Sortie de G'MIC 3.0 : Une troisième dose pour un traitement efficace de vos images !

Posté par **David Tschumperlé** (**site web personnel**) le 17/12/21 à 21:35. Édité par **4 contributeurs**. Modéré par **Ysabeau**. **Licence CC By-SA.**
Étiquettes : **g'mic** , **traitement_d'images** , **greyc** **+ Étiqueter**

Une nouvelle version majeure (numérotée **3.0.0**) de ***G'MIC*** (*GREYC's Magic for Image Computing*), **cadriciel** [W] libre pour **le traitement des images** [W], a été publiée le 9 décembre 2021. Ce projet, distribué sous licence libre ***CeCILL***, est développé principalement dans l'équipe ***IMAGE*** du ***GREYC***, laboratoire de recherche en Sciences et Technologies de l'Information et de la Communication (Unité Mixte de Recherche ***CNRS*** / ***ENSICAEN*** / **Université de Caen**).

La dernière dépêche *LinuxFr.org* sur ce logiciel avait été publiée il y a plus de deux ans, en août 2019. Depuis, de nouvelles fonctionnalités ont vu le jour. Dans cette dépêche, nous proposons d'en détailler quelques-unes parmi les plus significatives.

C'est le moment de se préparer une boisson chaude, de s'installer bien confortablement dans le canapé et de se laisser conter deux ans de développement d'un projet libre dynamique !
N. D. A. : cliquez sur les images de la dépêche pour en visualiser des versions de meilleure résolution.

Sommaire

1. *G'MIC* en un peu plus de *3.0.0* mots

G'MIC est un cadriciel (*framework*) libre pour la manipulation et le traitement des **images numériques** [W].
Il propose des interfaces utilisateurs variées pour la manipulation algorithmique (automatisée ou semi-automatisée) d'images et de signaux génériques, qui vont du signal 1D simple jusqu'aux séquences d'images volumiques 3D à nombre de canaux quelconque (ce qui inclut de fait les images couleurs classiques). Le cœur de ce projet

repose sur la définition d'un langage de script spécialisé (le ***« langage G'MIC »***) et sur une implémentation libre de son interpréteur associé. Ce langage métier a été spécifiquement élaboré pour faciliter le prototypage et l'implémentation de nouveaux algorithmes et opérateurs de traitement d'images. Les utilisateurs du cadriciel peuvent ainsi appliquer des opérateurs parmi les centaines déjà prédéfinis, mais ont également la possibilité d'écrire leurs propres pipelines complexes de traitement et les rendre accessibles dans les différentes interfaces utilisateurs du projet. *G'MIC* est donc, par essence, un cadriciel ouvert, extensible et en évolution constante.

*Fig. 1.1 : logo du projet G'MIC, cadriciel libre pour le traitement d'images, et sa mascotte « Gmicky » (réalisée par **David Revoy**).*

Le projet *G'MIC* est développé depuis 2008, principalement par deux membres de l'équipe ***IMAGE*** du laboratoire ***GREYC*** : **David Tschumperlé** (Chargé de Recherche ***CNRS***, responsable de l'équipe) et **Sébastien Fourey** (Maître de Conférences ***ENSICAEN***). L'équipe *IMAGE*, l'une des six équipes du *GREYC*, est composée d'une cinquantaine de membres (chercheurs, enseignants-chercheurs, doctorants, post-doctorants et ingénieurs), tous spécialisés dans les domaines de l'algorithmique et des mathématiques du traitement d'images.

Les interfaces utilisateurs de *G'MIC* les plus visibles sont : la commande `gmic`, **exploitable en ligne de commande** (complément indispensable à ***ImageMagick*** ou ***GraphicsMagick*** pour ceux qui traitent leurs images en passant par le terminal), le service Web ***G'MIC Online***, et surtout, le greffon ***G'MIC-Qt***, disponible pour plusieurs logiciels populaires d'édition d'images numériques (libres ou propriétaires) tels que ***GIMP***, ***Krita***, ***Paint.net***, et plus récemment, ***Adobe Photoshop*** [W] ou encore ***Affinity Photo*** [W]. Ce greffon permet d'enrichir ces logiciels de plus de 570 filtres et effets différents à appliquer sur des images.

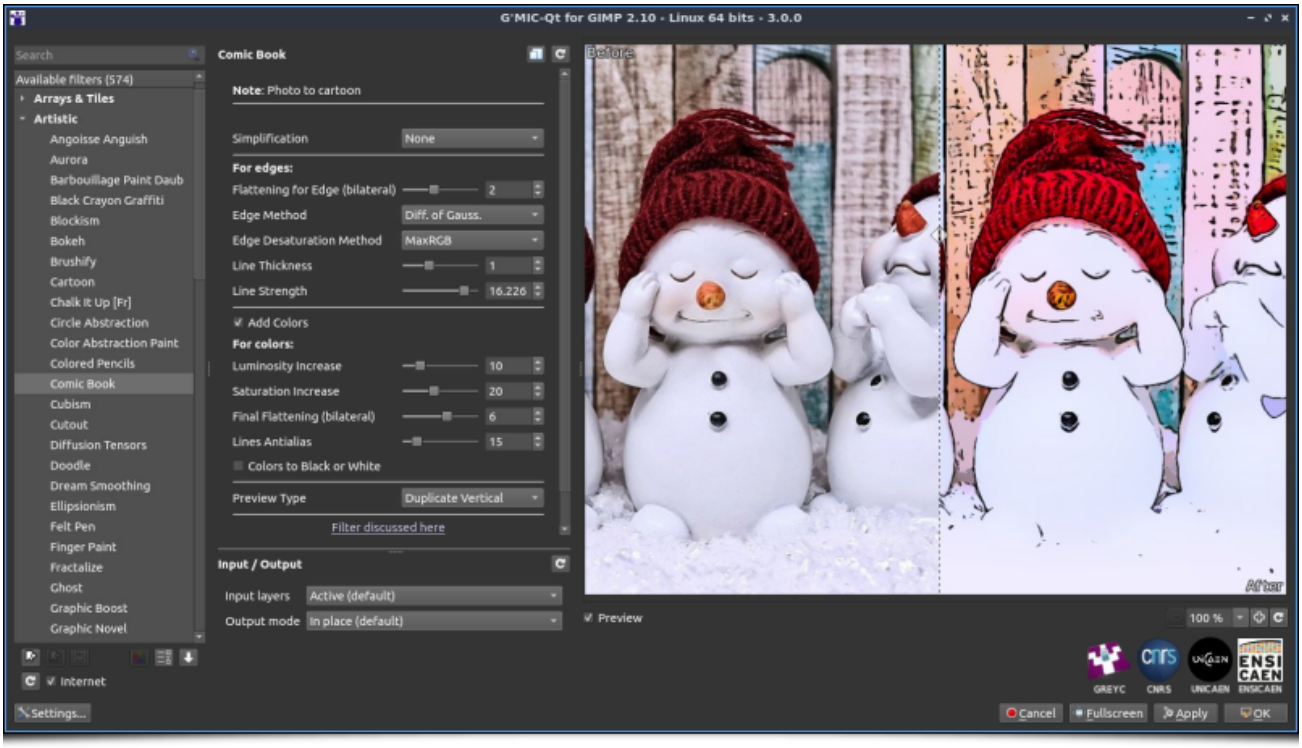


*Fig. 1.2 : Aperçu du greffon G'MIC-Qt, en version **3.0.0**, ici lancé depuis GIMP 2.10.*

Grâce à son langage de script intégré, facilitant le prototypage et l'implémentation d'algorithmes et d'opérateurs de traitement d'images, de nouveaux filtres sont régulièrement créés et ajoutés à *G'MIC*. Une grande partie de cette dépêche sera d'ailleurs consacrée à la description des nouveaux filtres et effets développés ces deux dernières années.

2. Nouveautés du greffon *G'MIC-Qt*

G'MIC-Qt étant l'interface la plus populaire et la plus utilisée du projet *G'MIC*, commençons donc par décrire les nouveautés la concernant. Nous avons sélectionné seulement un sous-ensemble des nouveaux filtres et effets ayant fait leur apparition dans le greffon, sans pouvoir être exhaustif, puisque c'est en réalité plus de cinquante nouvelles entrées qui ont été ajoutées en deux ans ! Nous avons essayé de suivre l'adage *« Une image vaut mille mots »*, en parsemant cette dépêche de nombreuses figures, illustrant les nouveautés présentées.

2.1. Effets artistiques

- Le nouveau filtre **Artistic / Paint With Brush** s'inscrit dans la grande tradition des effets cherchant à transformer une photographie en dessin ou en peinture. Son originalité réside dans le fait qu'il va repeindre entièrement l'image donnée en entrée sur une toile blanche virtuelle, en appliquant des coups de pinceau successifs à différentes échelles : d'abord avec des brosses grossières, puis avec des brosses de plus en plus fines pour recréer les moindres détails de l'image.

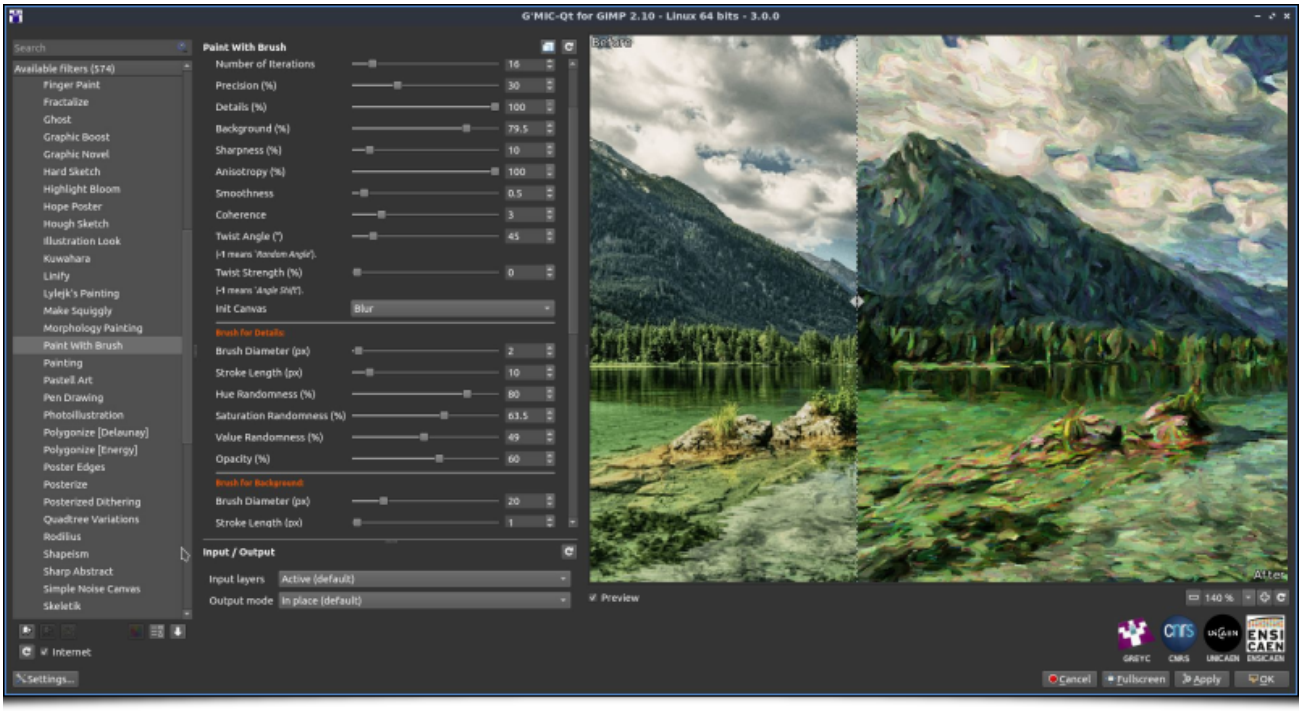


Fig.2.1.1. Le filtre « Artistic / Paint With Brush » en action.

Ici, tout est réglable : le nombre de coups de pinceau à chaque itération, la dynamique de la brosse (taille, opacité, couleur, orientation…), la précision géométrique des traits, etc. Cela en fait un filtre très polyvalent permettant de simuler des styles de peintures variés, comme on peut le voir sur l'animation ci-dessous, qui montre les résultats obtenus pour plusieurs pré-réglages proposés par ce filtre, et ce, pour une même image d'entrée.

Fig.2.1.2. Les différents préréglages proposés permettent de simuler plusieurs styles de peinture.

- Dans la même veine, le filtre **Artistic / Doodle** redessine une image sous forme de « gribouillages », à savoir un ensemble de traits noirs continus sur fond blanc, localisés principalement sur les contours des objets présents dans les images, comme illustré par le résultat ci-dessous.

Fig.2.1.3. Résultat du filtre « Artistic / Doodle ».

- Toujours dans l'optique de convertir une image en une illustration, le filtre **Artistic / Comic Book** a été récemment ajouté au greffon et permet de donner un aspect *cartoon* à vos images. C'est un filtre « communautaire », puisqu'il a été initié par **Claude Lion**, nouveau contributeur au projet, puis légèrement modifié par moi-même.

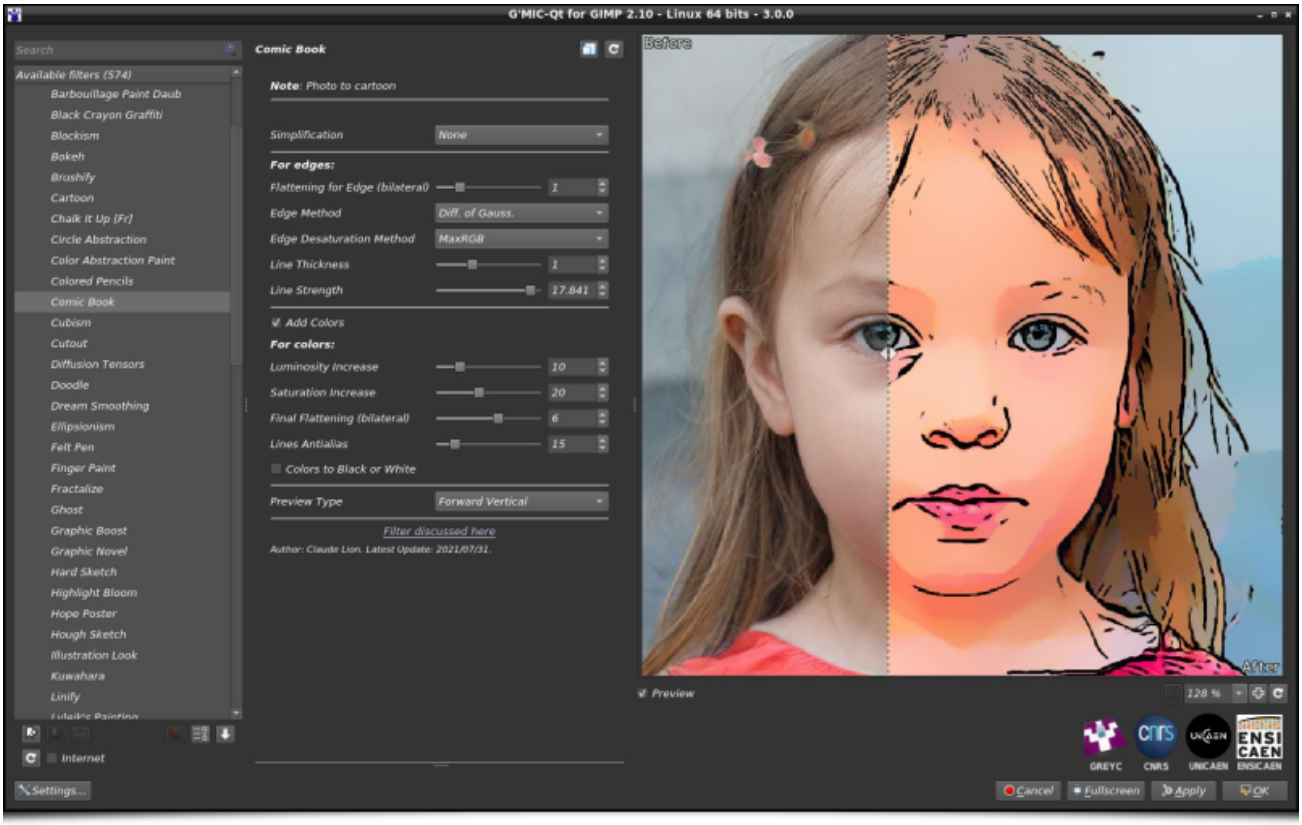


Fig.2.1.4. Le filtre « Artistic / Comic Book » en action.

- Vous souhaitez effrayer vos enfants juste avant de les coucher, pour être sûrs qu'ils passent une bonne nuit ? Le nouveau filtre **Artistic / Ghost** est fait pour vous ! Cet algorithme agglomère des segments blancs semi-transparents sur les contours des objets, ce qui a pour effet de générer le plus souvent des images assez cauchemardesques (notamment lorsqu'il est appliqué sur des portraits).

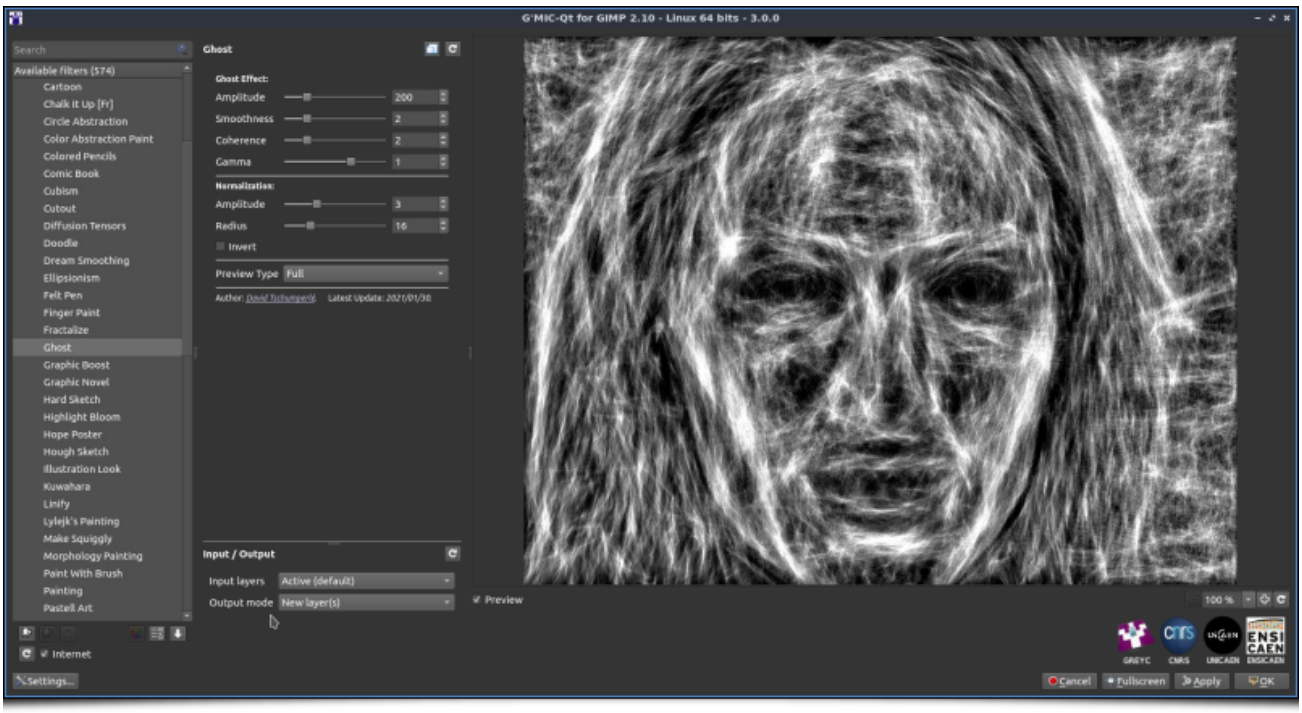


Fig.2.1.5. Le filtre « Artistic / Ghost » en action. Éloignez les enfants !

Il est alors facile, en mélangeant le résultat de cet effet avec l'image originale (par des modes de fusion de calques bien choisis) de générer des animations, qui transformeront n'importe laquelle de vos images en vision d'horreur ! (ici, nous l'avons utilisé conjointement avec la commande `morph` de *G'MIC* pour le rendu de l'animation ci-dessous).

Fig.2.1.6. Variation animée autour du filtre « Artistic / Ghost ».

- Enfin, mentionnons deux petites améliorations sur des filtres artistiques déjà existants. Tout d'abord, le filtre **Frames / Droste** se dote de points manipulables sur la fenêtre de prévisualisation, permettant de régler plus aisément la zone où va se positionner la répétition de l'image d'entrée. Ce sont les points de couleur visibles aux quatre coins du miroir, sur l'exemple ci-dessous.

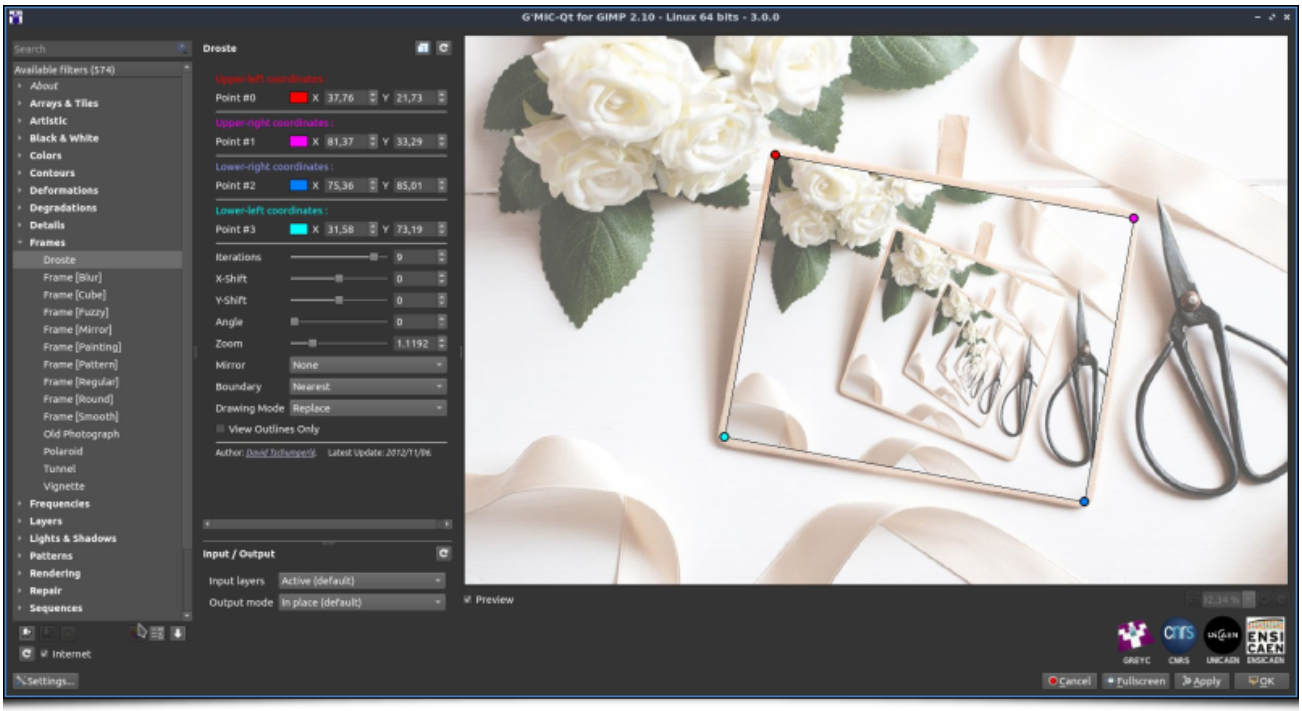


Fig.2.1.7. Ajout de points manipulables par l'utilisateur pour le réglage du filtre « Frames / Droste ».

Puis, le filtre **Artistic / Stylize** se dote de six nouveaux styles prédéfinis, élaborés par **Christine Garner**, une artiste qui avait beaucoup utilisé ce filtre pour donner un aspect texturé à ses images avant de proposer sa contribution. Ces nouveaux styles permettent par exemple de produire des images avec des textures de crayon, de craie, de pastel ou de fusain.

