Mise au point Software pour Cartouches Super-Oric (10)

Quelques Notions sur les Caractères Redéfinis (3)

par André C.

Lors du dernier épisode, nous avons redéfini et affiché le 'A' de CAPS en 4 couleurs (Lores 0). Voyons maintenant ce que ça donne en 16 couleurs.

Lores 1

Soit à redéfinir le caractère 'A' en mode Lores 1 (16 couleurs). Pour commencer on se contentera d'afficher les 8 pixels de la première ligne (n°0) avec les 8 couleurs de l'Oric et de laisser les lignes suivantes en couleur PAPER (couleur n°0). Le tableau suivant montre la répartition des couleurs sur la première ligne (n° 0 à 7 en binaire) :

Lignes	Couleurs							
0	0000	0001	0010	0011	0100	0101	0110	0111
1	0000	0000	0000	0000	0000	0000	0000	0000
2	etc.							

On décompose ensuite les bits de ces couleurs selon les plans 3 (bits n°3), plan 2 (bits n°2), plan 1 (bits n°1) et plan 0 (bits n°0), ce qui donne :

Ligne	Plan 3 = Les bits b3								Plan 2 = Les bits b2								Plan 1 = Les bits b1								Plan 0 = Les bits b0							
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	1	1	0	0	1	1	0	1	0	1	0	1	0	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	etc.																															

Et on peut 'lire' les 8 octets du résultat que l'on regroupe en deux 'words' : #000F (pour les plans 2 et 3) et #3355 (pour les plans 1 et 0). Tous les autres 'words' sont à zéro. On incorpore ces 'words' dans la commande DEFCHAR de la manière suivante : Les 8 premiers correspondent aux plans 1 et 0 des 8 lignes du caractère (exactement comme en mode Lores 0). Les 8 suivants correspondent aux plans 3 et 2 des 8 lignes du caractère (on a donc toujours un 'word' par ligne de 8 pixels). Le programme Basic devient :

```
10 LORES 1:CLS
20 DEFCHAR#41,#3355,#0,#0,#0,#0,#0,#0,#0,#F,#0,#0,#0,#0,#0,#0,#0
30 DEFINK0,#0,#1F,#3E0,#3FF,#7C00,#7C1F,#7FE0,#7FFF
40 END
```

On sauve sous Lores1a.tap et on fabrique une cartouche Lores1a.swc, selon la procédure dejà décrite, sauf que cette fois, il faut impérativement utiliser la Rom Super-Oric v1.1 de nom ORIC2.OBJ de Fabrice, car le mode Lores 1 n'est pas implémenté dans les versions précédentes. Voici ce que ça donne avec Snes9x :

Hé ! Je veux 16 couleurs

Qu'à cela ne tienne, définissons 8 couleurs supplémentaires et attribuons les aux couleurs n°8 à 15 de la palette. Prenons tout simplement 8 des couleurs définies dans le Ceo-Mag n°179, page 16. Reprenons le caractère 'A' précédemment redéfini en 8 couleurs et ajoutons une deuxième ligne où chacun des 8 pixels sera affiché avec une des 8 nouvelles couleurs. Le tableau ci-dessous représente la répartition des couleurs n°8 à 15 :

Lignes	Couleurs							
0	0000	0001	0010	0011	0100	0101	0110	0111
1	1000	1001	1010	1011	1100	1101	1110	1111
2	0000	0000	etc					

Nous décomposons les bits de manière à regrouper les plans 3, 2 1 et 0. Cela donne :

Ligne	Plan 3 = Les bits b3								Plan 2 = Les bits b2								Plan 1 = Les bits b1								Plan 0 = Les bits b0							
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	1	1	0	0	1	1	0	1	0	1	0	1	0	1
1	1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	0	0	1	1	0	0	1	1	0	1	0	1	0	1	0	1
2	0000 0000 0000 etc.																															

