Configuration du gameplay

Ce chapitre détaille les options disponibles dans le fichier `config.lua` pour adapter le script à votre serveur. Ce fichier est le cœur de la personnalisation du gameplay.

3.1 Framework & Permissions

Cette section permet de définir comment le script s'intègre à votre économie et à vos systèmes de métiers.

- **`Config.FrameworkMetier`** : Définit le framework utilisé par votre serveur.
  - `"esx"` : Pour les serveurs ESX Legacy ou v1.
  - `"qb"` : Pour les serveurs QBCore.
  - `"qbx"` : Pour les serveurs QBox (qbx_core).
  - `"ox"` : Pour les serveurs basés sur ox_core.
  - `"standalone"` : Aucune dépendance métier (tout le monde peut tout faire).
- **`Config.TargetSystem`** : Définit le système de visée (Third-eye).
  - `"ox"` : Recommandé (ox_target).
  - `"qb"` : Pour qb-target.
  - `"none"` : Désactive totalement le système de cible (interactions au clavier uniquement, voir `Config.DefaultKey`). Utile si votre serveur n'utilise aucun système de visée.
- **`Config.JobRestriction`** :
  - `true` : Active la restriction par métier. Seuls les métiers listés ci-dessous pourront utiliser les brancards.
  - `false` : Désactive les restrictions. Tout le monde peut utiliser le matériel.
- **`Config.AuthorizedJobs`** : Configuration fine des grades (si `JobRestriction` est activé). Le chiffre correspond au **grade minimum** requis.
  - `push` : Autorisation de pousser le brancard.
  - `patient` : Autorisation de manipuler le patient (installer/sortir).
  - `stow` : Autorisation de charger/décharger de l'ambulance.
  - `spawn` : Autorisation de faire apparaître un brancard via un item.
  - `extras` : Autorisation de modifier les accessoires (sac, oxygène...).
  - `delete` : Autorisation de supprimer le brancard (Admin/Boss).
- **`Config.MaxInteractionDistance`** (défaut `6.0`) : Distance maximale (en mètres) à laquelle un joueur peut interagir avec un brancard ou un véhicule configuré. Au-delà, les actions sont refusées côté serveur, même si le client tente de les déclencher.

3.2 Gameplay & Physique

Ces options modifient le ressenti et le comportement du brancard en jeu.

- **`Config.DefaultKey`** (`'E'`) : La touche clavier par défaut pour interagir (lâcher le brancard) si le Target n'est pas utilisé.
- **`Config.EnableSlopePhysics`** : Si `true`, le brancard soumis à la gravité roulera tout seul dans les pentes s'il n'est pas tenu ou freiné.
- **`Config.SlowDownWhenPushing`** : Si `true`, force le joueur à marcher (empêche le sprint) lorsqu'il pousse un brancard.
- **`Config.HolsterWeaponBeforePush`** : Si `true`, range automatiquement l'arme du joueur avant qu'il ne saisisse les poignées (évite les bugs visuels).
- **`Config.AutoBrakeOnRelease`** : Si `true`, les freins s'activent automatiquement dès que vous lâchez le brancard.
- **`Config.EnablePickupAnimation`** (défaut `false`, ⚠ **expérimental**) : Si `true`, le joueur se déplace automatiquement devant le brancard et effectue un geste de prise en main contextuel (accroupi si le brancard est au sol, prise directe s'il est debout) avant de le saisir, au lieu d'une prise en main instantanée. Les animations et distances utilisées sont réglables via `Config.PickupAnims` et `Config.PickupWalkDistance` (fichier `stretchers.lua`). Cette fonctionnalité est encore en cours d'ajustement — désactivez-la si vous rencontrez un comportement inattendu.

3.3 Audio et Immersion

- **`Config.EnableWheelSound`** : Active le bruitage réaliste de roulement.
- **`Config.WheelSoundVolume`** : Volume du son (0.0 à 1.0).
- **`Config.WheelSoundDistance`** : Distance à laquelle les autres joueurs entendent le brancard (défaut : 15 mètres).

3.4 Objets & Items

Vous pouvez définir quels items de votre inventaire font apparaître quel modèle de brancard.

Exemple de configuration :

```
Config.ItemsVeh = {
    {hash = `ferno-f2`, item = 'stretcher',  label = 'Ferno F2'},
    {hash = `fernof3`,  item = 'stretcher2', label = 'Ferno F3'},
    {hash = `stryker`,  item = 'stretcher3', label = 'Stryker M1'},
```

```
}
```

- hash : Le modèle 3D du brancard (doit exister dans le jeu).
- item : Le nom technique de l'item dans votre base de données (DB).
- label : Le nom affiché lors du spawn.

3.5 Système d'Inventaire

Par défaut, MasterStretcher utilise le système d'item natif de votre framework (ESX/QBCore/QBox). Si votre serveur utilise ox_inventory comme inventaire (indépendamment du framework de jobs utilisé), il faut le préciser :

- **`Config.InventorySystem`** :
  - `"native"` (défaut) : Utilise le système d'item de votre framework (`Config.FrameworkMetier`).
  - `"ox"` : Utilise ox_inventory.

Configuration requise pour ox_inventory

Si `Config.InventorySystem = "ox"`, MasterStretcher ne suffit pas à lui seul : il faut aussi déclarer l'item côté ox_inventory. Dans votre `ox_inventory/data/items.lua`, pour chaque item de `Config.ItemsVeh`, ajoutez :

```
['nom_item'] = {
    label = 'Brancard',
    weight = 5000,
    client = {
        export = 'MasterStretcher.stretchermod_use_item'
    }
}
```

Remplacez `MasterStretcher` par le nom exact du dossier de la ressource MasterStretcher sur votre serveur (par exemple `MasterStretcher_MasterMods_v17`).

⚠ Le retrait de l'item après usage se gère via le champ `consume` de votre configuration ox_inventory, pas via MasterStretcher.

Revision #11
Created 2025-12-26 21:34:25 UTC by Raph
Updated 2026-09-07 19:29:31 UTC by Raph