

Multigates

Développement d'un jeu multi-joueurs sur navigateur web.

Travail de maturité

Supervision : M. Grégoire Favre

JOEL IUNIUS 3M2

2014/2015

Remis le 17.11.14



Table des matières

Introduction.....	3
Pourquoi Multigates ?	3
Quels outils m'ont permis de débiter la conception de Multigates ?.....	4
Un projet	4
Des connaissances de base	5
Une ligne directrice	7
Un codage optimisé.....	7
Quelles évolutions auront été faites ?	8
Manière de coder	8
Manière de penser	9
Structure changée	9
Quels problèmes ont été rencontrés	10
Bugs ; comment les gérer ?	10
Changement de pensée.....	11
Trop de choses à faire	11
Explications du jeu :.....	12
Comment jouer ?.....	13
Comment est structuré Multigates ?	15
Le futur	16
Conclusion	17
Le parcours	17
Le dernier Mot.....	17

Introduction

Un de mes hobbies étant l'informatique, cela fait quelques temps que j'ai commencé à coder des sites web.

Voulant mener à bien un nouveau projet touchant des points que je n'avais pas encore explorés, j'ai trouvé judicieux de faire d'un projet informatique mon travail de maturité. J'ai donc choisi un thème libre en informatique.

Ainsi est née l'idée de Multigates, par la volonté de comprendre le développement d'un jeu multi-joueurs en ligne.

Après quelques 200 heures de codage, Multigates est un projet aujourd'hui finalisé. Dans ce dossier nous allons parcourir les méthodologies utilisées afin de coder le jeu, nous allons aussi parler du jeu lui-même ainsi que de l'enrichissement personnel que ce projet m'a apporté.

Pourquoi Multigates ?

Ayant quelques connaissances déjà exploitées auparavant en langage(s) web, je souhaitais avant tout créer plus qu'un simple site web, mais une vraie interface basée sur un langage dynamique, utilisant pratiquement toutes mes connaissances et me faisant en découvrir de nouvelles. L'idée d'un gros projet était donc née.

Etant un grand fan de la série StarGate¹ et ayant joué sur d'autres jeux en ligne basés sur cette série, j'avais envie de créer mon propre jeu. C'est pourquoi j'ai souhaité faire mon TM sur un jeu que je développerai en langage web basé sur l'univers StarGate.

Multigates devait me permettre d'approfondir mes connaissances dans ce domaine, c'était aussi l'occasion de mener un projet de A à Z en solitaire et ainsi voir si j'en étais capable.



Le vaisseau Atlantis dans la série StarGate Atlantis

http://fc01.deviantart.net/fs71/f/2014/007/7/8/stargate_atlantis_city_2014__cam__2_by_coldilian-d718kk8.png

¹ Stargate est une série tv de science-fiction ; plus d'infos ici <http://fr.wikipedia.org/wiki/Stargate>

Quels outils m'ont permis de débiter la conception de Multigates ?

Pour pouvoir développer un jeu tel que Multigates, il était nécessaire d'avoir quelques connaissances générales en informatique et en codage avant de débiter. Mais ce n'est pas tout : en effet, il est aussi nécessaire, avant de commencer, de développer clairement une idée de projet. C'est-à-dire être capable de visualiser notre jeu une fois celui-ci fini.

Il faut aussi utiliser un codage optimisé, et donc avoir déjà de l'expérience en développement de site web dit dynamiques.

Dans ces quatre sous-chapitres sont développés ces points importants.

Un projet

Pour commencer, il fallait trouver une idée de projet.

L'idée de Multigates m'est par exemple venue du simple fait que je jouais régulièrement sur un jeu appelé worldOfStargate² et qui alors n'était plus entretenu par son créateur. Devant chaque jour faire face à la stagnation de son développement, j'eus l'idée de faire moi-même un jeu Stargate du même style que WorldOfStargate, mais cette fois en pouvant créer les choses comme je le souhaitais et partir de zéro.

Ce que je recommande donc pour un projet d'étude, aussi bien scolaire que personnel, c'est une passion, vouloir créer quelque chose sur un hobby que l'on possède ou un univers que l'on aime.

Une fois la passion trouvée, il faut pouvoir visualiser un site web, ou en l'occurrence un jeu, basé sur cette passion.

Pour Multigates, je me suis inspiré des jeux déjà faits sur des thèmes de conquête galactique (tel que wos, Ogame³, Wootook⁴). Mais il faut aussi créer du neuf, j'ai donc voulu faire quelque chose que je n'avais pas encore vu pour Stargate : en l'occurrence, baser mon jeu sur la série StarGate Atlantis. Je voulais aussi faire du vaisseau la principale pièce du jeu.

J'ai donc commencé à imaginer ce jeu, faire des plans des différentes parties à concevoir, et donc visualiser par où je devais commencer.

C'est une partie essentielle du Travail de maturité, car elle permet de se faire une vague structuration de celui-ci et estimation des ressources ou temps nécessaire à sa conception.

² <http://www.worldofstargate.fr/>

³ <http://fr.ogame.gameforge.com/>

⁴ <http://wootook.org/>

Des connaissances de base

Mais avant tout pour coder un jeu ou n'importe quel site web, il faut avoir quelques connaissances de base en codage.

