**Documentation**

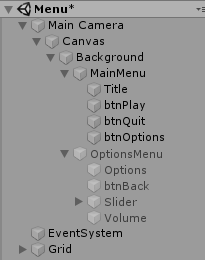
1. **Menu du jeu (Start, Quit, Options)**
   1. **But**

Le but est de créer un menu fonctionnel (bouton Start, Quit, Options) avec les designs



* 1. **Personne(s) chargé(s)**
* Lorenzo Di Benedetto
  1. **Développement**

J’ai créé un projet Unity dans lequel se trouve une scène



Dans cette scène j’ai mis des boutons qui fonctionnent à l’aide de script :

* Le bouton Start ouvre une nouvelle scène dans le jeu.
* Le bouton Options cache Le MainMenu et affiche l’OptionsMenu
* Le bouton Quit permet de quitter le jeu

1. **Sélection de niveau**
   1. **But**

Le but est de créer une sélection de niveau avec les designs



* 1. **Personne(s) chargé(s)**
* Lorenzo Di Benedetto
  1. **Développement**

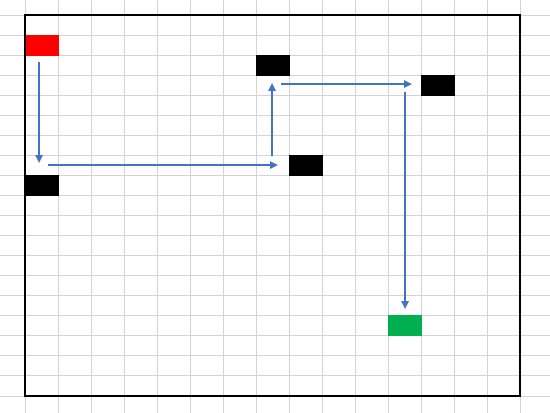
J’ai ajouté une scène à mon projet Unity



Dans cette scène j’ai mis des boutons qui fonctionnent à l’aide de script :

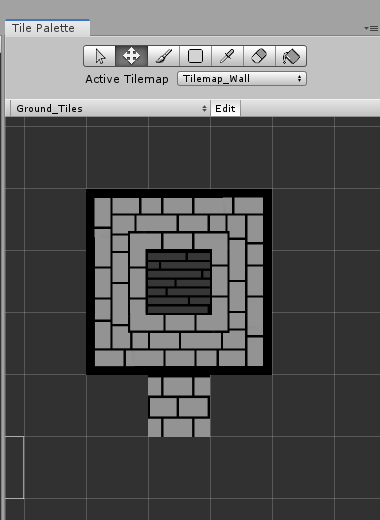
* Les boutons avec des numéros qui ouvrent des niveaux du jeu.
* Le bouton Back qui permet de retourner au menu

1. **Mouvement du joueur**
   1. **But**

Le but est de faire bouger le joueur grâce aux flèches directionnelles et qu’il se bloque à contre les murs

* 1. **Personne(s) chargé(s)**
* Ivan Serra
* Julien Breschbühler
  1. **Développement**

Notre idée à la base était de faire un tableau à deux dimensions dans lequel ont placerait le joueur, murs et autres et de le faire actualiser le joueur dans ce tableau à chaque déplacement cependant on a rencontrées quelques problèmes liées à Unity car c’est un gros moteur de jeux et il n’est pas fait pour ce genre de manipulations, ça faisait pas ce qu’on voulait malgré le temps passée pour résoudre le problème, ce qui nous a fait prendre 1 semaine de retard, j’ai donc pris la décision d’annuler cette vision du jeux et nous sommes parti de 0 sur une autre manière de faire pour le réaliser, cette fois on a employé certaines fonctions de Unity dont le Tilemap qui permettra d’avoir un genre d’éditeur de niveau :



