Comment colorer un jeu Game Boy sur sa PSP avec MasterBoy.

Prérequis

Tout d'abord, il est probablement inutile de préciser qu'avoir une PSP est indispensable, de même qu'un ordinateur (un PC sous Windows est conseillé si vous avez peu de connaissances techniques, mais ce n'est vraiment pas un problème de le faire sur les autres systèmes également).

Si vous lisez ce document, vous avez probablement téléchargé MasterBoy version 2.0 (ou une version plus récente) ; mais si ce n'est pas le cas, vous pourrez le trouver sur http://brunni.dev-fr.org/.

Pour commencer, il est rassurant de savoir qu'aucune notion de programmation n'est nécessaire. Même si nous allons utiliser un script pour définir les palettes et les associer aux éléments du décor, la procédure est toujours la même et vous n'aurez qu'à suivre les instructions de ce manuel en les adaptant au jeu que vous voulez colorer.

Ne vous en faites pas s'il y a beaucoup de texte, il y a presque toujours une image pour illustrer et vous n'êtes pas obligé de tout connaître.

Pour commencer

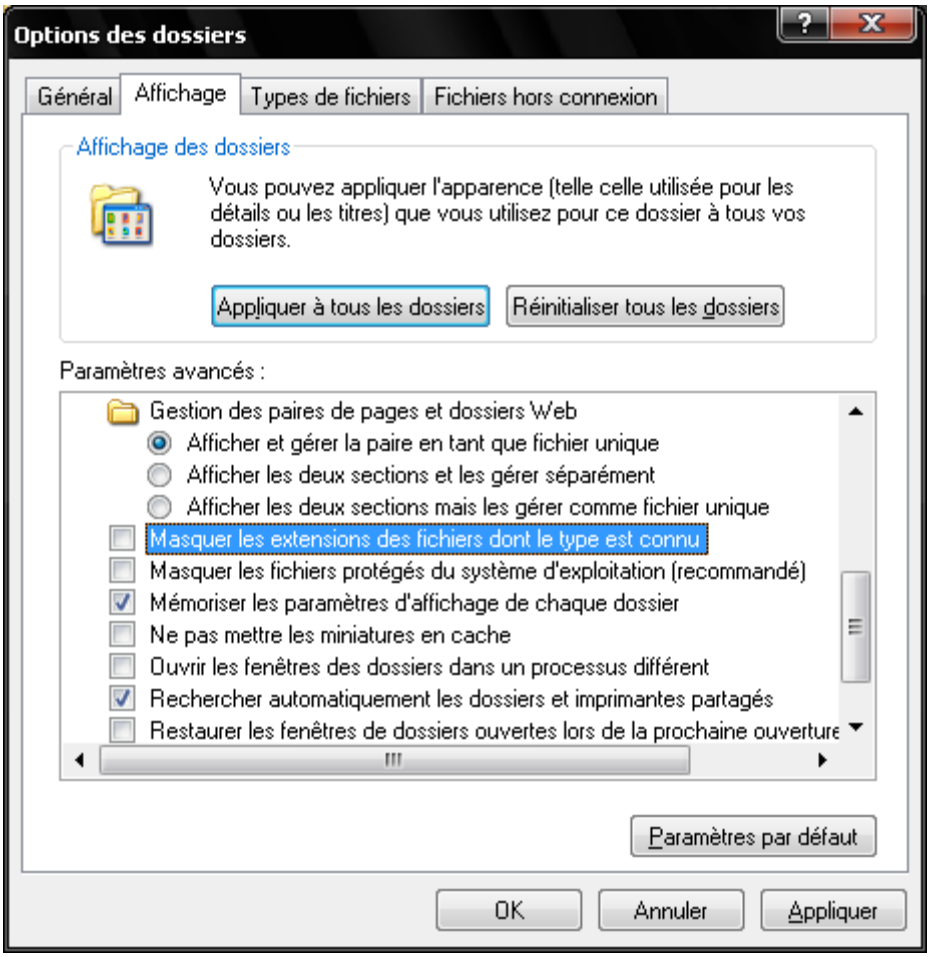
Maintenant que tout cela est dit, on va pouvoir installer MasterBoy sur la PSP et le démarrer (prenez la version kernel, pour des raisons qui seront décrites plus bas).

Ensuite, aller dans Miscenalleous, User Interface, USB Connection et appuyer sur croix pour l'activer. Si l'option est grisée, c'est que vous n'utilisez pas la version kernel. Sur PSP, le mode kernel est malheureusement nécessaire pour activer l'USB depuis un programme. Toutefois vous pouvez toujours utiliser un autre soft, tel que IRShell pour activer l'USB. Dans ce cas, démarrez simplement IRShell, activez l'USB (analog vers le haut) et démarrez ensuite MasterBoy ; l'USB restera activé durant toute la durée où MasterBoy est ouvert.

Ensuite mettez le jeu Game Boy que vous souhaitez colorer dans un dossier de votre PSP. Le nom du jeu importe peu, dans notre cas ce sera Wario Land. La ROM du jeu peut être au format gb, sgb, zip, etc. Créez dans le même dossier un fichier qui porte le même nom mais qui a l'extension .pal.ini. Exemple : Wario Land.pal.ini.

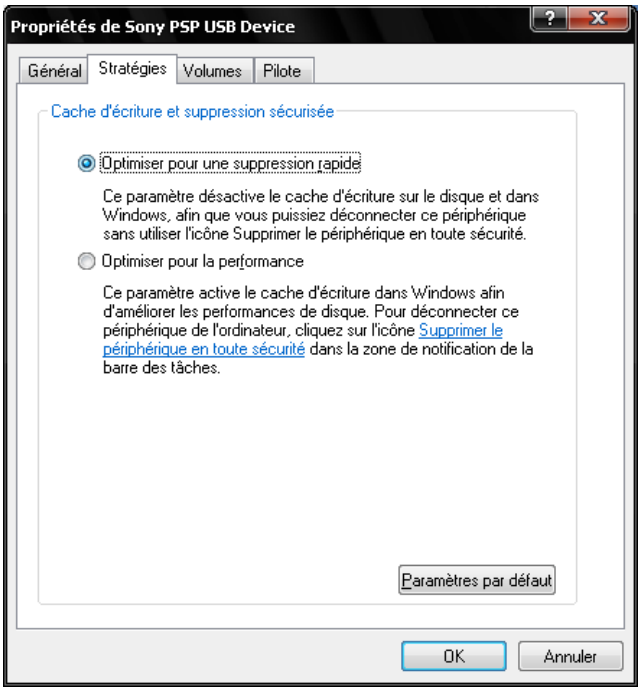
Important : Faites attention à ce que votre OS affiche les extensions des fichiers, ce qui n'est pas le cas par défaut sous Windows ! Sinon vous risquez d'avoir un fichier nommé .pal.ini.txt, et MasterBoy ne le trouvera pas.

Pour cela, allez sous Explorateur, Outils, Options des dossiers, onglet Affichage et décochez l'option « Masquer l'extension des fichiers dont le type est connu » :



Sur Windows Vista il y a un bouton du genre « Organiser » parmis les boutons de l'explorateur. Vous pourrez ensuite accéder à une boîte de dialogue quasiment identique.

Si vous utilisez un système autre que Windows, il vous faudra le paramétrer afin qu'il ne « buffère » pas les données USB mais les écrive immédiatement sur le périphérique au moment où vous enregistrez un fichier. Je ne peux malheureusement pas vous dire comment faire cela sous Linux ou Mac OS X, le mieux est de rechercher sur Internet.



Nous voilà donc avec nos fichiers sur la PSP.

