HEBMANN Julien IG3

2014-2015

COMPTE RENDU PROJET WEB

Sommaire

Introduction

Dans le cadre du projet web, j'ai décidé de faire un site sur un jeu que je connais bien, Dark Souls. L'intérêt principal est qu'ayant une grande connaissance de ce jeu, je n'ai pas besoin de passer beaucoup de temps à chercher des informations, ou à en comprendre le fonctionnement.

J'ai de plus décidé de faire mon site en anglais, car d'une part cela me permet de travailler celui-ci, d'autre part parce que les informations sont bien plus faciles à trouver sur internet en anglais, et enfin cela constitue un bon entraînement, car la plupart des sites web sont en anglais.

N'ayant que peu fait de web (seulement un site en PeiP, et pas très élaboré), je n'ai pas cherché à faire un site très complexe, afin de rendre le projet dans les temps, tout en faisant quelque chose de correct.

Principe du site

Je n'avais au départ qu'une idée vague du projet et pensais ne faire qu'un site lié à une base de données dont le but était de fournir des informations sur l'univers de ce jeu, à la manière d'un mini-wiki. Néanmoins, j'ai finalement décidé de rajouter une dimension plus intéressante via l'ajout de deux fonctionnalités : un compteur de morts et un compteur de « drop » d'objet.

En effet, une caractéristique majeure de Dark Souls est son côté extrêmement difficile, et dans lequel la mort du joueur fait partie inhérente. Les visiteurs du site peuvent donc partager cette expérience via un bouton leur permettant d'incrémenter le nombre de décès dus à un monstre, un autre joueur, ou encore de cause environnementale (chute mortelle, déclenchement d'un piège, …).

Un autre élément de Dark Souls est le « farm », c'est-à-dire tuer des monstres afin de récupérer des objets. Le système est alors le même, un bouton disposé à côté de chaque objet permet d'incrémenter le nombre d'objets trouvés.

Ce principe pourrait permettre de faire des statistiques potentiellement utiles pour les développeurs du jeu, afin d'équilibrer certaines zones ou ennemis.

Outils choisis

L'hébergement du site et de la base de données m'a posé quelques problèmes. En effet, je recherchais une solution gratuite si possible, et on m'a dit que free proposait ce que je voulais si on est adhérent, ce qui est mon cas. J'ai donc fait une demande le soir même du début du projet, et ma base de données et mon site étaient censés être disponibles sous 48 heures. Au final, ils n'ont été disponibles qu'à partir du 30 mai, ce qui était bien trop tard.

Après trois jours à attendre la réponse de free, je me suis mis à chercher ailleurs et ai trouvé un hébergeur répondant à mes attentes : olympe. Malheureusement, il m'est apparu que l'accès à la base de données et sa modification étaient d'une lenteur phénoménale (plusieurs MINUTES pour chaque action), ce qui rendait son utilisation impossible.

J'ai alors trouvé hostinger, gratuit lui aussi, mais dont l'accès à la base de données est question de quelques secondes. C'est au final cet hébergeur que j'ai gardé.

Au niveau de la base de données, seul MySQL était disponible, et c'est donc ce que j'ai choisi.

A partir du moment où mon site et ma base furent disponibles, j'y ai importé ce que j'avais jusque-là en local, et ai travaillé directement dessus (via phpMyAdmin pour la base de données et FileZilla pour le site). J'ai choisi FileZilla car l'accès FTP est autorisé par hostinger, que j'avais déjà un peu touché à ce logiciel en PeiP, et que je l'avais vraiment apprécié.

Au niveau du site, j'avais d'abord opté pour du HTML et du CSS (car c'était tout ce que je connaissais), et pensais utiliser angular. Néanmoins, je me suis rendu compte que j'avais du mal à maîtriser angular, et ai finalement opté pour du php, que j'ai rapidement compris et apprécié.

Enfin, j'ai utilisé un peu de javascript trouvé sur internet afin de rendre mes tableaux dynamiques. En effet, il est apparu qu'il serait intéressant de pouvoir trier les tableaux en fonction d'une colonne, et que, bien que cela soit disponible avec des requêtes SQL, ça n'est vraiment pas optimal. C'est le seul élément que j'ai recopié directement d'internet, car je n'ai malheureusement pas eu le temps de me former aussi au javascript.

Gestion de la base de données

J'ai passé beaucoup de temps sur la partie SQL, car je voulais que mon site soit le plus complet possible. Je me suis donc aidé du wiki déjà existant afin de connaître toutes les données que je voulais avoir.