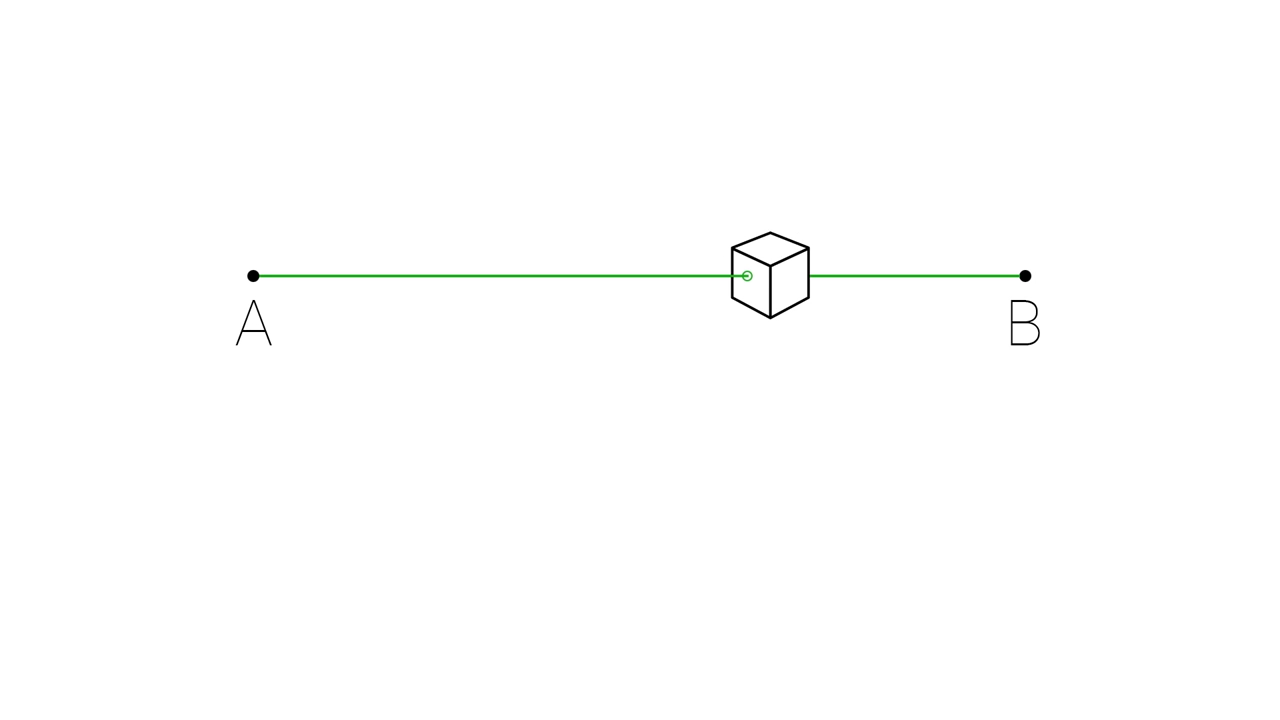
et aussi on a utilisé une Librairie « DOTween » qui permet de simplifier la manière de faire des mouvement en 2D sur Unity. On peut maintenant bouger grâce aux flèches.

1. **Le mur**
   1. **But**

Le but est que le mur bloque le joueur avant

* 1. **Personne(s) chargé(s)**
* Julien Brechbühler
  1. **Développement**

Il me fallait obtenir la position du mur et la faire arrêter pour cela j’ai utilisé des raycast de Unity, qui est un rayon qui quand il touche un objet me donne ces informations donc avec cela j’ai fait qu’il s’arrête avant.

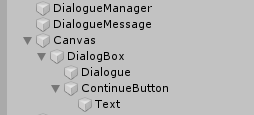


1. **Dialogues**
   1. **But**

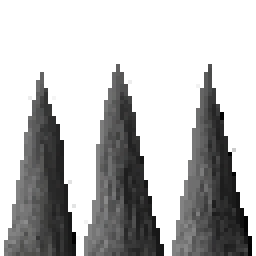
Le but est de créer un dialogue permettant de guider le joueur et de raconter l’histoire du jeu



* 1. **Personne(s) chargé(s)**
* Ivan Serra
  1. **Développement**

Pour cela on a 2 GameObject dans Unity, celui qui contient les dialogues et celui qui gère le toute, et aussi l’image dialogue qui est dans un Camvas. J’ai utilisé des « Queue » qui permettent d’entasser les dialogues les uns sur les autres et de les sortir dans le bon ordre. J’ai ajouté un bouton qui passe au dialogue suivant, et après j’ai ajouté une animation pour que sa rendre mieux et j’ai ajouté un moyen que il y aille un effet comme quand on tape un texte. Et voila tout est prêt pour faire des autant de dialogues qu’on veut.

