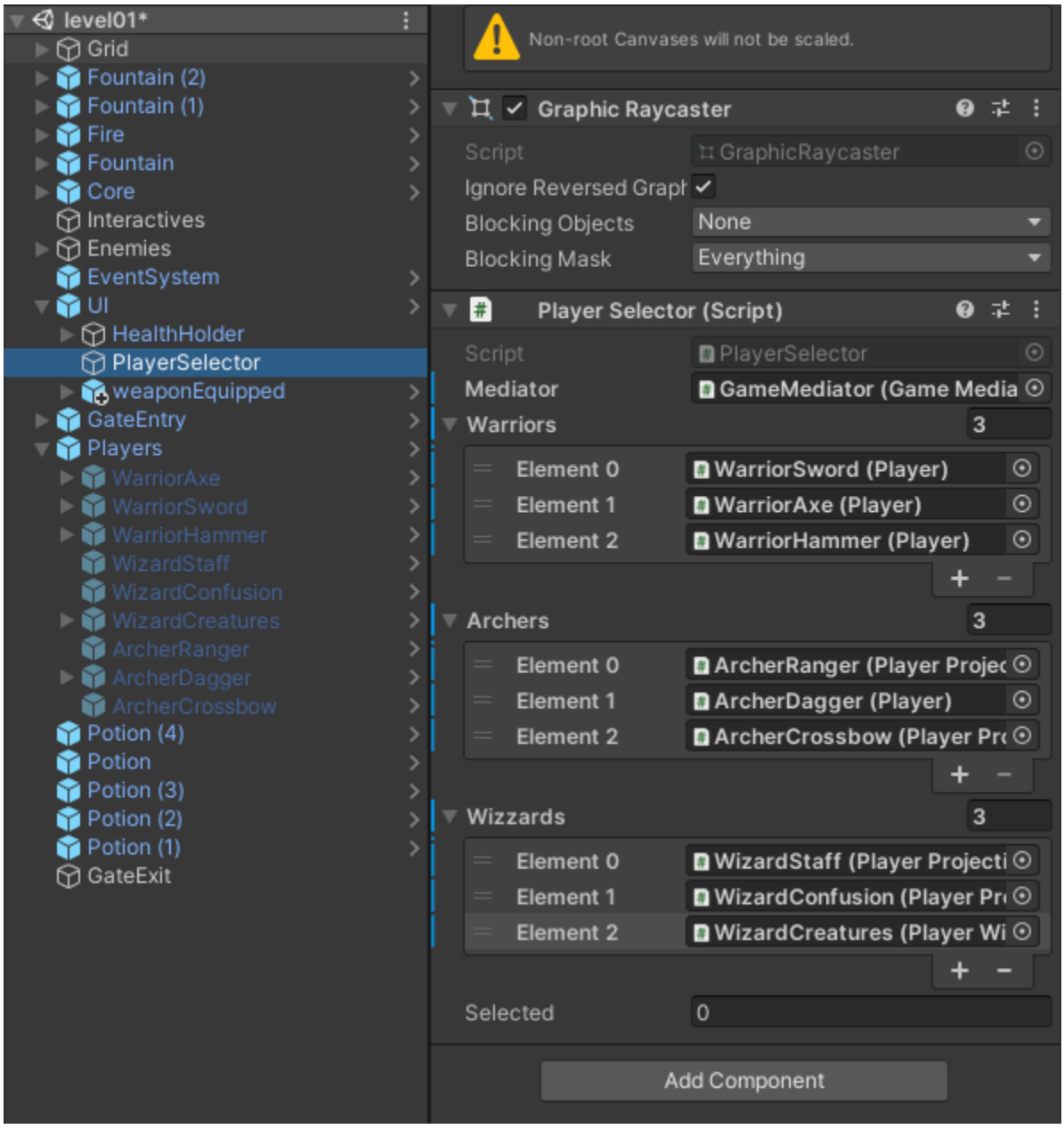
Ceci permet, entre autres, de faire des liaisons entre les différents game objects.
À l'instanciation de Game Mediator, la méthode Awake sera appelé par Unity pour faire des traitements dynamiques nécessaires

```
Message Unity | 0 références
private void Awake()
{
    command.Subscribe<HitBreakeableCommand>(OnHitBreakable);
    command.Subscribe<HpIncreaseCommand>(OnHpIncrease);
    command.Subscribe<HpDecreaseCommand>(OnHpDecrease);
    command.Subscribe<KnockbackCommand>(OnKnockback);

    PlayerLevel = (Level)PlayerPrefs.GetInt(selectedCharacterLevel, 0);
    PlayerClass = (CharacterClass)PlayerPrefs.GetInt(selectedCharacterDataName, 0);

    if (player == null)
    {
        enablePlayerSelector();
        weaponEquipped.changeText(PlayerSelector.PlayerWeaponName());
    }
    else
    {
        MainCamera.SetTarget(player.transform);
        GUIhp.gameObject.SetActive(true);
        PlayerSelector.gameObject.SetActive(false);
    }
}
```