Pour la réalisation de Multigates, qui est avant tout un site web dynamique, j'ai utilisé cinq langages différents, à savoir le html, css, javascript, php et sql. Avant de commencer le jeu je devais donc avoir quelques connaissances dans tous ces langages.

C'est là qu'entre en jeu l'apprentissage en solitaire. En effet, j'ai acquis toutes ces connaissances grâce à un site web et aux cours qu'il proposait. C'est donc tout seul que j'ai pu obtenir les bases en langage informatique et web pour commencer le projet.

Pour apprendre la base de toute page web, c'est-à-dire le HTML et le CSS, je me suis par exemple rendu sur ce tutoriel ; <http://openclassrooms.com/courses/apprenez-a-creer-votre-site-web-avec-html5-et-css3>

Pour le PHP et le SQL (MYSQL dans ce cas précis), c'était ce cours-là : <http://openclassrooms.com/courses/concevez-votre-site-web-avec-php-et-mysql>

Enfin pour le javascript, c'était ici : <http://openclassrooms.com/courses/dynamisez-vos-sites-web-avec-javascript>

Mais à quoi ces cours me servent-ils exactement ?

Voici les deux vidéos de présentation des deux premiers cours qui expliquent très bien ce que permettent de faire ces langages :

Html et css : <http://player.vimeo.com/video/88011078>

Php et sql : <http://player.vimeo.com/video/87516214>

Pour le javascript, il s'agit d'un langage permettant de modifier en directe ce qu'affiche un site web.

Une fois ces tutoriels lus il fallait pouvoir s'exercer avant de commencer réellement le jeu. Pour commencer, créer un site web en html et css uniquement m'a paru être une bonne idée. Pour ma part, j'ai commencé par starmine.fr, la première version, d'ailleurs disponible ici :

<http://starmine.fr/versions/version1/>

Puis j'ai voulu voir plus loin en faisant une deuxième version utilisant du php (surtout pour inclure des pages et créer des formulaires) :

<http://starmine.fr/versions/version2/>

Puis une autre version avec du javascript et un espace membre :

<http://starmine.fr/versions/version3/>

J'ai aussi tenté une version flat design qui n'a jamais été publiée :

<http://starmine.fr/versions/version6/>

Finalement, la dernière version de starmine.fr (version actuelle) qui incluait aussi du javascript m'a permis d'utiliser tous les langages nécessaires à la programmation de Multigates et ainsi être suffisamment entraîné pour commencer.

Mais j'ai aussi voulu, avant de commencer réellement le projet, tenter de reproduire quelques systèmes d'un jeu déjà existant, afin de me rassurer et de pouvoir m'exercer un peu plus. Ce fût donc une version jamais publiée dont il ne reste pour trace qu'une vidéo que voici : <http://youtu.be/uq8yslHTO2A> Ce test s'inspirait de l'ancienne version de World Of Stargate (une image de l'ancienne version est présente ici : http://www.jeux-en-ligne-gratuits.net/images_sites/444.jpg).

Bien entendu, cette version étant trop inspirée de wos, j'ai recommencé à zéro par la suite pour le travail de maturité, c'est d'ailleurs à ce moment que commençait véritablement le travail de maturité.

J'ai aussi à disposition un serveur web, et j'ai acheté le nom de domaine multigates.fr pour pouvoir héberger le jeu par la suite.



Version test de Multigates.fr

Une ligne directrice

Pour concevoir un gros projet tel que Multigates, il est important de se donner des lignes directrices. Elles permettent de mieux cadrer l'évolution du travail, et de ne pas être perdu dans la conception de celui-ci.

Cela se passe par la séparation de tous les points essentiels du jeu sous forme de plan (ci-joint).

Une fois ce plan fait, il faut être capable de visualiser les systèmes principaux qui seront utilisés, et d'essayer de les optimiser. C'est d'ailleurs là où entre le sous chapitre numéro 4.

Il est aussi important de garder une trace du développement, c'est-à-dire un résumé des modifications apportées au jeu. Pour Multigates, ce résumé est présent dans une page html du jeu, nommé versions.html, dont l'url actuelle est <http://multigates.fr/versions.html>

Un codage optimisé

Pour optimiser un maximum le développement du jeu, et ainsi gagner en temps et en facilité, il est important de bien structurer le jeu.

Pour ce faire, il faut visualiser les différents types de codage ou fonctions utilisés.

Pour Multigates, cela tourne par exemple sur ces axes-ci :

Une page principale d'accueil (Inscriptions, présentation, etc).

Une page principale de jeu (Où se trouve Atlantis).

Une partie css (contenant les fichiers mettant en forme le jeu).

Une partie javascript (contenant les fichiers javascript du jeu).

Une partie traitement (qui permet de communiquer entre le javascript et la base de données via php).

Une partie interfaces (qui contient les différentes interfaces du jeu).

Une partie bâtiments (qui contient les différents bâtiments du jeu).

Et finalement une partie fonctions (Qui contient toutes les fonctions php du jeu).

Quelques autres parties sont aussi présentes mais elles sont uniques à Multigates.

Ainsi, avoir un dossier de jeu structuré permet d'optimiser son codage et de savoir où se trouve telles ou telles fonctions. Ceci est indispensable sur un projet se déroulant sur plus d'une année.

Parmi les autres optimisations, on note aussi le codage en lui-même, mais cela sera traité plus tard.

