Ecole internationale des Sciences du Traitement de l'information

Classe préparatoire intégrée deuxième année

EISTI'GAME

Rapport de projet

Auteurs :
M. Théo Sagliot
M. Alexis Vandoit
M. Quentin Ducasse
M. Arthur Carré

Encadrant :
Pr. Elisabeth Ranisavljevic

Rapport du
15 avril 2021

Table des matières

1 Présentation du projet **2**

2 Jeu **3**

2.1 Présentation 3

2.2 Fonctionnement 4

2.2.1 Organisation de la page php 4

2.2.2 Choix Algorithmique 5

2.3 Difficulté rencontrées 6

2.4 Limitations et Améliorations possibles 6

3 Partie opératoire **7**

3.1 Connexion 7

3.2 Inscription 7

3.3 Jeu 7

3.4 Joueurs 8

3.5 Profil 8

3.6 Messagerie 8

3.7 Limitations et Améliorations possibles 9

1 Présentation du projet

Cette année pour notre projet de developpement Web nous avons eu le choix entre plusieurs sujets. Ces sujets sont les suivants **EISTI'GAME**, **EISTI'BOOK** et **EISTI'TRANET**.

Le projet que nous avons choisi est l'EISTI'GAME et a pour but de créer un site internet permettant de jouer à un jeu et établir un classement des meilleurs joueurs. Le cahier des charges qui nous a été donné n'impose aucune restriction sur l'esthétique du site. l'EISTI'GAME nous a paru être le sujet qui était le plus intéressant à programmer, le plus libre et le plus enrichissant sur le fonctionnement d'un tel site.

Notre site doit être fonctionnel et agréable à utiliser (ergonomique). Nous avons eu le choix du jeu à réaliser. Cependant, il doit utiliser les technologies web vues en cours et a dû être validé par notre enseignant Elisabeth Ranisavljevic.

Ce site doit proposer différents "types" d'utilisateurs tels que :
- des visiteurs
- des inscrits
- des adminitrateurs

Ayant chacun leurs droits et ainsi plus ou moins de liberté pendant leur utilisation du site.

Le site permet aux différents utilisateur de pouvoir gérer et/ou modifier leur profil, ainsi qu'avoir accès a une messagerie interne pour pouvoir communiquer entre les différents inscrits.
Pour ce qui est des administrateurs ils doivent pouvoir avoir les droits "classiques" qu'un administrateur a au sein d'un site (liste des inscrits, suppression de profil, bannissement d'utilisateurs ect..).

- *"Si vous avez un peu de patience, vous découvrirez qu'on peut utiliser les immenses ressources du web pour perdre son temps avec une efficacité que vous n'aviez jamais osé imaginer."* Dave Barry.

2 Jeu

2.1 Présentation

Le jeu que nous avons choisi de programmer pour notre site EISTI'GAME est inspiré de l'easter egg *Zerg rush* implémenté par la firme Google, lui même inspiré de Starcraft 1 et 2, des jeux créés par Chris Metzen et sortis respectivement en Novembre 1998 et juillet 2010.

Faisant référence à une stratégie de jeu basée sur les "*Zerg*", une tactique permettant de produire un grand nombre de petites unités dès le début d'une partie afin d'attaquer rapidement les adversaires. La tactique consiste donc à attaquer en grand nombre d'où le terme " Rush " qui signifie ruée.

Cette tactique de jeu est devenue mondialement connue, et dans le vocabulaire d'internet d'aujourd'hui être "Zerg rushed" signifie être en infériorité numérique et être débordé. Cependant même si notre jeu se veut inspiré du célèbre Zerg Rush, il n'en reste pas moins différent, pour programmer ce jeu, nous n'avons pas utilisé de code source, nous avons juste été inspiré d'un point de vu uniquement "visuel" pour ensuite en changer le but et l'histoire.

Le but n'est pas de défendre des liens d'une page google mais de se défendre soit même de l'invasion des "Vouzours" dont le fameux "Zerg" en est le chef. Vous avez pour seule arme votre click gauche, autant dire que vous avez interêt à être rapide.
Mais vous découvrirez tout cela en jouant au jeu !
Amusez-vous bien !

