Jeu d’aventure sur l’environnement programmé en JavaScript

Graphical user interface

Description automatically generated

Loïc Betschen – 3MG06

2021 – Lycée Denis-de-Rougemont – Manuel Fragnière

Table des matières

[Introduction 3](#_Toc84584412)

[Travail de maturité 3](#_Toc84584413)

[L’informatique 3](#_Toc84584414)

[Mes objectifs 4](#_Toc84584415)

[Le jeu 4](#_Toc84584416)

[Méthode et Manière 5](#_Toc84584417)

[Organisation de mon code 5](#_Toc84584418)

[Github 6](#_Toc84584419)

[Le Canevas 6](#_Toc84584420)

[Les cookies 7](#_Toc84584421)

[Graphisme 7](#_Toc84584422)

[La sauvegarde du joueur 7](#_Toc84584423)

[Matrix de la *map* 8](#_Toc84584424)

[Détection de collision 9](#_Toc84584425)

[Interaction avec les objets 9](#_Toc84584426)

[Méthode de l’objet 11](#_Toc84584427)

[Le saisi du joueur 11](#_Toc84584428)

[Déplacement du joueur 11](#_Toc84584429)

[Difficulté engendrée pendant la réalisation de mon jeu 12](#_Toc84584430)

[Résultats 14](#_Toc84584431)

[Accès au jeu 14](#_Toc84584432)

[La Page Menue 14](#_Toc84584433)

[La carte du monde 14](#_Toc84584434)

[Les niveaux 15](#_Toc84584435)

[La page de paramètre 17](#_Toc84584436)

[Bilan personnel 18](#_Toc84584437)

[Conclusion 18](#_Toc84584438)

[Bibliographie 18](#_Toc84584439)

# Introduction

L’idée de jeux vidéo est né dans les années 1950 au sein des universités. Ces jeux ne se font que connaitre véritablement après le boom des années 1970. L’un des premiers jeu vidéo étaient un simple jeu de morpions avec comme écran un tube cathodique et tourné sur l’ordinateur EDSAC, un ordinateur destiné à la recherche académique. Le jeu était joué tout seul avec comme opposant une intelligence artificielle. Ce jeu nommé OXO fut créé par le Britannique Alexander S. Douglas pour son doctorat en philosophie à l’université de Cambridge. Au fil du temps les ordinateurs sont devenus beaucoup plus puissants et accessibles au monde entier et les écrans ont une meilleure résolution. En plus de tout cela des applications très puissantes ont étaient développer pour les développeurs de jeux vidéo. Tout cela nous a permis de crée des jeux vidéo presque réel au niveau du graphisme et du fonctionnement. Ces jeux sont tellement performants qu’ils sont capables de nous divertir a telle point que nous payons pour regarder d’autres gens joué. La seule limite à ces jeux est notre imagination. Nous pouvons créer notre propre monde a nous seul, ou sauver la galaxie ou combattre des monstres fantastiques, etc. Cette idée que notre imagination est notre limite est l’un des raisons pour laquelle j’adore la programmation. C’est aussi la raison de mon choix de programmer un jeu vidéo.

# Travail de maturité

La branche de mon travail de maturité est l’informatique avec le sujet étant « L’informatique pour les « pros » de la programmation ». Le but était de programmer une application, un jeu ou un site web. Pour mon travail j’ai réalisé un jeu vidéo sur l’environnement en JavaScript sur une période de neuf mois. J’ai choisi l’environnement comme thème principal de mon jeu car c’est un sujet pertinent et important et je voulais qu’il ait un lien entre la vie réel et mon jeu. Mon objectif principal était de programmé un jeu vidéo en 2 dimensions (2D) en utilisant l’élément du canevas dans HTML.

## L’informatique

L’informatique est la théorie et traitement de l’information à l’aide de programmes mis en œuvre sur ordinateurs. La programmation peut être considéré comme la communication entre humain et machine. Cette « communication » se fait grâce aux langages de programmation qui sont entre le langage binaire (langage des ordinateurs) et le langage humain. J’ai choisi l’informatique car cela est un sujet qui me passionne beaucoup. Le fait de pouvoir écrire quelque ligne de code pour animer un écran, calculer des fonctions compliquées ou même analyser du data complexe m’éblouie et m’intrigue. J’ai déjà participé à quelques camps d’informatiques qui m’ont permis à mieux comprendre la programmation et avoir une affinité pour ce sujet.

Le langage de programmation le plus utiliser pour mon jeu était le JavaScript. JavaScript est un langage de programmation souvent utiliser avec le langage HTML (HyperText Markup Language) et CSS (Cascading Style Sheets). Ces trois langages sont utilisés pour créer, formater et animer les éléments d’un site web. Le JavaScript est utilisé pour animer les éléments HTML, l’HTML pour les créer et le CSS pour les formater. J’ai utilisé ces trois langages pour réaliser mon jeu vidéo, en utilisant majoritairement le JavaScript.