Quelles évolutions auront été faites ?

Le but premier d'un tel projet dans le cadre des études est avant tout de parvenir à de nouvelles découvertes, et donc surtout de pouvoir constater une évolution notable dans la manière de coder, de penser, ou de structurer un projet web. Via ces trois sous-chapitres nous allons donc découvrir ce que Multigates m'a apporté.

Manière de coder

Au commencement, les bases apprises grâce à d'autres projets permettent de débiter un jeu sans réel problème, mais après quelques semaines de codage, on remarque très vite que de manière très naturelle on commence à changer la manière de faire qu'on avait au début pour certains types d'algorithmes.

Prenons un exemple concret. Au début, je découvrais un peu le principe d'une fonction, je ne savais donc pas comment faire pour qu'une fonction php puisse utiliser la base de donnée sans avoir à chaque fois à la passer via une variable (fonction mafonction(\$bdd){}) J'ai donc procédé de cette manière-là pendant un certain temps avant de trouver sur internet une autre solution, celle de la déclaration en global de la variable \$bdd à l'intérieur d'une fonction. Ainsi cela devenait (fonction mafonction(){ global \$bdd ;}) ce qui permet entre autre de ne pas devoir toujours, lors de l'utilisation d'une fonction, transmettre la bdd avec (au lieu d'utiliser une fonction en faisant mafunction(\$bdd) ; je pouvais dès lors faire uniquement mafunction() ;) ce qui est une optimisation concrète.

Et c'est donc ce genre d'optimisation qui est réalisé tout au long, grâce à des découvertes personnelles, mais aussi à certaines recherches effectuées.

Ainsi Multigates réunis à l'heure actuelle du codage jeune et optimisé, les deux fonctionnant ensemble, je n'ai pas jugé nécessaire de reoptimiser mes anciens codages, car ceux-ci permettent aussi de visualiser au sein même du codage les différentes optimisations et essais réalisés en cours de route.

```
5263
5264 function ins_ressources($bdd, $quoi, $combien)
5265 {
5266
5267 $update = $bdd->prepare('UPDATE ressources SET :quoi = :combien WHERE id_joueur = :id');
5268 $update->execute(array(
5269     'quoi' => $quoi,
5270     'combien' => $combien,
5271     'id' => $_SESSION['id']
5272 ));
5273
5274 }
5275
```

Dans cet exemple, la \$bdd est transmise à la fonction

```
5071
5072 function get_planete_id($id, $quoi)
5073 {
5074     global $bdd;
5075
5076 $selection = $bdd->prepare('SELECT * FROM L_planete WHERE adresse = ?');
5077 $selection->execute(array($id));
5078 $planete = $selection->fetch();
5079 $selection->closeCursor();
5080
5081
5082 return $planete['id_planete'];
5083 }
5084
```

Dans cet exemple, la \$bdd est déclarée en global.

Manière de penser

Il n'y a pas que le codage qui a été optimisé en cours de route, mais aussi le rangement des fichiers, la disposition des interfaces, l'ouverture des bâtiments, et j'en passe.

J'ai d'ailleurs voulu tout au long du développement essayer différentes manières de faire, ainsi on retrouve par exemple pour les bâtiments, des bâtiments sans interface, d'autres avec de multiples interfaces, etc. La tour de contrôle est par exemple interactive, c'est-à-dire qu'il faut cliquer sur l'ordinateur ou le dhd pour que l'interface s'ouvre. Alors que l'armurerie, elle, ne contient pas d'interface, mais directement une page de gauche est de droite. Cela change fondamentalement la structure même du codage du bâtiment en question.



Dans cet exemple (armurerie), l'interface est incluse dans le bâtiment, ce n'est pas une sous-fenêtre



Dans cet exemple (contrôle), l'interface est une sous fenêtré, on peut en ouvrir une autre dans le même bâtiment.

Là où la page contrôle.php contient uniquement une image de la salle de contrôle avec les boutons des différentes interfaces, la salle armurerie.php, elle, contient deux includes, celle d'une page de gauche et d'une page de droite. Ainsi le contenu de la salle de contrôle se retrouve dans le dossier « interface » alors que le contenu de l'armurerie lui se situe dans le dossier box_grises.

Le but de ces différentes manières de faire est avant tout de trouver celle qui est la plus optimisée.

Mais il n'y a pas toujours une seule manière de faire, en l'occurrence cela peut aussi être un choix personnel ou de circonstance.

Les optimisations de « manière de faire » se font surtout ressentir sur les points semblables à différents systèmes du jeu, mais ayant une modification légèrement optimisée que pour certains de ces systèmes. Par exemple, j'ai trouvé utile de permettre d'actualiser une interface via le nom de celle-ci, mais cela n'est possible que dans certaines fenêtres. Pourtant il serait tout aussi judicieux de reproduire ce système dans les autres.

Structure changée

La structure même du jeu change donc en cours de route, que ça soit le codage comme la présentation de systèmes. Il est important de se rendre compte que si l'on repartait aujourd'hui de zéro, on ne ferait pas les choses de la même manière : en effet, elles seraient beaucoup plus optimisées. C'est cette optimisation constante qui permet de démontrer ce que peut apporter un tel projet en terme de connaissances.