Tout d'abord il faut initialiser le fichier de configuration de palette avec un script vide. On verra plus tard à quoi les commandes proposées servent. Ouvrez le fichier avec un éditeur de texte du type Bloc-notes et copiez-y le texte suivant.

```
#Initialization
Init:
  #The tiles we rely on for CRC
  ColorIt.addTileCrc 0, 383
  #For debugging
  ColorIt.autoShowVramCrc = true

#Fall down here to set the default profile upon initialization
[Default]:
  #Create a gray palette
  ColorIt.setPalette 0, rgb(255,255,255), rgb(168,168,168), rgb(88,88,88),
rgb(0,0,0)
  #By default, we use the standard gray palette for everything
  ColorIt.addTileRule 0, 383, 0
  End
```

Vous pouvez enregistrer ce fichier et le garder ouvert avec le Bloc-notes.

Maintenant lancez Wario Land avec VisualBoyAdvance et en parallèle avec MasterBoy. Pour débuter, nous n'allons nous occuper que du premier niveau, et créer une palette qui sera globale pour tout le jeu. Ne vous inquiétez pas, on pourra en faire une spécifique à chaque niveau par après.

[Default] : est ce qu'on appelle une **section**. Une section est définie par une ligne toujours terminée par le caractère ' :'. MasterBoy exécute le code d'une section à chaque fois que la palette pourrait avoir changé :

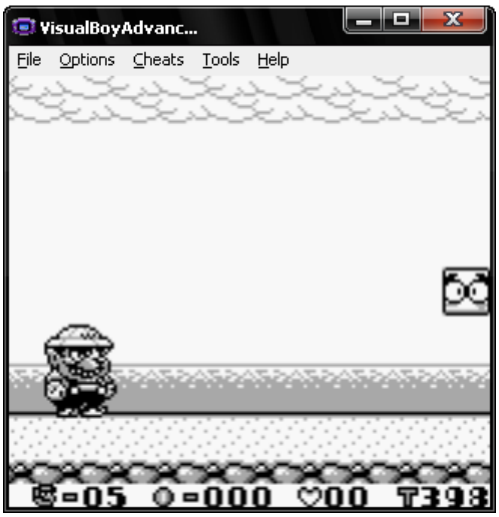
- Démarrage du jeu
- Changement de scène
- Chargement d'un state

La section [Default] est toujours exécutée, alors que certaines sections (nous verrons plus tard) ne sont exécutées que lorsque des conditions particulières sont réunies.
Pour l'instant, sachons juste que nous allons utiliser le fait que MasterBoy recharge le fichier lorsque nous chargerons une sauvegarde. Nous allons donc créer une sauvegarde pendant le premier niveau de jeu, celui que nous allons colorer (R+haut / bas pour choisir un slot libre, et R+Select pour sauvegarder, R+Start pour charger). Ensuite, lorsque vous aurez modifié le fichier .pal.ini et sauvegardé sur la Memory Stick, il vous suffira de charger l'état pour que MasterBoy passe à nouveau dans la section [Default] et charge la nouvelle palette.

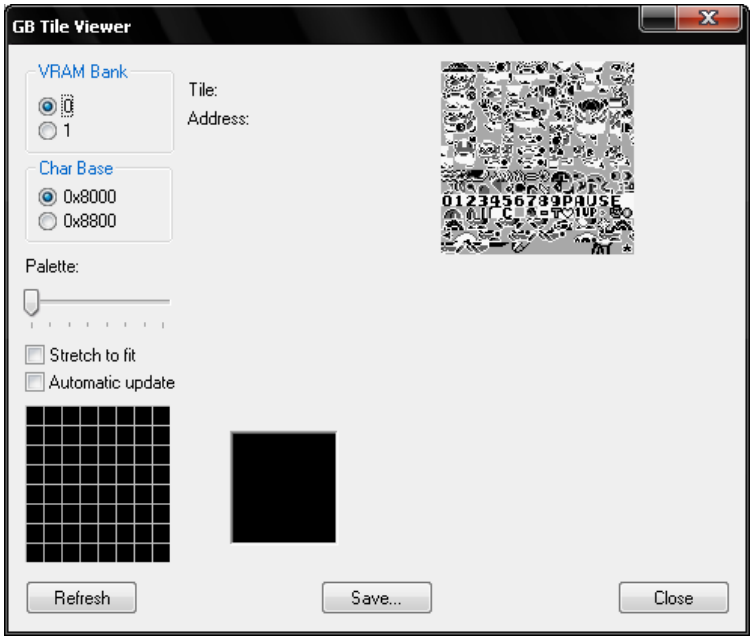
Important : Si vous utilisez la technique de l'USB, il arrive malheureusement que cela ne fonctionne pas bien : lorsque vous enregistrez le fichier depuis le notepad, il est écrit sur la Memory Stick mais MasterBoy voit l'ancienne version en le lisant. Quand ceci arrive, il vaut mieux relancer le jeu. Si le problème ne se résoud toujours pas, désactivez puis réactivez la connexion USB via le menu de MasterBoy.

Note : Ce problème ne provient pas du PC mais de la PSP qui garde en mémoire ce qu'elle vient de lire depuis la Memory Stick, afin qu'un éventuel accès ultérieur soit plus rapide (car les données sont déjà en mémoire), mais du coup elle n'est pas « au courant » que les données ont changé et continue de les utiliser, alors qu'elles ne sont plus à jour. Si vous rencontrez des problèmes **à tous les coups**, c'est probablement un problème de configuration de l'ordinateur. Vérifiez que vous avez bien fait ce qui est décrit au début du document dans la section *Pour commencer*. Pour un maximum de succès, essayez de défragmenter la Memory Stick (il faut donc qu'elle ne soit pas pleine).

Vous voilà maintenant en parallèle avec VisualBoyAdvance dans le premier niveau de jeu.

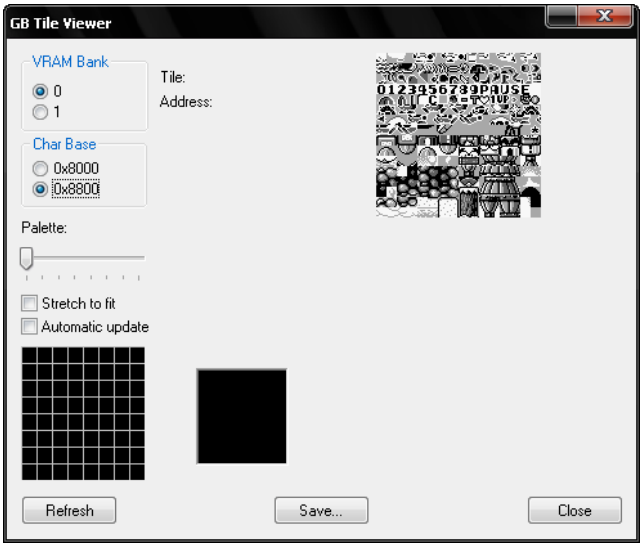


VisualBoyAdvance dispose d'une fonction intéressante qui vous permet d'afficher le contenu de la mémoire vidéo. Allez dans **Tools**, puis **Tile Viewer**. Vous verrez quelque chose comme ceci :