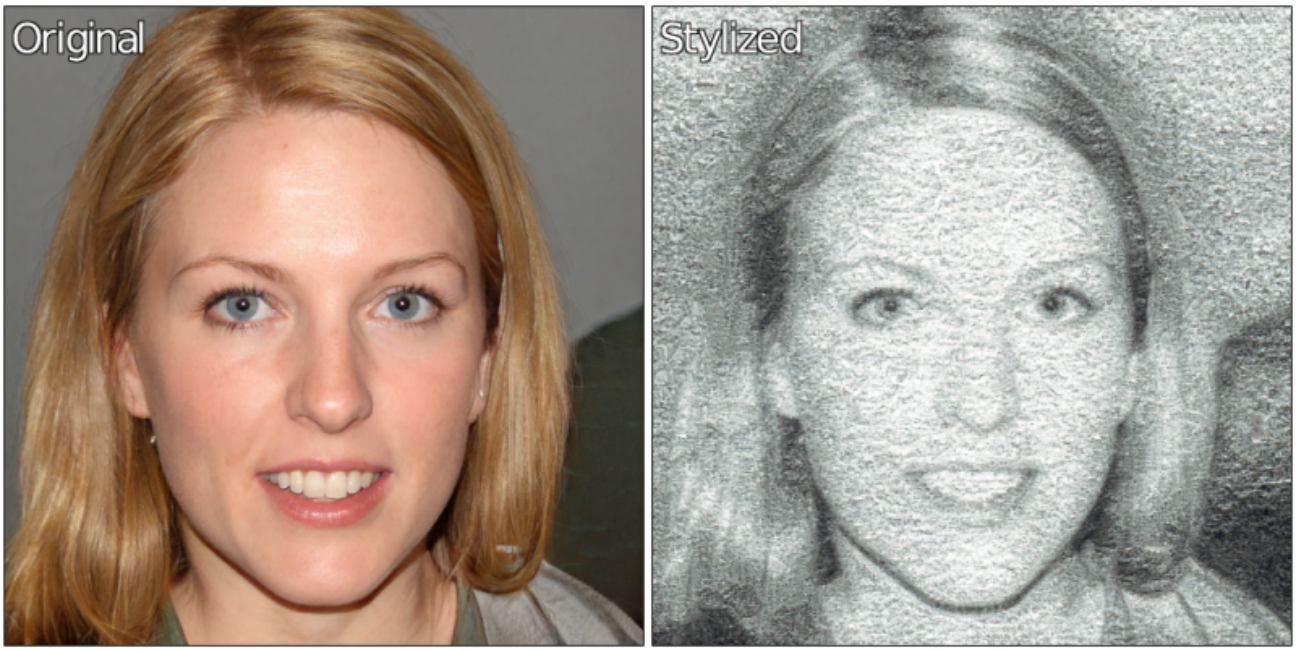


Fig.2.1.8. Un des nouveaux styles prédéfinis disponibles dans le filtre « Artistic / Stylize ».

2.2. Amélioration d'images

- Le greffon se dote encore d'un nouvel algorithme pour aider au débruitage des images, problème très classique mais ô combien difficile à résoudre, qui occupe les chercheurs en traitement d'images depuis plusieurs générations. Le nouveau filtre **Repair / Denoise** essaye de s'acquitter de cette tâche par une approche basée sur les **réseaux de neurones convolutifs** [W].

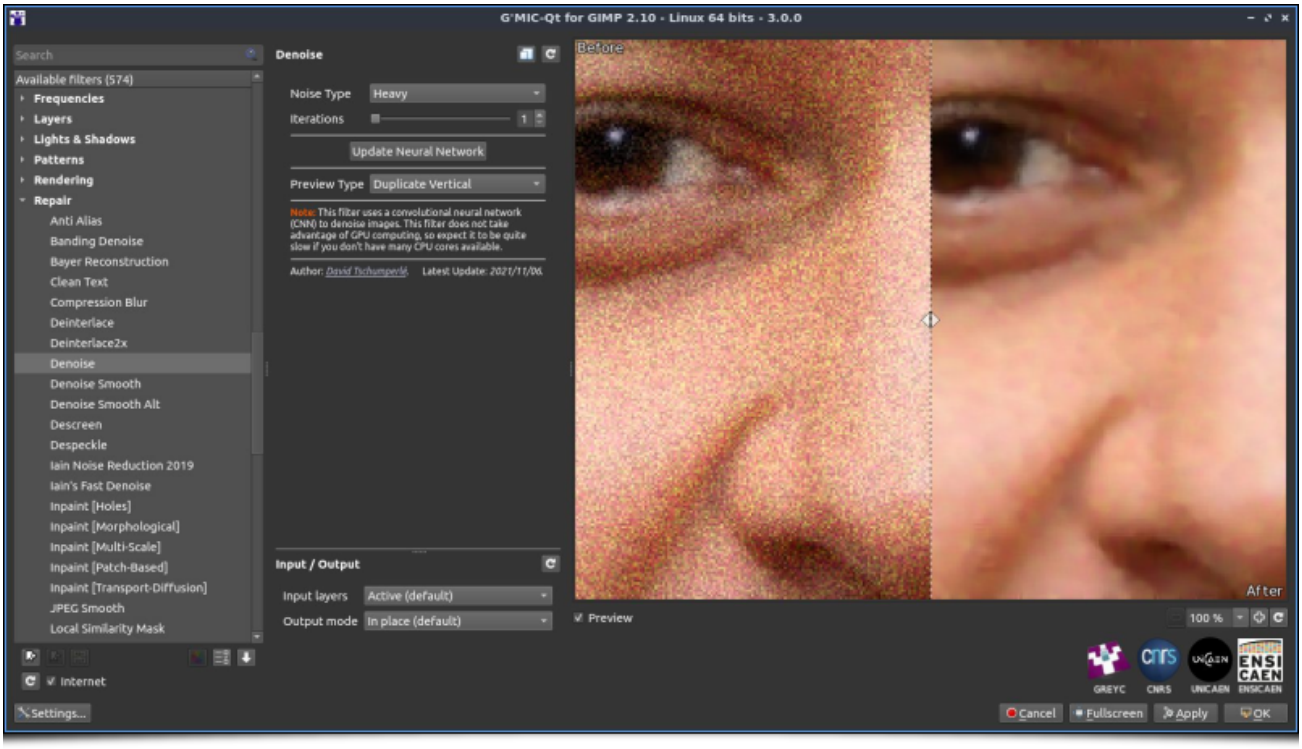


Fig.2.2.1. Le filtre « Repair / Denoise » en action.

À première vue, cette information peut sembler anodine, mais elle représente en réalité un fait marquant dans l'histoire de *G'MIC* (ainsi qu'un bon paquet de semaines de développement !). En effet, pour réaliser ce filtre, il a fallu implémenter une bibliothèque complète d'**apprentissage machine** [W], nommée `nn_lib` (pour « ***N**eural **N**etwork **Lib**rary* »). Cette bibliothèque permet aujourd'hui la construction, l'apprentissage et l'évaluation de **réseaux de neurones artificiels** [W]. C'est généralement sous le terme (un peu dévoyé) d'**« Intelligence Artificielle »** [W] que l'on désigne ce type de méthodes. Le filtre de débruitage **Repair / Denoise** est donc tout simplement le premier filtre de *G'MIC* basé sur cette nouvelle bibliothèque d'apprentissage machine !

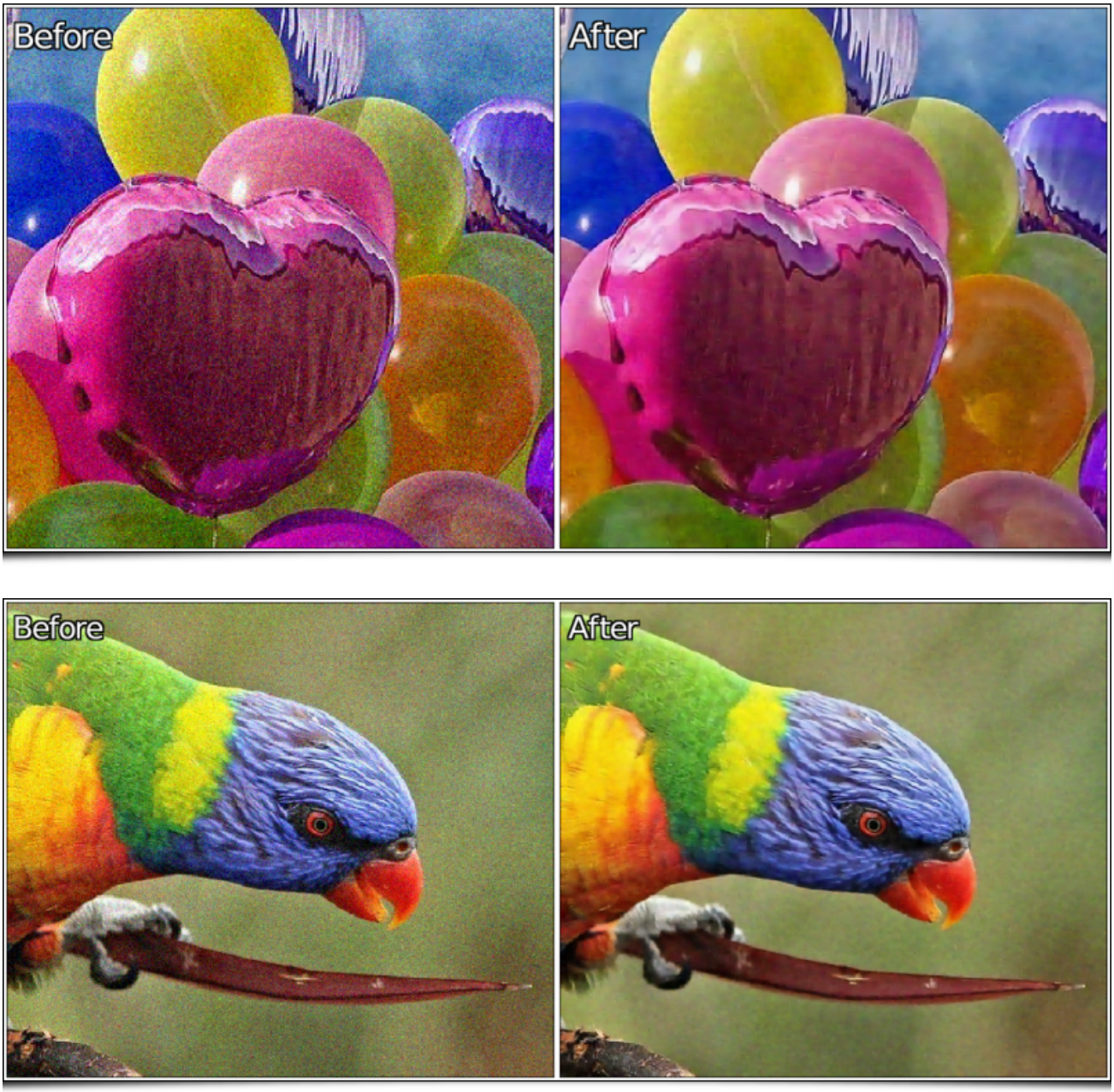


Fig.2.2.2. Deux exemples de débruitage d'images, obtenus avec le filtre « Repair / Denoise ».

Avec l'apparition de `nn_lib`, nous espérons proposer dans le futur de plus en plus de filtres basés sur les réseaux de neurones, pour la retouche et la génération d'images. De belles choses à venir en perspective !

- Un autre filtre d'amélioration d'images, plus spécialisé celui-ci, a été intégré au greffon *G'MIC-Qt*. Il s'agit du filtre **Repair / Unpurple** dont le rôle est de supprimer les **franges violettes** [W] qui peuvent apparaître dans les photographies. L'algorithme utilisé ici est un portage direct en langage *G'MIC* de l'algorithme **Unpurple** de **Martin Jambon**, portage réalisé par **Stanislav Paskalev**. On peut voir ci-dessous l'effet de ce filtre sur une portion d'image comportant ces fâcheuses franges violettes.

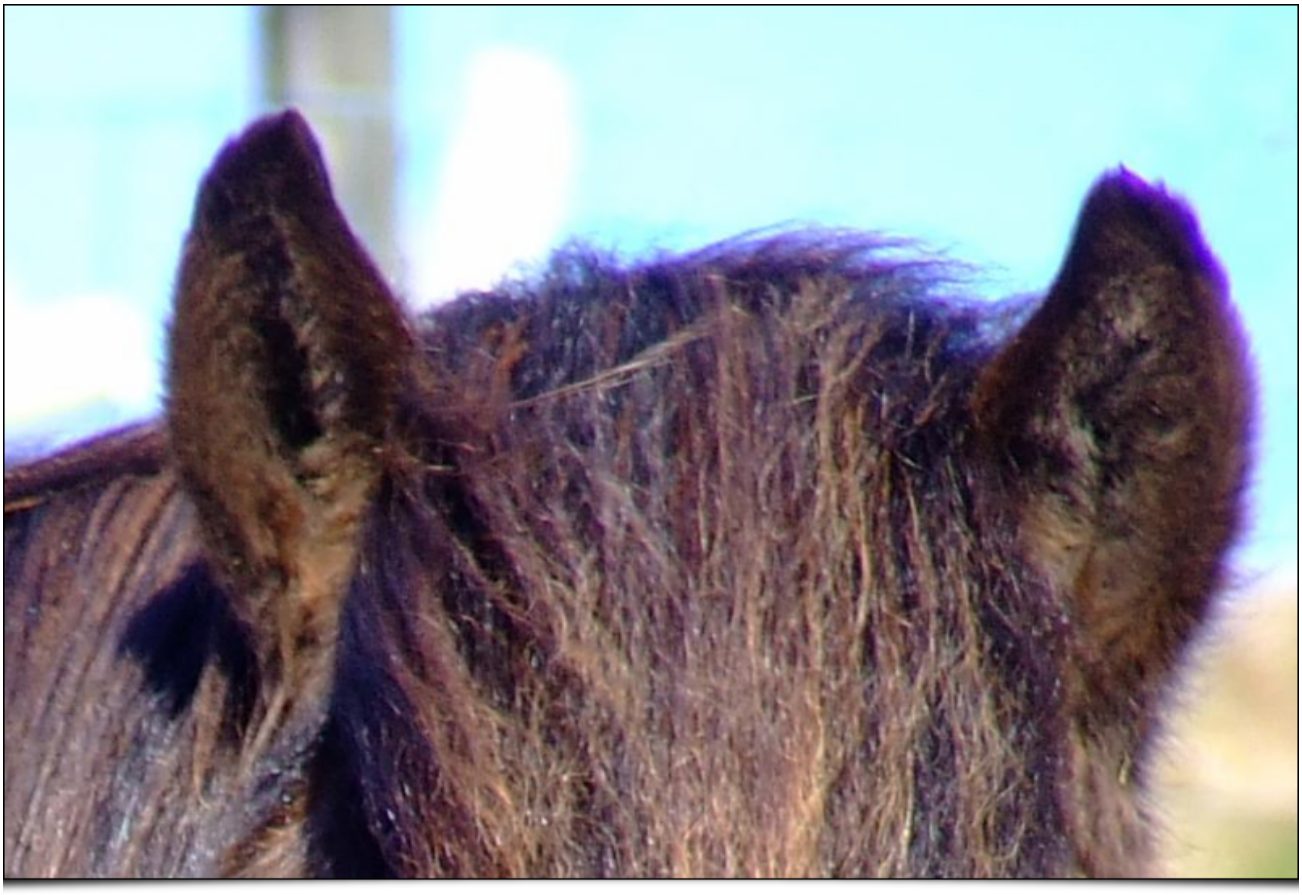


Fig.2.2.3. Portion de photographie couleur comportant des franges violettes.

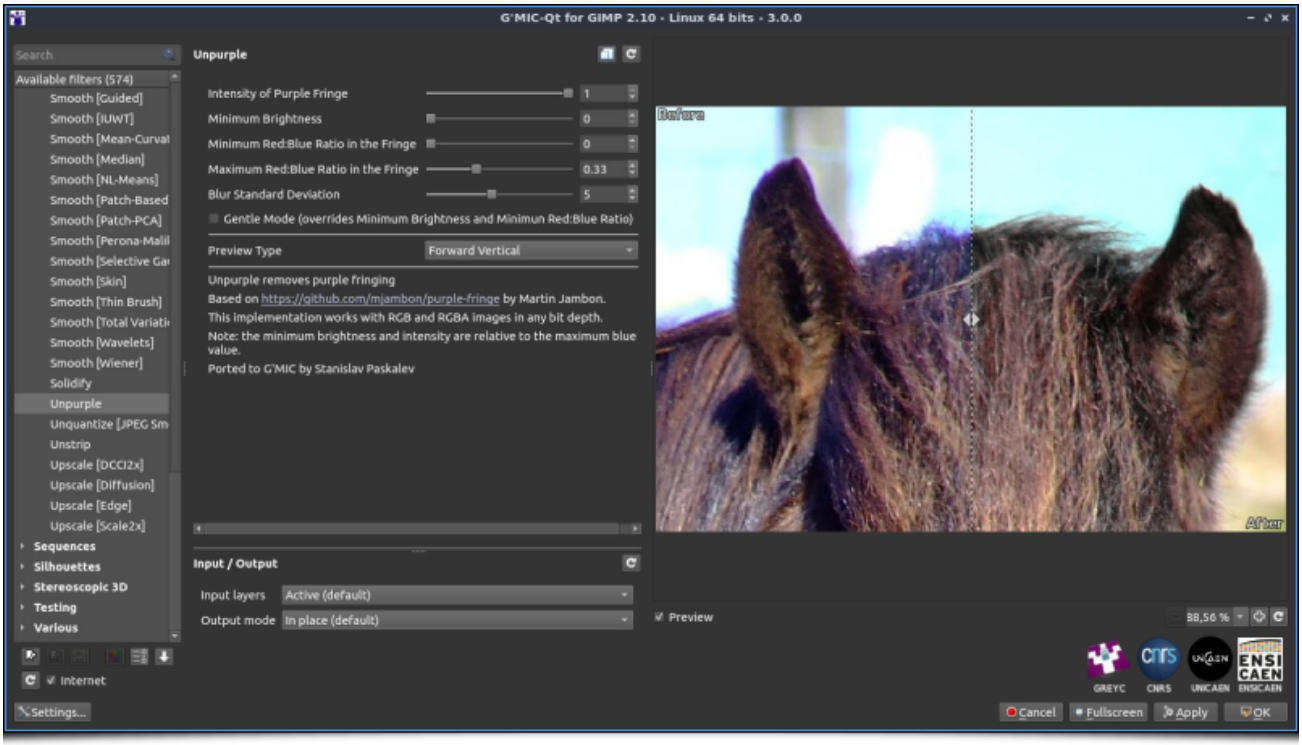


Fig.2.2.4. Le filtre « Repair / Unpurple » en action.

Fig.2.2.5. Résultat du filtre : les franges violettes ont disparu.

- Une envie soudaine de rehausser les détails dans vos images ? Le nouveau filtre **Details / Sharpen [Multiscale]** pourrait vous intéresser. Il vient compléter l'armada des algorithmes d'amélioration de netteté existants dans *G'MIC*. Il se base sur un algorithme de rehaussement multi-échelle pour faire ressortir les détails de différentes tailles présents dans les images.

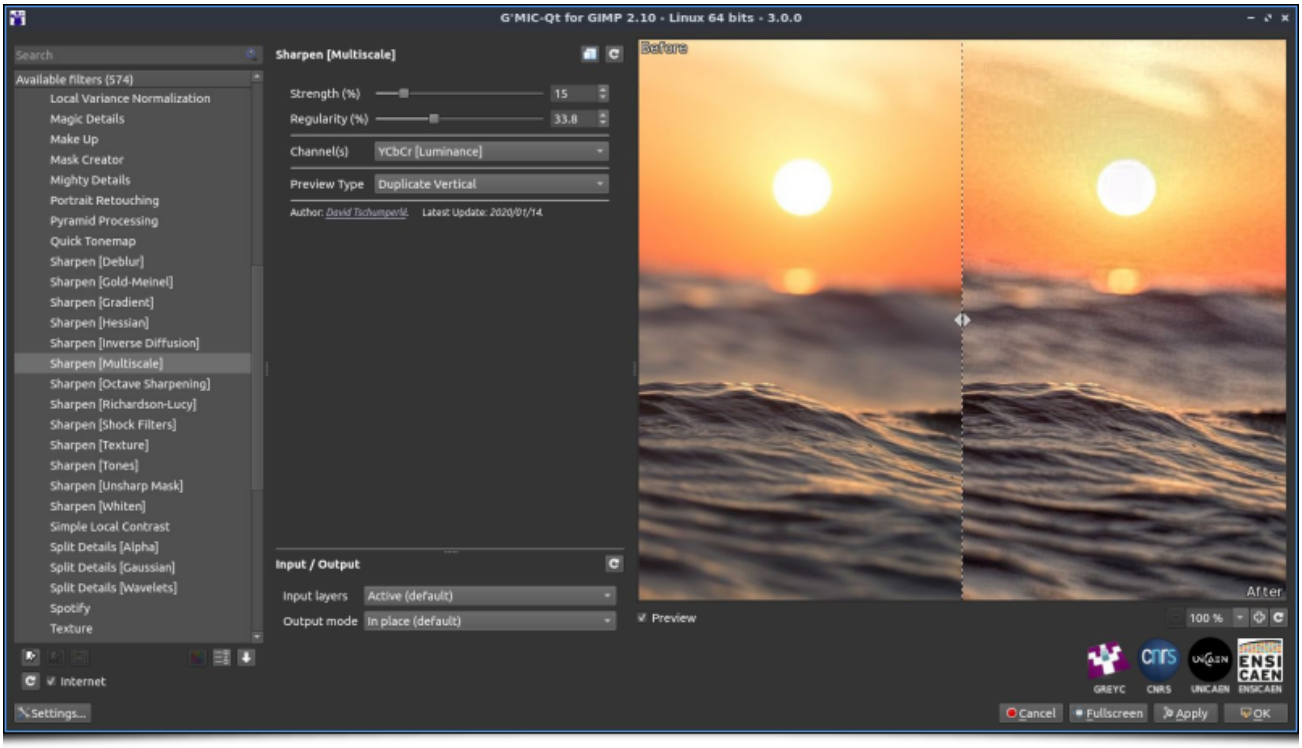


Fig.2.2.6. Le filtre « Details / Sharpen [Multiscale] » en action.

Fig.2.2.7. Résultat de l'application du filtre sur une photographie couleur.

2.3. Retouche des couleurs

Pour la retouche couleur des images, deux nouveaux filtres particulièrement intéressants font leur apparition dans *G'MIC-Qt* :

Premièrement, le filtre **Colors / Tune HSV Colors**, qui permet à l'utilisateur de définir, de façon très fine, une fonction de transformation opérant dans l'espace couleur ***HSV* (Teinte-Saturation-Valeur)**[W].
Ce filtre fonctionne de la manière suivante :

- Tout d'abord, le filtre va déterminer, dans l'image d'entrée, les couleurs existantes les plus proches des couleurs pures suivantes : le rouge (dénoté *R*), le jaune (*Y*), le vert (*G*), le cyan (*C*), le bleu (*B*) et le magenta (*M*). À ceci s'ajoute la couleur la plus sombre (*D*), la plus claire (*L*), ainsi que la couleur moyenne (*A*). Il arrive d'ailleurs que l'image d'entrée ne contienne pas du tout l'une ou l'autre de ces couleurs, et dans ce cas, la couleur extraite la plus proche peut être perceptivement assez éloignée de la couleur pure idéale. C'est le cas par exemple avec l'image du papillon illustrée ci-dessous : cette image ne contient aucun pixel dans les tons bleus.

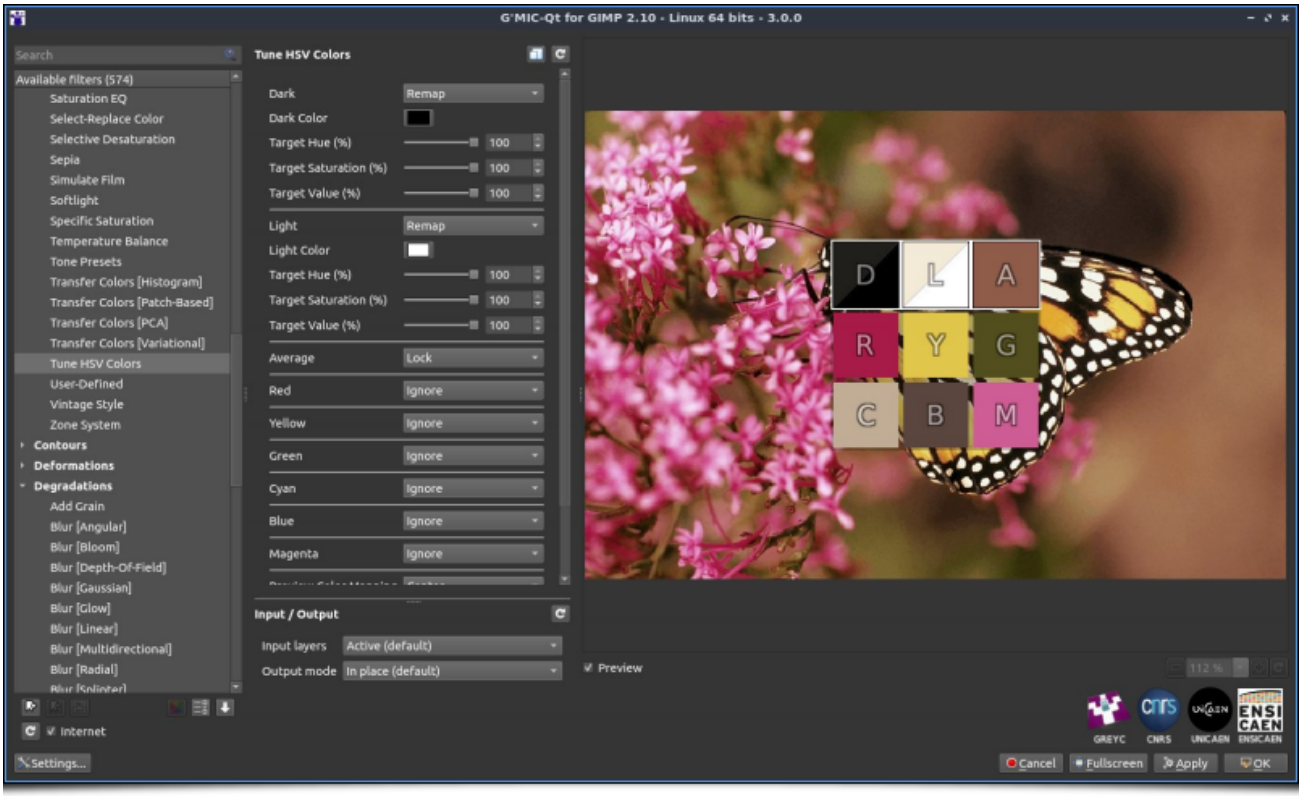


Fig.2.3.1. Le filtre « Colors / Tune HSV Colors » extrait tout d'abord des couleurs de base de l'image.

