TM-création d’un jeu vidéo

# Mes motivations

Avant de commencer ce long TM, j’aimerais vous racontez une histoire, mon histoire et ce qui m’a conduit à faire ce long projet. Dès que j’en ai eu l’âge…enfin lorsque j’ai eu ma première année, mon grand frère m’a mis une manette dans les mains. A ce moment, j’ai pu tout de suite ressenti une sensation de joie en gagnant ou en perdant, une envie de recommencer, une émotion forte de défi et de bonheur, quelques choses que je ne peux même pas exprimer…sans avoir ma manette branchée à la console. Des années sont passées, je commence de plus en plus à m’intéresser au sujet, vouloir aller plus que de simplement y jouer et j’ai eu envie de connaitre l’envers du décor. Au même moment, mon frère pense la même et décide donc de se lancer dans des études de *game design[[1]](#footnote-1)*. Grâce à lui, j’ai pu en connaitre beaucoup plus mais également rencontrer d’autre étudiant en game art[[2]](#footnote-2), qui est ce dont je m’intéresse. Après avoir vu plusieurs de leurs projets et de nombreux projet de jeux vidéo indépendants, j’ai rêvé d’en créer un moi-même.

# Inspirations

Avant de commencer mon projet, j’ai regardé énormément de jeux différents, de différents types et styles afin d’être plus renseigner sur ce qui se fait. La majorité de mes inspirations sont des jeux d’indépendants[[3]](#footnote-3) car les créateurs sont souvent une équipe restreinte contrairement aux jeux triples A qui peuvent aller jusqu’à 1000 personnes pour un seul jeu. L’avantage d’une petite équipe permet d’avoir, aux artistes, un univers artistique original et plus poussé. Voici mes principales inspirations :

« Don’t Starve Together » développer par *Klei Entertainment*, « Hollow Knight » par Team Cherry, « Limbo » par PlayDead, « Journey » par Jenova Chen, « Pinstripe » et « One Upon a Coma » par Atmos Games.

# Logiciels et matériels utilisés

Photoshop : Il s’agit d’un logiciel de la suite Adobe que je vais utiliser pour tout ce qui graphique dans le jeu pour les personnages et le décor.

Unity : C’est un moteur de jeu très populaire dans le domaine pour son efficacité, ses fonction 2D et 3D et le fait qu’il propose une version, très complète, gratuite à tous. Il utilise le langage de programmation c#, qui est un langage orienté Object.

Visual Studio

# Présentation du projet

# Journal de bord

## Création du personnage principal (Seth)

### Graphisme

Avant de débuter de dessiner le personnage, j’ai pensé à comment je voudrais qu’il soit graphiquement et aux différentes contraintes. Tout d’abord, il va s’agir du personnage principal est donc il devra convenir à toute sorte de situation, d’environnement ou d’ambiances. J’en ai donc convenu qu’il serait préférable, qu’il soit avec un style graphique simple et épuré et également en noir et blanc pour, comme dit précédemment, convenir à toute sorte d’ambiance colorées.

Après avoir établi tous ces objectifs, j’ai commencé à faire plusieurs tests sur Photoshop. Après avoir fini le personnage, il m’a fallu compléter certaines parties cachées par d’autres membres comme par exemple : le bras gauche du personnage avec l’épaule caché par le corps de celui-ci. J’ai fait cela afin de pouvoir l’animer par la suite et déplacer chaque membre à volonté. Pour le même but, chaque partie du corps (jambes, bras, corps et tête) doit se trouver sur un calque différent avant de les exporter dans Unity pour la prochaine étape. Afin de les exporter de Photoshop à Unity, j’ai tout d’abord installé un *package manager*[[4]](#footnote-4) (2D PSD Importer) capable de directement importer un format Adobe Photoshop dans la plateforme.



