Coloration de jeux Game Boy

Introduction

Heig-boy permet d'émuler des jeux prévus pour la Game Boy originale. Il inclut également une fonctionnalité permettant de jouer à ces jeux avec des couleurs définies par l'utilisateur, en lieu et place de l'écran noir et blanc traditionnellement inclus sur la console.

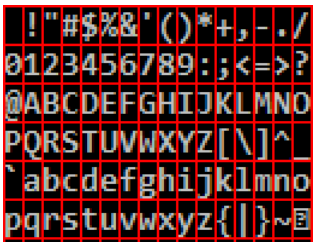
Configuration requise

Pour colorier un jeu avec ce guide, il vous faudra un ordinateur doté de la configuration suivante :

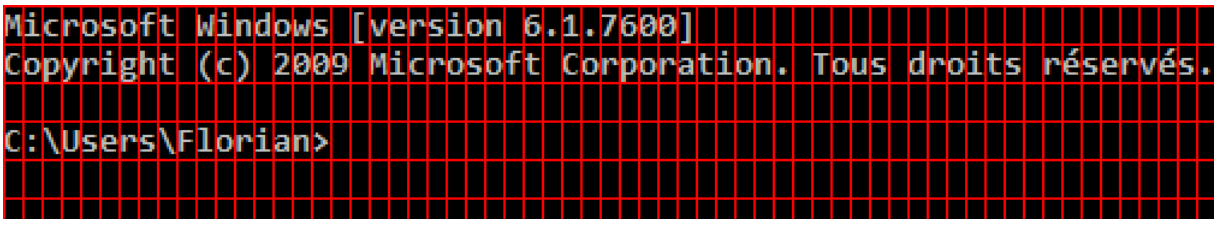
- Microsoft Windows XP, Vista ou 7
- Processeur Intel 32 bits à 300 MHz
- 256 Mo de mémoire RAM
- Microsoft .NET Framework 2.0 ou ultérieur
- Eventuellement Microsoft Visual C++ redistributable package

Concept

La technologie, nommée ColorIt, se base sur le principe même de l'affichage de la Game Boy (nous vous renvoyons au rapport pour plus d'informations). L'affichage est réalisé sous la forme d'une tilemap, c'est-à-dire composée d'une matrice de motifs carrés dont le contenu est défini par l'application. On distingue deux éléments : le tileset et la map. Un tileset (composé d'images de 8x8 pixels) représente une liste de motifs possibles et une map (composée d'index de motifs) représente une grille indiquant où utiliser quel motif sur l'écran.



Exemple de tileset, parfois également noté charset (jeu de caractères)



Exemple de map

Le jeu copie en mémoire vidéo (VRAM) la liste de tous les motifs utilisables à un moment donné. Ces motifs, au nombre de 384, ont chacun un numéro (de 0 à 383). A titre d'exemple, le motif n° 73 pourrait désigner une partie de maison. Avec ColorIt, vous allez définir que le motif n°73 doit être

colorié avec des nuances de brun. Le coloriage consiste à faire correspondre une couleur à chacune des quatre nuances de gris supportés par la console.

Exemple de motif (8x8 pixels) utilisant des nuances de gris

Le même motif colorié avec une palette (à droite)

Comme les motifs présents en mémoire vidéo peuvent changer avec le temps (par exemple dans un niveau montagneux, les motifs pourraient représenter des rochers, alors que dans un niveau marin ceux-ci pourraient être remplacés par des algues). Ainsi on doit définir la notion de scène. Définissons la scène comme dans un film, comme étant un écran particulier, délimité d'un autre par une transition par le noir (ou le blanc dans le cas de la Game Boy). Des exemples de scène sont l'écran titre, le générique, le niveau 1, le niveau 2, etc. Ainsi un script de coloration a la forme suivante :

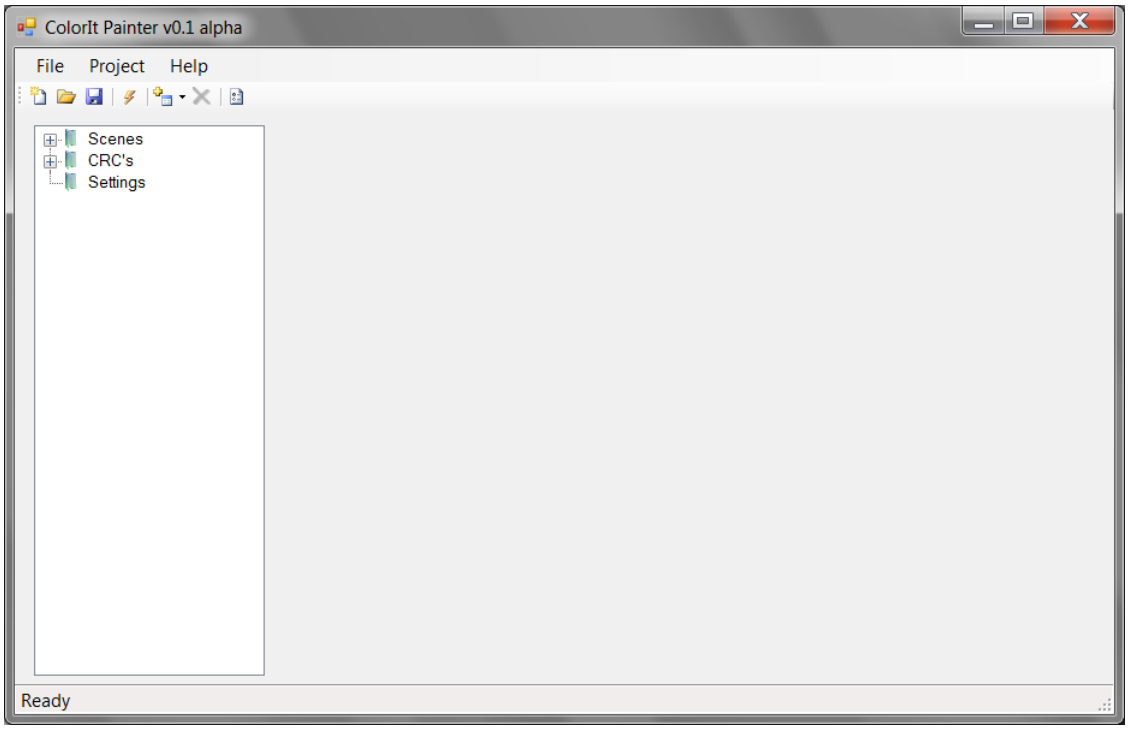
```
Init:
    AddTileCrc <liste des tiles permettant d'identifier la scène>

[default]:
    <Définition de la scène par défaut>
    End

[CRC de la scene]:
    SetPalette <n° de palette>, <couleurs>
    …
    AddTileRule <n° de motif>, <n° de palette à associer>
    …
    End

[CRC de la scène suivante]:
    …
```