Ces valeurs binaires correspondant à la deuxième ligne peuvent se 'lire' ainsi : #FF0F (pour les plans 3 et 2) et #3355 (pour les plans 1 et 0). Après avoir ajouté toutes ces nouvelles donnés dans le programme précédent, nous obtenons le programme Lores1b.tap suivant :

```
10 LORES 1:CLS
20 DEFCHAR#41,#3355,#3355,#0,#0,#0,#0,#0,#0,#F,#FF0F,#0,#0,#0,#0,#0,#0:'16 words pour A
30 DEFINK0,#0,#1F,#3E0,#3FF,#7C00,#7C1F,#7FE0,#7FFF:'8 couleurs Oric
40 'Noir, Rouge, Vert, Jaune, Bleu, Magenta, Cyan, Blanc
50 DEFINK8,#21F,#3F0,#3C1F,#1A9F,#7C0F,#4210,#300,#6318:'8 couleurs nouvelles
60 'Orange, Vert Pomme, Fushia, Jaune Or, Violet, Gris Fonce, Vert Fonce, Gris Clair
40 END
```

Dont nous faisons la cartouche Lores1b.swc comme d'habitude. Et voici ce que ça donne :

Le progrès par rapport à l'Oric classique est spectaculaire, non ? Nous sommes complètement affranchis des attributs 'série' et des problèmes qui en dérivent. Je suis désolé pour les abonnés au mag papier qui ne peuvent voir tous ces beaux résultats qu'en niveaux de gris. Ils trouveront dans la prochaine disquette trimestrielle de quoi voir ces couleurs pour de bon.

Les drapeaux de retournement

Nous avons vu précédemment que le ‘word’ qui défini chaque caractère dans l’écran comporte 2 bits ‘V’ et ‘H’ de retournement. Nous allons tester l’effet de ces bits en modifiant le programme précédent. Les 4 tableaux qui suivent montrent les 4 possibilités de retournement (voir ci-contre) :

V	H	P	Palette			n° du caractère (code Ascii)									
0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1

V	H	P	Palette			n° du caractère (code Ascii)									
1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	1

V	H	P	Palette			n° du caractère (code Ascii)									
0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	1

V	H	P	Palette			n° du caractère (code Ascii)									
1	1	1	0	0	0	0	0	0	1	0	0	0	0	0	1

Modifions notre programme afin que notre caractère redéfini soit affiché selon ces 4 possibilités, à l'emplacement des 4 caractères de CAPS. Cela donne le nouveau programme Lores1c.tap :

```
10 LORES 1:CLS
20 DEFCHAR#41,#3355,#3355,#0,#0,#0,#0,#0,#0,#F,#FF0F,#0,#0,#0,#0,#0,#0:'16 words pour A
30 DEFINK0,#0,#1F,#3E0,#3FF,#7C00,#7C1F,#7FE0,#7FFF:'8 couleurs Oric
40 'Noir, Rouge, Vert, Jaune, Bleu, Magenta, Cyan, Blanc
50 DEFINK8,#21F,#3F0,#3C1F,#1A9F,#7C0F,#4210,#300,#6318:'8 couleurs nouvelles
60 'Orange, Vert Pomme, Fushia, Jaune Or, Violet, Gris Fonce, Vert Fonce, Gris Clair
70 PLOT 28,0,#2041:'Sur le C de CAPS avec V=0 & H=0
72 PLOT 29,0,#A041:'Sur le C de CAPS avec V=0 & H=0
74 PLOT 30,0,#6041:'Sur le C de CAPS avec V=0 & H=0
76 PLOT 31,0,#E041:'Sur le C de CAPS avec V=0 & H=0
80 END
```

Le résultat est assez spectaculaire, comme vous pouvez le voir avec la recopie d'écran obtenues avec la cartouche Lores1c.swc. A l'emplacement du 'C' de CAPS, on retrouve le caractère 'A' précédemment redéfini, avec ses 16 couleurs sur les deux premières lignes, le pixel noir est dans le coin en haut à

gauche. A l'emplacement du 'A' de CAPS, notre caractère a subi un retournement vertical, le pixel noir est dans le coin en bas à gauche. A l'emplacement du 'P' de CAPS, on voit maintenant un retournement horizontal, le pixel noir est dans le coin en haut à droite. A l'emplacement du 'S' de CAPS, notre caractère a subi un double retournement vertical et horizontal, le pixel noir est dans le coin en bas à droite. Remarquez bien qu'un double retournement donne une rotation de 180°, comme le montre la figure de droite, qui est la même que ci-dessus, affichée avec une rotation de 180°. On voit aussi qu'un simple retournement vertical correspond à la rotation 180° d'un retournement horizontal et vice-versa.