Quels problèmes ont été rencontrés

Lors du développement d'un site web, comme d'un jeu, tout ne se passe pas toujours comme prévu ...

Bugs ; comment les gérer ?

Ah, les bugs, si détestés et pourtant si fréquents. Dans un projet amateur, ils sont omniprésents et pourtant il faut savoir y faire face.

Ctrl + z c'est l'outil favori lors de petites séances de débogages. En effet, lors du développement de Multigates, j'ai rencontré à maintes reprises une page blanche au chargement du jeu ... et oui, tout avait planté !

Comment faire dès lors ?

Simple ! Un petit retour en arrière, intelligent toutefois !

En effet l'idée de départ est simple, on retourne en arrière dans notre codage jusqu'à ce que ça refonctionne, ainsi l'on peut détecter l'emplacement du codage créant ce bug.

Et cela fonctionne rudement bien. Une fois une zone à problèmes détectée, on la relie, pour la plupart du temps découvrir une faute de frappe, ou un « ; » oublié, c'est dire les dégâts que cela peut causer !

On corrige cela et c'est reparti.

Mais ce genre de bug est dit classique, il arrive malheureusement parfois que cela ne soit pas suffisant ;

Dans ces cas-ci, il faut voir plus loin, une fonction retourne « banane » à la place de « 10 » ?

Alors créons des fonctions de débogage.

Par exemple, j'ai créé la fonction `ins_erreur($quoi)`. Elle est toute bête, elle permet d'insérer une valeur dans la base de données à n'importe quel moment en intégrant la date et l'heure à laquelle elle a été utilisée.

Ainsi si la valeur retournée par la fonction posant problème est 10 au début de son exécution et « banane » à la fin de celle-ci, je place une `ins_erreur()` au début, au milieu, à la fin, et je verrai après l'exécution de la fonction vers quel endroit le problème crée l'effet « banane ».

Ensuite on trouve l'erreur et la corrige.

Le réglage d'un bug peut prendre 10 secondes comme une demi-journée ...

Mais à la fin, grâce aux différents outils mis en place, on parvient à vaincre ces petites pertes de temps.

Changement de pensée.

Lorsque l'on rencontre des bugs de même type assez souvent, il devient intéressant de se demander « peut-on changer quelque chose au codage pour éviter ce genre de bug ? »

Il s'agit donc en quelque sorte d'optimisations provoquées par la rencontre de certains bugs.

Prenons un exemple, un tableau `while($i < 10){ $i++; }` cette boucle ne bug pas, mais que se passe-t-il si l'on oublie `$i++` ; ? Rien de bon... Et pourtant ça arrive quelques fois.

```
155  $i = 0;
156  while($i < 10)
157  {
158      echo 'banane';
159  }
160
```

Boucle à problème.

Donc je me suis intéressé aux boucles `for` grâce à cet agacement. Et j'ai découvert, certes un peu tardivement, leur fonctionnement. Non seulement ça s'est révélé pratique et ça m'évite de faire ces bugs, mais en plus cela m'a permis d'apprendre de nouvelles choses que j'ai par la suite utilisées pour d'autres fonctions.

On peut donc clairement dire qu'un bug peut, indirectement comme directement, nous faire progresser, et apprendre de nouvelles choses.

Trop de choses à faire

Malheureusement lorsqu'on s'attarde trop sur les détails, ce qui est important pour un tel projet, on prend vite conscience qu'on n'est pas aussi rapide qu'on le souhaiterait.

Mais il faut savoir faire face à ça, surtout lorsque l'on a une Dead line à respecter.

Pour Multigates par exemple, prenons les alliances, les préparatifs aux systèmes ont été codés, mais le système lui-même ne l'a pas été, par manque de temps, étant donné que j'ai préféré soigner d'autres aspects du jeu.

Il faut donc savoir ce qu'on peut retirer des fonctions prévues sans que cela impacte fortement le gameplay et cela afin de pouvoir respecter le planning.

J'ai donc, toujours dans cet exemple, enlevé le menu « alliance » du jeu.

Avoir un plan de départ avec beaucoup de points est toutefois à mon avis motivant, même si l'on sait dès le départ que certains points ne pourront être développés comme souhaités au début. Cela permet avant tout de toujours avoir quelque chose à faire, et surtout de pouvoir traiter plusieurs points en même temps, à la place de devoir coder quelque chose, puis seulement ensuite se demander ce qu'est le prochain module à développer.

