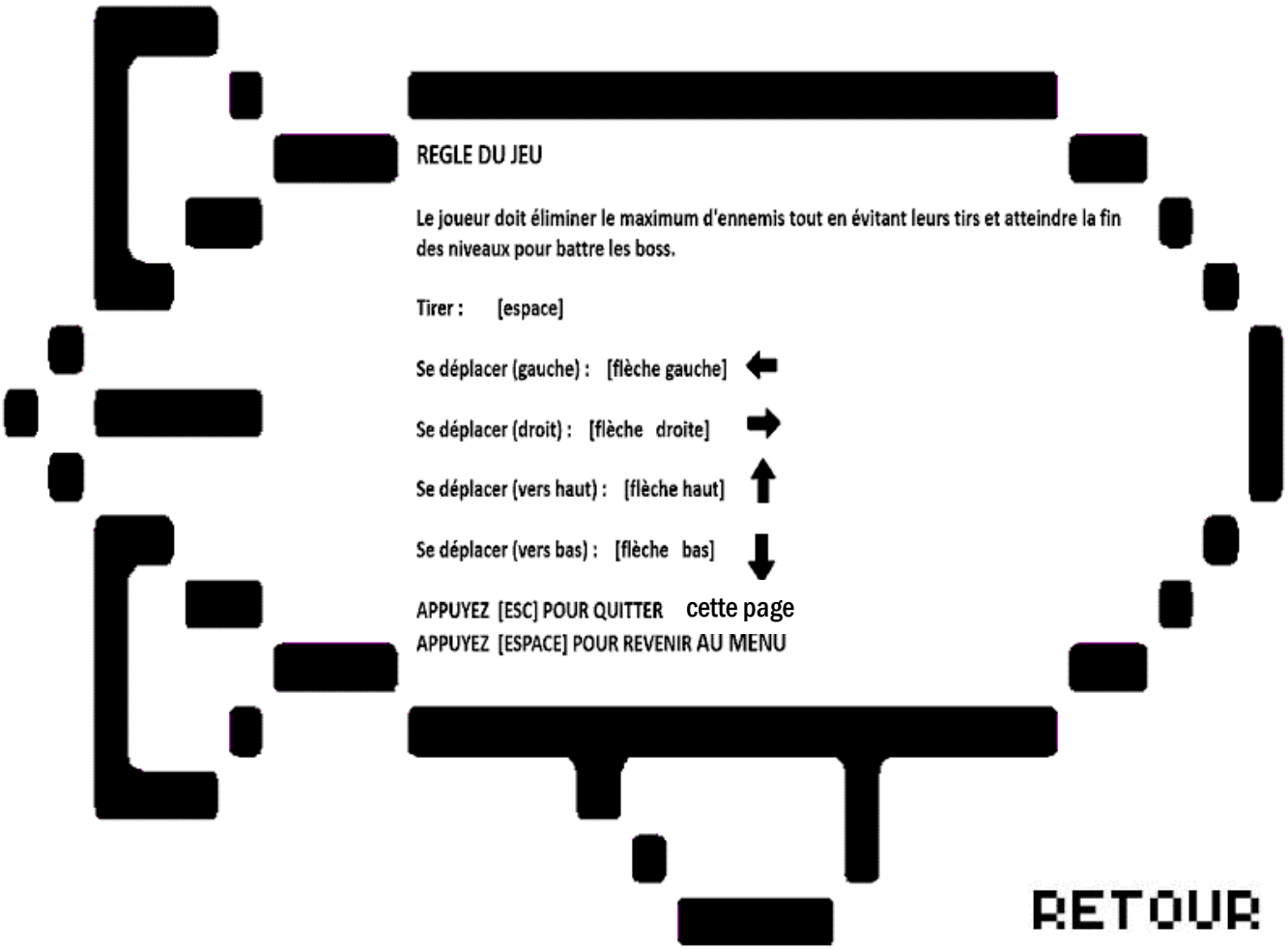
Etienne Corrège
Shi Hao YU
Matthieu Mouyart

Rapport de projet d'informatique

Sommaire :



Etienne Corrège
Shi Hao YU
Matthieu Mouyart

Nous sommes en 2021 et l'ECE est envahie par ses homologues écoles d'ingénieurs qui sont jaloux de l'ECE, et sont bien décidés à en finir avec nous. Heureusement que l'ECE et se ingénieurs en formations sont plus intelligents que cela et peuvent ensemble lutter contre l'assaillant, voici votre mission si vous l'acceptez.

Nous incarnons un dirigeable, le dirigeable grâce auquel, l'ECE saura survivre, malgré une vague d'invasions d'avions ennemis dirigés par leur charismatique chef, la montgolfière et le pilote fort bien connu de l'ECE. Notre but est de traverser les 3 niveaux successifs, et à la fin du dernier, de battre la montgolfière et son pilote redoutable afin d'éradiquer une bonne fois pour toutes l'invasion de ces faux ingénieurs jaloux de l'ECE.

Notre vaisseau est un dirigeable transformé qui possède des lances missiles pour attaquer les ennemis, il peut tirer ces missiles suivant une trajectoire horizontale, partant à une vitesse élevée pour pouvoir atteindre rapidement les ennemis avant qu'ils ne bougent de position.

Les ennemis quant à eux seront incarnés par de petits avions rouges qui tireront également des missiles rapides.

Notre dirigeable dispose de 200 points de vie, sachant que chaque missile lui retirera 10 points de vie, il faudra faire attention à rester en vie tout au long des niveaux.

Les ennemis quant à eux disposent de 20 points de vie et seront donc éliminés au bout de 2 tirs de missiles alliés.

Le boss final….

Explication de la dynamique du jeu

Lorsque le joueur lancera le programme principal, il sera affiché à l'écran un menu comportant 3 choix d'actions disponibles pour le joueur :

- jouer au jeu en choisissant parmi les 3 niveaux.

-revoir les règles du jeu.

