Shoot me up !

Une image contenant noir et blanc, éclairage, bâtiment, léger

Description générée automatiquement

Bastien Segalen – MID2B

ETML - Vennes

80p

Curchod – Mveng - Melly

Table des matières

[1 Spécifications 3](#_Toc181175283)

[1.1 Titre 3](#_Toc181175284)

[1.2 Description 3](#_Toc181175285)

[2 Planification Initiale 3](#_Toc181175286)

[3 Réalisation 3](#_Toc181175287)

[3.1 Dossier de Réalisation 3](#_Toc181175288)

[3.1.1 Module 106 (DB) : 3](#_Toc181175289)

[3.1.2 Module 320 (POO) : 4](#_Toc181175290)

[3.1.3 Module 322 (UX) : 4](#_Toc181175291)

[4 Tests 7](#_Toc181175292)

[4.1 Dossier des tests 7](#_Toc181175293)

[5 Conclusion 7](#_Toc181175294)

[5.1 Bilan des fonctionnalités demandées 7](#_Toc181175295)

[5.2 Bilan de la planification 7](#_Toc181175296)

[5.3 Bilan personnel 7](#_Toc181175297)

[6 Divers 7](#_Toc181175298)

[6.1 Journal de travail 7](#_Toc181175299)

[6.2 Webographie 7](#_Toc181175300)

[6.3 IA 7](#_Toc181175301)

[7 Annexes 7](#_Toc181175302)

# Spécifications

## Titre

Shoot me up !

## Description

Concevoir un jeu 2D modulaire de tir à la troisième personne et en réaliser une partie.

# Planification Initiale

<T-106-BastienSegalen-jdt-planif-DB.xlsm>

<T-320-BastienSegalen-jdt-planif-POO.xlsm>

<T-322-BastienSegalen-jdt-planif-UX.xlsm>

# Réalisation

## Dossier de Réalisation

### Module 106 (DB) :

Auto-évaluation DB : <auto-evaluation-DB.xlsx>

MCD et MLD: [MCD-MLD\mcd-shootmeup.loo](MCD-MLD/mcd-shootmeup.loo)

Fichier de création des données : [MCD-MLD\shootmeup-donnees.sql](MCD-MLD/shootmeup-donnees.sql)

Fichier de création de la base de données : [MCD-MLD\db\_shootmeup.sql](MCD-MLD/db_shootmeup.sql)

Utilisateurs :

Fichier de création des utilisateurs et des rôles :

[MCD-MLD\shootmeup\_create\_users.sql](MCD-MLD/shootmeup_create_users.sql)

Administrateurs : créer, lire, mettre à jour, supprimer toutes les tables.

Droits : ALL PRIVILEGES sur toutes les tables, GRANT OPTION

Gestionnaire du jeu : lire toutes les tables et mettre à jour les highscores.

Droits : SELECT sur toute les tables, INSERT sur les tables niveau (pour l’éditeur de niveau) et highscores.

INDEX :

1. Index sur les clés primaires, les clés étrangères et les champs « unique »
2. Plus rapide pour les requêtes, créer les index prend du temps (insertion de données)
3. Sur le champ « score » de la table « t\_highscores »

BACKUP/RESTORE :

Fichier dump de la base de données : [MCD-MLD\db\_shootmeup\_dump.sql](MCD-MLD/db_shootmeup_dump.sql)

1. Backup : Ouvrir un cmd à l’endroit où le dump doit être stocké, exécuter la commande :   
   docker exec -i id\_du\_container mysqldump -u root -proot db\_shootmeup > db\_shootmeup\_dump.sql
2. Restore : Une fois que la base de données et supprimé, se connecter a mysql et exécuter cette commande :   
   CREATE DATABASE db\_shootmeup ;  
   Ensuite, ouvrir un cmd à l’endroit où se trouve le dump, et exécuter cette commande :   
   docker exec -i id\_du\_container mysql -u root -proot db\_shootmeup < db\_shootmeup\_dump.sql

### Module 320 (POO) :

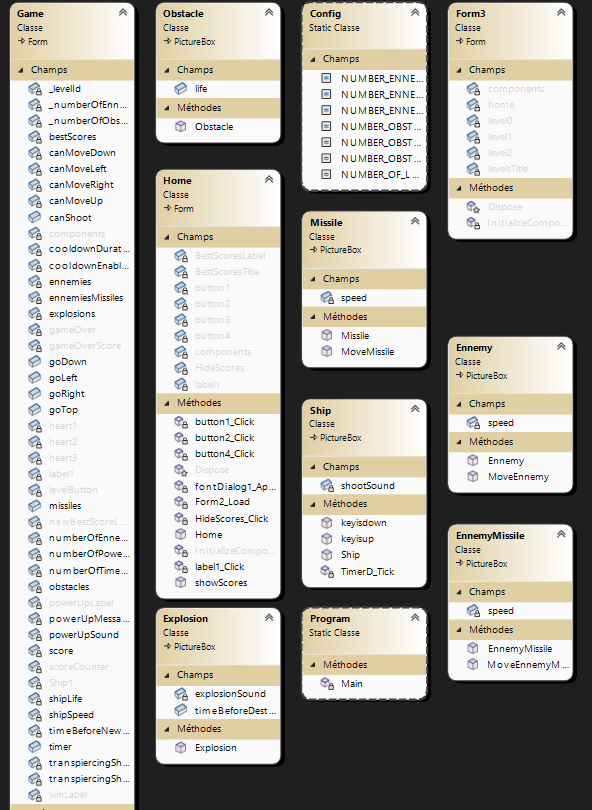
Description du shoot me up :