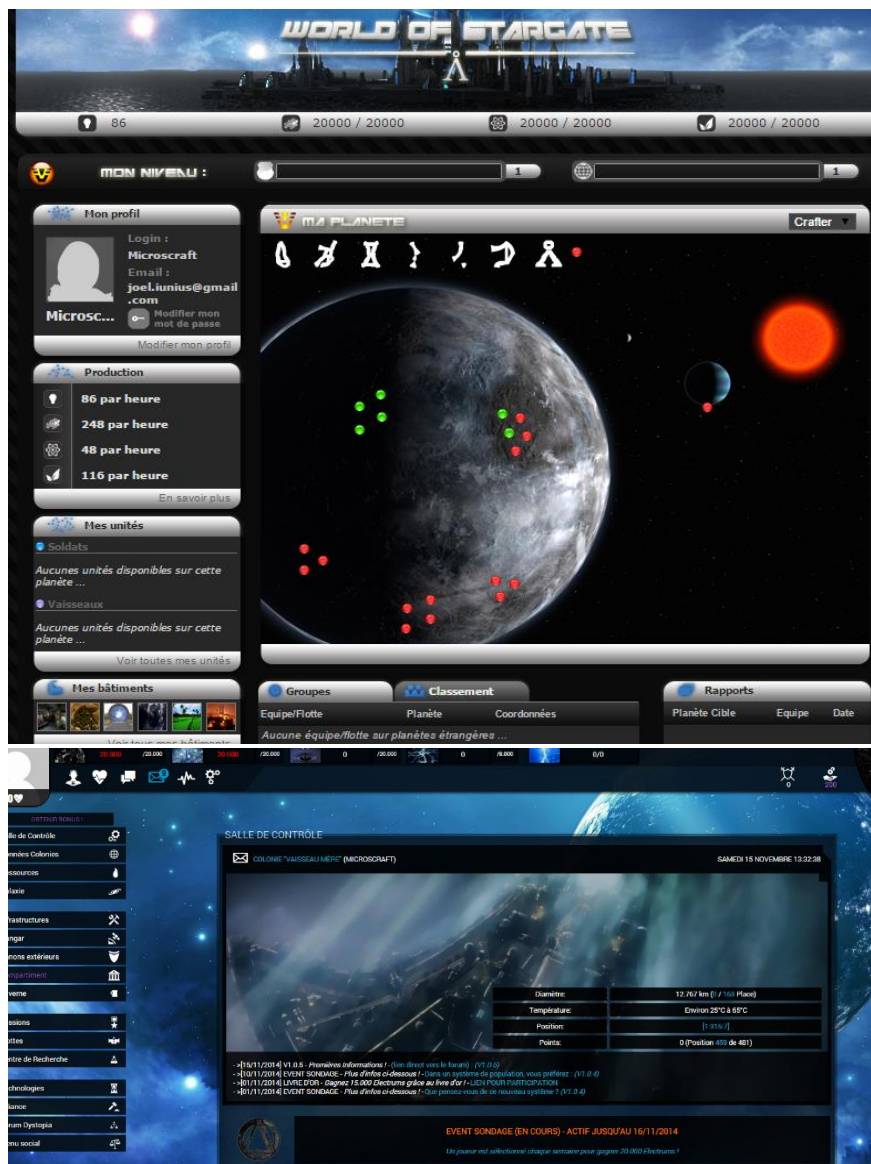
En soit prévoir plus qu'on ne peut réaliser est un aspect motivant, et un challenge, qui même s'il n'est pas atteint, nous satisfait tout autant.

Explications du jeu :

Multigates se base sur un système de jeu déjà exploité auparavant par de nombreuses autres plateformes. Il s'agit de pouvoir jouer en ligne sur le navigateur web de son choix avec d'autres joueurs. Quelques fonctions présentes dans chacun de ces jeux sont par exemple ; une page principale où l'on gère son objet (ici Atlantis, mais ça peut être une planète, un autre vaisseau, une base militaire etc), une page où l'on peut trouver les autres joueurs sur la carte, ainsi que des systèmes sociaux tels que la messagerie ou le tchat. En parlant de systèmes, tous ces jeux se basent aussi sur une évolution du joueur basée sur ses récoltes en ressources et ses capacités d'attaques qui lui permettent de débloquent de nouveaux bâtiments, vaisseaux et autres.

Multigates s'inspire donc grandement de ce qui a déjà été fait, cela dit c'est tout à fait normal, le but premier étant d'évoluer ma manière de coder, et non mon originalité. Toutefois j'ai essayé de faire un maximum dans le neuf, et je pense que c'est assez réussi, car même si les bases sont les mêmes que les autres jeux, Multigates se démarque tout de même sur bien des points (contrôler atlantis, ordinateur de bord, vaisseaux déplaçables sur la galaxie, et pleins d'autres fonctions inédites).

Dans ce chapitre nous allons détailler ces différents points.



Multigates s'inspire par exemple un peu de World Of StarGate ou encore Dystopia, etc

World Of Stargate :