Une unité "Zerg"

2.2 Fonctionnement

2.2.1 Organisation de la page php

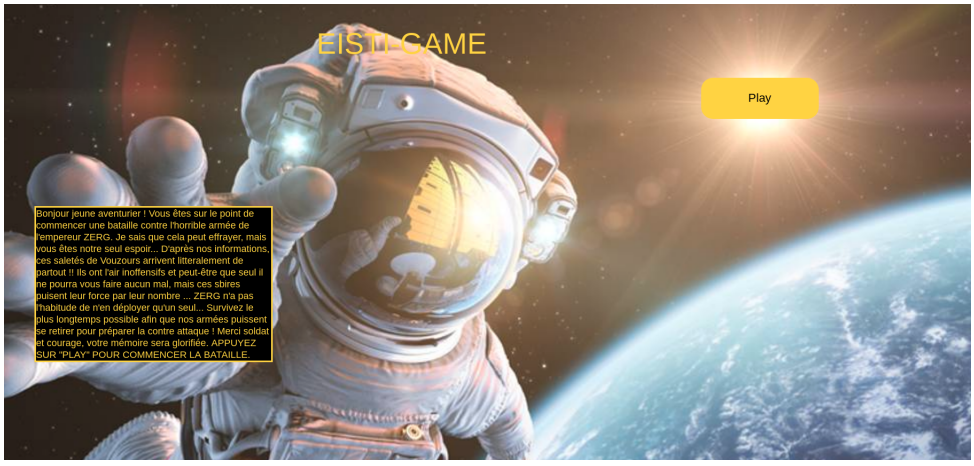
La page 'jeu.php' est divisé en plusieurs blocs principaux :

— Le bloc d'introduction : Ce bloc regroupe tout l'affichage d'intro avant de lancer le jeu, nous avons choisi d'y insérer un bloc de texte présentant l'histoire globale du jeu pour que l'utilisateur puisse d'immerger dans l'univers du jeu.

— Le bloc principal : C'est le bloc "in-game", il n'apparait qu'une fois que l'utilisateur a lancé sa partie, Le bloc regroupe donc le terrain de jeu et l'affichage en jeu, le joueur peut donc jouer, observer son score évoluant en cours de partie ou encore appuyer sur "pause" si il le désire.

— Le bloc de fin : Ce bloc regroupe l'affichage des scores à la fin du jeu, cet affichage regroupe donc :

1. Le score total de fin de partie.
2. Le nombre de monstres tués.
3. le bouton restart si l'utilisateur veut rejouer.



Page d'introduction de la page du jeu

2.2.2 Choix Algorithmique

Pour créer ce jeu, nous avions besoin d'un plateau de jeu dynamique. Nous avons réfléchi sur le sujet et avons finalement décidé d'utiliser la balise html "<canvas>" qui permet de créer une zone sur l'écran sur laquelle on peut dessiner pour réaliser des animations.

Pour nous familiariser avec ce nouvel élement nous avons suivi un tutoriel javascript sur les animations 2D que nous avons trouvé sur internet. Ce guide nous a permit de créer un jeu de case brique étape par étape et de bien comprendre comment utiliser le "canvas".

Nous devions maintenant trouver la manière avec laquelle l'utilisateur allait pouvoir intéragir avec les éléments du canvas. Dans une volontée de garder le même principe que l'easter egg Zerg Rush, nous avons choisi d'utiliser le clique gauche de la souris comme seul moyen d'intéraction entre l'utilisateur et les différents éléments du jeu.

Nous nous sommes donc inspiré d'un jeu de case brique pour gérer le déplacement des monstres de notre jeu. Pour créer l'animation d'un monstre nous avons compris (grâce au guide précedemment cité) qu'il suffisait de dessiner ce monstre à des intervalles de temps réguliers (toute les 10 milisecondes) à des positions différentes. Tout cela à bien entendu été programmé en JavaScript.
Pour gérer le déplacements de plusieurs monstres en même temps, nous les avons stockés dans un tableau. Car nous avions des erreurs à partir du moment ou nous avons voulu passer de la gestion d'un monstre à la gestion de tous les monstres en même temps, mais nous vous expliquerons cela dans la partie qui va suivre.

Dans une volonté de pouvoir afficher le meilleur score directement sur la page de jeu, nous avons tout de suite pensé à l'AJAX pour réussir cet objectif. Le score ainsi que le meilleur score de l'utilisateur est donc directement affiché pendant la partie, si l'utilisateur veut quitter la page de jeu, ce n'est pas un problème il pourra retrouver son meilleur score sur le page de son profil.

Au lancement de la page jeu.php, seul le bloc d'introduction est apparent. Les autres sont transparents et donc invisibles.
Lorsqu'on lance la partie (en cliquant sur le bouton 'Play'), le bloc intro devient invisible et le bloc principal devient visible. Jusqu'à ce que l'utilisateur termine sa partie, c'est alors le bloc de fin qui devient visible et les deux autres deviennent ou restent invisible.