-quitter le jeu

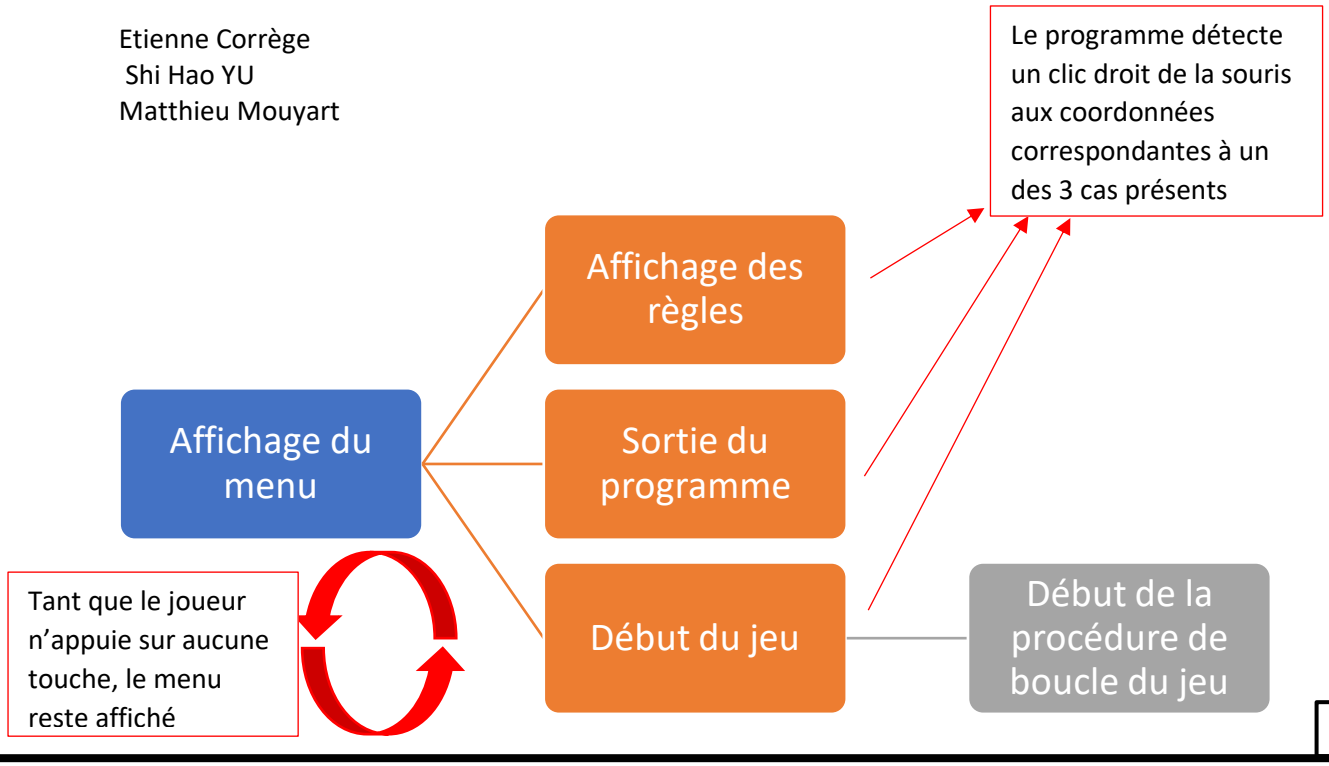
Tant que le joueur ne cliquera pas sur une de ces cases avec le clic gauche de la souris, ou qu'il n'appuie pas sur la touche « échap », il ne se passe rien.

Une fois que le joueur décide de jouer, alors le programme détecte une condition, s'il y a un point de sauvegarde activé. Si oui, alors le jeu reboucle à partir du niveau sauvegardé avec les changements pris en compte (nombre d'ennemis à faire apparaître, combien de temps il faudra encore faire défiler la carte…), et le joueur pourra reprendre là où il en était.

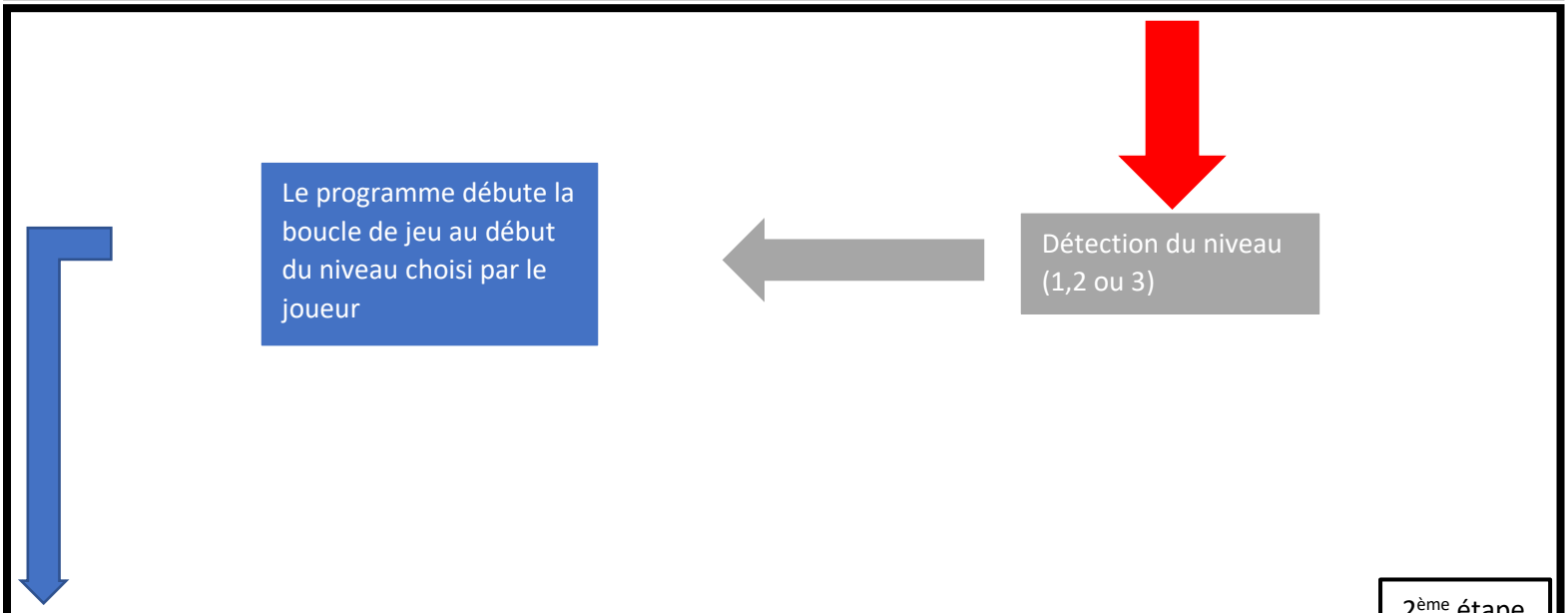
Mais, s'il n'y a pas de checkpoint activé alors le jeu boucle à partir du début du premier niveau avec toutes les ennemis a faire apparaître.