## Mes objectifs

Un de mes plus grands buts que je voulais atteindre était que le jeu ait un lien avec la vie réel en d’autres mots je voulais que mon jeu aborde une problématique qui est présent dans le monde actuel et de faire passer les solutions à travers ce jeu. Un simple jeu arcade est relativement facile a programmé lorsqu’on connait les bases d’un langage de programmation. Mais le fait d’intégrer un but qui est en lien avec la vie réel dans le jeu complique la chose entièrement. Il ne suffit pas juste de poser des simples buts du jeu comme étant de tuer le méchant ou de collectionner toutes les pièces ou même de sauver un personnage d’un dragon infernal. Si on veut faire passer un message a travers notre jeu, il faut d’abord trouver un problème qui est généraliser et ensuite trouver des solutions à cela. Puis il faut trouver une façon d’intégrer ce problème dans le jeu et de faire comprendre au joueur les solutions nécessaires pour résoudre cette problématique.

Un autre but visé était de programmé le jeu pour que le joueur puisse sauvegarder son progrès. Si le joueur peut arrêter à tout moment de jouer et de pouvoir reprendre sa session plus tard rends le jeu plus intéressant et moins ennuyeux. Cela nous permet aussi de rendre le jeu plus grand sans se soucier que le joueur n’atteindra jamais la fin.

Mon dernier but était de rendre le jeu graphiquement plaisant. Je voulais que le jeu ait une uniformité par rapport au style d’image et que le thème des ces images soit en lien avec le but du jeu. Ces trois buts ont été les plus gros défis que j’ai rencontré lors de la réalisation de mon jeu.

## Le jeu

Le jeu vidéo est un jeu d’aventure en 2D. Le joueur incarne un jeune personnage qui vient de trouver la solution à tous les problèmes qui concernent la dégradation de l’environnement et du climat. Le but est de transmettre cette solution au monde entier. Les six niveaux de mon jeu représentent les cinq continents sans l’Antarctique. Le joueur devra éviter les nuages toxiques appelés les « *polluers* », collecter des déchets qui sont représenter par des canettes et apporter la solution à chaque continent. Mon jeu a été programmé de façon à permettre la sauvegarde des progrès du joueur. Le jeu sauvegarde le nom du joueur, le score, les vies restantes, le volume du son du jeu et de la musique et le progrès dans chaque niveau donc les déchets,les vies collectionner, les *checkpoints* et les ennemies restant. Le jeu est fait de façon à pouvoir adapter sa taille, cela veut dire que si le joueur veut agrandir ou réduire la taille du jeu il peut le faire sans qu’il ait des déformations des images et du jeu*.* Les niveaux se débloquent au fur et à mesure que le joueur progresse dans le jeu.

# Méthode et Manière

## Organisation de mon code

Text

Description automatically generatedPour pouvoir mieux naviguer entre les fichiers, j’ai créé neuf dossiers. Ces dossiers s’appellent « *Menu* », « *Map* », « *Level\_1* », « *Level\_2* », « *Level\_3* », « *Level\_4*»*,* « *Level\_5* », « *Level\_6* »*,* « *GlobalScripts* » et«*Assets* ». Les huit premiers dossiers ont chacun un fichier HTML (\*.html) et des fichiers contenant le code JavaScript (\*.js). Le dossier *Assets* est composé d’images et de fichiers audio utilisés dans mon jeu. Le dossier *GlobalScripts* contient tous les fichiers JavaScript qui sont utilisé dans plusieurs des pages HTML.

Tous les dossiers, sauf *Assets* et *GlobalScripts*, ont chacun un fichier \*.html, un fichier

\*index.js, un fichier \*game.js et des fichiers JavaScript propres à leurs fonctionnalités.

Les dossiers des niveaux ont chacun un fichier *map.txt* et *IObjects.txt*. Ces fichiers texte sont remplis d’une matrice de nombres qui représentent une tuile de la *map* ou un objet interactif.

Les fichiers *\*.html* sont composées de balises et décrivent les éléments présents dans la page web. Les balises les plus utilisé dans mon jeu sont les balises *<img>* pour les images, *<audio>* pour les fichiers audios, *<canvas>* pour le canevas et *<script>* pour le code JavaScript. Ces balises chargent et nomment toutes les images, le canevas, les audio et les fichiers *\*index.js*.

