



## PROJET PROGRAMMATION S2

### RAPPORT DE SOUTENANCE

G.O.A.T.

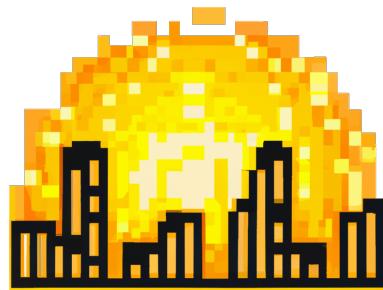
EPITA

INFORMATIQUE PRATIQUE

---

## Go On Another Town

---



*Auteurs :*

Denis MIROCHNIKOV,  
Zachée DESCAMPS,  
Dorian FOROT,  
Marc DESSEVRE

*Enseignant référent :*  
Martin Van Laere

7 Mars 2023

## Table des matières

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Origine et nature du projet . . . . .	2
1.2	Présentation des membres du groupe . . . . .	3
1.2.1	Denis Mirochnikov . . . . .	3
1.2.2	Zachée DESCAMPS . . . . .	3
1.2.3	Dorian FOROT . . . . .	4
1.2.4	Marc DESSEVRE . . . . .	4
<b>2</b>	<b>Réalisation, du point de vue de :</b>	<b>5</b>
2.1	Zachée Descams . . . . .	5
2.1.1	Le Moteur . . . . .	5
2.1.2	L'Éditeur . . . . .	7
2.1.3	Autre outils de développement . . . . .	8
2.2	Mark Dessevre . . . . .	9
2.2.1	Le monde du jeu . . . . .	9
2.2.2	Le menu . . . . .	10
2.3	Dorian Forot . . . . .	11
2.3.1	Les entités . . . . .	11
2.3.2	Les Items . . . . .	12
2.3.3	Les Blocks . . . . .	13
2.4	Denis Mirochnikov . . . . .	14
2.4.1	La décoration . . . . .	14
2.4.2	Les Mini Jeux . . . . .	16
2.4.3	Site-Web . . . . .	19
<b>3</b>	<b>Diverses retards</b>	<b>20</b>
<b>4</b>	<b>Conclusion</b>	<b>20</b>

## 1 Introduction

### 1.1 Origine et nature du projet

Dans le cadre de notre projet InfoSup, nous avons choisis de faire un jeux vidéo.

Il s'agit d'un jeu vidéo nommé "Go On Another Town", qui invite le joueur à évoluer dans un monde labyrinthique en 2D. Le joueur devra atteindre la ville GOAT, un soi-disant eldorado pour la survie. Car ce monde est devenu hostile, est la survie devient un véritable enjeu.

À travers ce rapport, nous vous invitons à comprendre d'où viennent nos idées, et comment elles ont pu être réalisé.



## 1.2 Présentation des membres du groupe

### 1.2.1 Denis Mirochnikov

Depuis mon plus jeune âge et jusqu'à maintenant j'adore la créativité, ça se ressentait par mes centres d'intérêt qui étaient assez différents de ceux des jeunes de mon âge. Au lieu de jouer aux jouets comme la majorité des enfants de 6 ans, je passais des journées à essayer de bricoler des objets dans le garage avec mon père. Il m'apprenait d'ailleurs énormément de choses sur la mécanique, et surtout sur le fonctionnement des différents objets comme par exemple un moteur électrique, ou même une pile. En bref on faisait pas mal d'expériences scientifiques ensemble afin que je puisse comprendre par la pratique le fonctionnement de ce monde.

C'est à partir de ce moment-là que j'ai compris que j'avais envie de lier ma vie à la science... Quelque chose de si vaste et complexe, mais tellement passionnant !

Et puis en 2014 tout s'est ruiné, on a dû quitter notre pays et commencer une nouvelle vie... Pour essayer de compenser les problèmes, je me suis intéressé à quelque chose qui était nouveau pour moi - la programmation. Elle m'a passionné, et j'ai commencé de plus en plus à m'intéresser et de découvrir de nouveaux langages de programmation, à participer à des concours ou même des expositions à travers des associations de mon lycée. À ce jour mon langage préféré reste le C++, même si j'apprécie énormément Python, Java et bien sûr C# .

Je pense que ce projet me permettra d'approfondir mes connaissances en création du jeu vidéo et surtout m'apprendra à travailler en équipe.

### 1.2.2 Zachée DESCAMPS

Depuis l'aube de l'humanité , l'humain est mauvais pour faire des introductions c'est pourquoi je vais passer dans le vif du sujet. J'ai toujours aimé créer des objets ou logiciels depuis tout petit , ça a commencé par créer des montes charges sur ma mezzanine et collé des kaplas ensemble au pistolet à colle ( je regrette cette action ils sont maintenant hors d'usage) puis par l'obtention de mon premier pc j'ai commencé à m'intéresser à la programmation par ce que je voulais parfaire mon image de geek ( c'est la réelle raison ) , j'ai commencé en logo puis j'ai continué en java pour faire des mods minecraft assez douteux (mais java est resté mon langage préféré jusqu'à aujourd'hui) . J'ai au fil des années porté mon attention sur le C et le C++ pour optimiser certains logiciels qui étaient trop coûteux en mémoire et calcul pour du java , sur le python qui m'as fait dire mes premières phrases de rabat joie tel que "c'est un langage pour ceux qui ne veulent pas programmer " ou encore "mélangé la mise en page et la syntaxe c'est une hérésie" ( oups le dernier est vrai ) le tout en passant par le powershell qui m'a laissé des séquelles ( plus jamais je ne veux voir des réseaux hyper V ) et encore d'autres comme le batch , la SQL , le javascript , le php et rapidement de l'assembleur x86 et 6502 pour du reverse engineering.