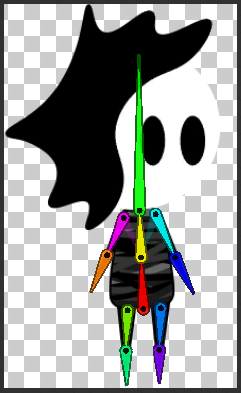
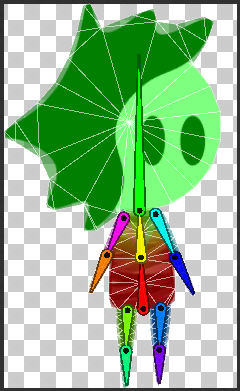
### Animation

Pour l’animation, trois façons étaient possible. La première était de dessiner image par image chaque partie de l’animation sur Photoshop. Mais cette méthode prend énormément de temps, vu qu’il faut dessiner chaque image une par une et n’aurait pas donner un rendu très lisse. La deuxième et la troisième se sont sur Unity directement. La seconde était de bouger membre par membre le personnage et le logiciel s’occupait de faire les mouvements entre deux positions. Mais cette méthode ne donnait pas un rendu très réaliste et cela prenait pas mal de temps également. J’ai donc finalement choisi la troisième méthode, qui est un système de *bones*[[5]](#footnote-5) dans le personnage afin de lui donner un rendu très réaliste et cela permet également de faire différentes torsions sur le corps, ce qui donne un résultat encore plus réaliste. Au début, j’avais beaucoup hésiter à utiliser cette méthode, car elle est souvent, presque toujours, utiliser pour la 3D alors que mon projet est 2D. Ensuite, j’ai fait plus de recherche, j’ai essayé pour voir ce que ça donnait et j’ai finalement j’ai décidé de l’utiliser pour mon projet.

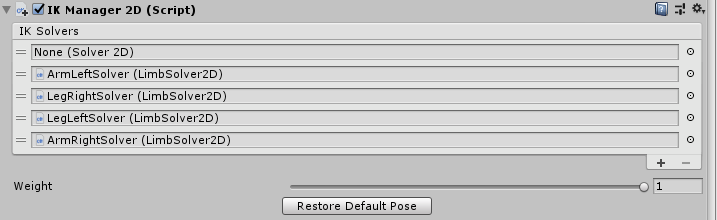
#### Squelette

Pour réaliser cette technique, avant de commencer, j’ai téléchargé dans Unity deux *package manager* : 2D Animation, qui contient tous les outils nécessaires pour l’animation grâce au squelette, et 2D IK, qui servira par la suite afin de pouvoir utiliser la cinématique inverse (qui sera utilisé pour pouvoir faire des mouvements et des rotations plus rapidement avec le personnage en question).

Après ces préparations, j’ai pu passer à la création des fameux *bones*, qui sont des vecteurs. J’ai décidé d’en placer deux pour le corps, ensuite deux pour chaque jambes (qui sont relié au *bone* du bas du corps), puis deux pour chaque bras (qui sont relié au *bone* du haut du corps) et pour finir, un pour la tête (relié au *bone* du haut du corps). L’étape suivant est de créer la *geometry[[6]](#footnote-6)* qui va servir aux os é déformer le personnage durant les mouvements. Elle peut être généré automatiquement, cela est plutôt efficace mais j’ai quand même dû faire quelques modifications manuellement. La dernière étape est de gérer le *weight* [[7]](#footnote-7), il s’agit de définit l’influence et la force de la torsion par exemple la tête n’influence que très peu le torse et donc j’ai diminuer le *weight* de la tête sur le torse.

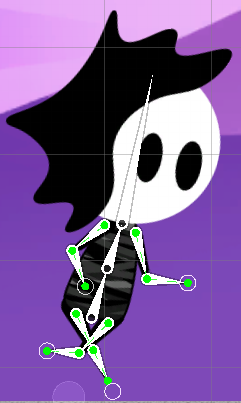
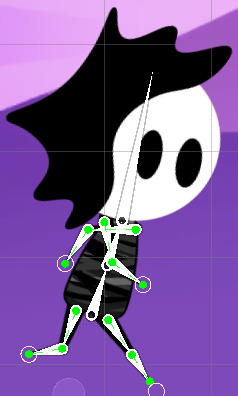
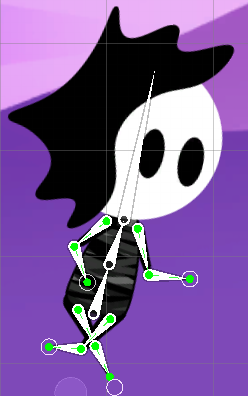
Départ Mise en place des *bones* *Weight* et *geometry*  

