

Les options dans Sacred Phoenix

(Tout ce qui est lié à la langue a déjà été réalisé au mois d'avril)

La version 24.49 de PSDK a changé la méthode de gestion des options :

• *En amont (liste des options disponibles) :*

Ce n'est plus codé en dur dans les scripts Ruby ni sur la partie « scripts » de RPG Maker mais dans le fichier **Data\project_indentity.yml**

Pour que les changements soient pris en compte, il est important de le convertir en “.rxdata” en lançant le jeu depuis le CMD puis en rentrant « game » en ligne de commande.

• *En aval (affichage des options disponibles)*

Nuri a complètement réécrit le menu d'options dans la même ampleur que pour le sac. Les anciennes méthodes sont donc devenues caduques, ce qui a cassé mon mod de gestion de la langue.

Conséquences :

– Le jeu crashait au démarrage à cause du mod “**Better Language Choice - GamePlay Load**”. J'ai dû le désactiver et créer un mod de substitution : “**14620 No language nagscreen.rb**”.

– Le jeu crashait à l'ouverture des options avec “**GamePlay Options.rb**” et “**GamePlay Option Element.rb**” J'ai dû les désactiver.

– Le mod “**08130 Better Language Choice - GameData Text.rb**” est le seul qui n'a pas été rejeté par cette MàJ, mais n'est qu'à moitié fonctionnel.

– L'écran de sélection de langue n'est plus au premier démarrage du jeu, mais au lancement d'une nouvelle partie. En effet, la langue est désormais gérée indépendamment sur chaque sauvegarde. (Cela a son importance sur les points suivants.)

– Sans aucun mod lié à la langue, le changement de langue est fonctionnel depuis les options, mais le jeu démarre en anglais pour l'écran titre et le menu principal (langue par défaut). Je préfère que ce soit le `settings.ini` qui détermine la langue par défaut.

– Avec le mod « GameData Text », le changement de langue via les options ne prend effet que sur les images appelées selon la langue (aka la variable interne à la sauvegarde : `$options&.language`). On a donc deux paramètres de langue qui se sont désolidarisés : celui interne à la sauvegarde (`$options&.language`) et celui rajouté par le mod dans le fichier `settings.ini` propre à tout le jeu, mais que le nouveau menu d'option ne sait plus ni lire, ni modifier.

– L'option « Fenêtre » que j'avais semi-neutralisé est de retour. Je souhaite la virer. (Ça me permettrait aussi de me débarrasser de 20 images doublon dans `\graphics\Windowskins`)

– L'option « Volume » est enfin... à moitié fonctionnel : seuls les canaux « SE » (sound effect) et « ME » (music effect) sont affectés. Le volume du canal BGM est actualisé avec du retard : uniquement lorsque la BGM est modifiée.

– Sans aucun mod lié à la langue, le bug de la non réactualisation du menu de jeu au moment de quitter les options... est de retour pour la seconde fois ! (Le fix de Cottens du mod “**14120 GamePlay Menu.rb**” ne fait plus effet. Pas de crash ou message d'erreur, mais la langue n'est plus actualisée à la sortie du menu d'option si elle a été changée.)

Liste des options dans PSDK vanilla (état actuel)

Pour rappel, les options actuelles de PSDK vanilla sont :

- **Vitesse du texte** (opérationnel)
- **Fenêtre** (opérationnel, non désiré)
- **Volume** (à moitié opérationnel : volume de la BGM non rafraichie si pas changée)
- **Animation de Combat** (opérationnel)
- **Style de combat** (opérationnel)
- **Langue** (opérationnel, mais éjection du menu du jeu si modifiée)

Toutes sont liés à la sauvegarde.

Objectifs souhaités pour Sacred Phoenix

Résumé rapide :