Bref j'aime la programmation parce que grâce à elle et avec juste un pc l'on peut créer des outils utiles au quotidien contrairement à l'électronique , la chimie et l'impression 3D qui demande des ressources coûteuse ( mes trois autres passions ).Pour ce qui est du jeux vidéo , je n'ai quasiment jamais eu de console .Pour jouer tous ce que j'avais sous la main était un vieil ordinateur portable qui laguait sur club penguin mais j'ai découvert assez tôt le monde de l'émulation c'est pourquoi les jeux de mon enfance sont les jeux les mieux notés sur Romstation ( Mario 64 , Ocarina of Time , Chrono Trigger , Banjo Kazooie ,etc .. ). Ce projet me permettra d'apprendre à travailler en groupe , ce à quoi je suis vraiment mauvais(cf. l'appréciation Parcoursup de mon professeur de NSI de terminale )

### 1.2.3 Dorian FOROT

Éminemment beau et spécialiste Ocaml et sortant de terminale, le bac en poche avec les spécialités Math-NSI, je me suis lancé dans l'aventure EPITA. Je ne dirais pas que je suis passionné par l'informatique, mais j'aime programmer. Depuis le collège je m'intéresse à ce domaine sans pour autant m'y lancer, par manque de temps et bien sûr de motivation. La spécialité NSI au lycée m'a initié à certains langages de programmations tel que le Python, mon amour de toujours, le php , la SQL ou encore le html css. Mais c'est oublier le précurseur de tout cela, le meilleur langage de programmation all time, celui avec lequel tout a commencé, j'ai nommé, Scratch. Comment pourrais-je ne pas rendre hommage au logiciel grâce auquel tout a commencé, celui qui m'a fait découvrir et aimer la programmation (même si c'est un bien grand mot pour ce que c'est). J'étais très fort en Scratch croyez le ou non, dans mon collège c'était moi le pro associable qui allait au cdi comme un sans ami faire du Scratch.

J'ai eu la chance (ou pas) d'avoir déjà eu l'occasion de faire un projet de ce type en NSI durant ma première avec mon professeur de NSI (cœur sur vous Mr.Lebret). Avec mon équipe de choc (non) nous avions codé de toute pièce un fps multijoueur (sans multijoueur mais c'est une histoire à part). Je connais donc la difficulté d'un tel projet, l'organisation et la rigueur qu'il faut avoir. J'ai très hâte de commencer le projet et de me donner à fond pour réussir à réaliser les objectifs que nous nous sommes fixés.

### 1.2.4 Marc DESSEVRE

Tout juste sortie de terminale en spécialité Math-Physique, comme beaucoup d'autres, j'ai décidé de me tourner vers EPITA car l'informatique m'a toujours remplie d'admiration. Je n'avais jamais osé m'y lancer pour faute de matériel et par peur de me perdre dans cet univers aussi grand. Cependant, aujourd'hui je suis à l'EPITA, en plein apprentissage du C#, du python et d'Ocaml, et la création d'un jeu vidéo est une très grande étape à franchir dans mon parcours.

D'autant plus que les jeux vidéo me passionnent (j'ai fait mon stage de troisième à Quantic Dream par exemple), ceux-ci m'apparaissent de plus en plus complexes au fil de mon apprentissage. J'ai par exemple appris l'existence d'une boucle de jeu très récemment.

Ainsi, ce projet est à la fois un défi important dans mon apprentissage, mais aussi une manière de mieux comprendre les fondations et la structure d'un jeu vidéo.

## 2 Réalisation, du point de vue de :

### 2.1 Zachée Descams

#### 2.1.1 Le Moteur

Tout d'abord pourquoi construire soi-même son moteur alors que "Unity" est autorisé ? Quand bien même Unity intègre de par lui-même l'affichage, la physique, la structure et les collisions, Unity n'est pas destiné à notre projet et la création d'un moteur est une chose à la fois passionnante et instructive. Avant le début du projet j'ai commencé à créer un moteur en Java (Langage dans lequel je suis le plus à l'aise) pour voir les possibilités ainsi que les difficultés. Après plusieurs semaines je suis arrivé à un résultat satisfaisant mais considérablement amélioré depuis dans le projet. Le plus dur et passionnant est, selon moi, de garder une structure cohérente entre tous les éléments afin de pouvoir facilement intégrer des nouveautés le tout en évitant une trop grande abstraction pour permettre l'optimisation matérielle là où elle est possible et/ou nécessaire. J'en suis donc venu à la structure suivante.

Le Jeu est divisé en salle ou niveau chacun représenté par une classe nommée Level , Le niveau sont sauvegardé sous forme d'image pour les blocs et sortie/entré les composants .Chaque pixel correspond à un bloc par sa couleur en RGB le rouge indique le type de blocs (décor, bloc physique, sortie, porte), le vert est l'indice du bloc dans son type et le bleu des données dépendante de son type (pour les sorties l'id de la room vers laquelle elle mène, pour les blocs l'orientation).Chaque Level possède une matrice pour ses collisions , ses textures ( système permettant d'économiser de la RAM ) , ses blocs spéciaux , ses blocs opaques (Pour optimiser l'affichage) , son fond découpé en blocs .Pour chaque niveau existe aussi un fichier XML encodant toutes les entités qu'il comporte ainsi que divers paramètre du niveau : "Est -il dans le noir " , "Quelle est son arrière-plan". Il ne peut y avoir qu'un niveau chargé à la fois pour des questions d'optimisation de la mémoire (j'essaie de ne pas dépasser les 100mo de mémoire vive. Il possède aussi une liste d'entités (classe propre au projet) et sert donc de passerelle pour toute interaction entre celles-ci mais il sert aussi à mettre à jour les entités et à retirer celle qui sont mortes. Etant donné que chaque est détruit lorsque le joueur change de salle, il sauvegarde l'état de ses entités dans un fichier XML afin de pouvoir revenir à l'état où on l'a laissé.

