

Code

Issues 38

Pull requests

Discussions

Actions

Projects

Wiki

Security

Insights

[Tutoriel - LUA]Faire un système de file d'attente #6

New issue


[Jump to bottom](#)

Open

Open-Wow opened this issue May 22, 2020 · 1 comment

Labels

[Tutoriel - LUA]

 **Open-Wow** commented May 22, 2020 • edited

...

Faire un système de file d'attente

Tutoriel par Kazuma

Salut Open-Wow ! Aujourd'hui je vous montre comment faire un système de file d'attente. Eh oui, c'est quand même plus simple et plus classe de faire bien les choses surtout quand il s'agit d'éviter les laborieux "Ceux qui veulent participer à l'événement faites "+1" !" de vos maîtres de jeu dans les chats... On est plus en 2008 (ou je sais pas quand ! 😊)

L'objectif ça va être de pouvoir permettre aux joueurs de pouvoir s'inscrire/se désinscrire auprès d'un PNJ proprement.

1 - Pré-requis

Avant toute chose vous vous en serez sûrement douter, il vous faut un gossip pour que le joueur puisse interagir avec le PNJ.

Concernant les gossip si vous ne savez pas ou ne comprenez pas comment ça marche je vous recommande de consulter [mon tutoriel sur les Gossip](#) car je ne vais pas ré-expliquer le fonctionnement dans ce tutoriel.

2 - Comment s'y prendre ?

En réalité c'est très simple, après avoir mis en place un gossip à minimum 2 choix sur un PNJ.



Ce qui nous importe c'est le mécanisme qui se cache derrière ces deux choix, donc direction votre fonction `onGossipSelect`

Vous devez avoir votre bloc d'instruction IF (Si) au moins comme ceci :

```
local function OnGossipSelect(event, player, object, sender, intid, code, menuid)
    local pName = player:GetName()
    local pGuid = player:GetGUIDLow()

    if (intid == 1) then
        -- choix 1
    elseif (intid == 2) then
        -- choix 2
    end
end
```

Pour que vous preniez l'habitude de faire du code propre, on ne va pas directement écrire du code pour chaque situation, on va se servir de deux fonctions qu'on va créer nous-même :

- **inscription** : On lui donnera comme paramètre le nom du joueur, elle gèrera l'inscription, elle devra vérifier si le joueur n'est pas déjà inscrit et si ce n'est pas le cas : l'inscrire.
- **desinscription** : On lui donnera comme paramètre le nom du joueur, elle gèrera la désinscription, elle devra vérifier si le joueur est déjà inscrit avant d'essayer de désinscrire le joueur.

A - Mais comment on inscrit un joueur ?

Bonne question, c'est ici que repose la clé du script !

Il y a deux façon d'enregistrer une valeur, de sauvegarder une donnée :

- **La base de donnée** : C'est la première chose à laquelle on peut penser j'imagine quand on est novice dans le domaine, en effet c'est une bonne façon de faire et c'est réalisable en Lua grâce à une requête SQL , et si c'est bien effectuer vous êtes sûr de conserver vos données.

Mais pourquoi s'embêter à stocker des informations qui ont déjà peu de valeur et qui ne vous seront absolument plus d'aucune utilité une fois passer un délai ?

Pourquoi faire compliqué quand on peut faire simple ?

(Question à prendre au second degré , parfois il est mieux de faire compliqué... 😊)

- **Les table LUA** : Les tables sont comme des tableaux que vous pouvez mettre en place dans votre code, c'est une fonctionnalité

vraiment puissante, car vous pouvez stocker énormément d'informations dans ces tableaux sans pour autant embêter la DB.
[Le tutoriel d'iThorgrim](#) saura vous expliquer parfaitement ce que vous devez savoir à ce sujet

Nous allons donc nous servir d'un table LUA afin de s'en servir de "Liste d'attente".

3 - Le Code

Rendez-vous au début du script ou vous créez un table LUA intitulé "ListeAttente" : dedans nous y stockeront les noms des joueurs inscrits.

Créez également une variable contenant le nombre de joueur nécessaire au déclenchement de l'événement , ça sera le même principe que les champs de batailles où un nombre de joueur est requis.

(Mettez minimum 2 pour un test plus concret)

```
6  -- Gestion file d'attente
7  local ListeAttente = {}; -- Liste d'attente vide (liste de nom)
8  local NombreJoueur = 1
9
```

Désormais, nous avons notre liste d'attente vide, et un nombre de joueur minimum, maintenant il nous manque nos deux fonctions dont j'ai parlé tout à l'heure.

Tout d'abord, placez-vous avant vos fonctions par rapport aux gossip !