Marche à suivre

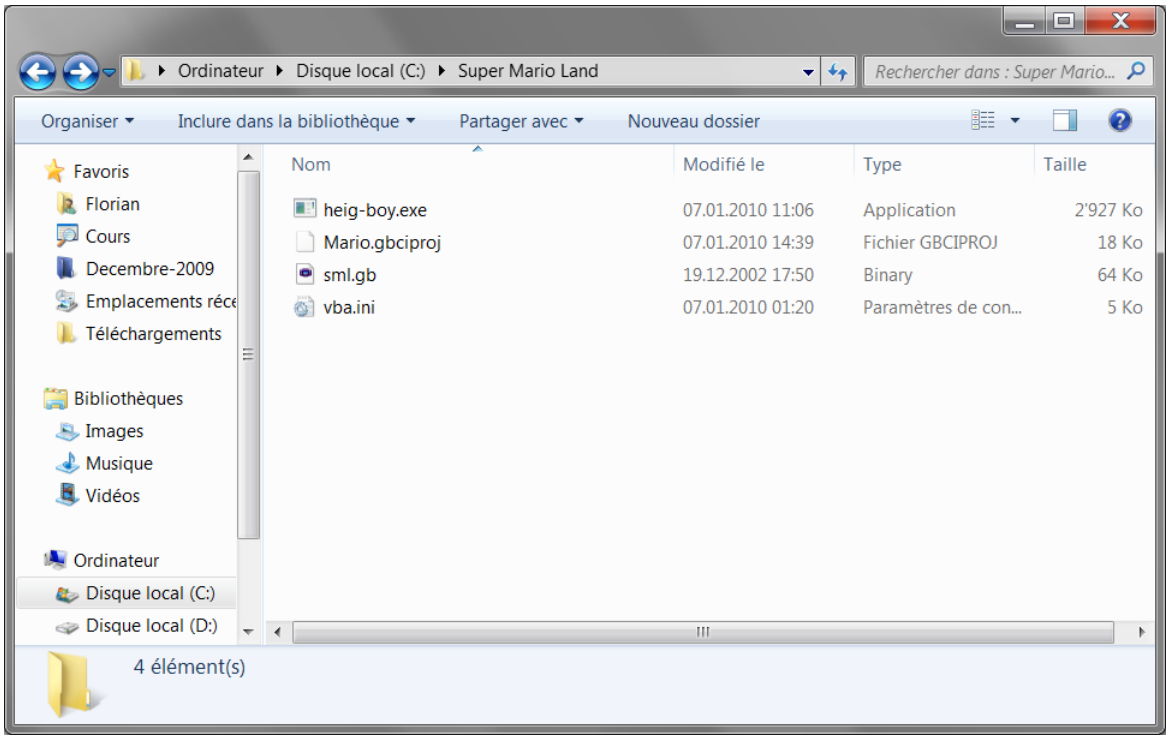
Nous présentons ici un petit didacticiel (marche à suivre assistée) qui visera à colorier une partie du jeu Super Mario Land. Ce jeu a été choisi car il est relativement simple et contient néanmoins tous les éléments particuliers (plusieurs scènes, défilement, etc.).

La première des choses à faire pour colorier un jeu est d'y jouer, ce qui en ravira plus d'un. Pour colorier une scène il est nécessaire d'y parvenir et de prendre une capture à cet endroit. Dans HEIG-Boy une capture (sauvegarde d'état) peut se faire via un raccourci clavier (F5 au moment de la rédaction de ce document). Les captures contiennent toutes les informations permettant de reprendre le jeu à l'état où la capture a été prise, et elles contiennent également des informations utiles pour colorier graphiquement un jeu (comme par exemple l'état de la mémoire vidéo).

Pour commencer, ouvrez le logiciel ColorIt Painter et créez un nouveau projet (un nouveau projet est automatiquement présenté lorsque vous ouvrez le logiciel, sinon vous pouvez cliquer sur la feuille blanche dans la barre d'outils).



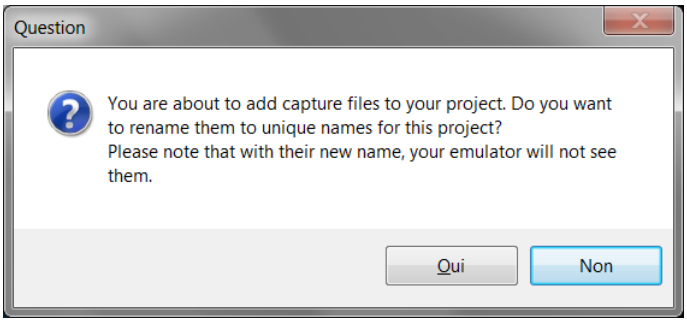
Il est conseillé d'enregistrer le projet dans le répertoire contenant l'image du jeu. Cela rendra les manipulations suivantes plus aisées. Cliquez pour cela sur la disquette bleue dans la barre d'outils. Dans notre cas le projet sera sauvegardé dans un dossier Super Mario Land avec le nom Mario. Nous y mettons le projet, le fichier image du jeu et un émulateur supportant la coloration (HEIG-Boy et VisualBoyAdvance 1.7.2 sont supportés actuellement).



Nous allons maintenant laisser de côté l'application Painter et lancer le jeu (sml.gb) avec heig-boy. Lorsqu'une scène que l'on souhaite colorier apparaît, appuyons sur F5 pour effectuer une sauvegarde d'état. Un message confirmant le succès de la manipulation devrait apparaître dans la barre d'état :

Vous devriez également apercevoir un nouveau fichier nommé <nom-du-jeu.stX> avec X un nombre de 0 à 9. Ce fichier contient les données représentant l'état de la Game Boy virtuelle au moment de l'appui sur la touche de sauvegarde.