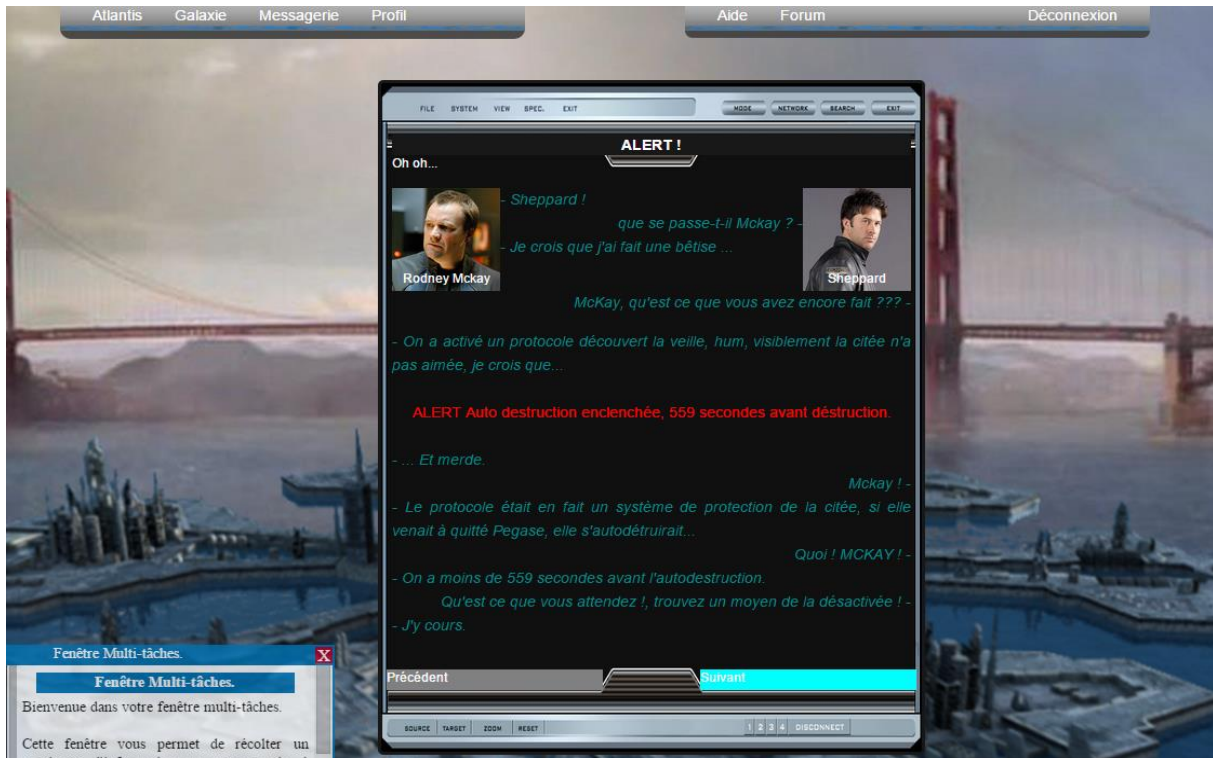
<http://www.worldofstargate.fr/>

Dystopia:

<http://dystopiaonline.fr/>

Comment jouer ?

Une fois un compte créé, le joueur se connectant pour la première fois à face à lui une petite fenêtre lui narrant les aventures de StarGate Atlantis. En effet, le jeu se situe à la fin de la saison 5 de cette série, Atlantis est sur terre, et le Dr. McKay a fait une petite bêtise... C'est ainsi qu'on embarque pour la première fois dans ce vaisseaux à travers Multigates.



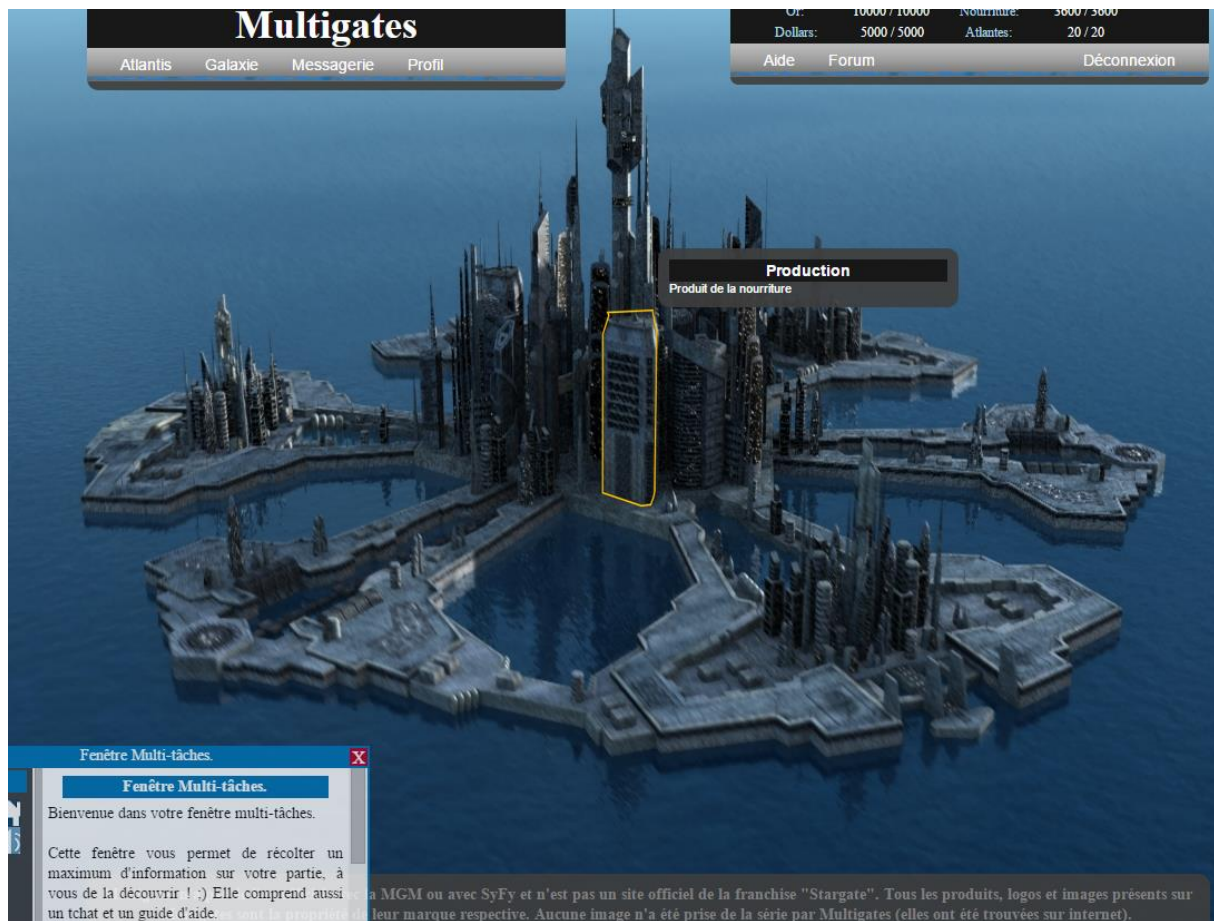
Où quand McKay fait une bêtise...

