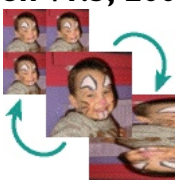
BouMaton v1.5, 2006-02-10



Frédéric Boulanger

1 Qu'est-ce ?

BouMaton est une application qui calcule deux transformations d'images :

– la transformation du **photomaton** qui répartit les pixels de l'image vers ses quatre coins, donnant ainsi un résultat qui semble contenir quatre sous-images identiques à l'original, comme dans un photomaton ;
– la transformation du **boulanger** qui étire une image horizontalement avant de la replier afin qu'elle conserve sa taille d'origine. Cette manœuvre est similaire au geste du boulanger qui pétrit sa pâte, d'où le nom.

Ces deux transformations conservent toute l'information présente dans l'image, il est donc toujours possible de retrouver l'original en appliquant les transformations inverses. De plus, ces transformations sont périodiques : après un certain nombre d'itérations, on retrouve l'image d'origine.

BouMaton peut appliquer ces transformations et leurs inverses un nombre quelconque de fois. Il calcule la période des transformations pour l'image choisie, et conserve un historique des transformations appliquées. Il peut aussi enregistrer le résultat des transformations au format JPEG si les classes Java nécessaires sont disponibles sur votre machine.

2 Le pied à l'étrier

BouMaton est une application Java. Sur la plupart des systèmes avec interface graphique, il suffit de double-cliquer le fichier BouMaton.jar pour la lancer. Dans un terminal en ligne de commande, tapez `java -jar BouMaton.jar` pour lancer BouMaton. Sous Mac OS X, utilisez plutôt l'application BouMaton (mais le fichier BouMaton.jar ou la ligne de commande fonctionnent aussi).

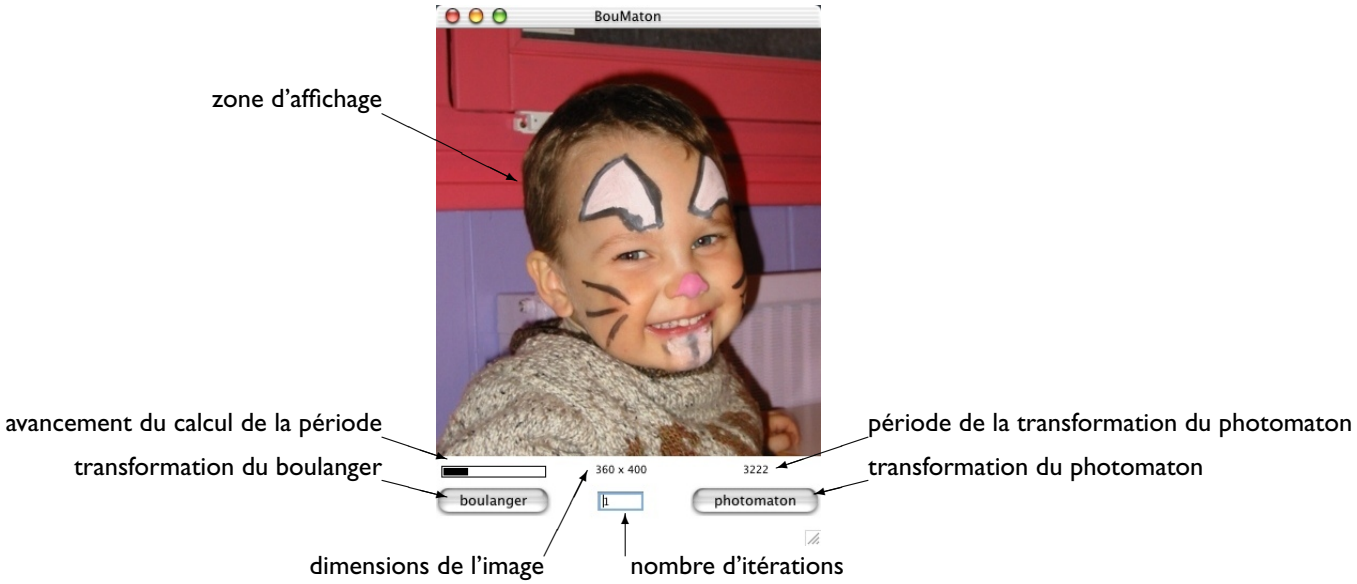
Lorsque BouMaton est lancé, choisissez l'article **Ouvrir...** du menu **Fichier** pour choisir une image. Les formats reconnus sont ceux supportés par Java 1.4 : jpeg, gif et png.