Ces 2 approches, champs sérialisés et les méthodes mono behaviour, permettent d'instancier les games objects sans utiliser des constructeurs.
Voici un exemple de game object PlayerSelector (à gauche) et comment on donne des références à divers autres objets nécessaires à son fonctionnement (à droite / inspecteur / script PlayerSelector.cs) :



Dans le code, les attribut de PlayerSelector.cs sont :

```
public class PlayerSelector : MonoBehaviour
{
    [SerializeField] private GameMediator mediator;
    [SerializeField] private Player[] warriors;
    [SerializeField] private Player[] archers;
    [SerializeField] private Player[] wizzards;

    [SerializeField] private int selected;
```

Les Coroutines

Certaines méthodes des classes, celles dont le nom finit par Co, sont les coroutines d'Unity. Il s'agit des méthodes dont on veut exécuter certaines parties du code à des moments différents. Elle retourne un IEnumerator.

```
virtual protected IEnumerator AttackCo()
{
    if (CharacterState != CharacterState.Dead)
    {

        CharacterState = CharacterState.Attack;
        animator.SetBool("attacking", true);
        yield return new WaitForSeconds(attackDuration); // attack duration wait
        animator.SetBool("attacking", false);
        yield return new WaitForSeconds(attackCoolDown);    // attack cool down wait
        // the character could have be killed during his attack cooldown (very likely)
        // we must not change his state to walk if he is dead, thus the codition must be rechecked
        if (CharacterState != CharacterState.Dead) CharacterState = CharacterState.Walk;
    }
}
```

Il s'agit de la coroutine de l'attaque de la classe mère Character. La routine est appelée avec la méthode StartCoroutine. Quand elle s'exécute, elle va s'arrêter au premier "yield return", le programme va alors poursuivre son exécution ailleurs, pour revenir dans la coroutine après le temps indiqué dans WaitForSeconds.

Mise en oeuvre du pattern Médiateur

Le pattern médiateur est utilisé pour faire interagir différentes entités entre elles, il s'occupe également de définir le comportement des différentes entités.

L'état actuel du jeu, les ennemis, les alliés, les sentiers sont entièrement gérés par le médiateur. Il possède des informations sur les différents personnages et en décide l'action de chaque personnage en temps réel.

EnemyBehaviour et AllyBehaviour
Elles gèrent le comportement des ennemis et alliés, tous les personnages non player, sur la map. Elles vont définir leur état actuel et décider qui ils vont attaquer en se basant sur l'état actuel du jeu.

Pour y parvenir, il maintient une liste des ennemis et des alliés vivants à jour.

Le comportement ennemi est décomposé comme suit :

- Le player est hors du champ de vision : la sentinelle reste à sa place, toute autre ennemi se déplace en random selon les périodes définies
- Le player est dans le champ de vision, mais pas dans la distance d'attaque : l'ennemi poursuit le player avec la vitesse de poursuite définie.
- Le player est dans la distance d'attaque : elle attaque le player.

Les alliés sont des mobs spawnés par le mage ou ceux atteints par le sort de confusion. Le comportement allié est décomposé comme suit :

- Aucun ennemi est dans le champ de vision : suivre le player
- Un ennemi est dans le champ de vision : poursuite
- Un ennemi est dans le champ d'attaque : attaquer.