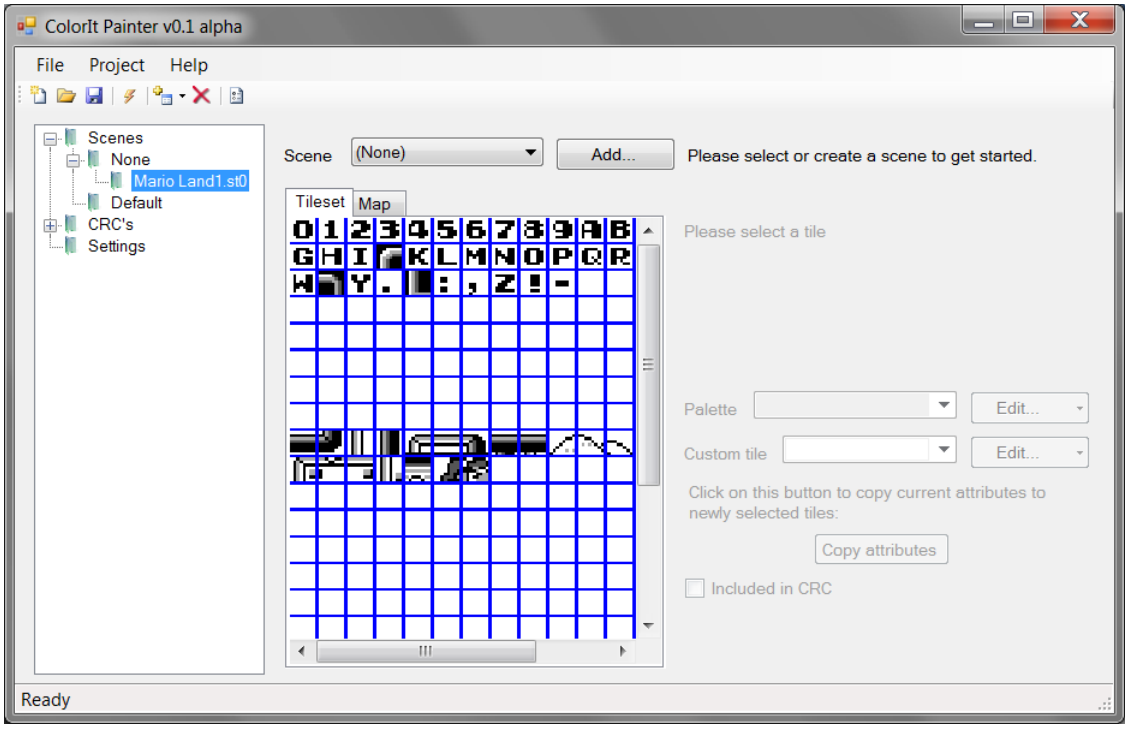
Vous pouvez maintenant retourner dans le logiciel Painter, où nous pourrons colorier cette scène. Cliquez pour cela sur Project, Add Capture dans le menu. Sélectionnez le fichier sml.st0. Un message vous invitera à renommer le fichier selon le nom du jeu. Cela n'est pas une mauvaise idée, car nous allons éditer plusieurs scènes et donc avoir besoin de plusieurs fichiers d'état différents. Si vous ne le renommez pas, l'émulateur écrasera sans cesse la sauvegarde précédente.[1]



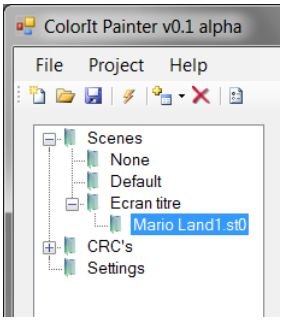
Si vous sélectionnez oui, il vous sera demandé le nom du jeu. Voyons cela maintenant car vous aurez à le donner au logiciel tôt ou tard. Il suffit dans cette boîte de dialogue de cliquer sur « Browse… » et sélectionner le fichier image du jeu (sml.gb dans notre exemple). Le logiciel extraira automatiquement le nom officiel de la cartouche à partir de celle-ci.

[1] Actuellement HEIG-Boy ne permet de prendre qu'une seule sauvegarde par jeu. Il est possible que dans sa version finale un système de « slot » soit implémenté. Il sera alors plus judicieux de prendre plusieurs sauvegardes pour chaque scène et toutes les importer d'un coup dans Painter.

Le fichier sera ensuite renommé et ajouté au projet, dans une scène par défaut nommée « None » et signifiant que le fichier n'est associé à aucune scène. Vous pouvez trouver le fichier en développant l'arbre du projet sur la gauche de la boîte de dialogue :

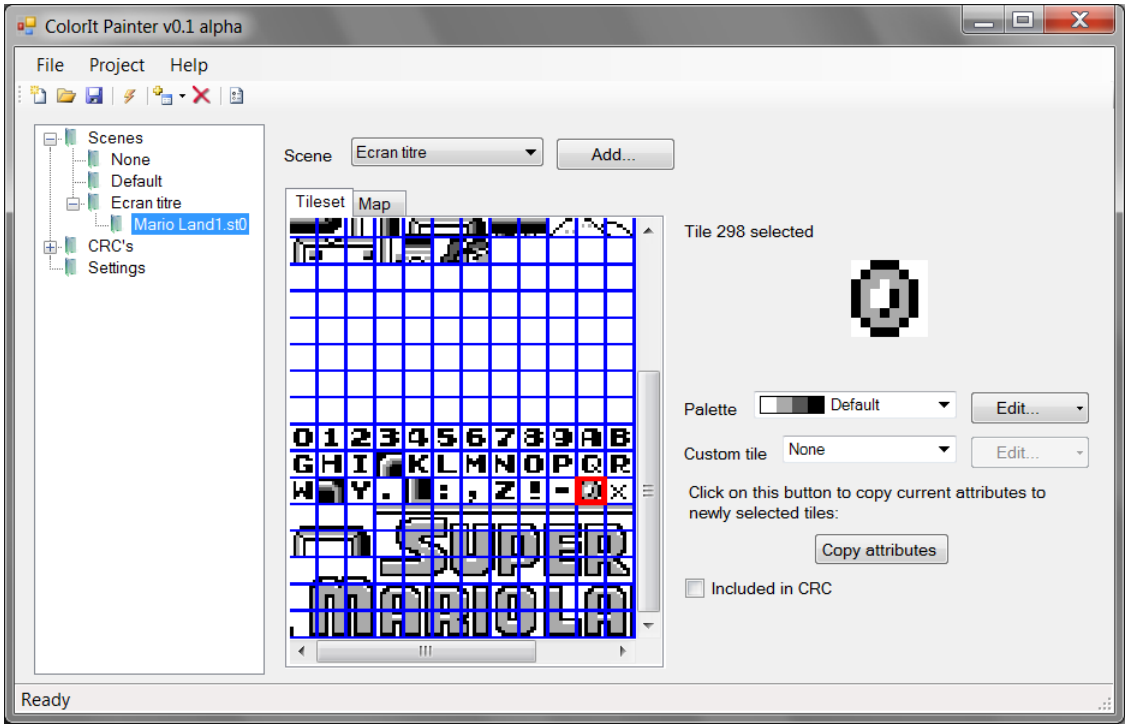


Le fait de sélectionner la capture ajoutée affichera son contenu sur la droite. Notons que pour l'instant l'édition n'est pas encore possible. Pour cela il faut tout d'abord l'affecter à une scène. Nous allons donc en créer une. Cliquez pour cela sur le bouton « Add… » et donnez-lui n'importe quel nom, comme par exemple « Ecran titre ». Les autres options seront laissées à leur valeur par défaut. La capture sera alors automatiquement affectée à cette nouvelle scène. Vous pouvez voir la modification dans l'arbre du projet :