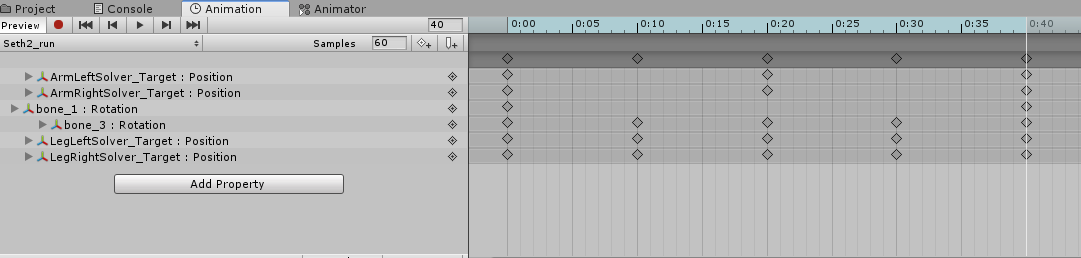
Puis afin de pouvoir bouger les bras et les jambes naturellement, j’ai utilisé la cinématique inverse pour pouvoir effectuer les rotations et les mouvements de ceux-ci. Je vais venir placer des points aux extrémités de chaque bras et jambes qui va chacun contrôler les deux bones correspondants. Il me suffira donc à bouger ces points pour pouvoir bouger les jambes/bras en entier et de manières plus naturelles.



Ensuite vient l’animation, pour ce personnage, j’ai fait 3 animations : le sur-place, la course, le saut. Toutes ces animations ont été faites sur Unity, ce logiciel permet de faire des animations lissent et soigner. Afin de faire une animation, il faut définir, en bougeant les bones ou les points, des pause clé de l’animation et le logiciel va s’occuper de faire la transition entre celle-ci.

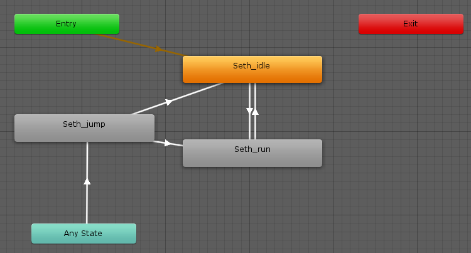
Les différentes poses clés pour la course





#### Animator Controller[[8]](#footnote-8)

L’*animator Controller* est ce qui décider quelle animation jouer et à quel moment. Il y a, sur cette photo, 3 animations : le sur-place (Seth\_idle), la course (Seth\_run) et le saut (Seth\_jump). L’animation orange est celle qui est sélectionner par défaut, cela veut dire que s’il ne se passe rien, ce sera cette animation qui sera jouée. Ensuite pour passer d’une animation à l’autre, j’ai créé des transitions entre elles. La première est celle qui passe du sur-place à la course, puis une autre inversement. Afin de jouer cette seconde animation, il faut qu’une certaine condition soit complété. Ces conditions sont appelées paramètre. Celui pour la course sera la vitesse, si la vitesse est plus grande que 0,01 alors l’animation course est activée et si la vitesse est plus basse que 0,01 alors le personnage arrête de courir. Puis, la prochaine animation est le saut, mais ce qui va changer est que l’animation va partir de *Any States*[[9]](#footnote-9) car l’animation doit se jouer peu importe si le personnage est sur-place ou en train de courir. Pour la condition, à place d’avoir un paramètre avec une valeur (comme la course), ce sera soit vrai soit faux. Si le paramètre « saute » est vrai alors la première transmission est activée, il faut également désactivé la répétition pour le saut, car l’animation ne doit jouer qu’une seule fois. Pour les transitions de sorti du saut, il y en aura une allant vers le saut et une vers le sur-place. Pour le saut, les conditions seront :  « saute » est faux et que la vitesse est inférieure à 0,01. Pour la course : « saute » est faux et que la vitesse soit supérieur à 0,01.

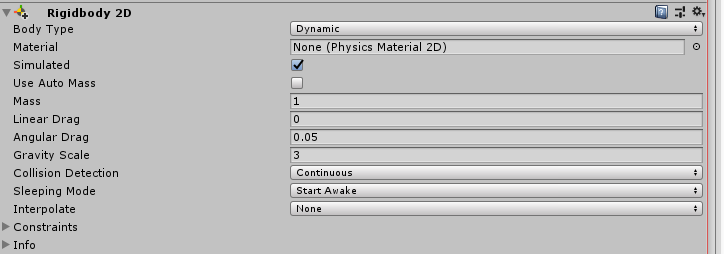


### Programmation

Afin de contrôler mon personnage, je vais lui ajouter plusieurs composant :

