

S E N S H A  
DOSSIER-PROJET

Jonas Mehtali & Guillaume Reitzer



## TABLE DES MATIÈRES

<b>TABLE DES MATIÈRES</b>	<b>3</b>
<b>INTRODUCTION</b>	<b>4</b>
<b>PARTIE I : LE PROJET ou ANALYSE DU BESOIN</b>	<b>5</b>
1. Contexte de définition du projet	5
2. Visée du produit	5
3. Proposition de produit	5
4. Evaluation des moyens	6
5. Démarches envisagées	6
7. Résultats anticipés	6
<b>PARTIE II : LE PROGRAMME</b>	<b>7</b>
1. Le développement	7
2. Les difficultés rencontrées	7
3. Fonctionnement simplifié	8
4. Mode d'emploi	9
5. Les problèmes non résolus	12
<b>PARTIE III : MES RÉALISATIONS</b>	<b>12</b>
1. Phase test	12
2. Le coeur du programme ou la classe “Virtuals”	12
3. Vecteurs, Trigonométrie & Distances	14
4. Base de données SQLite	15
5. Création visuelle	15
<b>PARTIE IV : TRAVAIL EN ÉQUIPE</b>	<b>17</b>
1. La répartition des tâches	17
2. Les difficultés du travail d'équipe	17
3. Les solutions	17
4. Les succès du travail en groupe	17
<b>CONCLUSION</b>	<b>18</b>
<b>ANNEXES</b>	<b>19</b>

## INTRODUCTION

“戦車” (**Sensha**) en japonais signifie “char d'assaut”, un véhicule automobile blindé. Utilisé pour la première fois au 20eme siècle, le char de combat est une invention récente qui a su se démarquer. Symbole de pouvoir ou de stabilité, le char d'assaut a fini par faire son apparition dans les bandes dessinées, le cinéma, la télévision ou encore dans les jeux vidéos.

Dans les jeux de stratégie tel “**Arma**”, “**Battlefield**” ou “**Supreme Commander**” les développeurs ont toujours intégré des chars qui ont la particularité d'être lents, encombrants et très coûteux mais qui représentent une force de frappe et un avantage stratégique majeur s'ils sont correctement placés et utilisés sur le champ de bataille.

C'est par ces placements et investissements stratégiques que nous avons été inspirés lors de la création de notre programme **Sensha**.

**Sensha** est un jeu en temps réel qui permet au joueur de devenir un grand stratège des champs de batailles modernes. Le joueur est amené à prendre des décisions difficiles, à investir des ressources limitées et à placer différents chars de manière réfléchie pour s'assurer une victoire décisive.

## PARTIE I : LE PROJET ou ANALYSE DU BESOIN

### 1. Contexte de définition du projet

Il nous a été demandé de créer un programme en langage de programmation Python (éventuellement avec des bibliothèques) pour l'épreuve Informatique et Sciences Numérique du Baccalauréat 2019. Ce projet s'effectue en équipe de deux ou trois étudiants de terminale qui devront le présenter devant un jury.

Le projet doit se limiter à une visualisation 2D et il faut veiller à ce que le projet ne soit pas trop ambitieux et ne prenne trop de temps à être réalisé.

Sachant que le programme sera présenté devant un jury, il est impératif qu'il soit compris par chaque membre du groupe.

Le programme fini devra être fonctionnel et ne comportera pas ou peu de bugs.

### 2. Visée du produit

Le but de ce projet est avant tout de nous permettre en tant que jeunes développeurs de découvrir les différentes étapes de la création d'un programme en Python et d'être confronté à des problèmes formateurs.

Nous avons décidé que notre produit devait amuser le client grâce à un jeu qui le place dans un mode imaginaire et immersif dont il est l'auteur. Nous voulons que notre produit amène le joueur à réfléchir à ses actions car toutes décisions prisent dans **Sensha** peuvent avoir des répercussions positives ou négatives sur la progression du jeu.

Pour finir, notre produit doit se démarquer des autres par son contenu original, créé par l'équipe en charge du projet. Cela signifie qu'aucun violation de la propriété intellectuelle ne sera commise.

### 3. Proposition de produit

Tout projet commence par une idée de départ. Notre idée de départ était de créer un "Brick Breaker", mais ce projet ne semblait pas assez ambitieux, c'est pourquoi j'ai proposé de développer un jeu de stratégie dans l'esprit des jeux de notre enfance tel "**Castle Attack**", "**Heavy Mach Defense**" ou bien encore "**Age of war**". Ce jeu doit à la fois lier des éléments de stratégie et de gestion dans le style des grands jeux RTS (Real Time Strategy).

