epsi

l'école d'ingénierie informatique

Rapport de stage

Danois Timéo

Chez Zyzomis

Avec Ludivoc Ressegand,
Président de l'association

Du 8 Mai 2023 au 23 Juin 2023

Table des matières

Remerciement

Je tiens tout d'abord à exprimer ma sincère gratitude envers Ludovic Ressegand, mon maître de stage et président de l'association Zyzomis, qui m'a offert l'opportunité d'effectuer mon stage au sein de leur structure et m'a accueilli chaleureusement. Grâce à sa bienveillance et à sa confiance, j'ai pu découvrir et apprendre énormément de choses tout au long de cette expérience.

J'aimerais également exprimer ma reconnaissance envers Freddy, un ami de première année, qui a partagé cette aventure avec moi au sein de l'association. Notre collaboration et notre entraide mutuelle ont grandement contribué à la réussite de ce stage.

Enfin, je souhaite adresser mes remerciements à toutes les personnes qui m'ont aidé à trouver cette opportunité, notamment à l'équipe pédagogique d'EPSI et à Cameron, un étudiant de troisième année, qui m'a généreusement transmis les coordonnées de Dragonium.

Introduction

J'ai eu l'opportunité de faire mon stage dans l'association Zyzomis pendant 7 semaines.

Elle a été créée en 1997 dans le but de développer la communication par le biais de l'informatique au début d'internet. Les membres de l'association travaillent dans plusieurs domaines comme l'animation évènementielle, les conférences et ateliers de sensibilisation, le streaming d'évènement et l'impression de documents en braille.

Aujourd'hui et depuis 2010 l'association gère et développe le jeu Dragonium. C'est devenu l'activité principale de Zyzomis. Après deux ans et demi de préparation, le jeu a été ouvert le 28 septembre 2012. Ils se sont véritablement engagés pour la cause des déficients visuels. Ils permettent au grand public de découvrir le braille et de faire d'autres expériences à travers des ateliers de sensibilisation par le jeu. Le problème du handicap est peu évoqué dans les jeux vidéo, et les handicaps visuels sont effectivement problématiques sachant que tout se passe sur un écran. Mais Dragonium est pour tous, accessible aux aveugles. C'est un jeu de rôle et de stratégie médiéval fantastique, gratuit qui se joue sur navigateur.

Mon rôle en tant que stagiaire était de mettre en œuvre des idées d'amélioration du jeu et aussi de régler des incohérences ou des bugs. J'avais donc besoin des connaissances en PHP, HTML et CSS que nous avions appris en classe.

Je travaillais du lundi au vendredi, de 9h à 17h30. Nous avions deux à trois réunions par jours avec Ludovic RESSEGAND, mon maitre de stage. Une en début de matinée, où nous discutions des missions de la journée et où il donnait aussi parfois des conseils par rapport au code que j'avais fait la veille. La seconde en début d'après-midi pour voir l'avancement de nos missions. Et la dernière réunion se faisait le soir avant de finir la journée pour débriefer et planifier le lendemain. Il y avait avec moi un autre stagiaire d'EPSI, Freddy MORILLEAU. Nous avons fait ce stage ensemble, nous avions toujours des petites choses à faire, comme du débogage, des petits ajouts ou modifications. Nous travaillions ensemble. Quand Ludovic nous montrait des nouvelles tâches à accomplir, nous nous partagions les missions, et nous

nous entraidions. À mes yeux, durant le stage, j'ai eu trois grandes missions. Premièrement l'ajout d'achievements, des médailles que les joueurs obtiennent en avançant dans le jeu. La deuxième fut l'ajout d'un piège que les joueurs utilisent pour tracker d'autres joueurs. Et la troisième grosse mission a été de faire en sorte que les utilisateurs puissent avoir plusieurs comptes et qu'ils passent facilement de l'un à l'autre. Je détaillerai ces missions ainsi que d'autres plus petites après vous avoir expliqué mes premiers jours de stage.

Les premiers jours en tant que stagiaire

Le premier jour de stage a été consacré à la découverte du code de Dragonium, des logiciels utilisés et de la base de données. J'ai commencé mon stage en même temps que Freddy MORILLEAU. Ludovic RESSEGAND, notre tuteur, a passé la journée avec nous pour nous former sur le fonctionnement du site.

Pour pouvoir tester notre code nous avons eu accès au serveur développeur de Dragonium. Il y a une copie de la base de données et des fichiers php pour qu'on puisse les modifier, ou en ajouter. Ainsi nous pouvions coder et tester notre code sans déranger les joueurs. Nous utilisions Total Commander pour faire le lien entre notre pc et ce serveur.

Le jeu est développé depuis 2010, il y a donc beaucoup de fichiers php, environ 2000 et la base de données compte plus de 150 tables.

Il y a un site qui liste tous les tickets de Dragonium, c'est comme cela que nous appelons les taches qu'il y a à réaliser. Les tickets peuvent être des idées de mise à jour comme des problèmes qu'un joueur a fait remonter à un administrateur. Grace a cela, si jamais nous finissions en avance nos missions, nous pouvions aller consulter la liste pour commencer une tache qui n'avait été attribuée à personne.

Une fois que nous avions accès à tout et que nous avions compris le fonctionnement des logiciels nous avons commencé à faire notre premier ticket avec Ludovic. Pour le comprendre je dois vous mettre dans le contexte

du jeu. Il y a sur l'île de Dragonium deux princesses Birgit et Morrigan, héritières du trône. Elles ont été séparées suite aux guerres fratricides et aux complots menés par leurs parents. Cela a donné lieu a deux factions, et chaque joueur doit faire un choix entre les Loups de Birgit et les Griffons de Morrigan, qui se livrent une guerre sans merci. Dans ce jeu de rôle les joueurs peuvent faire du commerce ou s'allier avec d'autre pour combattre monstres ou ennemis. Notre première tâche consistait à faire en sorte qu'on puisse voir le blason de faction des joueurs dans le chat et dans la liste des joueurs en ligne.

Pour faire cela nous avons réfléchi à une façons simple, peu encombrante et claire pour obtenir le résultat escompté. Pour la liste des joueurs en ligne nous avons décidé de rajouter à côté des noms un « **G** » rouge car c'est la couleur des Griffons de Morrigan ou un « **L** » bleu, la couleur des Loups de Birgit. Quand on passe la souris sur ces lettres nous affichons aussi une bulle avec une petite phrase comme « Karma Loup, pour Birgit ! ». Et pour le chat nous avons décidé de faire la même chose dans la liste des joueurs connectés au chat. Ainsi, si un joueur veut faire un marché avec un autre et qu'il se demande à camp il appartient, il lui suffit de regarder la liste qui est toujours affichée en haut à droite du chat.

Quand dans le code on sélectionnait le nom des joueurs dans la base de données nous avons aussi récupéré le karma de ces joueurs. Le karma est le moyen de connaitre la faction d'un joueur. C'est un nombre qui va de 0 à 99,999 qui est calculé par rapport aux actions des utilisateurs. S'il est inférieur ou égal à 49. 999, alors c'est le camps des Loups, sinon, s'il est supérieur ou égal à 50, alors c'est le camps des Griffons. Chaque joueur voit son karma afficher en dessous de la map, plus il fait des actions pour sa faction plus le karma se rapproche de 0 ou 99.

