Comment fonctionne le système de Spells?

Galathil

Important : Ce tutoriel a été rédigé par Nobodie pour le site wow-emu.fr. Je le recopie ici à l'identique, Il se peut que certaines informations soient obsolètes à ce jour. Bonne lecture !

Comment fonctionne le système de Spells?

Prélude : Ce sujet n'est pas original, il a été transféré d'un précédent forum d'émulation en date d'Octobre 2013. J'en suis l'auteur, je le transfère donc pour les besoins de ce forum. Bien qu'il soit en partie ré-écrit, il se peut que certains termes soient inadaptés ou ne soient plus d'actualité.

Bonsoir à tous,

nous allons voir aujourd'hui comment fonctionne le système de sorts. Vous le savez probablement, mais ce système est probablement le plus bordélique d'un serveur Wow x) Pour une raison toute simple : Les spells peuvent faire absolument tout et n'importe quoi :

- Faire des dégats
- Healer (faire des dégâts négatifs, en fait)
- Transformer un joueur (changer son display ID)
- Lui changer sa vitesse
- Invoquer un Pet
- Lancer un duel
- Invoquer une monture
- Jouer un son
- Détruire le décor autour de lui (certains spells permettent d'activer un effet d'un gameobject, voir de la map elle même)
- Changer de phase
- Se stacker pour apporter un bonus
- ...

En plus, ils peuvent être castés sur différents types de cible :

- Un joueur/créature ciblée
- Une zone au sol
- Soi même
- Une zone autour de soi
- ...

Et pour ne rien arranger, ils consomment différentes "ressources"

- De la mana
- De la rage
- Du Chi (coucou les pandas)
- De la vie
- ...

Ah j'oubliais, il peut aussi être casté de façon instantanée, ou demander un temps de cast. Certains ont un cooldown, d'autres non, d'autres encore dépendent d'autres spells (les talents par exemple) et sont modifiés selon les auras qu'on possède sur soi. Du bonheur en perspective, donc

Vous l'avez déjà compris, le système de spells est de nature très complexe pour la simple raison qu'il possède des ramifications dans l'ensemble du serveur. Mais alors, comment fonctionne un tel système ? C'est ce que nous allons voir dès à présent

1) Qu'est-ce qu'un sort?

1) Qu'est-ce qu'un sort?

Techniquement, un sort est un "objet" que l'on initialise lorsqu'on canalise le spell, puis qu'on lance à l'endroit souhaité (cette phase récupère la liste des cibles possibles - unique lorsqu'on cible quelqu'un, plusieurs lorsqu'il s'agit d'un sort de zone). Le sort "vole" et atterrit sur sa/ses cibles. A l'impact, le sort applique son/ses effets sur les cibles.

Voilà dans les grandes lignes comment fonctionne un spell. Bien évidemment, si tout était aussi simple, on aurait pas autant de soucis à corriger le système :3

2) Un effet ? Des effets ? Une aura ? Des auras ?

Reprenons étape par étape. L'initialisation du spell est simple, on récupère tout un tas d'informations sur le spell à l'aide des données présentes dans les DBCs (la base de données binaires récupérée depuis le client). On récupère notamment le temps de cast, tout va bien. On lance le spell à l'endroit souhaité, et on récupère la liste de cibles potentielles (on a bien évidemment récupéré les critères de ciblage du spell auparavant pour savoir quelles cibles peuvent être atteintes). Le sort "vole" un certain temps puis atteint sa/ses cibles. La, ça se complique. ça se complique beaucoup, même. Vous vous rappelez de ce qu'on a dit tout à l'heure ? un sort peut agir sur absolument tout et n'importe quoi en jeu ? On y arrive, justement, et on va prendre l'exemple de sorts du mage Feu.

Exemple de spell cool : La boule de feu (**Boule de feu**)

Si on regarde la page wowhead de ce sort, on remarque pleeein de choses. Premièrement, on reçoit tout un tas d'informations qui sont directement récupérées des DBCs (et oui, Wowhead fait comme nous, il cherche les infos là ou elles existent □). Le coût du sort, sa portée, son temps d'incantation. Tout ça, ça ne nous intéresse pas. Nous ce qui nous intéresse, c'est ce qui se trouve juste en dessous : la catégorie EFFET.

En EFFET (ahah), cette catégorie est très importante pour nous car chaque sort est en fait divisé en plusieurs effets. Sur cataclysm, un sort pouvait avoir entre 1 (ben oui parce qu'un sort qui n'a aucun effet, c'est légèrement useless) et 3 effets. Il existe en tout 183 effets "primitifs" différents sous Cataclysm. Ca devient compliqué ? Yeah !