Exemples avec des schémas :

Etienne Corrège
Shi Hao YU
Matthieu Mouyart



1ère étape



2ème étape

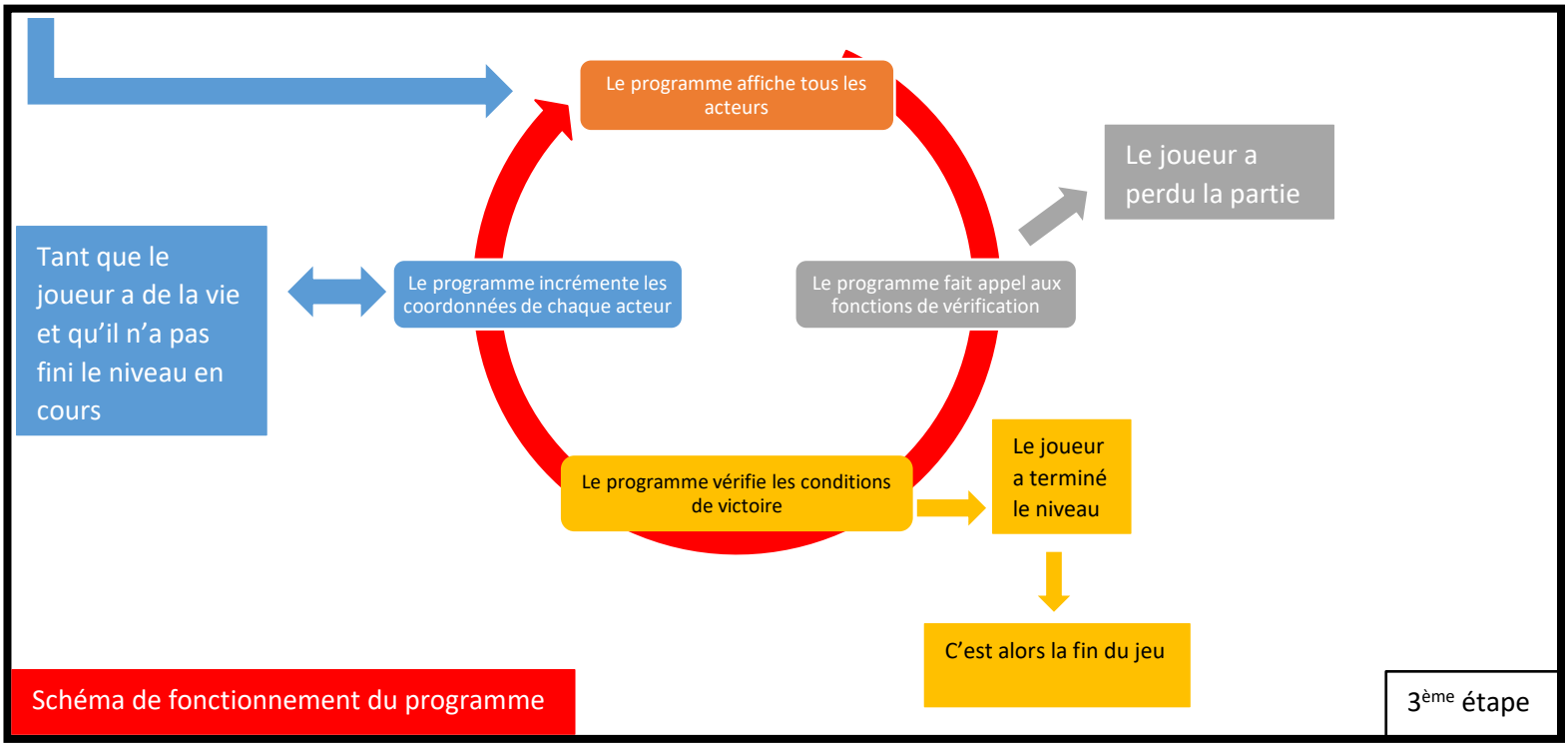


Schéma de fonctionnement du programme

3ème étape

Etienne Corrège
Shi Hao YU
Matthieu Mouyart
Lorsque le joueur décide de quitter le jeu (càd appuyer sur la touche « échap »), le programme affiche une fenêtre lui demandant la confirmation de son choix, et si celui-ci confirme le choix de quitter le jeu, alors le programme se ferme.

Le jeu évolue en forme de boucle infinie, qui continue tant que le joueur n'appuie pas sur la touche « échap ». A chaque tour de boucle, le programme affiche tous les acteurs nécessaires au tour, appelle les éventuels sous-programmes (vérification des tirs, tir…), vérifie les conditions de victoire du joueur, et passe à la boucle suivante. Il faut préciser que l'équivalent d'une boucle de jeu représente un laps de temps extrêmement court, dû à la rapidité du microprocesseur.

Sera également affiché à l'écran le score du joueur qui sera incrémenté à chaque fois que le joueur élimine des ennemis.

Il y a également sur la carte des bonus que le joueur pourra essayer de toucher lui permettant de faire plus de dégâts. En fonction d'un certain nombre de bonus de dégâts accumulés, le dirigeable va changer d'aspect (4 évolutions au total).

Le jeu évolue selon le concept suivant : notre dirigeable se déplace sur l'écran en bougeant (avant, arrière, droite, gauche) sur une carte qui défile automatiquement au fil du jeu. Le dirigeable sera toujours bloqué par les bords du jeu afin de ne pas disparaître de l'écran.

Il est également affiché sur une carte non présente à l'écran, représenté par un rectangle bleu de dimensions correspondant à une approximation de l'image du joueur. Cette carte est appelée la hitbox, elle permet de vérifier les tirs alliés/ennemis, et de bien gérer les dégâts de tir.

De même, chaque fois que le dirigeable tirera, il sera affiché à l'écran un missile, et sur la hitbox un rectangle de dimension correspondant à une approximation de l'image du missile. A chaque fois que le joueur voudra tirer, le programme va créer dans la mémoire un acteur missile agissant indépendamment des autres acteurs, qui sera à chaque fois déplacé sur l'écran et qui disparaîtra une fois qu'il sera sorti de l'écran.

Les ennemis de manière analogue sont affichés sur l'écran, et sur la hitbox par un rectangle rouge de dimensions correspondant à une approximation de l'image de l'ennemi qui évoluera sur la hitbox comme à l'écran. De même chaque fois qu'un ennemi tire, le missile est affiché sur l'écran et sur la hitbox par un rectangle rouge de dimensions correspondant à une approximation de l'image du missile.

La carte qui est affichée à l'écran est en réalité une image qui a été préalablement créée, et qui est à chaque fois affichée à l'écran en augmentant simplement les coordonnées en x (l'écran étant de plus petite longueur que l'image créée). Chaque fois que les coordonnées de l'image dépassent la taille de l'image, le programme affiche une partie de l'image (portion allant de la coordonnée en x de la gauche de l'écran jusqu'à la bordure de l'image), et une image de qui fera la taille manquante à afficher à l'écran. C'est une façon de générer la carte de façon infini sans jamais arriver au bout de l'image.

En revanche la hitbox est une image de la taille de l'écran que l'on ne fait pas défiler, sur laquelle on affiche les acteurs présents sur l'écran avec les mêmes coordonnées.

A chaque tour de boucle (boucle de jeu principale), le programme vérifiera les bordures de chaque acteur présent à l'écran sur la hitbox. Pour les missiles, le programme vérifiera si le rectangle représentant le missile rentre en collision avec un autre obstacle, quelque soit sa couleur, et, s'il détecte un obstacle, il disparaît de la hitbox et de l'écran.