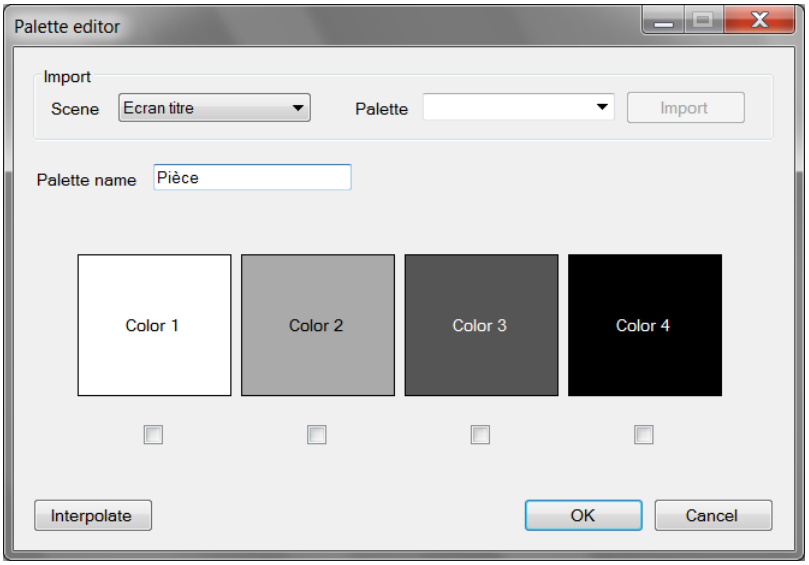


L'élément apparaissant à droite, sous l'onglet « Tileset » est appelé… le tileset. Il représente tous les motifs copiés en mémoire vidéo. Il y en a 384, et il est possible pour le jeu d'afficher n'importe lequel de ces motifs sur la map, mais uniquement ceux-ci. ColorIt base son fonctionnement sur cette propriété : si l'on colorie chacun de ces motifs, toute la scène sera automatiquement coloriée.

Vous pouvez sélectionner un motif en cliquant sur celui-ci. Vous aurez alors des informations supplémentaires sur le panneau de droite, comme une version agrandie du motif :



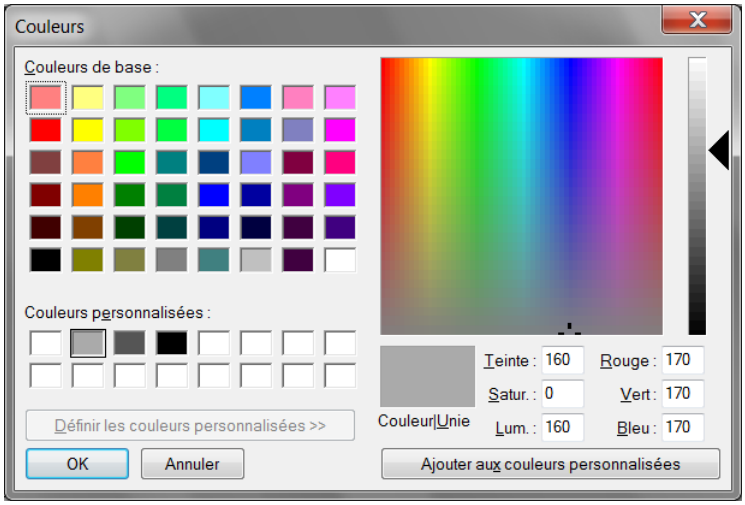
A partir de là il est possible d'attribuer une palette à ce motif. Cliquez pour cela sur la liste déroulante « Palette » et sélectionnez « Add… ». Une boîte de dialogue s'ouvre alors :



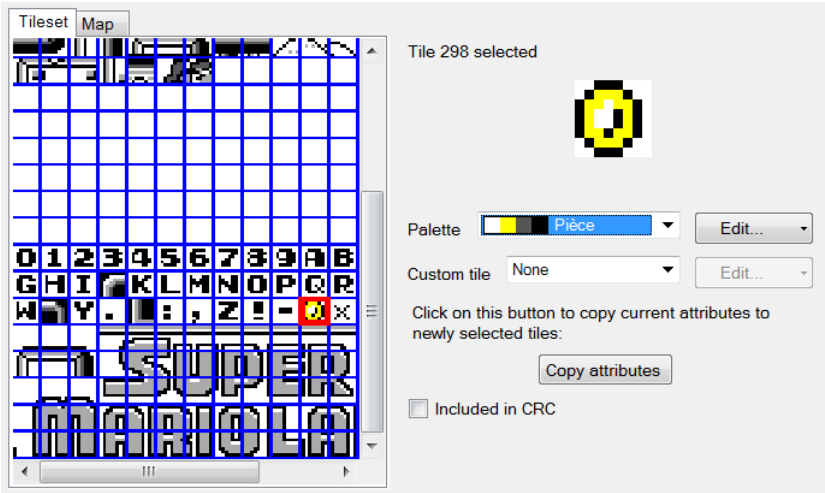
Pour commencer, n'hésitez pas à donner un nom à la palette. Une palette peut être affectée à plusieurs objets différents ; par exemple une palette composée de nuances de bleu pourrait être

utilisée à la fois pour l'eau et le ciel. Donnez-y un nom générique en fonction de l'usage que vous pensez en avoir.

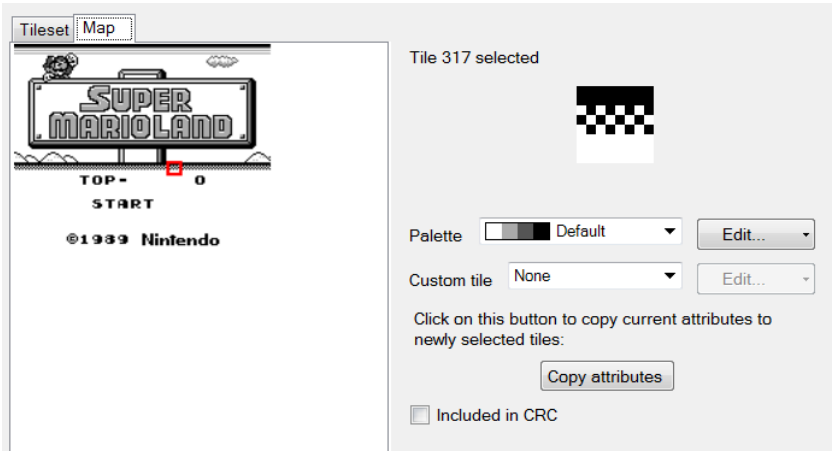
Plus bas vous pouvez apercevoir les couleurs. La correspondance avec les nuances de gris de l'image de la Game Boy est la suivante, en partant de la couleur n°1 à la couleur n°4 : Blanc, gris clair, gris foncé, noir. Il suffit de cliquer sur l'une d'elles pour la modifier. La boîte de dialogue standard de Windows apparaît alors, avec les couleurs courantes disponibles en bas à gauche :