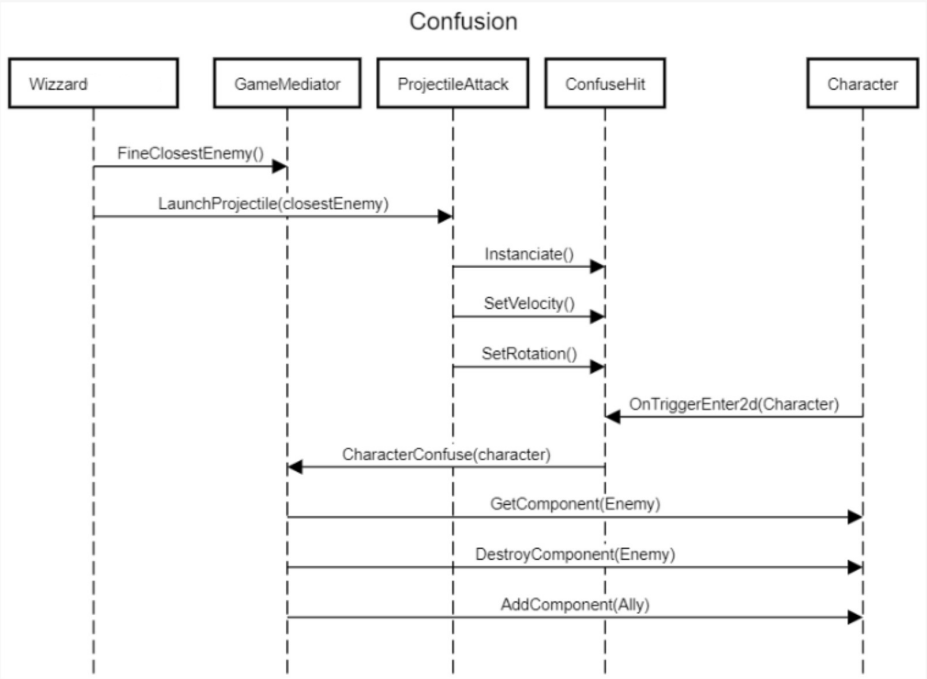
Le médiator possède des méthodes permettant de calculer les distances, les vecteur de directions et des angles. Le médiator définit la méthode permettant de retrouver les entités par rapport à leur position.

Il dirige les parties suivants :

- Le GUI affichant les HP du player
- L'arme active du player, le changement d'armes
- Décide de qui doit suivre la caméra
- Gère les transitions des niveaux, conditions et passages de niveau.
- Affichage des message : GUI
- Enregistrement des hits de différentes entités.
- La confusion d'une entité ennemie : Un ennemi est transformé en allié et placé dans la liste des alliés.
- Écoute des commandes : Hit sur objet cassables, augmentation et diminution de hp -> fontaine ou feux, knockback.