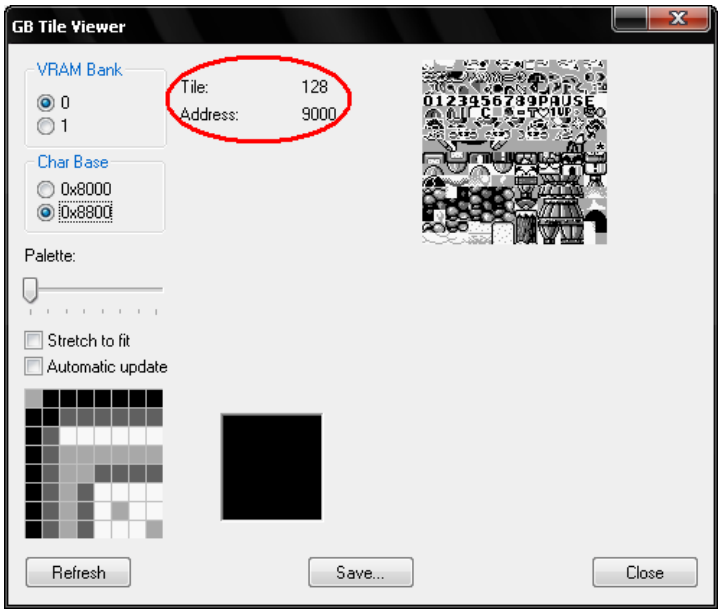
Pour info, la mémoire vidéo (ou aussi appelée VRAM) est un emplacement mémoire spécial de la Game Boy réservé aux graphismes. Dès qu'un jeu veut afficher quelque chose à l'écran (un personnage ou un fond) il devra copier les images en mémoire vidéo. Cela nous facilite bien le travail, car du coup on n'a pas à chercher ailleurs !

Cette boîte de dialogue liste les **tiles** présents dans la mémoire vidéo. Une tile est un bloc de 8x8 qui est utilisé pour composer l'écran. Chaque objet ou arrière plan de jeu est composé d'une série de blocs de 8x8 qui sont réutilisés. Dans l'image ci-dessus on peut reconnaître Wario (ajustez éventuellement la palette pour mieux voir). Si vous choisissez de voir la suite de la mémoire (tiles 128 et plus) en cliquant sur 0x8800, vous verrez quelque chose d'un peu plus intéressant :



On remarque ici tous les éléments de décor du niveau en cours ! Amusons nous donc à colorier les blocs  !

En cliquant sur le bloc, on voit que VisualBoyAdvance (VBA pour les intimes) indique le n° de tile.



En fait ce bloc n'est pas constitué que d'une tile mais de 4 tiles les unes à côté de les autres ! Hé oui, il fait 16x16 si on le mesure bien (4 tiles de 8x8). La première est la n° 128 et la dernière 131. On va donc créer une palette spécifique, à laquelle on assignera un numéro (ex. 1) et on indiquera à MasterBoy que les tiles entre 128 et 131 utilisent la palette n° 1, ainsi vous aurez un bloc coloré.

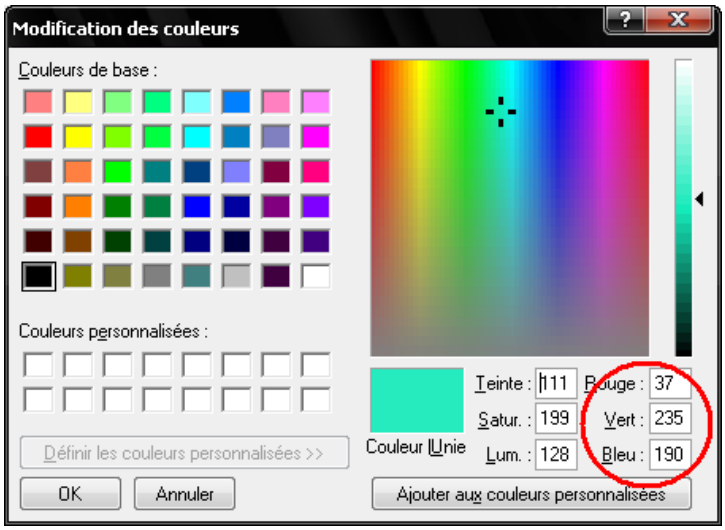
Création d'une palette

Une palette est simplement une liste de couleurs. Sur Game Boy, comme il y a quatre niveaux de gris, nous allons pouvoir définir une couleur pour chaque niveau de gris. Une palette sera donc constituée de 4 couleurs, la première correspondant au blanc, la deuxième au gris clair, la troisième au gris foncé et la dernière au noir.

Revenons dans notre fichier Wario Land.pal.ini, dans la section [Default]. On remarque un premier code :

```
  #Create a gray palette
  ColorIt.setPalette 0, rgb(255,255,255), rgb(168,168,168), rgb(88,88,88),
rgb(0,0,0)
  #By default, we use the standard gray palette for everything
  ColorIt.addTileRule 0, 383, 0
```

La première ligne crée une palette dont le n° est **0**, composé de **4 couleurs** (en bleu). Les lignes précédées par un # (que j'ai mises en gris) ne sont pas exécutées, ce sont des commentaires, juste là pour expliquer ce que les lignes font ; c'est plus pratique pour s'y retrouver.
La deuxième ligne définit que les tiles entre **0 et 383** doivent utiliser la palette n° **0**. Comme il y a 384 tiles au total (0 à 383) ce code signifie en gros que TOUT devient gris (car la palette 0 définie plus haut est constituée de tons de gris).
Voilà comment sont définies les couleurs : rgb(**1**, **2**, **3**), où **1** est la composante de rouge (entre 0 et 255), **2** la composante de vert et **3** la composante de bleu. Amusez-vous avec les couleurs de MSPaint pour comprendre comment ce système RGB fonctionne (menu Couleurs, Modifier les couleurs).



Si vous avez l'habitude du web, vous pouvez aussi simplement définir votre couleur en hexadécimal : 0xffffff donne du blanc par exemple.

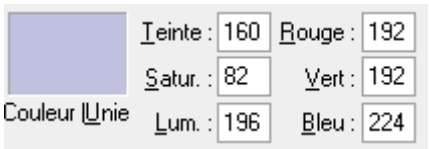
Nous allons définir plus bas des directives pour le bloc utilisant les tiles entre 128 et 131 (le bloc qu'on veut colorer), et créer une nouvelle palette qu'on leur associera. Le code qui fait ça est donc :

```
  #Palette des blocs
  ColorIt.setPalette 1, rgb(192,192,224), rgb(128,128,192), rgb(64,64,160),
rgb(0,0,128)
  #Associe la palette aux blocs
  ColorIt.addTileRule +128, +131, 1
```

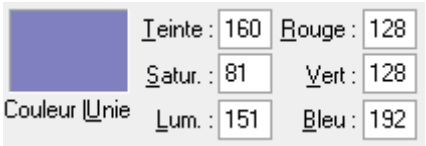
A ajouter juste avant le **End**, qui marque la fin de la section (rien de ce qui est après ne sera exécuté, donc ça ne sert à rien d'y mettre quoi que ce soit pour le moment). Le + devant le n° des tiles est nécessaire car nous nous situons sur la banque 0x8800 (souvenez-vous, on a cliqué sur ce bouton radio au tout début). Il n'y a pas de '+' si on est sur la banque 0x8000.