Dans le cas de la pièce, nous allons remplacer le gris clair par du jaune. En validant la boîte de dialogue ainsi que la précédente, on se retrouve sur l'écran d'édition du tileset avec une pièce en couleur :

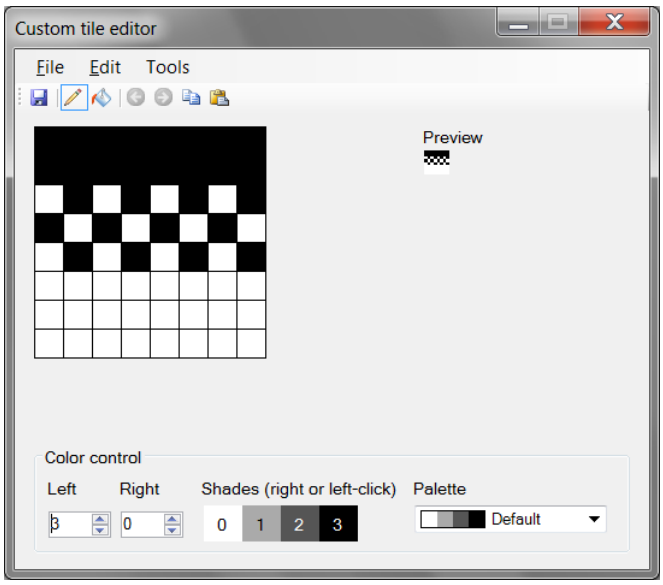


Voyons maintenant l'onglet « Map » qui présente l'état de la map de fond au moment où la sauvegarde a été prise. Vous devriez y voir apparaître le contenu de l'écran sans les objets et la fenêtre. Là aussi il est possible de sélectionner des tiles en cliquant sur la map :

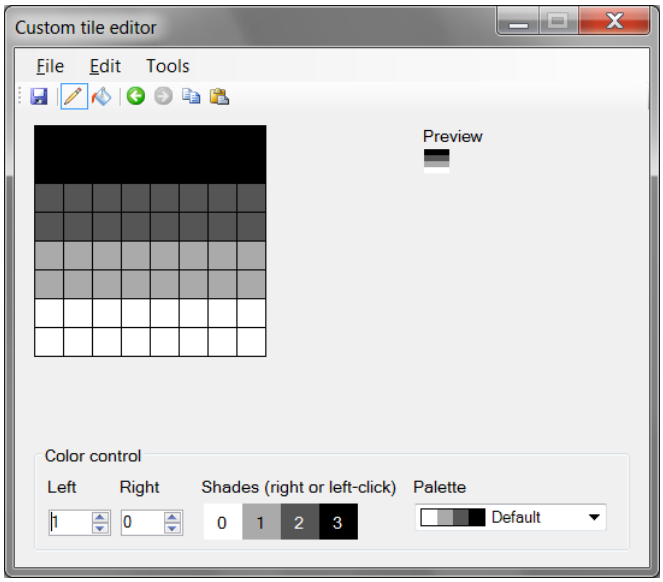


Il s'agit d'un autre moyen plus pratique de colorier le jeu. Toutefois seul l'écran actif peut être présenté ici. Si le jeu utilise un défilement, vous ne pourrez pas avoir la suite de l'écran, et vous devrez alors prendre plusieurs captures afin de couvrir le niveau entier. C'est pourquoi il est généralement plus judicieux de colorier directement le tileset.

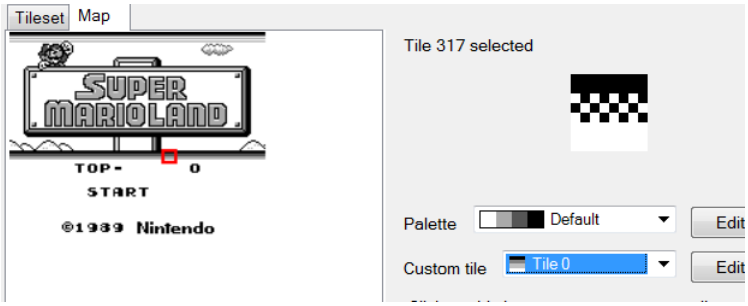
Dans certains cas il peut également être intéressant de retoucher légèrement les motifs. Par exemple le motif ci-dessus serait plus joli une fois colorié s'il utilisait un dégradé de couleurs. ColorIt propose pour cela les « Custom tiles ». Cliquez sur la liste déroulante « Custom tile » et faites « Add… ».



Apparaît alors une nouvelle boîte de dialogue, avec des outils basiques permettant d'éditer l'image du motif. Nous allons utiliser le pinceau et cliquer sur les tons de gris pour modifier le motif de façon à arriver à un dégradé comme ceci :



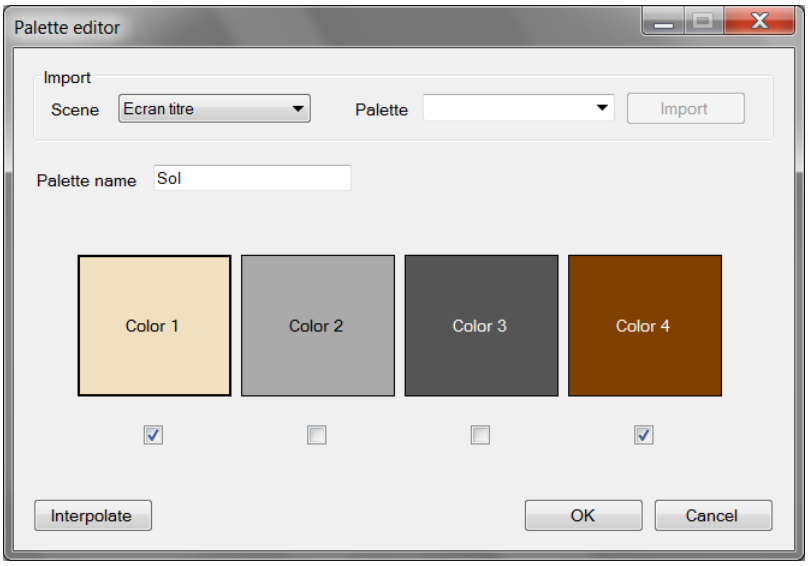
Il est possible de sélectionner une palette dans la liste déroulante au fond à droite pour obtenir un aperçu du résultat en utilisant une palette donnée. Une fois le dessin terminé, cliquez sur la disquette pour fermer l'éditeur.