Pour afficher ces niveaux il y a la classe mère Caméra, elle s'occupe de l'affichage mais aussi de la mise à jour du niveau et du joueur. Elle s'occupe du scrolling afin que le joueur soit toujours dans son angle de vue. La Caméra est un peu le paradis de l'optimisation. Pour chaque frame est définis un rectangle de dessin dans lequel sera affiché le niveau et le joueur afin de ne pas perdre du temps à dessiné dans le vide. La carte d'opacité des blocs du Level permet d'éviter de dessiner l'arrière-plan derrière une bloc complet. Dans le niveau sombre, ce qui est dans l'obscurité n'est tout simplement pas dessiné. Ce rectangle dessin permet aussi au niveau de ne pas mettre à jour les entités non présentes dans le champ de vision.

Les entités sont une suite d'héritage multiple de la classe mère Entity permettant d'avoir des propriétés différentes entre chaque entité :

**Entity :**

Une hitbox (Rectangle de collisions), des coordonnées et une update.

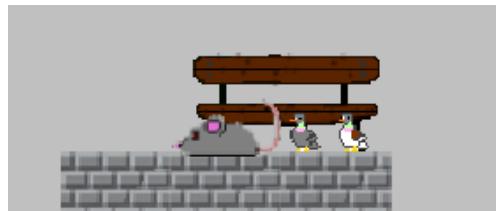
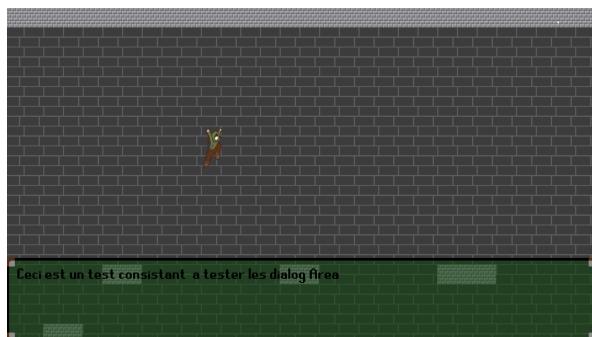
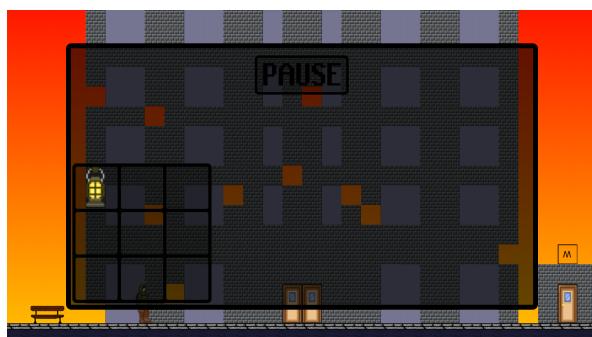
**ActiveEntity :**

L'ajout d'un Sprite ainsi qu'un SpriteManager (classe simplifiant la manipulation d'animation de l'entités) et de vecteur vitesse et accélération.

**LivingEntity :**

Des points de vie ainsi d'une méthode pour appliquer ses vecteurs en fonction du niveau actuel (collisions).

Je me suis aussi occupé du menu pause ainsi que les boîte de dialogue qui s'affichent lors d'un changement d'état du jeu, des Items qui sont fait pour fonctionner de couple avec l'entité itemHolder afin de ne pas trop complexifier leurs fonctionnements.

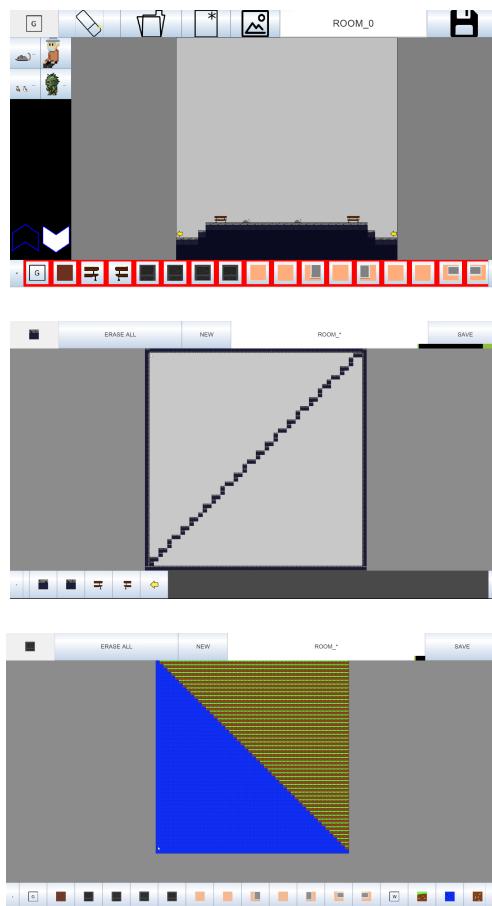


### 2.1.2 L'Éditeur

L'encodage des niveaux sous forme de couple XML-PNG est certes très simple à manipuler en code mais créer des niveaux à la main devient vite une vraie horreur. C'est pourquoi en parallèle du moteur j'ai développé un éditeur de niveau en java (car je suis plus à l'aise dans ce langage mais aussi car il est inter plateforme et permet donc à Denis de l'utiliser) qui se greffe sur le projet pour permettre de simplifier la création de chaque salle (d'où son nom Room Creator).