Le sujet de notre jeu est le char de combat, autour duquel tourne toute la question stratégique.

## 4. Evaluation des moyens

Pour pouvoir progresser efficacement, tout d'abord il nous a fallu évoluer les moyens à notre disposition. Le mélomane de notre équipe, Guillaume Reitzer a produit les différents bruitages et thèmes utilisés dans notre projet. Sa maîtrise du langage de programmation Python a permis au projet d'avancer avec plus de facilités.

De mon côté, j'ai déjà conçu de petits programmes informatiques et codé en langage de programmation varié. Je m'étais avancé pendant les vacances scolaires pour me familiariser au langage Python dans le développement d'applications.

De plus, ayant une passion pour l'art visuel et digital, je me suis occupé du développement de l'UI (User Interface) et de la création des assets de notre projet.

Cette évaluation n'a pas pour autant défini nos rôles car il s'est avéré que dans une équipe à effectif réduit, les rôles sont flexibles. Nous avons donc tous été amenés à travailler dans chacun des domaines.

## 5. Démarches envisagées

Avant de développer notre programme, il nous a fallu établir une **démarche**.

Dans notre démarche, nous avons tout d'abord prévu de conceptualiser notre projet sur papier pour proposer différentes idées et versions.

Une fois un concept retenu, nous créeront un prototype pour tester la viabilité de nos idées. Lorsque le prototype sera dans sa phase finale, nous utiliserons ce que nous avons appris pendant la phase "test" pour créer notre projet final.

Le programme terminé, nous reprendrons le code pour l'optimiser et le rendre plus lisible tout en répertoriant les classes et fonctions qu'il contient.

## 7. Résultats anticipés

Le programme terminé doit tout d'abord être adapté à tous, immersif et visuellement plaisant.

Le jeu doit être riche en contenu ce qui rallongera sa durée de vie. De plus, le joueur pourra sauvegarder sa progression dans une base de données.

Afin de vivre la meilleure expérience virtuelle, notre programme doit être performant, adaptable et ne contenir aucun bug.

Pour finir, le programme pourra être déployé sur toutes les plateformes Windows avec une version exécutable.

## PARTIE II : LE PROGRAMME

### 1. Le développement

Comme nous l'avions prévu dans la démarche, nous avons tout d'abord créé un prototype de notre programme. Cette phase était la plus longue de toutes ainsi que la plus importante car nous avons testé différentes théories. La progression était longue mais les découvertes étaient majeurs. Par exemple, c'est à ce moment que j'ai développé la classe <>Virtuals>>, les IA, le système de collisions ou la base de données.

Cette phase était composée de trois cycles répétitifs. Dans le premier nous avons transposé nos idées en code, dans la deuxième nous avons adapté notre développement selon les résultats obtenus et enfin, lorsque les résultats étaient positifs, nous avons intégré définitivement le code dans le programme test.

Il m'est arrivé de réécrire des sections entières du programme suite à une découverte importante.

Une fois que le programme test avait atteint un état stable, il ne nous restait plus qu'à réutiliser ce que nous avions appris pour créer le programme final.

La phase de développement final était elle aussi sujette à de multiples changements permettant de simplifier des procédés qui jusque-là demandaient une puissance de calcul importante.

De plus nous avons demandé à des volontaires de tester notre jeu pour nous donner leurs avis. Nous les avons pris en compte pour changer certains aspects du programme, en particulier l'interface utilisateur pour la rendre plus compréhensible et esthétique.

### 2. Les difficultés rencontrées

Nous avons rencontré des difficultés à tous les niveaux du développement de notre programme.

La première était de **structurer le code** pour éviter de s'y perdre et pour ne pas faire d'erreurs au fur et à mesure du codage du programme.

Ensuite, le code devait être **compris** par tous les membres du groupe, nous avons donc passé beaucoup de temps à revoir le code qui n'était pas compris.

Enfin, le projet nous a demandé beaucoup de temps et d'investissement mais heureusement nous avions fini le programme bien avant la date butoire.