Je me suis rapidement rendu compte qu'il ne serait pas possible de rentrer autant de données qu'en contient le wiki, mais qu'elles n'étaient pas toutes utiles vis-à-vis de mon projet. Je n'ai donc par exemple pas d'image pour les objets, ni de description. Malgré cela, j'ai voulu garder les caractéristiques que je juge importantes (par exemple, pour les objets j'ai gardé le nom ; pour les armes j'ai gardé les dégâts infligés par celle-ci, …).

Même en me limitant au niveau des informations, le remplissage de ma base de données m'a pris énormément de temps, car chaque objet, ennemi et endroit du jeu y est présent, pour une base de données d'un total de plus de 1500 lignes !

J'ai commencé par créer mes trois tables principales :

- ENEMY constitué de ID qui est sa clef primaire, de NAME qui est le nom de l'ennemi, de LIFE qui est sa vie, de BOSS qui est un booléen qui vaut TRUE si l'ennemi est un boss, et enfin de DEATH_AMOUNT qui est le compteur de morts face à cet ennemi.
- LOCATION constitué de ID qui est sa clef primaire, de NAME qui est le nom de l'endroit, de BOSSES_NBR qui est le nombre de boss présents dans la zone, de DLC qui est un booléen qui vaut TRUE si la zone est présente dans le DLC, et enfin de DEATH _AMOUNT qui est le compteur de morts dans la zone dont la cause est naturelle ou venant d'un autre joueur (qui ne vient donc pas d'un ennemi).
- ITEM constitué de ID qui est sa clef primaire, NAME qui est le nom de l'objet, TYPE qui est le type de l'objet (cette variable est de type enum(tous les types d'objets possibles)), et de DROP_AMOUNT qui est le compteur de « drops » de cet objet.

J’ai également eu besoin de plusieurs autres tables :

- DOUBLE_BOSS constituée de ID_ENEMY qui est une clef étrangère référençant ENEMY.ID, de LIFE_SECOND_BOSS qui est la vie du deuxième boss, de LIFE_SUPER_BOSS_1 qui est la vie du premier des deux boss si celui-ci entre dans sa deuxième phase plus puissante, et de LIFE_SUPER_BOSS_2 qui est la vie du deuxième des deux boss si celui-ci entre dans sa deuxième phase plus puissante. J’ai décidé de créer cette table plutôt que de modifier la table ENEMY afin qu’elle corresponde au schéma de n’importe quel ennemi, car seulement deux combats de boss sont des combats contre des double boss, et que j’ai préféré les enregistrer à part plutôt que d’avoir des champs vides pour la quasi-totalité des ennemis.
- IS_LOCATED constituée de ID_ENEMY qui est une clef étrangère référençant ENEMY.ID, et de ID_LOCATION qui est une clef étrangère référençant LOCATION.ID : cette table permet de faire le lien entre un ennemi et sa localisation.
- ARMOR, SHIELD et WEAPON, qui sont 3 tables correspondant à un type précis d’objet, car elles ont des attributs propres tels les dégâts pour WEAPON, la défense pour ARMOR ou encore la réduction des dégâts pour SHIELD. Ces trois tables ont également un champ ID_ITEM qui est une clef étrangère référençant ITEM.ID, DURABILITY qui est la durabilité de l’objet, et WEIGHT qui est le poids de l’objet.

J’ai enfin créé une table USER, en espérant avoir le temps de gérer des utilisateurs enregistrés sur le site, mais l’ai supprimée quand je me suis finalement rendu compte que je n’en avais pas le temps.

Ensuite, je me suis penché sur les triggers. A ce moment-là, je me suis rendu compte que phpMyadmin ne gérait pas correctement les clefs étrangères, c’est-à-dire que si on supprimait une ligne de LOCATION par exemple, les lignes de IS_LOCATED dont la clef étrangère correspond à la clef supprimée restaient dans la base de données. J’ai donc dû créer des triggers : SUPPR_ARMOR, SUPPR_SHIELD et SUPPR_WEAPON pour gérer le cas où on supprime des lignes

de ITEM, et SUPPR_LOCATION et SUPR_ENEMY pour supprimer les lignes de IS_LOCATED en cas de suppression sur LOCATION ET ENEMY.

Deux autres triggers m'ont été nécessaires : AJ_BOSS qui incrémente LOCATION.BOSSES_NBR quand on ajoute un boss dans la zone correspondante, et RET_BOSS, qui décrémente ce compteur si on supprime un boss de cette zone.