Une fois que nous avions compris cela et que nous avions récupéré le karma nous nous sommes servis d'un if ($karma <= 49. 999) et d'un else if ($karma >= 50) pour afficher la lettre. Cela donne un résultat comme cela :

.Vous remarquerez que certains, comme Skyline, ont un point d'intergation. Effectivement, nous avons rajouté un else car les administrateurs ont un karma neutre qui ne rentre pas dans nos deux précedentes conditions.

En faisant une tache simple comme celle-ci nous avons appris à trouver le fichier php dont nous avons besoin, à trouver la partie du code qui nous intéresse et à récupérer des informations dans la base de données.

Apres cette premiere mission Ludovic nous à donné à Freddy et moi une dizaine de taches semblables pour que nous nous fassions la main.

Parmis celle-ci je dois parler d'une qu'il ma confié, qui est importante pour comprendre le moyen de traduction de Dragonium. En effet le jeu est en

Français et en Anglais. Sur la page d'accueil il y a des sondages qui sont affichés. Mais les questions ainsi que les reponses s'affichaient en Français et en Anglais, comme vous pouvez le voir :

En fait tous les textes du jeu sont stockés dans une table dans la base de données et ils sont dans les deux langues, de cette facon : « Texte en Français.§ Text in English. ». Et quand un joueur se connecte ou change de langue alors on récupère tous les textes dans la langue desirée et on les stocke dans une variable de session. Pour cela on fait un select de cette manière : « SELECT SUBSTRING_INDEX(SUBSTRING_INDEX(texte, '§', ".$_SESSION['lang']."), '§', -1) AS texte ».

Pour afficher les textes, on se sert de la variable superglobale qui est un tableau de tous les textes qu'on a stockés. Ainsi, en se servant de cette variable avec l'id du texte qu'on veut afficher, le jeu affiche dans la langue sohaitée.

Et cette façon de traduire en séparant l'anglais et le francais avec un « § » est valable dans tous les noms dans la base de données, que ce soit les noms de pnj, de villes, d'objet, ou les questions et réponses des sondages.

C'est donc en modifiant la requête SQL utilisée pour récuperer les textes des sondages, et en rajoutant les SUBSTRING_INDEX que j'ai pu afficher les sondages correctement :

A la fin du deuxième jour, on nous a demandé de trouver des idées de médailles que les joueurs pourraient gagner comme des trophées. C'est là que commence la première grosse mission qu'on nous a confié à Freddy et moi.

Les Achievements

Une grande majorité des jeux vidéo utilise le système des achivements, médailles ou trophées, pour récompenser les joueurs qui passent du temps sur leur jeu. Par exemple, sur la plateforme de distribution Steam, ils utilisent cette méthode qu'ils appellent des « succès ». Ceux-ci peuvent être débloqués en avançant dans l'histoire principale du jeu ou en faisant des quêtes secondaires. Et quand un joueur regarde le compte Steam de son ami il peut voir les succès qu'il a débloqués. Quand certains aiment un jeu et qu'ils ont fini l'histoire principale, ils peuvent rester sur ce même jeu des heures de plus pour obtenir les succès qu'ils ne possèdent pas encore.

Dragonium avait donc comme objectif de rajouter des médailles qui seraient obtenables de toutes sortes des façons, qui récompenseraient l'avancement des joueurs dans leurs quêtes, mais aussi les joueurs actifs qui parlent dans le chat et qui contribuent à faire vivre le site. Il fallait que chacun puisse voir les médailles des autres et qu'elles se distribuent automatiquement. On nous a demandé au départ, à Freddy et moi, de trouver une vingtaine d'idées médailles.

Création des médailles

Nous avons d'abord écrit des idées simples qui nous venaient en tête naturellement comme « Tueur de monstre ». La map de Dragonium est remplie de monstres à combattre et cette médaille s'obtient après en avoir battu un certain nombre. La médailles « Explorateur » est aussi un classique des succès d'un jeu, une récompense qu'on recevrait après avoir exploré une grande partie de la carte.

Nous nous sommes aussi aidés de la base de données pour regarder les informations qu'on stockait sur les joueurs. Avec ces données on peut facilement trouver des objectifs. Par exemple, chaque joueur peut parrainer d'autres joueurs à leur inscription. Quand vous incitez un ami à se créer un compte il peut vous enregistrer comme parrain. On a donc fait la médaille « le Parrain », qui récompense les joueurs qui ont ramené le plus

d’utilisateurs. Ludovic nous a aussi aidés à valider ou non des médailles. On en avait noté une qui serait obtenue après qu’un joueur soit mort un grand nombre de fois. Mais celle-là n’a pas été retenue car elle aurait pu inciter les joueurs à se suicider en boucle. Elle s’est cependant transformée en « trompe la mort » une médaille qui se calcule en divisant le nombre de morts pas le nombre de monstres tués. Cela nous donne un ratio et les joueurs avec un faibles résultat, ceux qui sont donc morts peu de fois par rapport au nombre de monstres qu’ils ont vaincus, obtiennent cette médaille. Nous avons quand même mis comme condition que le joueur devait avoir vaincu plus de 10 000 monstres, car si un joueur qui commence le jeu bat sont premier monstre sans mourir il obtiendrait directement une médaille qui devrait être compliquée à avoir.

Nous avons aussi décidé que pour chaque médaille il y aurait 4 paliers différents, la médaille de cuivre, de bronze, d’argent et d’or. Le premier palier est simple à obtenir, il se fait naturellement. Ainsi un grand nombre de joueurs peuvent avoir la médaille de cuivre. Mais plus on avance et plus l’écart entre les paliers grandit. Comme cela, les médailles d’or demandent d’être actif dans le jeu. Pour déterminer les paliers de chaque médaille nous avons regardé les chiffres qui la concernent dans la base de données et avons comparé les données des joueurs. La médailles « Tueur de monstres » a donc son premier palier à 10 000, puis 50 000, 100 000 et pour finir la médaille d’or est à 500 000 montres tués.

Certaines n’ont qu’un seul palier, on donne directement la médaille d’or. Par exemple j’ai expliqué plus tôt le système de faction et comment fonctionne le karma. Nous avons fait les médailles « Pour Morrigan ! » et « Pour Birgit ! ». Elles s’obtiennent quand le joueur réussit à avoir un karma de 0 ou de 99.999. Comme elles sont compliquées à avoir on ob,tient la médaille d’or et on la garde.

Effectivement je précise que ces médailles sont définitives car nous avons deux types, des médailles définitives, que l’on garde une fois obtenues, et d’autres qu’on peut perdre ou rétrograder. Comme la médaille « Trompe la mort » si un joueur a la médaille d’or mais qu’il meurt en boucle sans battre de monstre, alors il peut la perdre et obtenir la médaille d’argent.

Médailles de classement

Il y a aussi des médailles qu'on a surnommées « médailles de classement ». Car Dragonium a une table dans la base de données qui enregistre pour chaque joueur actif des informations sur ce qu'il a fait pendant le mois. Comme le nombre de soins qu'il a donnés ou le nombre de villes qu'il a attaquées. Grace à cette table nous avons créé des médailles qui chaque mois changeraient de propriétaire. Par exemple celle du meilleur soigneur : le joueur ayant donné le plus de soins reçoit la médaille d'or, le deuxième d'argent, le suivant de bronze, et le quatrième de cuivre.

Tables de la base de données

Après avoir trouvé un peu plus d'une vingtaine de trophées et leurs paliers nous avons créé deux tables dans la base de données.