1. **Les piques**
   1. **But**

Le but est que les piques tuent le joueur

* 1. **Personne(s) chargé(s)**
* Ivan Serra
  1. **Développement**

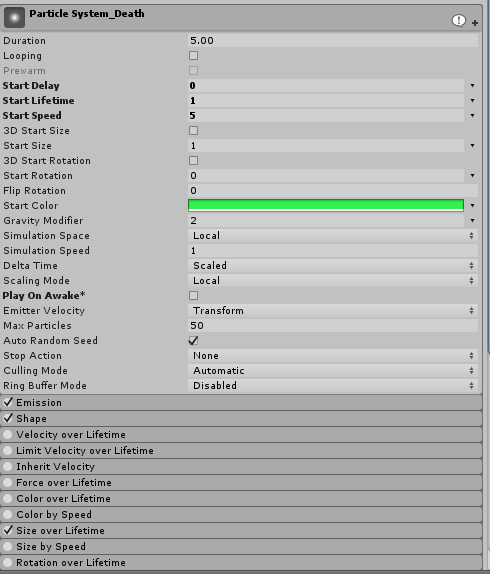
Il fallait que j’applique ce que julien à fait mais cette fois a la place d’arreter le joueur dans le mur fallait que je tue le joueur quand il touchait les piques. Mais fallait le faire rentrer un peu plus dans le bloc pour que il rentre plus dans les piques pour que ça soit plus réaliste. Et fallait bien évidement effacer le joueur.

1. **Effet quand le joueur meurt**
   1. **But**

Le but est de d’ajouter des particules pour que quand le joueur meurt ça donne un petit effet visuel qui rendra jolie.



* 1. **Personne(s) chargé(s)**
* Ivan Serra
  1. **Développement**

J’ai utilisé les particules système de unity qui se présente comme ci-dessous :

J’ai fait les réglages nécessaires pour que ça fasse une sorte d’explosion, puis j’ai du mettre dans le code que quand il touche des piques ça fasse cette effet

1. **Effet quand le joueur tape un mur**
   1. **But**

Le but est d’ajouter un effet de tremblement chaque fois que le joueur tape un mur.

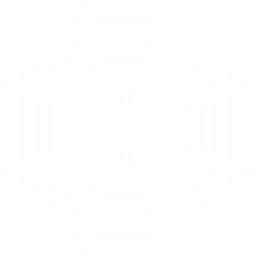
* 1. **Personne(s) chargé(s)**
* Ivan Serra
  1. **Développement**

J’ai fait pour cela que la camera bouge de quelque pixel pendant un certain moment et cela à chaque fois que le joueur tapait un mur .

Il me fallait après ce mouvement qu’elle revienne à sa position de base donc j’ai mis la caméra dans un container CameraHolder.

1. **La toile**
   1. **But**

Le but est que cette toile fasse que le joueur reste sur la toile.



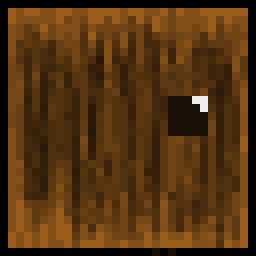
* 1. **Personne(s) chargé(s)**
* Julien Brechbühler
  1. **Développement**

J’ai essayé d’appliquer ce que j’avais fait pour la collision mais ça marchait pas car, le rayon si on était dessus, il s’arretait avant.

J’ai du changer le code du mouvement du joueur car il fallait que je fasse partir la Raycast un peu plus loin pour que il s’arrete dessus et puisse repartir sans que il ne touche jamais la substance dans laquelle il était dessus.

1. **La porte de fin**
   1. **But**

Le but est que quand le joueur arrive sur la porte ça change de niveau.



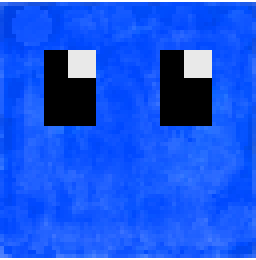
* 1. **Personne(s) chargé(s)**
* Julien Brechbühler
  1. **Développement**

J’ai appliqué ce que j’avais fait pour la collision des murs et à la place que ça l’arrête j’ai fait que la porte fasse que on change de scene pour que quand le joueur passe la porte le niveau suivant ce charge.



