Rapport du projet Flamefract

Valérian Rousset Guy-Laurent Subri

2 juin 2013

Améliorations

Chaque sous-section est séparée en deux parties, la première est la description de l'amélioration d'un point de vue de l'utilisateur, la second parle de l'implemtation en java.

Barre de chargement

Nous avons ajouté une barre de chargement sous l'image de la fractale. Cette barre s'affiche uniquement lorsque la fractale est calculée. Elle est discrète et utile pour l'utilisateur, d'autant plus que celui-ci voit sa fractale apparaître petit à petit (amélioration expliquée plus tard).

L'implémentation est facile à mettre en place après l'ajout de l'apparition incrémentale de la fractale ; comme nous calculons une serie de nouveau points à chaque rafraichissement de l'interface, nous avons le nombre total de points calculé jusqu'à lors et le nombre total de point à calculer, nous avons donc un ratio. Il suffit après de dessiner un rectangle en bas de la fractale après chaque rafraichissement, et de ne plus en faire quand le fractale est finie.

Apparation "incrémentale" de la fractale

L'apparition "incrémentale" de la fractale permet à l'utilisateur de voir la fractale se déssiner petit à petit devant ses yeux émerveillé par la beauté de l'image qu'il a crée (ou presque) ! Avant cette feature, nous devions attendre, sans savoir quoi que ce soit sur l'état du calcul, avant de voir apparître brusquement une nouvelle fractale (en particulier si on met le GUI en plein écarn ...). Cette feature est plus "user-friendly" qu'autre chose, mais elle est assez élégante. De plus, l'interface se bloquait le temps de calculer la fractale, ce qui est en général très dérangeant pour l'utilisateur lambda.

L'idée est d'avoir un Timer (de swing) pour gerer le lancement périodique du rafraichissement, avec une valeur décidée par l'utilisateur (voir amélioration "Fichier de configuration") ou calculée dynamiquement avec une algorithme adaptif. Cet algorithme est simple, on calcul un premier set de point, plus petit, on regarde combien de temps cela nous a pris, et nous augmentons le nombre de points à calculer en fonction du temps possible d'utilisateur ; a l'inverse, si nous prennons trop de temps, on reduit le nombre de point calculé. Si, même avec un très faible nombre de points, nous n'arrivons pas à mettre à jour assez vite, nous augmentons le temps entre chaque rafraichissement. Maintenant, la partie incrémentale : l'idée est de relancer la fonction de rafraichissement tant que le nombre total de point n'est pas dessiné, via un Timer. Il suffit après de gérer les différents cas : s'il le Timer ne tourne plus, nous le créons et le lanceons, de plus nous créons un nouveau FlameAccumulator.Builder que nous gardons dans la class ; s'il nous reste des points à calculer, nous en calculons un autre set et l'affichons ; si le nombre total de point est calculé, on arrête le Timer. Dernier cas, si la taille de l'accumulateur n'est pas le même que celui demandé par swing, on arrête le calcul.

Fichier de configuration

Tous les deux, nous aimons avoir le contrôle sur notre système et sur les programmes que nous utilisons, et nous apprécions particulièrement le fait de pouvoir configurer les programmes à notre guise. C'est pourquoi nous avons mis en place un fichier de configuration ! Dans ce fichier, nous pouvons changer la matrice de base, les poids accordés aux différentes transformations, la "density" de l'image, ...Le fichier de configuration supporte les commentaires. Nous avons la possibilité d'avoir des valeurs "aléatoires" (pas pour tous les champs configurables, comme par exemple le refresh rate, car il évident qu'avoir un refresh rate aléatoire est insensé).

En utilisant un patron de type Builder, nous avons une classe chargeant tout le contenu du fichier de preférence, le parse et en crée un instance de Preferences. La mise en place du parsing est plutôt simple : nous lisons chaque ligne du fichier "flamefract.conf", chaque ligne est soit un commentaire, soit une variable avec une valeur avec ou sans commentaire postfixant. Enlever les commentaires et les éspaces non-necessaires sont deux regex chainé, un test pour voir si la chaine resultant est vide autrement, un gros switch. Dans ce switch, si nous avons une valeur connue, nous assignons la bonne variable avec la valeur que nous parsons, autrement, nous affichons un message d'erreur et quittons. Si le magic word "random" apparait dans le champ de valeur, chaqu'une des fonctions le supportant génére une valeur aléatoire pour ce qu'il doit retourner. Par design, il est facile d'ajout de nouveaux parametres, car la plus par des fonctions tel que parseDouble, parseRectangle, parseColor sont implementée et retourne la respectivement un double, un rectangle et une couleur. L'utilisation du fichier est décrite simplement dans le fichier par défaut crée s'il n'existe pas déjà, en particulier que chaque nouvelle matrix dans le fichier s'ajoute au constructeur au lieu d'écrasser l'ancienne valeur comme pour le reste des variables.

Menus et raccourcis

Nous avons décidé d'implémenter des menus et des raccourcis clavier (car nous aimons pouvoir gérer le maximum de choses sans avoir besoin de la souris). Nous avons plusieurs menus, qui permettent d'afficher d'autres fonctionnalités (terminées ou non), comme, par exemple, la sauvegarde de l'image, la sauvegarde de la configuration actuelle du programme (dans le fichier de configuration sus-mentionnée) ou le plein-écran.

Les menus principaux ("Fichier" et "Affichage"), ainsi que leur contenu sont des enums, facilitant l'ajout et la supression, un switch se faisant pour decider quelle action en fait pour chaque sous-menu.

« en bas, sous-section de menu/raccourcis »

Sauvegarde de l'image Générer des fractales dans une fenêtre créée avec Swing, c'est bien, mais avoir cette fractale en fond d'écran, c'est mieux. C'est pourquoi nous voulions, dès le départ, avoir la possibilité de sauver les images que nous avons créées.

« en bas, sous-section de menu/raccourcis »

Sauvegarde de la configuraiton de l'image L'idée de sauver la configuration (c'est-à-dire le poids des variations, le nombre de transformations, etc...) est venue du fait que si nous enregistrions une image, nous ne pouvions pas reprendre l'image et modifier quelques paramètres. Nous nous sommes donc dit qu'il nous *fallait* pouvoir reprendre la configuration d'une image.

« en bas, sous-section de menu/raccourcis »

Plein écran

Nous voulions avoir la possibilité de mettre notre image en plein écran, pour plusieurs raisons :
- C'est pratique, pour voir les détails de la fractale
- On a une meilleure idée du résultat de la sauvegarde de l'image
- Ça le fait...

Divers

Réinitilisation

Nous nous sommes dit qu'il était pratique de pouvoir revenir à l'image initiale sans avoir à fermer et relancer le programme. Nous avons donc mis en place un menu et un raccourci clavier (Ctrl-N) qui réinitialisait les paramètres du programme.