C'est important car vous ferez appel à ces fonctions (inscription & desinscription) dans votre fonction OnGossipSelect, c'est pourquoi quand l'interpréteur va lire votre script de haut en bas, il doit connaître la fonction inscription pour pouvoir l'utiliser, sinon il y aura erreur !

```
11 -- Fonctions : Inscription
12 local function inscription(pName)
13
14
15 end
```

- **Paramètre de la fonction :**

- **pName** : Quand vous ferez appel à la fonction, vous devrez donner le nom du joueur entre les parenthèses, je trouve ça plus propre de procéder ainsi plutôt qu'en allant chercher son nom avec sa fonction alors qu'on aurait pu directement le donner entre les parenthèses de la fonction en tant que paramètre.

```
local trouve = true

for numero, nom in pairs(ListeAttente) do -- Parcours liste d'attente
    if (nom == pName) then -- Si il existe déjà ce nom sur la liste
        trouve = false
        break
    end
end

if (trouve == true) then
    table.insert(ListeAttente, pName) -- Inscription dans la liste
end

return trouve
```

```

11  -- Fonctions : Inscription & Désinscription
12  local function inscription(pName)
13
14      local trouve = true
15
16      for numero, nom in pairs(ListeAttente) do -- Parcours liste d'attente
17          if (nom == pName) then -- Si il existe déjà ce nom sur la liste
18              trouve = false
19              break
20          end
21      end
22
23      if (trouve == true) then
24          table.insert(ListeAttente, pName) -- Inscription dans la liste
25      end
26
27      return trouve
28  end

```

Voici notre fonction, c'est beaucoup de ligne qui apparaissent d'un coup, alors qu'est-ce qu'il se passe exactement ?

- **boucle for** : Ici grâce à une boucle for, on parcourt le table ListeAttente, ainsi on va pouvoir regarder tout les noms contenus dans la liste. Je rappelle que les noms dans la liste représente les joueurs inscrit. la variable "numero" correspond à l'index, la clé ou l'indice (plusieurs façons de l'appeler), disons qu'il représente le numéro de ligne comme dans un tableau excel (1, 2, 3, 4, 5 ...) chaque ligne à sa clé. la variable "nom" est tout simplement la valeur de la clé dans le table, ainsi, chaque ligne contient un nom de joueur.

Imaginez le table visuellement comme ceci :

```

1 - Kazuma
2 - Thanatos
3 - iThorgrim

```

- **trouve = true** : C'est une variable de type booléen, sa valeur est soit "true" (vrai), soit "false" (faux). On va se servir de cette variable pour nous indiquer si le joueur existe dans la liste ou pas.
- **if (nom == pName)** : Ici c'est tout bête on compare le nom trouvé à telle ligne et le nom du joueur qui souhaite s'inscrire.
- **trouve = false** : On met la variable à false, ce qui indique que le joueur ne peut pas être inscrit. (car il est déjà inscrit)
- **break** : C'est un mot-clé en lua qui va vous permettre d'interrompre et sortir d'un bloc d'instruction, nous sortons donc de la boucle car nous avons pu tester le joueur, si nous continuons, le script va machinalement continuer et traiter les autres joueurs alors que ce n'est pas ce que nous voulons.
- **if (trouve == true) then** : Si la valeur de la variable est vrai (donc n'a pas changé)
- **table.insert(ListeAttente, pName)** : On ajoute le nom du joueur dans le table
- **return trouve** : On renvoie la valeur de la variable, ainsi, vous verrez que plus tard à l'utilisation de la fonction, la fonction saura répondre si OUI ou NON le joueur à pu être inscrit.

La fonction desinscription est exactement l'inverse, il vous est facile de la comprendre, on vérifiera si le joueur est bel et bien inscrit, si oui : on le supprime de la table avec table.remove. Sinon, il ne se passera rien :

```

local function desinscription(pName)

    local trouve = false

    for numero, nom in pairs(ListeAttente) do -- Parcours liste d'attente
        if (nom == pName) then -- Si il est bien dans la liste
            table.remove(ListeAttente, numero) -- Suppression dans la liste
            trouve = true
            break
        end
    end

    return trouve
end

```

```

31 local function desinscription(pName)
32
33     local trouve = false
34
35     for numero, nom in pairs(ListeAttente) do -- Parcoure liste d'attente
36         if (nom == pName) then -- Si il est bien dans la liste
37             table.remove(ListeAttente, numero) -- Suppression dans la liste
38             trouve = true
39             break
40         end
41     end
42
43     return trouve
44 end

```

Maintenant que nos deux fonctions sont prêtes on retourne à notre fonction `OnGossipSelect` où l'on va les appeler selon les choix :

```

65 local function OnGossipSelect(event, player, object, sender, intid, code, menuid)
66     local pName = player:GetName()
67     local pGuid = player:GetGUIDLow()
68
69     if (intid == 1) then
70         if (inscription(pName) == true) then
71             player:SendNotification("Vous venez d'être inscrit à la file d'attente avec succès !")
72
73         else
74             player:SendNotification("Vous êtes déjà inscrit à la file d'attente")
75         end
76
77         OnGossipHello(event, player, object) -- Evite de se retrouver avec un gossip figé
78
79     elseif (intid == 2) then
80         if (desinscription(pName) == true) then
81             player:SendNotification("Vous n'êtes plus inscrit à la file d'attente")
82         else
83             player:SendNotification("Vous n'êtes pas inscrit dans la file d'attente")
84         end
85
86         OnGossipHello(event, player, object) -- Evite de se retrouver avec un gossip figé
87     end
88 end

```

```
if (inscription(pName) == true) then
```

Vous remarquerez que nous mettons à profit l'utilisation du " `return trouve` " dans les fonctions. Les fonctions retournent une valeur, elles retournent soit " `true` " ("vrai") soit " `false` " ("faux")

Ce qui nous permet de savoir si le joueur a bien été inscrit ou non. Pratique non ?