-   (**Urgent**) : Retrouver le fonctionnement de la langue tel qu'il était dans Sacred Phoenix en version 24.48 (aka lié à tout le jeu et dépendant d'un fichier de configuration)
-   (**Urgent**) : Réparer encore ce bug de non actualisation du menu du jeu à la sortie des options en cas de changement de langue.
-   (**Urgent**) : Virer le choix des fenêtres de dialogue. (Sacred Phoenix ne proposera qu'un seul style : celui du parchemin.)
-   (**Conseillé**) : Débugger le réglage de la BGM et le désolidariser du réglage des canaux ME/SE. (Donc 2 options de réglage du volume au lieu d'une seule.)
-   (**Conseillé**) : Réparer le choix du style du combat « choix / défini » qui de toutes manière ne fonctionne pas. (La permission de switch gratuit ou non sera dictée dans 95 % des cas par le scénario du jeu et non par le joueur.) (Finalement je laisserai l'option pour les 5 % restants.)
-   (**Conseillé**) : Intégrer dans les options le choix de la difficulté qui dépendent de variables et interrupteurs RMXP. À l'heure actuelle, il en existe trois : **Niveau des dresseurs**, **Mort de vos Pokémon** et **Mort des Pokémon adverses**. Lorsque les clauses Nuzlocke seront implémentées, d'autres options se rajouteront.
-   (**Bonus**) : Coder la gestion du style de combat (choix / défini) et permettre de toggle ce paramètre à travers un interrupteur RMXP.
-   (**Bonus**) : Permettre l'accès au menu « Options » depuis le menu principal (ce qui suit l'écran-titre). Corolaire : en cas d'absence de sauvegarde, le menu principal proposera « Nouvelle partie » et « Options ». (À terme, c'est de me rapprocher du fonctionnement de *Pokémon Uranium*.)
-   (**Bonus**) : Jouer la musique « Intro - Main Menu » dans ce menu principal. (Comme pour Pokémon or et argent.)
-   (**Bonus**) : Créer trois (ou quatre) sous-menus d'options : Réglages, Difficulté, Langue (et Commandes). Quand le joueur choisira « Options » depuis le menu du jeu (ou depuis le menu principal si c'est implémenté), un menu de second niveau proposera ces 3 ou 4 choix. Je peux gérer le menu de langue en full événement making si la scène appelle un événement commun.

Ce qui est « **Urgent** » est indispensable à faire pour que je puisse release une version propre de Sacred Phoenix.

Ce qui est « **Conseillé** » fera un peu plus propre et sera la bienvenue si vous avez le temps de vous en occuper, mais je peux release sans ces objectifs atteints.

Ce qui est « **Bonus** » est un bonus. Ce sont des objectifs de moyen et long-terme.

Cheminement pour atteindre ces objectifs

1) Permettre aux options de se baser en écriture / lecture sur 3 types de variables :

– **Des variables internes à la sauvegarde (fonctionnement actuel de toutes les options)**

Cela sera propre à chaque sauvegarde.

En nous privant de coder en dur dans le code les mécanismes de lecture / écriture de chaque option via la centralisation sur le **project_indentity.yml**, le panachage de méthodes est devenu impossible. Seul ce fonctionnement est possible sur PSDK Vanilla.

– **Des variables et interrupteurs RPG Maker XP**

Cela sera propre à chaque sauvegarde.

Utilisées notamment pour le paramétrage de la difficulté. C'est ce qui me permettra de rajouter des options custom le plus facilement, l'évent-making m'étant bien plus accessible que coder une nouvelle option en dur dans des scripts.

À noter qu'une fois toutes ses clauses implémentées à 100 %, le Mode Nuzlocke ne sera plus ré-activable depuis les options (seulement via des événements en début de jeu). L'inverse sera néanmoins possible. (Désactivable à tout moment, mais plus ré-activable s'il a été désactivé.) En attendant, le joueur pourra changer librement tous ses paramètres de difficulté depuis le menu.

– **Des variables liées à tout le jeu, notées en clair sur un setting.ini**

Cela sera commun à toutes les parties. La **langue** en fait partie de même que les **commandes**.

Je me tâte aussi de mettre le **réglage du volume** ici pour des raisons évidentes. (Éviter la musique assourdissante à l'écran-titre si le joueur a baissé significativement le volume au sein de sa sauvegarde.)

À noter qu'un fichier **input.yml** est présent dans le dossier "Saves" et renferme le mapping des touches clavier. Si c'est plus sécurisant, lire/écrire les paramètres des options sur ce format « yml ».

2) Trois (ou quatre) menus séparés :

Je me tâte potentiellement de faire 3 ou 4 menus séparés :

- un pour les options classiques nommé « **Réglages** » (vitesse texte, son, animation combat...)
- un pour les paramètres de gameplay nommé « **Difficulté** ». Correspond aux modes de mortalité des Pokémon, à la, force des dresseurs et toutes les futures clauses de Nuzlocke.
- un pour la langue nommé « **Langue** ».
- Et le quatrième qui pourrait rejoindre cette famille s'appellerait « **Commandes** ». Il existe déjà : il suffit de presser [F1] en jeu pour y accéder, mais peu de joueurs le savent.