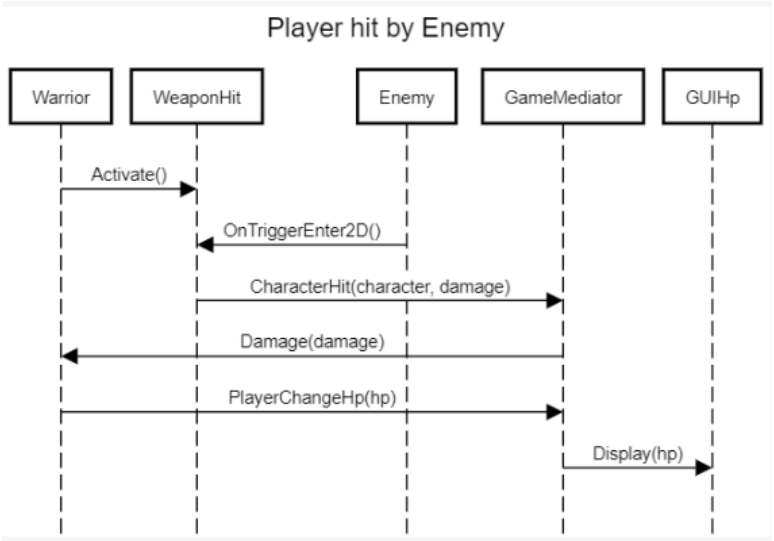
Quelques Interactions

- **Le sort “Confusion”:**



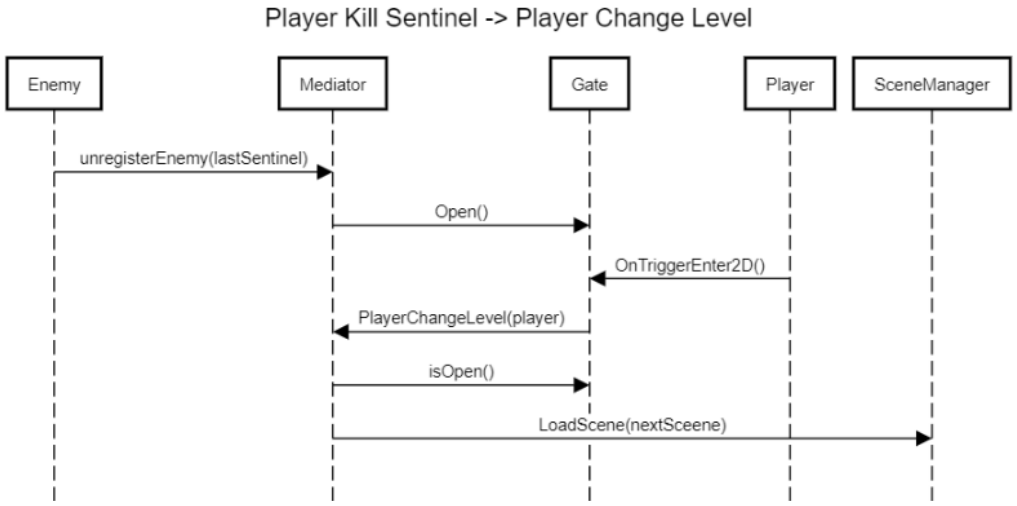
Le mage va demander au médiateur où se trouve l’ennemi le plus proche. Avec cette nouvelle information, il va lancer son attaque sous forme de projectile, un collider qui va toucher un ennemi “attaquable”. Quand cet ennemi est touché par le projectile, il va alors informer le médiateur de son changement d’état qui va se charger de détruire la forme “ennemi” pour la transformer en “allié” du joueur.

- **Les attaques des ennemis sur le joueur :**



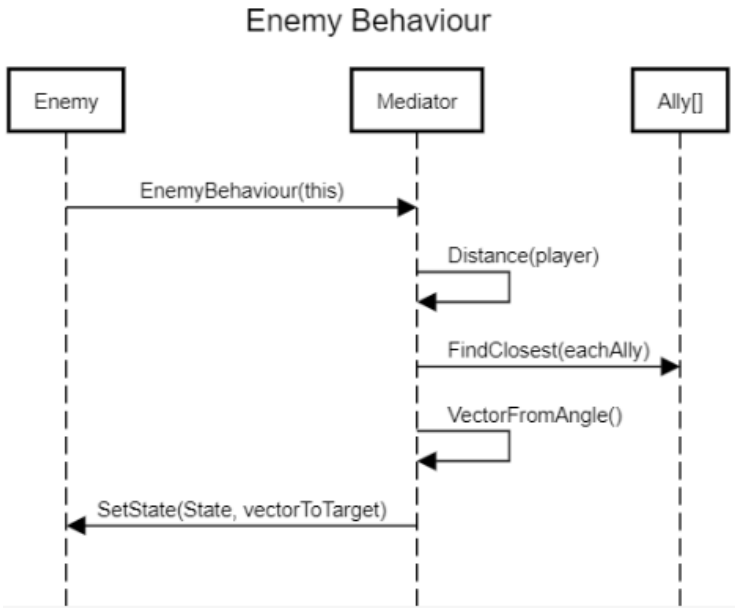
Si le guerrier, ou n'importe quel personnage du joueur, se trouve dans la zone d'attaque d'un ennemi, le Weapon Hit va informer le médiateur que le joueur a subi des dégâts, renvoyer au joueur le nombre de dégâts en HP puis modifier le nombre de point de vie avant de l'afficher dans le GUI.

- **Les attaques des boss, dit "Sentinel" pour l'ouverture des portes vers le niveau supérieur :**