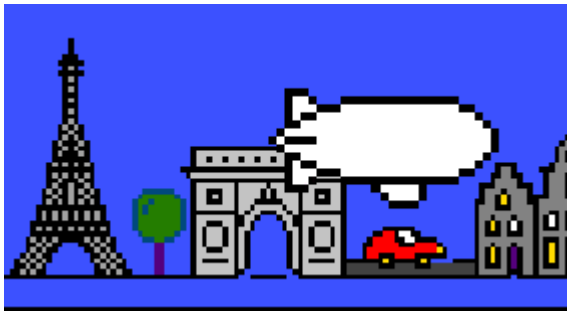
Etienne Corrège
Shi Hao YU
Matthieu Mouyart

En revanche pour le dirigeable, le programme vérifie les bordures du rectangle bleu sur la hitbox, et s'il détecte que celui-ci rentre en collision avec un rectangle rouge (missile ennemi ou ennemi), il lui retire 10 points de vie par bordure touchée.

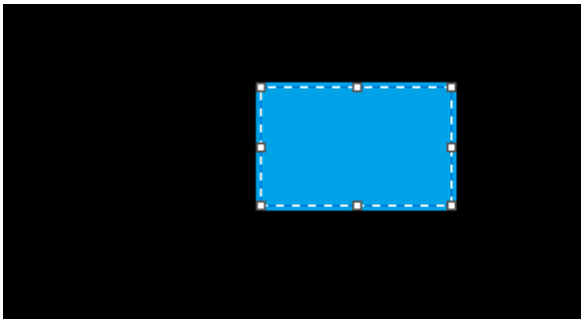
Et pour les ennemis, le programme vérifie les bordures du rectangle rouge sur la hitbox, si celui-ci détecte que le rectangle est rentré en collision avec un rectangle bleu (missile allié ou dirigeable), il retire des points de vie par bordure touché à cet ennemi.

Il faut préciser que sur la hitbox, chaque acteur est représenté par un rectangle dont les dimensions correspondent à l'approximation faite des bordures « réelles » de l'image de l'acteur, elles ne correspondent donc pas aux bordures réelles.

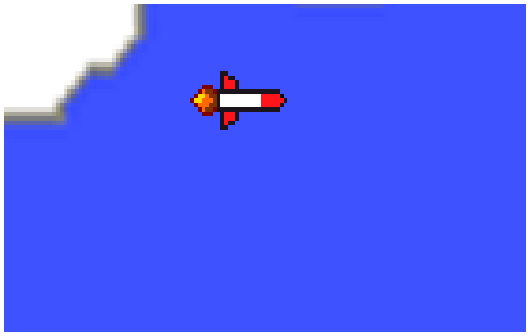
Exemples de détections d'affichage sur la hitbox



Affichage du dirigeable sur l'écran



Affichage du dirigeable sur la hitbox



Affichage d'un missile allié sur l'écran

Affichage d'un missile allié sur la hitbox

Etienne Corrège
Shi Hao YU
Matthieu Mouyart

Affichage d'un ennemi sur l'écran

Affichage d'un ennemi sur la hitbox

Affichage d'un missile ennemi sur l'écran

Affichage d'un missile ennemi sur la hitbox

Situation de détection d'un tir ennemi sur le dirigeable

Sur la hitbox

Situation de détection d'un tir allié sur un ennemi

Etienne Corrège
Shi Hao YU
Matthieu Mouyart
1er cas sur l'écran 2ème cas sur la hitbox

N.B. Les images illustrant la hitbox sont zoomées, donnant l'impression de taille plus grande que les images sur l'écran. Mais leur taille est bien la même que celle des images à l'écran.

Chaque acteur présent sur l'écran sera donc également affiché sur la hitbox, et fera appelle au sous-programme de détection de tir. Comme déjà expliqué précédemment, pour les missiles, si sur la hitbox il détecte une quelconque collision, il ne sera plus affiché à la boucle suivante (sur l'écran et la hitbox).

En revanche pour le dirigeable, à chaque fois, il fait appel au sous-programme détectant les tirs, et s'il détecte des collisions sur une bordure, il lui sera retiré à chaque bordure touché, 10 points de vie.

Si, à la fin de la vérification, le dirigeable n'a plus de vie, alors celui-ci n'est plus affiché à l'écran, et c'est la fin du jeu.

La dynamique est exactement la même pour les ennemis, et, si à la fin de la boucle de vérification, ils n'ont plus de vie, alors ils ne sont plus affichés à l'écran.

Pour chaque niveau, l'apparition se fait de manière aléatoire, sauf pour le boss qui lui sera présent dès le début du niveau.

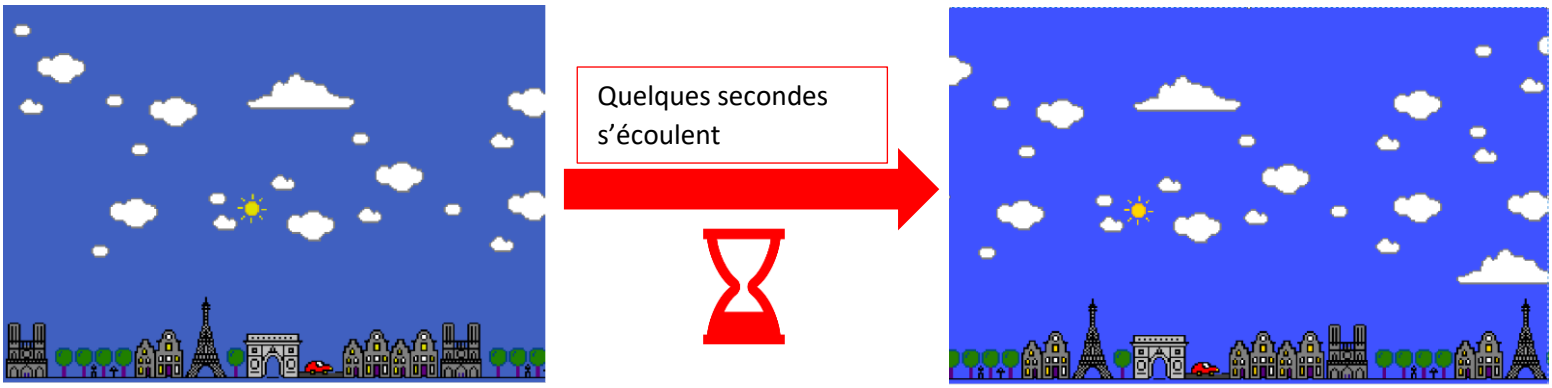
Pour chaque niveau, à partir d'un certain temps, le programme vérifie si le joueur est toujours en vie et s'il a éliminé tous les ennemis présents sur l'écran, et, si c'est le cas, alors le programme indique au joueur qu'il a terminé le niveau.