### 3. Fonctionnement simplifié

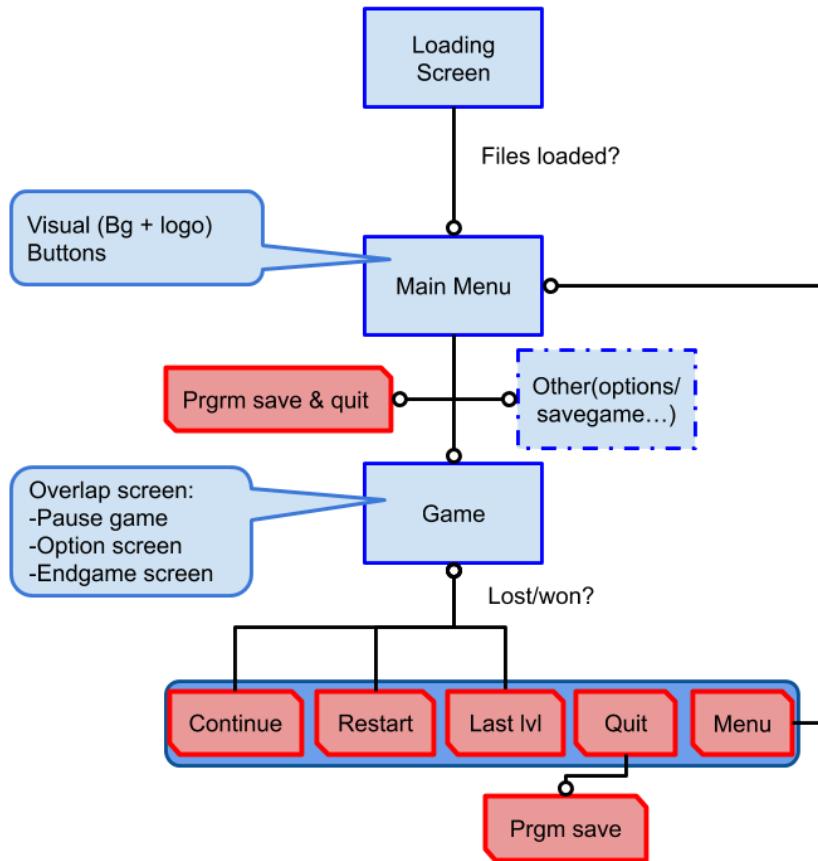
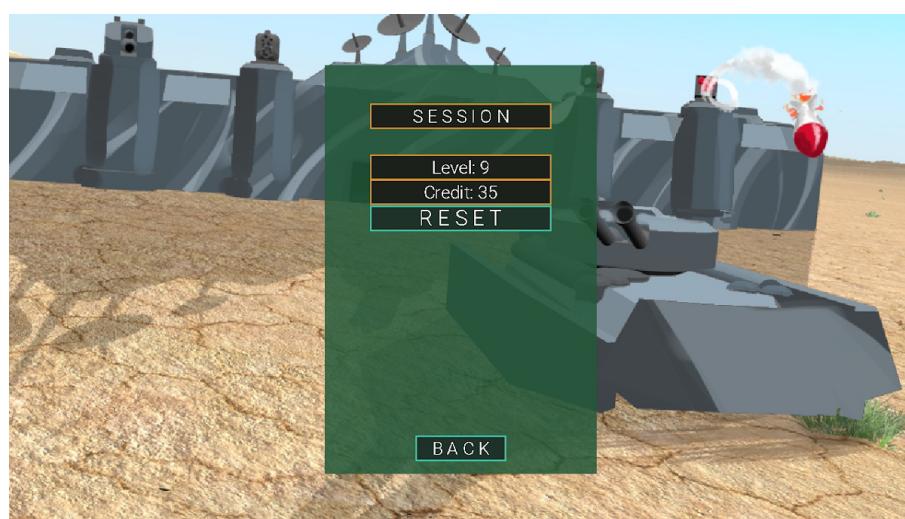
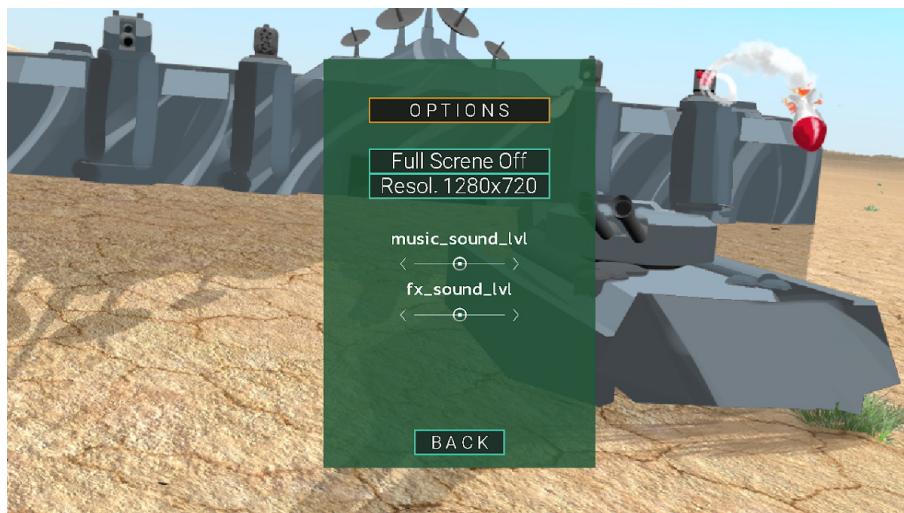