Et la transparence ?

Nous avons déjà évoqué la question de la couleur n°0 et signalé que (bogue de la Snes ou intention délibérée ?) toutes les couleurs n°0 des 'palettes' Lores n°1 à 7 sont remplacées d'office par la couleur n°0 de la 'palette' n°0 (remplacées lors de la mise en oeuvre, pas dans la palette globale de 256 couleurs). Voyons cela de plus près, avec le programme ci-dessous, sauvé sous le nom Lores1d.tap :

```
10 LORES 1:'Pour remplacer le mode par défaut Lores 0
20 HIRES:HIRES:'Mode Hires et initialisation ecran
30 CURSET 224,8,0:DRAW 31,0,1:DRAW 0,7,1
40 DRAW -31,0,1:DRAW 0,-7,1:FILL 230,10,1
50 GOSUB 1000:'Pour verifier
60 DEFCHAR#41,#3355,#3355,#0,#0,#0,#0,#0,#0,#F,#FF0F,#0,#0,#0,#0,#0,#0:'16 words pour A
70 DEFINK0,#0,#1F,#3E0,#3FF,#7C00,#7C1F,#7FE0,#7FFF:'8 couleurs Oric
80 'Noir, Rouge, Vert, Jaune, Bleu, Magenta, Cyan, Blanc
90 DEFINK8,#21F,#3F0,#3C1F,#1A9F,#7C0F,#4210,#300,#6318:'8 couleurs nouvelles
100 'Orange, Vert Pomme, Fushia, Jaune Or, Violet, Gris Fonce, Vert Fonce, Gris Clair
110 PLOT 29,1,#41:'Priorite 0 (derriere Hires)
120 PLOT 30,1,#2041:'Priorite 1 (devant Hires)
130 PLOT 29,2,#41:'Priorite 0 (derriere Hires)
140 PLOT 30,2,#2041:'Priorite 1 (devant Hires)
150 GOSUB 1000:'Pour verifier
999 END
1000 J=0:REPEAT
1010 J=USR(0)
1020 UNTIL J<>0
1030 WAIT 50:RETURN
```

V	H	P	Palette			n° du caractère (code Ascii)									
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1

Comme vous pouvez le voir, j'ai PLOTé le word #0041 pour afficher la caractère 'A' avec la priorité zéro. J'ai obtenu cette valeur à l'aide du tableau ci-dessus. Voir page précédente pour le détail du 'word' #2041 utilisé pour la priorité à 1. Une cartouche Lores1d.swc a été fabriquée en suivant la procédure habituelle. Nous allons voir à la page suivante les deux recopies d'écran obtenues lors des pauses aménagées par les deux GOSUB 1000 (GETR$ adapté au joypad).

L'écran de gauche montre la première étape de notre programme : Un rectangle de 24 x 8 pixels, tracé dans l'écran Hires et destiné à cacher ou non les caractères qui seront affichés derrière ou devant. Remarquer qu'en absence de toute définition de couleurs les couleurs par défaut (qui se limitent à deux) sont la couleur n°0 (PAPER, Noir) et la couleur n°1 (INK, Blanc). Examinons en détails le deuxième écran. Il saute tout de suite aux yeux que le 2e caractère, affiché en priorité 0 est invisible. Essayons de comprendre tout ce qu'on voit. Pour cela, il faut connaître les choix qui ont été faits par Fabrice dans son Basic 1.1 Super-Oric.

Veuillez m'excuser si je suis un peu long dans mes explications. Sans entrer dans les détails des modes d'affichages de la Snes, sachez qu'au cœur du système d'affichage se trouve l'écran Hires. Devant ce plan se trouve l'écran Text (Lores 0 ou Lores1) qui par défaut est en priorité 1. Ca veut dire que les PRINT s'affichent dans cet écran, c'est à dire devant l'écran Hires. Derrière l'écran Hires se trouve un écran Text (toujours Lores 0 ou Lores1) en priorité 0. En somme, c'est comme un sandwich à trois couches : D'avant en arrière, l'écran Text P=1, l'écran Hires et enfin l'écran Text P=0. On ne peut voir ce qui affiché dans les deuxième et troisième couches que si rien n'est affiché dans la ou les couches qui sont devant, à moins que ce soit avec une couleur transparente, la fameuse couleur n°0 (PAPER). Quand les trois couches sont 'vides', elles sont en fait remplies de pixels de la couleur n°0. Cette couleur n°0 cesse d'être transparente pour devenir visible lorsque la dernière couche est rencontrée. On ne la voit jamais autrement. Si une autre couleur est affichée dans un des écrans qui se trouvent devant, c'est