La première ci-dessous qui enregistre les informations de chaque médaille :

Colonne	Type	Commentaire
id_medaille	mediumint(8) unsigned *Incrément automatique*	
nom_medaille	text	
image_medaille	tinytext	
description_medaille	mediumtext	
palier1_medaille	decimal(11,2) unsigned	Cuivre
palier2_medaille	decimal(11,2) unsigned	Bronze
palier3_medaille	decimal(11,2) unsigned	Argent
palier4_medaille	decimal(11,2) unsigned	Or
inverse_medaille	tinyint(1) unsigned [**0**]	1 = Si INFÉRIEUR au palier (0=si supérieur
temporaire_medaille	tinyint(1) unsigned [**0**]	1 = Supprimée chaque 1er du mois

-Le nom et la description des médailles écrit en français et anglais séparé de « § »

-Le nom de l'image de la médaille

-Les différents paliers

-Inverse_médaille est utilisé pour l'affichage des médailles. Si égal à 0 alors les paliers sont croissants, sinon, quand il est égal à 1, alors les paliers sont décroissants

-temporaire_medaille est utile quand on donne ou supprime les médailles. Il nous indique si la médaille est traitée comme une médaille de classement ou une médaille ordinaire.

La seconde table sert à faire le lien entre les joueurs et les médailles qu'ils possèdent.

Colonne	Type	Commentaire
joueur_medaille_joueur	bigint(20) unsigned	
medaille_medaille_joueur	mediumint(8) unsigned	
valeur_medaille_joueur	decimal(11,2) unsigned	Si nécessaire, on enregistre une valeur
valeur_complementaire_medaille_joueur	decimal(11,2) unsigned *NULL*	Si nécessaire, on enregistre une valeur complémentaire

valeur_medaille_joueur sert à enregistrer les informations du joueur, comme par exemple le nombre de monstres qu'il a vaincus, et valeur_complementaire_medaille_joueur sert a enregistrer des informations du même style mais plus précises pour certaines médailles comme la médaille « Tueur de Dracoliche » dont je parlerai plus tard. Ces champs peuvent paraître flous pour l'instant mais vous allez comprendre.

Programmation de la distribution des médailles

Une fois les tables créées nous avons commencé à réfléchir au programme qui distribue les médailles aux joueurs. Notre maitre de stage nous a expliqué que Dragonium utilisait plusieurs crons et que nous allions devoir en faire un nouveau pour programmer la distribution des médailles. Un cron est un programme que le serveur exécute automatiquement à l'heure voulue. Ça peut être une fois par minutes, par jour, par mois, ça dépend de comment on le paramètre. Nous avons donc décidé de faire un cron qui s'exécutera chaque jour à minuit.

Pour le codé, comme nous étions deux voire trois à travailler dessus, nous avons utilisé un site « codeshare » qui permet de coder en plusieurs langages à plusieurs en même temps. C'est un site qui est pratique bien qu'il y ait quelques problèmes d'indentation et certains bugs d'affichage quand un autre utilisateur supprime une partie de ce qu'il a écrit.

Dans notre cron nous sélectionnons toutes les médailles de la base de données et tous les joueurs actifs. Comme il y a beaucoup de champs dans la table des joueurs nous récupérons juste les champs utiles. Nous rangeons chaque médaille dans un array appelé « arrayMed ». Grace à la boucle while ligne 44 nous allons pour chaque joueur répéter la distribution.

Nous utilisons la boucle for ligne 46 pour regarder chaque médaille. Chacune d'elle est différente bien sûr, mais le traitement est presque identique. Je vais donc expliquer la première, le médaille tueur de monstres, celle qui a l'id 1 (annexe 1).

On regarde si on est à la boucle 1 et si le joueur a tué assez de monstres pour avoir le premier palier de la médaille. S'il entre dans le if alors on regarde si il a déjà eu la médaille. S'il ne l'a pas eue, nous la lui créons dans la table qui fait le lien entre les joueurs et les médailles. On met son id, l'id de la médaille qui dans notre cas est $bcl et on enregistre le nombre de monstres que le joueur a tués dans le champ valeur_medaille_joueur qui sert à l'affichage.

Si le joueur a déjà une médaille mais que son nombre de monstres tués a augmenté alors on UPDATE le champ valeur_medaille_joueur de sa médaille. Ainsi il est à jour.

Une fois qu'on a utilisé ce système pour les médailles normales on regarde si on est le premier du mois. Si oui alors on va affecter les médailles temporaires. Nous supprimons déjà les médailles temporaires dans le cron qui initialise la table avec les classements des joueurs. Donc il nous suffit de sélectionner les médailles qui ont temporaire_medaille=1. On les range dans un tableau. Ensuite dans une boucle for nous regardons pour chaque médaille, le classement qui la concerne (annexe 2).

Nous voulions utiliser la fonction ROW_NUMBER mais elle ne fonctionnait pas sur la base de données. Nous avons donc simulé une fonction de numérotation des lignes en utilisant une variable utilisateur (@row_number) et en l'incrémentant à chaque ligne, dans le select grâce à « (@row_number:=@row_number + 1) AS row_num ». En faisant comme cela nous obtenons les quatre premiers du classement avec leur position et leur id. Nous pouvons donc maintenant leur enregistrer une médaille avec leur id de joueur, l'id de la médaille et leur place dans le classement.

Une médaille nous a posé problèmes, elle s'appelle tueur de Dracoliche. Les dracoliches sont des dragons, des créatures imaginaires qui ont l'apparence d'un dragon squelettique ou en décomposition très avancée (annexe 3). Il existe dans Dragonium 5 types de dracoliches aux couleurs différentes, blanche, rouge, verte, bleue et jaune. Nous devions donc créer une médaille qu'on reçoit après avoir tué deux types diffèrents, puis à chaque nouveau type battu on débloque le palier suivant. Mais nous n'avions aucun champ dans la base de données qui enregistrait ces informations. De plus un joueur peut tuer plusieurs fois une dracoliche de la même couleur. Nous devions donc trouver un moyen d'enregistrer quelle couleur un joueur a battue. Pour ce faire nous avons utilisé valeur_complementaire_medaille_joueur dans la table qui lie les joueurs et les médailles. Nous avons utilisé le binaire, avec 5 bits, chaque bit égal à 1 signifie que la dracoliche est tuée. Vert est le premier bit, bleu le deuxième, jaune le quatrième et le cinquième bit est la dracoliche blanche. Si nous avons tué la dracoliche blanche et la verte alors ça nous donne 10001 qui est égal à 17 en décimal. Nous engriserons donc 17 dans le champ valeur_complementaire_medaille_joueur et 2 (nombre de dracoliches tuées) dans le champ valeur_medaille_joueur. Et pour cette médaille on ne la traite donc pas avec les autres mais quand un joueur tue un monstre. Quand il gagne on regarde si c'est une dracoliche et qu'elle couleur est cette dracoliche. Ensuite on regarde s'il a la médaille, sinon alors on lui ajoute une médaille avec 1 en valeur_medaille_joueur, et la valeur qui est adaptée en valeur_complémentaire. Effectivement la médaille est créée même s'il n'a pas le premier palier mais ça ne pose pas de problèmes car dans l'affichage des médailles nous ne l'affichons pas dans ce cas. Si le joueur a déjà la médaille alors on met en binaire la valeur complémentaire et on regarde si le bit de la dracoliche tuée est à 1 ou 0. S'il est à 1 nous ne faisons rien, sinon s'il est à 0 alors nous modifions les valeurs de la médaille pour ajouter une dracoliche vaincue (annexe 4).