- Ces neuf couleurs extraites de l'image vont ensuite servir de base pour la définition d'une transformation couleur personnalisée. On va pouvoir en effet modifier chacune de ces couleurs indépendamment dans l'espace *HSV*, en rendant la couleur plus ou moins claire, plus ou moins saturée, ou en changeant carrément sa teinte. C'est ce que nous faisons dans l'exemple ci-dessous où l'on modifie les couleurs les plus rouges et magentas pour les rendre vertes et changer ainsi la couleur de la fleur sur laquelle est posé le papillon (dont on a également modifié légèrement la teinte en passant).

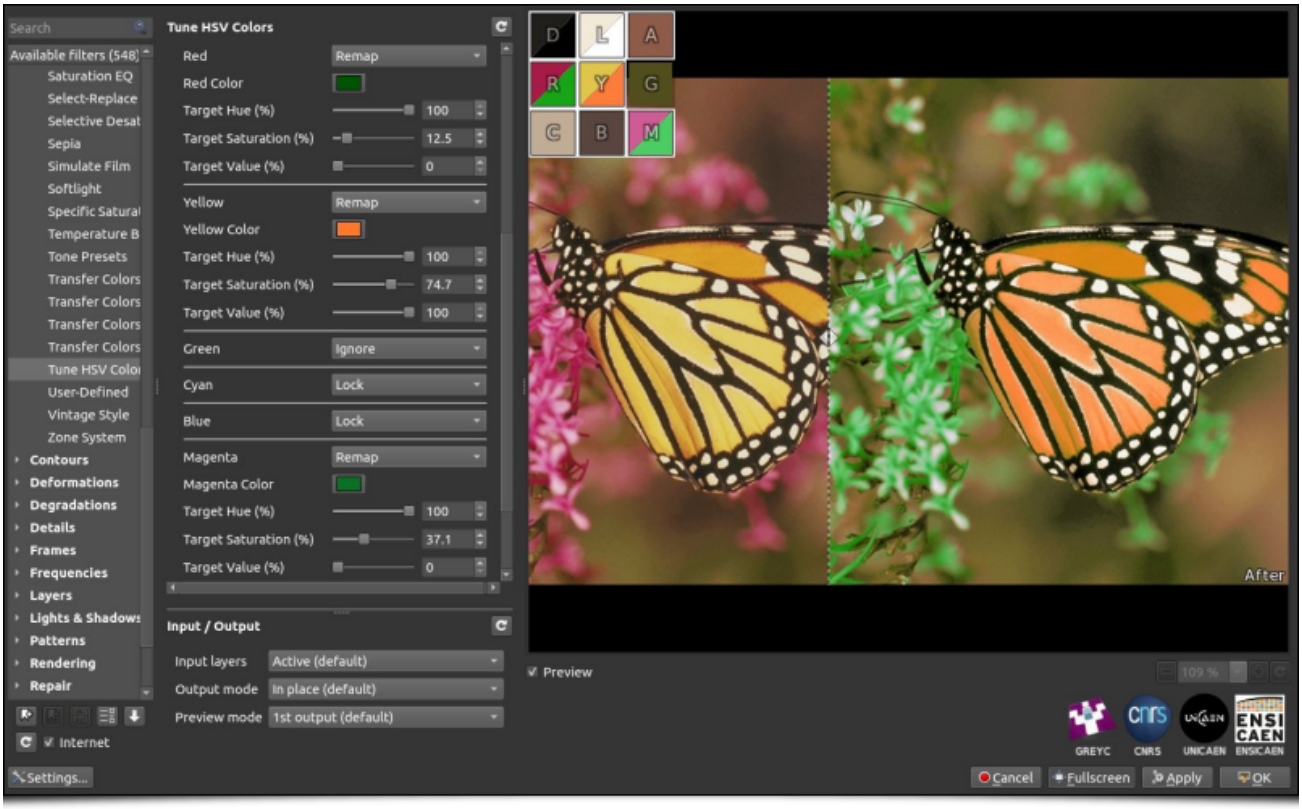


Fig.2.3.2. L'utilisateur peut agir sur chacune des 9 « couleurs-clés » individuellement pour définir une transformation couleur personnalisée.

D’un point de vue technique, cette redéfinition des « couleurs-clés » entraîne dans un premier temps la construction d’une ***CLUT (Color LUT)*** interpolée dense en 3D, qui sert à modifier dans un second temps l’image d’entrée suivant les désidératas de l’utilisateur, comme illustré ci-dessous dans notre exemple avec le papillon :

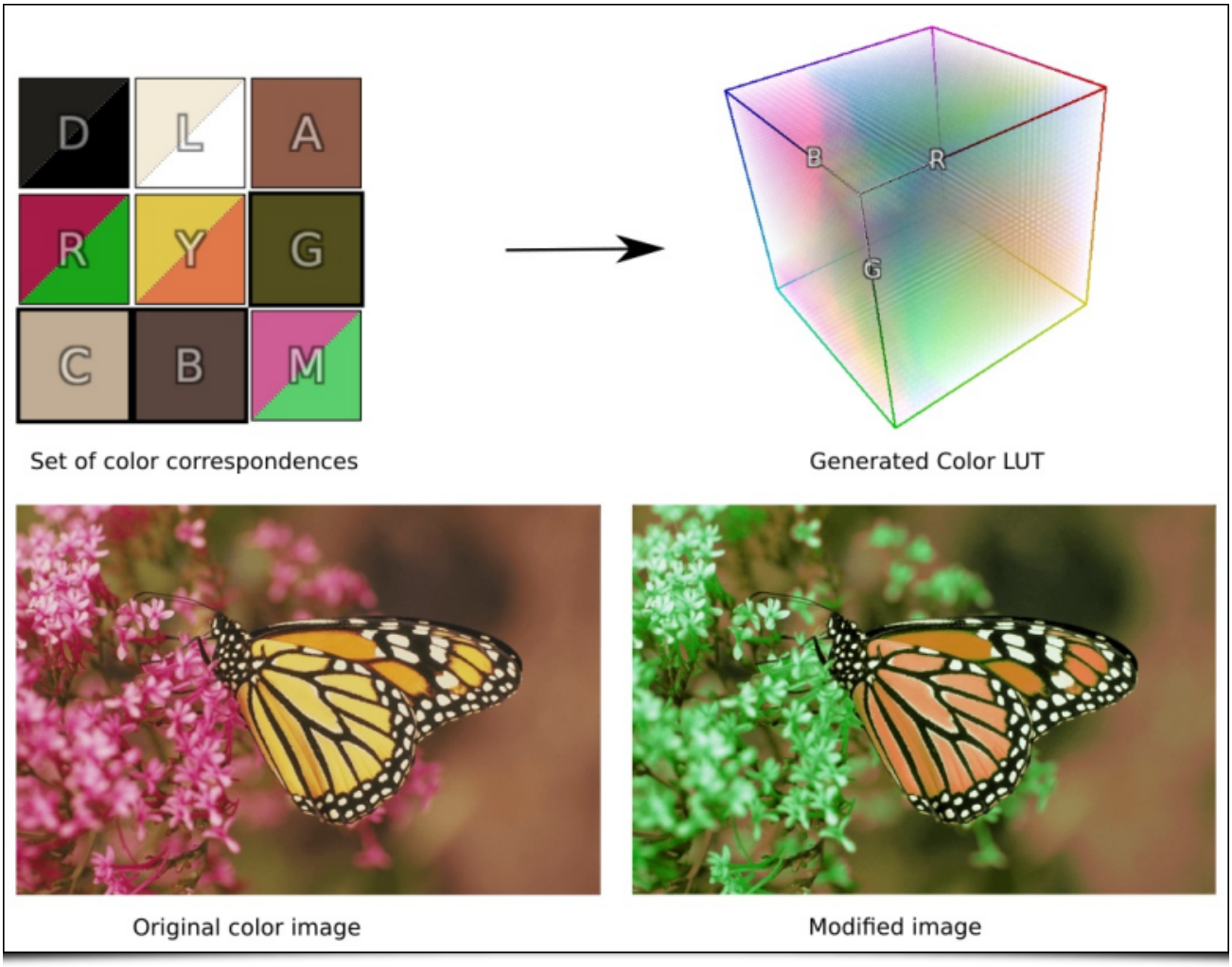


Fig.2.3.3. En interne, G’MIC génère une CLUT 3D modélisant la transformation colorimétrique souhaitée.

Les cas d’utilisation de ce filtre sont assez nombreux. Que ce soit pour une retouche locale de couleur (transformation d’une couleur en une autre), ou une retouche colorimétrique plus globale, ce filtre trouvera facilement sa place dans la boîte à outils du bidouilleur de couleurs. Dans la figure ci-dessous, nous utilisons par exemple ce filtre pour définir une ambiance couleur crépusculaire.

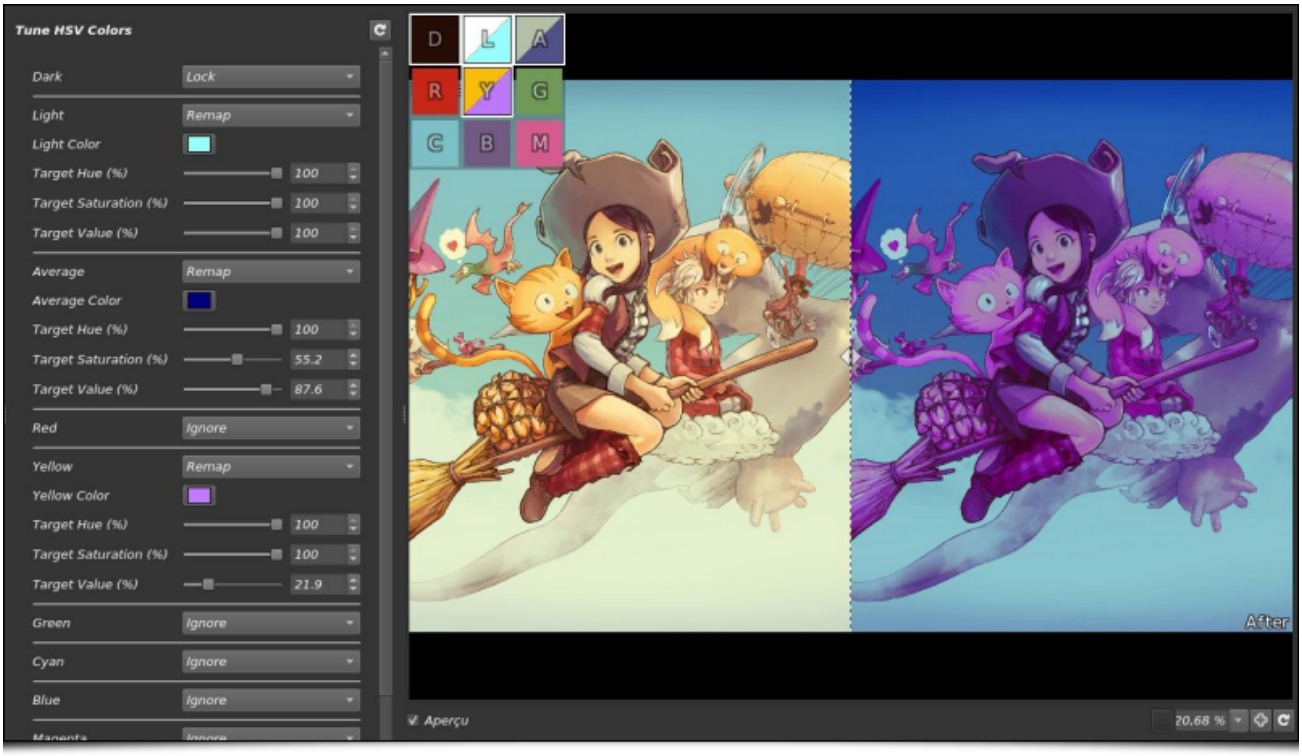


*Fig.2.3.4. Transformation couleur globale appliquée sur une image de **Pepper & Carrot** de l'artiste **David Revoy**.*

Une vidéo de démonstration de ce filtre est visible sur la chaîne Youtube de Pierre « El Lobo », qui poste régulièrement des tutoriels en français sur l'utilisation de *GIMP*. Cliquez sur l'image ci-dessous pour y accéder. Plus de détails sur l'utilisation de ce filtre peuvent également être trouvés sur le **forum de G'MIC** (en anglais).

Le deuxième filtre de retouche colorimétrique notable se nomme **Colors / Transfer Colors [PCA]**. Il permet de transférer les couleurs d'une image de référence vers une image que l'on souhaite modifier. Techniquement, ce transfert est réalisé en imposant la matrice de covariance de l'image à modifier pour correspondre à celle de l'image de référence.

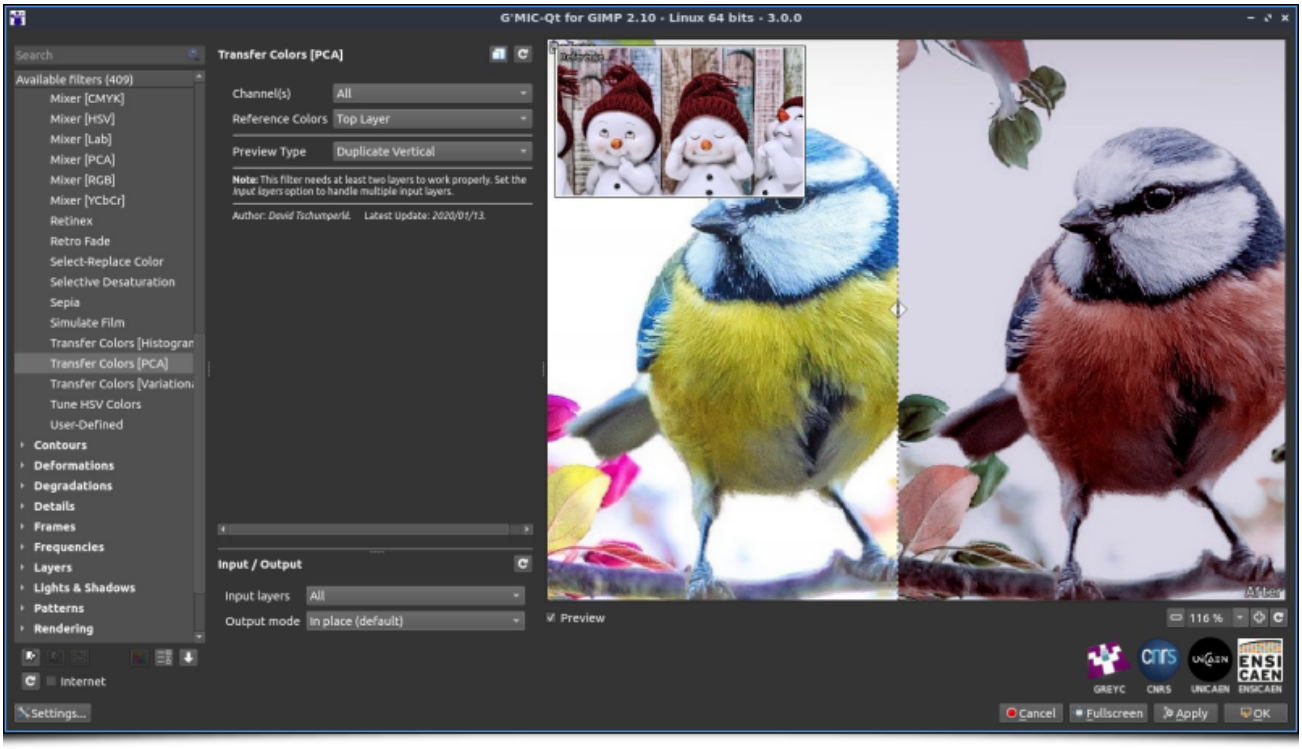


Fig.2.3.5. Le filtre « Colors / Transfer Colors [PCA] » en action.

Fig.2.3.6. Résultat du transfert de couleur. Les couleurs de l'image de référence (bonhommes de neige) ont été transférées à l'image de l'oiseau.

À noter que l'algorithme de transfert de couleurs sous-jacent n'est pas utilisé uniquement pour ce filtre, mais a également été intégré comme une sous-partie de traitements plus complexes (par exemple, le filtre **Artistic / Stylize** évoqué précédemment).

2.4. Déformations et dégradations

Les amateurs de déformations d'images et de Glitch art [W] peuvent aussi trouver quelque intérêt dans cette nouvelle version *3.0* de *G'MIC*.

- D'abord, avec le nouveau filtre **Deformations / Breaks**, qui introduit aléatoirement des discontinuités triangulaires dans les images. Ce filtre provient initialement d'une demande de l'artiste David Revoy pour générer une déformation proche de ce que donnait un autre filtre plus ancien, **Deformations / Crease**,

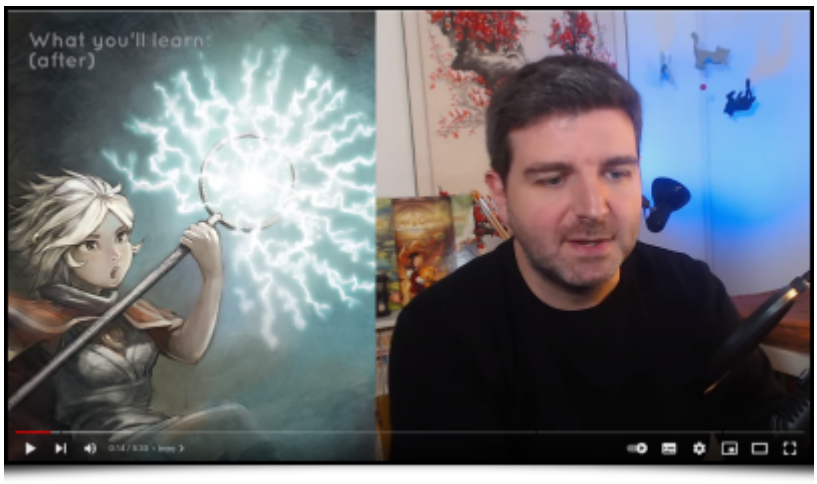
mais avec des coupures plus nettes. Deux nouveaux modes de déformations ont ainsi été implémentés pour cet effet (modes *Flat* et *Relief*) avec le résultat suivant :

Fig.2.4.1. Application des deux modes différents du filtre « Deformations / Breaks » sur une image couleur.

David Revoy nous montre ci-dessous une application de ce filtre de « cassures » d'images dans le domaine de l'illustration, pour un rendu d'éclairs magiques du plus bel effet !

Fig.2.4.2. Utilisation du filtre « Deformations / Breaks » pour perturber la forme d'éclairs magiques.

Une vidéo plus détaillée d'utilisation de ce type de filtres par David Revoy est visible en suivant le lien suivant :



- Autre effet très *glitchesque* : le filtre **Degradations / Rebuild From Blocs** s'évertue à reconstruire une image à partir de l'ensemble de ses blocs, en s'imposant néanmoins de ne pas les repositionner à leur place initiale, mais en s'autorisant à réutiliser un même bloc, si besoin. Comme on s'en doute, le résultat ressemble vaguement à l'image initiale mais avec un effet de bloc très prononcé.

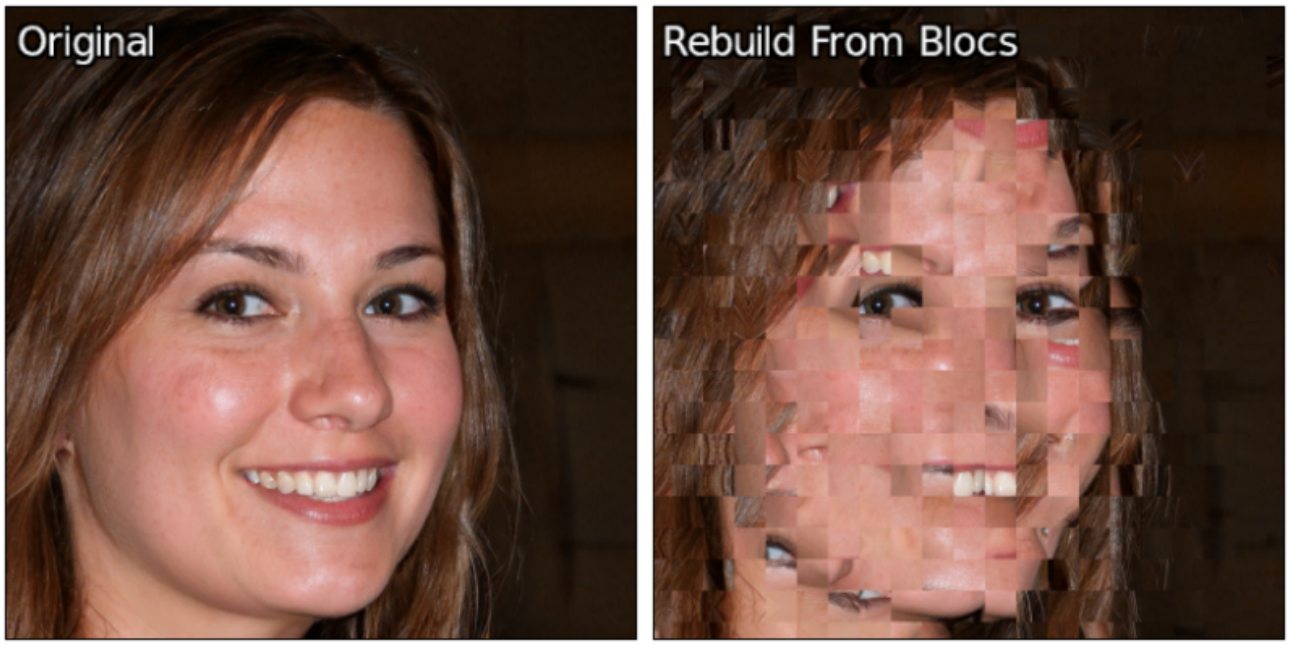


Fig.2.4.3. Le filtre « Degradations / Rebuild From Blocs » va changer les positions des blocs de l'image.

Il est amusant de remarquer que dans l'exemple ci-dessus, les blocs correspondant aux yeux de la demoiselle ont simplement été inversés dans les deux images !

- Enfin, le filtre **Degradations / Blur [Multidirectional]** applique, comme son nom l'indique, un effet de flou multi-directionnel sur l'image. Il est possible d'ajuster le nombre de directions de flous simultanées, ainsi que d'éventuellement rehausser les contrastes locaux dans l'image obtenue.