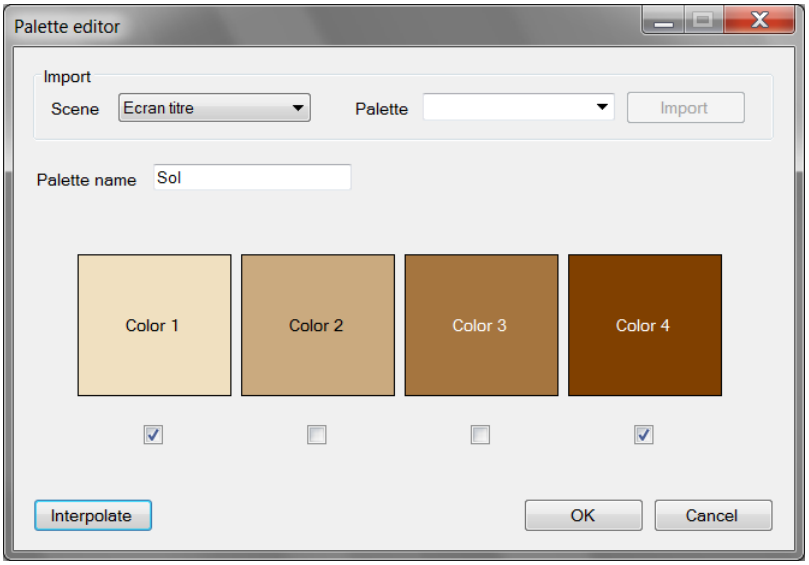
Il est un peu difficile de voir cela dans la petite copie d'écran présentée ci-dessus, mais à présent tout ce qui utilisait le motif n°317 que nous avons édité est remplacé par un motif en dégradé. Comme ce motif était répété sur toute la longueur de l'écran, on voit ainsi apparaître une ligne horizontale en dégradé.

Ainsi si un même motif est utilisé plusieurs fois, la modification est visible partout. Gardez cela en tête car cela peut poser des problèmes si un même motif est utilisé dans plusieurs contextes (par exemple un motif gris uni pour les nuages et la terre). Il n'existe pas moyen de résoudre ce problème (à moins de modifier le code du jeu, aussi appelé *romhacking*). Dans ces cas, il faudra choisir des couleurs aussi neutres que possibles pour colorier les deux éléments sans que cela choque le joueur.

On va maintenant colorier ce dégradé en lui appliquant une palette. Cliquez sur la liste déroulante palette, et choisissez deux couleurs aux extrémités pour la nouvelle palette :



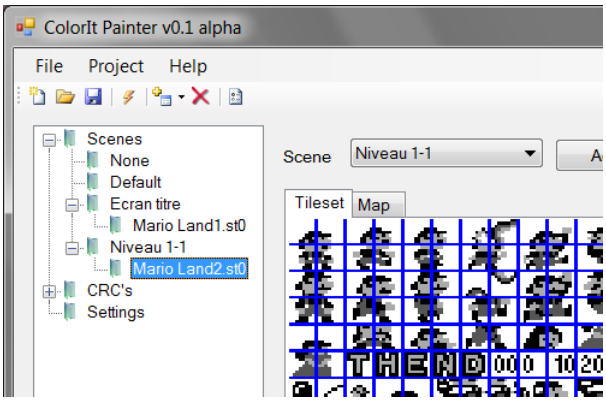
Vous aurez peut être remarqué les intrigantes cases à cocher présentes en-dessous des couleurs. Celles-ci vont de pair avec le bouton « Interpolate » en bas à gauche de la boîte. Lorsque vous cliquez sur ce bouton, toutes les couleurs non cochées sont déduites à partir des couleurs cochées. Dans notre cas les deux couleurs aux extrémités sont cochées automatiquement car on les a définies et on ne souhaite donc logiquement pas les perdre. Maintenant si on clique sur le bouton « Interpolate » les couleurs au milieu seront automatiquement déduites des extrémités, créant un joli dégradé :



Notons que n'importe quelle combinaison de couleurs peut être cochée, quoi qu'il en soit les couleurs non-cochées seront déduites à partir des couleurs voisines. En cliquant sur le bouton OK, cette nouvelle palette va remplacer les niveaux de gris utilisés précédemment.

Satisfaits ? Sinon il est toujours possible d'éditer à nouveau la palette en cliquant sur le bouton « Edit… » à côté de la liste déroulante. Il en va de même pour les « Custom tiles ».

A présent, nous allons ajouter un deuxième état de sauvegarde pour colorier une deuxième scène : le niveau 1-1 du jeu. Cela se réalise de façon identique au titre, excepté que la sauvegarde sera prise à n'importe quel moment durant le jeu (mais toujours dans le niveau 1-1). Ajoutez la capture au projet et créez une nouvelle scène. L'arborescence devrait ressembler à cela :

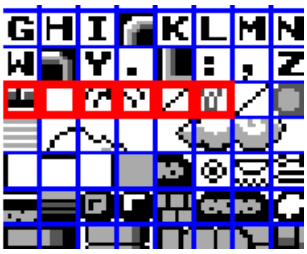


La coloration peut se faire de façon similaire à la scène précédente. Une fois ceci terminé, il reste encore une étape avant de pouvoir jouer à votre jeu favori en couleur. En effet, nous parlons de scène, mais comme ce concept est abstrait pour la Game Boy elle même il faut définir un moyen permettant d'identifier la scène courante, afin que l'émulateur sache quelles règles appliquer pour les motifs.

Avec ColorIt, la scène est identifiée par un CRC partiel de la VRAM. En effet, comme chaque scène a un contenu particulier (par exemple un niveau se déroulant dans la ville aura des motifs de bâtiment, un niveau se déroulant dans l'espace aura des motifs représentant des étoiles). Le but ici est de déterminer quels n° de motifs permettent d'identifier une scène à coup sûr.