1. **Plusieurs joueurs** 
   1. **But**

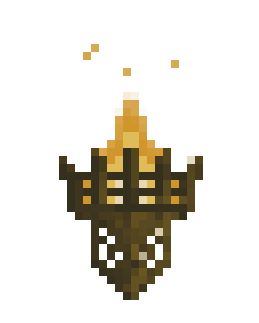
Le but est de compliquer la chose en ayant pas un joueur à gérer mais 2 ou plus pour les distinguer on a mis des couleurs différentes pour les démarquer. Donc maintenant si un slime meurt, ça fera recommencer le niveau et il faudra que tous les slime du niveau soit dans la porte pour le compléter



* 1. **Personne(s) chargé(s)**
* Ivan Serra
  1. **Développement**

Pour ce faire on a changé le code pour la condition de victoire pour qu’il. Faille obligatoirement 2 slime pour finir le niveau et aussi on a créé les différents slime et change le code de certaines fonctions pour les appliquer à double.

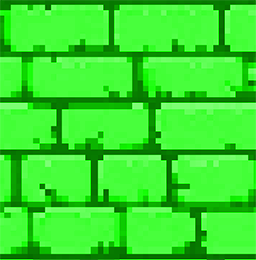
1. **Effet de lumière**
   1. **But**

Le but est de rendre le jeu plus agréable aux yeux de l’utilisateur en mettant un effet de lumière on rend le jeu un peu plus réaliste et vivant.

* 1. **Personne(s) chargé(s)**
* Ivan Serra
* Lorenzo Di Benedetto
  1. **Développement**

Pour les effets de lumière on a ajouté un élément « torche » qui éclaire une zone du jeu et on a assombri le niveau pour qu’on se rende bien compte de la luminosité

1. **Mur couleur**
   1. **But**

Avec l’arrivée de nouveau joueur on a pensé à faire des murs qui seraient passable avec la couleur des joueurs mais qui bloqueraient les autres couleurs

* 1. **Personne(s) chargé(s)**
* Ivan Serra
  1. **Développement**

Pour ce faire on a mis en place un système qui détecte la couleur de chaque joueur et ensuite quand il touche le mur, c’est lui qui lui dira s’il peut passer ou pas.

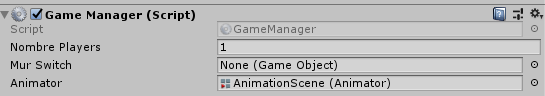
1. **Mur switch**
   1. **But**

Le but est qu’à chaque mouvement du joueur le mur disparaît ou apparaît et faudrait donc calculer à l’avance si on peut passer ou pas.

* 1. **Personne(s) chargé(s)**
* Ivan Serra
  1. **Développement**

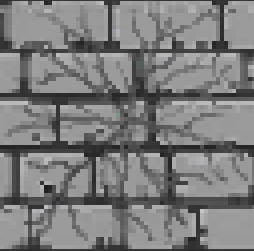
Pour ce faire on a créé un « manager controller » qui contrôlera chaque fois que le joueur appuiera sur espace et mettra le mur visible puis invisible.





1. **Break Wall**
   1. **But**

Le but est de crée un mur qui se casse au contact du joueur.



* 1. **Personne(s) chargé(s)**
* Brechbühler Julien
  1. **Développement**

Pour faire le mur qui se casse je suis passé par plein de dériver, j’ai essayé de supprimer la Tile, sans succès, j’ai essayé de la faire disparaitre, sans succès non plus, pis j’ai trouvé une idée qui est que quand l’on touche un « break wall » il trouve les position et remplace la tile par rien de se faite il se « supprime ».

1. **Portes+Clés** 
   1. **But**

Le but est de crée une porte qui s’ouvre avec une clé de la **même couleur**.

* 1. **Personne(s) chargé(s)**
* Brechbühler Julien
  1. **Développement**

Vous ne pourez pas traverser la porte tant que vous n’avez pas trouver la clé de la couleur correspondante.  
J’ai fait en sorte que les nom des TileBase (images des Tiles) soit nommée suivant une règle « Porte/Cle\_*nom de la couleur* » pour que l’on puisse différancier les couleurs des clé.



1. **Portail**
   1. **But**

Le but est de crée 2 portails pour pouvoir se téléporter à travers eux.