2.3 Difficulté rencontrées

Comme cité précédemment, notre première difficulté rencontrée a été de stocker tous les monstres actifs dans un tableaux et gérer leurs déplacements indépendemment. Nous avons eu beaucoup de mal à transformer notre code pour gérer chaque déplacements de chaques montres. En effet lorsqu'un monstre apparaissait, l'ancien disparaissait ce qui était très problématique. Finalement, après une semaine nous avons réussi à adapter le code entier en insérant un tableau regroupant tous les montres.

Par la suite, nous avons assez rapidement eu une nouvelle difficulté lorsque nous nous sommes intéréssé à la *HitBox* de chaque monstre, c'est à dire la surface cliquable d'un monstre. En effet cela a été assez compliqué de détetecter la position exacte de la souris sur un monstre en mouvement sans que cela soit imprécis. Après de nombreux tests nous avons finalement trouvé une formule générale permettant de détecter des positions de la souris sur un monstre de la manière la plus précise possible. De même, pour détecter la fin de la partie, lorsque les monstres atteignent le centre il nous a fallut réaliser de nombreux test pour que cela fonctionne. De plus, nous avions un problème quand à potentiellement *Glitch* (dans le domaine des jeux vidéos un glitch désigne un bug qui peut être exploité par les joueurs pour tricher), en effet, lorsque l'utilisateur positionnait son curseur au centre de la boule rouge à défendre, et qu'il cliquait en continu, il ne pouvait pas perdre.
Comme pour le problème précédent et après plusieurs tests nous avons reussi à adapter la hitbox.

Après quelques essais du jeu, nous nous sommes apperçu que certain monstres suivaient la même trajectoire et étaient confondu. Cela ne devait pas être un problème mais nous nous sommes apperçus que l'utilisateur n'avait besoin que d'un seul clique pour "tuer" les deux monstres confondus.
Le fait est que lorsque nous cliquions sur les deux monstres confondus, on rentrait dans la boucle du jeu, et ainsi cela supprimait tous les monstres en un clique. Il nous a donc fallut isolé cette fonction de la boucle et le problème a été résolu.

2.4 Limitations et Améliorations possibles

Tout d'abord nous avons été limité au niveau du design du jeu, nous voulions à la base remplacer les carrés par des images de monstres cependant quand nous avons appliqué une image sur le canva, cette image clignotait très rapidement. Le problème ne venait pas de "taille" de l'image puisque nous avions fait attention à cela. Nous en avons conclu que cela était dû au rafraichissement de la page. En effet, comme nous effacions et redessignons monstres toutes les 10 milisecondes, l'ordinateur n'était pas assez puissant pour pouvoir le faire sans clignotement.

Nous avons donc du abandonner cette idée. Pour ce qui est des améliorations possibles, nous avons pensé à un système de "vie". En effet notre première vision du jeu incluait le fait que chaque monstre possédait une barre de vie, cependant nous avons du nous raviser quand nous avons fait face à tous les problèmes cités précédemment, mais avec plus de temps nous aurions pu implémenter cette fonction.

3 Partie opératoire

3.1 Connexion

La page *connexion.php* est la page sur laquelle l'utilisateur arrive, en autres mots, il s'agit de la page d'accueil. On peut soit accéder à *inscription.php* soit se connecter en tant que visiteur ou avec un compte déjà existant et être renvoyé vers *jeu.php*. En appuyant sur le bouton "Me connecter", une fonction AJAX se lance, on utilise AJAX ici pour faciliter les vérifications du pseudo et du mot de passe en php, avec ceux dans la BDD. Dans l'ordre, on regarde d'abord s'il existe un utilisateur existant avec les mêmes identifiants, puis si cet utilisateur est banni ou non, et on accorde finalement l'accès au jeu.

3.2 Inscription

La page *inscription.php* gère, comme son nom l'indique, l'inscription de nouveaux joueurs, elle ne peut être accédée que depuis *connexion.php*. Sur cette page, un nouvel utilisateur peut rentrer plusieurs types d'informations, seuls le pseudo et le mot de passe sont obligatoires, pour effectuer l'inscription, on utlise aussi AJAX, pour enregistrer le compte dans la BDD. Une vérification importante à faire est aussi de vérifier que le pseudo n'est pas déjà pris, car cela peut perturber les autres pages, notamment Profil et Messagerie.