Schéma fonctionnel du programme

## 4. Mode d'emploi





**LE MENU AMELIORATIONS**

Le menu des améliorations vous permet de débloquer du contenu (véhicules, bonus, emplacements, etc...)



Amélioration débloquée



Amélioration non débloquée

**BACK**

Bouton BACK pour retourner au jeu

**LE MENU DE FIN DE JEU**

+250 Cr  
Bonus de victoire

**NEXT >>>**

Bouton NEXT pour jouer le prochain niveau

**RESTART**

Bouton RESTART pour recommencer le niveau

**<<< LAST**

Bouton LAST pour jouer le niveau précédent

**MENU**

Bouton MENU pour retourner au menu principal

**QUIT**

Bouton QUIT pour quitter le jeu

## 5. Les problèmes non résolus

Bien que nous ayons investi beaucoup d'efforts pour rendre le programme aussi performant que possible, des problèmes subsistent.

Par exemple, les véhicules sont dotés d'une **IA rudimentaire** qui peine à contourner les obstacles quand elle est coincée entre un obstacle et le bord de la carte. Cela est dû au fait que l'IA prend en compte l'obstacle mais pas le bord.

Ou encore un **bug mineur** qui se produit rarement mais qui a été aperçu quand la résolution est changée dans le jeu. Il s'agit d'une téléportation de quelques pixels des bases. Ce problème est le résultat d'un arrondissement des valeurs lors du changement de résolution.

Mais grâce à la coopération de testeurs volontaires nous avons pu éliminer la quasi totalité des problèmes.

Si le temps ne nous était pas compté, nous aurions élargi le contenu du jeu (mettre différents niveaux, améliorer les véhicules...).

# PARTIE III : MES RÉALISATIONS

## 1. Phase test

Avant de débuter le développement de notre projet, nous savions qu'il nous fallait tester la fiabilité de nos idées. C'est pourquoi j'ai proposé de développer une **version d'essai** de notre programme pendant les vacances d'hiver.

Tout d'abord il a fallu se familiariser avec le langage Python ainsi qu'avec la librairie Pygame. J'ai tout d'abord visionné des vidéos de programmation en adaptant les démarches dans mon prototype.

Une fois les bases fondamentales comprises, il m'a fallu trouver comment transformer mes idées de projet en code valable.

Une **liste des éléments découverts** pendant la phase test est fournie dans l'annexe "**patch notes**". Patch Notes a été mon journal de développement où j'ai écrit ce qui était à faire tout en gardant une trace de ce qui a été fait.

## 2. Le coeur du programme ou la classe "Virtuals"

Pendant le développement, nous nous sommes très vite aperçus que notre programme allait atteindre un **niveau de complexité critique** si nous continuions à écrire tout le code à la main ("hard coding"). C'est à dire que nous avons été contraints d'écrire des lignes de code à chaque fois que nous voulions nommer un élément.

Au départ nous avons utilisé la classe “**groupe**” de Pygame. Pour simplifier, cette classe nous a permis de mettre une classe en mémoire et de la lire à chaque boucle.

Mais je n'étais pas satisfait car cette classe nous restreignait et était uniquement conçue pour les classes contenant une image.

Alors j'ai décidé de créer ma propre classe de regroupement permettant de simplifier la gestion des classes. Je l'ai nommée “**Virtuals**”.

J'ai choisi ce nom car la classe **Virtuals** permet de “virtualiser” une classe en la sauvegardant dans une bibliothèque. Cette classe a été créée avec différentes fonctionnalités comme **.add()** qui permet d'ajouter une classe ou encore **.update()** qui lit selon un ordre défini par les “layers” chaque fonction **.update()** des classes ayant été ajoutées à **Virtuals**.