Affichage des médailles

Une fois que nous avions fini de créer les médailles, nous avons commencé à programmer l'affichage. Après réflexions nous avons décidé que les joueurs voyaient leurs médailles sur leur profil. Ils peuvent aussi voir les médailles d'un autre joueur en cliquant sur le profil de ce joueur quand il est à proximité. Mais on doit laisser la liberté au joueur de rendre public ou non

ses médailles. Bien sûr, quand un administrateur regarde la fiche d'un joueur, il doit pouvoir voir les médailles qu'il a.

Pour le paramétrage nous avons juste ajouté une ligne dans la page option qui demande si on veut qu'elles soient publiques ou non (annexe 5).

Pour afficher les médailles nous avons décidé d'utiliser une image gif qui a un symbole qui nous sert de base, et ensuite nous créons un fond de couleur en php. Dans une boucle nous affichons chaque médaille avec la couleur adéquate. Et comme le jeu doit être adapté aux mal-voyants nous mettons bien le nom de la médaille, sa description, et son niveau dans les alt et title. Ça l'affiche aussi ainsi dans une boule quand on passe la souris dessus. Pour créer les images php et le niveau dans la description nous sélectionnons chaque médaille du joueur avec le champ valeur_medaille_joueur. Bien sûr, si ce champ est plus petit que le premier palier de la médaille recherchée, alors nous ne l'affichons pas. Vous pouvez voir le résultat dans les annexes 6 et 7.

Cette mission a été intéressante à réaliser. J'ai dû réfléchir à des trophées, regarder quel palier serait réalisable mais pas trop simple, et programmer la façon de les donner. J'ai aussi appris à créer des images en php ce que je n'avais jamais fait. Il a fallu réfléchir à tous les endroits où nous devions les afficher. Se mettre à la place d'un utilisateur et aussi penser à toutes les possibilités d'affichage, quoi afficher quand le joueur est en privé, quoi afficher quand il n'a pas de médailles et comment être adapté aux mal-voyants. On peut avoir l'impressions que c'est une petite chose mais avec cette mission j'ai touché à une dizaine de fichier php. Nous avons réussi à répondre aux demandes et à obtenir le résultat attendu.

Ajout d'un piège

Dans Dragonium il existe plusieurs classes : les joueurs peuvent décider d'être un soigneur, un guerrier, un mage, un paladin ou un chasseur. Les chasseurs peuvent avoir trois métiers différents, ils sont tous braconniers et ils peuvent être aussi ingénieurs ou outilleurs. A la base, il existait un piège que les chasseurs étaient les seules à pouvoir poser. Les ingénieurs pouvaient les poser et les outilleurs pouvaient les créer. On a donné à Freddy et moi la mission de faire un nouveau piège pour les chasseurs. L'idée était d'utiliser de la peinture et que le poseur de piège pouvait suivre à la trace la personne qui marchait dedans pendant une courte durée. On nous a aussi dit qu'il faudrait modifier le système de craft et de pose des pièges pour que les ingénieurs et outilleurs puissent le construire et le poser.

Conception et mise en œuvre du nouveau piège

Nous avons commencé par réfléchir avec notre maitre de stage au craft du piège et à l'effet qu'il devait produire. Il en est ressorti que le piège serait de la peinture ultra-violette. Ludovic nous a créé l'objet avec son l'image, vous pouvez le voir annexe 8. Pour le créer il faut un petit engrenage, une carapace mauve, une carapace bleue, de l'eau et des bandelettes d'acier. Nous avons aussi décidé que quand un joueur marche dans le piège il n'est pas prévenu. Sa position doit être indiquée au-dessus de la map du piégeur, et s'il est dans les cases alentour alors on affiche des traces de pas sur sa position.

Comme il existait déjà sur la page des outilleurs un moyen de construire un piège nous avons modifié le traitement du formulaire pour que le pot de peinture ultraviolette puisse se construire. Et nous avons récupéré ce formulaire et le traitement pour le mettre sur la page des ingénieurs, mais comme demandé nous avons juste autorisé le craft du nouveau piège. Vous pouvez voir le rendu annexe 9.

Pour la pose du pot de peinture nous avons procédé un peu de la même manière. Nous avons regardé comment le piège à mâchoire se posait et nous avons adapté les pages avec le pot de peinture. Quand un piège est

posé et qu'un joueur est piégé, nous enregistrons des transactions en base de données. Les transactions servent à suivre les actes des joueurs pour comprendre ce qui se passe quand il y des bugs, des plaintes, ou de la triche. Nous avons aussi modifié le formulaire qui sert a posé un piège car nous avons remarqué certains problèmes. Quand un chasseur a des pièges de posés, ceux-ci sont listés dans les zones où ils sont. Mais la liste ne marchait que pour un piège. Nous l'avons changée pour qu'elle fonctionne correctement. Nous avons aussi rajouté la condition qui fait que quand un joueur n'a pas de piège de posé il est écrit « aucun piège placé ». Ensuite une fois que le formulaire fonctionnait correctement pour l'ingénieur, nous l'avons aussi intégré dans la page de l'outilleur pour qu'il puisse poser le pot de peinture ultraviolette. Vous pouvez voir les différents résultats en Annexe 10 et 11.

Intégration du piège et améliorations générales

La carte de dragonium est très grande mais un joueur voit en permanence la map qui est en annexe 12. Maintenant que le joueur peut poser le piège sur la case de son choix, hors forteresse ou ville, je vais vous montrer ce qui s'affiche sur la map du chasseur quand un autre joueur marche sur son piège. Quand un piège s'active il est supprimé de la case car il ne peut piéger qu'une personne et nous enregistrons en base de données une transaction avec l'id du piégeur, celui de la victime et la date et heure du moment où la victime marche dans la peinture. Grâce à cette date nous savons si ça fait plus de quinze minutes ou moins. Pendant les quinze premières minutes nous affichons au-dessus le mini-map un texte que vous pouvez voir ci-dessous. Il indique la position et le nom de la cible, ainsi même si le chasseur est loin il peut aller jusqu'à elle et savoir où elle est.

Si la victime du piège est dans le périmètre de la mini-map du chasseur alors nous affichons sur sa carte des traces de pas à la position de la cible. Les pieds ainsi que le texte au-dessus de la carte s'adapte en direct à la position de la victime, le chasseur peut donc savoir exactement sa position pendant quinze minutes.

Pendant cette mission nous avons touché beaucoup de page. En modifiant les formulaires de craft des outilleurs nous avons remarqué que le code avait des incohérences et qu'il pouvait être simplifié. Il y avait d'ailleurs un ticket qui parlait d'un problème sur un formulaire de craft d'un autre métier. Et nous avons compris que ce problème venait du code qui était mal fait. Nous avons donc modifié tous les formulaires de chaque métier.