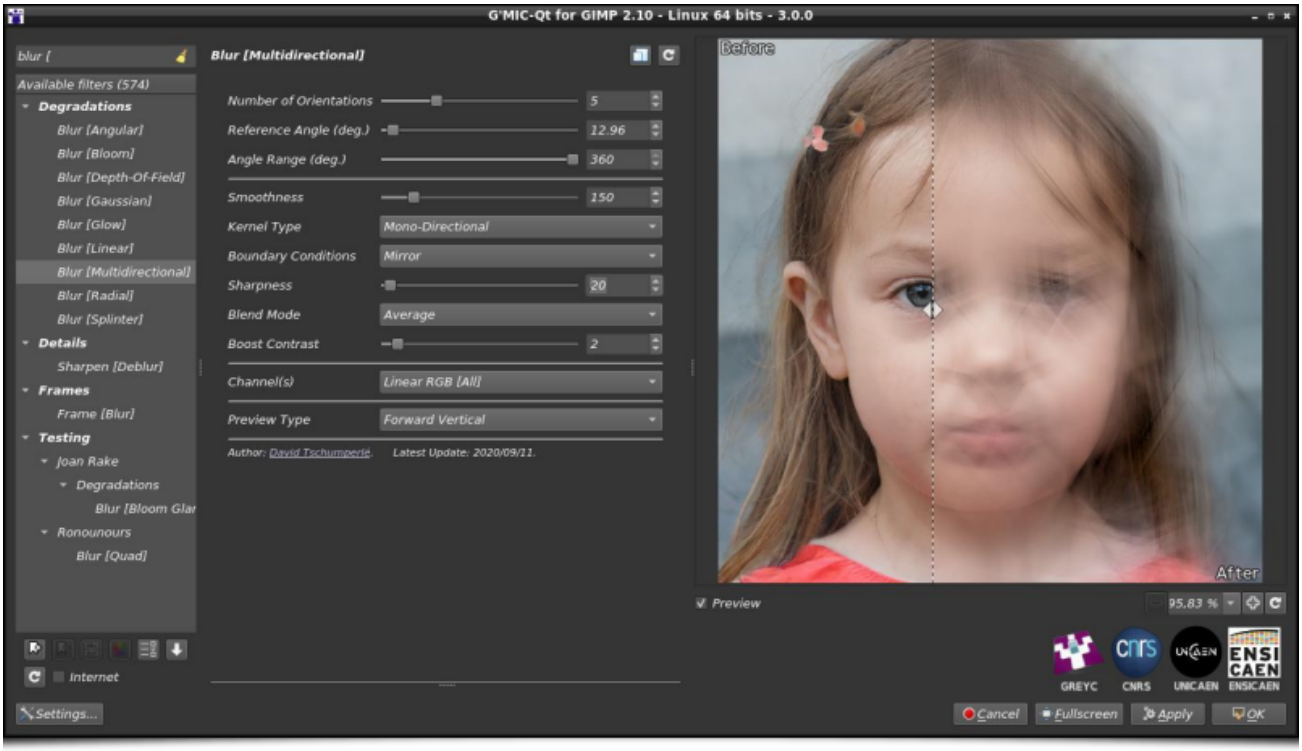


Fig.2.4.4. Le filtre « Degradations / Blur [Multidirectional] » en action.

Fig.2.4.5. Résultats obtenus avec différents paramétrages du filtre.

2.5. Rendus de formes et de motifs

Rentrons maintenant dans le monde merveilleux des filtres de rendus. C'est une catégorie sans pareille où l'on déniche toutes sortes de bizarreries, des filtres qui semblent n'avoir aucune utilité, jusqu'au jour où ils deviennent parfaitement indispensables… ou pas ! Les filtres suivants, qui ont fait leur apparition dans *G'MIC 3.0*, n'échappent pas à cette règle. Ils vous serviront sans doute rarement, mais ils sont quand même sympas !

- Commençons par le plus simple d'entre eux, le filtre **Silhouettes / Others / Dragon Curve** dont le but est de tracer la **Courbe du Dragon** [W], une courbe fractale composée d'un unique tracé continu, construit de manière itérative.

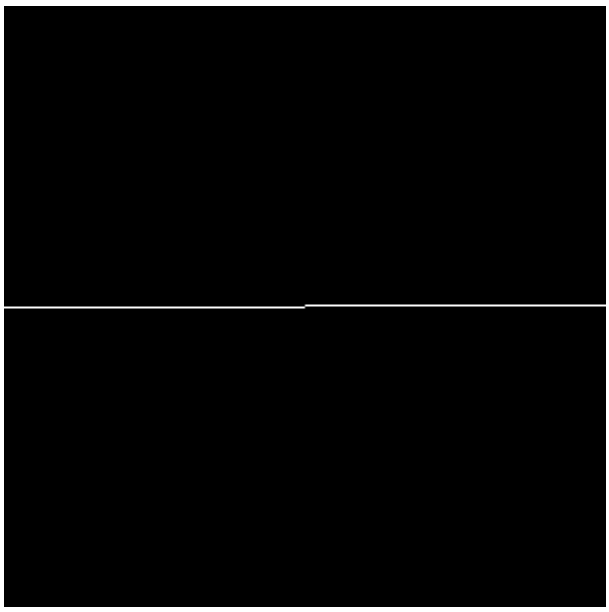


Fig.2.5.1. Différentes itérations de construction de la courbe du dragon.

- Enchaînons rapidement avec le filtre **Rendering / Hypotrochoid** qui, là aussi, a pour but de tracer une courbe sinusoïdale particulière, à savoir une **Hypotrochoïde** [W]. À quoi ça sert ? On ne sait pas vraiment, mais il ne faut jamais sous-estimer la puissance créatrice des artistes utilisateurs !

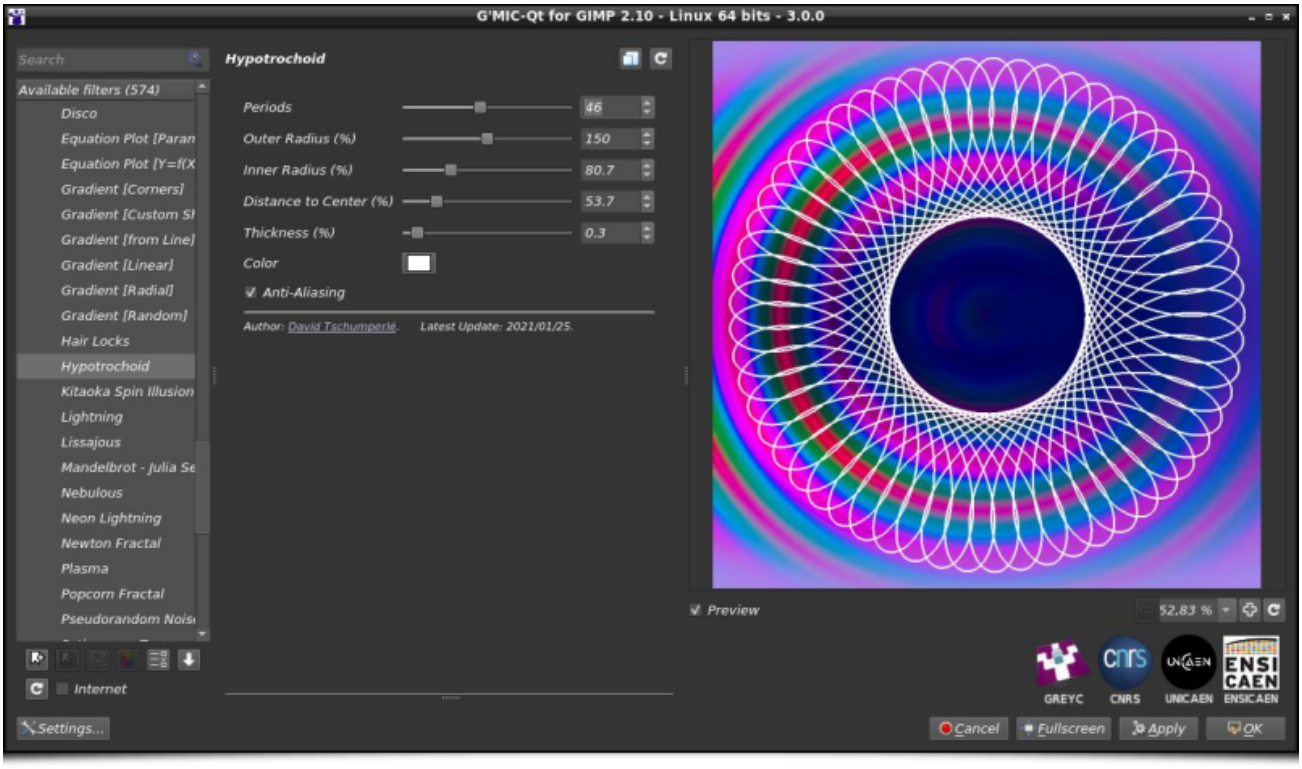


Fig.2.5.2. Le filtre « Rendering / Hypochotroid » en action.

- Toujours dans l'univers du tracé de courbes, attardons-nous un peu plus longuement sur le nouveau filtre **Rendering / Sine Curve**, qui définit une famille de **courbes paramétriques** [W] 2D ou 3D, avec un grand nombre de paramètres réglables par l'utilisateur (près de 50 !).

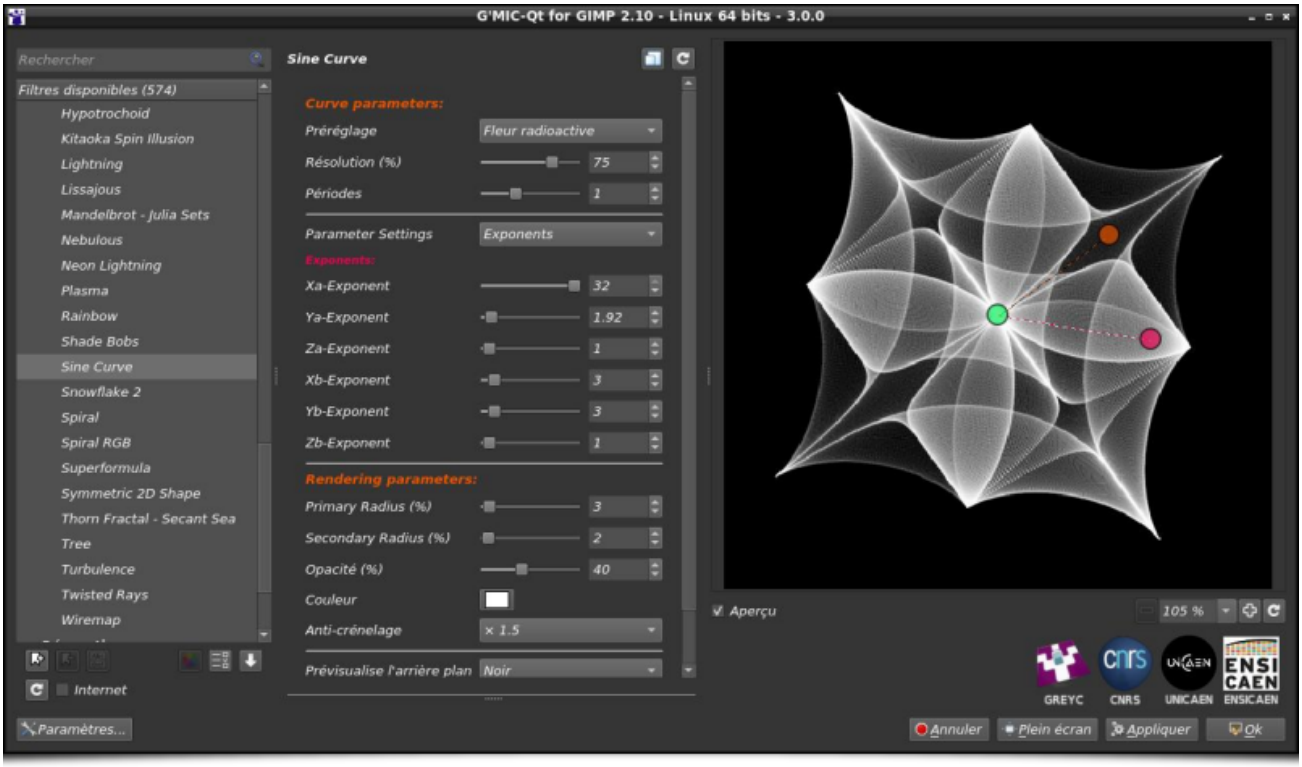


Fig.2.5.3. Le filtre « Rendering / Sine Curve » en action.

Le nombre de paramètres est vraiment élevé et ce filtre propose donc un ensemble de pré-réglages servant de base pour jouer avec toutes les variations de courbes possibles. La figure ci-dessous illustre quelques-unes des formes prédéfinies offertes par l'algorithme.

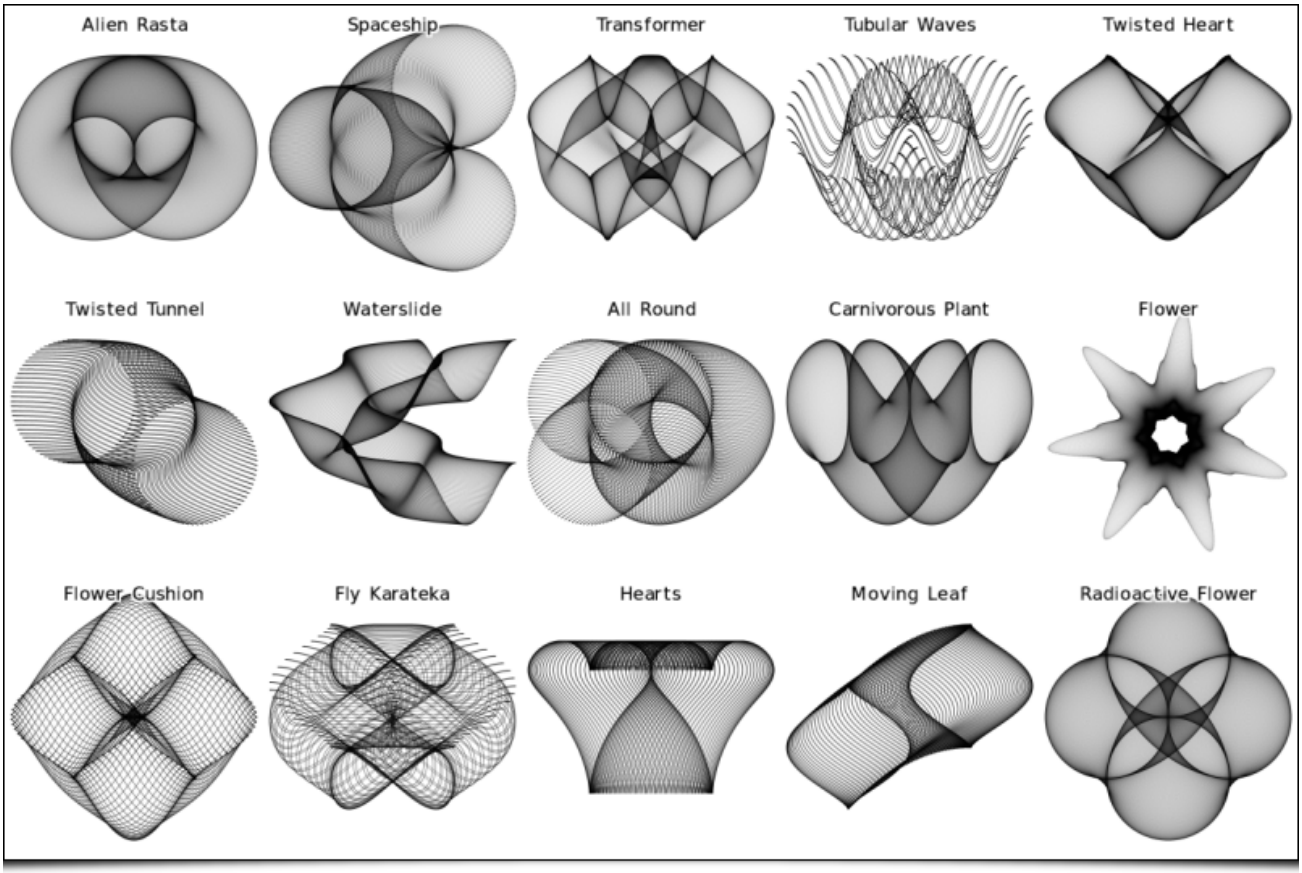
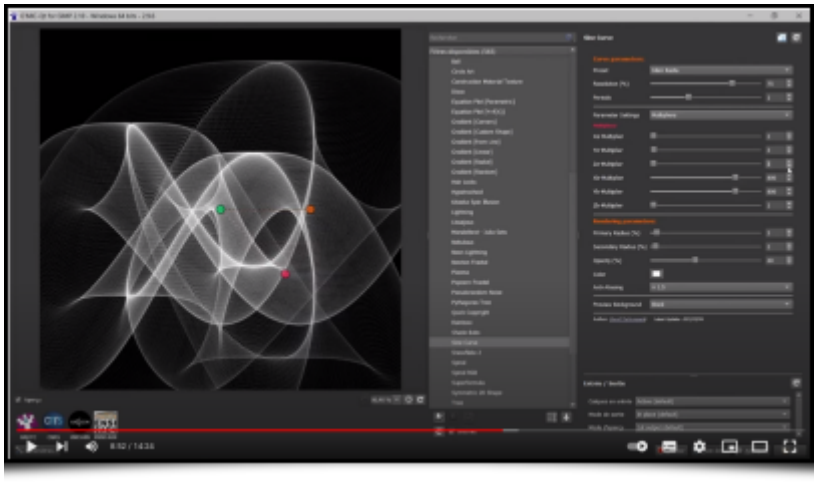


Fig.2.5.4. Quelques pré-réglages de paramètres proposés par le filtre « Rendering / Sine Curve ».

Vous l'aurez compris, ce filtre permet de dessiner des courbes sinusoïdales paramétrées très variées. On pourra l'utiliser notamment pour générer des fonds d'écrans chatoyants. Pierre « El Lobo » a d'ailleurs réalisé un petit

tutoriel vidéo sur ce filtre, visible sur sa chaîne Youtube (suivre le lien ci-dessous)



- Et si, plutôt que de se contenter de rendus de courbes 1D, on s'intéressait maintenant à la génération paramétrique d'images en 2D ? C'est ce que propose le filtre **Patterns / Random Pattern**, dont le fonctionnement est assez original : il génère une fonction mathématique aléatoire `f(z)` (`z` étant une variable complexe), composée de fonctions complexes de base (`cos()`, `sin()`, `pow()`…) et d'opérateurs (`+`, `-`, `*`, `/`…), en construisant récursivement un arbre d'expression. `f(z)` est ensuite évaluée pour tous les points `z` de l'image, localisés dans un domaine rectangulaire, lui-même choisi aléatoirement. Les valeurs de `f(z)` sont finalement transformées en valeurs *RGB* pour déterminer les couleurs de tous les pixels de l'image.

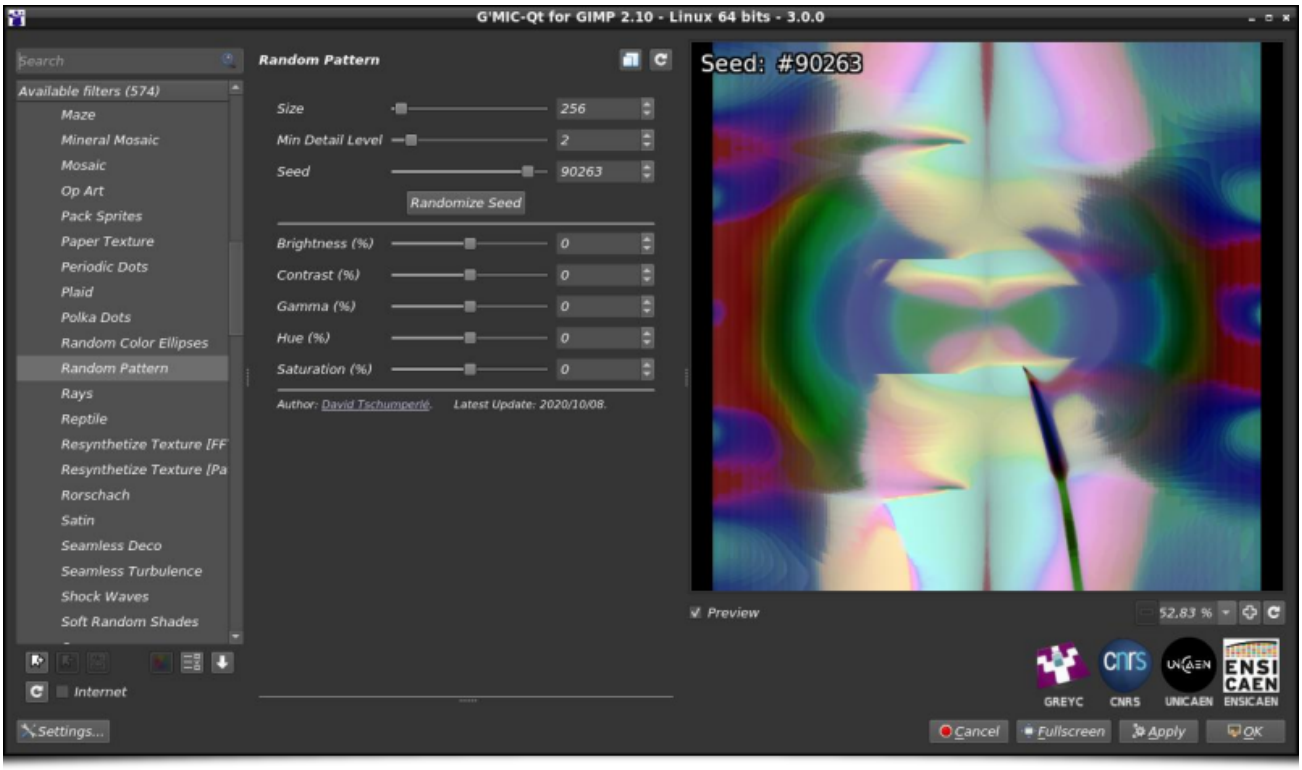


Fig.2.5.5. Le filtre « Patterns / Random Pattern » en action.

Notons que la construction de la fonction `f(z)` dépend uniquement de la graine aléatoire utilisée pour engendrer l'arbre d'expression. La figure ci-dessous illustre quelques rendus d'images obtenus avec différentes valeurs de graines aléatoires.

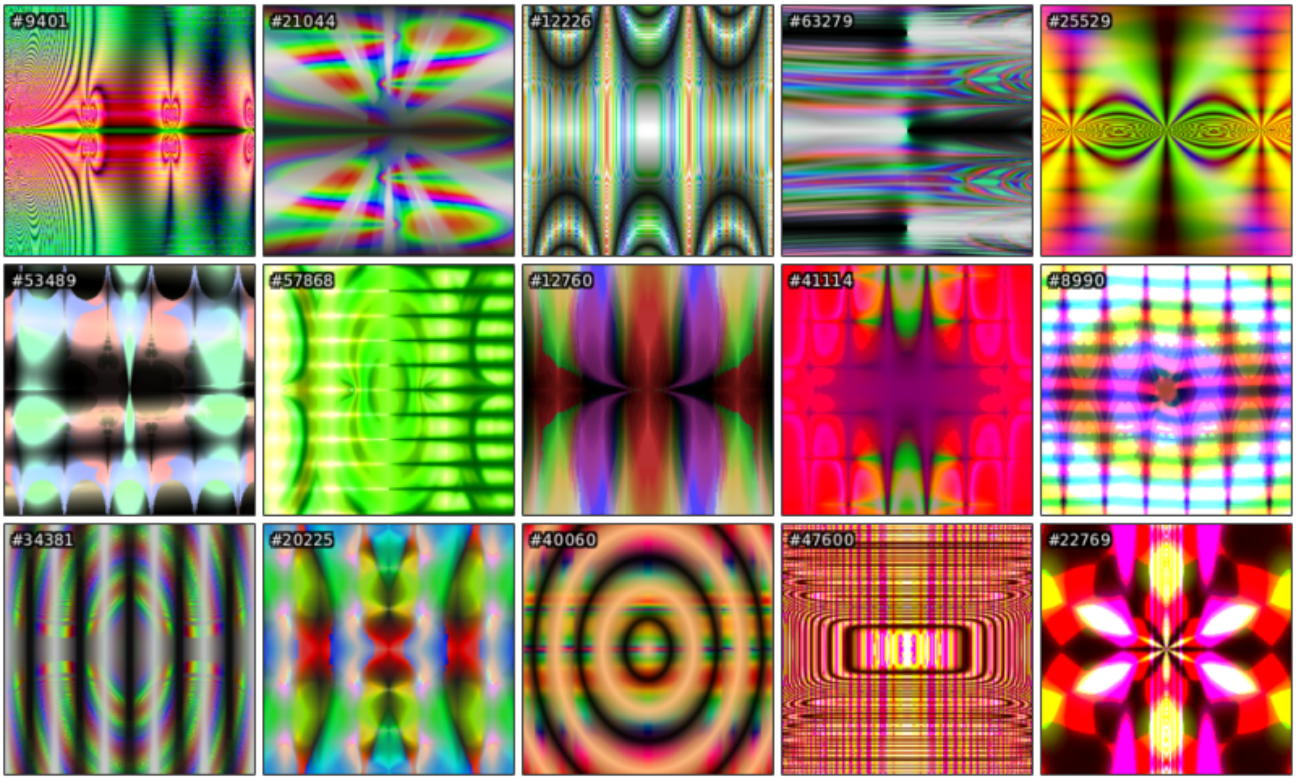


Fig.2.5.6. Quelques exemples de motifs aléatoires générés par le filtre « Patterns / Random Pattern ».

- Les méthodes de génération récursive d'images peuvent produire des motifs géométriques originaux et intéressants, ce que semble également confirmer le filtre **Patterns / Triangular Pattern**. Il génère des images constituées de triangles récursivement subdivisés de différentes façons, chacune ayant une probabilité d'occurrence réglable par l'utilisateur.