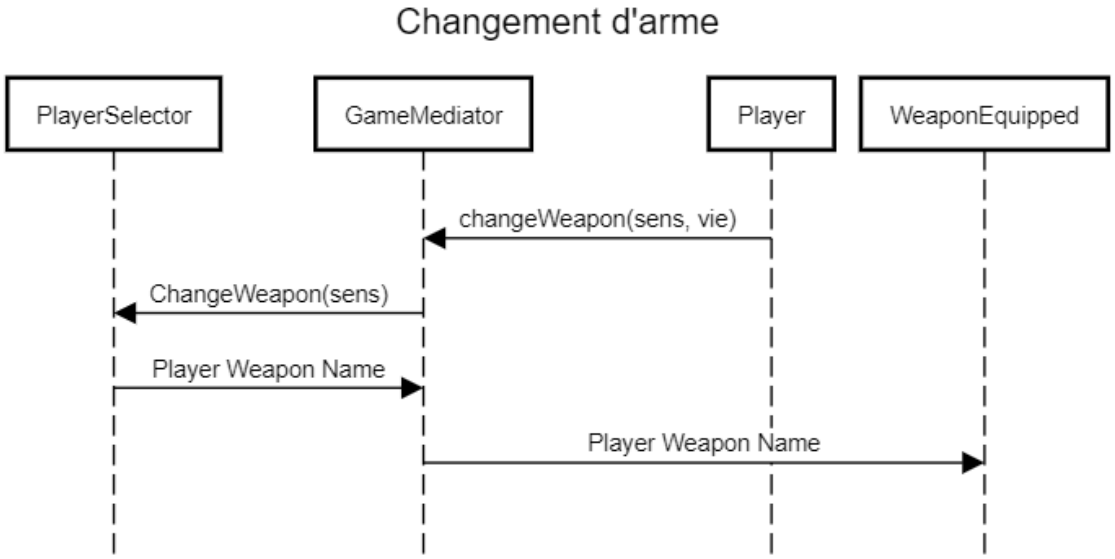
Une fois que le joueur a tué un ennemi déclaré comme "sentinel" ou un boss de fin de niveau, celui-ci va envoyer au médiateur son "désabonnement". Le médiateur, recevant cette information, va alors ouvrir lui-même la porte vers le niveau suivant.
Pour ce qui est du joueur, il va informer au médiateur au travers de "gate" qu'il veut passer au niveau suivant et le médiateur va charger le niveau suivant pour ce joueur.

- **Le comportement des ennemis :**



La classe ennemie demande au médiateur quel comportement adopter avec elle-même comme attribut. Le médiateur va récupérer la position du player ainsi que celle des entités alliés, puis récupérer celui qui est le plus proche de l'ennemi. Suivant si l'ennemi est suffisamment proche ou pas de l'entité choisie, le médiateur peut alors donner le comportement de prendre en chasse l'entité avec le vecteur contenant la direction qu'il doit prendre, d'attaque si suffisamment proche, de retourner à sa position d'origine si l'ennemi est une sentry, de marcher aléatoirement si l'entité la plus proche est hors d'atteinte ou de prendre aucune action si il s'agit d'une sentry.

- **Le changement d'armes :**



Lorsque le player va appuyer sur une des touches pour changer d'armes, il va appeler le médiateur. Pour mener à bien cette opération, le médiateur va devoir enregistrer la position actuelle du player ainsi que son nombre de points de vie, le player lui envoie ses points de vie et le choix de l'arme suivante ou précédente. Puis le médiateur va devoir appeler le PlayerSelector pour lui demander si possible de désactiver le personnage actuel pour activer celui avec l'arme souhaitée. À la suite de cette opération le PlayerSelector va envoyer une string contenant le nom de l'arme nouvellement équipée ou actuelle si le PlayerSelector n'a pas pu changer d'arme. Le médiator va pouvoir ainsi transporter le personnage activé à l'ancien emplacement, celui où le joueur a fait la demande de changer d'arme, et lui attribuer les points de vie qu'il avait avant l'opération. Pour terminer le PlayerSelector va envoyer l'information du nom de l'arme à la classe WeaponEquipped, cette classe affiche le nom de l'arme sur l'interface de l'utilisateur.

Chargement d'un niveau

Le médiateur est la seul classe à posséder une méthode awake, il agit ainsi en premier.

- Il commence par se mettre à l'écoute de certains types de commandes, tel un observateur.
- Il récupère la classe et le niveau du personnage.
- Dans le cas où il n'y pas de personnage pré-sélectionné, le médiateur va appeler le playerSelector pour déterminer quel personnage avec quelle arme activer.

UML

Voir le fichier .sly en annexe

Remarques

Nous ne garantissons pas l'équilibrage des classes des personnages.
L'interface utilisateur et les menus ont une mise à l'échelle selon la taille de l'écran, ainsi il est possible que certains éléments soient décalés ou pas à une bonne échelle.

Installation

Le projet est disponible sur github à l'adresse :
https://github.com/AliceThunderWind/MCR-Project-RPG

Exécution du jeu

Afin de lancer le jeu, il suffit simplement d'exécuter RPG.exe dans le dossier /Build

Sources

Le projet Unity (sources) est dans le dossier /RPG:

Afin de pouvoir lancer le projet, il faut installer Unity :
https://store.unity.com/download?ref=personal en version 2020.3.11f1

Nous recommandons d'utiliser Visual Studio pour sa bonne intégration avec Unity.