Quand le joueur sélectionnera « Options » depuis le menu du jeu, un sous-menu avec ces trois ou 4 embranchements apparaîtra.

Le fait de séparer en 3 (ou 4) menus est lié aux trois types de variables possibles que je souhaite pour des options.

Le réglage du volume fera probablement exception, car je souhaite le rendre commun au jeu en entier (histoire que le volume soit au bon niveau dès le démarrage du jeu).

3) Optimiser la gestion de la langue :

On m'a dit qu'un fichier ini comporte un risque de sécurité.

Dans le dossier « Saves », j'ai constaté la présence d'un « input.yml » qui est utilisé pour le mapping des touches. (Mapping commun à tout le jeu et non une sauvegarde spécifique... on notera d'ailleurs que le menu de mapping des touches est distinct du menu

Si c'est plus approprié : **passer au format YML pour le paramétrage de la langue.**

J'ai constaté aussi que 2 variables sont utilisées pour un changement de langue sur PSDK moddé (contre une seule en version vanilla).

À voir si une seule pourrait suffire.

Le choix de la langue Réalisé

J'ai fait un petit tableau récapitulatif.

Par « **textes** », j'entends les textes appelés depuis les CSV.

Par « **images** », j'entends les images appelées par script et qui sont différenciées linguistiquement selon la valeur de `$options&.language`. (Spritesheets des types, Spritesheets des statuts, scène de nommage, scène des noms des starters à la Colline du Phénix...)

Mécanisme	24.48 vanilla	24.48 moddé (idéal)	24.49 vanilla	24.49 moddé (à déboguer)
Langue initiale	Choix imposé au premier démarrage, puis figé	Celui paramétré dans le setting.ini	Choix imposé au démarrage d'une nouvelle partie	Partiellement celui paramétré dans le setting.ini, anglais (par défaut) pour les images
Variables utilisées pour la langue	Une seule : <code>\$options&.language</code>	Deux en tout : La valeur du fichier .INI + <code>\$options&.language</code>	Une seule : <code>\$options&.language</code>	Deux en tout : La valeur du fichier .INI + <code>\$options&.language</code>
Variable utilisée pour charger les textes	<code>\$options&.language</code>	Fichier .INI	<code>\$options&.language</code>	Fichier .INI
Variable utilisée pour charger les images	<code>\$options&.language</code>	<code>\$options&.language</code>	<code>\$options&.language</code>	<code>\$options&.language</code>
Lecture de la variable dans le menu options	Néant	Fichier .INI ou <code>\$options&.language</code> (?)	<code>\$options&.language</code>	<code>\$options&.language</code>
Écriture de la variable via les options	Impossible	Les deux à la fois	La valeur de <code>\$options&.language</code>	La valeur de <code>\$options&.language</code> uniquement
Autres méthodes	Forçable via événement-making	Via événement-making (commande complète)	Via événement-making (commande simple)	Via événement-making (commande complète)
Langue liée à	Tout le jeu	Tout le jeu	La sauvegarde active	À moitié tout le jeu et à moitié la sauvegarde active

Je constate qu'avec mon mod, deux variables sont utilisées en parallèle pour déterminer la langue.

En 24.48 moddé, lorsque la langue était changée dans les options, les 2 variables étaient modifiées de pair via la « commande complète ». C'est celle que j'applique actuellement via événement-making pour changer de langue. (Fonctionnel à 100% en 24.48 moddé et 24.49 moddé)

```
#---- Commande complète ----
lang = "en"
value = "en"

if File.exist?("saves/settings.ini")
  settings = File.read("saves/settings.ini").sub(/LANG=.*/, 'LANG=' + lang)
  File.open("saves/settings.ini", "w") { |f| f.puts settings }
else
  File.open("saves/settings.ini", "w") { |f| f.puts "LANG=#{lang}" }
end

GameData::Text.change_lang(value)
$options&.language = value
GameData::Text.load
```

Comme la variable `$options&.language` était commune à tout le jeu, cela ne causait aucun problème : ça restait synchronisé sur la valeur du fichier INI.