Le vaisseau repart dans l'espace, afin d'éviter d'exploser, et une fois ceci fait, on obtient le plein contrôle du vaisseau. Il s'agit dès lors de commencer à développer son vaisseau au milieu de la galaxie, trouver une planète, construire des soldats, attaquer des joueurs, gérer son énergie, être un bon scientifique, et plus encore, c'est les qualités nécessaires au joueur pour progresser.



Gérer son énergie.

Pour commencer, on clique donc sur les bâtiments auxquels on a accès, ce qui aura pour effet d'ouvrir l'interface du bâtiment concerné, afin que l'on puisse l'utiliser. L'utilisation d'un bâtiment se veut instinctif et réaliste. Le but recherché est en effet la simplicité et le réalisme (plus d'infos sur le sous-chapitre 2).



Rien de plus simple, un seul clique et le bâtiment s'ouvre !

Une fois la version beta terminée, les temps de construction, d'attaque, et autre seront rallongés. En effet, ce type de jeu est fait pour être joué par petites tranches de quelques minutes plusieurs fois par jour, lorsqu'on fait de petites pauses, par exemple. En d'autres termes, on ne joue pas à Multigates car on a envie de jouer à un jeu, mais on y joue, car on a cinq minutes de pause dans une autre activité. Multigates est une activité secondaire. Du moins dans cette version (voir sous-chapitre 3).

Comment est structuré Multigates ?

Le jeu se veut intuitif, il a donc fallu le structurer d'une manière claire et accueillante. C'est d'ailleurs ce qui m'a donné l'idée principale, faire du vaisseau la base même du jeu, le fond d'écran de celui-ci. En effet, dans la plupart des jeux, le bâtiment principal ou la planète principale a une place réduite dans l'interface.

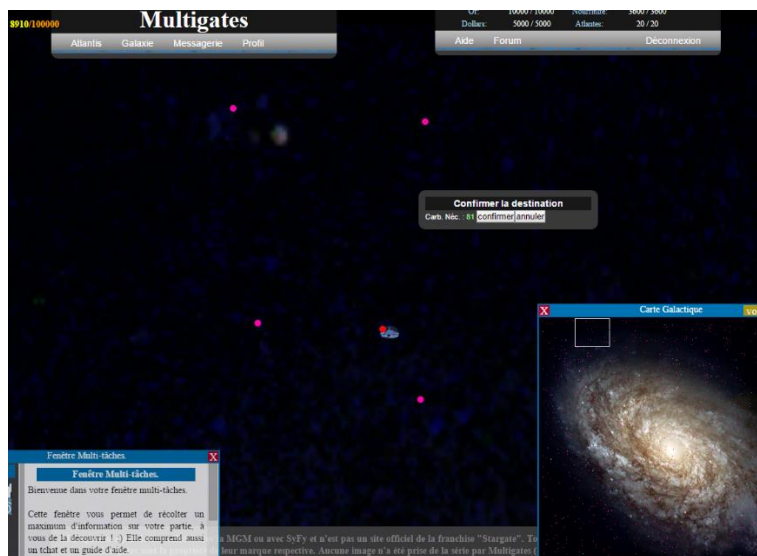
Alors que dans Multigates il s'agissait vraiment d'inviter le joueur sur Atlantis et donc d'en faire le point central.

Et c'est un pari réussi haut la main : l'interface fonctionne comme je le souhaitais. De plus, elle s'adapte à tous les types d'écrans.

Je tenais à ce que les joueurs ne perdent jamais de vue Atlantis pour bien ancrer cette idée de pièce maitresse. Ainsi j'ai choisis un système de fenêtre pour les bâtiments : au clique d'un bâtiment, ce n'est pas une autre page qui est chargée, mais bien une fenêtre qui s'ouvre sur Atlantis. Cela permet non seulement de ne pas toujours devoir charger une page lorsqu'on clique (enfin si, mais uniquement en javascript), de garder Atlantis de vue et de pouvoir y retourner à n'importe quel moment, simplement en cliquant à côté de la fenêtre ouverte.

L'interface est donc, de mon point de vue, une des plus grandes réussites du jeu.

La Galaxie, seule page où l'on ne voit plus Atlantis en arrière-plan, est effectivement une exception, mais elle est mûrement réfléchie. Lorsque j'ai voulu créer la galaxie, j'ai tout de suite pensé à ce système, car il permet de se sentir tout petit dans cette vaste galaxie, mais surtout d'y être intégré. Une simple fenêtre galaxie n'aurait en effet pas permis de se sentir « en plein dedans », car elle n'aurait été qu'une simple interface. Alors que là, c'est tout ce que l'on voit, une vaste galaxie remplie d'autres joueurs, cela invite le joueur à s'y balader. Et c'est l'objectif principal.



