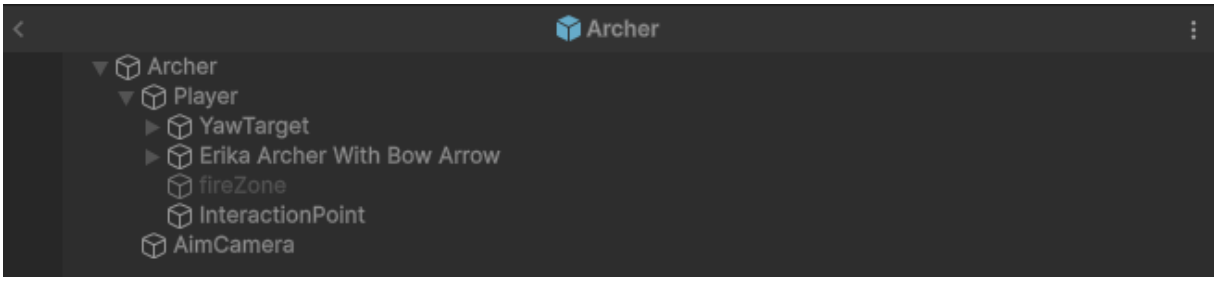
Personnage & Item

Ce document vise à décrire le processus qui permettra à n'importe quel contributeur d'ajouter un nouveau personnage au projet et également un nouvel « Item » *(objet à collecter)*.

Il considère que vous avez pris connaissance des documents d'introduction au projet tel que le README que vous retrouverez dans le repository GitHub qui décrit notamment comment contribuer au jeu. Le document considère également que vous ayez quelques notions de base comme les « prefab » ou « component » ou encore « asset ».

Personnage

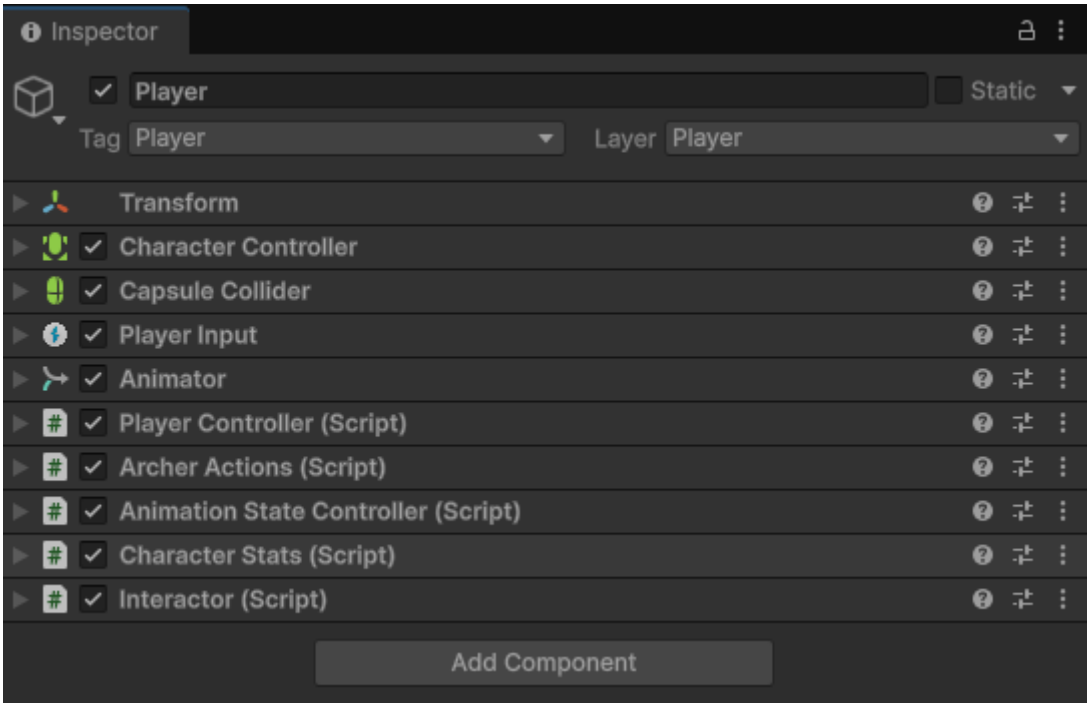
Un prefab est fournis vous pouvez le dupliquer *(CTRL + D)* afin de vous en servir comme base.



Il se compose de plusieurs éléments :

- Un objet **Player** qui contient tous les scripts du personnage, le « **Mesh** » ainsi que quelques composants utilisé pour les capacités du personnage et d'un InteractionPoint qui sert à la détection et l'achat des Items.
- Une **Cinemachine Camera** qui gère le comportement de la « Main Camera » sur la scène.