Cliquez sur l'un des boutons **boulanger** ou **photomaton**, ou sélectionnez l'article correspondant dans le menu **Transformations** pour appliquer une transformation à l'image affichée. Vous pouvez indiquer le nombre d'itérations de la transformation (négatif pour obtenir la transformation inverse) dans le champ de saisie situé entre les deux boutons.

Si vous avez lancé le calcul d'un grand nombre d'itérations d'une transformation sur une image de grande taille et trouvez que cela prend trop de temps, vous pouvez interrompre le calcul grâce à l'article **Interrompre** du menu **Transformations**.

3 Une description plus détaillée

La fenêtre principale de BouMaton contient les éléments suivants :

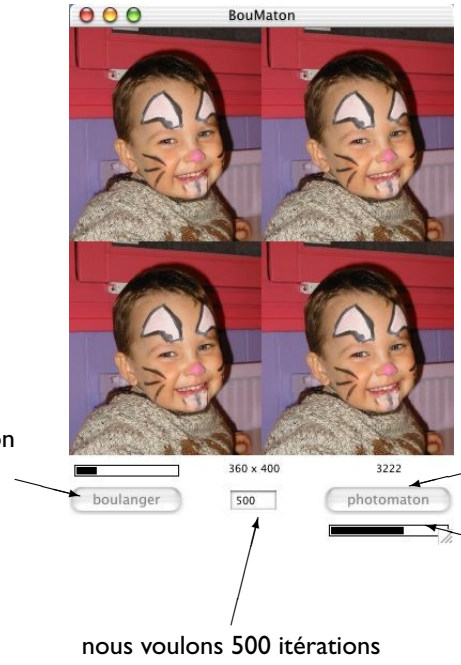


La zone d'affichage montre le résultat de l'application des transformations à l'image. La barre d'avancement du calcul de la période indique que la période de la transformation du boulanger pour cette image est en cours de calcul. Quand le calcul sera terminé, la barre d'avancement sera remplacée par le résultat.

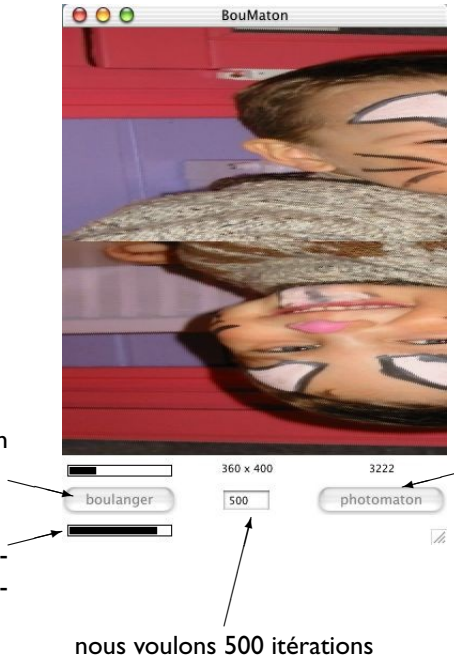
Les boutons **boulanger** et **photomaton** provoquent le calcul de la transformation correspondante, appliquée autant de fois qu'indiqué dans le champ de saisie.

Sur la droite, vous pouvez voir la période de la transformation du photomaton pour cette image (elle est beaucoup plus rapide à calculer que celle de la transformation du boulanger).