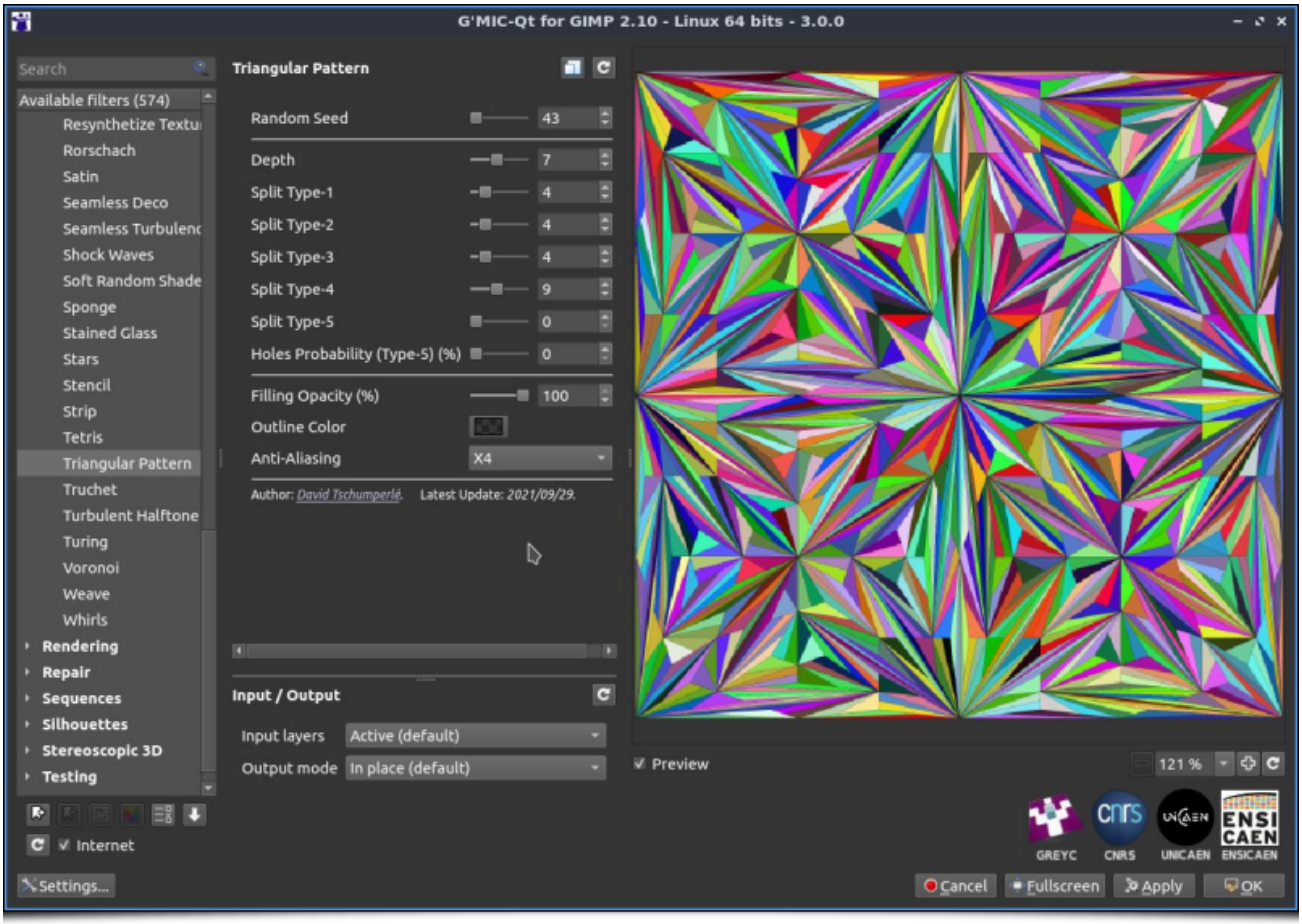


Fig.2.5.7. Le filtre « Patterns / Triangular Pattern » en action.

Au fur et à mesure des itérations, on voit se former dans l'image des motifs géométriques de plus en plus complexes, comme le montre l'animation ci-dessous.

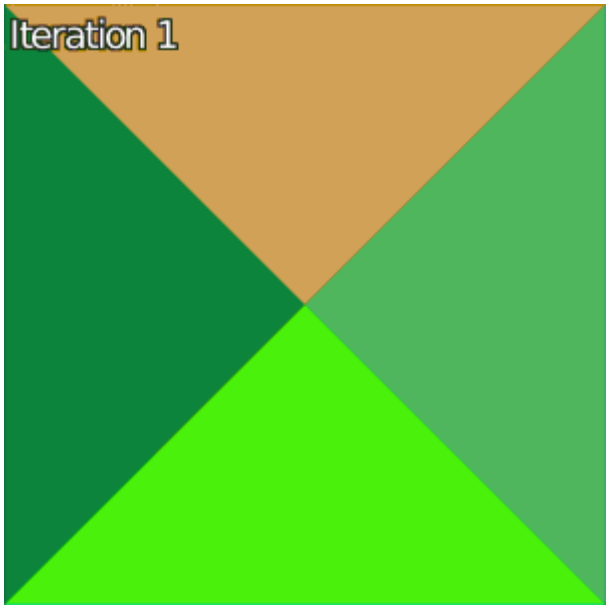


Fig.2.5.8. Visualisation des itérations de construction du motif triangulaire par le filtre « Patterns / Triangular Pattern ».

En augmentant le nombre d'itérations et en ne traçant que les contours des triangles subdivisés, on arrive à générer des motifs géométriques fractals vraiment remarquables, étant donnée la simplicité apparente de l'algorithme de subdivision de triangles, comme l'illustre la figure suivante. De quoi donner des idées de motifs pour carreler sa cuisine ou sa salle de bain !

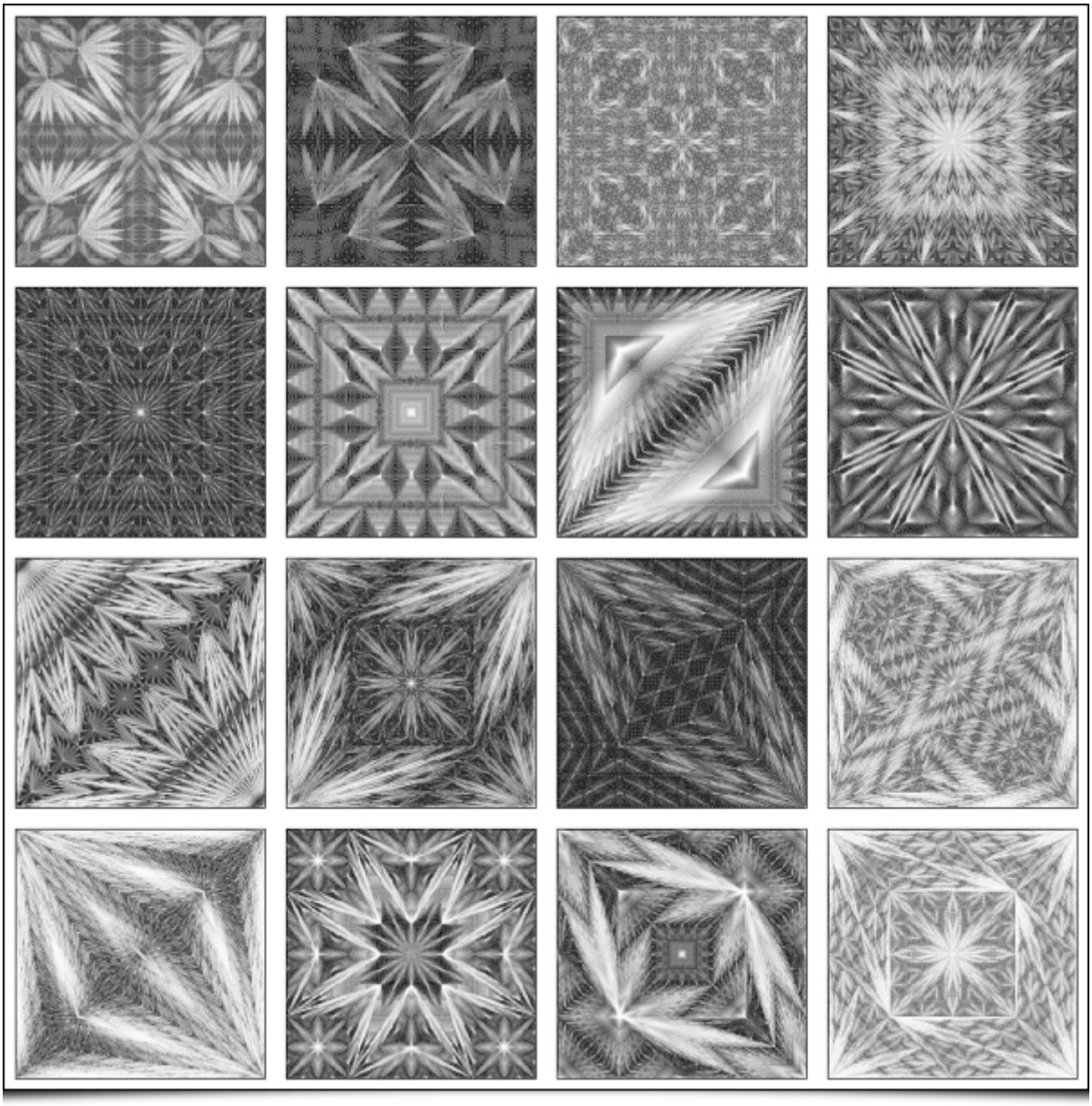


Fig.2.5.9. Quelques exemples de subdivisions triangulaires récursives aléatoires obtenues par le filtre « Patterns / Triangular Pattern ».

- Et en parlant de fractales, venons-en au filtre **Rendering / Newton Fractal** qui permet justement de naviguer dans les **Fractales de Newton** [W], avec un grand nombre de paramètres ajustables. Ces fractales sont obtenues en cherchant à trouver les zéros d'une fonction complexe `f(z)` (souvent un polynôme) par la **méthode de Newton** [W]. Cette technique itérative construit une suite censée converger vers l'un des zéros de la fonction. Pour chaque point (complexe) `z` de l'image, on détermine donc sa couleur affichée en fonction du nombre d'itérations nécessaires pour arriver à convergence, ce qui donne ces motifs géométriques fractals de toute beauté.

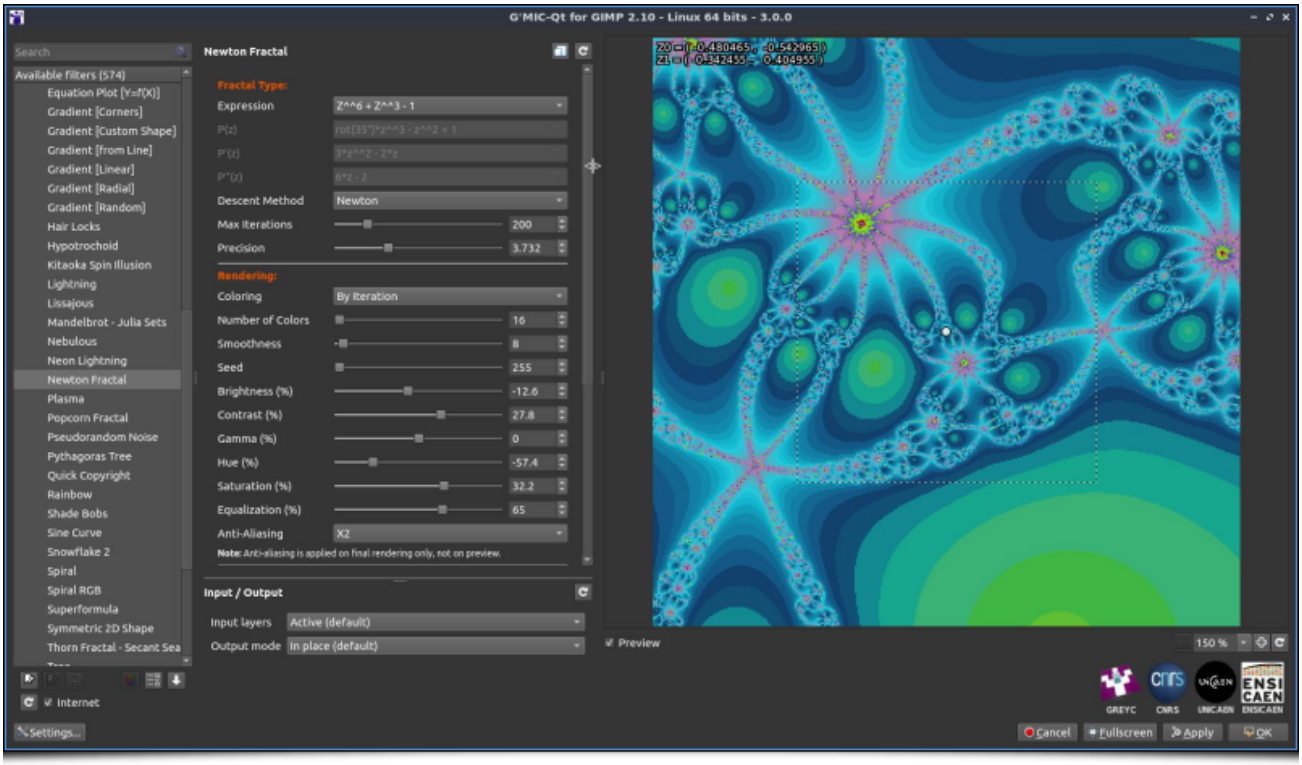


Fig.2.5.10. Le filtre « Rendering / Newton Fractal » en action.

Dans *G'MIC-Qt*, le filtre propose à l'utilisateur de rentrer sa propre formule mathématique pour définir la fonction `f(z)`, ainsi que ses dérivées. Il offre aussi des options pour coloriser la fractale de manière personnalisée et pour se balader facilement dans l'espace fractal grâce à son navigateur intégré, simple mais efficace. Enfin, il étend le principe du calcul à d'autres méthodes de résolutions (**méthode de la Sécante**[W] et **méthode de Householder**[W]). Et comme toujours avec *G'MIC*, il est aussi possible de scripter ce filtre à partir de la ligne de commande (avec la commande `gmic`), pour générer par exemple ce type d'animations :

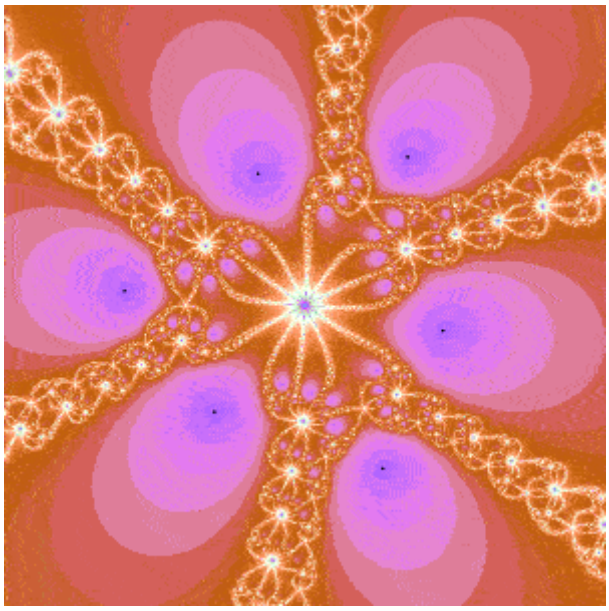


Fig.2.5.11. Génération d'un zoom dans une fractale de Newton, scriptée avec `gmic`.

- Toujours dans l'esprit « fractal », le nouveau filtre **Rendering / Tree** vous propose de générer des **arbres fractals**, là encore avec de nombreux paramètres réglables pour varier les plaisirs.

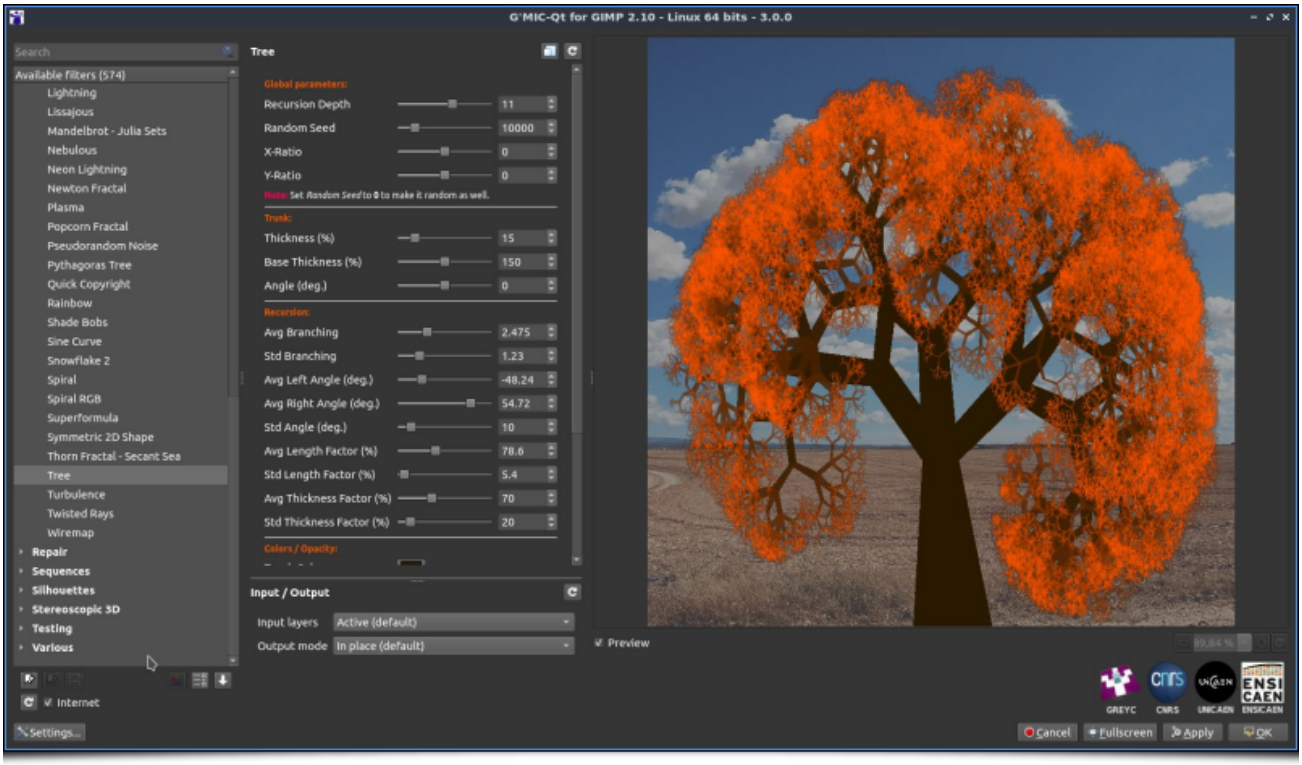


Fig.2.5.12. Le filtre « Rendering / Tree » en action.

On peut ainsi ajuster, outre les couleurs du tronc et des feuilles, les tailles des branches et les probabilités de séparation des branches en plusieurs morceaux. Cela permet de générer des arbres à la géométrie et aux looks assez différents, comme vous pouvez le voir sur la figure ci-dessous.

Fig.2.5.13. Exemples de rendus d'arbres avec des paramétrages différents, par le filtre « Rendering / Tree ».

La vidéo suivante illustre l'utilisation de ce filtre dans *GIMP* (cliquez sur l'image ci-dessous pour accéder à la vidéo).

- Un autre effet de rendu qui mérite d'être mis en lumière (sans mauvais jeu de mot…) est le filtre **Light & Shadows / Guided Light Rays**. Ce filtre permet de lancer des rayons lumineux à partir d'une source de lumière ponctuelle, vers ou à travers une forme quelconque dessinée par l'utilisateur et définie sous la forme d'un masque opaque placé sur un calque transparent au-dessus de l'image d'entrée.

Fig.2.5.14. Le filtre « Light & Shadows / Guided Light Rays » en action.

Dans l'exemple ci-dessus, les rayons lumineux s'arrêtent lorsqu'ils rencontrent les pixels du logo *G'MIC*. Mais on peut aussi adapter les paramètres du filtre pour que les rayons traversent la forme. C'est ce qui est fait dans l'exemple ci-dessous, où le masque défini par l'utilisateur correspond à la surface vitrée des fenêtres d'un salon.

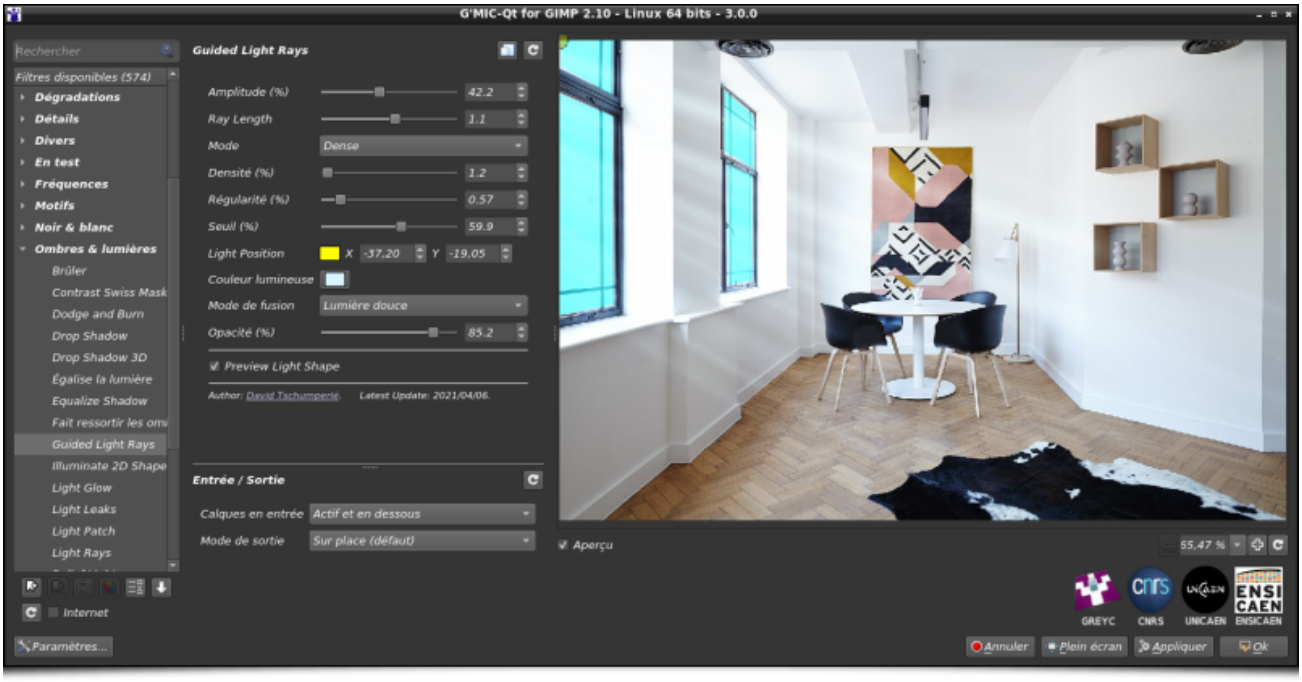
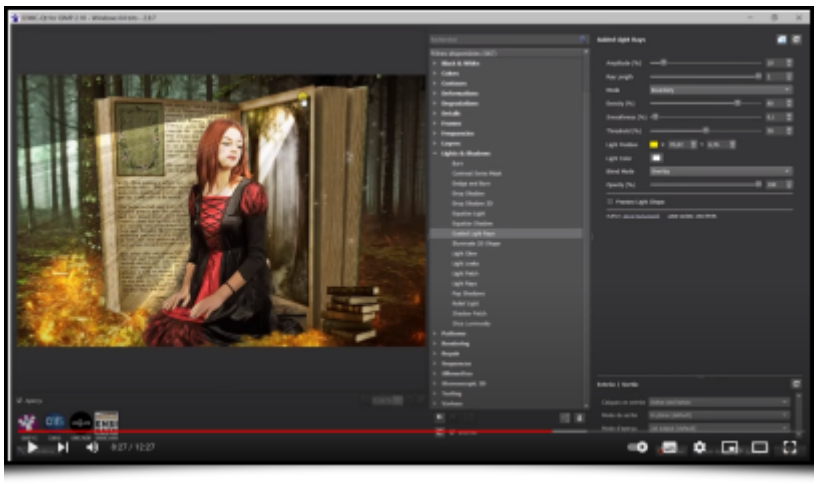


Fig.2.5.15. Rayons lumineux synthétisés pour passer à travers un masque défini par l'utilisateur.

Avec ce filtre, il est ainsi facile d'ajouter des effets d'éclairages personnalisés sur des photographies, comme montré dans l'exemple suivant. Le tout est de bien définir le masque de la forme guidant les rayons lumineux, et le filtre fait le reste !

Fig.2.5.16. Ajout d'un effet d'éclairage latéral grâce au filtre « Light & Shadows / Guided Light Rays ».