celle-ci que l'on voit et en dernier ressort, c'est l'écran Text de priorité 1 qui 'a toujours le dessus'. En d'autres termes, les couleurs (sauf PAPER) de l'écran Text de priorité 1 peuvent cacher tout ce qui se trouve derrière. L'intérêt de tout ça est de pouvoir créer l'illusion d'un décor en trois dimensions avec des éléments mobiles qui se croisent par-devant ou par-derrière.

A la lumière de ces explications, reprenons notre analyse de la recopie d'écran de droite. Pour simplifier, donnons un numéro aux 5 caractères redéfinis. Le 'A' de CAPS aura le n°1. Puis les deux caractères que nous avons PLOTés sous la ligne de CAPS : Le n°2 pour celui de gauche (cf. ligne 110 PLOT 29,1,#41:'Priorite 0 (derriere Hires)) et le n°3 pour celui de droite (cf. ligne 120 PLOT 30,1,#2041:'Priorite 1 (devant Hires)). Et enfin les deux caractères de la troisième ligne : Le n°4 pour celui de gauche (cf. ligne 130 PLOT 29,2,#41:'Priorite 0 (derriere Hires)) et le n°5 pour celui de droite (cf. ligne 140 PLOT 30,2,#2041:'Priorite 1 (devant Hires)). Essayons de voir ce qui appartient à :

1) La couche Text de priorité 1 (couche de devant) :

-Le message CAPS, y compris le 'A' redéfini. Mais aucun des pixels en couleur n°0 (noir, dans notre exemple) qui appartiennent à l'une des couches situées derrières. C'est le cas du premier pixel de la première ligne du 'A' et de tous les autres pixels qui apparaissent en noir.

-Le caractère n°3, que nous avons PLOTé en priorité 1 et qui cache une partie du rectangle rouge dessiné dans la deuxième couche (Hires). En partie seulement, car ses pixels noirs sont transparents et laissent voir le rouge. De nouveau, c'est cas du premier pixel de la première ligne et de tous les autres pixels qui apparaissent en noir. Par contre le deuxième pixel (rouge) appartient bien au caractère n°3 et non au rectangle rouge.

-Le caractère n°5, que nous avons aussi PLOTé en priorité 1. Tous ses pixels en couleur n°0 (encore le noir dans notre exemple) sont transparents et laissent voir ce qu'il y a derrière (ici, en l'occurrence, aussi du noir).

2) La couche Hires (couche intermédiaire) :

-Le rectangle rouge, sauf là où il est caché par le caractère n°3 (du moins, par ses pixels qui ne sont pas transparents).

-Le fond d'écran, de couleur n°0 (toujours le noir dans cet exemple) est transparent et laisse voir ce qu'il y a derrière (ici, en l'occurrence, aussi du noir).

3) La couche Text de priorité 0 (couche de derrière) :

-Le caractère n°2 est complètement caché par le rectangle rouge de l'écran Hires.

-Le caractère n°4 est complètement visible, car il n'y a au-dessus de lui que des pixels transparents dans les deux couches supérieures.

-Tout le noir de l'écran correspond à des pixels de couleurs n°0. La couleur n°0 est toujours transparente, sauf dans la dernière couche, où elle prend la couleur PAPER, définie pour cette couleur n°0. Un pixel vu en couleur PAPER correspond donc en fait à la superposition de trois pixels de couleur PAPER, les deux premiers étant transparent et le troisième prenant effectivement la couleur de définition n°0.

Ces explications ont été longues et redondantes, mais elles m'ont semblé nécessaires, car une fois le phénomène bien compris, on voit qu'il est très simple.