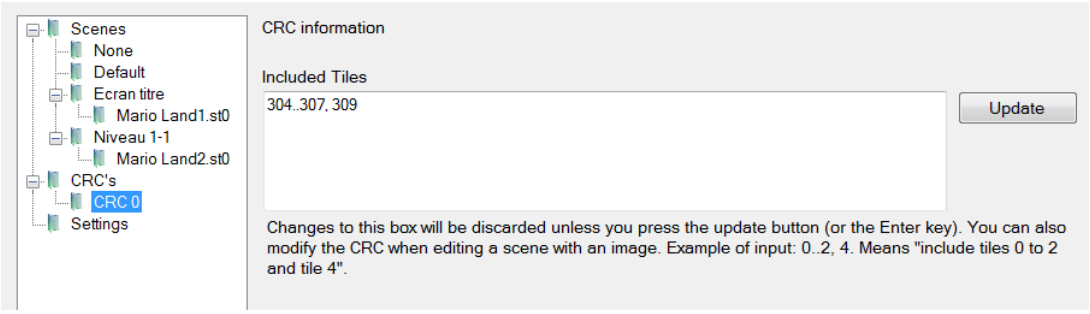
En d'autres termes, il faut choisir des motifs qui changent à chaque scène ET qui ne changent pas au milieu de la scène. Dans l'exemple plus haut, inclure des motifs du personnage principal, Mario, n'est certainement pas une bonne idée, puisque celui-ci va se retrouver en VRAM dans tous les niveaux. En revanche si vous prenez des motifs spécifiques au niveau en cours, vous êtes presque sûr de ne pas vous tromper.

Il reste juste à vérifier que les motifs ne sont pas modifiés en cours de scène, car dans ce cas une scène pourrait produire plusieurs CRC en fonction de l'état des motifs en question, ce qui serait difficile à maintenir. Généralement ce n'est pas un problème car l'accès à la VRAM est très limité lorsque l'écran est allumé. A titre d'ordre de grandeur, les jeux peuvent rarement se permettre de modifier plus de 6 motifs sur 384. D'expérience, évitez de choisir des motifs qui ressemblent à une flamme de bougie animée par exemple. Choisissez autant que possible des éléments statiques (non animés) spécifiques à chaque scène. Dans le cas de Super Mario Land, les motifs suivants (entourés d'un carré rouge) n'ont à priori pas l'air d'être un mauvais choix :



On pourra toujours modifier cela plus tard. Le logiciel peut nous avertir si le CRC ne permet pas d'identifier de manière unique certaines scènes, en se basant sur les captures. Si vous avez plusieurs captures pour une même scène, il peut également détecter qu'un motif inclus au CRC change dans les différentes captures, indiquant qu'il ne faudrait pas l'inclure.

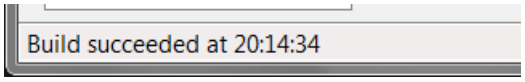
Dans le logiciel, pour inclure des motifs au CRC il suffit de les sélectionner comme dans la capture plus haut et cliquer sur la case à cocher « Included in CRC ». Vous pouvez obtenir de façon textuelle la liste des motifs inclus en sélectionnant « CRCs » puis « CRC 0 » dans l'arbre du projet.



Il est difficile d'établir un ordre de grandeur du nombre de motifs à inclure au CRC, mais il n'est généralement pas nécessaire qu'ils soient nombreux (8 par exemple). Plus ce nombre est grand, plus les chances d'avoir des problèmes de motifs animés augmentent. Au contraire, plus ce nombre est petit plus il y a de chances que deux scènes différentes produisent le même CRC, empêchant de ce fait leur détection. Le deuxième cas est préférable car Painter peut le détecter facilement et de façon fiable.

Une fois que vous avez déterminé les motifs à inclure (c'est un peu ennuyeux au début, mais pour tester vous pouvez juste sélectionner quelques motifs au hasard, vous vous apercevrez lorsque le nombre de scènes devient grand si cela cause un problème) il est possible de construire le projet.

Vous pouvez cliquer pour cela sur le petit éclair dans la barre d'outils. Si quelque chose n'a pas fonctionné, un message d'erreur apparaîtra avec suffisamment d'indications pour résoudre le problème. Sinon, la barre d'état indiquera discrètement le succès de l'opération.



Un nouveau fichier devrait apparaître dans le répertoire, d'extension .pal.ini :

Nom	Modifié le	Type	Taille
heig-boy.exe	07.01.2010 11:06	Application	2'927 Ko
Mario Land.pal.ini	07.01.2010 20:14	Paramètres de con...	1 Ko
Mario Land1.st0	07.01.2010 16:22	Fichier ST0	17 Ko
Mario Land2.st0	07.01.2010 19:57	Fichier ST0	17 Ko
Mario.gbciproj	07.01.2010 19:57	Fichier GBCIPROJ	36 Ko
sml.gb	19.12.2002 17:50	Binary	64 Ko
sml.sav	07.01.2010 19:57	Fichier SAV	0 Ko
vba.ini	07.01.2010 01:20	Paramètres de con...	5 Ko

Il s'agit du script qui pourra être automatiquement lu et interprété par l'émulateur pour colorier le jeu. Si le script est dans le même répertoire que le jeu (ce qui est le cas ici) il sera chargé automatiquement. Notez qu'il ne porte pas le même nom que le jeu : il porte le nom officiel donné à la cartouche par Nintendo. Cela permet donc à l'émulateur d'appliquer le script à un jeu donné de façon transparente, quel que soit le nom de fichier.

En effet, si on démarre HEIG-Boy et qu'on charge sml.gb, on obtient ceci :

Ce n'est que la première étape. Vous pouvez maintenant colorier les autres éléments en utilisant les concepts vus dans ce document et surtout selon vos envies.

A vos palettes !

Index du script

Le format des scripts ColorIt est brièvement décrit ici. Le langage ne porte pas l'extension .ini par hasard. En fait ce langage était initialement fait pour définir une collection de propriétés et était utilisé pour sauvegarder la configuration d'une application dans un format modifiable par l'utilisateur. Il existe donc 3 types principaux d'instructions :

1) Affectations. La forme est la suivante :
   Namespace.Propriété = Valeur
2) Appels :
   Namespace.Fonction paramètre1, paramètre2, …
3) Instructions de contrôle

Le langage est rudimentaire. L'utilisateur ne peut pas définir de variable, de ce fait la plupart des éléments de la programmation impératives sont impossibles. Notons que le langage n'est pas sensible à la casse. En cas d'erreur, un message est affiché à l'utilisateur et l'exécution du script est stoppée immédiatement.

A propos du langage : les spécifications de ce système de coloration (ColorIt), ont été définies par Florian Brönnimann en 2007 et sont actuellement implémentées dans quelques émulateurs.

Eléments syntaxiques de base

Label

Syntaxe :
label:

Description : permet de définir un label, accessible par une instruction Goto. Il est également possible de les utiliser comme des fonctions via l'instruction Call.

Section

Syntaxe :
[*section*]:

Description : permet de définir une section. Les sections sont appelées automatiquement par le système lorsqu'un événement défini est déclenché. Dans le cas de ColorIt, les sections portent pour nom le CRC de la scène que le code les suivant définit. Lorsque le système détecte une nouvelle scène, il calcule le CRC et appelle la section correspondante. La section [default] est appelée lorsqu'aucune section définie dans le fichier ne correspond à la scène courante.