Il déduit une palette de bloc depuis les fichiers du projet ainsi qu'une liste d'entités disponibles grâce au Sprite de celles-ci. L'éditeur est d'une complexité similaire au moteur par certains aspects car il s'agit d'une sorte de Paint modifié. Mais j'étais loin de me douter que recoder Paint sera si complexe. J'ai donc dû recoder des fonctionnalités auxquelles je n'avais pas pensé comme le zoom, le déplacement de l'image, le remplissage d'une aire ou encore le dessin de ligne mais aussi des fonctionnalités propres au projet comme le déplacement entre salles via les cliques sur une sortie, la génération de fichier XML compréhensible par java et C#, la possibilité de donner un nom et des arguments aux entités (du texte pour les PNJ par exemple), l'ajout d'un fond, de rotation des blocs et d'autres. Grâce à tout cela il est possible de créer des niveaux sans écrire une ligne de code !

A vrai je pense même réutiliser cet éditeur pour créer d'autres jeux tant il peut simplifier la tâche laborieuse de création de niveau (dans mes anciens jeux je faisais ça à la main).



### 2.1.3 Autre outils de développement

Au fur et à mesure de la progression du projet j'ai pu trouver certaines tâches répétitives comme la création de Sprite, j'ai donc fait rapidement deux petits scripts pour agencer les Sprite ou encore découper des images sous formes de blocs.

Au-delà de la structure du projet j'ai aussi participé à la création de son contenu en créant des entités comme les entités décoratives ou les plateformes mouvante, les pigeons, les bulles de texte et Entrés/Sortie ainsi que les Portes de niveau mais aussi un item : la lampe qui fait suite directe à la création des niveau sombre (qui m'ont donné beaucoup de mal pour l'optimisation). J'ai aussi passé du temps à créer des outils simplifiant certaine chose comme les animations avec le SpriteManager qui évite d'avoir à découper chaque Sprite d'une animation ou encore la Palette qui permet d'obtenir directement le bloc associé à une couleur et de le mettre à la bonne taille (en fonction du niveau et de l'écran sur lequel est affiché le jeu). Les voidArea qui permettent de définir une zone ayant un comportement quand le joueur y entre.

Un moteur est une chose complexe et assez capricieuse (surtout le mien) , et je n'ai pas voulu pénaliser mon groupe à devoir relire tous mes codes pour y ajouter des fonctionnalités et j'ai donc beaucoup travaillé sur la simplicité d'ajout de nouvelle entité, items ou niveau. Le grand point fort de ce projet réside là-dedans. Ajouter une entité ne demande que de créer un classe et une image ayant le même nom, les collisions sont déjà gérées, l'affichage est automatique tout comme l'update, l'ajout dans un niveau se fait directement par l'éditeur et il en va de même pour les items. En somme pas besoin de comprendre une grande partie mon code pour créer du contenu.

## 2.2 Mark Dessevre

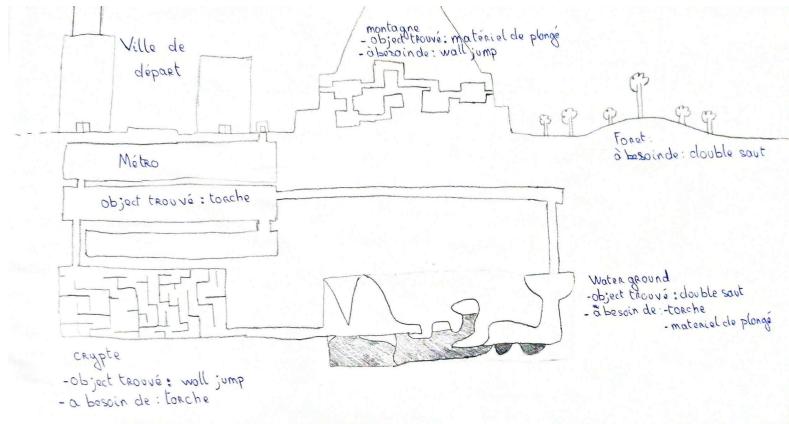
### 2.2.1 Le monde du jeu

Avant même de pouvoir dessiner nos niveaux, nous avons dû nous décider sur l'univers dans lequel allait évoluer le joueur. Nous avons décidé de nous placer dans un univers postapocalyptique, ce qui nous permet de créer des niveaux dans un monde qui nous est relativement familier. En effet, nous n'avons pas besoin de créer un monde plausible de toute pièce, le notre nous offrant déjà tout ce dont nous avons besoin en termes de design.

Nous avons alors un monde. Il nous reste maintenant à l'exploiter. J'ai alors fait une première ébauche du monde, avec ces différents environnements et leur particularité. Par exemple, les cryptes seraient sombres et impraticables sans lampe ou torche, ou encore les montagnes seraient infranchissables sans matériel d'escalade. Le monde semblait alors plutôt cohérent

Mais alors que je dessinais les niveaux et les implémentait dans le jeu, nous nous sommes rendu compte que ces niveaux étaient bien trop linéaires, et ne ressemblait en rien à un metroid-vania. Il y avait bien quelques aller retours par-ci par-là, mais rien de vraiment labyrinthique. J'ai alors repensé le monde, en dispersant les objets obligatoires un peu partout, mais surtout en retirant les zones inutiles. De 11 environnements, nous sommes descendus à 7. De plus, cette baisse du nombre totale d'environnement a permis de renforcer la cohérence du monde, en retirant la présence d'un désert qui n'avait rien à faire là, par exemple.