Le but est donc de revenir au mécanisme de la 24.48. Après, je ne crache pas sur le fait de revenir à une unique variable, ce qui permettrait de changer la langue avec la commande simplifiée, celle qui fonctionne sur PSDK vanilla :

```
#---- Commande simplifiée ----
lang = "en"
value = "en"
$options&.language = value
GameData::Text.load
```

... sauf qu'il serait difficilement envisageable d'avoir cette valeur inscrite en dur sur un fichier INI (ou YAML) externe, au moins que PSDK puisse interpréter une commande faite sur `$options&.language` comme un ordre de lecture ou d'écriture sur l'attribut LANG du fichier INI (ou YAML).

Donc si on revient au mécanisme 24.48, il faudra que : (Résumé des objectifs)

- 1) La variable `$options&.language` soit commune à tout le jeu et non une sauvegarde spécifique.
- 2) Que la variable `$options&.language` soit synchronisée dès le démarrage du jeu sur la valeur indiquée par le fichier ini. (Comme ça, la langue sera la bonne dès l'écran-titre.) Par effet de propagation, la langue « lue » par le menu d'option sera automatiquement la bonne comme en 24.48.
- 3) Que la langue par défaut à la création d'une nouvelle partie soit celle inscrite sur le fichier INI. (Je précise, car actuellement la scène « Load » de PSDK contourne ce réglage : mes images sont en anglais avec le fichier INI en français.)
- 4) Qu'au changement de langue dans les options, PSDK change à la fois la variable `$options&.language` et celle du fichier INI.
- 5) Qu'au moment de quitter le menu d'options, la langue du menu précédent soit immédiatement actualisée.

• **Bonus** : En cas de fichier INI erroné (absence de la ligne “LANG=” ou ligne “LANG=” avec une langue invalide), la ligne LANG sera régénérée (si elle n’existe pas) ou corrigée avec la langue par défaut (si elle existait déjà). (PSDK crashe si une valeur erronée est indiquée sur le INI.)

Comme ça, si je partage aux anglais (ou à d’autres nationalités), il me suffira de leur dire d’installer un fichier de quelques octets en sus pour avoir la bonne langue au premier démarrage.

À noter qu’en cas d’absence de ce fichier INI, il est déjà auto-généré avec la langue par défaut (anglais). (C’est un bon point ça.)

Les autres options

1) Volume

Comme dit plus haut, le splitter en 2 paramètres :

- Volume musique (gère le volume des canaux BGM et BGS)
- Volume effets sonores (gère le volume des canaux ME et SE)

+ Débugger la gestion du volume musique qui ne fonctionne pas.

D’après le code du Project Identity, le volume aurait dû se régler avec un slider (réglette), mais ça ne semble pas encore être programmé sur PSDK... À voir si ça peut se faire.

Quoi qu’il en soit, j’augmenterai le pas d’incrément de 1 à 5 %. (Il n’y a pas besoin d’une précision si fine.)

Ces paramètres doivent être codés dans le settings.ini (ou settings.yml si on part sur le format yml) et seront donc commun à tout le jeu. Cela permettra au réglage sonore d’être actif dès le démarrage du jeu.

2) Style de combat



Réalisé

L’option n’était ni fonctionnelle, ni accessible au joueur dans le souhait de mon gameplay, ça me donne deux raisons pour la supprimer de la liste.

Néanmoins, la fonctionnalité de choix ou défini devra être en elle-même fonctionnelle pour les besoins de mon scénario.

- Par défaut « défini ». Pourquoi le héros aurait-il le privilège de changer de Pokémon gratuitement après un KO de l’adversaire ?
- Dans certains matchs scénarisés, ce privilège sera accordé. C’est notamment le cas des combats de champions d’arène.

La bascule sera gérée via un interrupteur RPG Maker. Je dois donc pouvoir modifier ce paramètre via une commande d’événement-making.

3) Mode de difficulté

(Avec l’étoffeement du Death Overhaul, plus de choix sont apparus.)

Tout ceci est lié à mon mod « Death Overhaul ». Trois options sont actuellement accessibles :

Paramètre	Choix possibles		
Niveau des dresseurs	Faible	Normal*	Élevé
Mort de vos Pokémon	Traditionnel	Réaliste*	Nuzlocke
Mort des Pokémon adverses	Activé*		Désactivé

* = option par défaut

Sur le long-terme, un système de « preset » sera proposé en premier dans la liste et permettra de sélectionner un réglage pré-paramétré pour chaque sous-option.

Cela sera parfait avec un menu dédié (sous-menu d'options) pour les réglages liés à la difficulté. (Comme cela est déjà le cas sur Pokémon Uranium : les réglages de difficulté sont un menu d'options à part.)