En plus des triggers, j'ai créé trois procédures ADD_ARMOR, ADD_SHIELD et ADD_WEAPON, qui m'ont permis d'insérer des valeurs directement dans la table ITEM et la table ARMOR, SHIELD ou WEAPON. Ces procédures m'ont fait gagner beaucoup de temps, car ces trois tables contiennent respectivement 240, 44 et 140 lignes ! Elles m'ont de plus permis d'éviter de possibles erreurs d'inattention tel que ne pas rentrer la même ID dans les deux tables.

Enfin, j'ai créé une vue nommée LOCATION_DEATHS constitué des champs de LOCATION ainsi que d'une colonne TOTAL comptant le nombre total de morts dans la zone (morts liées à l'environnement et aux joueurs + morts à cause de tous les monstres de la zone). Cette vue n'a été créée que bien plus tard dans le projet, quand je me suis rendu compte que le nombre total de morts n'était pas trivial à récupérer ; et j'ai préféré créer cette vue plutôt que de faire à chaque fois une requête SQL assez longue.

Gestion du site

Au commencement du projet, je n'avais que quelques bases en HTML et CSS acquises en PeiP. J'ai donc commencé par créer un fichier html pour chaque page dont j'aurais eu besoin. J'ai également cherché la police de mon titre, ainsi que l'image de fond. Puis, en faisant des recherches sur le web, je me suis aperçu que je ne pouvais pas accéder à ma base de données uniquement avec du HTML, mais que c'était possible grâce au PHP, que je me suis donc mis à apprendre. Au final, toutes mes pages sont écrites en php.

J'ai tout d'abord créé ma page de garde, puis ai essayé de faire un menu permettant d'afficher les différentes tables de ma base, puis, au survol de la souris, afficher les sous types disponibles, puis les sous-sous types, etc. au survol du parent correspondant. Malheureusement, cela était au-delà de mes compétences personnelles et j'ai dû m'inspirer de menus trouvés sur internet afin de créer le mien. J'ai également cherché comment le rendre compatible avec les appareils tactiles, mais le résultat final reste un peu décevant, car il faut relâcher le doigt à chaque sélection d'un dossier parent. J'ai ensuite enregistré ce menu dans un fichier menu.php qui est appelé par toutes les autres pages.

Plus tard, en plus d'ajouter le lien vers la page principale, j'ai également ajouté au menu un lien vers une page de recherche.

Celle-ci m'a à nouveau demandé des recherches, afin de déterminer comment marchent les variables GET et POST. Grâce à ces connaissances, j'ai pu non seulement créer cette page de recherche, mais aussi supprimer beaucoup de fichiers correspondants à des sous types via l'envoi de variable. Par exemple, j'ai un fichier Spells.php qui affiche tous les sorts, mais j'ai pu supprimer les fichiers Miracles.php, Sorceries.php et Pyromancies.php et remplacer les liens renvoyant à ces pages par Spells.php?type=miracles, Spells.php?type=sorceries et Spells.php?type=pyromancies. J'ai donc ensuite modifié Spells.php afin de prendre en compte l'arrivée de cette variable.

J'ai également ajouté une fonctionnalité à la page de recherche qui surligne en marron foncé (afin que ça ne soit pas trop visible) les occurrences du texte recherché, afin de montrer où il se trouve dans les résultats retournés. De plus, j'ai interdits les caractères autres qu'alphanumériques, ainsi que les mots pouvant être des requêtes SQL tels SELECT, DROP ou INSERT par exemple, afin d'éviter les injections SQL.

Le schéma de chaque page est globalement le même : on affiche le nom du type choisi, puis on affiche les autres types disponibles dans un menu horizontal, puis on affiche le sous-type sélectionné, puis les autres sous-types disponibles, etc. et enfin on affiche le tableau contenant les données correspondantes. De plus, quelques valeurs sont cliquables, par exemple si on affiche toutes les zones, cliquer sur la valeur dans la zone DLC renvoie sur la page contenant les zones présentes (ou pas, en fonction de la valeur qu'on a cliquée) dans le DLC.