**Virtuals** étant un outil que j'ai développé, il m'a été possible de l'adapter aux besoins de notre programme.

Cette classe est devenue le fondement de notre programme et nous a permis de créer un programme plus complexe.

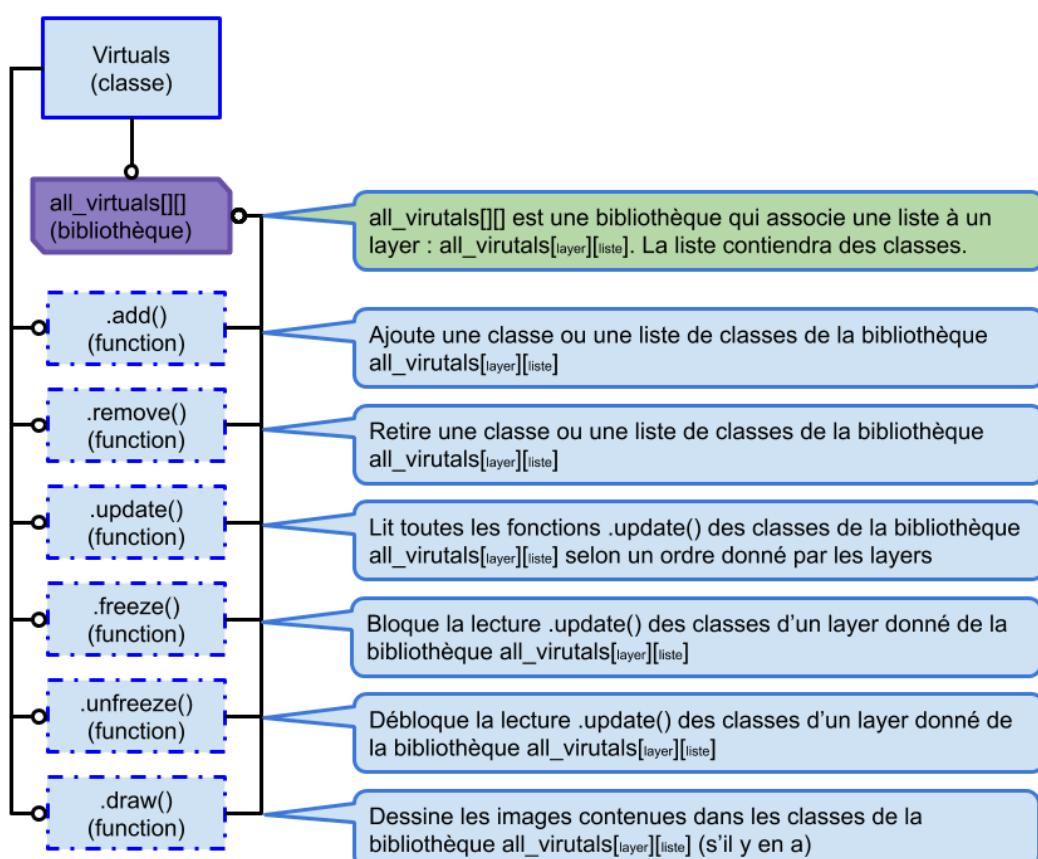


Schéma fonctionnel de la classe **Virtuals**

### 3. Vecteurs, Trigonométrie & Distances

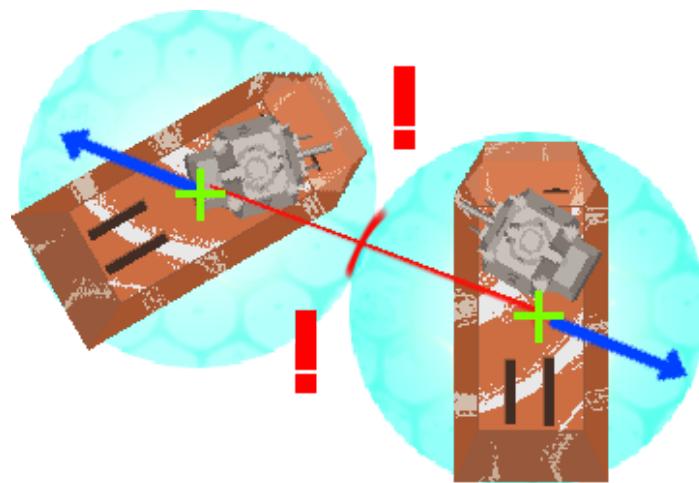
Pour calculer des distances, des mouvements et des collisions il a fallu utiliser mes connaissances mathématiques et physiques.