Et donc on peut informer le joueur du résultat via la fonction `SendNotification` 😊

4 - Déclenchement de l'événement

Bon c'est bien beau mais il reste une dernière étape, il faut provoquer le déclenchement de votre événement si suffisamment de joueurs sont inscrits.

Pour l'exemple on va mettre une fonction qui téléporte tout les joueurs inscrits.

```

46 local function evenement()
47
48     if (#ListeAttente == NombreJoueur) then
49         for numero, nom in pairs(ListeAttente) do
50             for _, player in ipairs(GetPlayersInWorld()) do
51                 if (player:GetName() == nom) then
52                     player:Teleport(571, 5804.149902, 624.70996, 647.767029, 1.640000)
53                 end
54             end
55         end
56
57         ListeAttente = {};
58     end
59 end

```

- **if (#ListeAttente == NombreJoueur) then** : On vérifie si le nombre de joueur inscrit est égal au nombre de joueur requis pour l'événement.
- **for numero, nom in pairs(ListeAttente) do** : On parcourt la liste d'attente contenant les noms.
- **for _, player in ipairs(GetPlayersInWorld()) do** : On parcourt une table contenant tout les joueurs en ligne.
- **if (player:GetName() == nom) then** : Si le nom du joueur en ligne correspond au nom figurant dans la liste d'attente
- **player:Teleport(571, 5804.149902, 624.70996, 647.767029, 1.640000)** : On téléporte le joueur à Dalaran (par exemple).
- **ListeAttente = {};** : A la fin de la fonction on vide la liste d'attente pour pouvoir s'en resservir correctement, les noms n'ont plus lieu d'y figurer.

Vous n'avez plus qu'à ajouter l'appel de la fonction à l'endroit convenu et vous avez désormais en votre possession un système de file d'attente sympathique en Lua (façon Kazuma).

```

71 local function OnGossipSelect(event, player, object, sender, intid, code, menuid)
72     local pName = player:GetName()
73     local pGuid = player:GetGUIDLow()
74
75     if (intid == 1) then
76         if (inscription(pName) == true) then
77             player:SendNotification("Vous venez d'être inscrit à la file d'attente avec succès !")
78             evenement()
79         else
80             player:SendNotification("Vous êtes déjà inscrit à la file d'attente")
81         end
82
83         OnGossipHello(event, player, object) -- Evite de se retrouver avec un gossip figé
84
85     elseif (intid == 2) then
86         if (desinscription(pName) == true) then
87             player:SendNotification("Vous n'êtes plus inscrit à la file d'attente")
88         else
89             player:SendNotification("Vous n'êtes pas inscrit dans la file d'attente")
90         end
91         OnGossipHello(event, player, object) -- Evite de se retrouver avec un gossip figé
92     end
93 end
94 end

```

5 - Aperçu en jeu

"Se désinscrire sans être inscrit"

Vous n'êtes pas inscrit dans la file d'attente

Merleaux le Magnifique
<Mj>Kazuma



"S'inscrire et téléportation dû au déclenchement de l'événement"



 **Open-Wow** added the [\[Tutoriel - LUA\]](#) label [May 22, 2020](#)



 **Open-Wow** mentioned this issue [May 22, 2020](#)


[Partages - LUA]Système de file d'attente #25

 Open



 **iThorgrim-Hub** changed the title **Faire un système de file d'attente [Tutoriel - LUA] - Faire un système de file d'attente** [Jun 27, 2020](#)



 **iThorgrim-Hub** changed the title **~~[Tutoriel - LUA] - Faire un système de file d'attente~~ [Tutoriel - LUA]Faire un système de file d'attente** [Dec 7, 2020](#)

 **Torvahal** commented [Mar 4, 2021](#)

...

Super, merci pour le partage :D

[Sign up for free](#)

to join this conversation on **GitHub**. Already have an account? [Sign in to comment](#)

Assignees

No one assigned

Labels

[\[Tutoriel - LUA\]](#)

Projects

None yet

Milestone

No milestone

Linked pull requests

Successfully merging a pull request may close this issue.

None yet

1 participant



© 2022 GitHub, Inc.

[Terms](#)

[Privacy](#)

[Security](#)

[Status](#)

[Docs](#)

[Contact GitHub](#)

[Pricing](#)

[API](#)

[Training](#)

[Blog](#)

[About](#)