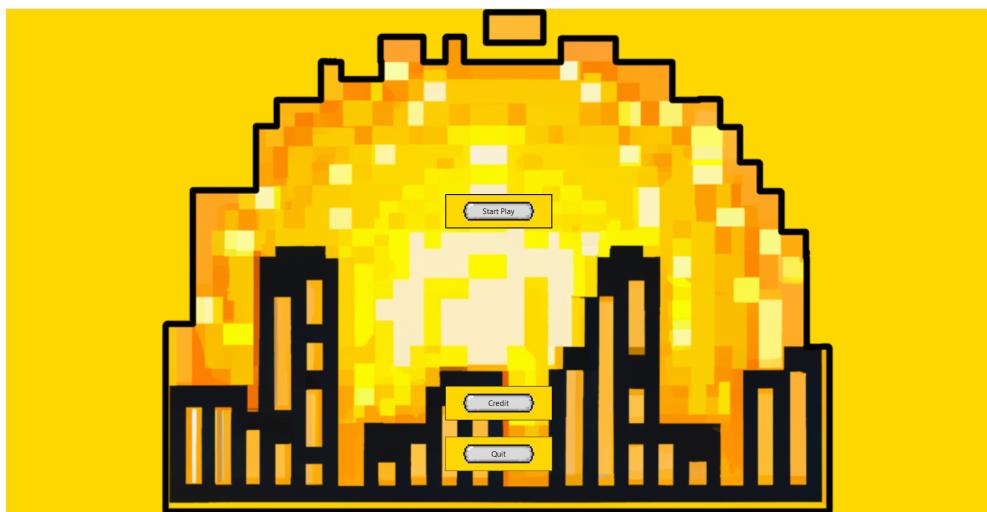




## 2.2.2 Le menu

J'ai été chargé de créer un menu de lancement du jeu. J'ai donc dû me renseigner sur les différentes manières de manipuler Winform. Je me suis renseigné, j'ai essayé, bien sûr ça ne fonctionnait pas. Le problème était la manière de lancer le projet. Celui utilisait Winform aussi, ce qui rendait impossible l'utilisation de la méthode `Process.Start()`. J'ai essayé divers méthodes, tel que passer le projet en référence, ou créer une variable du type `Process`, afin de rajouter des arguments dans la manière de lancer le .exe. La dernière a fonctionné, mais non sans difficulté.

L'aspect technique étant fonctionnel, il fallait maintenant s'intéresser à l'aspect esthétique. Ici, tout s'est relativement bien passé. Les seuls ralentissements étaient dû à une mauvaise connaissance de Winform. Les boutons s'affichaient mal en grand écran, l'image d'arrière-plan ne se centrait pas, bref, rien de très problématique. J'ai donc décidé de rester dans un style Pixel Art, avec des teintes jaunes-oranges.



## 2.3 Dorian Forot

### 2.3.1 Les entités

L'implémentation d'entités est un élément presque nécessaire dans un jeu de type Metroidvania. C'est pourquoi nous n'avons pas manqué à l'appel en implémentant un grand nombre d'entités pour différentes utilisations. Tout d'abord, il était nécessaire pour nous de nous exercer sur l'implémentation en créant d'abord des unités "faciles", telles qu'un rat sans réelle interaction qui nous sera surtout utile à des fins décoratives. Nous avons donc créé différentes classes, les unes héritant des autres, ce qui nous permet de faire la distinction entre les fonctionnalités et le caractère des ennemis. Par exemple, la classe "Entity" nous permet d'attribuer des coordonnées, des fonctions qui nous permettent de connaître la distance, ainsi que de nombreuses variables utiles pour connaître l'état des classes qui en héritent. L'implémentation des différentes classes qui nous serviront à coder les entités pour le reste du jeu, à savoir les classes "ActiveEntity", "LivingEntity" et "TriggerEntity", chacune héritant de la précédente, nous a permis d'avoir un travail linéaire sur la fabrication des éléments du jeu.

Ainsi, le gros du travail s'est focalisé sur ces trois classes. De plus, le jeu fonctionne par "frames", c'est-à-dire le nombre d'images par seconde qui seront chargées lors de l'exécution du jeu. Tous les changements se font à certains intervalles de temps. C'est pourquoi tous les changements que subiront les entités seront recensés dans les fonctions "Update()" et "UpdateAnimation()", toutes deux présentes dans la classe "ActiveEntity". La première nous permet de mettre à jour tous les éléments qui sont en lien avec notre ennemi, par exemple ses points de vie, s'il est toujours en vie, ses différents déplacements et les possibles collisions qu'il peut avoir avec la carte, le joueur ou les objets qu'il possède. La deuxième nous permet de charger les différentes images nous permettant d'illustrer les différents états de nos créatures, dans le but d'avoir un gameplay plus agréable à jouer.

La seule difficulté lors de la création des entités était de leur attribuer des mouvements qui ne dépendent pas du joueur, c'est-à-dire de faire en sorte qu'elles puissent se déplacer toutes seules quand elles sont trop loin pour agresser le joueur. Nous avons donc décidé de rester simples et de leur donner une liste simple de différents mouvements qu'elles peuvent effectuer en fonction de leur environnement et de manière aléatoire. Néanmoins, dans l'ensemble, l'implémentation de ces créatures était très simple grâce au travail de Zachée sur les collisions avec la carte. Notre but à terme serait d'avoir une sorte d'intelligence artificielle, pas un deuxième joueur qui serait contrôlé par l'ordinateur, mais des créatures qui soient vraiment "intelligentes" et qui puissent se déplacer par elles-mêmes, qu'elles puissent interagir non seulement avec la carte et le joueur, mais également avec les objets. Pour cela, nous envisageons de créer un système qui permettrait aux créatures d'interagir avec le joueur en fonction de ce qu'il possède ou de ce qu'il ne possède pas.

Grace à cet exercice qui était d'implémenter des entités sans utiliser Unity, uniquement avec Rider via C# j'ai pu en apprendre beaucoup sur le comment fonctionne réellement le code en C#. J'ai compris vraiment comment marchait le système de classes avec la POO, notamment les héritages, les mots-clés comme Override, Virtual ainsi que l'utilisation des getters et des setters pour l'accessibilité à des variables d'une classe. J'ai également appris à optimiser mon code, en effet j'ai pu comprendre vraiment le pourquoi il faut optimiser, pourquoi il ne faut pas coder "bêtement" et vraiment comprendre ce que l'on fait pour pouvoir avoir le moins d'exécutions possibles pour faire en sorte que le jeu puisse tourner dans des conditions raisonnables sur n'importe quel outil.