Section Init

Syntaxe :
Init:

Description : cette section un peu spéciale n'a pas besoin des crochets. Elle est appelée au démarrage du script. C'est ici que vous devriez définir tous les paramètres par défaut, et en particulier les tiles à inclure au CRC. En effet, si le CRC n'est pas défini, aucune autre scène ne pourra jamais être appelée puisqu'aucune ne correspondra à aucun CRC.

Instruction Goto

Syntaxe :
`Goto label`

Description : permet de continuer l'exécution à un label défini ailleurs dans le fichier. Une erreur sera déclenchée si le label n'existe pas.

Instruction Call

Syntaxe :
`Call label`

Description : comme l'instruction Goto, branche l'exécution du script à un label défini ailleurs dans le fichier. L'état est sauvegardé et l'exécution sera poursuivie juste après cette instruction lorsque le script rencontrera une instruction End.

Important : Cette fonction n'est supportée que par la version actuelle du langage. Il est possible que d'autres émulateurs implémentant la version précédente ne le supportent pas. Il faudra donc éviter de l'utiliser.

Instruction End

Syntaxe :
`End`

Description : instruction équivalente à Return dans les langages évolués. Retourne d'une fonction appelée à l'aide de l'instruction Call. Si aucune fonction n'a été appelée (en d'autres termes si la pile d'appels est vide) le script est terminée. Toute section doit être terminée par un End ou l'exécution du script continuera jusqu'à la fin du fichier, ce qui n'est pas souhaitable.

Type nombre

Syntaxe :
```
1234 pour un nombre décimal
0xabcd pour un nombre hexadécimal
0b1010 pour un nombre binaire
rgb(r, g, b) pour une couleur (avec r, g, b entiers)
```

Description : Les nombres et les couleurs utilisent le même type. Les couleurs sont définies par un nombre 24 bits, c'est-à-dire soit en décimal (0 à 16777215), en hexadécimal (0x000000 à 0xffffff) ou en RGB, en mettant rgb(255, 128, 0) par exemple. Voir plus haut pour des explications additionnelles.

Type chaine

Syntaxe :
`"texte"`

Description : les chaînes permettent de stocker du texte. Notez qu'au contraire des autres éléments syntaxique, le texte à l'intérieur d'une chaîne conserve la casse !

Commande MsgBox

Utilisation :
`MsgBox "Texte"`

Description : affiche une boîte de dialogue. Utile pour le débogage.

Eléments du langage ColorIt

Ces éléments devraient être précédés du préfixe ColorIt et suivies d'un point. Il est possible avec HEIG-Boy de ne pas utiliser ce préfixe, mais il n'est pas garanti que d'autres émulateurs supportant la coloration ne puissent s'en passer. C'est pourquoi il est conseillé de toujours ajouté le préfixe, même si vous ne le verrez pas dans les pages suivantes.

Commande SetPalette

Utilisation :
`ColorIt.SetPalette palNo, palColor0, palColor1, palColor2, palColor3`

Description : cette commande définit une palette. *PalNo* est le numéro de palette (entre 0 et 255), *palColor0*, *palColor1*, … sont les 4 couleurs de la palette que vous voulez définir. Avec cette commande, vous pouvez remplacer une palette existante (p. ex. la palette n° 0 existait déjà mais vous voulez lui mettre des couleurs différentes). Tous les arguments sont des nombres, cf. le chapitre précédent pour plus d'information sur les couleurs.

Commande AddTileRule

Utilisation :
`ColorIt.AddTileRule tileStart, tileEnd, palNo`

Description : cette commande permet de définir une règle pour un motif. En d'autres termes cela va associer une palette (*palNo*) aux motifs entre *tileStart* et *tileEnd*, bornes incluses.

Commande AddTileCRC

Utilisation :
`ColorIt.AddTileCRC tileStart, tileEnd`

Description : inclut les motifs entre *tileStart* et *tileEnd* compris au CRC. Les motifs inclus au CRC sont utilisés pour calculer le CRC. Par défaut aucun motif n'est inclus, il est donc nécessaire d'appeler cette fonction au moins une fois dans la section *Init*.

Commande SetTileData

Utilisation :
`ColorIt.SetTileData customTileNo, "hextiledata"`
`ColorIt.SetTileData customTileNo, L"humantiledata"`

Description : cette fonction crée un motif personnel et en définit les données de l'image, avec *customTileNo* le n° de motif personnel (entre 0 et 383).

Dans la première syntaxe, les données du motif (mises entre guillemets) sont exprimées en **hexadécimal**. Le format d'image de la Game Boy est le suivant :

- 1 octet représentant 8 pixels (une ligne) du plane 1
- 1 octet représentant 8 pixels du plane 2
- 1 octet représentant 8 prochains pixels du plane 1
- etc.

Un octet représente 8 pixels car dans un octet il y a 8 bits, chacun représentant si le pixel est allumé ou non. Voici les résultats en fonction de si les bits sont allumés ou éteints :

- Plane 1 éteint, plane 2 éteint : Gris clair
- Plane 1 allumé, plane 2 éteint : Blanc
- Plane 1 éteint, plane 2 allumé : Gris foncé
- Plane 1 allumé, plane 2 allumé : Noir

Dans la deuxième syntaxe, les données du motif sont définies pixel par pixel, de gauche à droite, par ligne de 8 pixels de haut en bas. Vous pouvez séparer les lignes par un espace pour une meilleure lisibilité. Chaque pixel est défini par un nombre entre 0 et 3, 0 représentant le blanc, 1 le gris clair, 2 le gris foncé et 3 le noir.

Commande SetTile

Utilisation :
`ColorIt.SetTile` *`tileNo, customTileNo`*
`ColorIt.SetTile` *`tileStart, tileEnd, customTileNo`*

Description : affecte un motif personnel à un ou plusieurs motifs. Le motif personnel doit avoir été créé avec ColorIt.SetTileData.

Note : si vous associez un motif personnel à un groupe de motifs, le n° est incrémenté à chaque motif. Par exemple si vous définissez le motif personnel n° 0 pour les motifs 128 à 131, le motif 128 utilisera le motif personnel n° 0, le motif 129 le motif personnel n° 1, et ainsi de suite.

Commande AutoShowVramCRC

Utilisation :
`ColorIt.AutoShowVramCRC = True | False`

Description : permet d'activer ou désactiver l'affichage automatique du CRC dans la fenêtre de l'émulateur lorsque le CRC change. Cela permet d'identifier des scènes avant de les ajouter au script. Cette fonctionnalité est utile au développement seulement et n'est pas supportée par HEIG-Boy, car l'application ne dispose pas de journal ou élément fonctionnel similaire (le fait de modifier cette propriété n'aura aucun effet mais ne provoquera pas d'erreur).