Les fichiers \**index.js* définissent la taille du canevas, appellent la fonction *start()* du script \**game.js* et font tourner la fonction *GameLoop()* qui est appelée à chaque fois que la page est redessinée. Dans la fonction *GameLoop()*, les fonctions *draw()* et *Update()* du fichiers game.js sont appelées à chaque rafraichissement de la page. Pour appeler la fonction *GameLoop()* à chaque image on écrit :

Text

Description automatically generated

Le timestamp représente un point dans le temps indépendant de tout fuseau horaire ou calendrier. *RequestAnimationFrame()* appelle la fonction *GameLoop()* à chaque fois que l’image est redessiner sur l’ordinateur.

Les fichiers game.js importent tous les scripts nécessaires pour sa page HTML. Le code ressemble à ceci :

Text

Description automatically generated

Une classe est ensuite définit. Elle comporte un constructeur et les différents fonctions comme *start(),* *update()* et *draw().*Le constructeur définit à la classe ses propriétés et les méthodes associées. La fonction *start()* est utiliser pour appeler ou définir certaine variable quand le page est ouverte. La fonction *update()* mets tous ces variables a jour à chaque image. La fonction *draw()* dessine tous les objets du jeu, comme par exemple le jour, la *map*, les ennemies, les objets interactive, etc.

Tous les fichiers, sauf le fichier Input.js et PlayerProgress.js, JavaScript comporte chacun une classe avec une fonction *update()* et *draw()* et divers autres fonctions propre à chaque script.

## Github

Github est un service de gestion de versions basé sur le web pour les développeurs. J’ai utilisé cet outil pour pouvoir accéder à mon code partout et publier mon jeu sur un server ce qui a permis l’accessibilité du jeu.

## Le Canevas

Le canevas est un élément HMTL ou l’on peut dessiner des images et des objets. Pour dessiner en 2D sur le canevas on doit appeler le canevas et définir le contexte du canevas comme étant en 2D. La variable du contexte se nomme *ctx*. Text

Description automatically generated

Pour le système de coordonné du canevas le 0 de l’axe X et de l’axe Y est en haut à gauche. On peut ensuite écrire ceci pour dessiner une image sur le canevas :

Text

Description automatically generated

Les variables *x\_image* et *y\_image* représentent la position de l’image sur le canevas. Et les variables *width\_image* et *height\_image* représentent la taille l’image. On écrit ceci pour recadrer l’image :



Les variables *x\_recadrer* et *y\_recadrer* représentent le point où le recadrage va commencer. Les variables *width\_recadrer* et *height\_recadrer* représentent la taille du recadrage. Pour effacer le canevas, on écrit ceci :



Ce code effacera un rectangle du canevas. La taille et la position sont définies par *x*, *y*, *width* et *height*. Ce ligne de code est appeler dans la fonction *GameLoop()* pour effacer les images sur le canevas. Ces images seront redessiner juste après grâce la fonction *draw().*

## Les cookies

Un cookie est un fichier texte liée à un site web. Ce fichier est composé de double variable sous cette forme : « *name=Charles* ». Tous les cookies ont une date d’échéance qui ressemble à cela : « *expires=Thu, 18 december 2022 12 :00 :00 UTC* ». Les cookies sont souvent utilisés pour suivre, personnaliser et enregistrer des informations sur la session de chaque utilisateur sur le site web. Ils sont enregistrés localement dans votre navigateur internet. Il y deux différents types de cookies, les cookies de sessions et les cookies persistants. Les premiers sont des cookies qui s’effacera à la fin de votre session et les deuxièmes sont garder jusqu’à leur date d’échéance.

## Graphisme

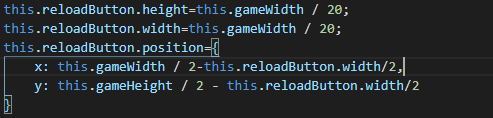
La grande majorité des images utilisées dans mon jeu sont libres de droit que j’ai trouvé sur des sites spécialement fait pour les développeurs de jeux vidéo. Le graphisme était le plus gros obstacle que j’ai rencontré pendant la réalisation de mon jeu. Je devais trouver des images de préférence sans droit d’auteurs, du même style et qui était en concordance avec la charte graphique du jeu. Ces trois critères m’ont obligé à redéfinir le fil rouge de l’histoire plusieurs fois car je n’étais pas capable de trouver des images correspondantes.

## Définition d’un objet du jeu

Pour définir la taille et é la position dans mon jeu, nous utilisant la méthode de l’objet et du constructeur. On définit l’objet comme étant une image et en lui attribut la source de l’image dans le constructeur d’une classe :



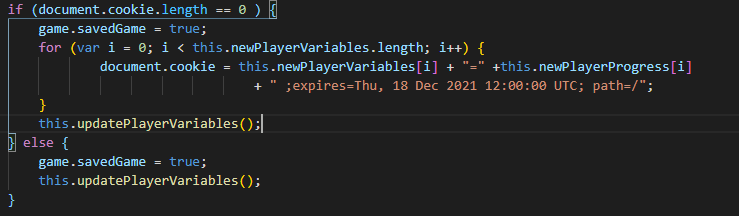
Puis on définit sa taille et sa position dans le même constructeur :



On peut mettre à jour la taille et la position de cette objet en copiant ce code dans la fonction *update().* On peut aussi ajouter d’autres variables a cet objet en écrivant le nom de l’objet suivi d’un point puis du nom de la variable voulu.