A chaque niveau, des ennemis sont affichés au fur et à mesure sur l'écran (qui défilent de la droite vers la gauche). Et est affiché également un « boss » qui lui se déplace en de manière aléatoire sans sortir de l'écran. Le joueur va donc de voir esquiver les ennemis et leurs missiles ainsi que le boss et ses missiles.

Dès lors que le joueur aura éliminé le boss et tous les autres ennemis, alors le joueur aura gagné la partie.

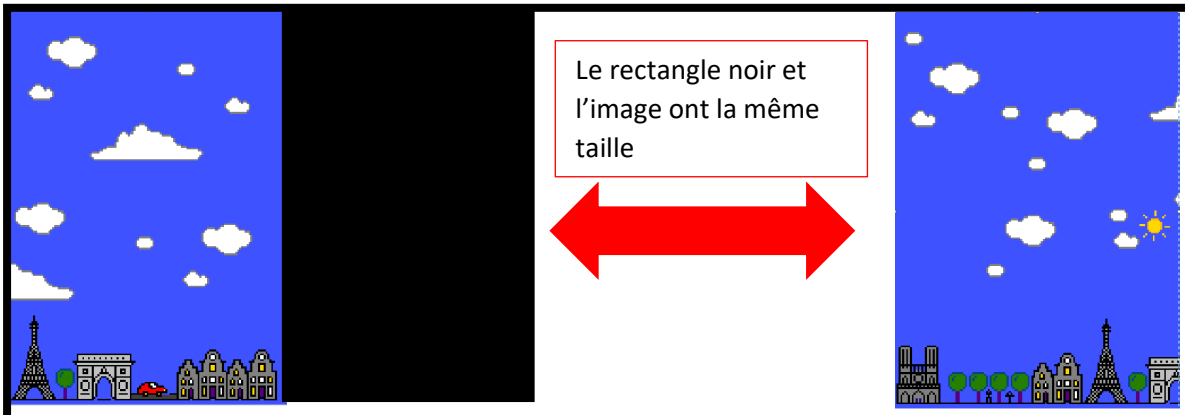
Etienne Corrège
Shi Hao YU
Matthieu Mouyart

Schéma explicatif du défilement infini de la carte



Affichage de la carte au début de partie

Affichage de la carte défilée après un certain laps de temps

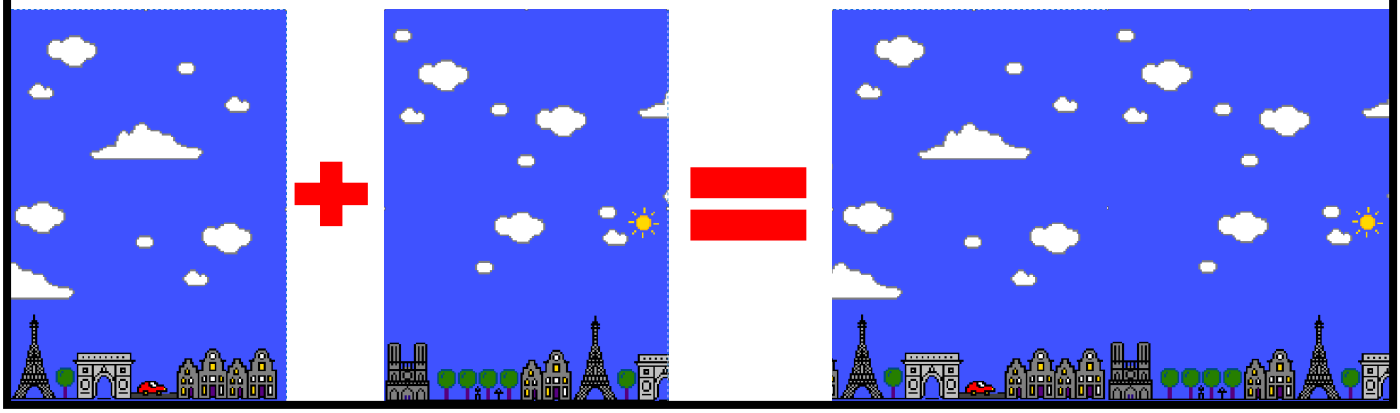


A partir d'un moment, on arrive en bout de carte

voici ce qui serait affiché à l'écran

Solution, on prend un morceau de carte au début de celle-ci pour compléter l'écran

Résultat :



Etienne Corrège
Shi Hao YU
Matthieu Mouyart

Différences entre chaque niveau

Ce qui change entre chaque niveau, est le degré de difficulté de ce dernier. Selon le degré de difficulté du niveau, il y a aura plus d'ennemis présents sur la carte, et ils disposeront également de plus de vie. De même, les boss propres à chaque niveau sont différents, ils auront pus de vie si le niveau augmente.

Et, en fonction du niveau la carte affichée à l'écran sera différente.

Tout au long de la progression, le joueur gardera son niveau de vie.

Explications des différents sous programmes

Les structures utilisées lors du programme

Afin de faciliter les échanges entre les différents sous-programmes, et la hiérarchisation des données, nous avons créé plusieurs structures, une pour le joueur, une pour les ennemis et une pour les boss. La structure du boss sera elle initialisée à chaque début de niveau.

La structure joueur est composée d'une variable image (le dirigeable que l'on affiche à l'écran), les coordonnées initiales au début de la boucle de jeu, le nombre de pixels correspondant au déplacement du joueur, et enfin, la vie du joueur.