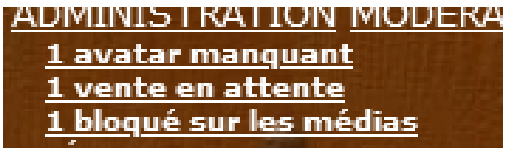
Suppressions mystérieuses des avatars

Depuis plusieurs semaines des joueurs et des administrateurs de Dragonium ont remarqué que certains avatars de joueurs avaient un problème. Ils ne s'affichaient pas et quand on regardait sur le profil il y avait écrit « avatar du personnage » là où il devait y avoir l'image, c'est le alt de l'image. Ce problème vient du fait que l'image n'est pas enregistrée. On nous a demandé de dresser une liste de chaque joueur ayant ce problème d'avatar. Nous avons donc décidé de faire une page qui vérifie pour chaque joueur si l'avatar existe. Nous avons simplement sélectionné toutes les images, les id et les noms de tous les joueurs actifs. Et grâce a la fonction php « file_exists » nous savions si l'avatar était dans les images du serveur ou non. Si l'avatar n'était pas enregistré alors nous affichions l'id, le nom et le nom de l'image dans un tableau. Ça donne ce résultat :

Id	Nom du joueur	Image
13377	Ého	avatar-dazdafsqfqsfefaut.png

Seuls les administrateurs peuvent voir cette page. En cliquant sur la ligne du joueur. Ils sont envoyés sur la fiche du joueur et peuvent leur remettre l'avatar par default et leur envoyer un message pour dire que leur avatar a eu un problème.

Nous avons aussi rajouté sur le menu des administrateurs, la ligne « X avatars manquants ». Elle s'affiche quand un ou plusieurs avatars sont indisponible, ainsi ils sont prévenus.



Même avec la liste des avatars manquants nous n'avons pas trouvé de points communs entre les images, cela doit apparemment venir du serveur qui supprime des images. Mais nous ne savons pas si c'est une perte ou un bug dû aux noms, aux formats ou aux codes.

Discord

Dragonium, comme beaucoup de jeux, est présent sur les réseaux sociaux. Quand le jeu a commencé, Skype était beaucoup utilisé et donc un compte Skype a été créé pour que les joueurs puissent les contacter. Mais depuis plusieurs année Skype est très peut utilisé et Discord l'a remplacé. Dragonium a donc maintenant un compte sur discord aussi. Mais au début du stage j'ai remarqué que sur la page d'accueil l'image à côté du pseudo était la même que celle à côté du pseudo Skype.

J'ai donc proposé de modifier cette erreur, et on m'a dit que ça avait été rajouté rapidement et qu'il fallait donc aussi rajouter le pseudo discord partout où il y avait celui de skype. J'ai donc utilisé Total commander pour trouver toutes les pages qui parlent de skype et j'ai rajouté les image et pseudo discord.

Un mois plus tard Discord a décidé de changer leur réglementation sur les pseudos, ils ne veulent plus utiliser les # avec des chiffres après. Donc tous les pseudos doivent être changés. Ce qui veut dire qu'il a fallu remodifier les pages.

Liaison de comptes et changement rapide de personnage

Sur Dragonium les joueurs ont le droit d'avoir plusieurs personnages, même si c'est surveillé et qu'ils n'ont pas le droit d'aider un de leurs personnages avec un compte secondaire. Pour simplifier la vie aux joueurs qui utilisent plusieurs comptes, on nous a donné comme mission de créer une interface qui permet de changer de compte sans avoir à se déconnecter ou à retaper le mot de passe. Seuls les comptes avec une même adresse mail valide peuvent se lier. Nous devions donc créer un moyen de lier ou délier les comptes et un moyen de switcher de compte.

La liaison de multicompte

Pour savoir quel compte est lié à qui nous avons créé une table dans la base de données qui sert à associer les différents identifiants. Il y a 2 champs (annexe 13). Si la compte « A » veut se lier au compte « B », on enregistre l'id de « A » dans le premier champ et l'id de « B » dans le deuxième. Et si le joueur veut lier un compte « C » à ses comptes alors on lie « A » avec « C » et « B » avec « C ».

Dans une partie appeler « Multicomptes » dans le page des options, nous avons ajouté la liste des comptes liés au joueur et un menu déroulant avec tous les comptes qui ont son adresse email. Pour lier un compte il choisit le personnage dans le menu déroulant que vous pouvez voir annexe 14. Une fois que l'utilisateur a sélectionné un compte et qu'il clique sur « ok » on ouvre une pop-up qui demande d'écrire le mot de passe du compte choisi. Bien sûr cette pop-up revérifie que le joueur a le droit, qu'il a le même email et qu'il n'est pas déjà lié au compte. Vous pouvez voir l'affichage de la pop-up annexe 15. Quand un mot de passe est entré nous comparons le hash du mot de passe en base de données et celui du mot de passe entré. Le hachage de mot de passe est une des pratiques de sécurité basiques qui doit être effectuée. Sans cela, chaque mot de passe stocké peut être volé si le support de stockage est compromis. Donc une fois les deux hash comparés, s'ils sont identiques, le mot de passe est correct donc nous lions les comptes. Si l'utilisateur s'est trompé de mot de passe il doit recommencer l'opération, au bout de 5 essais on nous a donné comme

consigne de bannir une quinzaine de minutes le compte et d'enregistrer une fraude, c'est-à-dire que l'administrateur pourra voir qu'il essaye de se connecter sans avoir le mot de passe.

Comme je l'ai dit il y a aussi dans les options la liste des personnages liés avec à coté une croix pour supprimer les liaisons (annexe 16). Quand un joueur clique sur la croix on lui demande s'il est sûr de vouloir supprimer la liaison. Quand il clique sur « oui » nous revérifions qu'il ait les droits sur le compte et nous supprimons toutes les liaisons du compte. Pour reprendre l'exemple précédant quand « A » demande de supprimer la liaison avec «B». Dans la base de données on supprime les liaisons A/B et B/C. Comme ça si le joueur va sur le compte C il ne voit plus B.

Comme les comptes doivent avoir le même email nous avons fait en sorte de supprimer les liaisons à tous les endroits ou un joueur change de mail, même si c'est un administrateur qui lui modifie sont mail. Nous les supprimons également quand il modifie son mot de passe.

Changement de compte

Pour le switch de compte nous avons commencé par modifier le fichier Carte_menu qui correspond au menu sous la mini-map. Nous avons ajouté un lien appeler « multicomptes ». Il amène à la page multicomptes.php. Dans cette page nous affichons les comptes du joueur auxquels il a le droit de se connecter, c'est-à-dire ceux qui sont liés à lui et qui ne sont pas en cours de bannissement. Il peut voir les avatars, les noms des personnages et les boutons « changer ». Quand il clique sur le bouton d'un compte alors on vérifie comme toujours qu'il se connecte à un de ses personnages puis s'il a le droit on déconnecte le personnage auquel avec lequel il jouait et on le connecte sur celui qu'il a choisi. Pour l'utilisateur c'est comme s'il venait de se connecter au personnage.

Page multicomptes :

Cette mission a demandé beaucoup de sécurité mais nous avons réussi avec Freddy à l'accomplir et le système de multicompte est désormais en ligne sur Dragonium.

Interface de gestion des libellés

Pour la dernière tâche qu'on m'a confiée nous étions, nous étions trois stagiaires qui finissions notre stage le lendemain. Cette mission consistait à réaliser une page sur laquelle nous pouvons modifier tous les textes de Dragonium. Comme je l'ai déjà expliqué les textes dans la base de données sont en français puis il y a un séparateur « § » et en anglais. Seulement deux langues sont disponibles et tous les textes ont été traduit depuis un moment. Mais le site ne veut pas se limiter à ça. Même si pour l'instant ce n'est pas une priorité, la traduction dans d'autres langues est toujours envisagée. Cette page doit donc servir à modifier des textes si jamais il y a des erreurs et à rajouter de nouvelles traductions si une langue est ajoutée. Les consignes sont les suivantes.