### 2.3.2 Les Items

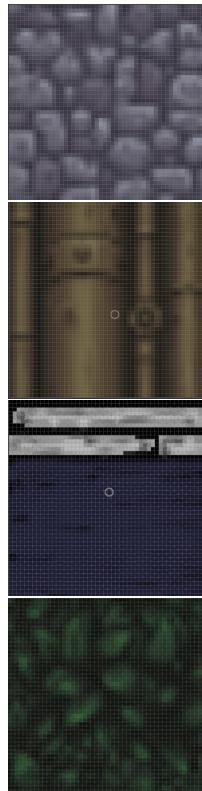
Pour ce qui est des items la tâche était plus simple que celle des entités. En effet pour les items nous n'avons pas à gérer tout ce qui est le déplacement sur la map ni l'interaction face au joueur. Pour gérer les items plus facilement nous avons créé dans la Class du player un Inventaire. Celui-ci nous permet avec l'aide de requêtes LINQ de savoir si oui ou non un item est possédé pour savoir si le joueur peut effectuer une telle action ou non. Pour implémenter les Items plus facilement nous avons décidé de passer par la création d'une Class qui héritent des ActiveEntity, ce qui nous permet de récupérer des informations qui peuvent être très utiles, mais surtout de faire en sorte que tant que l'item n'est pas récupéré il soit considéré comme une entité afin qu'il puisse être bien dessiné lors du chargement du niveau.

Nous n'avons pour l'instant pas encore implémenté beaucoup d'items, en effet cela n'est pas notre priorité, nous préférons d'abord nous focaliser sur les éléments plus essentiels. Nous nous projetons d'ajouter nombres d'items qui nous permettraient de donner plus retord aux joueurs dans les différents niveaux en offrant plus de réflexions et de difficulté. Notamment plus d'armes qui permettront au joueur de se débarrasser de certaines créatures. Ou des items tels que des interrupteurs qui eux pourraient permettre d'ouvrir des portes et donner accès à d'autres parties du niveau.

### 2.3.3 Les Blocks

Un élément essentiel dans tout jeu est le décors et l'environnement dans lequel le joueur sera amené à évoluer. Nous avons donc utilisé principalement l'application gratuite Paint.net. En effet nous avons décidé de fonctionner par pixel, ainsi les blocks pour le jeu sont tous fait sur du 50\*50 pixels. C'était donc assez simple de les faire grâce à l'éditeur Paint.net. De plus nous avons choisi des graphismes simples pour notre jeu pour rejoindre les graphismes assez classiques des autres metroid-vania.

Le logiciel Paint.net était très simple à prendre en mains et ses nombreuses fonctionnalités ont grandement facilité le travail sur les blocks que nous utilisons dans le jeu. De plus l'implémentation de ceux-ci dans le jeu ont été grandement simplifiés grâce au ROOMCreator, il suffisait simplement de mettre les blocks dans le dossier en les nommant pour qu'ils soient ensuite reconnaissables par le Moteur et qu'ils soient bien dessinés dans le jeu lors de l'exécution des niveaux. Néanmoins faire les blocks de cette façon-là était assez longue, en effet il fallait la plupart du temps faire pixel par pixel ce qui était assez long et fastidieux.



## 2.4 Denis Mirochnikov

### 2.4.1 La décoration

L'ambiance est un élément crucial de tout jeu vidéo car elle peut aider à immerger le joueur dans l'univers du jeu. Elle peut évoquer des émotions et des sensations qui ajoutent une profondeur et une dimension supplémentaires à l'expérience de jeu. Sans une ambiance appropriée, un jeu peut sembler plat et sans vie, ce qui peut dissuader les joueurs de continuer à jouer.

Nous avons nous-mêmes ressenti ce manque d'ambiance après la création des niveaux de notre jeu. Les niveaux semblaient vides et sans vie, nous avons alors eu l'idée de rajouter des éléments de décors pour donner vie au jeu.

Je me suis alors engagé à travailler cet aspect en commençant par étudier chaque niveau pour comprendre son thème et son ambiance. Le niveau "Metro" m'a semblé particulièrement intéressant à améliorer car par définition c'est un endroit sombre où l'on se cache du danger extérieur. Rendre ce lieu plus vivant et plus rempli était donc une tâche intéressante. J'ai alors pris mon iPad et a commencé à travailler sur les éventuels objets à l'aide de "ProCreate" ; à ce jour le niveau "Metro" possède des nombreuses décorations comme par exemple les poubelles, les wagons cassées (et en marche) de trains, de la décoration extérieure tels que des panneaux, ou encore les échelles et les toiles sur les coins des murs. Toutes ses petites améliorations ont joué le jeu et ont pu finalement rendre le niveau "Metro" beaucoup plus intéressant visuellement et donc immersif pour les joueurs.

On va continuer dans cette voie pour chaque niveau afin de faire de notre création - quelque chose de particulier et mémorable pour nos utilisateurs.



## 2.4.2 Les Mini Jeux

Je me souviens quand on a commencé à travailler sur, Go On Another Town. Au début, on pensait que le jeu principal était suffisamment captivant pour garder l'attention des joueurs tout au long de leur expérience. Cependant, au fil du temps, nous avons commencé à réaliser qu'il manquait quelque chose d'essentiel pour donner du piquant au jeu.

En jouant moi-même, j'ai remarqué que le passage d'une pièce à l'autre était plutôt simple et sans réelle difficulté. Il n'y avait pas de réel challenge à relever, et cela pouvait même parfois faire disparaître l'esprit de compétition chez certains joueurs.