Dans l'exemple suivant, la zone d'affichage montre le résultat de la transformation du photomaton appliquée à l'image de départ, et le programme est en train de calculer 500 itérations de la transformation du photomaton :



Pour la transformation du boulanger, le comportement est similaire : dans l'exemple suivant, le résultat de la transformation du boulanger appliquée à l'image de départ est affiché, et le programme est en train de calculer 500 itérations de la transformation du boulanger :



4 Enregistrement des images

Depuis sa version 1.4, **BouMaton** requiert au moins la version 1.4 de Java, mais en contrepartie, il est capable d'enregistrer les images aux formats JPEG et PNG.

Le format JPEG donne des fichiers plus petits pour les photographies, mais il supprime des détails qui sont considérés comme mineurs lorsqu'on prend en compte la manière dont un être humain voit. Lorsqu'on enregistre une image au format JPEG, on peut choisir la quantité d'information qui sera perdue en ajustant la qualité de l'image enregistrée.Le réglage varie de 0 (forte compression, fichier de petite taille, mais qualité très faible), à 100 (compression faible, fichier de plus grande taille, mais meilleure qualité). Pour vous aider dans votre choix, BouMaton peut calculer une estimation de la taille qu'aurait le fichier avec le réglage courant si vous cliquez le bouton **Calculer la taille** de ce dialogue. Si la taille estimée est plus petite que la taille du fichier JPEG d'origine, votre réglage est sûrement trop faible. Si elle est beaucoup plus élevée, votre réglage est sûrement trop fort. Comme les transformations rendent l'image de plus en plus complexe, il est normal que la taille estimée croisse pour les résultats de transformations successives, à réglage de qualité JPEG constant.

Le format PNG n'élimine aucune information et fournit donc généralement des fichiers beaucoup plus gros que le format JPEG. Toutefois, ce format utilise un algorithme de compression sans perte qui est plus efficace que la compression JPEG sur les images qui ne contiennent que des lignes et des zones de couleur uniforme. Comme le format PNG ne supprime pas d'information, vous pouvez l'utiliser pour enregistrer des images transformées dont vous souhaitez restaurer plus tard l'aspect d'origine en appliquant les transformations inverses.

5 Historique des transformations

BouMaton conserve la trace des transformations appliquées à une image depuis son chargement. Vous pouvez afficher cet historique grâce à l'article **Afficher l'historique** du menu **Transformations** :

Les transformations successives identiques sont fusionnées dans l'historique, de sorte que deux transformations du photomaton successives donneront une seule entrée (avec un nombre d'itérations égal à 2) dans l'historique. Chaque entrée donne la nature de la transformation, le nombre d'itérations demandées, le nombre d'itérations effectivement calculées, et le temps mis pour effectuer le calcul. Le nombre d'itérations effectivement calculées peut être inférieur au nombre d'itérations demandées, car calculer 3225 itérations d'une transformation dont la période est 3222 revient exactement au même que calculer seulement 3 itérations, ce qui prend beaucoup moins de temps. Avec la même période, si vous demandez 3200 itérations, BouMaton calculera 22 itérations de la transformation inverse puisque c'est plus rapide.

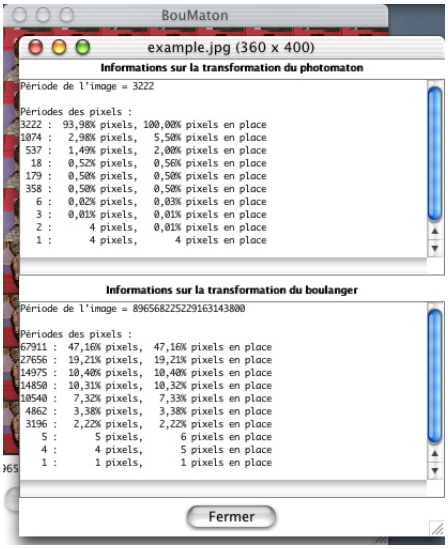
Vous pouvez toujours revenir à l'image d'origine grâce à l'article **Restaurer l'original** du menu **Fichier**. Cela efface aussi l'historique des transformations.