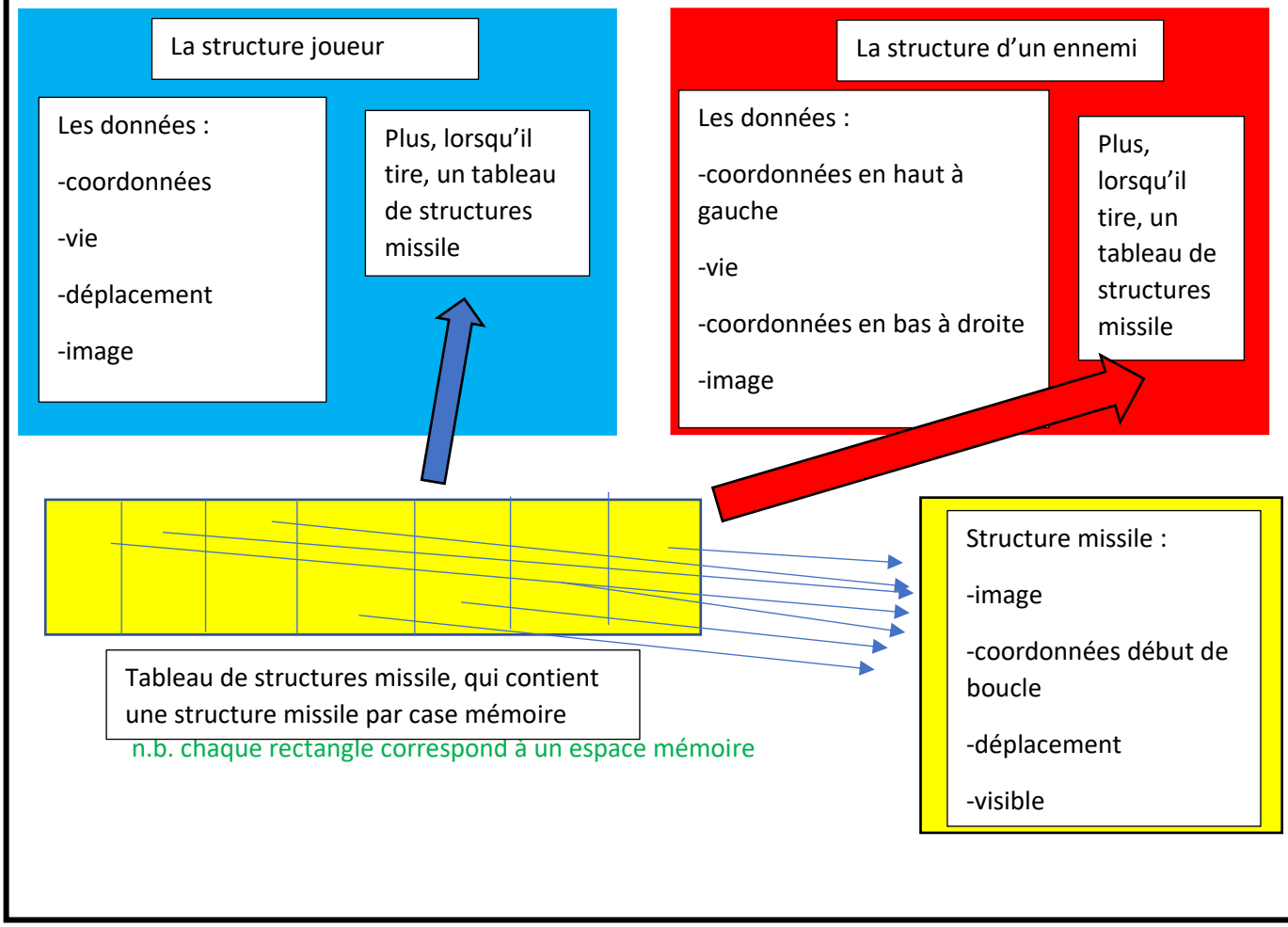
Les ennemis ont eux aussi une variable image (le petit avion rouge affiché à l'écran, ou les boss), les coordonnées au début de boucle, le nombre de pixels correspondant au déplacement de l'ennemi, ainsi que les coordonnées correspondant au coin bas droit du rectangle affiché sur la hitbox, et enfin, de la vie.

Les missiles, qui sont un peu plus particulier disposent également d'une structure, composée d'une image (le missile affiché à l'écran), de leurs coordonnées en début de boucle, du nombre de pixels correspondant au déplacement du missile, et d'une variable servant à détecter si le missile doit être affiché ou non à l'écran (par exemple si le missile touche un obstacle (ennemi, missile ennemi…) alors la variable est incrémentée pour faire comprendre au programme que ce missile ne doit plus être affiché).

Et, en complément avec les missiles, nous avons créé une structure qui gère tous les missiles à afficher sur l'écran, il a comme variables le nombre maximum de missiles a afficher à l'écran, le nombre de missiles actuellement à l'écran, et un tableau de missiles qui permettra de s'occuper de chaque missile affiché à l'écran.

Etienne Corrège
Shi Hao YU
Matthieu Mouyart

Voici quelques illustrations pour mieux comprendre le fonctionnement de ces structures.



Sous-programme création du joueur

Ce sous-programme est utilisé au tout début de la partie, il sert à allouer en mémoire une place pour la structure du joueur, il va également lui attribuer une image (le dirigeable), des coordonnées de déplacement et de la vie.

Sous-programme vérification du joueur

Ce sous-programme est utilisé à chaque boucle de jeu, il permet de vérifier les collisions du joueur avec d'autres objets sur l'écran et de prendre les décisions en conséquence.

1er cas, il est touché par un missile d'un ennemi, la variable état est incrémentée.

2ème cas, il a touché un bonus, la variable dégât est incrémentée.

3ème cas, il est touché par le missile d'un boss, la variable missile est incrémentée.

Etienne Corrège
Shi Hao YU
Matthieu Mouyart
Si la variable état est incrémentée, le dirigeable perd 1 point de vie.

Si la variable dégât est incrémentée, le dirigeable fait plus de dégât.

Si la variable missile est incrémentée, le dirigeable perd 20 points de vie.

En fonction du nombre de dégât que fait le dirigeable, son image va évoluer (4 évolutions au total, pour 7, 9, 11 et 13 points de dégâts).

Sous-programme créer bonus

Ce sous-programme est utilisé lorsque le nombre maximal de bonus présents sur la carte n'est pas atteint pour pouvoir afficher un autre bonus.

Il va allouer pour une structure bonus, des coordonnées, une image, et un angle (l'image se déplaçant selon un mouvement sinusoïdal, l'angle sert à gérer la valeur de ce déplacement).

Sous-programme d'actualisation des bonus

Ce sous-programme est utilisé à chaque tour de boucle pour vérifier si les bonus sont touchés par le joueur.

Il va parcourir les bordures du bonus sur la hitbox, et, s'il touche un allié, a variable visible passera à zéro. Et il ne sera plus affiché à la tour de jeu suivante.

Sous-programme créer missile

Ce sous-programme est utilisé lorsque le programme détecte que le joueur veut tirer (il a appuyé sur la barre espace), et qu'il va ajouter une structure missile dans le tableau de missiles de la structure qui gère tous les missiles.

Dans ce tableau, il y a plusieurs cases qui vont avoir chacune un accès direct en mémoire sur une structure missile.

Ce sous-programme va allouer en mémoire une place nécessaire à la case du tableau utilisée, il va ensuite allouer à cette structure en mémoire l'image du missile (qui va changer en fonction de l'appartenance de ce dernier, joueur, ennemi ou boss) ainsi que les coordonnées (celles correspondants à celles du joueur ou de l'ennemi qui souhaite tirer), et il va initialiser le nombre de pixels correspondant au déplacement du missile (qui va varier en fonction de l'appartenance du missile).

Sous-programme d'actualisation des missiles

Ce sous-programme va être utilisé par le tableau de missiles pour chaque case remplie (correspondant à un missile affiché sur l'écran), pour pouvoir actualiser ses coordonnées d'affichage. Il est utilisé à chaque fin de boucle de jeu pour actualiser les positions des missiles au prochain tour de boucle.