## La sauvegarde du joueur

Le progrès du joueur est sauvegardé dans des cookies. En utilisant un code optimiser et des fonctions bien précises, la tache de sauvegarder le progrès devient très simple. Comme la plupart des fichiers JavaScript de mon jeu, *PlayerProgress* contient une classe *PlayerProgress* avec un constructeur et des fonctions. Le constructeur est composé de quatre *array* : *newPlayerProgress*, *newPlayerVariables*, *playerProgress* et *playerVariables*. Les deux premiers stocks les variables et les valeurs de base et les deux autres sont vides. Ces deux dernières vont englobées les valeurs du joueur. Ces valeurs seront modifiées au fil du jeu. Il y a quatre fonctions, *getSavedPlayer*, *updatePlayerVariables*, *getCookie* et *changeCookie*. *GetSavedPlayer* écrit les cookies si aucun jeu sauvegardé n’est détecté. S’il y a un jeu sauvegardé, la fonction appellera la fonction *updatePlayerVariables*.



La fonction *updatePlayerVariables* est utilisé quand on lance le jeu ou quand on modifie un cookie.

Text

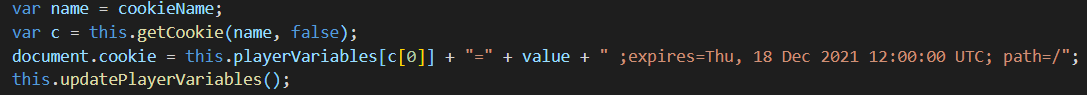
Description automatically generated

La fonction *getCookie* est utilisé pour récupérer une variable stocker dans les cookies ou une variable de base.

Text

Description automatically generated

La dernière fonction *changeCookie* modifie un certain cookie avec une valeur donné.



## Matrice de la *map*

Une matrice de chiffre est utilisée pour créer, changer et dessiner facilement la *map* des niveaux. Un code général lira ensuite ce matrix et dessinera les tuiles de la *map* dans la bonne position. La matrice ressemble à ceci :

A screenshot of a computer

Description automatically generated with medium confidence

Shape

Description automatically generatedLe matrix a 200 colonnes et 23 lignes. Ces nombres correspondent à une tuile dans la *map* des tuiles. La *map* des tuiles est une image avec plusieurs tuiles et elles ressemblent à ceci :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 7 | 8 | 9 | 10 | 11 | 12 |

Un code dessinera la tuile correspondant aux chiffres présents dans la matrix en recadrant la photo sur la bonne tuile.

Text

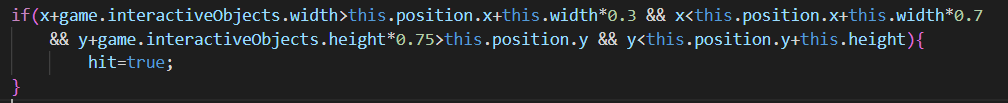
Description automatically generated

Les variables sources représente les coordonnées pour recadrer la *map* de tuile et les variables *x* et *y* sont les coordonnées de la tuile sur le canevas. Cette méthode est aussi utilisée pour les objets interactifs.

## Détection de collision

## Interaction avec les objets

Les objets interactifs sont des objets avec laquelle le joueur peut interagir avec comme les coffres que le joueur peut ouvrir pour soi récupérer des « canettes » ou des munitions pour son pouvoir. Ces objets englobent les canettes, les coffres, les ennemies et la boite aux lettres qui représente la fin du niveau. Pour déterminer si le joueur peut interagir avec un de ces objets il faut contrôler si le personnage est en contact avec ces objets. Voici le code :



Ce code se trouve dans le fichier player.js pour pouvoir avoir tous les algorithmes de collisions dans un seul fichier. Voici un schéma expliquant le code ci-dessus :

2

1

X et Y

Joueur

3

4

Les carres noires représentent la zone de collision des objets interactifs et le carré noir celui du joueur. Les croix noires sont l’endroits où les coordonné x et y sont définis. Ces carre sont juste en collision avec le joueur (le schéma est exagérer) et ils représentent chacun une des consignes du if(). Donc pour qu’un objet soit en collision avec le joueur le carre doit être dans une position entre ces quatre carres noires. L’instruction if contrôle cela grâce a quatre conditions.

1. 
2. 
3. 
4. 