3.3 Jeu

Cette page concerne la fenêtre de jeu, ainsi que le tableau des 10 meilleurs scores parmi les utilisateurs, On utilise une fonction AJAX pour générer le tableau des meilleurs scores, encore une fois, pour faciliter l'accès à la BDD. On affiche les 10 premiers résultats que nous retourne la BDD, ou le nombre total de scores s'il y en a moins de 10. Lorsque la partie est terminée, on enregistre le score obtenu dans la BDD, encore une fois, avec une fonction AJAX pour faciliter l'accès à la BDD.

3.4 Joueurs

Cette page gère la recherche de joueurs, à l'arrivée, une liste déroulante est alimentée avec les pseudos de tous les joueurs, cliquer sur un des pseudos des joueurs renvoie vers leur profil par un changement d'URL simple, avec une variable GET l'accompagnant. (Voir Profil pour l'explication.) La barre de recherche déclenche une fonction AJAX lorsqu'on clique sur le bouton de recherche, cette fonction récupère tous les pseudos contenant le mot clé, supprime tous les éléments de la liste déroulante, et la ralimente avec les pseudos possédant le mot clé.

3.5 Profil

Cette page affiche le profil des joueurs ou d'un autre joueur que l'utilisateur veut regarder. Si l'utilisateur veut regarder le profil d'un autre, une variable GET sera présente dans l'URL, sinon, si on charge la page avec un variable POST fixée, alors l'utilisateur est en trainde modifier son propre profil, et si aucunes de ces deux conditions est vérifié, alors dans ce cas, on affiche le profil de l'utilisateur. Pour la modification du profil, on génère des zones de texte lorsque l'utilisateur clique sur le bouton, et une fois que l'utilisateur confirme les modifications, on recharge la page avec cette fois-çi une variable POST, qui va alors déclencher la modification de données dans la BDD. Une remarque à faire, on utilise une variable GET pour reconnaître si on regarde la page de profil d'un autre, et pas une variable POST, pour éviter la confusion avec les variable POST qui vont déclencher la modification du profil. Il existe aussi une variété de boutons, qui puevent seulement être vus par certains utilisateurs, les admins sont capables de voir tous les boutons :

- Modifier et Supprimer profil sont visibles seulement par le propriétaire du profil.
- Bloquer Utilisateur est disponible pour tout les utilisateurs. Ce bouton déclenche une fonction AJAX qui rajoute l'utilisateur en question avec le propriétaire du profil dans la table Bloques.
- Bannir Utilisateur n'est visible que par les admins. Il rajoute le propiétaire du profil dans la table de Bannis.

3.6 Messagerie

Finalement, cette page gère la messagerie, on dispose d'une liste déroulante d'utilisateurs et d'une barre de recherche qui fonctionne de la même manière que la barre de recherche dans *joueurs.php*, cliquer sur un des profils aura pour effet d'afficher chaque message de la conversation entree l'utilisateur, et le pseudo sur lequel on vient de cliquer, dans l'ordre du plus ancient au plus récent. On affiche aussi une zone de texte pour pouvoir rentrer des messages et les envoyer, l'envoi et la réception de messages se gèrent toutes les deux par AJAX, pour encore une fois, faciliter l'accès à la BDD, qui contient tout les

messages et leur contenu.

3.7 Limitations et Améliorations possibles

Plusieures recherches *SQL* ne sont pas optimisées, notamment celles où l'ont reprend l'ID d'un utilisateur, il y aurait aussi une possibilité de simplifier la page *profil.php*, où l'on performe beacoup de vérifications (l'utilisateur est-il un visiteur ? est-il banni ? Est-il bloqué ? Etc...). De plus, l'enregistrement des scores n'est pas limité, ce qui veut dire que n'importe quel utilisateur peut entrer un grand nombre de score, ce qui va innonder la BDD, au lieu que dans ce site web, seuls les 10 premiers meilleurs scores sont utilisés. Sur les améliorations, on ne peut pas prévenir un utilisateur quand il a reçu un nouveau message, au mieux, on pourrait créer une nouvelle variable "status" dans la table Messages, pour savoir si le récepteur d'un message à lu ou non un message envoyé. Alors, tant qu'un message n'est pas encore lu, on pourrait rajouter une pastille sur l'entête de Messages, pour prévenir que l'utilisateur a reçu un message qu'il n'a pas encore lu, de même, mettre une pastille similaire à côté du pseudo de l'envoyeur.