C'est à ce moment-là que l'idée de créer des mini-jeux pour chaque passage entre les pièces m'est venue en tête. Je voulais donner aux joueurs quelque chose de différent et d'amusant à faire pour casser la monotonie du jeu principal et maintenir leur intérêt. Chaque mini-jeu serait débloqué à la fin de chaque pièce, offrant une petite pause ludique et une occasion de se divertir tout en restant dans l'univers du jeu. C'est là que j'ai pu rencontrer pas mal de difficultés.

Après avoir eu l'idée de créer des mini-jeux pour casser la monotonie de notre jeu principal, j'ai commencé à réfléchir à la façon la plus efficace de les créer. Finalement, j'ai opté pour l'utilisation de WinForms. Cependant, j'ai rapidement réalisé que ce framework n'était pas conçu pour fonctionner sur des machines tournant sous MacOs. Cela a représenté un véritable défi et j'ai dû consacrer beaucoup de temps pour trouver une solution.

Face à cette difficulté, nous avons finalement décidé de modifier notre approche en matière de développement. Zachée a donc pris en charge les tâches qui m'avaient été initialement assignées liées au moteur du jeu, tandis que je me suis concentré sur la recherche d'une solution pour faire fonctionner WinForms sur MacOs, ainsi que sur d'autres tâches qui ne nécessitaient pas l'utilisation de WinForms, comme le développement de sites web, dont je vais vous parler par la suite. Après plusieurs tentatives, j'ai enfin réussi à trouver une machine virtuelle qui me permettait de travailler efficacement sur le développement des mini-jeux.

En somme, même si l'utilisation de WinForms représentait un véritable challenge pour notre projet, nous avons réussi à surmonter cette difficulté en trouvant une solution alternative et en répartissant les tâches entre les membres de l'équipe. Cela nous a permis de continuer à avancer et à améliorer l'expérience de jeu pour nos utilisateurs.

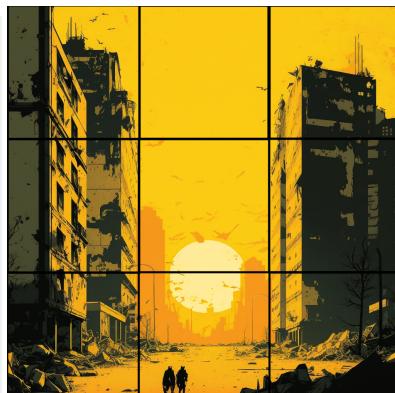
En ce qui concerne les mini-jeux, j'ai pu en développer deux jusqu'à présent, bien qu'ils ne soient pas encore reliés au moteur du jeu principal. Nous cherchons actuellement la solution la plus optimale pour les intégrer sans rencontrer de difficultés. L'idée principale est de les intégrer de la même manière que le menu de pause, c'est-à-dire en arrêtant l'avancement du jeu principal et en lançant le mini-jeu choisi en fonction de la Room.

Le premier jeu est assez simple et c'est le premier que j'ai pu réaliser. Il s'agit en quelque sorte d'un puzzle à partir d'une image choisie en fonction du Room pour pouvoir connecter le niveau au mini-jeu. Le joueur devra résoudre ce puzzle pour accéder au niveau suivant, ce qui ajoute un aspect d'éénigme et donc va attirer l'attention de joueur.

Ce mini-jeu a un potentiel intéressant qui peut être encore amélioré. Par exemple, on pourrait ajouter un chronomètre pour indiquer au joueur le temps qu'il a pour résoudre le puzzle. Cela ajouterait une dimension de compétition au jeu, d'avoir terminé le puzzle a un temps donné.

De plus, on pourrait envisager de remplacer l'image actuelle par une série de nombres pour rendre le jeu plus stimulant. Cela permettrait aux joueurs de s'exercer à la reconnaissance de chiffres et de motifs, tout en améliorant leur capacité à résoudre des puzzles. Enfin, on pourrait ajouter d'autres niveaux de difficulté pour que les joueurs puissent progresser et continuer à s'amuser avec le jeu.

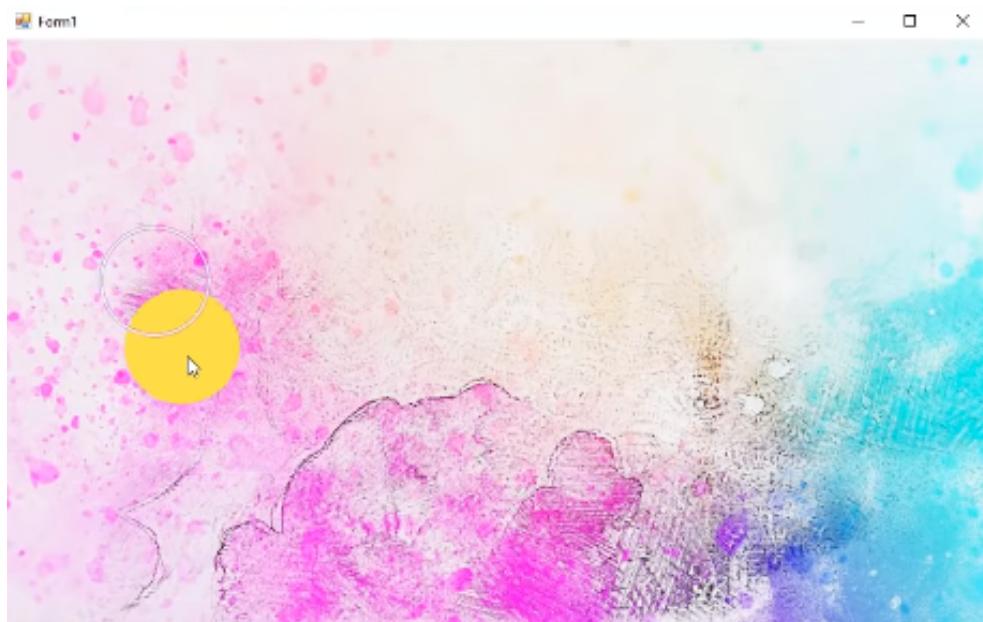
<b>9</b>	<b>10</b>	<b>1</b>	<b>2</b>
<b>7</b>	<b>14</b>	<b>6</b>	<b>15</b>
<b>5</b>	<b>4</b>	<b>11</b>	
<b>13</b>	<b>12</b>	<b>8</b>	<b>3</b>



Le deuxième mini jeu que j'ai commencé à développer il n'y a pas si longtemps est un jeu de rythme et de gestuelle qui ressemble à "OSU!". Le concept de ce jeu est de créer une expérience immersive où les joueurs doivent suivre des rythmes et des gestes spécifiques pour réussir le passage à un autre niveau.