Simplement il va incrémenter les coordonnées en x et y de la valeur de déplacement du missile (variables de la structure missile), et va vérifier si le missile ne rentre pas en collision avec un objet sur la hitbox, ou s'il ne dépasse pas l'écran. S'il dépasse l'écran ou rentre en collision avec un objet alors la variable « valide » est décrémentée, et au prochain tour, le programme 'saura' que ce missile ne doit pas être affiché et qu'il faudra supprimer sa place en mémoire.

Sous-programme ajouter missile

Etienne Corrège
Shi Hao YU
Matthieu Mouyart

Ce sous-programme sera utilisé par le tableau de missiles lorsque le programme détecte un tir (allié ou ennemi), il va d'abord vérifier le nombre de missiles déjà affichés à l'écran. S'il y a encore de la place, il va passer à l'étape suivante, sinon il met fin à la procédure.

Dans l'hypothèse où il reste encore de la place pour afficher un missile, le programme va parcourir le tableau pour détecter une case libre, puis va appeler la fonction créer missile pour ajouter en mémoire la place de ce missile et lier la case de ce tableau avec la structure missile en mémoire. Puis il va incrémenter le nombre de missiles présents sur l'écran.

Sous-programme supprimer missile

Ce sous-programme est appelé lorsque le programme détecte dans le tableau de missiles, une case contenant un missile avec la variable « valide » à 0.

Le sous-programme va d'abord s'assurer qu'il y a bien une structure en mémoire liée à cette case du tableau, puis elle va simplement effacer de la mémoire la place de la structure missile de la case, et va remettre à NULL la valeur de la case (pour que le programme après puisse bien détecter que la case est dorénavant libre).

Sous-programme créer ennemi

Ce programme est utilisé par le programme lorsqu'il détecte (dans les paramètres de fonctionnement que nous avons prédéfinis) qu'il doit faire apparaître un nouvel ennemi.

Dans le programme principal est déclaré un tableau de structures 'ennemi', qui va lors de la création d'un nouvel ennemi faire appel à ce sous-programme.

Celui-ci va donc créer en mémoire une place pour une structure 'ennemi', il va allouer une image, des coordonnées, et le nombre de pixels correspondants au déplacement ainsi que la vie. Une fois cela fait, il va lier la case du tableau à l'espace mémoire alloué pour la structure 'ennemi'.

Sous-programme actualiser ennemis

Ce sous-programme est utilisé à chaque tour de boucle par le programme principal. Il parcourt les cases du tableau de structures d'ennemis, et pour chaque case appelle la fonction vérification des tirs alliés. Si la fonction de vérification des tirs retire plus de 20 points de vie à la structure, alors le sous-programme actualiser ennemis n'affichera pas cet ennemi.

Sinon, pour les ennemis qui n'ont pas été touchés par des missiles alliés, ou du moins sont encore en vie malgré cela, la fonction met à jour les coordonnées de l'ennemi (avec un déplacement aléatoire), tout en s'assurant que la structure ne va pas dépasser l'écran en largeur (car les avions se déplacent horizontalement et donc passé une certaine distance de déplacement seront automatiquement effacés, donc on ne se soucie pas du dépassement de la longueur de l'écran.).

Sous-programme créer boss

Ce sous-programme est utilisé à chaque début de niveau pur initialiser en mémoire le boss. Il va vérifier le niveau actuel et va en conséquence lui allouer une image, des coordonnées, un déplacement et de la vie (cela variant en fonction de chaque boss).

Sous-programme d'actualisation des boss

Ce sous-programme est utilisé à chaque niveau à chaque boucle de jeu. Il sert à vérifier si le boss est touché par un missile allié sur la hitbox, et actualise ses coordonnées par rapport à son déplacement.

Etienne Corrège
Shi Hao YU
Matthieu Mouyart
Et surtout, il va vérifier la vie du boss, et, ne l'affichera plus s'il n'a plus de vie.

Sous-programme détection des tirs alliés

Ce sous-programme est appelé à chaque tour de boucle par le programme principal, il va vérifier chaque case du tableau de structures « ennemi ».

Le sous-programme va vérifier les contours de l'ennemi sur la hitbox. En effet sur la hitbox les ennemis sont approximés à un rectangle de dimensions préétablies, et sont affichés en rouge, alors que les alliés (joueur et missiles) sont affichés en bleu.

Le sous-programme va donc vérifier chaque bordure de l'ennemi et, s'il détecte un pixel bleu va incrémenter une variable. A la fin de la vérification de la bordure, si la variable est différente de 0, le sous-programme saura que cette bordure a été touchée et retirera 10 points de vie à l'ennemi. Le sous-programme fera la même manipulation pour chaque bordure.

Sous-programme de vérification des tirs ennemis

Ce sous-programme va être utilisé par le programme principal à chaque tour de boucle, il suivra le même schéma de fonctionnement que le programme de détection des tirs allié. Il va vérifier les bordures du rectangle correspondant au dirigeable (le joueur) sur la hitbox. Pour chaque bordure il va vérifier s'il détecte un pixel de couleur ennemie, si c'est le cas, alors il va incrémenter une variable. Et à la fin de la vérification de la bordure s'il détecte que la variable est différente de 0, il va retirer 1 point de vie au joueur si c'est un missile d'un ennemi, et 20 points de vie si c'est un boss. Et la même manipulation sera effectuée sur chaque bordure.

Sous-programme créer liste

Ce sous-programme va être utilisé à chaque fois qu'un nouvel ennemi est créé en mémoire et qu'il va être affiché à l'écran. Il va créer en mémoire la place d'un tableau de structures missile qui sera donc lié à l'ennemi en question.

Il va initialiser le nombre maximum de missiles que peut contenir ce tableau, et mettre à NULL toutes les cases (optimisation de la mémoire pour éviter les fuites).

Sous-programme dessiner liste

Ce sous-programme est utilisé à chaque tour pour chaque ennemi présent sur l'écran. Il va permettre de parcourir le tableau de structures missile et d'afficher sur la hitbox chaque case qui n'est pas vide (cela veut dire que sur l'écran le missile de cette case est affiché). Il va afficher sur la hitbox tous les missiles que les ennemis auront tirés.

Sous-programme actualiser liste

Ce sous-programme est utilisé à chaque tour pour chaque ennemi, et servira à actualiser les coordonnées de chaque missile tiré par cet ennemi. Donc pour chaque ennemi présent sur l'écran, il va parcourir le tableau de structures missile, et, pour chaque case qui n'est pas vide (correspondant à un missile affiché à l'écran), les coordonnées sont mises à jour (coordonnées correspondant au déplacement du missile) par le sous-programme actualiser explicité ci-dessous.