* 1. **Personne(s) chargé(s)**
* Brechbühler Julien
  1. **Développement**

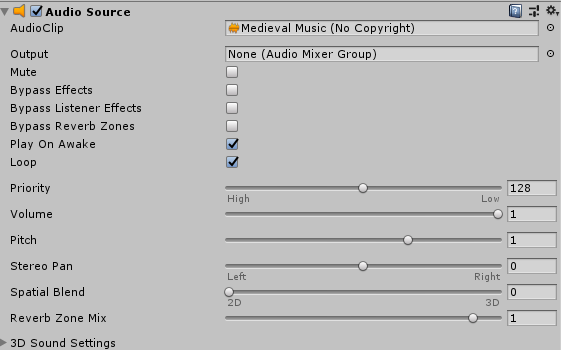
Pour les portails j’ai dû les mettre hors des TileMap du faite que je ne pouvais pas trouver l’emplacement du 2e portail.

Comme pour les portes j’ai nommé son nom du portail en manière à pouvoir les trouver facilement « Portal*\_nom de la couleur »*

1. **Musique**
   1. **But**

Le but est d’amener au jeux du son, pour entrer dans l’ambiance du jeu.

* 1. **Personne(s) chargé(s)**
* Serra Ivan
  1. **Développement**

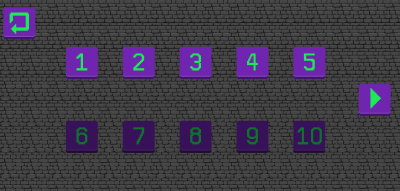
J’ai ajouté un Audio source qui correspond à l’image en dessous, avec j’ai les réglages pour qu’elle soit en boucle et la musique choisi.

Cependant je devais coder quelque chose pour que la musique soit permanente tout au long du jeu car la musique partait après un changement de scène pour cela j’ai trouvé une fonctionnalité qui permet d’avoir la musique dans toutes les scènes



1. **Sauvegarde**
   1. **But**

Le but est de créer une sauvegarde permettant de ne pas perdre notre progression dans les niveaux même si l’on ferme l’application.

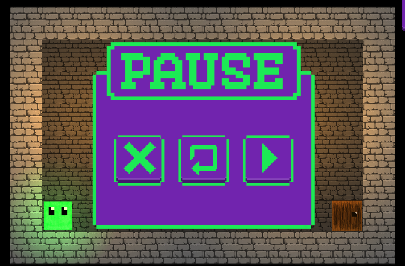


* 1. **Personne(s) chargé(s)**
* Lorenzo Di Benedetto
  1. **Développement**

J’ai modifié le script dans lequel se trouve le changement de niveau lorsque l’on réussit à terminer un niveau et j’ai aussi modifié le script qui permet de sélectionner un niveau. J’ai utilisé les PlayerPrefs c’est une classe qui permet de stocker des « paramétrage » qu’a fait un joueur dans un jeu. On stock une valeur à un endroit défini par sa clé, et on récupère cette même valeur via la même clé. 3 types de données sont supporté dans cet enregistrement : les nombres entiers (int), les nombres à virgule (float), et les chaînes de caractères (string). J’ai donc stocké comme valeur le nom de la scène du dernier niveau dans lequel le joueur se trouve dans une clé que j’ai nommée « Level ». Avec cette information stockée je la récupère en désactivant tous les boutons ayant une valeur supérieure à cette dernière (exemple sur le screen la valeur stocké est le niveau 5).

1. **Pause**
   1. **But**

Mettre pause au jeu et pouvoir le quitter



* 1. **Personne(s) chargé(s)**
* Brechbühler Julien
  1. **Développement**

J’ai placé 3 boutons (1 pour resume, 1 pour restart et le dernier pour revenir à la séléction des niveaux). On peut y accéder en pressant sur le bouton « pause » en haut à droite ou en pressant sur « escape ».

1. **Sauvegarde**
   1. **But**

Le but est de mettre l’application sous forme Android.



* 1. **Personne(s) chargé(s)**
* Ivan Serra
  1. **Développement**

Pour cela je dois modifier la façon de bouger le joueur. Pour cela je calcule la position du doigt de départ et celui de l’arrivé et regarde dans quelle direction de l’axe des x ou y. Ou ça se complique c’est que afin de le tester sur téléphone, j’ai rencontré quelques problèmes car Unity ont une application spécial « Unity remote 5 » mais il faut modifier les bons paramètres pour que ça marche.