L'inversion vidéo

Dans un Oric classique, le curseur s'affiche en vidéo inverse. Vous avez sans doute remarqué que les huit couleurs 'Oric' forment 4 paires de deux couleurs qui sont l'inverse l'une de l'autre, soit Noir/Blanc, Rouge/Cyan, Vert/Magenta et Bleu/Jaune. Les caractères Oric classiques sont codés sur 7 bits, le huitième bit détermine si le caractère doit être affiché en vidéo normale (b7=0) ou inverse (b7=1).

J'ai cherché à reproduire ce système de vidéo inverse avec le Super-Oric, à vérifier si on a

toujours les même quatre couples de couleurs et ce qui se passe avec toutes les autres couleurs. Puisque les caractères Super-Oric sont codés sur 15 bits et que le seizième est inutilisé, j'ai essayé (à tout hasard) de voir si Fabrice avait utilisé un truc analogue, mais ça ne marche pas. Apparemment la Snes n'a pas été conçue pour que ce seizième bit puisse être utilisé. Autre tentative, j'ai étudié comment se passe l'inversion vidéo dans le monde PC. Les logiciels de traitement de photos sont capables d'inverser les couleurs. Par exemple, avec Photoshop on obtient une image négative avec la commande Ctrl/i. Les couleurs RVB dont codées sur 3 octets (24 bits), de #000000 (pour le noir) à #FFFFFF (pour le blanc). Il est facile de voir que l'inversion vidéo s'obtient en complétant cette valeur à #FFFFFF. Par exemple L'inverse du Rouge (#FF0000) est le Cyan (#00FFFF). Par analogie, pour les couleurs Snes, qui sont codées sur 15 bits de #0000 (pour le noir) à #7FFF (pour le blanc), on trouve facilement qu'il faut compléter à #7FFF, c'est à dire à inverser tous les bits (sauf le seizième, inutilisable et qui reste à zéro). Le tableau suivant montre ce que ça donne :

Couleurs 'Oric Classique'	Couleurs Normales		Couleurs Inversées	
	Hexadécimal	Binaire	Hexadécimal	Binaire
Noir	#0000	000 0000 0000 0000	#7FFF	111 1111 1111 1111
Rouge	#001F	000 0000 0001 1111	#7FE0	111 1111 1110 0000
Vert	#03E0	000 0011 1110 0000	#7C1F	111 1100 0001 1111
Jaune	#03FF	000 0011 1111 1111	#7C00	111 1100 0000 0000
Bleu	#7C00	111 1100 0000 0000	#03FF	000 0011 1111 1111
Magenta	#7C1F	111 1100 0001 1111	#03E0	000 0011 1110 0000
Cyan	#7FE0	111 1111 1110 0000	#001F	000 0000 0001 1111
Blanc	#7FFF	111 1111 1111 1111	#0000	000 0000 0000 0000

Une fois encore, on retrouve les 4 paires Noir/Blanc, Rouge/Cyan, Vert/Magenta et Bleu/Jaune. Voyons ça avec le petit programme Lores1e.tap :

```
10 LORES 1:'Pour remplacer le mode par défaut Lores 0
20 DEFCHAR#41,#3355,#3355,#0,#0,#0,#0,#0,#0,#F,#FF0F,#0,#0,#0,#0,#0,#0:'16 words pour A
30 DEFINK0,#0,#1F,#3E0,#3FF,#7C00,#7C1F,#7FE0,#7FFF:'8 couleurs Oric
40 'Noir, Rouge, Vert, Jaune, Bleu, Magenta, Cyan, Blanc
50 DEFINK8,#7FFF,#7FE0,#7C1F,#7C00,#3FF,#3E0,#1F,#0:'8 couleurs Oric inversees
60 'Noir, Rouge, Vert, Jaune, Bleu, Magenta, Cyan, Blanc en vidéo inverse
70 END
```

La cartouche Lores1e.swc fabriquée avec ce programme affiche le résultat suivant :

On a bien de l'encre rouge sur fond noir pour le message CAPS, à l'exception du 'A' dont la première ligne de pixels montre les 8 couleurs de l'Oric classique. La deuxième ligne montre les mêmes couleurs affichées 'en vidéo inverse'. On a une inversion de gauche à droite, le noir devient blanc etc.