Tout objet dessiné sur l'écran de notre programme possède des coordonnées vectorielles 2D qui peuvent changer à chaque boucle dans le cas des véhicules.

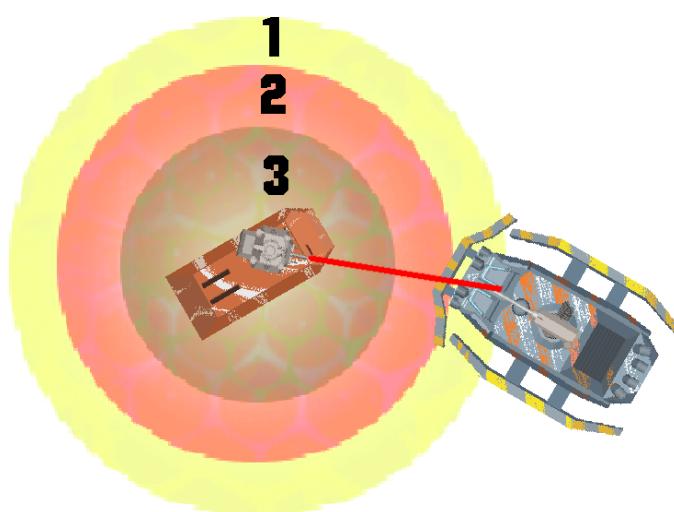
Pour pouvoir permettre des interactions entre deux objets il faut leur attribuer un **rayon** qui englobe leurs dimensions et il faut connaître la **distance qui sépare leur centre** ainsi que **l'angle de la droite reliant ces deux points** par rapport à l'**axe des abscisses**.

Connaître **l'angle** signifie qu'il nous est possible d'appliquer une **force directionnelle** si la **distance** entre les deux points est inférieure à la **somme de leur rayon**.

Ainsi il est possible de simuler des interactions physiques.



De manière analogue, il est possible d'utiliser ces calculs lorsqu'un objet doit en détecter un autre. On lui attribuera un **rayon de détection**.



## 4. Base de données SQLite

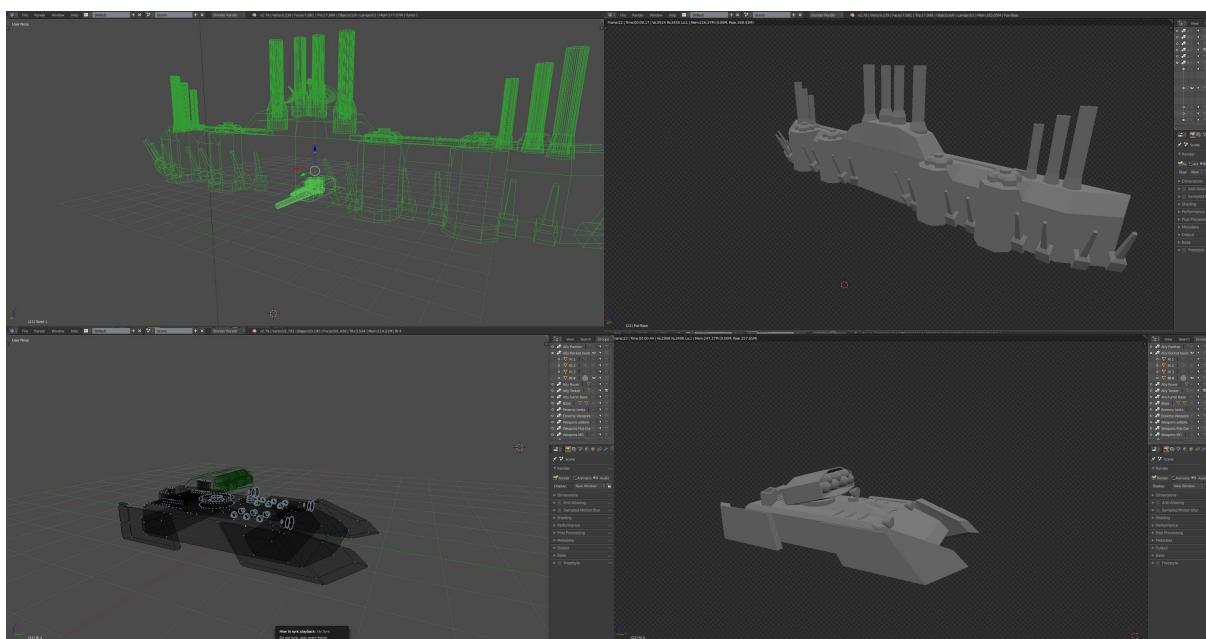
Pour éviter que le joueur soit frustré après avoir accidentellement quitté le jeu, il nous faut sauvegarder son avancement ainsi que les autres paramètres de sa partie. C'est pourquoi j'ai décidé d'utiliser la bibliothèque **SQLite** qui permet de créer un fichier.db dans lequel on peut sauvegarder et retirer des données.