La vidéo tutorielle suivante montre l'utilisation de ce filtre sous *GIMP* :



- Enfin, pour conclure notre tour des nouveautés concernant les effets de rendus, terminons avec le filtre **Sequences / Moiré Animation** qui nous entraîne dans le monde merveilleux de l'animation basée sur les technologies (robustes) du siècle dernier, à savoir : la feuille de papier et le calque transparent ! Ce filtre va en effet permettre de créer, à peu de frais, une illusion d'optique animée, en générant des images ayant un **effet Moiré** [W]. À partir d'une animation donnée en entrée (contenant peu d'images), le filtre va produire une image fixe ainsi qu'un masque binaire contenant des barres verticales comme ci-dessous :

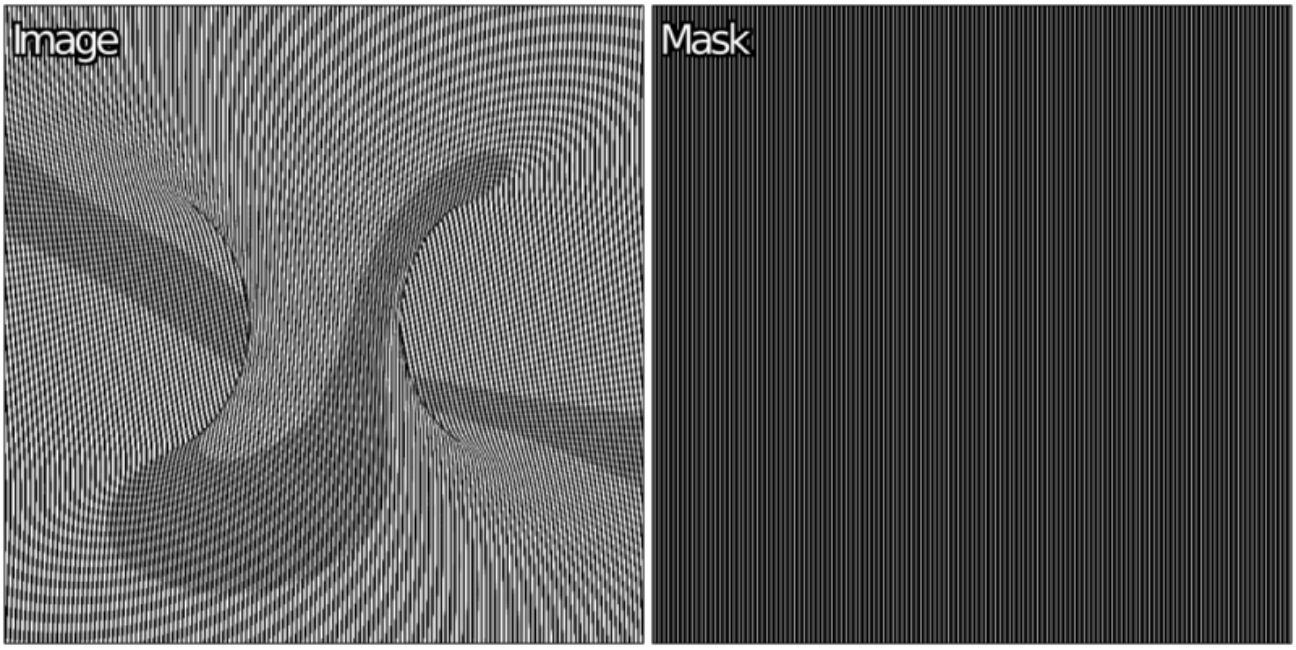


Fig.2.5.17. Images générées par le filtre « Sequences / Moiré Animation » à partir d'une animation constituée de 5 images.

L'astuce consiste à imprimer la première image sur une page blanche et la seconde sur un calque transparent de même taille. En positionnant le transparent sur la page de manière adéquate, et en déplaçant celui-ci latéralement, on rend visible l'animation. En effet, les barres verticales imprimées sur le transparent ne laissent passer qu'une seule image de l'animation à la fois, ces images se succédant lorsque le transparent se déplace :

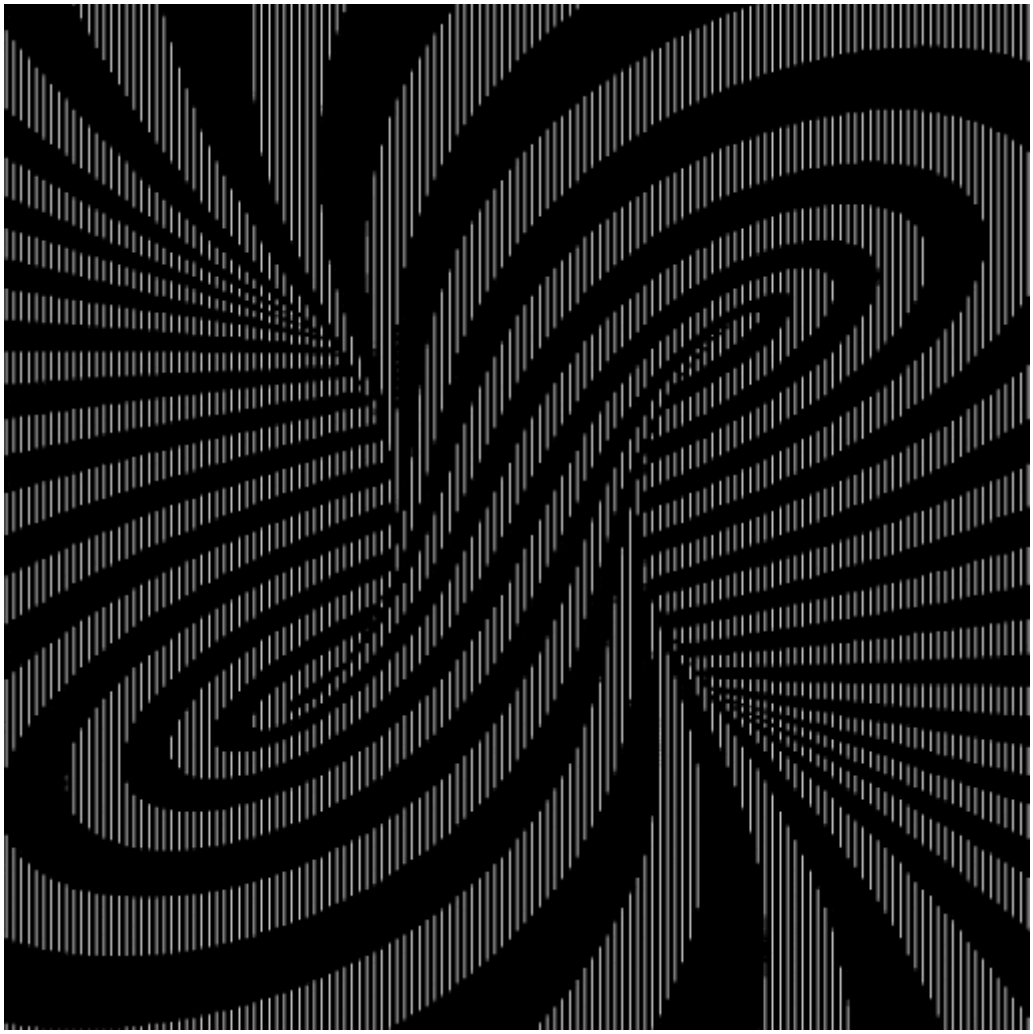


Fig.2.5.18. Rendu de l'animation lorsque le transparent se déplace sur la page imprimée.

À noter que si le nombre d'étapes-clés de l'animation doit être effectivement restreint (en général, moins d'une dizaine, il n'est évidemment pas question de convertir un film de deux heures sous cette forme !), il n'y a pas de raisons de se limiter à des images en niveaux de gris. Les deux figures suivantes illustrent, par exemple, le rendu d'une animation couleur avec ce filtre.

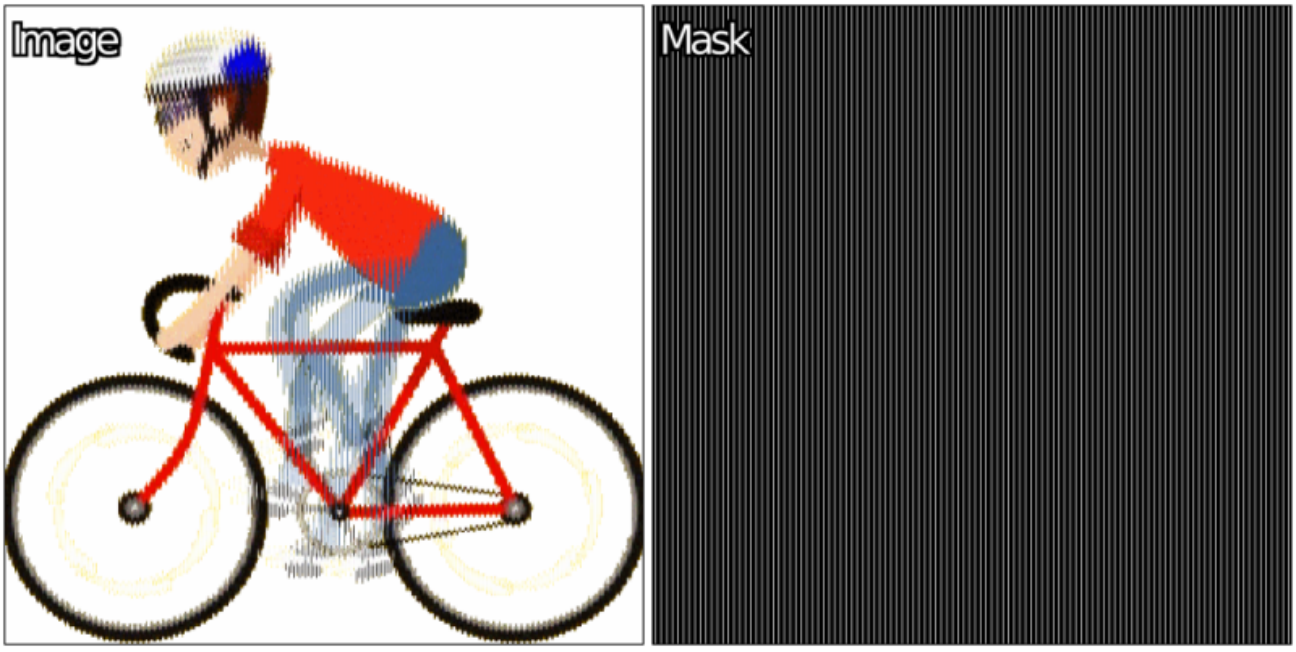


Fig.2.5.19. Images générées par le filtre « Sequences / Moiré Animation » à partir d'une animation couleur de 6 images.

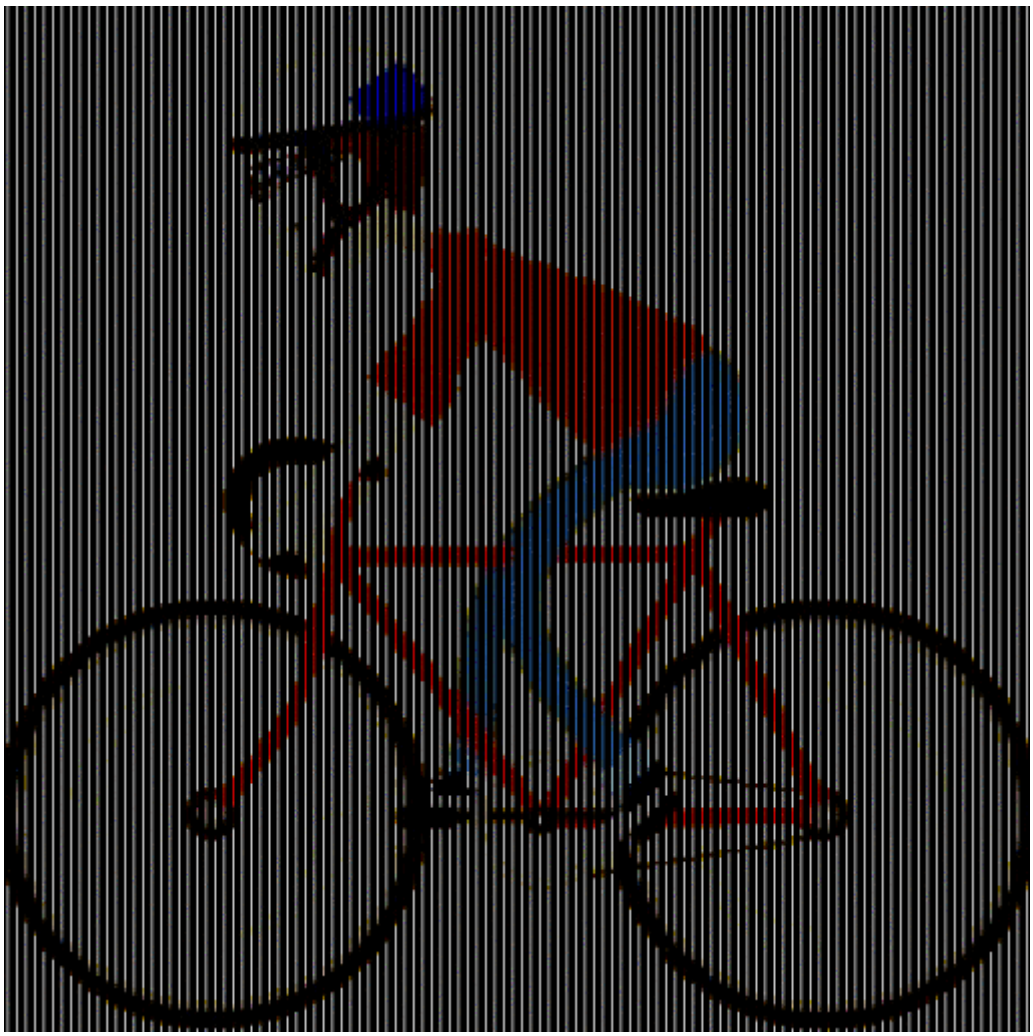
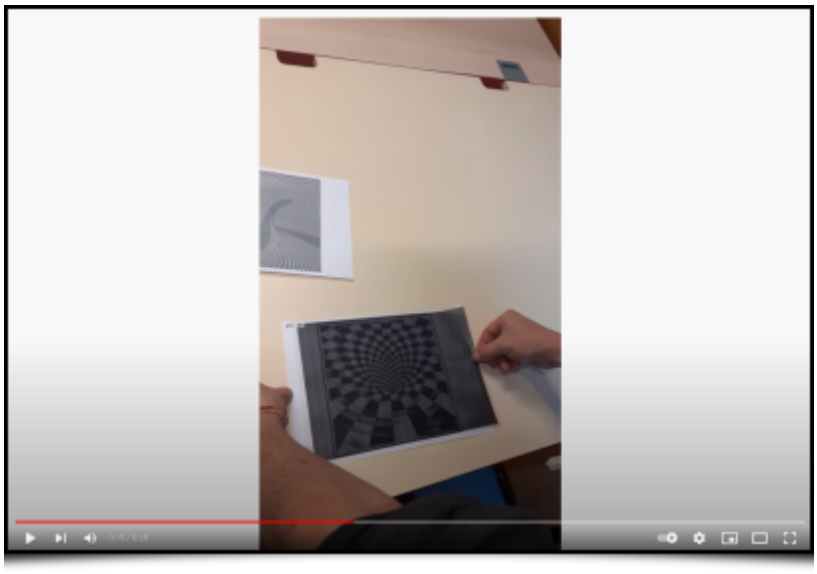


Fig.2.5.20. Rendu de l'animation lorsque le transparent se déplace sur la page imprimée.

« Tes exemples, c'est bien joli, mais en vrai, ça donne quoi ? », m'objecterez-vous. Eh bien, le ressenti d'animation est vraiment présent, comme on peut l'apprécier dans la vidéo ci-dessous :



Au final, c'est une manière peu onéreuse et amusante de créer des supports d'animation personnalisées. Les enfants, pas encore ~~lobotomisés~~ blasés par l'application *Youtube* de leur téléphone portable, adoreront !

2.6. Se maintenir à jour

L'ensemble des filtres que nous venons de décrire ne représente qu'une sous-partie de tous les nouveaux filtres ajoutés ces deux dernières années. Il existe en particulier de nombreux filtres « communautaires » qui restent parfois dans l'ombre un certain temps avant d'intégrer la liste principale des filtres proposés (filtres de la catégorie

Testing / notamment). Pour faciliter le suivi de ces mises à jour, *G'MIC-Qt* s'est donc doté de la fonction **About** / **What's New?**, dont le rôle est de lister les récentes additions et suppressions de filtres dans le greffon. Cette liste est adaptée à l'utilisateur, puisqu'elle est élaborée en fonction de la liste précédemment consultée par celui-ci.

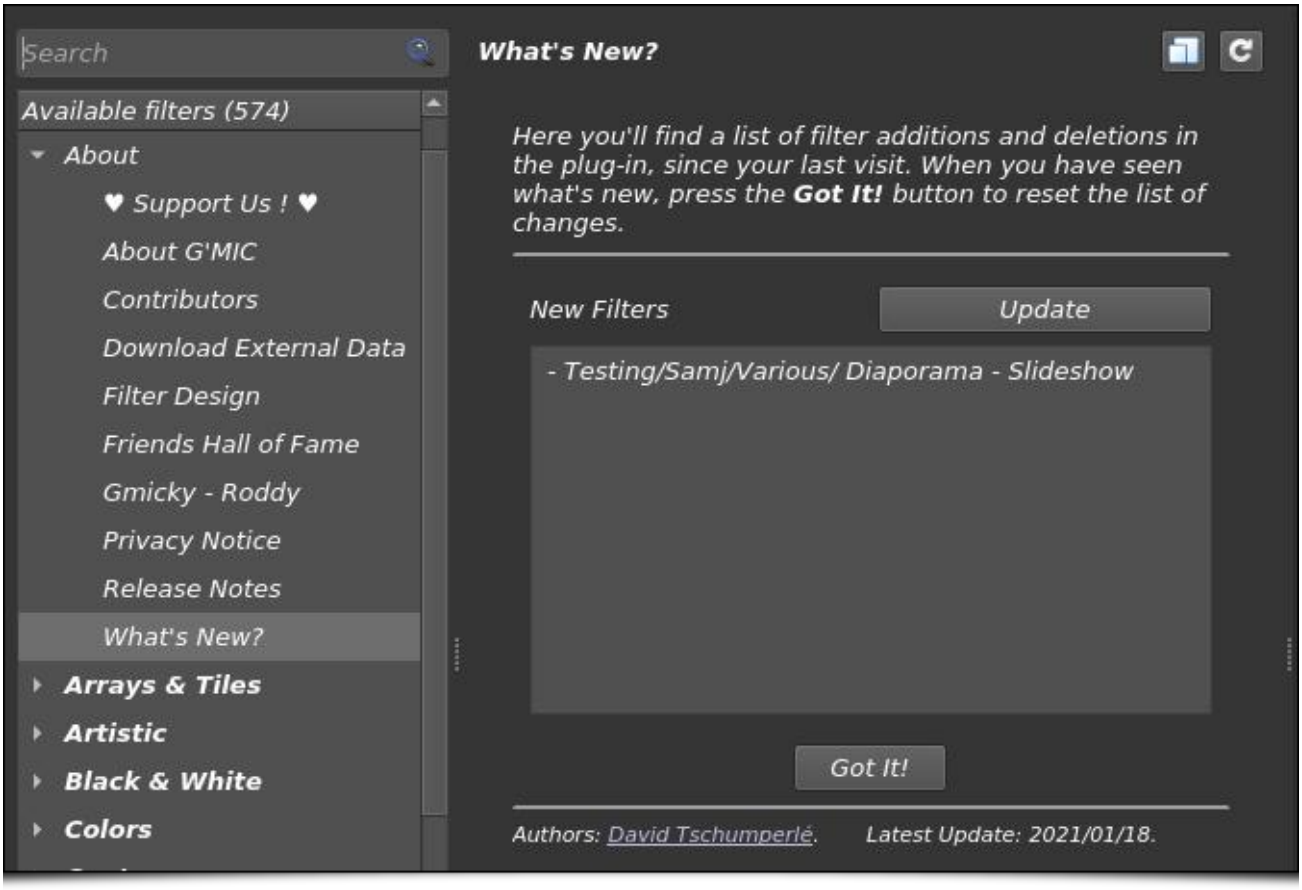


Fig.2.6.1. Le filtre « About / What's New? » en action.

Pour rappel, le bouton **Mettre à jour les filtres** situé sous la liste déroulante des filtres permettra au greffon de vérifier si de nouveaux filtres sont disponibles, et le cas échéant, de les télécharger.

Ce tour d'horizon des nouveaux filtres de *G'MIC-Qt* est maintenant terminé. Mais d'autres améliorations du greffon nous attendent !

2.7 Toujours plus de logiciels hôtes

- L'une des nouvelles marquantes dans la vie de *G'MIC-Qt* concerne sa mise à disposition sous la forme d'un greffon compatible avec le protocole ***8bf***. Cette *API* a été imaginée par *Adobe* dans les années 90 pour le développement de greffons pour son logiciel phare (et non-libre) *Photoshop*. C'est un protocole toujours utilisé, malgré son grand âge (et ses limitations techniques). Avec les années, cette *API* de greffon a même été adoptée par de nombreux autres logiciels d'illustration et de retouche photo. Par conséquent, cette version du greffon *G'MIC-Qt* peut en pratique être maintenant utilisée à l'intérieur de nombreux logiciels populaires, compatibles avec le protocole *8bf*, à savoir : ***Adobe Photoshop***[W], ***Affinity Photo***[W], ***Paint Shop Pro***[W], ***Photoline***, ***XnView***, pour n'en citer que quelques-uns.

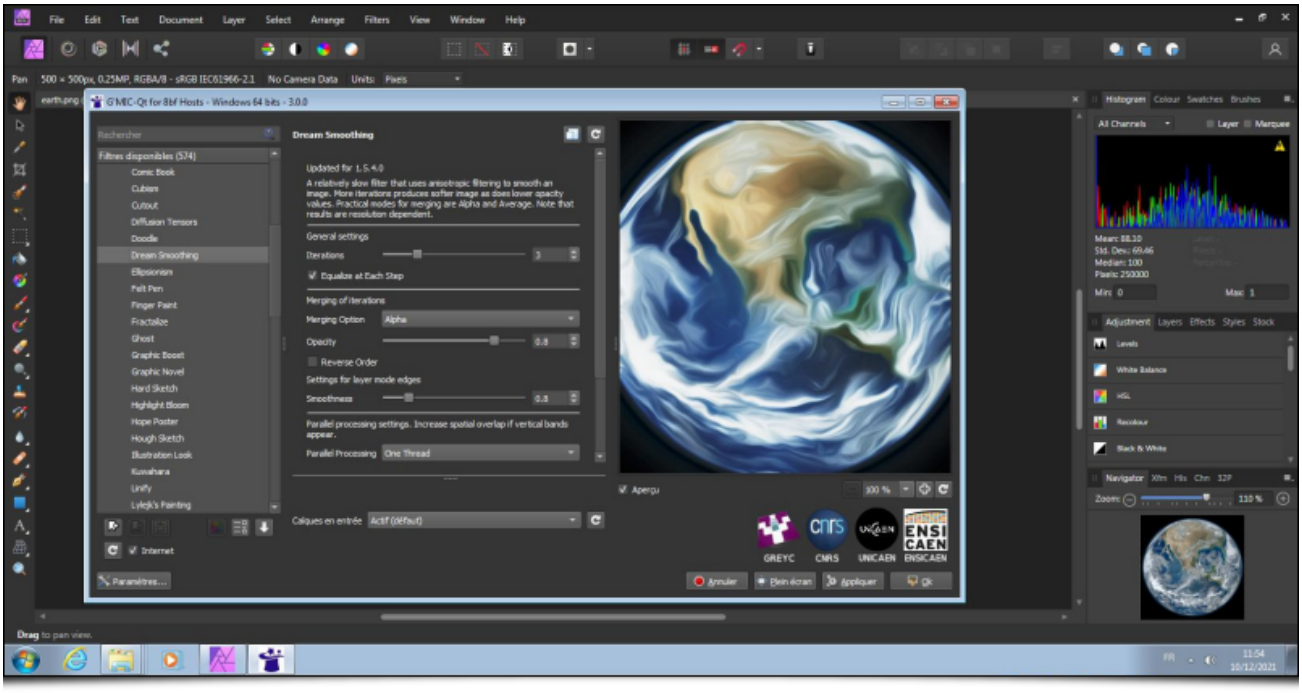


Fig.2.1.1. *Le greffon G'MIC-Qt tournant sous Affinity Photo / Windows.*

C'est grâce à ***Nicholas Hayes***, déjà l'auteur du portage du greffon pour le logiciel ***Paint.NET***, que cette version *8bf* de *G'MIC-Qt* a été réalisée. C'est une bonne nouvelle pour tous les utilisateurs de ces logiciels propriétaires, qui pourront bénéficier d'un nouveau greffon bien fourni en filtres divers de traitement d'images, avec la possibilité de les modifier et de les améliorer, et tout cela sans rien payer (ça va les changer ! ☺).