Les personnes qui ont le droit d'accéder à cette page doivent avoir le rôle de traducteur et chaque traducteur ne peut pas modifier toutes les langues, ils modifient seulement celle qu'ils parlent et que les administrateurs leurs ont autorisé à modifier.

Il ne faut pas afficher tous les textes. Il y a presque 5000 libellés donc si la page les chargent tous elle sera trop lente.

On doit pouvoir choisir les langues qu'on affiche, pour l'instant il y a juste français et anglais mais si on a quinze langues ça peut prendre trop de place sur l'écran.

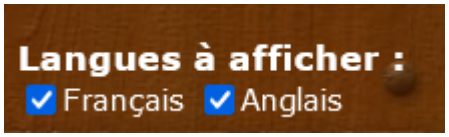
Il faut pouvoir effectuer des recherches par texte, par id, et par commentaire.

La page doit en fait ressembler à ce qu'on voit quand on regarde la table des textes dans la base de données, mais nous ne pouvons pas supprimer de libellés et les textes seront plus lisibles.

Affichage et modification des libellés

Pour savoir quelle langue un traducteur peut modifier nous avons créé un nouveau champ dans la base de données qui contient ces langues.

Sur la page de gestion, en dessous de la barre de recherche il y a toutes les langues disponibles avec une case coché ou non à côté. Quand un traducteur arrive sur



la page, les langues qu'il peut modifier sont pré-cochées, ainsi que le francais qui est coché par default. Si la case est cochée ça signifie que le texte de la langue est affiché. Si on a un doute on peut afficher des langues qu'on n'a pas le droit de modifier pour voir le texte et mieux comprendre le sens de la phrase.

Vous pouvez voir annexe 17 qu'on affiche l'id, les textes dans les differentes langues, le commentaire qui contient en fait les pages dans lequel les libellés sont utilisés et à la fin un bouton modifier pour valider les changements. Si par exemple je n'ai pas le droit de modifier l'anglais alors je ne peux pas changer le texte, la case l'affiche juste.

La barre de recherche

Elle se présente de cette manière, un menu déroulant où nous avons le choix entre l'id, le texte et le commentaire. Le menu déroulant suivant nous donne le choix entre =, <=, >=, et %LIKE%. Ensuite nous avons le texte de la recherche et le nombre de résultats souhaité. Bien sûr au traitement du formulaire on verifie que tout est valide pour ne pas faire de mauvaise requête SQL. Normalement si nous faisons « texte = une_recherche » nous devons écrire dans la recherche « le texte français§the english text ». Mais dans le traitement du formulaire nous avons fait en sorte que si on écrit « le texte français » alors nous avons le même résultat. Ainsi les traducteurs peuvent coller le texte qu'ils ont vu avec une erreur et ils trouvent le résultat souhaité.

Le menu administrateur

C'est dans ce menu que les administrateurs peuvent modifier les grades des joueurs. En regardant la fiche d'un joueur ils peuvent donner plusieurs grades et faire d'autre modifications. Le système pour mettre le grade « traducteur » était déjà fait mais il nous restait à donner la possibilité d'assigner des langues. Pour ce faire nous avons décidé d'ajouter une partie « traduction » si le joueur a en grade principal ou secondaire traducteur (annexe 18). Dans ce cadre on peut voir les langues autorisées à modifier. On peut en ajouter grâce au menu déroulant et en supprimer grâce à la croix. Ces langues sont donc enregistrées dans le champ créé plus tôt.

Le résultat de cette mission est satisfaisant car il est fonctionnel et ressemble à ce qu'on peut voir en base de données mais est plus lisible. Malheureusement quand mon stage s'est fini nous n'avions pas entièrement finalisé la page. Nous avons manqué de temps. Le système de page suivante avait quelques problèmes et il aurait été bien de pouvoir rajouter de nouveaux textes directement de cette page. Je pense que si une telle page avait été créée plus tôt, mon stage aurait été diffèrent. Sur cette page les textes sont plus lisibles qu'en base de données, je l'aurais surement utilisé.

Conclusion

Durant mon stage chez Zyzomis j'ai eu la chance de voir mes programmes mis en ligne et utilisés par les joueurs. Je trouve ça gratifiant et satisfaisant de constater que des joueurs s'amusent avec notre code. C'est d'autant plus intéressant de devoir réfléchir à des moyens de réaliser les missions qui sont intuitifs pour les utilisateurs, et les moins encombrants pour le site. Bien que nous réfléchissions déjà comme cela en cours, savoir que des personnes l'utilisent derrière est toujours plus motivant.

Je passais mon temps sur le serveur de développement de Dragonium mais quand j'allais sur le site où il y avait les joueurs et que je voyais les messages qui annonçaient les nouveautés que nous avions programmées, ça me faisait toujours plaisir. Par exemple un matin j'ai regardé les annonces que Dragonium envoie par message dans le jeu, il parlait des bugs réglés et à la fin ce message « Ça gronde ? On a repéré des vibrations suspectes sous les montagnes de Ser et d'Adak. Une rumeur circule, disant que les Mages Noirs seraient en train de fomenter quelque chose. Cela faisait longtemps qu'on n'avait pas entendu parler d'eux... Il va falloir suivre cela de près. ». Je savais qu'il parlait d'un tunnel avec un boss qui doit servir de raccourci pour la faction qui tue ce monstre. Nous avions codé ce tunnel la veille et Ludovic préparait une histoire et un évènement pour l'intégrer dans le jeu. J'ai aimé le fait de programmer une nouveauté puis de chercher ensuite un moyen amusant de l'intégrer. De la même façon quand nous avons fini de faire et tester les multicomptes, le lendemain une multitude de gens avait utilisé la page pour lier leur personnage.

Les missions étaient parfois plus qu'importantes pour le site, comme certains bugs, mais d'autres étaient plutôt esthétiques. Les médailles par exemple ne sont pas vitales pour le jeu, Dragonium a bien tenu sans jusqu'ici. Mais à mon avis ça apporte un plus pour les joueurs. Le système de multicomptes quant à lui est pour moi une idée et un moyen vraiment pratique, qui s'simplifie la vie aux joueurs.

Dans ce rapport de stage j'ai parlé des grosses missions auquelles j'ai participé. Mais entre chacune d'elles on avait plusieurs petites taches à faire et même pendant que nous codions les médailles ou les multicomptes nous avions des petits tickets à faire. Je ne pouvais pas tout expliquer mais j'ai

appris et vu beaucoup de choses. J'ai appris des nouvelles façons de faire des requêtes SQL, des nouvelles fonctions SQL, j'ai perfectionné mon PHP.

Certains disent que dans ce genre de stage en première année ils ne font rien, mais pour ma part nous avions toujours plusieurs taches en même temps. Ça m'a permis de m'améliorer.

Nous travaillions avec des horaires convenables et il y avait une ambiance chaleureuse, Ludovic est plus que gentil. Le télétravail était pratique bien que parfois, avec du recul, je me dis que c'est bien de pouvoir voir et discuter en face à face avec nos collègues.