6 Informations sur les transformations

Lorsqu'il calcule la période des transformations, BouMaton collecte des informations sur la période de chaque pixel de l'image. Au cours de transformations successives, chaque pixel parcourt une orbite (il passe par une suite de positions successives) et peut revenir à sa position initiale bien avant que toute l'image retrouve sa forme initiale. La période de la transformation est le plus petit nombre d'itérations tel que tous les pixels de l'image ont fait un nombre entier de tours sur leur orbite.

Il peut arriver qu'après un certain nombre d'itérations, un grand nombre de pixels soient revenus à leur position initiale, sans qu'ils le soient tous. L'image de départ semble alors émerger d'une sorte de brouillard aléatoire de pixels.

En choisissant l'article **Afficher les informations** du menu **Transformations**, vous affichez les informations collectées sur les deux transformations. Les informations concernant la transformation du boulanger sont assez longues à calculer et n'apparaissent que lorsque la période de cette transformation est calculée.



Cet exemple montre les informations obtenues pour une image de 360×400 pixels (les périodes ne dépendent que de la géométrie de l'image, pas de son contenu). On peut voir que la période de la transformation du boulanger est vraiment grande, mais qu'après 67911 itérations, 47,16 % des pixels ont retrouvé leur position initiale.

7 Exemples

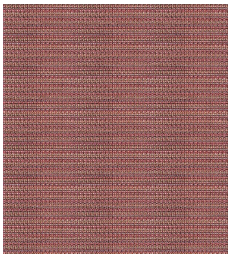
Les suites d'images suivantes montrent l'évolution d'une image pour chacune des transformations. Partant de l'original, on passe ensuite à des images où toute information semble perdue, ainsi qu'à d'autres où des motifs fantômes apparaissent, pour finalement revenir à l'image d'origine.

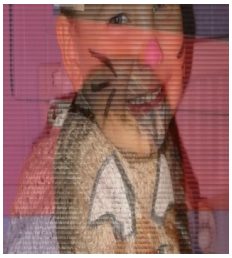
7.1 Itérations de la transformation du photomaton


image d'origine

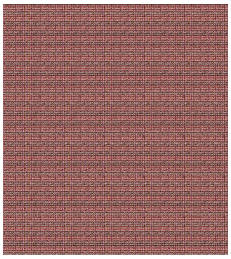

après une itération

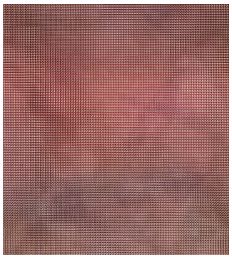

après 3 itérations

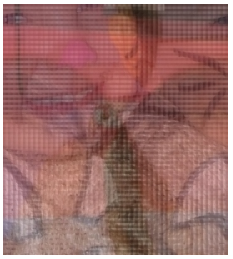

après 8 itérations


quelque chose apparaît après 179 itérations


on est près de l'original après 180 itérations


6 itérations avant le retour à l'original

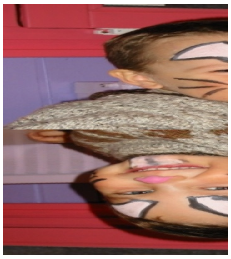

2 itérations avant le retour à l'original


1 itération avant le retour à l'original

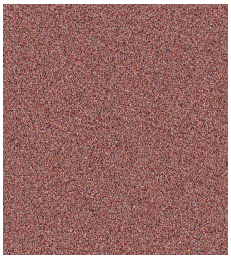

retour à l'original après 3222 itérations

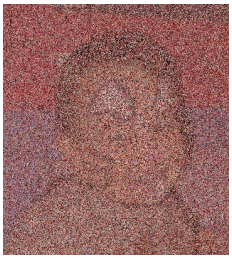
7.2 Itérations de la transformation du boulanger