Néanmoins, cela m'a demandé beaucoup de temps pour gérer le fait qu'on doive afficher le type au pluriel (par exemple SHIELDS), alors qu'il est présent dans la base de données au singulier. Les fichiers sont donc au final assez remplis à cause d'exceptions tels SORCERY qui ne devient pas SORCERYS au pluriel, mais SORCERIES, ou encore MULTIPLAYERITEM qui devient MULTIPLAYER ITEMS. J'avais de plus choisi comme convention de mettre les chaînes de caractères dans GET en minuscule et au pluriel, et j'ai donc dû jongler avec toutes ces appellations. De plus, le cas de ORDINARY WEAPONS a été assez complexe. En effet, ce sous-sous-type contient deux sous-sous-sous-types, MELEE WEAPONS et RANGED WEAPONS, qui contiennent chacun plusieurs sous-sous-sous-sous-types. Le problème était que j'avais décidé de respecter les restrictions que je m'étais fixé, c'est-à-dire faire une seule page php au maximum pour un sous-sous-type. La gestion de la valeur de la variable type de GET a été assez complexe, de même que l'affichage du tableau.

Les menus horizontaux m'ont également demandé un temps considérable, notamment pour éviter que les mots ne prennent trop de place et fassent un retour à la ligne, car je voulais des cases de même taille.

Au final, j'ai choisi de faire plusieurs fichiers nommés « menu'type'.php », qui ont les mêmes propriétés dans le css, mais dont la largeur est différente grâce à

une balise style dans le fichier menu’type’.php en fonction du nombre d’éléments du menu.

Enfin, j’ai créé une page 404.php permettant de revenir à la page principale, ou de naviguer dans le site via le menu, toujours présent.

Limites et améliorations possibles

Je n'ai malheureusement pas eu le temps d'implémenter un système d'utilisateurs, ce qui serait utile afin d'éviter, ou au moins de limiter les risques d'ajout de fausses données (notamment le fait de cliquer beaucoup de fois sur un même bouton ADD).

Un autre défaut est que certains objets ne sont pas « droppables », c'est-à-dire qu'on ne peut pas les récupérer en tuant des monstres, et il faudrait prendre ça en compte.

De même, je n'ai pas implémenté de système permettant de changer l'URI, ce qui serait intéressant.

J'ai également essayé de rendre le site le plus sécurisé possible au niveau des injections SQL (notamment dans la barre de recherche), mais je ne suis pas sûr d'avoir réussi à le sécuriser complètement, et je n'ai pas eu le loisir de le sécuriser contre d'autres types d'attaques.

Enfin, j'aurais bien aimé utiliser angular afin de rendre le site plus dynamique, et permettre que ce dynamisme se passe du côté du client, plutôt que du côté serveur.