On peut bouger le vaisseau dans toutes les directions, les ennemis descendent en ligne et en tirant des lasers. Quand on tue les ennemis, on peut obtenir un power up, qui est soit un tir transperçant qui peut tuer plusieurs ennemis, soit un power up qui désactive le rechargement, on peut donc tirer très vite. Il y a des obstacles pour pouvoir se protéger des tirs ennemis.

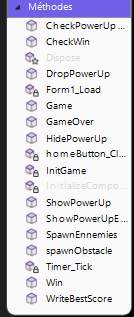
Niveaux : Il y a deux niveaux avec un nombre fix d’ennemis et un niveau “survie” qui est infini.

On peut tuer les ennemis avec nos tirs ou avec notre vaisseau, au cout d’une vie

Les obstacles peuvent être détruit par les tir adverse, par nos tirs ou quand un ennemi touche un obstacle.

Diagramme de classe : 

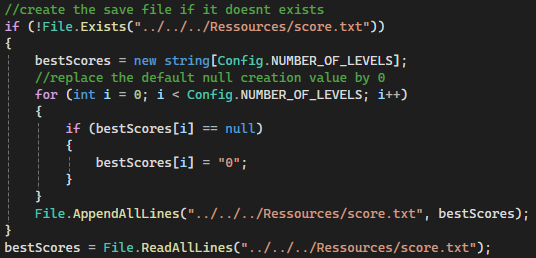
Méthode de Game :



TODO : docFX, test,

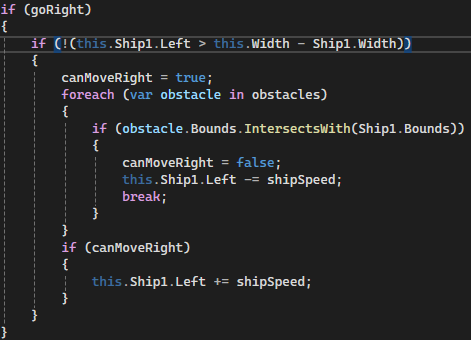
Détail d’implémentation spécifique :

Création d’un fichier de sauvegarde :



On regarde si le fichier existe déjà. S’il n’existe pas, on crée on tableau dont la taille est le nombre de niveau. Ensuite on remplit le tableau de zéro pour éviter des problèmes a la fin du niveau. Ensuite on écrit le tableau dans le fichier avec la méthode File.AppendAllLines(), qui créer aussi le fichier s’il n’existe pas. Et enfin on stocke les scores dans un tableau pour pouvoir comparer les scores plus tard.

Gestion des collisions entre le vaisseau et les obstacles :



Quand on presse sur D, goRight = true.

Si le vaisseau n’est pas en dehors, on met canMoveRight à true.

Ensuite on regarde si le vaisseau est en collision avec les obstacles.

S’il n’y a pas de collision canMoveRight sera true est le vaisseau pourra bouger.

S’il y a une collision, on met canMoveRight à false, on déplace le vaisseau pour qu’il n’y pas de collision et on sort de la boucle.

### Module 322 (UX) :

Personnas : [UX\personnas.txt](UX/personnas.txt)

Une image contenant texte, Visage humain, capture d’écran, personne

Description générée automatiquement

Une image contenant texte, Appareils électroniques, ordinateur, Visage humain

Description générée automatiquement

Une image contenant texte, Visage humain, personne, capture d’écran

Description générée automatiquement

Description du shoot me up :

Un mode “histoire” composée de plusieurs niveaux de +en + difficile, et un mode “survie” qui est infini, devenant plus dur + on survit

Si on gagne un niveau, on gagne de l’or qu’on pourra utiliser pour acheter des améliorations.

Éco conception : 115 bonne pratique :

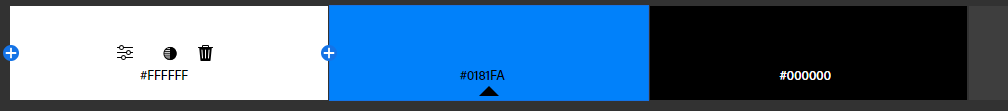
5 : Optimiser le parcours utilisateur :

Il y a peu de bouton, les interface sont simple et facile à comprendre.