Le *x* et le *y* sont les coordonne des objets interactifs et toutes les variables avec *this* définissent le joueur. Chaque condition est associée au carré avec le même numéro. Le signe < signifie que chiffre est plus petit qu’un autre (nombre < autre nombre) et > qu’un chiffre est plus grand qu’un autre (autre nombre > nombre) si c’est le cas le code dans l’instructions if est lu.

## Méthode de l’objet

Un objet dans JavaScript est une entité qui possède des propriétés. Prenant par exemple un crayon. Le crayon est l’objet et il a plusieurs propriétés comme sa taille, sa couleur ou son poids, etc. Cet objet peut être définit dans une classe a l’aide d’un constructeur. Une classe est une fonction qui a des propriétés définit par un constructeur.

## Le saisi du joueur

On saisi les frappes de touches et le mouvement de la souris grâce un simple code.

Graphical user interface, text

Description automatically generated

Ce code ajoutera un gestionnaire d’événements au canevas. Dans ce cas l’événements est l’appui sur une touche. On peut changer « *keydown* » a « *keyup* », « click », « *mousemove* », « *mousedown* » ou « *mouseup* ». Il y encore plus d’événements possible mais ceux-ci sont ceux qui sont utiliser dans mon jeu. On utilisant *keydown* ou *keyup* on peut récupérer le nombre de la touche qui est appuier ou relâcher. Ensuite on utilise un switch case pour définir la conséquence dépendant de la touche concernée.

Text

Description automatically generated

On récupère la touche grâce à « *event.keyCode* » puis la bonne fonction est appeler en fonction de la touche concerné.

## Déplacement du joueur

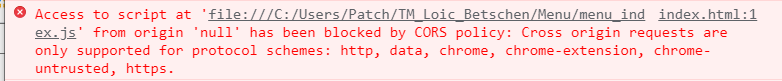
Le déplacement du joueur se fait en deux parties, la première est quand le joueur est aux extrémités à droite ou à gauches et la deuxième est quand le joueur est entre ces deux extrémités. Quand le joueur se déplace dans les extrémités ce sont les coordonnés du personnage qui change et la *map* reste statique, mais quand le joueur se déplace entre ces deux bornes ce sont la *map* et les objets qui changent de coordonne. Cela donne l’impression que le joueur se déplace alors qu’il reste statique sur l’écran. Cela donne la possibilité de crée des niveaux plus longs avec un meilleure visuel.

Text

Description automatically generated

*This.offsetX* représente le décalage de la *map* par rapport à la position du joueur quand il est au milieu de l’écran. Le décalage est négatif car la *map* se déplace de droite à gauche.

## Difficulté engendrée pendant la réalisation de mon jeu

Comme tout programmeur j’ai rencontré des difficultés au fil que je programmais mon jeu. Le premier problème que j’ai rencontré est celui d’un problème de politique CORS(Cross-Origin Reference Sharing). 

Cette politique a interdit mon fichier index.html à accéder au fichier *\*.js* car mon disque dur n’était pas défini comme étant un server. Pour résoudre ce problème j’ai décidé d’utiliser GitHub pour mettre mon jeu sur un server et d’en faire un site web accessible par tout le monde. Cela m’a permis de faire en sorte que personne d’autre ai ce problème.

Le deuxième problème était celui de la résolution des images sur le canevas. Si on définit la taille du canevas dans la balise du canevas ce n’est pas la taille du canevas dans la page qui va changer mais les pixels inclus dans le canevas. Il suffisait de définir la taille du canevas dans la balise <*style>* comme étant automatique et d’ensuite déclarer la hauteur et la largeur du canevas dans le fichier *\*index.js*.

Graphical user interface, text

Description automatically generated

Ensuite il y a pas mal de soucis avec la sauvegarde du joueur. Je devais trouver un moyen de sauvegarder le progrès du joueur et ensuite pouvoir récupérer, modifier et supprimer certaine variable. Au début j’ai pensé utiliser un fichier *.text* pour sauvegarder le progrès du joueur mais cela je devais utiliser une base de données. Heureusement en faisant un peu de recherche je suis tombé sur les cookies. Les cookies sont beaucoup plus simples à utiliser qu’une base de données et c’est plus rapides de récupéré, modifier et supprimer les variables du joueur.