- Une autre bonne nouvelle concerne l'amélioration du support du greffon *G'MIC-Qt* pour *GIMP*, notamment sous *Windows*, où plusieurs utilisateurs nous avaient rapporté des problèmes de compatibilités avec les `.dll` fournies, lors de la sortie de la dernière version de *GIMP* (la 2.10.28). De plus, grâce aux modifications apportées par **Jan Tojnar**, le greffon *G'MIC-Qt* peut maintenant se compiler pour la version 2.99 de *GIMP* (nom de la version de développement de *GIMP* qui deviendra la très attendue *3.0*). Mentionnons également l'arrivée du greffon *G'MIC-Qt* pour la version **Flatpack** de *GIMP*, grâce au travail d'**Hubert Figuière** qui l'a empaqueté pour **Flathub**.

- Enfin, notons l'intégration attendue de *G'MIC-Qt 3.0* dans la prochaine version majeure *5.0* du logiciel *Krita*, dont la sortie est planifiée ce mois de décembre. Cette intégration est prévue pour être complète et systématique, c'est-à-dire que *G'MIC-Qt* sera inclus par défaut dans *Krita*, et non plus installé comme un greffon extérieur (comme c'est le cas actuellement). Cela promet une meilleure intégration dans *Krita* et devrait régler plusieurs problèmes récurrents, en particulier pour les utilisateurs de *Mac*. C'est ***amyspark***, avec qui nous sommes en contact, qui est en charge de cette intégration.

2.8. Améliorations de l'interface

Le greffon a aussi vu son interface graphique s'améliorer, avec en particulier :

- La mise à disposition d'un système de *tags* colorés pouvant être assignés à un ou plusieurs filtres. Ces pastilles de couleur (6 couleurs différentes) permettent à l'utilisateur qui le souhaite de marquer certains filtres pour les retrouver plus facilement par la suite. Chaque utilisateur décide bien sûr du sens qu'il attribue à chaque couleur et les cas d'utilisation sont donc potentiellement nombreux : on peut utiliser ces *tags* de manière temporaire pour une session donnée, ou au contraire marquer les filtres sur du plus long terme. L'interface du greffon propose un système simple mais efficace pour gérer ces différents *tags* comme le montre l'animation ci-dessous :

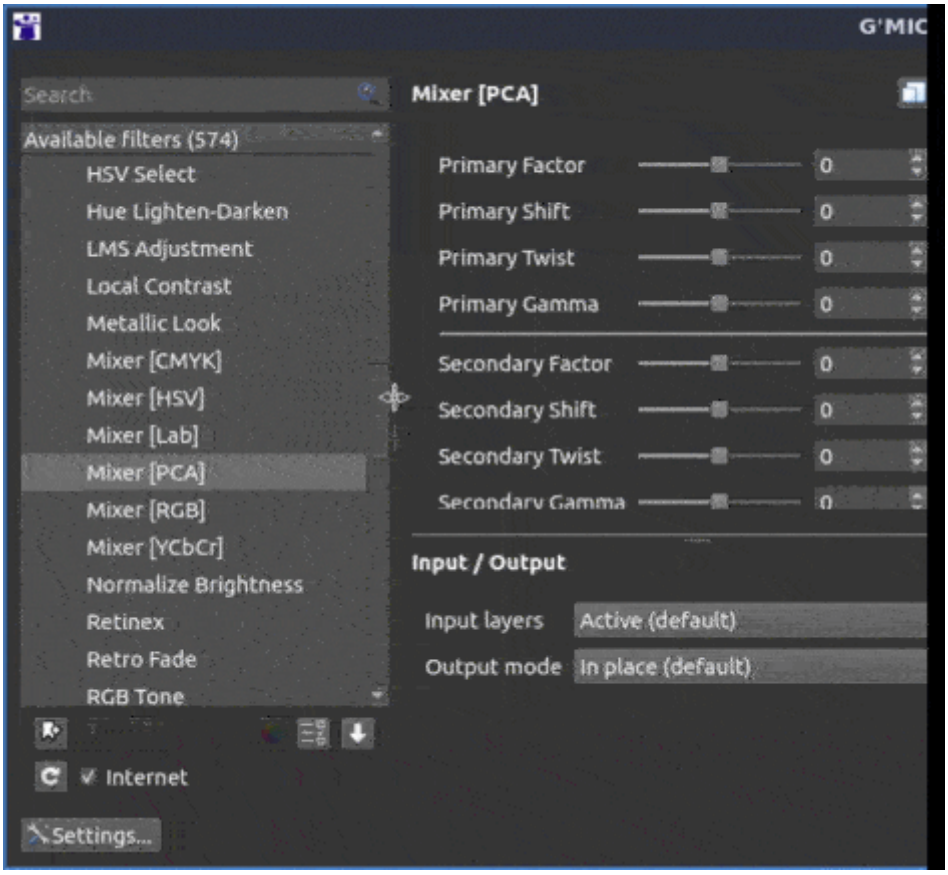


Fig.2.8.1. Un nouveau système de « tags » colorés pour marquer les filtres a fait son apparition dans le greffon.

- L'apparition d'un bouton *« Copier la commande G'MIC dans le presse-papier »* localisé en haut à gauche du panneau contenant la liste des filtres :

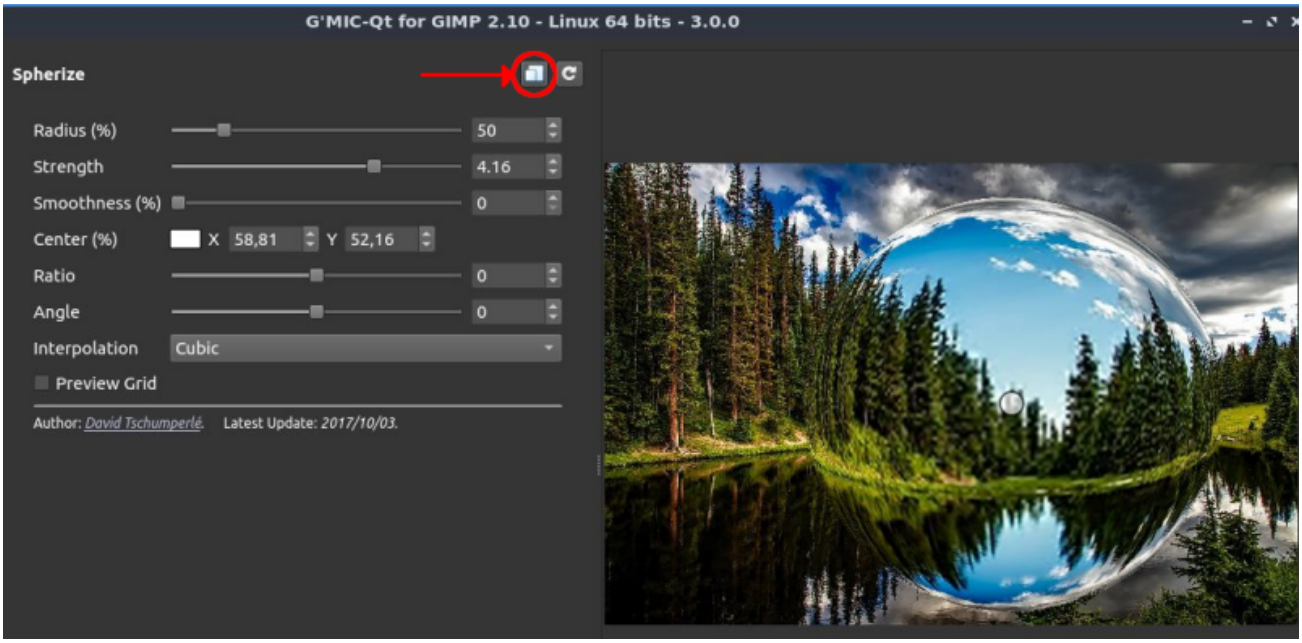


Fig.2.8.2. Nouveau bouton de copie de la commande G'MIC dans le presse-papier.

Ce bouton ravira tous les scripteurs : en appuyant dessus, on obtient la commande *G'MIC* à invoquer dans le terminal avec l'outil en ligne de commande `gmic`, afin d'appliquer le filtre actuellement sélectionné avec ses valeurs de paramètres courantes. Voici par exemple ce que contient le presse-papier après avoir appuyé sur ce bouton dans l'exemple ci-dessus :

```
fx_spherize 50,4.16,0,58.8136,52.1628,0,0,2,0
```

On peut alors recoller cette chaîne dans un appel à `gmic` en ligne de commande, pour obtenir exactement le même effet sur d'autres images, ou pour l'intégrer dans un pipeline plus complexe de traitement :

Fig.2.8.3. Application via le terminal de la commande G'MIC copiée depuis le greffon G'MIC-Qt, sur d'autres images.

C'est une fonctionnalité vraiment pratique, pour peu que vous écriviez vos propres scripts de traitement. On peut ainsi profiter de la commodité de l'interface graphique du greffon (en particulier sa fenêtre de prévisualisation) pour régler les paramètres d'un filtre, mais aussi du confort de la ligne de commande pour appliquer un traitement par lots sur plusieurs dizaines ou centaines de fichiers images.

- Citons également entre autres améliorations, le fait que *G'MIC-Qt* se dote d'un système de traduction amélioré (une version française complète est d'ailleurs prévue à court terme), et que l'*API* proposée par le greffon a été enrichie pour permettre plus de contrôle par le logiciel hôte. Ces améliorations sont par exemple utilisées par *G'MIC-Qt* pour proposer de **nouvelles options** de contrôle via un appel en ligne de commande.

3. Améliorations du cœur de G'MIC

Cette description des nouveautés visibles dans le greffon *G'MIC-Qt* étant achevée, abordons les autres améliorations apportées par cette version *3.0*. Beaucoup de choses intéressantes ont été ajoutées au cœur du projet *G'MIC*, à savoir son interpréteur et sa bibliothèque de traitement d'image associée ***CImg***, qui passent également en version *3.0* pour l'occasion. Ces nouveautés sont certes moins visibles, mais tout aussi importantes puisqu'elles ont un impact potentiel sur l'ensemble des interfaces du projet !

3.1. Améliorations du langage et de l'interpréteur

- Commençons par la fonctionnalité qui a demandé le plus d'effort de développement, que nous avons déjà évoquée précédemment : l'implémentation d'une bibliothèque interne d'**apprentissage machine** [W] (`nn_lib`). Cette bibliothèque permet de manipuler des **réseaux de neurones** [W] génériques et prend en charge à la fois la phase d'apprentissage du réseau et la phase d'inférence. Cette bibliothèque a été réimplémentée à partir de zéro, ce qui représente un effort réellement conséquent de recherche, d'implémen-

tation et de tests (mais ça a été aussi un travail ô combien instructif !). `nn_lib` permet la construction dans *G'MIC* de réseaux de neurones comportant des modules convolutifs ou complètement connectés, des modules de *pooling*, des modules résiduels, etc. Plusieurs optimiseurs ont été implémentés pour gérer l'apprentissage (`SGD` , `RMSprop` , `Adam` , `Adamax`) et il est donc déjà possible d'entraîner des réseaux pour certaines tâches de traitement d'images avec cette nouvelle bibliothèque. Le diagramme ci-dessous illustre par exemple l'une des architectures de réseau qui a été entraînée dans le cadre du nouveau filtre de débruitage **Repair / Denoise** cité auparavant.

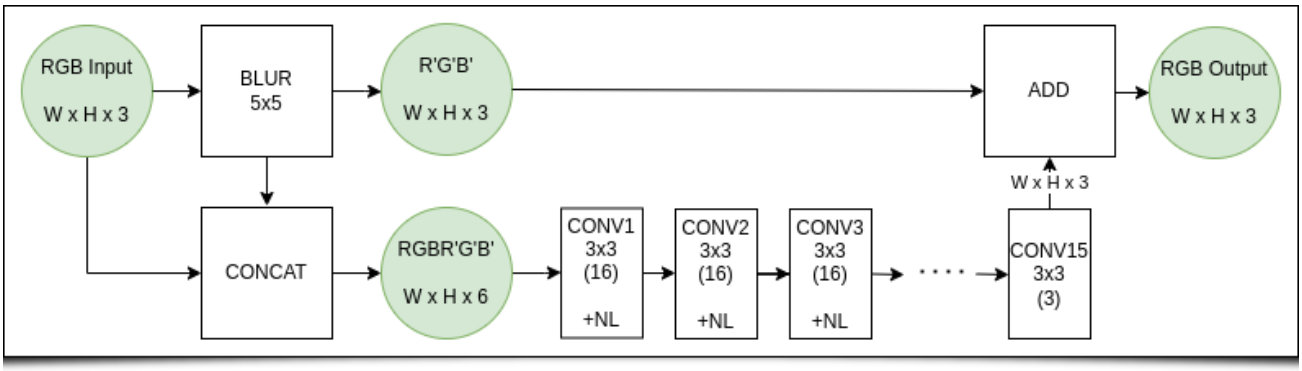


Fig.3.1. Une des architectures de réseau de neurones entraîné pour le nouveau filtre de débruitage « Repair / Denoise ».

Ce n'est bien sûr encore qu'un début, mais nous espérons généraliser l'utilisation de `nn_lib` pour élaborer de nouveaux filtres intéressants dans un avenir proche. Plus de détails techniques sur la bibliothèque `nn_lib` sont visibles dans **l'article dédié** (en anglais) sur le forum de *G'MIC*.

- Une autre amélioration notable concerne la ré-implémentation dans *G'MIC* d'un interpréteur de langage ***Markdown*** [W] étendu : le `gmd` (comme ***G'MIC Mark*d*own***). Ce moteur de rendu *Markdown* est maintenant celui utilisé pour produire la **documentation de référence** et les **pages de tutoriels**, visibles sur la page web du projet. Il est aussi utilisé pour l'affichage de l'aide sur le terminal lorsque l'on invoque *G'MIC* en ligne de commande (avec `gmic -h` par exemple). L'apparition de ce nouvel analyseur *Markdown* va donc de pair avec une amélioration globale de la documentation disponible pour le projet.

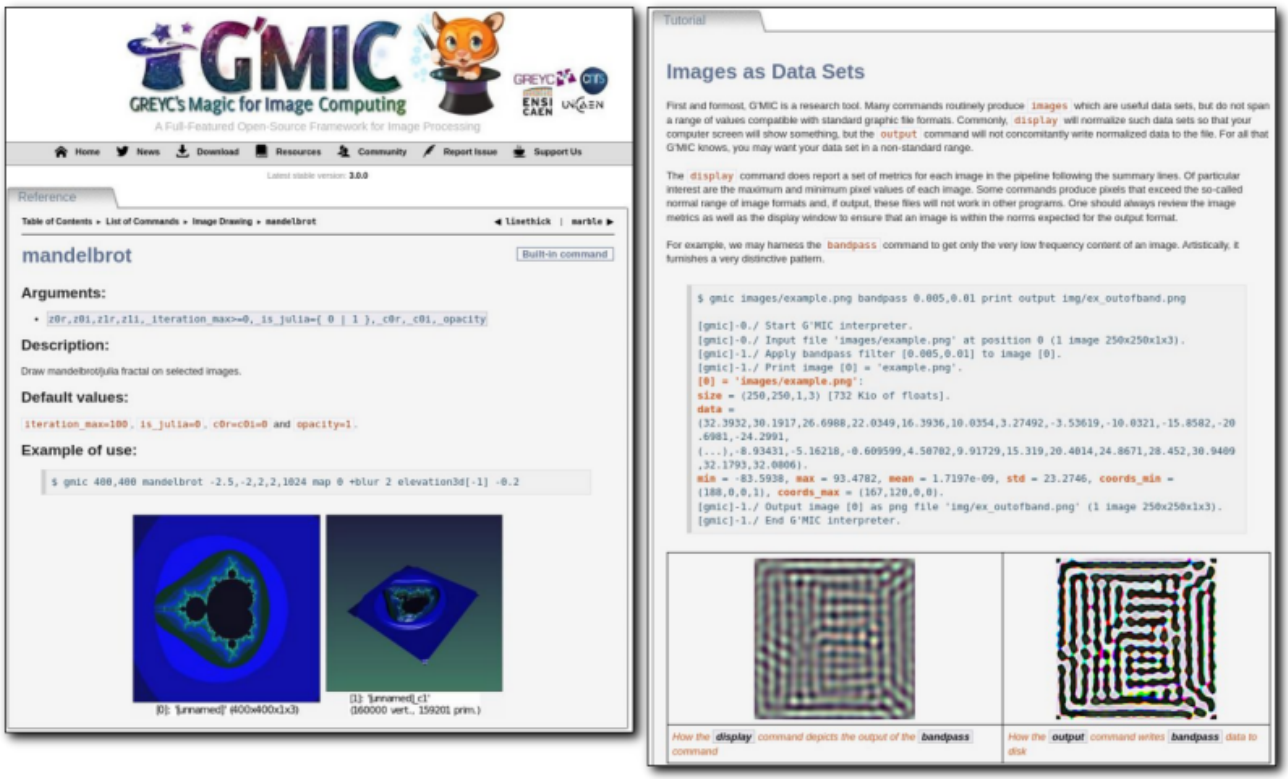


Fig.3.1. Le nouvel analyseur Markdown de G'MIC est utilisé pour le rendu de la documentation web de référence.

- La gestion de nouvelles palettes de couleurs personnalisées a été améliorée, avec l'introduction d'une nouvelle commande `palette` mettant à disposition 34 palettes de couleurs prédéfinies, chacune composée de 256 couleurs *RGB* (certaines étant récupérées depuis **CMOcean** et **LOSPEC**) :

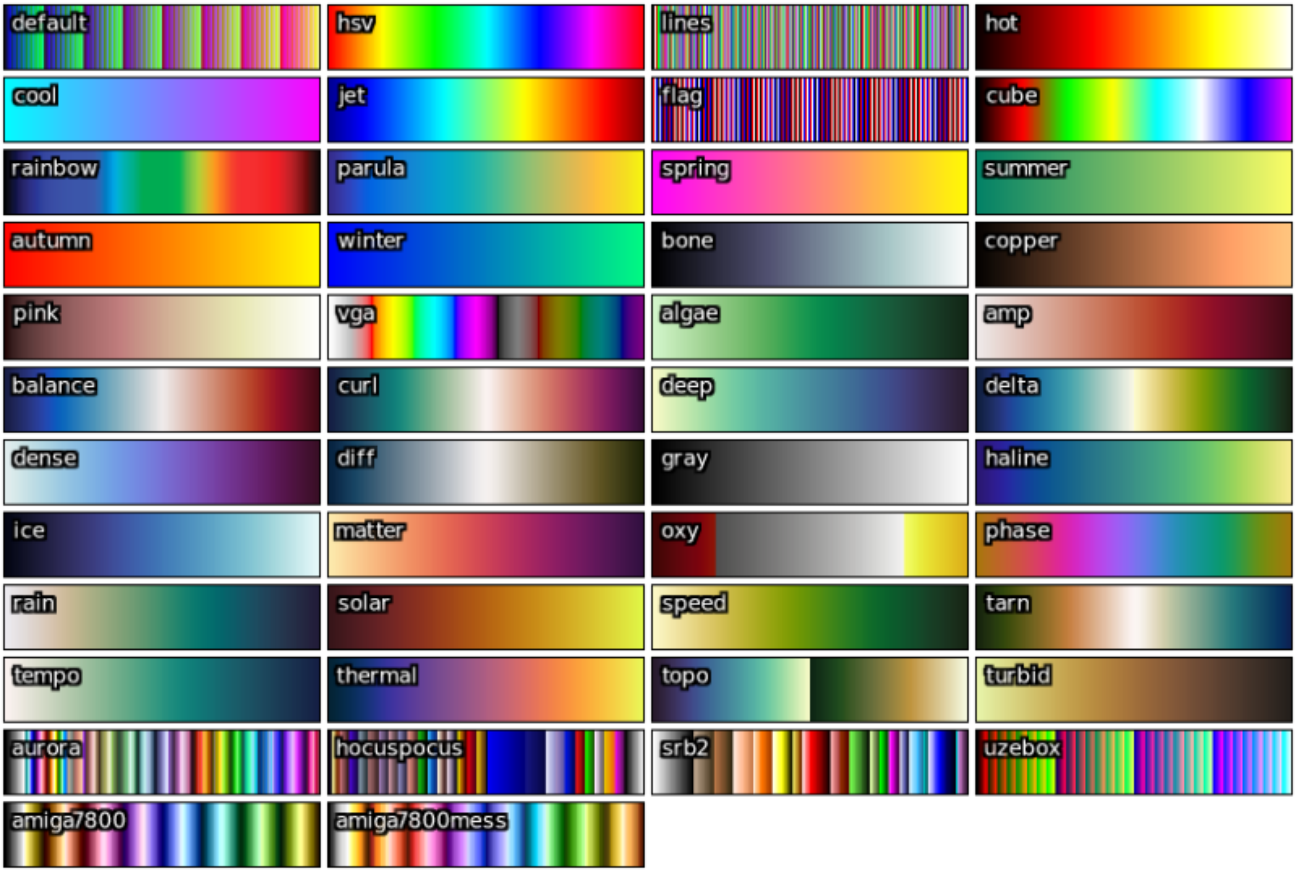


Fig.3.2. Ensemble des palettes de couleurs proposées par défaut dans G'MIC.

De plus l'utilisateur de l'interpréteur *G'MIC* a maintenant la possibilité de définir et nommer ses propres palettes de couleurs qui seront utilisables dans les commandes `map` ou `index`, servant à appliquer ces palettes sur des images couleurs. Les dites palettes sont généralement utiles à des fins de visualisation couleur de données scalaires. Par exemple, la colorisation d'une **fractale de Mandelbrot** [W] peut se faire très simplement en *G'MIC* avec la ligne de commande suivante :

```
$ gmic 600,600 mandelbrot -1.0132,-0.316356,-1.00227,-0.305418,512 map amiga7800
```

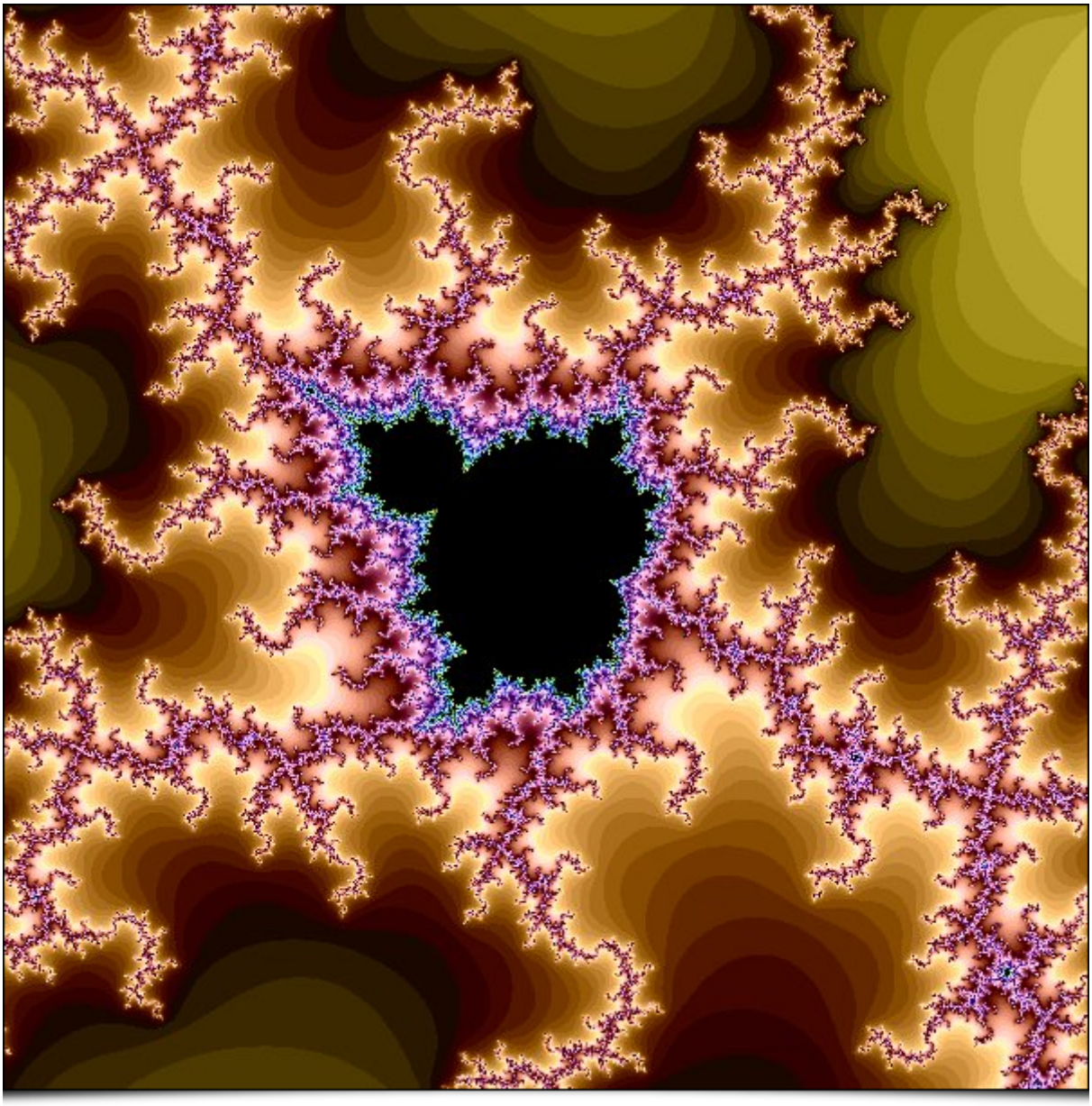


Fig.3.3. Résultat de l'application de la palette de couleur `amiga7800` *sur une fractale de Mandelbrot.*

- Au niveau du rendu d'objets maillés en 3D, deux nouveautés sont à signaler : d'abord la ré-implémentation du visualiseur 3D (commande `display3d`) qui a été ré-écrit directement en langage *G'MIC* et qui sera donc plus simple à maintenir et à faire évoluer.