Sous-programme actualiser

Ce sous-programme est donc appelé par le sous-programme actualiser missile, et il sert à mettre à jour les coordonnées de chaque missile présent sur l'écran. Il va également détecter si le missile ne

sort pas de l'écran, auquel cas il mettra à 0 la variable 'visible', pour que le programme comprenne qu'au tour prochain ce missile devra être supprimé.

Sous-programme afficher

Ce sous-programme est utilisé par chaque ennemi présent sur l'écran pour afficher chaque missile tiré, et par le joueur pour afficher chaque missile tiré. Simplement, il va afficher sur un écran tampon le missile.

Fonctionnement de l'affichage sur l'écran

Afin d'avoir un jeu qui soit graphiquement agréable à regarder, et qui puisse être fluide, nous devons utiliser une technique d'affichage appelée double buffering, qui permet justement d'éviter les effets stroboscopiques à l'écran.

Cette technique consiste à nouveau tour de boucle, à afficher tous les éléments présents à l'écran à la boucle précédente sur une page tampon (un buffer). Puis à la fin de cela d'afficher la page tampon à l'écran. Cela permet donc d'avoir un jeu graphiquement fluide.

Etienne Corrège
Shi Hao YU
Matthieu Mouyart

Conclusion sur notre projet

Ce projet était très instructif car il nous a permis de voir à un moindre niveau, la gestion et la création d'un jeu vidéo. Cela nous a surtout permis de nous rendre compte qu'une jeu vidéo représente énormément de rigueur dans son développement, ainsi que beaucoup de réflexions sur toutes les variables à prendre en compte.

Cela nous a montré que même pour un jeu vidéo très simple en 2d, cela reste très complexe à gérer, car il faut prendre en compte toute la gestion mémoire qui peut vite devenir casse-tête car elle devient très vite chargée avec toutes les structures créées. Cela nous a aussi montré que le travail d'équipe prime pour que chacun puisse travailler sur sa partie, car sinon, cela devient beaucoup trop compliqué pour une seule personne de s'occuper des plusieurs aspects du jeu en même temps.

Finalement, nous en sommes ressortis très enrichis, car nous avons encore pu développer de nouvelles techniques de conception pour s'assurer un maximum de rentabilité et d'efficacité. Nous avons également compris que la recherche permanente d'un maximum de clarté dans les explications, de simplification de chaque partie du code, sont primordiales pour que chaque personne puisse comprendre et réexpliquer/ réutiliser la partie de développement qu'il n'a pas fait.

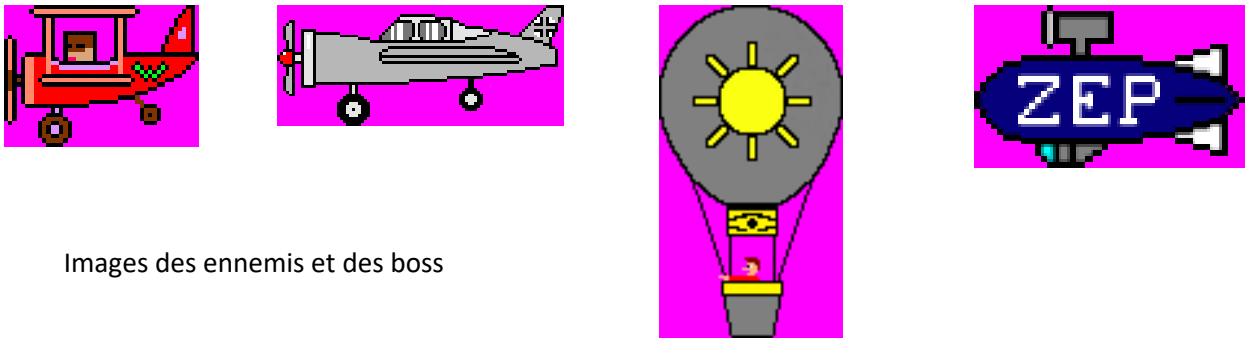
Nous avons clairement compris que le développement se faisant partie par partie en divisant les tâches, il est primordial que chacun commente son code, le simplifie au maximum pour les autres. Car tout le temps, il est nécessaire pour avancer d'utiliser le code déjà développé par d'autres, et conséquemment il est indispensable que ce code soit au maximum compréhensible dans ses moindres fonctionnements.

Cette expérience fut très enrichissante pour nous apporter une vision critique de la conception et du développement pour tout type de projet dès lors qu'il devient conséquent.

Etienne Corrège
Shi Hao YU
Matthieu Mouyart

Bibliothèque d'image du jeu

5 images du dirigeable

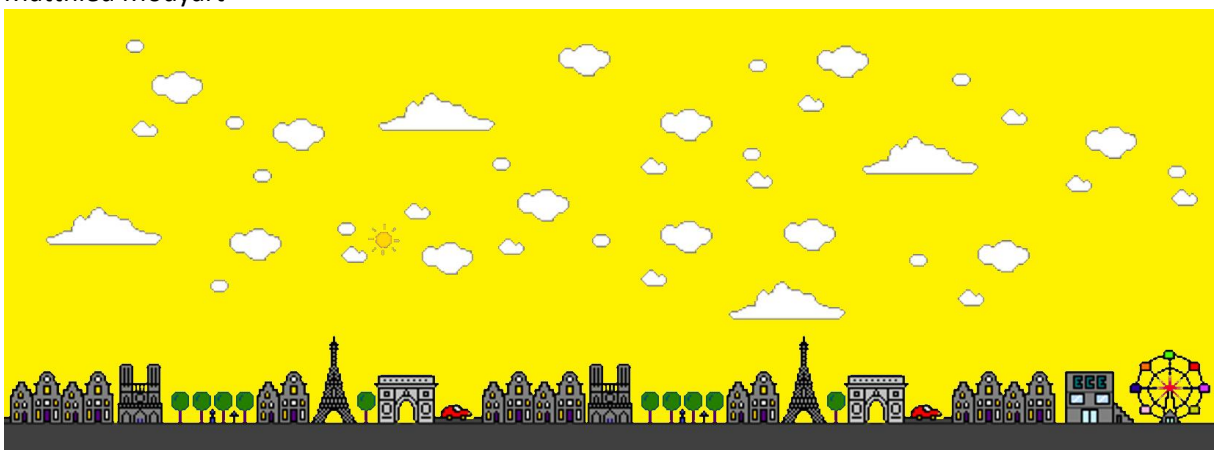


Images des ennemis et des boss

Images des missiles (allié à gauche et ennemis à droite).

Fond du 1er niveau

Etienne Corrège
Shi Hao YU
Matthieu Mouyart



Fond du 2ème niveau

Fond du 3ème niveau

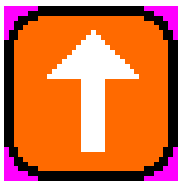


Image du bonus