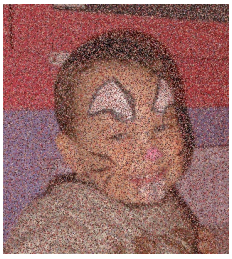

image d'origine


après une itération

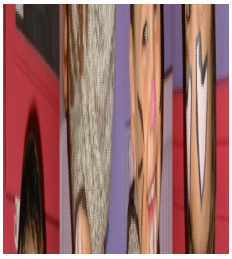

après 4 itérations


après 20 itérations


19,2% des pixels sont en place après 27656 itérations


47.2% des pixels sont en place après 67911 itérations


6 itérations avant le retour à l'original


2 itérations avant le retour à l'original


1 itération avant le retour à l'original


retour à l'original après 89656822522916314380 0 itérations

8 Licence

BouMaton est copyright Frédéric Boulanger 1997-2006, tous droits réservés.

BouMaton et son code source peuvent être redistribués librement tant qu'ils ne sont pas modifiés et que la distribution inclut cette documentation. Les traductions de la documentation et de l'application sont autorisées, mais les versions originales (voir 10 plus bas) doivent toujours faire partie de la distribution.

BouMaton est distribué « en l'état » avec absolument aucune garantie concernant sa qualité, sa précision ou son utilité pour quoi que ce soit.

Des fragments du code source de **BouMaton** peuvent être utilisés librement dans d'autres applications à condition que cette utilisation soit mentionnée dans la documentation et dans l'application elle-même, par exemple dans le dialogue « À propos de » ou dans le message de démarrage si l'application n'a pas d'interface graphique.

Le nom « **BouMaton** » ne peut pas être utilisé pour des versions modifiées de **BouMaton**, et l'auteur de la version modifiée doit indiquer clairement que je ne suis pas l'auteur principal de son application.

9 L'histoire de BouMaton

Tout a commencé avec un article de Jean-Paul Delahaye et Philippe Mathieu dans le numéro de décembre 1997 (242) de « Pour la Science ». Cet article présentait les transformations du boulanger et du photomaton, et j'ai alors commencé à coder ces transformations en C. Pour éviter d'avoir à écrire une application complète, j'ai utilisé le mécanisme de modules d'extension de GraphicConverter, une élégante application pour Mac OS qui est capable de lire et de créer presque n'importe quel type de fichier graphique (`http://www.bonnaure.com/GraphicConverter/info.html`). Cela déboucha sur la création de deux modules d'extension nommés **PhotoMaton** et **Boulanger** (`http://wwwsi.supelec.fr/fb/Developpement.html#progs`).

GraphicConverter a évolué vers Carbon lorsque Mac OS X est sorti, et au lieu de passer mes modules d'extension à Carbon, j'ai décidé d'écrire une application Java combinant les fonctionnalités des deux modules. Cela m'a permis de gérer l'historique des transformations appliquées à une image et d'afficher les informations concernant la période pour l'image complète ainsi que pour chacun de ses pixels. En utilisant la classe BigInteger, j'ai pu corriger une erreur dans le calcul de la période de la transformation du boulanger qui provoque fréquemment des débordements arithmétiques lorsqu'on utilise les `unsigned long` de C.

Le résultat est **BouMaton**, et j'espère que les utilisateurs des modules GraphicConverter l'apprécieront, et que les utilisateurs de Windows ou d'Unix profiteront de la devise « coder une fois, exécuter partout » de Java...

La version 1.0 est la première version publique de **BouMaton**.

La version 1.1 règle un problème de rafraîchissement des images.

La version 1.2 améliore la gestion des situations où la mémoire vient à manquer, et ajoute un dialogue permettant de choisir la qualité de la compression JPEG lors de l'enregistrement des images.

La version 1.3 utilise un nouvel algorithme pour calculer la période de la transformation du boulanger beaucoup plus rapidement.

La version 1.4 utilise le package **javax.imageio** et permet d'enregistrer les images au format PNG.

La version 1.5 résoud un problème de lecture des images PNG.

10 Détails techniques

Cette section expose des détails techniques de **BouMaton** et peut être ignorée par les utilisateurs qui souhaitent simplement appliquer des transformations à leurs images.