Le jeu mettra à l'épreuve les compétences de coordination et de réflexes des joueurs. En suivant le rythme et en effectuant les gestes corrects, les joueurs pourront gagner et donc passer au niveau suivant du jeu principal. Le but est de rendre le jeu principal plus amusant et stimulant en ajoutant cet aspect de rythm.

Le développement de ce jeu n'en est qu'à ses débuts, mais j'ai de grandes ambitions pour lui et j'espère que d'ici peu de temps il sera terminé et on pourra le rajouter dans le mini bibliothèque des jeux de Go On Another Town.



### 2.4.3 Site-Web

Suite aux problèmes rencontrés avec le framework Windows Forms, j'ai pris la décision de me charger intégralement de la création de notre site web afin d'optimiser notre investissement. J'ai choisi de travailler avec le framework Django, écrit en Python.

Django est un framework très populaire qui offre de nombreux avantages pour le développement de sites web. Tout d'abord, sa simplicité d'utilisation facilite la tâche des développeurs. Les fonctionnalités sont bien organisées et documentées, ce qui permet une prise en main rapide.

En outre, Django est également très efficace en termes d'organisation. Il offre une structure de projet claire et bien définie, avec un système de modèles et de vues qui permettent une séparation nette des différentes couches de l'application. Cela facilite la maintenance et l'évolution du site web à long terme.

Enfin, Django est très extensible et modulaire, ce qui permet de développer des applications web évolutives et personnalisées en fonction des besoins de l'entreprise. Il dispose également d'une grande communauté de développeurs et d'utilisateurs actifs qui contribuent constamment à son développement.

En somme, mon choix de travailler avec Django a été motivé par ses nombreux avantages, notamment sa simplicité, son organisation et sa modularité, ainsi que par la grande communauté active qui l'entoure. Nous sommes convaincus que cela permettra de créer un site web performant et facile à maintenir à long terme.

Mais travailler avec Django n'est pas une chose simple dès le départ, j'ai passé pas mal de temps sur des forums ainsi que la documentation pour comprendre son fonctionnement (et je suis toujours entrain d'apprendre des nuances très importantes de ce framework). Finalement, l'un des principaux défis que j'ai rencontrés en travaillant avec Django était de comprendre sa logique de modèles et son organisation assez stricte, notamment en ce qui concerne la dénomination des éléments statiques tels que les images, les fichiers CSS et JavaScript, etc.

Cela m'a demandé un effort supplémentaire pour comprendre la structure de l'application et la façon dont les différents composants étaient interconnectés. Toutefois, cela m'a permis de développer mes compétences en matière de développement web et de mieux comprendre la structure sous-jacente des applications web modernes.

En fin de compte, malgré les défis initiaux que j'ai rencontrés en travaillant avec Django, j'ai rapidement pu découvrir de nouvelles techniques de développement web. Les concepts de modèle et de vue de Django m'ont permis de mieux comprendre la structure des applications web modernes, tout en me permettant de développer des fonctionnalités sophistiquées de manière plus efficace.

En travaillant sur le site web, j'ai pu développer mes compétences en matière de développement de logiciels, de résolution de problèmes et de gestion de projet. Cela m'a également permis de renforcer mes compétences en matière de communication et de collaboration avec mes collègues, ce qui s'est révélé essentiel pour assurer la réussite du projet.

Bien que le site web ne soit pas encore terminé, j'ai été en mesure de le développer suffisamment pour le présenter lors de notre première soutenance. J'ai hâte de présenter sa version finale une fois que le projet sera terminé et de montrer les progrès réalisées.

### 3 Diverses retards

Lors de notre avancée dans ce projet, nous nous sommes rendus compte que certaines choses que nous pensions prioritaires ne l'étaient pas tant que ça finalement. Ainsi, si nous avons pris du retard sur les entités et les items, c'est parce que nous avons essayé de mieux penser le monde de notre jeu. Où placer cet item ? Est-il réellement obligatoire ? Ces questions nous ont ralenti, mais elles nous ont permis d'avoir un monde plus cohérent, plus agréable. Ces questions se posaient aussi pour les entités, bien évidemment.

Un autre retard est présent, celui de la création de niveaux en duo afin d'accueillir le multijoueur. En fait, ces derniers sont encore en cours de réflexion, mais les idées fusent. Les mini-jeux de Denis seraient un parfait support pour le multijoueur, mais l'envie de créer des niveaux spéciaux est aussi très présente. Ce qui est sûr, c'est que nos niveaux en solo étaient en fait une priorité, car ils nous servent de bases pour construire la suite. Voilà ce qui explique nos éventuels retards vis-à-vis du cahier des charges. Bien sûr, ces derniers seront rattrapés lors de notre prochaine soutenance.

### 4 Conclusion

Ainsi, ce projet part sur de très bonnes bases. En effet, même si nous avons eu des moments de doute, quelques divergences d'opinions, nous avons réussi à nous entendre et à surmonter les problèmes qui survenaient. Cependant, il reste encore énormément de choses à améliorer, et surtout le multijoueur à implémenter, mais si nous continuons sur cette voie, cela devrait bien se passer.