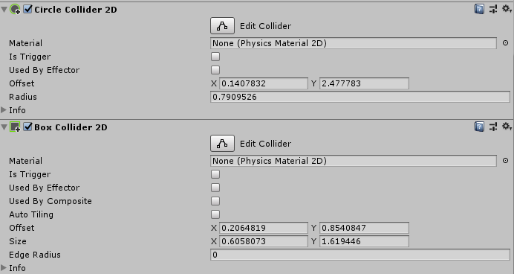
#### Rigibody 2D[[10]](#footnote-10)

*Rigibody* est un composant qui va ajouter de la physique au personnage. C’est grâce à lui que je peux régler la gravité et la masse, principalement sans avoir à faire des lignes de codes.



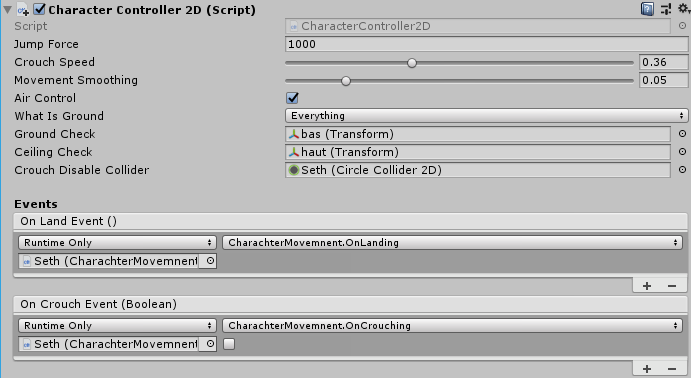
#### Circle & box collider[[11]](#footnote-11)

Ce composant va ajouter permettre au personnage d’interagir avec d’autre élément et ne pas passer à travers. Il va permettre d’ajouter des formes, carrées ou rondes, et se sont celle-ci qui entrera en collision avec d’autre *collider*. Dans mon cas, j’ai utilisé un rectangle pour le corps et un rond pour la tête. Ce sont donc les deux formes (contours verts) qui interagiront avec le décors et les autres personnages.



#### Charachter Controller 2D[[12]](#footnote-12)

Ce contrôleur est un script, mis à liberté d’utilisation[[13]](#footnote-13), permettant une base pour les mouvements du personnage que je vais modifier afin de l’améliorer et le faire convenir à ma situation. Grâce à ce script, je vais pouvoir modifier plusieurs éléments relatifs au personnage, comme la puissance du saut (Jump Force), pouvoir/ou pas contrôler le personnage dans les airs (*Air Control*), …



#### Charachter Movement [[14]](#footnote-14)

Ce composant est un script que j’ai écrit afin de faire bouger le personnage. Il y a premièrement le fait de vouloir bouger soit à gauche soit à droite. Afin de faire, il faut imaginer un axe horizontal sur lequel le personnage va se balader. Quand le joueur pressera la flèche gauche ou A alors le joueur ira à -1 et donc à gauche, pareil pour aller à droite avec la flèche gauche ou D, il ira à 1 et donc é droite.

-1

1

0



### Camera

Pour la caméra, j’ai décider qu’elle suivrait le personnage. Afin d’arriver a ceci, il a fallu créer un script. Grâce à celui-ci la caméra va suivre un object que j’ai placer sur le personnage et régler la taille de celle-ci afin qu’elle aille avec la taille du décors et du personnage.

1. La conception de jeux [↑](#footnote-ref-1)
2. L’aspect artistique d’un jeu [↑](#footnote-ref-2)
3. Jeu créer sans aide financière d’un éditeur [↑](#footnote-ref-3)
4. Gestionnaire de paquets : extension de Unity permettant certaines améliorations [↑](#footnote-ref-4)
5. Os : vecteurs permettant de récréer des os dans un personnage [↑](#footnote-ref-5)
6. Géométrie [↑](#footnote-ref-6)
7. Poids [↑](#footnote-ref-7)
8. Contrôleur d’animation [↑](#footnote-ref-8)
9. Tous les états [↑](#footnote-ref-9)
10. Corps rigide 2D [↑](#footnote-ref-10)
11. Cercle et box de collisions [↑](#footnote-ref-11)
12. Contrôleur du personnage 2D [↑](#footnote-ref-12)
13. Tiré de : <https://github.com/Brackeys/2D-Character-Controller> [↑](#footnote-ref-13)
14. Mouvement du personnage [↑](#footnote-ref-14)