Page Galaxie

De plus, lorsqu'Atlantis est dans l'espace, on la voit sur la galaxie, et l'on peut la déplacer. Ce système aurait été ridicule dans une fenêtre, car on aurait eu Atlantis en fond d'écran, dessus une fenêtre de la galaxie, et, sur la galaxie, Atlantis se déplaçant en tout petit. Assez perturbant de mon point de vue.

Ainsi le jeu s'articule sur des interfaces mûrement réfléchies.

Pour l'ordinateur de bord, j'ai voulu laisser la possibilité au joueur de l'utiliser comme il le souhaitait, en permettant de le déplacer, éteindre, agrandir, réduire, etc. Ce qui permet au joueur de l'utiliser à sa guise, et de différentes manières.

Le futur

Après la publication de la version une, je travaillerai certainement sur un système d'alliance entre joueurs. Mais la chose la plus importante est que j'ouvrirai ce futur aux autres personnes, dans le sens où je proposerai à n'importe qui d'amener des idées, optimisations et autre via le forum, que je développerai par la suite dans le jeu en fonction de la demande.

C'est en effet un point essentiel que j'aimerais amener à Multigates ; la collaboration. Faire de ce jeu un projet ouvert, et écoutant son publique.

J'envisage aussi d'intégrer peut-être dans le futur des mini-jeux sous invitation à jouer avec des amis une fois connecté sur Multigates. Cela fera alors du jeu non plus une page secondaire ouverte sur le navigateur, mais bien un jeu où l'on peut rester jouer avec des amis plus de quelques minutes d'affilées et cela dans une ambiance sympathique et conviviale.

En tout cas, il reste beaucoup de travail, alors le futur du jeu est assuré grâce aux nouveautés qui arriveront prochainement.

Toutefois, pour l'instant et jusqu'au premier janvier, je me concentrerai uniquement sur les corrections et optimisation du codage déjà présent.

```
249 ||Galaxie||
250 Déplacement de flottes
251 ->algorithmes combats
252 ->Pas de tire
253
254
255 ||Planete||
256 -mise en forme
257 -Nourriture
258 -Or
259 -Déplacement Jumper
260 -Lieux
261 -Hangar à jumper Juin
262
263 ||Alliance||
264 -mise en forme
265
266
267 ||Inscription revoir||
268 -tables, etc
269
270 ||Tchat||
271 -finir fenetre
272 -guide aide
273 -reste
274
275 ||Classement||
276 -Faire
277
278
279 ||Scénario||
280 -base
281
```

Il en reste des choses à coder !

Conclusion

Après 2 ans de développement et quelques 215h de travail, Multigates est enfin ouvert au publique.

Comment conclure une telle aventure ? Qu'est-ce que ce parcours m'a apporté ? Quels souvenirs vais-je garder de mon travail de maturité ?

Le parcours

S'il y a bien quelque chose dont je suis fier, c'est le chemin parcouru pour en arriver là. Ça n'a pas toujours été facile, j'ai manqué de temps pour me consacrer à Multigates certaines fois, j'ai même fait d'autres choses pendant plusieurs semaines d'affilées, j'ai été confronté à certains bugs répétitifs et lassants, mais, toutefois, je n'ai pas abandonné.

On dit que la plupart des projets web meurent sans être nés, qu'on vise souvent trop haut, ou que, si l'on est seul, on se démotive rapidement. Oui, j'ai quelques fois senti ces points, mais j'ai toujours pu lutter et avancer, jusqu'à arriver ici aujourd'hui, et cela est donc une belle réussite.

A la fin d'un tel projet, on se sent fatigué, mais aussi fier du devoir accompli. De plus, l'ouverture du projet aux joueurs sera en quelque sorte un renouveau. Laissez les joueurs me dicter les futures améliorations, rendre le jeu communautaire, c'est quelque chose qui me motivera et je m'en réjouis.

Multigates est un projet qui m'est venu lorsque j'ai vu un autre jeu de ce type presque mourir, j'avais à l'époque presque aucune connaissance en mysql, très peu en php, encore moins en javascript, et pourtant je n'avais qu'une seule envie, folle à l'époque, mais je l'avais et j'ai persisté dans mes démarches : je voulais créer mon propre jeu !

Après un long parcours, c'est chose faite, et c'est vraiment beau.

Le dernier Mot

Ce projet de maturité m'a donc apporté énormément, bien plus que de nouvelles connaissances en langage web. En effet, cela m'a permis de réaliser un projet de A à Z par moi-même, de voir ce que cela représentait et surtout de le faire sur le long terme.

J'ai dû faire face à des situations qui demandaient beaucoup de logique ou d'improvisation. J'ai dû apprendre à mieux structurer mon code, ou encore à créer de la nouveauté à partir de l'inspiration ressentie via d'autres jeux.

Multigates n'est pas que quelques lignes de code, c'est bien plus, c'est un vrai projet de maturité, car c'est bien là une chose que cela m'a apporté dans ma manière de travailler : de la maturité.

Alors, je n'ai plus qu'une chose à dire : bons jeux à vous sur Multigates !