Le programme n'a besoin d'appeler la base de données que deux fois. Lors du lancement du programme les données sont copiées vers une bibliothèque qui associe un nom à chaque donnée. Puis lorsque le programme est "terminé", les données de la bibliothèque sont à nouveau mises en mémoire. Ce nombre réduit d'appels libère de la puissance de calcul pour d'autres opérations, ce qui augmente la performance générale.

## 5. Création visuelle

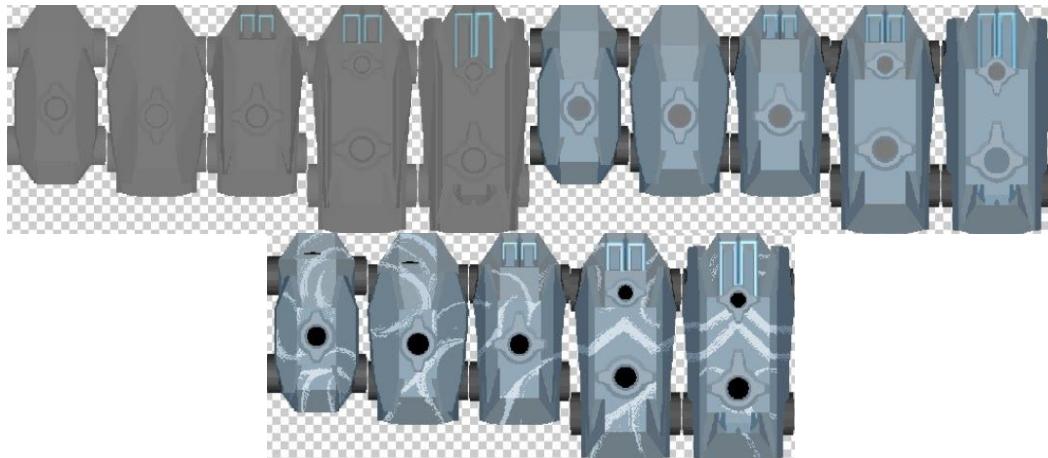
Pour communiquer avec l'utilisateur, nous avons choisi la représentation visuelle. Nous voulions que notre projet soit original, c'est pourquoi j'ai créé toutes les images, animations et effets visuels utilisés dans notre jeu. Pour se faire, j'ai utilisé différentes applications numériques.

La première est **Blender**, un logiciel de modélisation 3D avec lequel j'ai modélisé les véhicules, tourelles et bases de notre jeu.



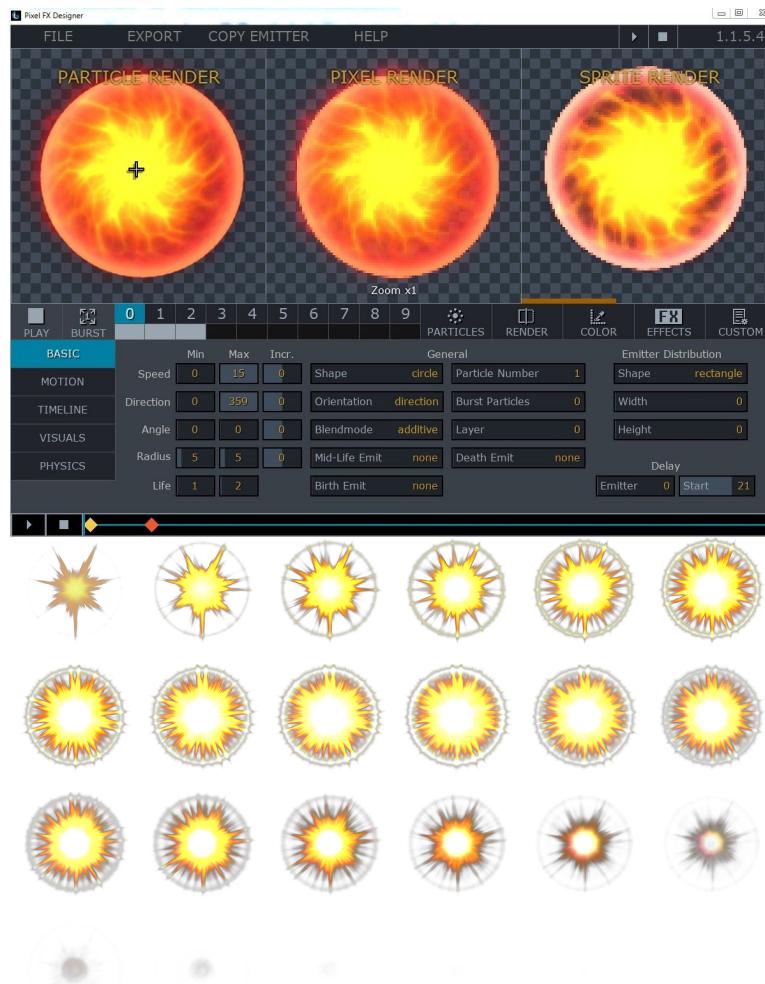
(Captures de Blender)