Quant aux 4 couleurs, je les ai choisies à partir de paint. La première représente la nuance de blanc, elle est donc naturellement plus claire. La seconde le gris clair (donc un peu plus foncée), la troisième le gris foncé (donc plus foncée encore) et la dernière le noir (donc une couleur très sombre en général, le noir étant souvent recommandé pour un meilleur contraste) :

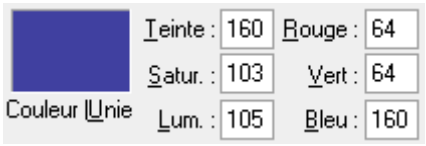
`rgb(192,192,224) :`



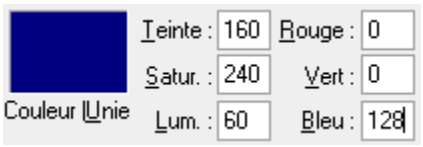
`rgb(128,128,192) :`



`rgb(64,64,160) :`



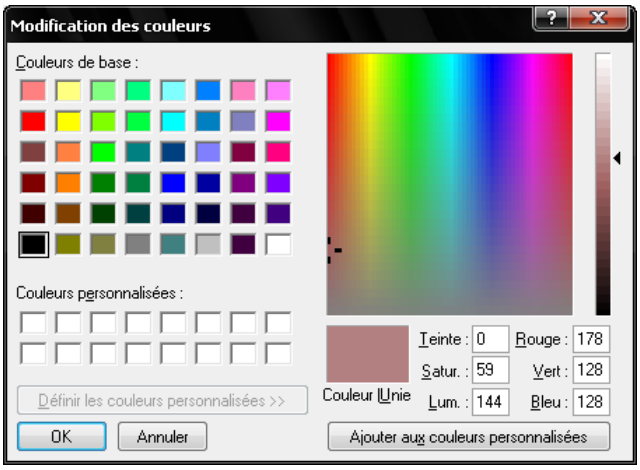
`rgb(0,0,128)` :



Astuce : Si vous divisez tous les composantes (rouge, vert et bleu) par deux, vous obtiendrez la même couleur mais en version 2x plus foncée. Ca peut souvent être utile pour donner des effets de relief. De la même manière, si vous faites une moyenne entre toutes les composantes et 255, vous obtiendrez une couleur 2x plus claire. Par exemple pour 255, 128 et 0, la version plus claire sera 255, 192, 128.
De plus, en général plus les composantes sont éloignées, plus la couleur sera vive. Par exemple 255, 0, 0 représente du rouge vif, mais 192, 64, 64 représente un rouge fade, et lorsque les trois composantes sont identiques, on a du gris, dont la luminosité dépend des trois composantes. Par exemple 64, 64, 64 représente du gris foncé, 128, 128, 128 du gris intermédiaire, et ainsi de suite.

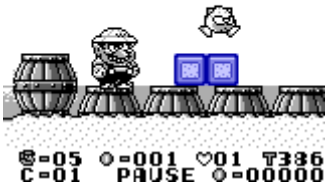
Ne vous fiez pas trop au curseur de Paint (tout à droite), il contrôle la luminance, qui n'est pas exactement pareille que la luminosité des couleurs ; lorsque le curseur est au milieu, la couleur est toujours dans sa version la plus colorée, ce qui n'est pas le cas avec la luminosité réelle.

Par exemple, le rouge n'est pas la version plus claire de cette couleur. La version plus claire ressemblerait plutôt à ça (selon le calcul que j'ai cité plus haut) :



D'ailleurs si on fait le test, on remarque que l'effet de relief semble beaucoup plus naturel ainsi.

Pour continuer, enregistrez le fichier et chargez le save state : les blocs devraient apparaître en bleu.



Si vous avez déjà utilisé le Super Game Boy, ce ne sera sûrement pas un problème pour vous de choisir les couleurs qui vont bien. Dans le cas inverse, vous pouvez effectuer divers essais pour voir ce que cela donne (il suffit de modifier, d'enregistrer à nouveau et de charger le save state).

Vous pouvez ensuite colorer plusieurs tiles ou groupes de tiles avec la même palette, ou créer une autre palette pour les groupes de tiles suivants. Vous n'avez qu'à mettre le code à la suite, de la même façon :

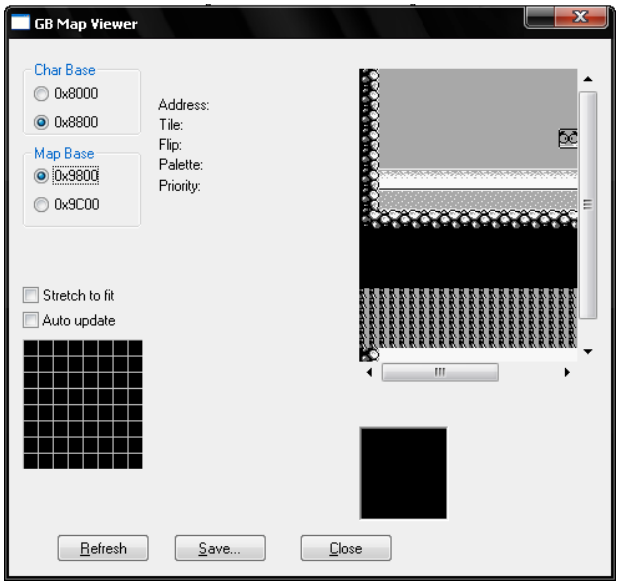
```
  #Palette n° 1 : les blocs
  ColorIt.setPalette 1, rgb(192,192,224), rgb(128,128,192), rgb(64,64,160), rgb(0,0,128)
  #Associe la palette aux blocs
  ColorIt.addTileRule +128, +131, 1
  #Les tiles 136 à 139 représentent un bloc fissuré, donc mêmes couleurs, même palette (mais on aurait pu les rendre différents en leur donnant leur propre palette)
  ColorIt.addTileRule +136, +139, 1

  #Palette n° 2 : Wario
```

```
  ColorIt.setPalette 2, rgb(255,255,255), rgb(255,224,128),
rgb(128,112,64), rgb(0,0,0)
  #Associée aux tiles de Wario
  ColorIt.addTileRule 0, 143, 2
  #Ne pas oublier le End pour terminer
  End
```

C'est à peu près tout ce dont on aura besoin.

Si vous ne trouvez pas une tile…

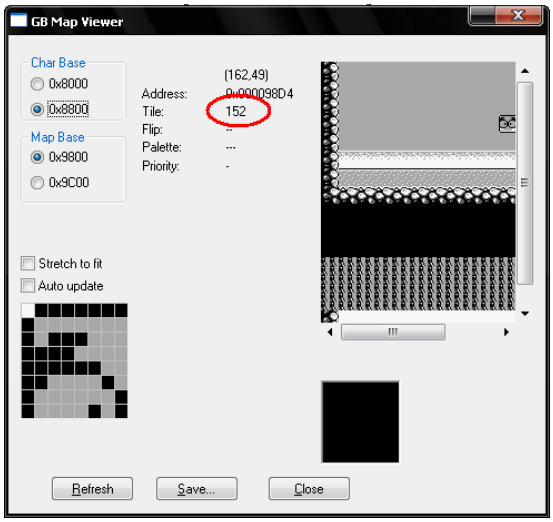
Le **Tile viewer** affiche toute la mémoire vidéo, et parfois cela peut paraître plutôt fouilli. Si vous n'y retrouvez pas une tile que vous voyez à l'écran, vous pouvez ouvrir le **Map Viewer**, du menu **Tools**.



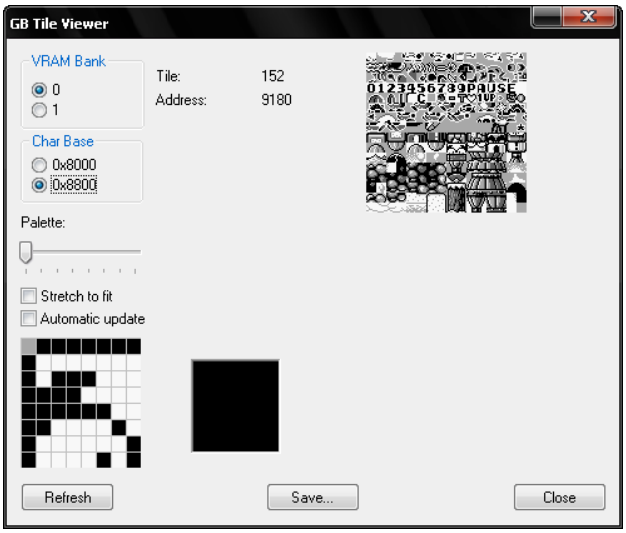
Si vous ne voyez rien de beau apparaître, essayez de cliquer sur un des boutons de **Char Base** et **Map Base** jusqu'à ce que cela ressemble à ce qu'il y a sur l'écran (c'est l'une des quatre possibilités).