10.1 Traduire BouMaton

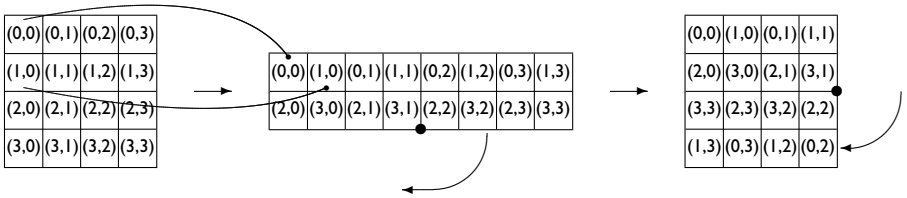
Le manuel de **BouMaton** est écrit en LaTeX. Vous pouvez vous inspirer de la version anglaise ou de la version française du manuel pour le traduire en une autre langue. Vous devez alors indiquer que la traduction est de vous.

Les chaînes de caractères utilisées dans **BouMaton** peuvent être traduites en d'autres langues. Le fichier jar contient un fichier **BouMatonStrings.properties** qui correspond à la version en anglais utilisée par défaut. Le fichier **BouMatonStrings_fr.properties** est utilisé quand la locale par défaut indique que la langue est le français. Si vous souhaitez traduire **BouMaton** en une autre langue, vous devez écrire un fichier **BouMatonStrings_<lang>.properties**, où **<lang>** est le code de cette langue pour la locale (par exemple **de** pour l'allemand, **it** pour l'italien etc). La manière la plus simple d'écrire ce fichier est de partir d'une copie d'un des fichiers existants et de traduire chaque expression située à droite du premier signe « = » de chaque ligne. Le nouveau fichier doit être ajouté à l'archive jar pour que les nouvelles chaînes de caractères soient utilisées lorsque la locale correspondante est active.

Si vous traduisez **BouMaton**, merci de me faire parvenir les fichiers correspondants afin que je les inclue dans les prochaines versions. Si une traduction du manuel accompagne les fichiers, ce sera parfait.

10.2 La transformation du boulanger

L'étirement horizontal de l'image dans la transformation du boulanger est obtenu en entrelaçant les pixels des lignes impaires avec ceux des lignes paires. Le repliement consiste à couper l'image étirée à la largeur de l'image de départ, puis à faire pivoter la partie droite autour de son coin inférieur gauche, comme indiqué ci-dessous pour une image 4×4 :



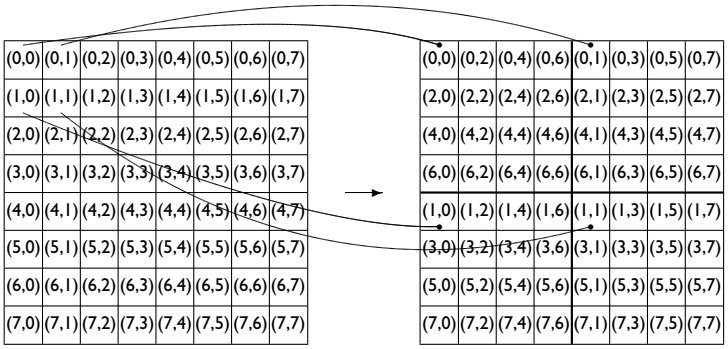
Il n'y a aucune perte d'information au cours de cette transformation, et après un certain nombre d'itérations, l'image de départ reparaît. Toutefois, je ne connais pas de formule donnant cette période pour n'importe quelle image. Pour une image de 2^m par 2^m pixels, l'image de départ reparaît après $2m + 1$ itérations. Pour une image de 2^m par 2^n pixels, il faut attendre $(2m + 1)(2n + 1) + 1$ itérations.