Schémas

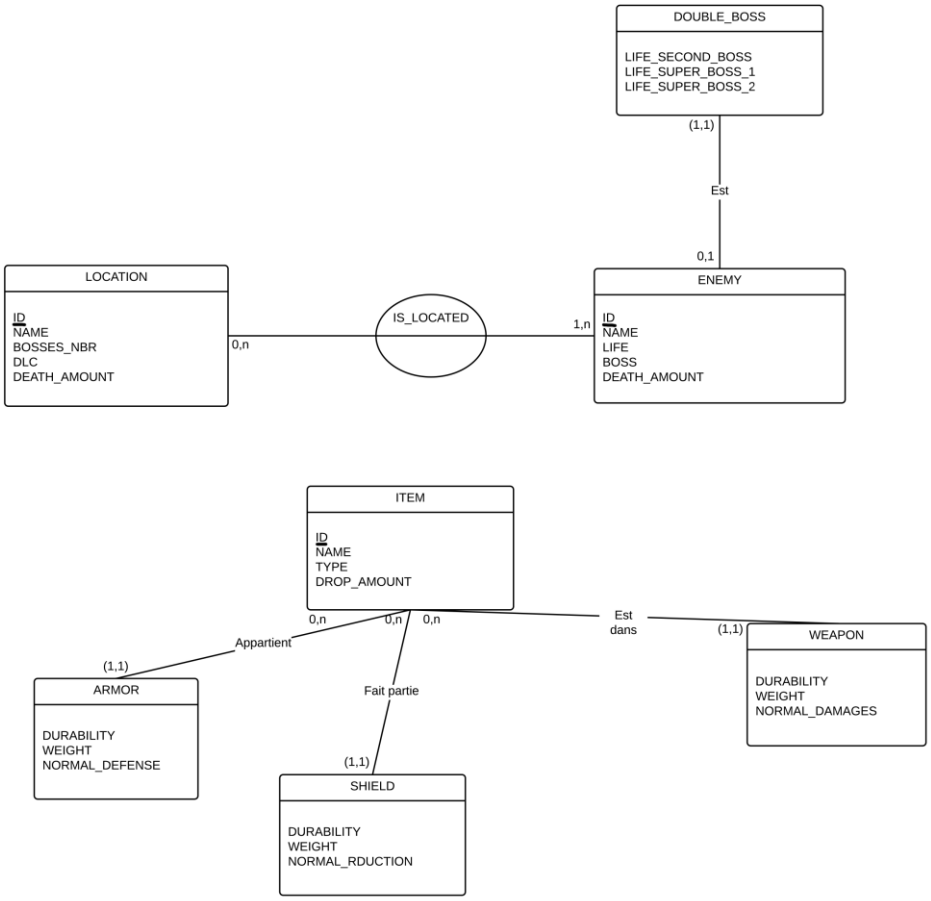

Schéma base de données

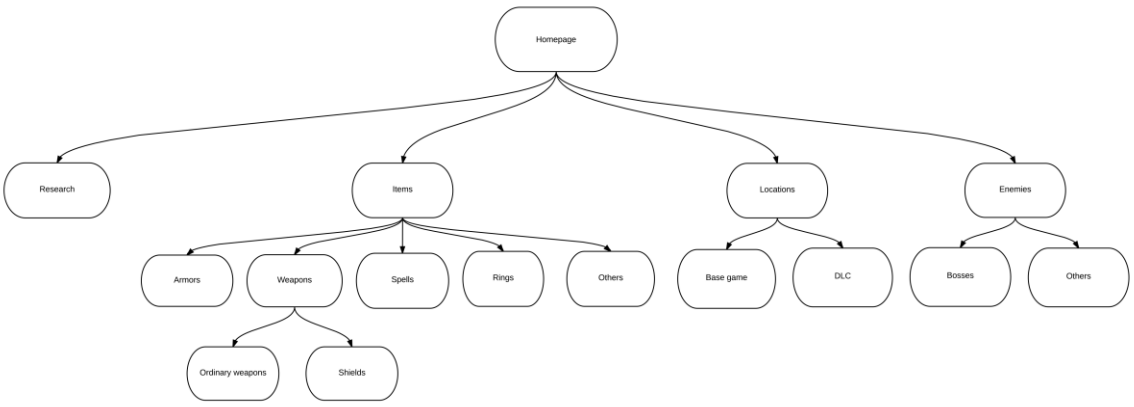

Schéma du site

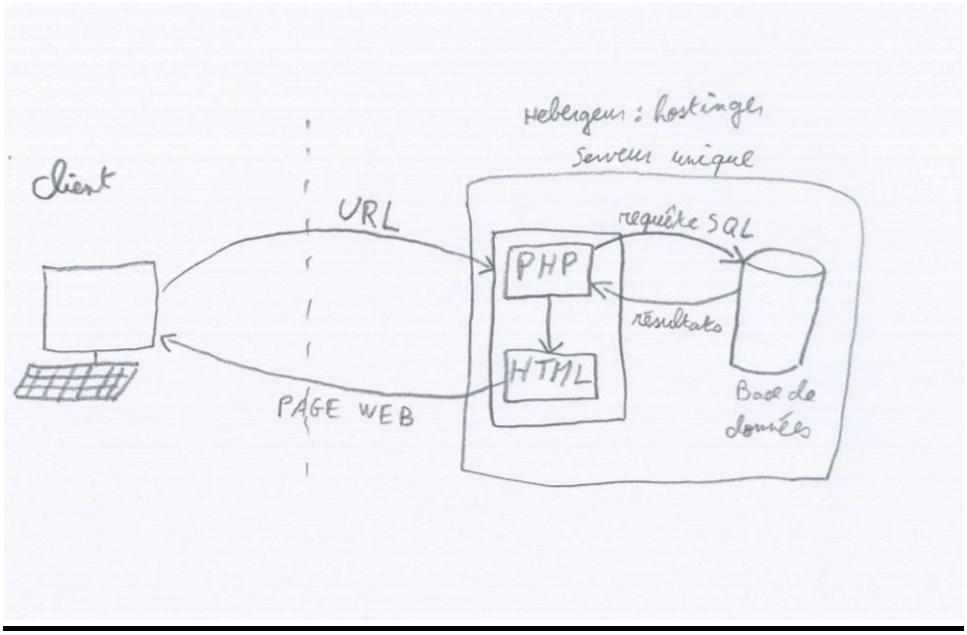

Schéma architecture Web

Sources

http://1-art.eu/images/backgrounds/rice-paper/ricepaper.htm

http://www.w3schools.com/

http://www.dafont.com/fr/darksskyrimfont.font

http://darksouls.wikidot.com/

http://sql.sh/

php.net

dev.mysql.com

stackoverflow.com

http://www.kryogenix.org/code/browser/sorttable/

openclassrooms.com