Parmi ces effets, on peut voir les effets suivants :

- EffectSchoolDamage
- EffectBlock
- EffectSummon
- EffectHeal/EffectHealPct
- EffectScriptEffect/EffectDummy
- EffectApplyAura

Euh attends Nobo, les quatre premiers, je vois ce que c'est, mais EffectScriptEffect/EffectDummy, c'est quoi ? Patience mousquetaire □

L'effect Dummy est effectivement tout à fait particulier, car il regroupe en fait tout ce que Blizzard n'a pas pu classer dans les autres. Quand vous voyez un EffectDummy ou un EffectScriptEffect (comme dans **ce spell**), et bien vous êtes mal barrés ^ ^ Ces deux effets signifient en fait que leur système est développé du côté du serveur officiel, et que donc vous devez vous même faire le travail pour créer le script par vous même. Impossible d'automatiser, il faut le refaire. Fun, non ?

Et l'EffectApplyAura ? J'y viens

Il existe deux grands types de "choses" que peut faire un spell. Les effets, qui sont instantanés (ils se produisent à un moment précis) et les auras qui durent un certain temps (on les caste sur la cible et ils durent un certain temps avant de se dissiper, comme les poisons, ou les heal sur une durée). Les effets, c'est bon, on vient d'en parler. Mais les auras, comment ça marche ? Et bien techniquement, une aura qui dure un certain temps, on est bien d'accord que c'est un effet de durée qui se produit à partir d'un moment précis, n'est-ce pas ? Et bien, vous venez de comprendre l'EffectApplyAura. Il s'agit d'un effet qui s'execute au moment du cast, et qui a pour objectif ... de lancer une Aura qui va durer un certain temps après le cast x) Plutôt ingénieux, non ?

Prenons par exemple ce spell (**Masse critique**), vous remarquez qu'il y a deux types d'aura différents, n'est-ce pas ? Et bien c'est tout simplement parce qu'effectivement, il existe plusieurs types d'auras. 323 sur Cataclysm exactement □

Petite annexe : les chiffres pour MoP

Comme MoP se complique un peu pour tous les systèmes, celui de spell n'échappe pas à la règle. Désormais, un spell peut avoir jusqu'à non pas 3 effets, mais 32 effets différents. Il existe 4 nouveaux SpellEffects et 31 nouveaux SpellAurasEffects. Fun n'est-ce pas ?

Comment ça marche à l'intérieur du Core?

Tout ce système est ingénieux mais est assez complexe. Pour implémenter un tel projet, le choix (judicieux) s'est porté sur un système de tableau de fonctions. Petit préambule.

Imaginons que vous souhaitiez développer une calculatrice. Vous avez les quatre opérations de base : l'addition, la soustraction, la multiplication, et la division. Imaginons un système très basique. Vous entrez vos deux nombres, et vous cliquez sur le bouton d'addition. La, vous définissez expressément que si vous appuyez sur le bouton +, alors vous faites une addition. Si par contre, vous appuyez sur le bouton - , alors vous appelez la fonction de soustraction, idem pour les deux autres options. Ce système, ça va très bien pour une petite calculatrice, mais imaginez développer une calculatrice avec 100 boutons pour 100 calculs différents. Allez-vous faire 100 conditions selon chacun des boutons cliqués pour accéder à la fonction correspondante ? non, vous allez utiliser un tableau de fonctions.

Un tableau de fonctions n'est ni plus ni moins qu'une liste qui associe un index (le numéro de la case dans le tableau) à une fonction, la fonction addition par exemple. Il suffit par exemple d'identifier le bouton + par le nombre 14 par exemple (choix arbitraire) et d'associer la fonction d'addition à la case 14 du tableau. Ainsi, à chaque pression sur le bouton +, le système va directement à la quatorzième case du tableau ... et appelle la fonction "addition". Magique \o/

Revenons à nos moutons : les effets de spell. 183 spelleffects, ça ressemble légèrement à un projet de grosse calculatrice, n'est-ce pas ? appliquons le même raisonnement.