Puis j'ai exporté les modèles et j'y ai appliqué des textures sur **Photoshop**.



(Captures de Photoshop)

Pour les différentes effets et animations j'ai décidé de me procurer un petit logiciel appelé **Pixel FX Designer**. Cette application peut à la fois créer des spritesheets et des sprites séparés.



(Captures de Pixel FX Designer)

## PARTIE IV : TRAVAIL EN ÉQUIPE

### 1. La répartition des tâches

Jonas	Guillaume
<ul style="list-style-type: none"> <li>- Conceptualisation</li> <li>- Code (virtuals, interaction, IA generale, véhicules, collisions)</li> <li>- Contrôle du code avant intégration</li> <li>- Créations visuelles</li> <li>- Répertorisation du programme</li> </ul>	<ul style="list-style-type: none"> <li>- Conceptualisation</li> <li>- Code (barre de son, IA base, music player)</li> <li>- Réalisations et compositions audio</li> <li>- Répertorisation du programme</li> </ul>

### 2. Les difficultés du travail d'équipe

Le travail en équipe est bénéfique mais demande une participation égale de tous ses membres. Malheureusement cette participation égale n'est pas toujours garantie car les individus ont différentes expériences et connaissances.

De plus, les membres d'un groupe peuvent avoir des opinions différentes ce qui peut bloquer entièrement la progression d'un projet.

Lors de la programmation, nous avons passé beaucoup de temps à revoir le code car il n'était pas entièrement compris.

### 3. Les solutions

Sachant que le code d'un autre n'est pas toujours simple à comprendre, nous avons décidé d'organiser des réunions fréquentes afin d'expliquer notre code.

Bien que cela nous a pris du temps, les effets étaient bénéfiques, et a favorisé par exemple, une meilleure productivité dans le travail d'équipe.

De plus nous nous sommes vu fréquemment pendant la phase de développement pour éviter toute divergence de programmation.

### 4. Les succès du travail en groupe

Les différences dans notre groupe se sont avérées être une force plutôt qu'un désavantage. Guillaume Reitzer étant un passionné de musique, a réalisé la production audio de notre projet. Personnellement, ayant une affinité pour le visuel j'ai créé toutes les interfaces et images utilisées dans le programme (sauf les images de planètes libre de droit). Cette diversité nous a permis de travailler sur plusieurs aspects du projet à la fois.

Le travail en équipe, nous a permis d'optimiser notre temps.

## CONCLUSION

Ainsi, ce projet a été un triple succès à mes yeux.

Premièrement, créer ce programme a été une expérience formatrice. J'ai été confronté à des problèmes dans tous les domaines du développement et je les ai résolus. L'expérience que j'en ai tiré m'a permis d'éviter d'autres problèmes par la suite. De plus, j'ai pu m'approprier le langage Python et comprendre les différentes étapes du développement d'une application. En outre, j'ai appris à gérer un travail d'équipe. Nous avons tout d'abord dû établir un plan de travail avant de nous répartir les tâches. Puis nous expliquer l'un à l'autre ce que nous avions fait. Le travail en équipe n'a pas été simple mais nécessaire à la réussite de notre projet.

Pour finir, l'an prochain je ferai mes études informatique dans le domaine de la représentation visuelle et ce projet m'a permis de consolider ma décision d'orientation.

## ANNEXES

### Patch notes

#### Programme version papier:

- main.py
- display.py
- game.py
- hangar.py
- sprites.py
- ai\_system.py
- settings.py
- file\_loader.py