Les différents scripts se trouvent dans le dossier /RPG/Assets/Scripts

Manuel d’utilisation

Commandes

Après avoir lancé l’exécutable, il faudra cliquer sur le bouton “Play” puis choisir entre un des 3 personnages “warrior”, “ranger” ou “wizard”.

Une fois le personnage choisi, il faudra utiliser les touches ‘W’, ‘A’, ‘S’, ‘D’ pour pouvoir se déplacer dans le monde. Pour attaquer, il faudra appuyer sur ‘ESPACE’ et enfin pour changer d’armes, il faudra utiliser les touches ‘Q’ et ‘E’ (possible dès le niveau 2).

Déroulement du jeu

On choisit la classe de personnage avec laquelle on souhaite jouer, on ne peut changer la classe qu'une fois choisie sans recommencer.

Pour pouvoir passer d’un monde à un autre, il faudra tuer le monstre à la fin de la map (Grand méchant Loup, Grand Squelette et Grand Minotaure) qui débloque la porte vers le niveau supérieur. A chaque nouveau niveau, le joueur débloque une nouvelle arme.

Des potions ou des fontaines se trouvent à des endroits du monde et permettent au joueur de gagner en points de vie (HP pour health point). En revanche, le feu, lui, en fera perdre.

Si le joueur n’a plus de points de vie, c’est-à-dire 0 sur les 100 de base, il échoue le niveau et le jeu lui permet de recommencer au niveau où il est mort. Il réussit le jeu lorsqu’il bat le minotaure avec un écran de crédit.

L’écran de game over permet soit de reprendre sa partie depuis le début du niveau, soit de revenir au menu principal, la dernière option étant de quitter le jeu.

Depuis l’écran des crédits, il est possible de retourner au menu principal ou de quitter le jeu.

Les classes de personnages

Nous avons défini 3 classes de personnages :

Le **guerrier** : il possède 3 armes que sont l'épée au niveau 1, la hache au niveau 2 et le marteau au niveau 3 :

L'épée touche les ennemis avec 40 points de dégâts

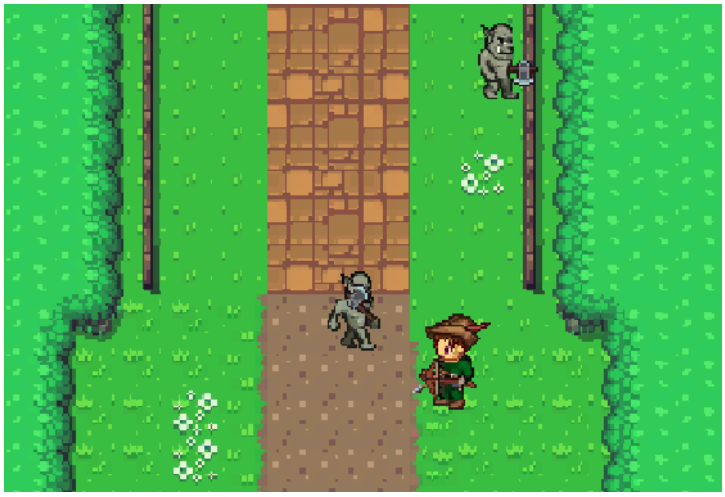
La hache touche les ennemis avec 75 points de dégâts

Le marteau touche les ennemis avec 100 points de dégâts

L'**archer** : il possède 3 armes que sont l'arc au niveau 1, la dague au niveau 2, et l'arbalète au niveau 3 :

L'arc tire une flèche sur un ennemi avec 35 de dégâts

La dague touche les ennemis avec 40 points de dégâts



L’arbalète tire trois flèches sur un ennemi de 35 points de dégâts

Le **mage**: il possède 1 arme et deux sorts que sont le bâton de feu au niveau 1, le sort de confusion au niveau 2 et l’invocation de gargouilles au niveau 3 :

Le bâton tire plusieurs boules de feu de 15 points de dégâts

La confusion est un sort qui va permettre à un ennemi d'attaquer un autre ennemi au lieu du joueur

Le sort d'invocation va permettre de faire apparaître 4 gargouilles qui vont toucher les ennemis avec 20 points de dégâts chacun. Ceux-ci vont disparaître au bout de 10 secondes.