Note : seul le fond d'écran (décor) apparaît, les personnages et tout ce qui bouge ne sont pas inclus (il faut utiliser l'**OAM Viewer** pour cela).

Cliquez sur un des blocs de la map, et son n° de tile apparaîtra :



Notez le n° de tile, 152. Vous pouvez ensuite aller la rechercher dans le **Tile viewer**.



Si ce n'est pas la bonne tile, essayez de cliquer sur 0x8000 ou 0x8800 (**Char Base**). Cette valeur devrait être la même que dans le **Map viewer**.

Astuce

Dans les paragraphes plus haut, nous avons vu que dans la section **[Default]** on créait une palette grise qu'on associait à tous les éléments par défaut. Lorsque vous associez plusieurs palettes à un même bloc, c'est toujours le dernier qui l'emporte. Ici par exemple tous les blocs entre 0 et 383 devenaient gris avec la première commande, mais la deuxième recolorait les blocs 128 à 131 en bleu. Le résultat est que tous les blocs seront gris, excepté ceux entre 128 et 131 qui ont été redéfinis plus tard. Cette astuce peut être bien utile dans certaines situations.

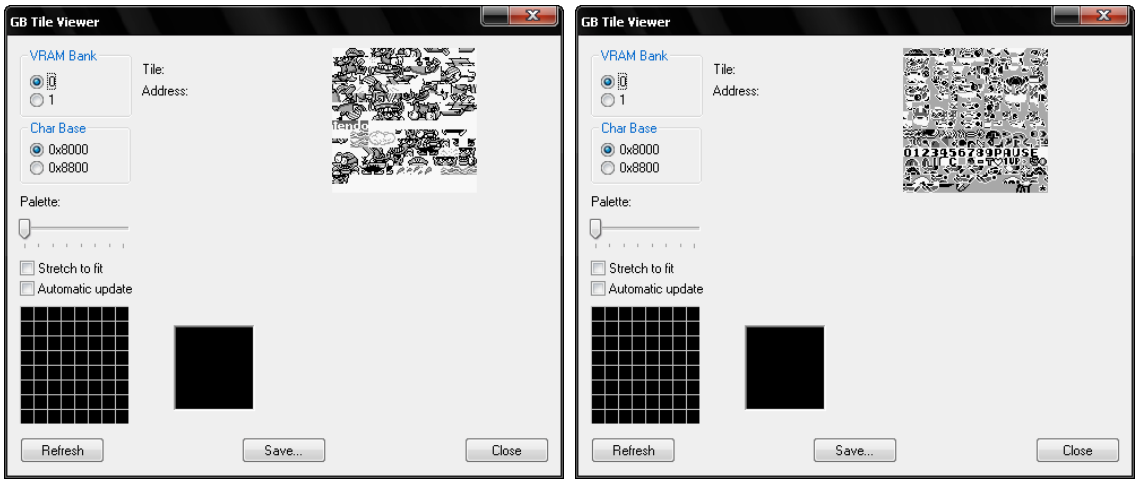
Tenir compte du niveau en cours…

Hé oui, pour le moment c'est bien beau tout ça, c'est joli sur le niveau 1, mais très moche sur les autres. Nous allons voir comment faire pour définir une configuration spécifique à un niveau.

Tout d'abord il faut savoir que, excepté pour le Super Game Boy, les jeux Game Boy n'ont pas du tout été prévus pour être colorés. De ce fait il n'y a pas d'indication du type « je viens de changer de niveau, les couleurs seront peut être différentes ». En fait, on ne sait pas du tout quand le niveau de jeu change. Pour pallier à ce problème, il fallait bien trouver une solution !

Si vous regardez le **Tile viewer** de VisualBoyAdvance, et que vous cliquez sur la case à cocher **Automatic update**, vous remarquerez que les tiles sont toujours identiques pour toute la durée du niveau (excepté celles qui sont animées, mais nous en parlerons plus tard). Lorsqu'on change de niveau par contre, en général beaucoup de tiles changent, seuls les éléments communs tels que le personnage, le texte et certains ennemis restent en mémoire vidéo. Ainsi la solution proposée par MasterBoy est d'identifier un niveau en fonction de ce qui se trouve en mémoire vidéo ! En fonction du cas, on saura quelles palettes charger.

A gauche, les tiles de l'écran titre, à droite les tiles du niveau 1 :



Afin de reconnaître le contenu de la mémoire vidéo, on va utiliser une astuce que vous avez peut être déjà rencontrée sur les fichiers : le CRC. Un CRC (ou somme de contrôle) donne un nombre unique en fonction des données que le fichier contient. Ainsi le CRC de deux fichiers identiques est également identique. Mais si une erreur de transmission s'est produite et qu'un des fichiers diffère, ne serait-ce que d'un octet, le CRC sera différent. Le CRC permet ainsi d'être sûr qu'on est bien en présence du fichier qu'on attend.

Dans le cas de la mémoire vidéo, on pourra ainsi être sûr qu'on est bien sur le niveau qu'on attend, car la mémoire vidéo différera sur un autre niveau et donc son CRC également. En gros, nous allons donc définir une série de palettes qui seront appliquées en fonction du CRC de la mémoire vidéo.

<u>Pour commencer…</u>

Nous avons vu plus haut un problème : certaines tiles sont animées et donc changent en milieu de niveau ! Si on les prend en compte dans le CRC, le CRC diffèrera à chaque fois que ces tiles changent, ce qui n'est pas pratique. MasterBoy permet de choisir sur quelles tiles le CRC doit être appliqué pour contourner ce problème.

Pour choisir sur quelles tiles vous devriez appliquer le CRC, parcourez quelques niveaux du jeu avec le **Tile Viewer** de VisualBoyAdvance (en activant la case à cocher **Automatic update**), et trouvez quelques tiles qui changent à chaque niveau et qui ne sont pas animées. En gros, trouvez simplement quelques tiles qui vous permettent de déterminer **à coup sûr** à quel niveau vous êtes, sans confusion possible.

Note : dans le cas de Wario Land, comme beaucoup de jeux, on remarque que plusieurs courses (stages) utilisent le même décor. Par exemple celui de la course 1 (« plage ») et également utilisé dans la course 3. Dans ce cas c'est pratique, il suffit de détecter le décor de type « plage » et le colorer, et toutes les courses qui utilisent le même décor seront directement colorées elles-aussi.

Si on prend Wario Land, on peut déjà dire que les tiles que j'ai encadrées en rouge sont de très bons candidats, car si on connaît le jeu on sait qu'on ne les retrouve que dans ce niveau, donc qu'elles ont très peu de chances d'exister dans la mémoire vidéo des autres niveaux. Il nous reste à vérifier qu'aucune d'entre elles n'est animée (dans ce niveau comme les autres). L'idéal est d'avoir le moins de tiles possible incluses au CRC (cela limite les risques), mais toutefois suffisamment pour être certain du niveau.

Un petit contrôle dans un autre niveau nous confirmera qu'elles ont bien changé.

