Jeu d’aventure sur l’environnement programmé en JavaScript

Graphical user interface

Description automatically generated

Table des matières

[Introduction 3](#_Toc83827604)

[Travail de maturité 3](#_Toc83827605)

[L’informatique 3](#_Toc83827606)

[Le jeu 3](#_Toc83827607)

[Réalisation 3](#_Toc83827608)

[Difficulté engendrée pendant la réalisation de mon jeu 3](#_Toc83827609)

[Github 5](#_Toc83827610)

[La Page Menu 5](#_Toc83827611)

[La carte du monde 6](#_Toc83827612)

[Les niveaux 6](#_Toc83827613)

[La page de paramètre 6](#_Toc83827614)

[Graphisme 6](#_Toc83827615)

[Méthode et Manière 6](#_Toc83827616)

[Organisation de mon code 6](#_Toc83827617)

[Le Canevas 7](#_Toc83827618)

[Qu’est-ce un cookie ? 8](#_Toc83827619)

[Sauvegarder le progrès du joueur 8](#_Toc83827620)

[Matrix de la map 9](#_Toc83827621)

[Détection de collision 10](#_Toc83827622)

[Interaction avec les objets 10](#_Toc83827623)

[Méthode de l’objet 10](#_Toc83827624)

[L’input 10](#_Toc83827625)

[Déplacement du joueur 10](#_Toc83827626)

[Résultats 11](#_Toc83827627)

[Bilan personnel 11](#_Toc83827628)

[Conclusion 12](#_Toc83827629)

[Bibliographie 12](#_Toc83827630)

[Annexes 12](#_Toc83827631)

# Introduction

## 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 en neuf mois. 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 et formater les éléments d’un site web. Le JavaScript est utilisé pour animer les éléments HTML, l’HTML pour créer les éléments et le CSS pour formater les éléments. J’ai utilisé ces trois langages pour réaliser mon jeu vidéo, en utilisant majoritairement le JavaScript. J’ai choisi l’environnement comme thème principal de mon jeu car c’est un sujet pertinent et important et je voulais que mon jeu ait un lien avec la vie réel. Mes objectifs étaient de programmé un jeu vidéo en 2D en utilisant l’élément du canevas sur HTML.

## L’informatique

L’informatique est la théorie et traitement de l’information à l’aide de programmes mis en œuvre sur ordinateurs. L’informatique 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 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 jeu

Le jeu vidéo que j’ai programmé 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 et apporter la solution pour 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 et 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 *bugs.* Les niveaux se débloquent au fur et à mesure que le joueur progresse dans le jeu.

# Le jeu

## La Page Menu

La page menu 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.

## La carte du monde

## Les niveaux

## 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.

## 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 devant trouver des images 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 car je n’étais pas capable de trouver des images correspondantes .

# 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 chargent et nomment toutes les images, le canevas, les audio et les fichiers \*index.js. Ils cachent aussi les images et décrivent les différents éléments présents dans la page affichée dans le navigateur.

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() a 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 a ceci :

Text

Description automatically generated

Le ./ importe un fichier qui réside dans le même dossier que game.js et le ../ importe les fichiers qui résident en dehors du dossier. A redéfinir !!!!!!!!!!!!

Une classe est ensuite définit. Elle comporte un constructeur et les différents fonctions comme start(), update() et draw(). La fonction start() est utiliser pour appeler ou nommer certaine variable quand le page est ouverte. La fonction update() me 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. A redéfinir !!!!!!!!!!!!

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 a chaque script.

## Le Canevas

Le canevas est un élément HMTL ou l’on peut dessiner des images et des objets. Pour dessiner sur le canevas on doit définir contexte du canevas comme étant en 2d. La variable du contexte se nomme ctx. Pour le système de coordonné du canevas le 0 de l’axe X et de l’axe Y est en haut a 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.

## 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 a la fin de votre session et les deuxièmes sont garder jusqu’à leur date d’échéance.

## La sauvegarde du joueur

Le progrès du joueur est sauvegardé dans des cookies. En utilisant un code optimiser et des functions bien precises, 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 constructor et des fonctions. Le constructeur est composé de quatres array : newPlayerProgress, newPlayerVariables, playerProgress et playerVariables. Les deux premiers stocks les variables et les valeurs de base et les deux autres sont vides. Il y a quatre fonctions, getSavedPlayer, updatePlayerVariables, getCookie et changeCookie. GetSavedPlayer écrit les cookies si aucun jeu sauvegardé n’est detecté. Si il y a un jeu sauvegardé, la fonction appellera la fonction updatePlayerVariables. Text

Description automatically generated

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

Text

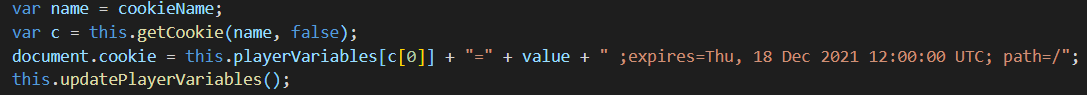
Description automatically generated

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.



## Matrix de la *map*

Un matrix de chiffre est utilisé 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. Le matrix 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 a ceci :

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

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

Text

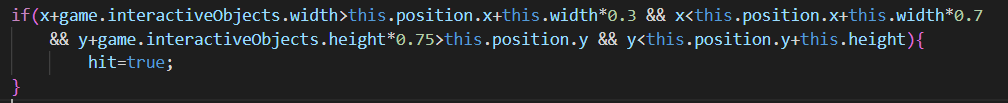
Description automatically generated

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 tout les variables avec *this* définissent le joueur. Chaque conditions est associé au carré avec le même numero. Le signe < signifie que chiffre est plus petit qu’un autre ( numbre < 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

## 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 concerné.

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 a droite ou a 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

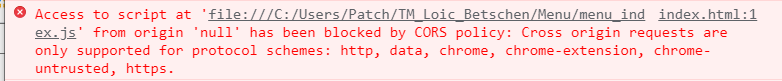
Description automatically generated

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.

## Github

Github est un service de gestion de versions basé sur le web pour les développeurs. J’ai utilisé ce site pour pouvoir accéder à mon code partout et publier mon jeu sur un server ce qui a permis de régler un souci de contrainte HTML.

## 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 rencontrer est celui d’un problème de politique CORS(Cross-Origin Reference Sharing). 

Cette politique a interdit mon fichier index.html a accéder au fichier \*.js car mon disque dur n’était pas définit 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 probleme était celui de la resolution 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 penser utiliser un fichier .text pour sauvegarder le progrès du joueur mais cela je devais utiliser un base de donnée. Heureusement en faisant un peu de recherche je suis tomber sur les cookies. Les cookies sont beaucoup plus simples a utiliser qu’un base de donné et c’est plus rapides de recupere, modifier et supprimer les variables du joueur.

Quand j’ai commencer a introduire plus d’images dans le jeu, j’ai eu un problème de telechargement des images. Au début j’appeler mes images dans les fichier \*.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 ce sont pas charger 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’element. 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 jeux 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.

* Images not complying with what I wanted
* Offset of map
* Player animation
* Reading a text file
* Player collision
* Settings scroll buttons

# Résultats

Jsp

# 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 a 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.

# Conclusion

# Bibliographie

# Annexes