Et pour les autres couleurs ?

Le dernier programme peut facilement être modifié en remplaçant les 8 couleurs de l'Oric par d'autres couleurs, par exemple celles déjà publiées dans le Ceo-Mag n°179 page 16. Le tableau de la page suivante montre les valeurs de ces couleurs pour affichage en vidéo normale et en vidéo inverse.

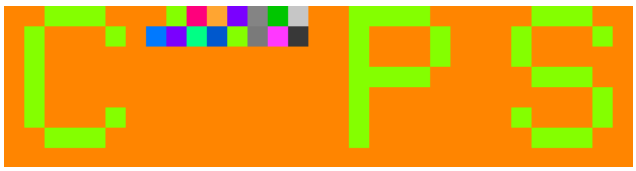
On remarque qu'à une exception près, il n'y a de 'paires' comme avec les couleurs de l'Oric classique. L'exception est la paire Vert Pomme / Violet. La chance de tomber sur une telle paire en puisant au hasard dans les 32768 couleurs possibles du Super-Oric était pourtant très faible ! En fait, les 32768 couleurs peuvent être divisées en 16384 paires puisqu'en inversant un à un les 15 bits qui définissent une couleur on tombe forcément sur une autre des 32768 couleurs. Le nouveau programme, Lores1f.tap devient alors :

Nouvelles Couleurs	Couleurs Normales		Couleurs Inversées	
	Hexadécimal	Binaire	Hexadécimal	Binaire
Orange	#021F	000 0010 0001 1111	#7DE0	111 1101 1110 0000
Vert Pomme	#03F0	000 0011 1111 0000	#7C0F	111 1100 0000 1111
Fushia	#3C1F	011 1100 0001 1111	#43E0	100 0011 1110 0000
Jaune d'Or	#1A9F	001 1010 1001 1111	#6560	110 0101 0110 0000
Violet	#7C0F	111 1100 0000 1111	#03F0	000 0011 1111 0000
Gris Foncé	#4210	100 0010 0001 0000	#3DEF	011 1101 1110 1111
Vert Foncé	#0300	000 0011 0000 0000	#7CFF	111 1100 1111 1111
Gris clair	#6318	110 0011 0001 1000	#1CE7	001 1100 1110 0111

```
10 LORES 1:'Pour remplacer le mode par défaut Lores 0
20 DEFCHAR#41,#3355,#3355,#0,#0,#0,#0,#0,#0,#F,#FF0F,#0,#0,#0,#0,#0,#0:'16 words pour A
30 DEFINK0,#21F,#3F0,#3C1F,#1A9F,#7C0F,#4210,#300,#6318:'8 couleurs nouvelles
40 'Orange, Vert Pomme, Fushia, Jaune Or, Violet, Gris Fonce, Vert Fonce, Gris Clair
50 DEFINK8,#7DE0,#7C0F,#43E0,#6560,#3F0,#3DEF,#7CFF,#1CE7:'Idem en vidéo inverse
60 'Orange, Vert Pomme, Fushia, Jaune Or, Violet, Gris Fonce, Vert Fonce, Gris Clair
70 END
```

J'ai rencontré un petit problème pour élaborer une cartouche Super-Oric à partir de ce programme. En effet lors de l'exécution, j'obtenais une 'Syntax Error' à la ligne 50. En décomposant cette ligne, j'ai trouvé que cela venait du #3DEF. Lors de la conversion du fichier .txt en fichier .tap, l'utilitaire Txt2bas tokénise le mot clef 'DEF' présent dans cette valeur ! On retrouve un #B8 au lieu de #444546 dans le fichier .tap. En attendant que Fabrice corrige Txt2bas.exe, j'ai changé le #3DEF du fichier .txt en #3DEE. Puis, lors de la modification de l'octet n°7 du fichier .tap pour le mettre en AUTO, j'en ai profité pour corriger le #3DEE en #3DEF.

Bref, voilà ce que ça donne :



La conclusion de cet exercice, est que si dans un programme on veut aussi utiliser les couleurs en vidéo inverse, il semble judicieux de ne pas choisir les couleurs de base seules, mais de penser dès le départ à des paires de couleurs, afin que toutes soient cohérentes et utilisables.

à suivre…