Quand j’ai commencé à introduire plus d’images dans le jeu, j’ai eu un problème de téléchargement des images. Au début j’appeler mes images dans les fichiers *\*.js* comme ci-dessous :

Graphical user interface, text

Description automatically generated

Malheureusement l’image n’est que charger quand le code lit ces deux lignes. Donc la page charger et rien ne s’afficher car les images ne se sont pas chargées dans la page. Pour éviter ce petit bug il faut tout simplement appeler les images dans des balise *<img>* et leur donner un nom à l’élément. Cela fait en sorte que les images seront chargées au chargement et au lancement de la page. Il faut ensuite cacher les images pour qu’ils n’apparaissent pas sur la page en dehors du canevas en utilisant ce code :

Text

Description automatically generated

Et pour ensuit appeler les images dans le jeu il suffit d’écrire :

Text

Description automatically generated

Un des plus grands obstacles rencontrés pendant la réalisation de mon jeu était le graphisme. J’avais de la peine a trouvé des images qui correspondaient à ce que je voulais. Je voulais que toute les images aient le même style pour rendre mon jeu uniforme. A finir…

La plus grosse difficulté rencontrée dans la programmation de mon jeu était la gestion du volume du personnage. J’ai essayé différentes méthodes pour créer un algorithme de collision mais à chaque essai je retrouvais des petits *bugs*. Ces *bugs* se présentaient souvent quand le joueur saute. A finir…

# Résultats

## Accès au jeu

Le jeu a été fait de sorte que toutes personnes puissent accéder au jeu. Pour y accéder il suffit d’entrer ce lien dans votre web browser : <https://lbetschen.github.io/TM_Loic_Betschen/>

## La Page Menue

La page menue est la plus importante car c’est l’interface d’accueil du jeu que l’on voit en premier. Cette page est composée de plusieurs éléments HTML différents. Il y a la balise *<video>,* *<input>* et bien sûr la balise *<canvas>.* La première balise permet l’affichage de la vidéo dans l’arrière-plan. La deuxième sert à récupérer le nom du joueur. Et la dernière affiche les boutons et la page des paramètres.

Graphical user interface

Description automatically generated

## La carte du monde

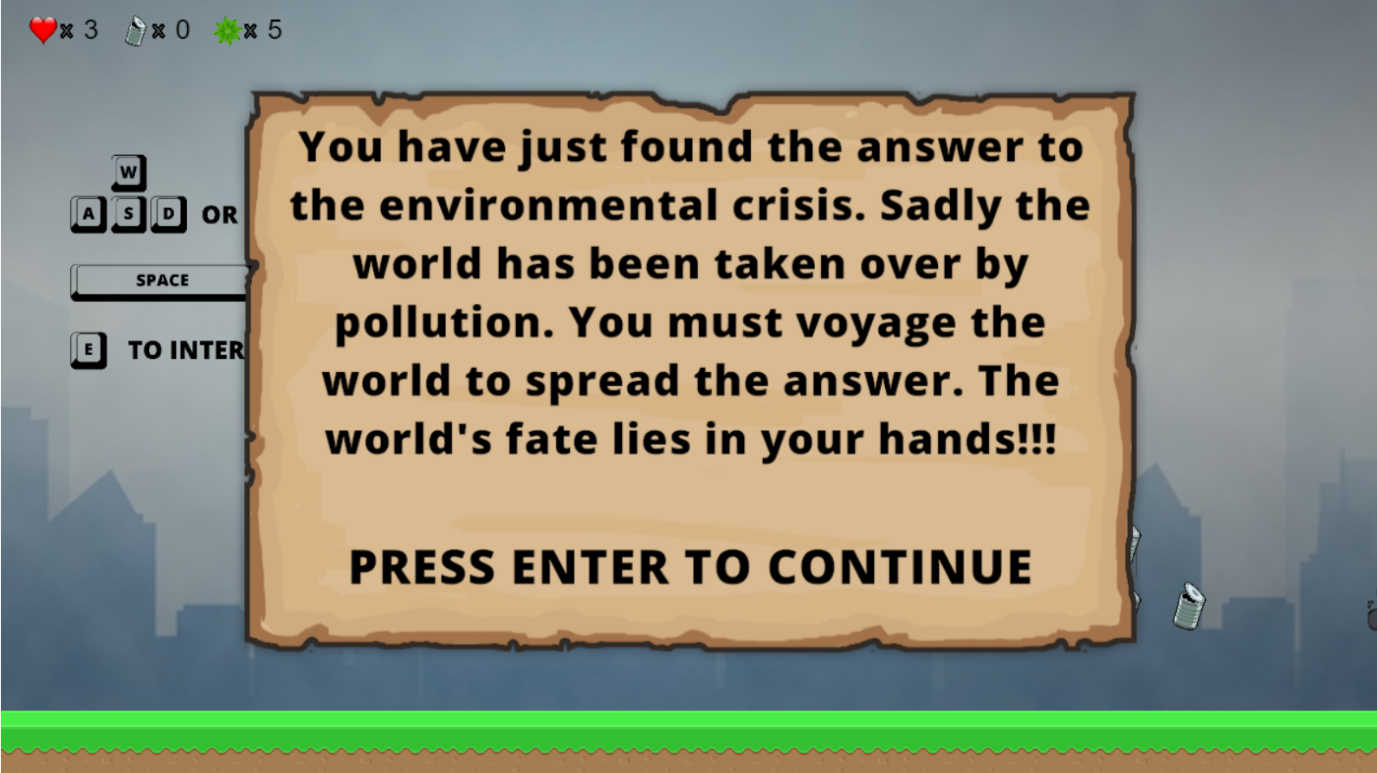
Cette page est consacrée au choix des niveau. Sur cette page il y les boutons pour les niveaux, le bouton retour pour retourner à la page menue et le bouton paramètre. Cette page sert à visualiser le progrès du joueur dans le jeu grâce à la carte du monde.