Assuré vous que l'objet **Player** est bien muni des composants et scripts suivant :

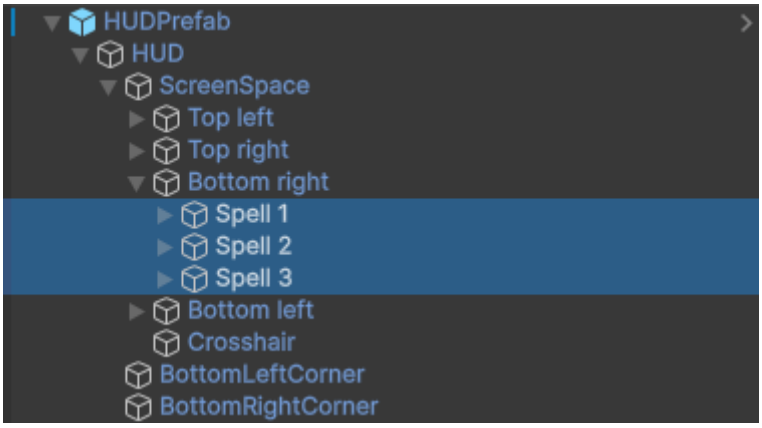


Le **Mesh** est l'Asset que vous mettez à votre personnage. Vous êtes libre d'y mettre l'asset que vous voulez, en l'occurrence on y a mis les assets qu'on a trouvé sur Mixamo *(ce qui facilite l'obtention d'animation également)*.

Maintenant que vous avez votre Player contenant l'asset de votre choix, il est temps pour vous d'ajouter des animations.

Créé votre *Create* > *Animation* > *Animator Controller* puis ajouter vos états et transitions. Ajoutez-y vos animations puis inspirer vous du script **Archer Action** et modifiez-le à votre guise.

En terme de HUD/UI il vous faudra trois images distinctes qu'il vous faudra ajouter en tant que RawImage au prefab HUDPrefab qui se trouve dans *UI/UI_HUD/HUDPrefab*.



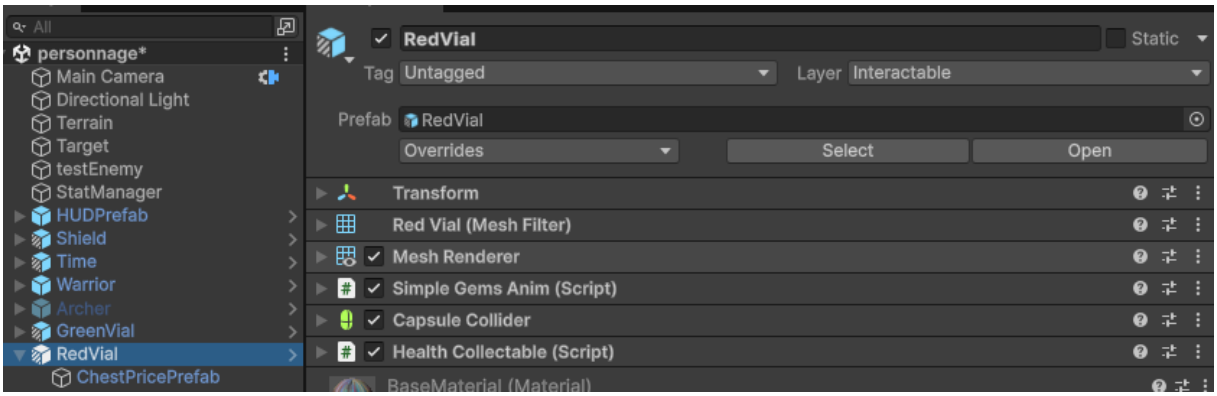
Attention à bien référencer les bonnes images lors du Start() du script **PersonnageActions** afin d'activer ces dernières afin de les afficher en jeu.

```
0 references
private void Start()
{
    animationState = GetComponent<AnimationStateController>();
    playerStats = GetComponent<CharacterStats>();

    // Activate UI for the Warrior class
    var allRawImages = FindObjectsOfType<RawImage>(true);
    foreach (var raw in allRawImages)
    {
        if (raw.name == "WarriorSpell1" && raw.transform.parent.name == "Spell 1")
            UISpell1 = raw;
        if (raw.name == "WarriorSpell2" && raw.transform.parent.name == "Spell 2")
            UISpell2 = raw;
        if (raw.name == "WarriorSpell3" && raw.transform.parent.name == "Spell 3")
            UISpell3 = raw;
    }

    if (UISpell1 != null)
        UISpell1.gameObject.SetActive(true);
    if (UISpell2 != null)
        UISpell2.gameObject.SetActive(true);
    if (UISpell3 != null)
        UISpell3.gameObject.SetActive(true);
}
```

Item

Quelques modèles d'items préexistants sont disponibles dans le dossier *Assets/Items/BTM_Items_Gems/Prefabs*. Ils ont été pris depuis le Unity Store par BenjaTheMaker.

RedVial, GreenVial, Shield et Time sont des items déjà implémentés. Inspirez-vous de ceux-là.



Il concevez un script de CaractéristiqueCollectable afin d'augmenter la caractéristique de votre choix de CharacterStats.