Accessibilité :

Fonctionnalité d’accessibilité mise en place : contraste élevé pour les personnes malvoyantes.

Palette de couleur :



Élément original : Arbre de compétence :

Les joueurs peuvent acheter des améliorations pour leur vaisseau avec des pièces qu’ils obtiennent en gagnant des niveaux.

Pour acheter une amélioration on doit avoir Acheter la précédente, et les améliorations ne sont pas cumulative, par exemple si ont à acheter +1 vie et +2 vies, on aura deux vies supplémentaires et pas trois.

Conception :

Maquettes haute et basse fidélité : [UX\shoot-me-up.pdf](UX/shoot-me-up.pdf)

Prototype cliquable Figma : [UX\shoot-me-up.fig](UX/shoot-me-up.fig)

Choix effectués :

* Écran d’accueil : Il y a un titre et quatre boutons : Jouer, score, éditeur de niveau et paramètres. L’écran est simple et l’utilisateur comprendra facilement ce qu’il peut faire.
* Écran de scores : Il y a un tableau de score avec pour chaque ligne place, nom et score. Il y a un bouton pour changer de niveau et un pour revenir à l’accueil.
* Écran jouer : L’écran est simple, il y a seulement des boutons pour sélectionner le niveau qu’on veut jouer et un pour pour aller sur l’écran d’amélioration.
* Écran amélioration : Il y a l’argent qu’on a et les boutons pour acheter des compétences.
* Écran de paramètres : On peut activer le mode contraste élevé pour les personnes malvoyante et on peut changer de nom.
* Écran éditeur de niveau : Sur la gauche il y a la zone des paramètres, ou on peut changer les paramètres du vaisseau, des ennemis et des obstacles. La zone est divisée en plusieurs parties (joueur, obstacle, etc.) et on peut réduire les zones si on n’en a pas besoin. Si on a toutes les zones dépliées, le contenu sera trop grand pour être afficher et on pourra scroller pour accéder aux options qui sont en bas. La partie droite est la zone de jeu, ou on peut voir à quoi ressemblera le niveau et on peut aussi changer la position du vaisseau et des obstacles, et leur taille. En bas a gauche il y a deux boutons : tester le niveau et publier le niveau. Quand on clique sur publier le niveau, une fenêtre s’ouvre et on peut donner un nom a notre niveau et on doit sélectionner la difficulté du niveau. Il y a aussi un bouton pour revenir à l’accueil et un autre pour fermer la zone des options.

# Tests

## Dossier des tests

On dresse le bilan des tests effectués (qui, quand, avec quelles données…) sous forme de procédure. Lorsque cela est possible, fournir un tableau des tests effectués avec les résultats obtenus et les actions à entreprendre en conséquence (et une estimation de leur durée).

Si des tests prévus dans la stratégie n'ont pas pu être effectués :

Raison, décisions, etc.

Liste des bugs répertoriés avec la date de découverte et leur état :

Corrigé, date de correction, corrigé par, etc.

# Conclusion

## Bilan des fonctionnalités demandées

Il s’agit de reprendre point par point les fonctionnalités décrites dans les spécifications de départ et de définir si elles sont atteintes ou pas, et pourquoi.

Si ce n’est pas le cas, estimer en « % » ou en « temps supplémentaire » le travail qu’il reste à accomplir pour terminer le tout.

## Bilan de la planification

Distinguer et expliquer les tâches qui ont généré des retards ou de l'avance dans la gestion du projet. Indiquer les différences entre les planifications initiales et détaillées avec le journal de travail.

## Bilan personnel

Si c’était à refaire:

Qu’est-ce qu’il faudrait garder ? Les plus et les moins ?

Qu’est-ce qu’il faudrait gérer, réaliser ou traiter différemment ?

Qu’est que ce projet m’a appris ?

Suite à donner, améliorations souhaitables, …

Remerciements, signature, etc.

# Divers

## Journal de travail

<T-106-BastienSegalen-jdt-planif-DB.xlsm>

<T-320-BastienSegalen-jdt-planif-POO.xlsm>

<T-322-BastienSegalen-jdt-planif-UX.xlsm>

## Webographie

Références des sites Internet consultés durant le projet.

## IA

L’IA a été utilisé pour la réalisation des personnas. (IA utilisé : Copilot et ChatGPT)

ChatGPT a été utilisé pour générer des données pour la base de données.

# Annexes

Listing du code source (partiel ou, plus rarement complet)

Guide(s) d’utilisation et/ou guide de l’administrateur

Etat ou « dump » de la configuration des équipements (routeur, switch, robot, etc.).

Extraits de catalogue, documentation de fabricant, etc.