Nous avons utilisé plusieurs logiciels. Total commander fonctionne correctement et il y a beaucoup de raccourcis mais il a un gros inconvénient à mes yeux. En effet, quand nous modifions un fichier et que nous l'envoyons au serveur, ça écrase l'ancien fichier sans faire de sauvegarde. Certains logiciels permettent de voir qui a fait quelle modification ou ajout, et aussi donnent la possibilité de retrouver directement l'ancienne version.

J'ai aussi utilisé Notepad++ pour coder pendant le stage et il y avait moins de raccourcis qui simplifient la vie d'un développeur. Je ne pense pas retourner sur notepad++ mais son utilisation a peut-être eu un bon point, maintenant j'imagine que je suis plus habitué à écrire les fonctions et coder en général. Quand nous étions plusieurs sur le même code nous utilisions aussi Codeshare. C'est un site pratique, bien qu'il ait une mauvaise indentation. C'est un moyen simple, efficace et gratuit de développer à plusieurs mains. Je pense que pendant les projets de groupe en classe je proposerai ce site si on doit être sur la même page. Ça permet à tout le monde de voir l'avancement et ça simplifie le regroupement des codes.

J'ai toujours aimé programmer, les jeux sont toujours amusants à développer et en développer un qui est en ligne et qui a des utilisateurs actifs me conforte dans cette idée. Je ne suis pas encore sûr de l'endroit où je souhaiterais aller en stage l'année prochaine. Nous avons créé beaucoup de sites durant l'année et j'aimerais voir comment ça se passe dans une entreprise de création de site web. Mais l'univers du jeu est aussi attirant.

En résumé, ce fut un stage instructif et intéressant, je remercie encore une fois Ludovic RESSEGAND et Zyzomis en général de m'avoir pris comme

stagiaire et de m’avoir appris tant de choses. J’ai passé de bons moments à travailler pour Zyzomis.

Annexe

Annexe 1 :

```php
//On fait la requête pour lister les joueurs
$reqJoueurs = mysqli_query($bdd, "SELECT nb_monstres_tues_joueur, id_joueur, nb_contrats_tues_joueur, nb_morts_recues_joueur, mises_paris_br_joueur, inscriptions_eleveur_br_joueur, nb_or_vole_joueur, nombre_boxes_joueur, karma_joueur, nb_soins_donnes_joueur, nb_villes_tuees_joueur, date_creation_joueur, code_vip_joueur FROM joueurs WHERE grade_joueur >= 4 ORDER BY id_joueur");

// Pour chaque joueur, on vérifie les critères
//Si le critère est rempli, on vérifie qu'on n'a pas déjà la médaille
//Si on n'a pas la médaille, on l'ajoute
while ($donJoueurs = mysqli_fetch_array($reqJoueurs)) {
    //Pour chaque médaille, on regarde ce qu'on doit regarder - on liste chaque ID de médaille
    for ($bcl=1;$bcl<sizeof($arrayMed);$bcl++) {
        //Nb de monstres tués
        if ($bcl==1 && $donJoueurs['nb_monstres_tues_joueur'] >= $arrayMed[$bcl]['palier1_medaille']) {
            //On check si on a déjà la médaille
            $reqComptMed = mysqli_fetch_array(mysqli_query($bdd, "SELECT *, COUNT(*) AS nb FROM medaille_joueur WHERE joueur_medaille_joueur = ".$donJoueurs['id_joueur']." AND medaille_medaille_joueur = ".$bcl));
            if ($reqComptMed['nb'] == 0) {
                //Requête d'insertion de nouvelle médaille
                $reqNewMed = mysqli_query($bdd, "INSERT INTO medaille_joueur VALUE (".$donJoueurs['id_joueur'].", ".$bcl.",".$donJoueurs['nb_monstres_tues_joueur'].", NULL)");
            }
            else {
                if ($donJoueurs['nb_monstres_tues_joueur'] > $reqComptMed['valeur_medaille_joueur']) {
                    //On écrit la nouvelle valeur
                    $reqNewMed = mysqli_query($bdd, "UPDATE medaille_joueur SET valeur_medaille_joueur = ".$donJoueurs['nb_monstres_tues_joueur']." WHERE joueur_medaille_joueur = ".$donJoueurs['id_joueur']." AND medaille_medaille_joueur = ".$bcl);
                }
            }
        }
```

Annexe 2 :

```php
for ($bcl=0;$bcl<sizeof($arrayMedaillesTemporaires);$bcl++) {
    if ($arrayMedaillesTemporaires[$bcl]==18) {
        //Meilleur tueur de loups
        $reqclassement = mysqli_query($bdd, "SELECT (@row_number:=@row_number + 1) AS row_num, joueur_classement FROM classements, (SELECT @row_number:=0) AS t ORDER BY date_classement DESC, nb_morts_donnees_loup_mois_classement DESC, niveau_classement DESC LIMIT 4");
        while ($donClassement = mysqli_fetch_array($reqclassement)) {
            $reqNewMed = mysqli_query($bdd, "INSERT INTO medaille_joueur VALUE (".$donClassement['joueur_classement'].", ".$arrayMedaillesTemporaires[$bcl].", ".$donClassement['row_num'].", NULL)");
        }
    }
    else if ($arrayMedaillesTemporaires[$bcl]==19) {
        //Meilleur tueur de griffons>
        $reqclassement = mysqli_query($bdd, "SELECT (@row_number:=@row_number + 1) AS row_num, joueur_classement FROM classements, (SELECT @row_number:=0) AS t ORDER BY date_classement DESC, nb_morts_donnees_griffon_mois_classement DESC, niveau_classement DESC LIMIT 4");
        while ($donClassement = mysqli_fetch_array($reqclassement)) {
            $reqNewMed = mysqli_query($bdd, "INSERT INTO medaille_joueur VALUE (".$donClassement['joueur_classement'].", ".$arrayMedaillesTemporaires[$bcl].", ".$donClassement['row_num'].", NULL)");
        }
    }
    else if ($arrayMedaillesTemporaires[$bcl]==20) {
```

Annexe 3 : Dracoliche rouge dans Dragonium

Annexe 4 : Vérification de Dracoliche en binaire

```php
//On regarde si il a déjà tué ce dracoliche
$valCompDraco = decbin($reqMedDracoliche['valeur_complementaire_medaille_joueur']);
//si besoin ajouter des 0 avant pour avoir 5 caractères
$valCompDraco = str_pad($valCompDraco, 5, "0", STR_PAD_LEFT);
//On met chaque caractère valCompDraco dans un array
$arrayValCompDraco = str_split($valCompDraco);

//on créer un array avec le binaire du dracoliche
if($id_monstre == 2){
    $arraydraco = [0, 0, 0, 0, 1];
}
else if($id_monstre == 9){
    $arraydraco = [0, 0, 0, 1, 0];
}
else if($id_monstre == 10){
    $arraydraco = [0, 0, 1, 0, 0];
}
else if($id_monstre == 11){
    $arraydraco = [0, 1, 0, 0, 0];
}
else if($id_monstre == 164){
    $arraydraco = [1, 0, 0, 0, 0];
}

//Si arraydraco[bcl] est pas ==0 on compare les deux array et on si besoin on change ajoute a l'autre un 1
for($bclDraco=0; $bclDraco < 5; $bclDraco++){
    if($arraydraco[$bclDraco] == 1 && $arrayValCompDraco[$bclDraco] == 0){
        $arrayValCompDraco[$bclDraco] = 1;

        $valCompDraco = implode("", $arrayValCompDraco);
        //on repasse le array "arrayValCompDraco" en décimal
        $NewValCompDraco = bindec($valCompDraco);

        $reqNewMed = mysqli_query($bdd, "UPDATE medaille_joueur SET valeur_medaille_joueur = valeur_medaille_joueur + 1,
        valeur_complementaire_medaille_joueur = ".$NewValCompDraco." WHERE joueur_medaille_joueur = ".$id_joueur." AND medaille_medaille_joueur = 22");
        //Termine le for
        $bclDraco = 6;
    }
}
```