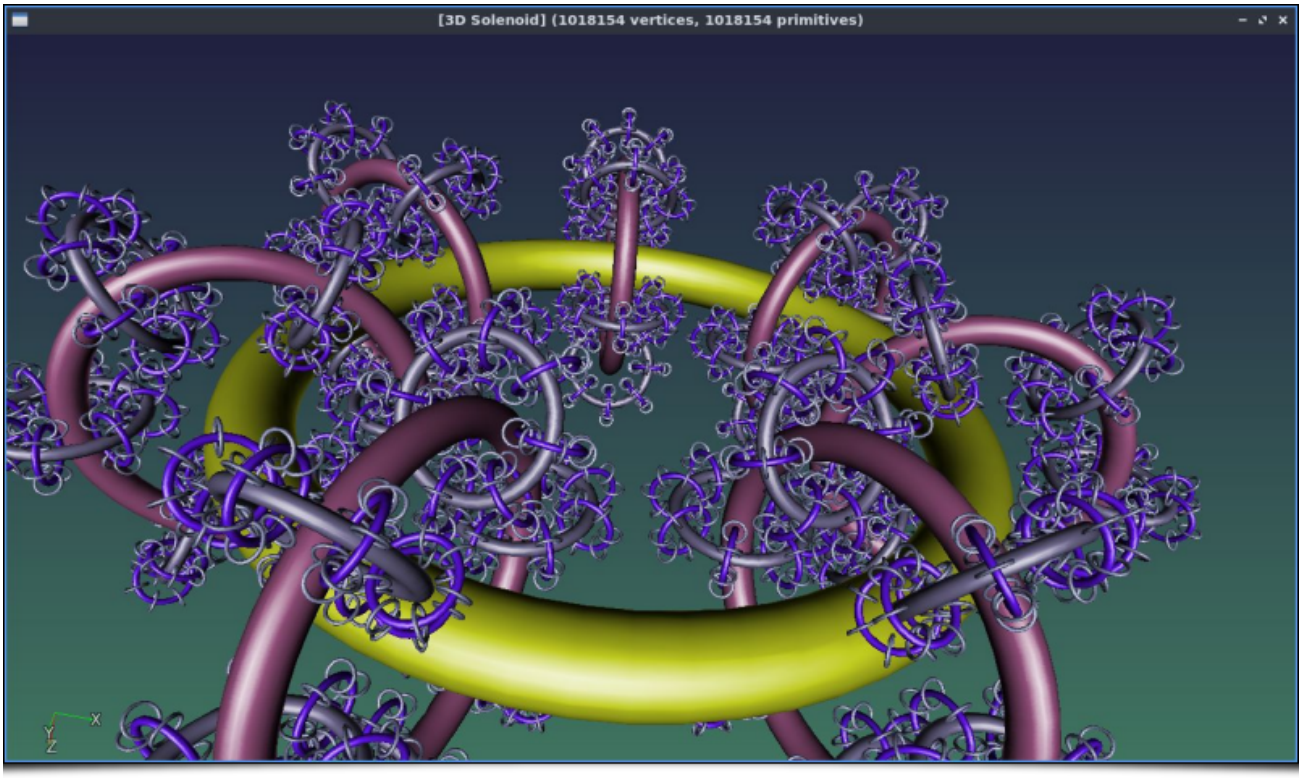


Fig.3.4. Aperçu du nouveau visualiseur 3D d'objets maillés de G'MIC.

Ensuite, notons la possibilité maintenant offerte d'exporter un objet 3D maillé construit dans *G'MIC* (typiquement avec une génération procédurale) sous la forme d'un fichier `.obj` en format ***Wavefront***[W], un format *ASCII* simple que la plupart des modeleurs 3D sait relire. Voici, illustré ci-dessous, un exemple d'importation d'un objet 3D généré récursivement sous *G'MIC* et ré-importé dans ***Blender***, grâce à l'utilisation de ce format de fichier *Wavefront*.

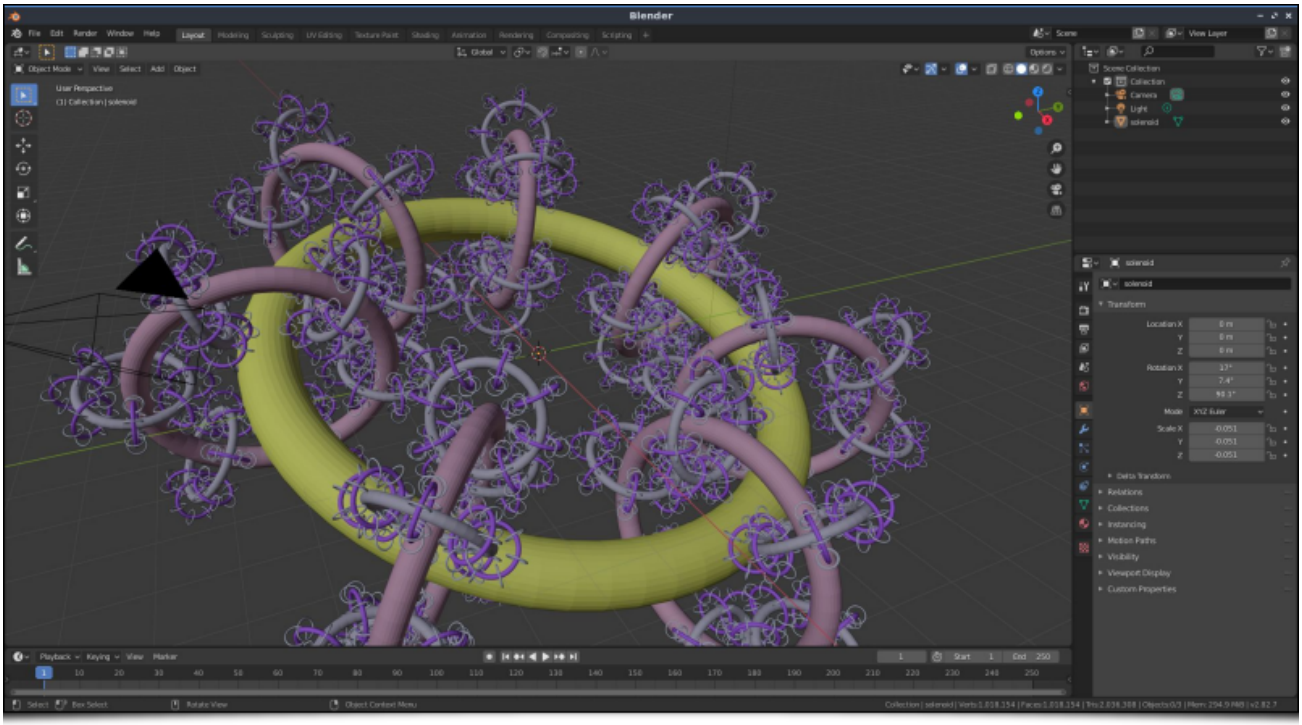


Fig.3.5. Importation dans Blender d'un objet 3D généré par G'MIC de manière procédurale.

- Toujours en restant dans le domaine des entrées-sorties, notons l'apparition ou l'amélioration du support de certains formats de fichiers, tels que ceux ayant les extensions `.png`, `.tiff`, `.csv`, `.webp`, `.arw`, `.cr2`, `.nef`, `.dng`, `.heic`, `.avif` (ces deux derniers n'étant cependant pas activés par défaut).

- Deux nouvelles commandes `lorem` et `portrait` apparaissent également et permettent de récupérer des images aléatoires à partir de deux services web différents : ***Lorem Picsum*** et ***ThisPersonDoesNotExist***. Elles sont assez pratiques pour tester des filtres rapidement sur des images quelconques, sans avoir à les stocker explicitement sur le disque dur. Par exemple, la commande :

```
$ gmic +lorem 800 +portrait 800
```

pourra vous afficher ceci :

Fig.3.6. Aperçu du résultat des nouvelles commandes `lorem` *et* `portrait`.

- Notons enfin que `gmic`, l'interface en ligne de commande de *G'MIC*, accepte désormais la définition de **points d'entrées** [W] dans des fichiers scripts d'extension `.gmic`, et que l'on peut donc rendre désormais de tels fichiers exécutables (avec le **Shebang** [W]). Par exemple, sous *Unix*, le fichier `test.gmic` suivant pourra être rendu directement exécutable et lancé depuis un terminal :

```
#!/usr/bin/env gmic
# File 'test.gmic'
echo[] "Salut les amis de Linuxfr !"
```

3.2. Améliorations de l'évaluateur d'expressions mathématiques

Le domaine du traitement numérique des images se base essentiellement sur des modélisations mathématiques des images et des algorithmes que l'on souhaite appliquer. *G'MIC*, en bon professionnel du traitement d'images, possède donc son propre évaluateur d'expressions mathématiques, et celui-ci a aussi vu son état général s'améliorer avec le passage à cette nouvelle version *3.0*. Pour résumer, c'est plus de 40 nouvelles fonctions mathématiques qui ont été implémentées dans cet évaluateur ces deux dernières années, afin d'améliorer ses capacités de calcul :

`store()`, `date()`, `begin_t()`, `end_t()`, `merge()`, `f2ui()`, `ui2f()`, `ccos()`, `csin()`, `ctan()`, `ccosh()`, `csinh()`, `ctanh()`, `lerp()`, `maxabs()`, `minabs()`, `argmaxabs()`, `argminabs()`, `da_size()`, `da_insert()`, `da_push()`, `da_remove()`, `da_pop()`, `da_back()`, `isvarname()`, `resize()`, `fill()`, `repeat()`, `set()`, `deg2rad()`, `rad2deg()`, `swap()`, `vargkth()`, `vargmin()`, `vargmax()`, `vargminabs()`, `vargmaxabs()`, `vavg()`, `vkth()`, `vmin()`, `vmax()`, `vminabs()`, `vmaxabs()`, `vmed()`, `vprod()`, `vstd()`, `vsum()`, `vvar()`, `string()`.

Ces fonctions ont été ajoutées au fur et à mesure des besoins et aujourd'hui, de plus en plus de filtres complexes de *G'MIC* en bénéficient. Un travail d'optimisation important a également été réalisé ces dernières années pour rendre cet évaluateur de plus en plus performant, notamment en favorisant le calcul parallèle lorsque cela est possible. Plutôt que de rentrer dans des détails techniques rébarbatifs sur l'implémentation et l'optimisation de ce module, je vous propose ici un exemple qui montre la façon dont on peut utiliser cet évaluateur dans un pipeline *G'MIC*. Ici, on cherche à produire une image de **Buddhabrot** [W], une représentation particulière de l'**ensemble de Mandelbrot** [W]. Ci-dessous à gauche, on peut voir le contenu du fichier `buddhabrot.gmic` définissant la commande `buddhabrot`, qui produit l'image de droite. L'évaluateur d'expressions mathématiques de *G'MIC* est appelé entre les lignes `9` et `27`, ce qui correspond donc à une unique expression. Comme on le voit, cet évaluateur est capable de compiler puis d'exécuter une expression qui se trouve être en réalité un vrai petit programme !

```
buddhabrot :
  repeat 3 _buddhabrot_mono[] ${"arg0 "$>",100,1000,3000"} done
  n 0,500 c 0,255 a c denoise_cnn 0 adjust_colors 0,10,10,0,30
  to_rgb

_buddhabrot_mono :
  1024,1024
  32,1,1,1,":
    begin(
      const itermax = $1;
      zs = vector(#itermax*2);
    );

    repeat (0.25*w#0*h#0,
      c = [ u(-2,1),u(-1.5,1.5) ];
      z = [ 0,0 ];
      for (iter = 0, cabs(z)<=2 && iter<itermax, ++iter,
        z = z**z + c;
        copy(zs[2*iter],z);
      );
      iter>0 && iter<itermax ?
        repeat(iter,k,
          x = lerp(0,w#0 - 1,(zs[2*k] + 2)/3);
          y = lerp(0,h#0 - 1,(zs[2*k + 1] + 1.5)/3);
          ++i(#0,x,y);
        )
    )"
  rm. rotate. 90
```

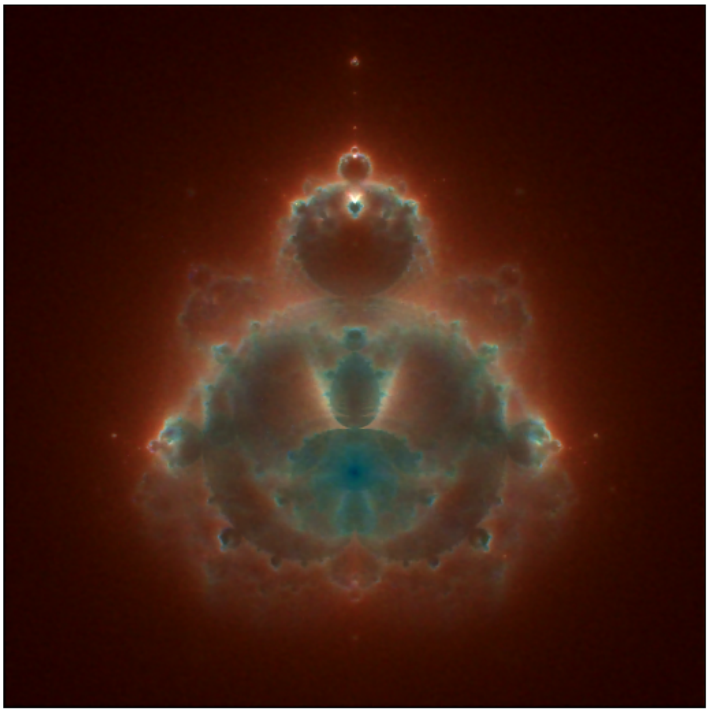


Fig.3.2.1. Génération d'une image de Buddhabrot avec un script G'MIC.

L'évaluateur mathématique est vraiment un élément essentiel du cadriciel *G'MIC* et continuera probablement d'évoluer au fil des prochaines versions.

4. Autres informations notables

En a-t-on terminé avec cette dépêche ? Pas encore ! Car travailler sur le projet *G'MIC*, c'est évidemment passer beaucoup de soirées à faire de la recherche, de la programmation et du test de nouvelles fonctionnalités, mais une part importante du temps passé concerne également la gestion des éléments annexes, à savoir : répondre aux questions sur les forums ou les forges *github*, trouver des financements pour faire avancer le projet plus rapidement, ou encore communiquer autour du projet (par exemple en écrivant des dépêches *LinuxFr.org* ☺ !). Et sur ces différents points, il s'est passé des choses appréciables, depuis fin 2019 :

- D'abord, nous avons eu le privilège d'obtenir deux financements (deux *CDD 12 mois*), dans le cadre de l'appel *« Soutien Plateforme »* de l'**Institut INS2I** du CNRS. Ces financements ont permis consécutivement le recrutement de deux ingénieurs de développement pour travailler sur le projet *G'MIC*. Le premier (2018-2019), on le connait tous ici, puisqu'il s'agit de **Jehan** qui écrit les fameuses dépêches *LinuxFr.org* sur *GIMP* (et qui est par ailleurs contributeur majeur de *GIMP*). Nous avions d'ailleurs décrit ses apports dans **notre dépêche précédente**.
- Le deuxième ingénieur (2019-2020), **Jonathan-David Schröder** a réalisé un travail conséquent pour l'implémentation d'un *binding G'MIC* pour le langage *Python* : **https://pypi.org/project/gmic/**. Ce *binding* est

pour le moment figé à une version antérieure (*2.9.2*) de *G'MIC* et il reste a priori quelques corrections à réaliser, mais c'est une interface qu'on aimerait bien mettre sur le devant de la scène. Espérons que cela suscitera l'intérêt de nouveaux contributeurs extérieurs, experts en *Python*, pourquoi pas ? En 2020, Jonathan-David a présenté l'avancement de son travail lors du ***Libre Graphics Meeting*** qui a eu lieu en ligne. Sa présentation est disponible en cliquant sur le lien suivant :

Dans les deux cas, ces financements ont été fructueux et ont permis d'ouvrir le projet *G'MIC* à d'autres horizons (sans parler du plaisir d'interagir avec ces deux ingénieurs de talent !).

- Une autre nouvelle notable concerne le volet « communication » autour du projet, avec le développement et la mise en place de la borne tactile ***Virtual Artist*** (« Artiste Virtuel » en français). Cette borne interactive montre le principe du transfert de style entre deux images. Elle permet d'initier le grand-public au domaine de l'algorithmique du traitement d'images, avec une application artistique directe et amusante. Elle est constituée d'une table tactile sur laquelle tourne un script *G'MIC* qui implémente à la fois l'interface utilisateur et l'algorithme de transfert de style démontré. La vidéo ci-dessous illustre le fonctionnement de cette borne (cliquez sur l'image pour ouvrir le lien) :

Virtual Artist constitue un support intéressant pour la médiation scientifique auprès du grand-public. Elle a été utilisée par exemple lors du ***FÉNO 2021*** (Festival de l'Excellence Normande qui s'est tenu à Rouen), sur le stand du CNRS.

Fig.4.1. La table tactile « Virtual Artist » du GREYC, en action, lors du FÉNO 2021.

- Et toujours pour rester dans le domaine de la communication, notons que *G'MIC* a eu droit à une petite séquence vidéo de présentation lors de la **Fête de la Science 2020**, réalisée par le service communication de la **délégation CNRS Normandie**. Cliquez sur l'image ci-dessous pour découvrir cette vidéo :

- Toujours en ce qui concerne la communication, nous voulions en profiter pour annoncer que la mascotte « Gmicky » du projet (le petit tigre magicien) avait été réalisée en crochet par Florence, de **Doomyflocrochet**. Florence réalise des versions crochet de **mascottes de projets libres** qui sont bien sympathiques, et qu'on vous invite à aller voir (et pourquoi pas, à commander).

Fig.4.1. « Gmicky », la mascotte du projet G'MIC, réalisée en crochet.

- Enfin, plus généralement, pour ceux qui souhaiteraient se plonger plus profondément dans la programmation d'algorithmes de traitement d'images, mentionnons la sortie du livre **« Le traitement numérique des images en C++ »** (Éd. Ellipses, 318 pages), que j'ai coécrit avec mes collègues de l'Université Clermont Auvergne, **Vincent Barra** et **Christophe Tilmant**, pendant la période de confinement de 2020. Vous y

trouverez une présentation de **CImg**, la bibliothèque C++ de traitement d'images (dont je suis l'auteur) et sur laquelle la majeure partie des fonctionnalités de *G'MIC* reposent, ainsi que des ateliers variés autour de différentes thématiques et applications du traitement d'images.

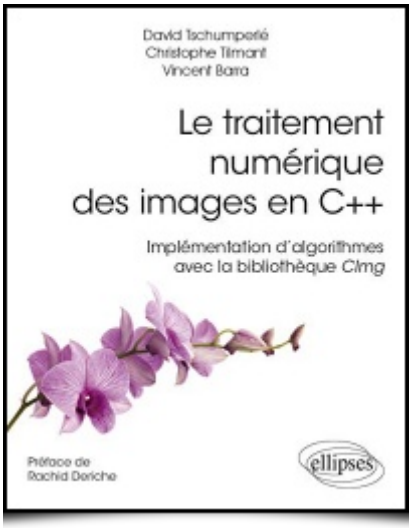


Fig.4.2. Un livre pour approfondir ses connaissances en algorithmique du traitement d'images, avec la bibliothèque CImg, sur laquelle est basé G'MIC.

5. Et ensuite ?

Depuis 2008, *G'MIC* n'a cessé d'être un projet actif, et ce, même si nous n'avons jamais défini de feuille de route très précise. Le développement de nouvelles fonctionnalités se réalise en fonction du temps et des opportunités de chacun, ainsi que des activités de recherche en traitement d'images que nous menons en parallèle dans notre équipe de recherche au laboratoire *GREYC*. Nous espérons seulement que ce temps ne finira pas inévitablement par nous faire défaut ! Avoir une activité de chercheur ou d'enseignant-chercheur semble parfois assez incompatible avec une activité de développement logiciel libre, qui est un travail à la fois très prenant et en même temps pas forcément très bien valorisé lors d'un suivi de carrière. Il peut donc être tentant de se focaliser sur des activités plus « rentables » d'un point de vue professionnel.

À titre personnel, je souhaiterais donc adresser des **remerciements appuyés** au **laboratoire *GREYC*** (notamment la direction, le service gestion et le service DDA), la **Délégation *CNRS* Normandie** (notamment le service Valorisation, le service Communication et le service *RH*), l'**Institut *INS2I*** du *CNRS*, l'association ***LILA***, mon collègue **Sébastien Fourey** (co-développeur principal du projet), les membres de l'association ***PIXLS.US*** (qui héberge **notre forum de discussion**) ainsi que tous les contributeurs au projet, les utilisateurs faisant des retours gentils, **utiles**, ou même **financiers**. Toutes ces personnes qui ont cru au projet à un moment donné et en la capacité de ses développeurs nous ont apporté leur soutien d'une manière ou d'une autre. Cela nous a permis de ne jamais abandonner le développement de *G'MIC*, depuis 2008. Qu'ils en soient vraiment remerciés !

Cette dépêche est quasiment terminée, merci d'avoir tenu jusqu'au bout ! J'essayerai autant que possible de répondre aux questions que vous pourrez poser dans les commentaires.

6. Ressources complémentaires

Pour vraiment clore cette dépêche et contenter les lecteurs encore réveillés et qui pourraient être restés sur leur faim, voici une petite sélection de vidéos à regarder, pour occuper les longues soirées d'hiver qui arriveront dans les prochaines semaines :

- Découverte de G'MIC pour GIMP (en français) : *« GIMP 2.10.22 : Tuto 198 (Filtres G'Mic) »*.
- Découverte de *G'MIC* pour *Affinity Photo* (en français) : *« Tutoriel sur l'installation et l'utilisation du plugin G'MIC pour Affinity Photo »*.

- Découverte de *G'MIC* (en anglais) : « G'MIC—Free Image Manipulation Powerhouse ».
- Découverte de *G'MIC* (en anglais) : *« G'MIC: An Amazing, Free Plugin »*.
- Présentation de différentes techniques pour transformer une photographie en *cartoon* (en anglais) : *« G'mic for Krita - Three ways to turn a photograph into a cartoon »*.
- Présentation de G'MIC pour Krita (en anglais) : *« G'mic for Krita - Step by Step tutorial on how to use this amazing FREE image manipulation plugin »*.
- Présentation *G'MIC* au *Libre Graphics Meeting 2021* (en anglais) : *« How to make 890+ Color LUTs fit in 3.3Mb ? »*.
- Utilisation du filtre **Degradations / Rain & Snow** de *G'MIC* (en français) : *« Tutoriel Krita - Comment dessiner la pluie dans Krita avec G'MIC »*.
- Utilisation du filtre **Artistic / Hough Sketch** de *G'MIC* (en français) : *« Amusez vous avec Hough Sketch de G'MIC dans GIMP »*.
- Utilisation du filtre **Artistic / Stylize** de *G'MIC* (en anglais) : *« Using the G'MIC Stylize filter in Krita with flat texture patterns »*.
- **Note finale** : cette dépêche est un résumé (si, si !) des notes de versions détaillées suivantes : **notes de version *2.8***, **notes de version *2.9***, **notes de version *3.0***.

Aller plus loin

- **Le projet G'MIC** (122 clics)
- **Fil Twitter des nouvelles du projet** (11 clics)
- **Série d'articles G'MIC sur LinuxFr.org** (37 clics)

Hype manquante

Posté par **cosmocat** le 17/12/21 à 22:24. Évalué à 3.

> Dépêche très intéressante mais il me semble qu'il manque une fonctionnalité indispensable pour être au summum de la hype:
>
> le support du format QOI ! ⬤
>
> **https://qoiformat.org/**

Re: Hype manquante

Posté par **Pierre Jarillon** (**site web personnel**) le 17/12/21 à 22:31. Évalué à 3.

> > *il manque une fonctionnalité indispensable pour être au summum de la hype:*
> > *le support du format QOI ! ⬤*
>
> Bon, tu sais maintenant ce que tu dois faire : le coder !

Re: Hype manquante

Posté par **cosmocat** le 18/12/21 à 00:35. Évalué à 3.

> Après 20 ans sans avoir fait de C++, je suis pas sûr qu'ils voudraient de ma contribution ⬤ (et je suis aussi sur que quelqu'un aura déjà fini, juste le temps que j'arrive à installer/configurer tout le nécessaire pour