Pour calculer la période de la transformation pour une image donnée, **BouMaton** calcule la période $P(x, y)$ de chaque pixel de l'image (le nombre d'itérations nécessaires pour qu'il reprenne sa position initiale dans l'image). La période P pour l'image est le plus petit multiple commun à toutes les périodes $P(x, y)$ des pixels de l'image. Ce calcul peut être assez long, et **BouMaton** utilise un flot de contrôle (thread) distinct pour le calculer, afin que l'on puisse faire autre chose pendant qu'il s'exécute. La période de cette transformation peut être très grande (896568225229163143800 pour une image 360×400), et **BouMaton** utilise la classe BigInteger de Java pour calculer le PPCM des périodes des pixels.

Depuis la version 1.3, **BouMaton** utilise un nouvel algorithme pour calculer cette période bien plus rapidement qu'avant. Lorsqu'on calcule la période d'un pixel, on trouve tous les pixels qui appartiennent à son orbite. Tous ces pixels ont la même période puisqu'ils appartiennent à la même orbite, il n'est donc pas nécessaire de calculer leur période ultérieurement. Comme il y a en général peu d'orbites dans une image, le nombre de périodes $P(x, y)$ à calculer est très fortement réduit lorsqu'on utilise cette optimisation, et le calcul de la période de la transformation du boulanger est maintenant aussi rapide que celui de la transformation du photomaton.

10.3 La transformation du photomaton

Cette transformation éclate une image en ce qui semble être quatre copies identiques de l'original. Toutefois, les quatre sous-images ne sont pas exactement identiques car elles sont obtenues en prenant pour chacune d'elle un pixel de chaque carré 4×4 de l'image de départ, comme indiqué si dessous pour une image 8×8 :



Puisqu'aucune information n'est perdue au cours de cette transformation, et que le nombre de permutations des pixels de l'image est fini, l'image de départ reparaît inévitablement après un certain nombre d'itérations. Je ne connais pas de formule donnant cette période pour une image quelconque. Pour une image de 2^m par 2^n pixels, la période est le plus petit multiple commun à m et n. Pour une image 512×512, l'image reparaît après 9 itérations, alors que pour une image 128×256, il faudra attendre 56 itérations.

Toutefois, la période de la transformation du photomaton est beaucoup plus facile à calculer que celle de la transformation du boulanger. En effet, l'abscisse d'un point a une période qui ne dépend que de sa valeur, et il en est de même pour son ordonnée. En d'autres termes, tous les points situés sur une ligne verticale reviennent sur cette ligne au bout du même nombre d'itérations, et il en est de même pour les lignes horizontales. La période $P(x, y)$ pour un pixel est le plus petit multiple commun aux périodes de son abscisse et de son ordonnée. La période de l'image complète est le PPCM des $P(x, y)$ pour tous les pixels de l'image.

10.4 Optimisations

Quand la période P d'une transformation pour une image a été calculée, **BouMaton** l'utilise pour optimiser le calcul de n itérations de la transformation : il ne calcule que $n \bmod P$ itérations. Si $n \bmod P$ est plus proche de P que de 0, il calcule seulement $P - (n \bmod P)$ itérations de la transformation inverse. De plus, pour chaque pixel de l'image, **BouMaton** utilise la période de la transformation pour ce pixel de la même manière afin d'optimiser le calcul de la nouvelle position de ce pixel.

10.5 Enregistrement en JPEG des images transformées

BouMaton permet d'enregistrer les images transformées aux formats JPEG et PNG. Nous avons vu qu'aucune des transformations ne perd d'information sur les images, de sorte qu'il est toujours possible de retrouver l'original en appliquant les transformations inverses. Toutefois, comme la compression JPEG supprime des informations de l'image, une image transformée, enregistrée en JPEG, puis rechargée dans **BouMaton** ne donnera pas l'image de départ si on lui applique les transformations inverses. L'aspect général de l'image sera préservé, mais les couleurs seront délavées et l'image sera bruitée à cause de la perte des informations que la compression JPEG a éliminées parce qu'elles ne semblaient pas critiques dans l'image transformée. Ce problème disparaît avec le format PNG puisqu'il ne supprime aucune information.

Frédéric Boulanger, 2006-02-10