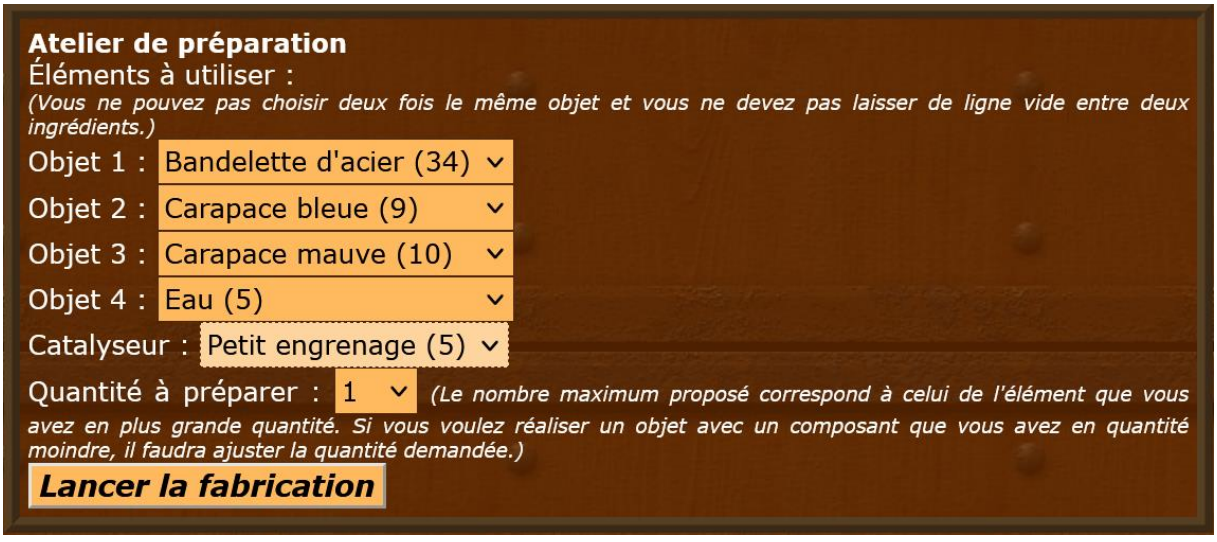
Annexe 5 :

Annexe 6 : Affichage des médailles d'un joueur

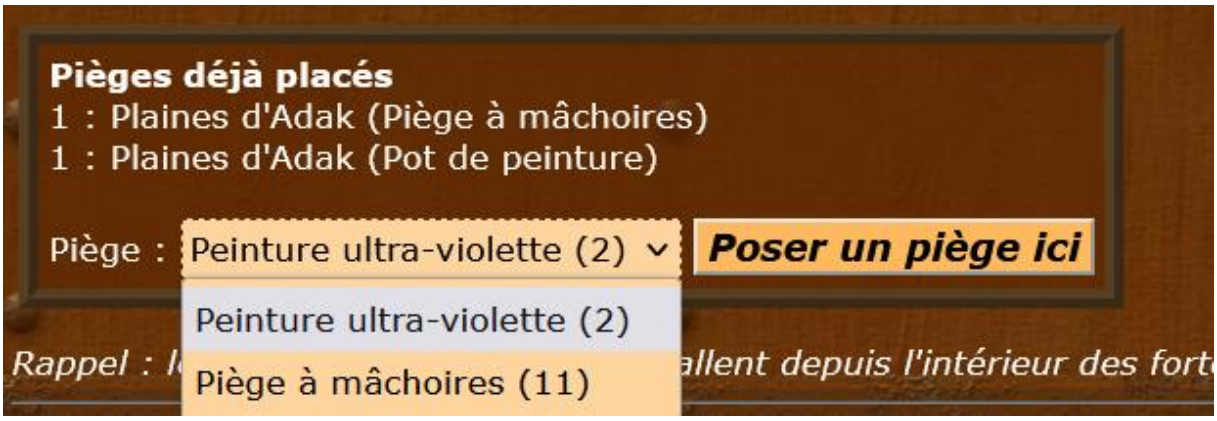
Annexe 7 : Affichage de la description d'une médaille

Annexe 8 : La peinture ultra-violette

Annexe 9 : Atelier de préparation du pot de peinture

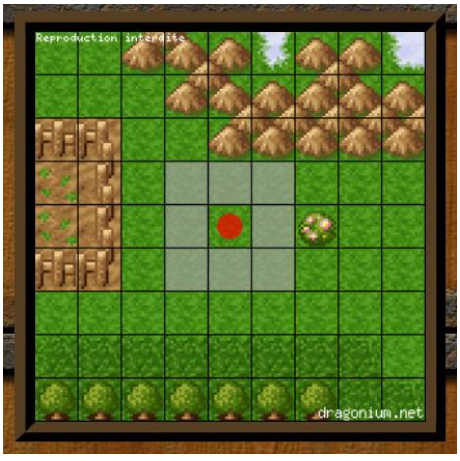


Annexe 10 : Pose des pièges pour ingénieurs



Annexe 11 : Pose des pièges pour outilleur

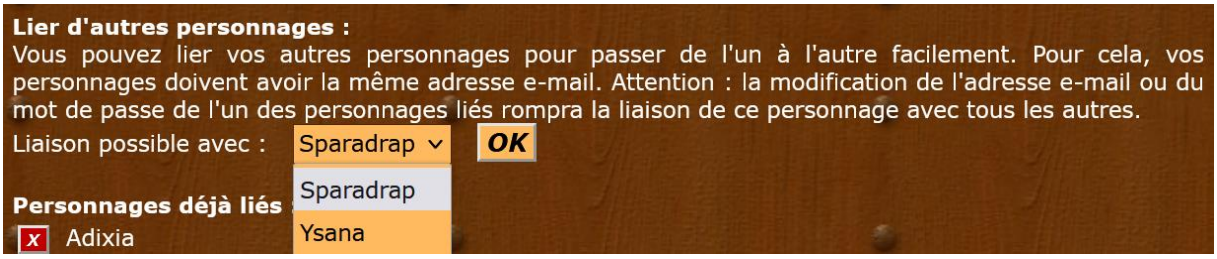
Annexe 12 : Mini-map de Dragonium, le point rouge est l'emplacement du joueur



Annexe 13 : Table de liaison des multicomptes

Colonne	Type
j1_multicompte	bigint(20) unsigned
j2_multicompte	bigint(20) unsigned

Annexe 14 : Menu déroulant des comptes que le joueur a le droit de ler à lui



Annexe 15 : Demande de mot de passe pour lier un compte

Annexe 16 : Liste des multicomptes dans les options

Annexe 17 : Liste des libellés

Annexe 18 : Cadre traduction pour ajouter des langues