Dans l'idéal, on peut également s'assurer qu'il soit possible de distinguer l'écran titre, le générique et les mini-jeux, mais ça donne beaucoup d'heures de jeu ! Donc ce n'est pas nécessaire normalement. Au pire, si vous avez inclus trop ou trop peu de tiles, vous pourrez toujours adapter plus tard, mais malheureusement vous devrez recalculer tous les CRC car ils dépendent du nombre de tiles ! (ça veut dire repasser dans tous les niveaux pour que MasterBoy vous l'indique, comme on verra plus loin)

Cela nous apprend une chose importante avant de commencer : gardez toujours un Save State à chaque niveau afin de pouvoir y revenir plus tard. Dans le cas de Wario Land ce n'est pas nécessaire, grâce à la carte qui permet d'explorer les anciens niveaux.

Comment mettre tout ça en place ?

Cela se fera dans le fichier script, grâce à une modification aux commandes que vous avez copiées au début :

```
#Initialization
Init:
  #The tiles we rely on for CRC
  ColorIt.addTileCrc 0, 383
  #For debugging
  ColorIt.autoShowVramCrc = true
```

Ce code, qui est exécuté au chargement du jeu, définit quelles tiles vous voulez utiliser pour calculer le CRC. Par défaut, il les inclut toutes, de 0 à 383 (il y a 384 tiles au total, la dernière étant +255, le + signifiant un offset de 128 comme indiqué dans la section *Pour les experts*, soit 383).

La deuxième ligne, autoShowVramCrc, permet d'activer l'affichage automatique du CRC lorsque vous changez de **scène**. MasterBoy considère qu'une **scène** se termine lorsque l'écran devient blanc, comme lors du passage du logo Nintendo à l'écran titre, ou d'un niveau à l'autre. C'est à ce moment-là qu'il faudra éventuellement changer de palette.

Note : il existe peut être des jeux qui changent de niveau sans passer par un écran blanc. Dans ce cas, vous ne verrez jamais de message « New VRAM CRC » au fond de l'écran. Ces jeux sont malheureusement incompatibles avec le système actuel, mais vous pouvez me signaler si cela arrive afin que je puisse tenter de trouver une solution.

Dans le cas de Wario Land, j'ai décidé d'utiliser les tiles 176 à 191. Espérons que c'est un bon choix. Le code pour définir celles-ci devient :

```
  #The tiles we rely on for CRC
  ColorIt.addTileCrc 176, 191
```

Vous pouvez ajouter des groupes de tiles dans des zones différentes en mettant plusieurs lignes. Exemple pour inclure les tiles 1 à 2, 10 à 11 et 20 à 50 :

```
  ColorIt.addTileCrc 1, 2
  ColorIt.addTileCrc 10, 11
  ColorIt.addTileCrc 20, 50
```

La ligne autoShowVramCrc pourra être commentée ou effacée une fois que la coloration du jeu sera terminée, mais en attendant elle vous permettra de connaître les CRC des différentes **scènes** du jeu. Lorsque vous voyez par exemple ce message apparaître sur l'écran titre, notez le CRC associé, par exemple :

Le reste est simple : plutôt que mettre notre code créant les palettes dans default, on mettra dans une section portant le CRC entre crochets. Exemple :

```
#Ecran titre
[7d0f0982]:
  ..............
  End
```

Voici un exemple de fichier contenant tout ce que nous avons vu :

```
#Initialization
Init:
  #The tiles we rely on for CRC
  ColorIt.addTileCrc 176, 191
  #For debugging
  ColorIt.autoShowVramCrc = true

#Fall down here to set the default profile upon initialization
[Default]:
  #Create a gray palette
  ColorIt.setPalette 0, rgb(255,255,255), rgb(168,168,168), rgb(88,88,88),
rgb(0,0,0)
  #By default, we use the standard gray palette for everything
  ColorIt.addTileRule 0, 383, 0
  End

#Title screen
[7d0f0982]:
  #Create some palette
  ColorIt.setPalette 1, rgb(255,0,0), rgb(168,0,0), rgb(88,0,0), rgb(0,0,0)
  #Color some tiles with it
  ColorIt.addTileRule +128, +191, 1
  ColorIt.addTileRule +1, +10, 1
  End
```

Astuce

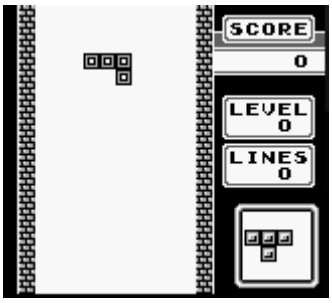
Notez que dans ce document on a appris comment colorer les jeux avec beaucoup de détails, mais en réalité tout ce qu'on pourrait se contenter de faire est de redéfinir la palette n° 0 à chaque niveau : elle est associée à tous les éléments du décor par défaut. Ainsi on peut définir 4 couleurs exactement comme sur un Super Game Boy.

```
#Initialization
Init:
  #The tiles we rely on for CRC
  ColorIt.addTileCrc 176, 191
  #For debugging
  ColorIt.autoShowVramCrc = true

#Fall down here to set the default profile upon initialization
[Default]:
  #Create a gray palette
  ColorIt.setPalette 0, rgb(255,255,255), rgb(168,168,168), rgb(88,88,88),
rgb(0,0,0)
  #By default, we use the standard gray palette for everything
  ColorIt.addTileRule 0, 383, 0
  #Reset tiles so that all use the palette 0
  ColorIt.addTileRule 0, 383, 0
  End

#Title screen
[7d0f0982]:
  #Redefine the main palette
  ColorIt.setPalette 0, rgb(255,0,0), rgb(168,0,0), rgb(88,0,0), rgb(0,0,0)
  End
```

Une autre possibilité est aussi de faire tout ce qui est commun à tous les niveaux dans la section Default. Par exemple, on voit que le personnage de Wario se trouve toujours sur les mêmes numéros de tile, quel que soit le niveau. Ainsi on peut simplement créer une palette et colorer ces tiles dans la section Default. Si jamais un niveau a besoin de quelque chose de différent, il peut toujours le spécifier plus tard (souvenez vous que l'effet des commandes est cascadé ; en cas de conflit, la dernière prime).

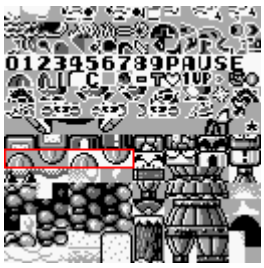
Redéfinition des graphismes

Il peut arriver que dans certains cas vous vous retrouviez « bloqué » car vous ne pouvez pas trouver 4 couleurs qui satisfassent une tile. Ou alors vous avez simplement envie de « customiser » un peu les graphismes, rajouter du relief aux pièces ou pimper un peu cette vieille police de caractères !



En gros, la redéfinition de graphismes consiste juste à redessiner une tile donnée. Cela va éventuellement nous aider à résoudre des problèmes importants, comme on en retrouve justement dans Wario Land (quel hasard !).

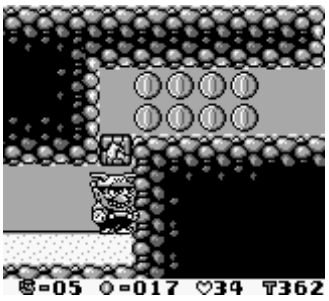
Prenons donc Wario Land.



Vous voyez ces pièces (entourées en rouge) ? En fait, on les retrouve en deux exemplaires, une fois lorsqu'elles sont sur un fond blanc et une fois sur un fond gris.