Chaque effet a un numéro, et à chaque case du tableau, on associe la fonction correspondante. On a donc un tableau de 183 éléments. On reçoit l'effet 134 ? aucun problème, on appelle directement la fonction liée à l'effet 134 (ici, l'effet KillCredit qui incrémente un compteur de morts pour un HF en particulier - on parlera du système de HFs dans un autre guide, rassurez-vous). Mais alors, que fait-on de l'effet 6 qui correspond au EffectApplyAura ? Rappelez-vous, on a 323 auras différentes à traiter encore. Mais ... ça ne ressemble pas également à une grosse calculatrice ? En réalité, on applique exactement le même principe. Chaque identifiant d'aura est lié à une fonction dans un second tableau d'auras cette fois-ci. On a donc deux tableaux de "handlers" (les fonctions chargées de traiter des informatons) qui sont appelés dynamiquement lorsqu'on reçoit un spell autant de fois qu'il n'y a d'effets/auras dans le sort.

Il ne reste plus qu'à coder les 183 effets du système de spell et les 323 auras et vous avez un serveur full debug =3 (oui bon, il reste tous les effectDummy et les aurasDummy qui doivent être développés manuellement, ainsi que deux-trois "trucs" dont nous parlons juste après

4) Des trucs ? Quels trucs ?

Evidemment, il ne faudrait quand même pas que ce soit trop simple. Jusqu'à présent, on a assumé le fait de récupérer les données depuis les DBCs qui sont extraites du client. Ca, c'est dans l'idéal ou on a tout compris des DBCs. Le problème ? On ne comprend justement pas la totalité des DBCs. Les informations des spells sont divisées en une trentaine de tables binaires qu'il a fallu découper et comprendre. Or, nous ne connaissons pas toutes les informations qu'elles comportent. Par exemple, certaines colonnes de données sont vides la plupart du temps, mais exceptionnelement pour 20 spells différents (sur plus de 130 000 spells existants), la colonne va contenir des données différentes de 0. Que veulent-elles dire ? Certainement quelque chose d'utile, mais pour 20 spells différents, c'est plus simple de corriger manuellement les données que de déployer des ressources pour 1) comprendre la dbc 2) identifier ce champ 3) imaginer un système qui gère cette colonne 4) le coder 5) le tester. C'est le cas pour beaucoup de colonnes inconnues, et c'est la raison principale pour laquelle le système de spells contient une fonction LoadCustomSpellInfos() dans laquelle on renseigne manuellement les informations non gérées des DBCs. Vous l'imaginez, tous ces "hardfixes" complexifient la maintenance du système et rendent le système bancal. C'est pour cette raison que beaucoup de serveurs privés ont des problèmes avec le système de spells

5) A la fin qui que c'est qui gagne ?

Si on met Blizzard dans la liste de serveur, c'est évidemment Blizzard qui gagne Ils savent ce que contiennent les DBCs (évidemment, ils les ont eux mêmes créées -_-). Blizzard mis à part, le serveur qui s'en sort le mieux, c'est celui qui a le mieux compris les DBCs et qui a implémenté le maximum d'effets. Plus d'effets automatisés, c'est moins de Dummy à corriger au cas par cas. On passe du temps une fois sur un effet bien complexe, et on corrige potentiellement un grand nombre de sorts d'un coup. C'est tout de même mieux d'annoncer que du jour au lendemain, Le ciblage de 3200 spells a été corrigé, n'est-ce pas ?

Une fois de plus, tout est une question de connaissance du système. Celui qui se fout des DBCs et espère tout corriger à la main, n'a pas fini d'en baver. Rappelez-vous, 130 000 spells existants. Il y a peut être d'autres choses à faire, plus intelligentes, que de corriger chaque spell à part :3

Voilà qui conclue mon troisième "guide" du fonctionnement de Wow. N'hésitez pas à me donner votre point de vue constructif, Je suis resté au plus simple, toujours pas de code pur et dur, les explications sont vulgarisées au maximum afin de toucher un maximum de gens et de leur

donner, pourquoi pas, le goût de l'aventure et l'envie de se lancer à corps perdu dans l'émulation N'oubliez pas que beaucoup de développeurs de l'émulation ont été autodidactes, alors profitez des guides que nous mettons à votre disposition pour comprendre plus rapidement le système. Et à côté de ça, travaillez votre connaissance de la programmation orientée objet, en faisant du C++, du Java, ou n'importe quel autre langage moins bon que le C++ (oh c'est bon, il est 5H30, un petit troll du matin ça fait pas de mal :x)

---

kazuma

Merci pour avoir ressorti ce guide des abysses, il est vraiment très riche en informations, exactement les informations que je cherchais depuis looongtemps