Le preset « Normal » correspond aux réglages par défaut.

Preset	Casual	Facile	Normal	Difficile	Mortel	Nuzlocke	Nuzlocke hardcore
Niveau des dresseurs	Faible		Normal	Élevé	Faible	Normal	Élevé
Mort de vos Pokémon	Traditionnel		Réaliste		Nuzlocke		
Capture unique par zone (Area Clause)	Non					Oui	
Clause Chromatique (Clause de capture unique requise)	« Oui » par défaut sur tous les presets / « Non » si décochée manuellement Grisée si clause de la capture unique non cochée.						
Clause des Doublons (Clause de capture unique requise)	« Oui » par défaut sur tous les presets / « Non » si décochée manuellement Grisée si clause de la capture unique non cochée						
Clause Black-Out (Game over si toute l'équipe morte)	Non (Grisée si la mort des Pokémon du joueur est réglée sur « Traditionnel »)						Oui
Interdiction de tous les objets en combat**	Non						Oui
Mort des Pokémon adverses	Non			Oui			
Poison à l'ancienne	Non			Oui			

** De base, seules les baies seront utilisables en combat à Keltios, mais le joueur peut s'imposer une clause d'interdiction absolue à l'instar des matchs officiels.

*** Death Overhaul inclus la Legendary Clause. Elle sera toujours active dans Sacred Phoenix pour des raisons de scénario, donc pas proposée au joueur.

Pour refléter in-game la difficulté active, un léger changement graphique se fera voir en combat à l'instar de ce qui se faisait sur Pokémon Gemme avec le petit symbole coloré « I », « II » ou « III ».

- Selon la mortalité de vos Pokémon, leur nom au-dessus de la barre des PV sera affiché respectivement en bronze clair, argent ou or. (L'ombre autour des lettres restera noire.) L'affichage en or nécessitera en plus d'avoir le l'Area Clause active (sinon, ça sera en argent).

- Selon le niveau des dresseurs, le nom de leurs Pokémon au-dessus de la barre des PV seront affichés respectivement en bronze clair, argent ou or. (L'ombre autour des lettres restera noire.)

Si avant la remise de la première Pokéball, le joueur a activé les deux règles pilier du Nuzlocke (mortalité réglée sur « Nuzlocke » + Capture unique par zone actif), un tag « Nuzlocke actif » sera appliqué à sa partie. (Fenêtre de sauvegarde affichée dans une autre couleur + nom de ses Pokés écrit en or en combat.)

S'il désactive l'une de ces options – même temporairement – le tag sautera définitivement et tous les privilèges graphiques associés seront perdus à jamais.

Le joueur sera bien entendu prévenu de ce risque et aura la possibilité de ne pas valider ces changements si jamais il a désactivé l'un de ces deux réglages.

À noter que le joueur pourra librement changer ses paramètres de difficulté sans pénalité ou restriction jusqu'à la remise des premières Pokéballs (moment fatidique où le Nuzlocke commence réellement dans tout jeu Pokémon.)

Fonctionnement du réglage de preset.

- Cela se présentera sous la forme d'une réglette avec 13 crans.
- 7 crans pour les paliers principaux
- 6 crans « intermédiaires » si jamais le joueur modifie des paramètres. Le cran sera alors calculé sur le premier preset qui remplira tous les critères de difficulté (exclusion faite de la Mort des Pokémon adverses et du Poison à l'ancienne qui ne seront testés qu'à partir du preset « Mortel ») ; et le terme « (personnalisé) » sera ajouté sur le nom du preset.

=> Par exemple, la mention affichée sera « **Normal (personnalisé)** » pour des réglages qui respectent tous les paramètres de difficulté du niveau normal, mais ne répondant aux pas exigences du preset supérieur.

- Revenir sur la réglette du preset et la bouger remettra tous les réglages sur le preset en question.

Le preset « Mortel » est présent pour des semi-Nuzlockes ou Nuzlockes aux règles personnalisées style Chainlocke ou Egglocke. Death overhaul permet en effet de séparer la règle de la mortalité si KO de celle de la capture unique par zone.

Une fois le prologue terminé, le jeu ne permettra plus au joueur de réactiver l'Area Clause si jamais il la désactive (ou qu'elle n'était pas activée) (et donc d'avoir le nom de ses Pokémon affichés en doré en combat = vrai Nuzlocke actif). La cause de mortalité restera néanmoins changeable à la hausse.