Sur fond blanc :

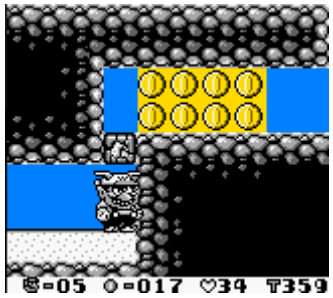
Sur fond gris :



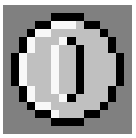
On peut facilement colorer la pièce sur fond blanc en choisissant des couleurs du genre blanc, jaune, jaune foncé et noir.

Malheureusement c'est beaucoup moins évident pour celle sur fond gris ! La nuance « gris clair » est utilisée à la fois pour le milieu de la pièce et pour le fond (dans les bords). Voici ce qu'on aura si on utilise la même palette que pour la pièce sur fond blanc :

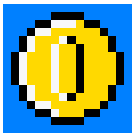
Dans le jeu, ça risque de ne vraiment pas être beau !



La seule solution est donc de faire en sorte que la couleur utilisée pour le fond soit différente dans l'image même ! Si on redessine par exemple l'image ainsi :



On perd l'usage du gris foncé au milieu de la pièce, mais on le réserve pour l'extérieur ! Ainsi on pourra utiliser le bleu pour l'extérieur. Une fois coloré en utilisant du jaune pour le gris clair et du bleu pour le gris foncé, on a ce qu'il faut :



Alors, comment redessiner une tile ? Reprenons notre script, et voyons ce qu'il faut ajouter (au même endroit que d'habitude).

```
#Crée une tile personnelle numérotée 0
ColorIt.setTilesetData 0, "ffffffff 00ff00ff ff00ff00 00000000"
#Cette tile perso remplacera la vraie tile n° +211
ColorIt.setTile +211, 0
```

Les caractères entre guillemets sont les données de l'image au format Game Boy. Ce format est expliqué dans la section **Pour les experts**. Si vous en avez le courage, c'est une bonne idée de l'utiliser car il est plus compact.

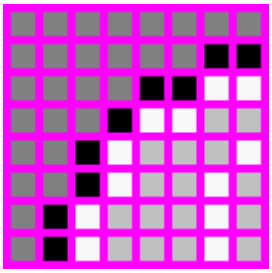
Mais en attendant, nous allons voir le mode « simple ». Dans ce mode, spécifié par l'ajout de la lettre L juste avant les guillemets, vous définissez les pixels un à un, un 0 représentant du blanc, 1 du gris clair, 2 du gris foncé et 3 du noir. Un exemple :

```
  ColorIt.setTilesetData 0, L "00000000 11111111 22222222 33333333 00000000
11111111 22222222 33333333"
```

Les pixels sont définis de gauche à droite, par lignes (ici séparées par un espace) de haut en bas. Cet exemple donne une tile avec une première ligne entière (8 pixels) de blanc (0), une deuxième de gris clair, une troisième de gris foncé, et une dernière de noir, et ce motif est répété. Le résultat est le suivant :

Redessinons maintenant notre pièce comme indiqué plus haut. Il va s'agir de 4 tiles individuelles. Comme ceci :

Nous allons faire la première tile ensemble, j'ai entouré chaque pixel par un carré rose :



- Notre première ligne (tout en haut) est constituée uniquement de pixels gris foncé, donnant un premier groupe : 22222222.
- Pour la deuxième ligne, ça commence avec 6 pixels de gris foncé, et ensuite 2 pixels de noir, donnant un groupe 22222233.
- La troisième ligne est constituée de 4 pixels de gris foncé, 2 pixels de noir puis 2 pixels de blanc, donnant 22223300.
- Et ainsi de suite, donnant la chaîne finale suivante : "22222222 22222233 22223300 22230011 22301110 22301101 23011101 23011101".

Cela donne donc le code :

```
  ColorIt.setTilesetData 0, L "22222222 22222233 22223300 22230011 22301110
22301101 23011101 23011101"
```

Ce qui crée une tile personnelle numérotée **0**. On va ensuite créer les tiles n° 1, 2 et 3 pour terminer cette pièce.

```
  ColorIt.setTilesetData 1, L "22222222 33222222 01332222 11113222 31111322
13111322 13111132 13111132"
  ColorIt.setTilesetData 2, L "23011101 23011101 22301101 22301110 22230011
22223301 22222233 22222222"
  ColorIt.setTilesetData 3, L "13111132 13111132 13111322 31111322 11113222
11332222 33222222 22222222"
```

Qu'on va ensuite associer à des tiles existantes (cela les remplacera). En l'occurrence on va prendre celles des pièces (144 à 147) :

```
ColorIt.SetTile +144, 0
ColorIt.SetTile +145, 1
ColorIt.SetTile +146, 2
ColorIt.SetTile +147, 3
```

Qu'on peut remplacer par une simple commande :

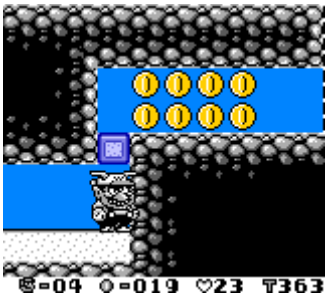
```
ColorIt.SetTile +144, +147, 0
```

MasterBoy va assigner une tile personnelle à toutes les tiles de 144 à 147 en incrémentant le n° de tile personnelle à chaque fois (0, 1, 2, 3). Voir la section **Pour les experts** pour plus d'informations.

Ne reste plus qu'à colorer ces tiles comme on l'avait dit :

```
ColorIt.setPalette 2, rgb(255,255,255), rgb(255,207,0), rgb(0,128,255), rgb(0,0,0)
ColorIt.addTileRule +144, +147, 2
```

Et ô magie, voilà le résultat !



Cas insolubles

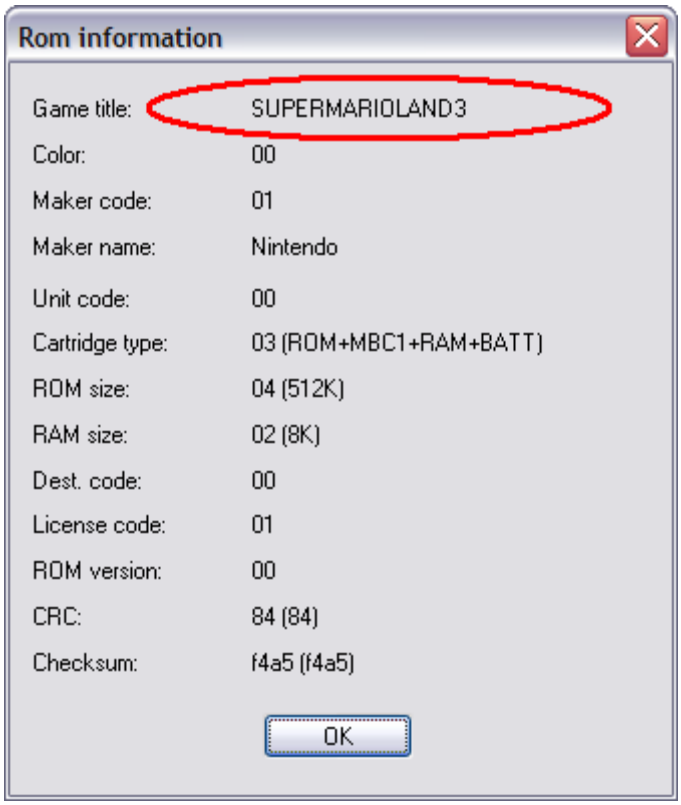
Il existe des cas où une même tile a plusieurs utilisations dans un même niveau. Par exemple une tile grise unie peut être utilisée pour le ciel mais aussi pour le remplissage du sol par exemple. Il n'y a malheureusement pas la possibilité de les distinguer et ce sera la même palette qui sera appliquée partout (car c'est la même tile).
Dans ce cas, il n'y a pas de solution simple, il faut trouver des couleurs neutres qui conviennent pour tous les objets qui l'utilisent.
Heureusement ces cas sont plutôt rares, mais lorsqu'on colore des fonds unis, il vaut mieux se méfier et utiliser des couleurs plus neutres , pour si jamais… (ci-dessus, l'exemple du fond bleu dans Wario Land est donc une mauvaise idée).
D'expérience, il vaut mieux également éviter de colorer les fonds blancs en bleu ciel par exemple, car le blanc uni est énormément utilisé dans les jeux Game Boy en général, et vous avez un grand risque de conflit.