Diagram

Description automatically generated

## Les niveaux

Les niveaux sont tous très semblables et sont composées des mêmes éléments dans le canevas. La seule différence qu’il y a entre ces différents niveaux est la *map*. Elle est unique et diffèrent pour chaque niveau. Le niveau un a aussi en plus des autres une petite explication du jeu et comment jouer au tout début.

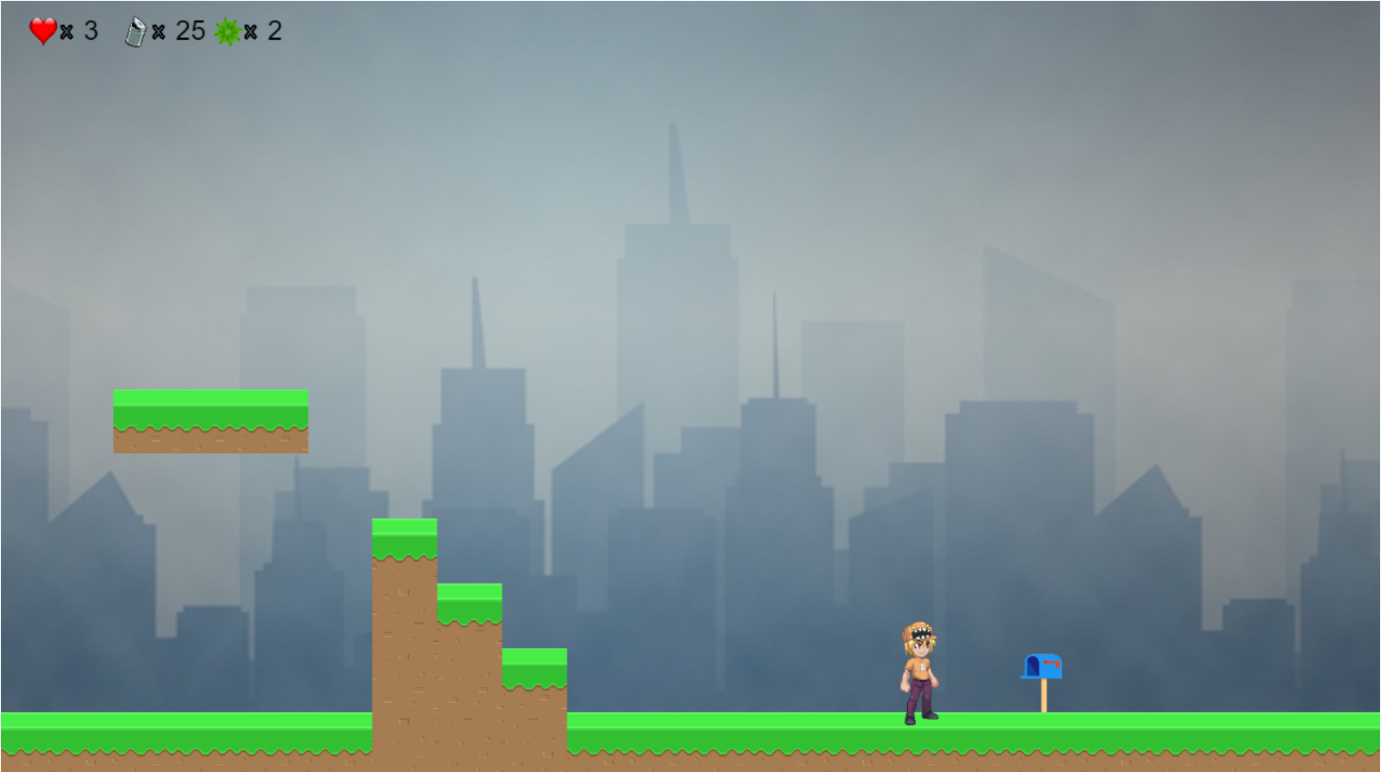




En haut à gauche se trouve les vies restantes du joueurs, les canettes collectionner et le nombre de munition restants. Les canettes sont immobiles contrairement au petit nuage, le « *polluer* » en bas a droite. Il se déplace de dans une zone délimiter tant qu’il n’a pas tuer le joueur ou qu’il se fasse tuer par le joueur.