Une fois terminé…

Une fois que vous avez terminé votre fichier de script et que vous souhaitez le distribuer, vous pouvez le renommer en le nom de la cartouche Game Boy originale. Il suffira de le mettre dans le

répertoire **Colorpak** (à la racine du dossier de MasterBoy) et MasterBoy ira l'exécuter, quel que soit le nom du fichier zip de votre ROM (par exemple **SUPERMARIOLAND3** est le nom de la carte et MasterBoy le trouvera, que votre ROM s'appelle Wario Land (UE) [!].zip, Wario Land - Super Mario Land 3.zip ou WarioLand.gb).

Pour ce faire, il suffit d'ouvrir le jeu avec VisualBoyAdvance et aller dans **File**, **Rom information**.



Renommez votre fichier SUPERMARIOLAND3.pal.ini et mettez-le dans **Colorpak** ; ainsi feront ceux qui voudront tester votre colorisation.

<u>Pour les experts…</u>

Voici le mode d'emploi des commandes à utiliser. Une commande est une ligne dans le fichier script censée effectuer une opération. La ligne commence toujours par la description de ce que vous voulez faire (il faut apprendre le nom de la commande correspondante) et s'en suit les **arguments**. Les arguments sont des valeurs numériques qui permettent à la commande de savoir quoi faire.

Un exemple typique de commande sous Windows est :

```
del fichier.txt
```

La commande est **del**, elle efface un fichier (c'est à retenir) et demande un argument : le nom du fichier à effacer. Forcément, **del** ne sert pas à grand-chose si elle ne sait pas quel fichier effacer, d'où l'utilité des arguments un peu partout.
Certaines fonctions peuvent avoir besoin de plusieurs arguments. Dans ce cas ils sont séparés par une virgule.

Commande ColorIt.SetPalette

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

Cette commande définit une palette. PalNo est le numéro de palette (entre 0 et 47), palColor0, palColor1, … sont les 4 couleurs de la palette que vous voulez définir. Avec cette commande, vous pouvez remplacer une palette existante (e.g. la palette n° 0 existait déjà mais vous voulez lui mettre des couleurs différentes).
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.

Commande ColorIt.AddTileRule

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

Cette fonction définit quelle palette doit être utilisée pour une liste de tiles données. Le premier nombre est le n° de tile de départ et le deuxième le n° de tile d'arrivée. Ces numéros sont éventuellement précédés chacun d'un caractère + pour indiquer qu'elles se trouvent dans la deuxième partie de la mémoire vidéo, c'est-à-dire avec un offset de 128 tiles depuis le début. Le + est à mettre si vous récupérez le n° de tile avec VisualBoyAdvance et que vous avez cliqué sur le bouton 0x8800, et pas de + pour 0x8000.

Commande ColorIt.AddTileCRC

Utilisation :
```
ColorIt.addTileCrc tileStart, tileEnd
```

Ajoute un groupe de tiles spécifique au calcul du CRC. Voir plus haut pour des explications.

Commande ColorIt.SetTileData

Utilisation :
```
ColorIt.setTileData tileNo, "hextiledata"
```
Ou :
```
ColorIt.setTileData tileNo, L "humantiledata"
```

Cette fonction crée une tile personnelle et en définit les données de l'image.

TileNo : N° de tile personnelle, entre 0 et 383.

Dans le premier cas, les données de la tile (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 le deuxième cas, les données de la tile 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 ColorIt.SetTile

Utilisation :

```
ColorIt.setTile tileNo, newTileNo
```

Ou bien :

```
ColorIt.setTile tileStart, tileEnd, newTileNo
```

Associe une tile personnelle à une tile ou à un groupe de tile. La tile personnelle doit avoir été créée avec ColorIt.SetTileData.

Note : si vous associez une tile personnelle à un groupe de tile, le n° est incrémenté à chaque tile. Par exemple si vous définissez la tile personnelle n° 0 pour les tiles 128 à 131, la tile 128 utilisera la tile personnelle n° 0, la tile 129 la tile personnelle n° 1, et ainsi de suite.

Commande ColorIt.AutoShowVramCrc

Utilisation (activer) :

```
ColorIt.autoShowVramCrc = true
```

Utilisation (désactiver) :

```
ColorIt.autoShowVramCrc = false
```

Commande End

Arrête l'exécution du programme. Cette commande est nécessaire à la fin de chaque traitement. Si vous avez plusieurs sections mais que vous oubliez de mettre un **End** entre deux, MasterBoy va continuer l'exécution dans la prochaine section, et ainsi de suite de haut en bas, jusqu'à ce qu'il

rencontre un End. Des problèmes assez vicieux peuvent apparaître, alors vérifiez toujours que vous n'avez pas oublié de End.

Commande Goto

Utilisation :

```
Goto section
```

Cette commande continue l'exécution vers une autre section du code. Exemple :

```
      Instruction1
      Goto Label
      Instruction2

Label:
      Instruction 3
      End
```

Dans ce cas, l'instruction 1 sera exécutée, puis l'instruction 3. Cela peut être utile si vous voulez définir une section de code commune, qui doit être exécutée dans un grand nombre de cas, comme la coloration du personnage principal. Il vous suffira de faire **Goto votreSection** dès que vous aurez terminé les instructions spécifiques au niveau en cours.

Note : Si le nom de la section est entouré de crochets, comme [Default], il faut également les fournir à **Goto**.

Problèmes fréquents

Q : J'ai utilisé la redéfinition des tiles pour ajuster une tile qui me gênait, mais dans certaines scènes, elle apparaît différemment (le gris clair devient blanc et inversement par exemple).

R : Ceci est normal. Le processeur graphique de la Game Boy a un registre spécial appelé « palette », qui permet de redéfinir l'usage des quatre « couleurs ». Il est par exemple utilisé lors des fades (transitions d'un écran à l'autre par le blanc) : le noir devient gris foncé, le gris foncé devient clair, et ainsi de suite, donnant l'effet d'un éclaircissement global de l'écran.
Si le registre de palette est modifié, votre tile personnelle sera également affectée, et la signification des couleurs peut s'en trouver modifiée. A vous de tenir cela en compte lorsque vous dessinez votre tile personnelle ou lorsque vous la colorez.

Q : Certains utilisateurs n'arrivent pas à voir la colorisation automatique pour certains jeux, malgré que le fichier soit nommé selon le nom de la carte originale.

R : Malheureusement, on trouve des ROMs sur Internet dont le header a été modifié, et le nom a changé. Il est impossible de reconnaître de quel jeu il s'agit dans ce cas, excepté de demander à

l’utilisateur de renommer son fichier (à défaut de modifier le header lui-même). Par exemple <Nom d’une ROM>.zip deviendra SUPERMARIOLAND3.zip dans le cas de Wario Land.
Notez que si beaucoup d’utilisateurs vous font la remarque, c’est peut être vous qui avez un fichier dont le header est mauvais.