La fleur en bas à gauche représente un checkpoint, quand elle est fermée le joueur ne le pas encore atteint. Dès que le joueur la touche elle ouvre ses pétales. La poubelle en bas à droite contient cinq canettes que le joueur peut récupérer. Les poubelles vertes contiennent des munitions pour le pouvoir du joueur.



La boite aux lettres signifie la fin du niveau dès que le joueur l’atteint et dépose la lettre contenant la solution, le joueur retourne la carte du monde ou le prochain niveau est débloquer.

## La page de paramètre

La page de paramètre est très simple, il y deux boutons coulissants et deux interrupteurs pour le volume de la musique et le volume des effets sonores du jeu. Et un dernier bouton pour sauver les paramètres et quitter la page des paramètres. Le volume de la musique et les effets sonores du jeu se modifient lorsqu’on déplace les boutons coulissants. Quand on clique sur les interrupteurs, les boutons coulissants se déplacent pour représenter la valeur du volume.

A picture containing text, indoor

Description automatically generated

## Amélioration

Le jeu peut être améliorer de plusieurs façons différentes. Les niveaux pourraient être plus complexes et plus longs. En ajoutant un *OffsetY* comme nous avons fait avec le *OffsetX,* on aurait pu agrandir l’aire ou le joueur peut se déplacer et donc rendre les niveau plus long et compliqué. Le fait d’ajouter plus d’animations au jeu, aurait permis une meilleure fluidité. Comme quand le joueur meurt, on aurait pu envisager une animation de l’écran ou du joueur. On peut aussi appliquer cela au *polluers.* Des objets interactifs par rapport à la *map* aurait rendu les choses plus intéressantes comme des plateformes mobiles, ou des portes amenant à une chambre secrète. On aurait pu ajouter d’autres pouvoir au joueur et ajouter d’autres méchants ou obstacles au jeu. Comme des flaques d’eau polluer qui tuera le joueur s’il marche dessus. On pourrait envisager à ajouter des petits conseille ou information par rapport au thème de l’environnement.

# Bilan personnel

Je trouve que ma méthode de travail était bien, je me suis organisé correctement et je me suis investi considérablement dans la réalisation de ce projet. J’ai travaillé en moyenne quatre à six heures par semaines. J’ai consacré les deux premiers mois à la découverte du canevas et de son fonctionnement. J’ai pris ensuite trois mois pour poser les bases de mon jeu. Le mois d’août, j’ai principalement optimisé mon code et régler quelque beugue. Le mois de septembre fut consacré à mon rapport et la création de la map de mes niveaux. Le dernier mois j’ai finis mon rapport et a finis mon jeu. Mon seul souci est que je n’étais pas assez précis sur le fil rouge et le but de mon jeu. J’étais très ambitieux de créer un jeu avec un six niveau différents correspondant à un problème lié a la dégradation de l’environnement. Malheureusement ce n’étaient pas mes capacités à coder qui ne m’a pas permis de réaliser jeu mais plutôt les graphismes. J’avais beaucoup de peine à trouver des images uniformes et convenant à ce que je voulais pour mon jeu. J’ai passé autant de temps à chercher pour des images qu’a coder mon jeu. Créer toutes les images moi-même n’était pas une option car je ne suis pas un bon artiste. Finalement je suis satisfait de ce que j’ai réussi à réaliser même si le fil rouge est moins relié au combat contre la dégradation de l’environnement.

# Bibliographie

Stackoverflow, <https://stackoverflow.com/> , consultée à chaque session de code

W3schools, <https://www.w3schools.com/> , consultée à chaque session de code

Intro to Game Development with JavaScript - Full Tutorial, <https://www.youtube.com/watch?v=3EMxBkqC4z0> , consultée entre mars 2021 et juin 2021

Unity Asset Store, <https://assetstore.unity.com/> , consultée lors de ma recherche d’image

Toppng, <https://toppng.com/> , consultée lors de ma recherche d’image

VectorStock, <https://www.vectorstock.com/> , consultée lors de ma recherche d’image

Free Vector, <https://www.freevector.com/> , consultée lors de ma recherche d’image

Vecteezy, <https://www.vecteezy.com/> , consultée lors de ma recherche d’image

Freepik, <https://www.freepik.com/vectors/art> , consultée lors de ma recherche d’image

PNGTree, <https://pngtree.com/>, consultée lors de ma recherche d’image

The Like Minded, *Environment Awareness Animation - The Effects of Deforestation,* UK, 2019, <https://www.youtube.com/watch?v=E_091pd_KHM>

OXO (jeux vidéo), <https://fr.wikipedia.org/wiki/OXO_(jeu_vid%C3%A9o)>, consultée le 10 octobre 2021

Histoire du jeu vidéo, <https://fr.wikipedia.org/wiki/Histoire_du_jeu_vid%C3%A9o>, consultée le 10 octobre 2021.