Arnold von Bauer Gauss
Yasmine Cherif

PTI 2013
Rapport sur le bonus du projet

Notre groupe a ajouté les fonctionnalités suivantes au projet :

- Une liste de fractales prédéfinies
- Changer la fractale courante parmi les fractales de la liste
- Ajouter une nouvelle fractale à la liste
- Supprimer une fractale de la liste
- Renommer une fractale de la liste
- Copier une fractale de la liste
- Sauvegarder la liste de fractale courante dans un fichier (nommé flamelist.sav) le fichier contient toutes les fractales de la liste avec leur nom, leur liste de FlameTransformation, leur frame (Rectangle) et leur palette
- Ouvrir une liste de fractale (actuellement une seule liste qui est le fichier flamelist.sav), et la restaurer dans le programme
- Remettre la liste de fractales par défaut, qui contient la fractale Shark et la fractale Turbulence

La mise en œuvre de la liste de fractales a nécessité des changements dans la structure du programme.

Pour pouvoir créer une liste de fractales prédéfinies, nous avons premièrement créé une classe PresetFlame en utilisant le patron Decorator : la classe PresetFlame possède un attribut nom pour le nom de la fractale, un ObservableFlameBuilder, un rectangle frame, une palette, une JList pour la liste de FlameTransformation, et un int pour l'index de la FlameTransformation selectionnée.
L'idée de cette classe est d'y transposer le patron observer mis en place à l'origine pour la liste de transformations : désormais, chaque fractale possède sa propre liste de transformations et donc sa propre JList associée. PresetFlame possède toutes les méthodes de ObservableFlameBuilder, ainsi que quelques méthodes supplémentaires utiles au bonus.

Au niveau de FlameMakerGUI, le patron Observer est aussi appliqué mais cette fois pour mettre à jour la sélection de la fractale courante, de la même manière que nous l'avons fait dans la version standard pour la liste de FlameTransformation; en revanche l'observateur de selectedFlameIndex (index de la flame sélectionnée) doit aussi mettre à jour la fractale contenue dans les composants AffineTransformation et FlameBuilderPreview, grâce aux méthodes setNewFlame() qui ont été ajoutées à chacune de ces classes (voir ligne 400).

L'observateur met aussi à jour la liste de FlameTransformation en récupérant celle-ci depuis la PresetFlame, et en la chargeant dans le JScrollPane de la liste de transformations (raison pour laquelle celui-ci est devenu un attribut de la classe, à l'instar des champs de variation des poids et du JScrollPane de la liste de Flames).

Comme une JList est également utilisée pour la liste de Flames prédéfinies, la classe FlameMakerGUI accueille également un AbstractListModel pour gérer la liste. Le modèle AbstractListModel des FlameTransformation est déplacé dans la classe PresetFlame.

Après cette amélioration implémentée, nous avons également créé un menu en utilisant la partie NORTH du BorderLayout qui n'était utiliée que par un label. Ce menu comporte trois boutons : Ouvrir, Sauvegarder, Par défaut.

Pour sauvegarder et ouvrir une liste de PresetFlames, nous avons opté pour l'écriture et la lecture dans un fichier à l'aide de ObjectInputStream et ObjectOutputStream. Ces streams sont très simples à utiliser pour sauvegarder et restaurer des objets, à condition que ceux-ci soient « serializables ». Pour cela, nous avons utilisé le patron Adapter en créant une classe StorableFlame, qui contient seulement l'essentiel de la PresetFlame (la partie structurelle de celle-ci), et qui implémente l'interface Serializable. Il a également fallu faire implémenter cette interface à tous les attributs de StorableFlame, et donc les classes FlameTransformation, Affinetransformation, Color, InterpolatedPalette, RandomPalette et Rectangle ont toutes été modifiées en ce sens.

La mise en place du bonus aurait dû préfigurer d'autres améliorations telles que la sauvegarde de la fractale en image, la possibilité de modifier et personnaliser la palette et le rectangle frame, et la possibilité de générer une fractale aléatoire, mais le manque de temps nous a dissuadé d'intégrer ceux-ci à travers l'interface. La possibilité de créer une fractale aléatoire et de personnaliser la palette ayant été auparavant implémentés, les paramètres étant configurables à travers la console, nous vous montrons les morceaux de codes concernés ici :

Arnold von Bauer Gauss
Yasmine Cherif

Fractale aléatoire:

```
Random randy = new Random();
ArrayList<FlameTransformation> listTransfos = new
ArrayList<FlameTransformation>();
int nb = 0;
while (nb < 1) {
	System.out.println("Selectionnez le nombre de transformations a
	inserer dans la fractale :");
	nb = scanner.nextInt();
	if (nb < 1)
		System.out.println("Le nombre de transformations doit etre
strictement positif !");
}

for (int i = 0; i < nb; i++) {
	double[] coeffs = new double[6];
	for (int j = 0; j < 6; j++) {
		double value = randy.nextDouble();
		coeffs[j] = randy.nextBoolean() ? value : -value;
	}
	listTransfos.add(new FlameTransformation(
	new AffineTransformation(coeffs[0], coeffs[1], coeffs[2],
	coeffs[3], coeffs[4], coeffs[5]),
	new double[] { randy.nextDouble(), randy.nextDouble(),
	randy.nextDouble(), randy.nextDouble(), randy.nextDouble(),
	randy.nextDouble()}));
}

PresetFlame f = new PresetFlame("Random", listTransfos,
	new Rectangle(new Point(0, 0), 5, 5), generatePalette(0));
flameList.add(f);
setSelectedFlameIndex(flameList.size() - 1);
```

Palette personnalisée:

```
private Palette generatePalette(int mode) {
ArrayList<Color> colorList = new ArrayList<Color>();
while (mode < 1 || mode > 3) {
	System.out.println("Selectionnez une option:");
	System.out.println("1. Palette classique {R, G, B}");
	System.out.println("2. Palette aleatoire");
	System.out.println("3. Palette personnalisee");
	mode = scanner.nextInt();
	if (mode < 1 || mode > 3)
		System.out.println("Selection invalide !");
}

switch (mode) {
case 2:
	System.out.print("Veuillez entrez le nombre de couleurs a
inserer dans la palette: ");
	int sizeR = scanner.nextInt();
```

```
            return new RandomPalette(sizeR);
        case 3:
            int sizeC = 0;
            while (sizeC < 1) {
                System.out.println("Combien de couleurs souhaitez-vous
inserer dans la palette ?");
                sizeC = scanner.nextInt();
                if (sizeC < 1)
                    System.out.println("Le nombre de couleurs doit etre
strictement positif !");
            }

            for (int i = 0; i < sizeC; i++) {
                double r = -1, g = -1, b = -1;
                System.out.println("Couleur n°" + (i + 1));
                while (r < 0 || r > 1) {
                    System.out.print("Indiquez la composante n°1 de la
couleur a ajouter (0-1): ");
                    r = scanner.nextDouble();
                    if (r < 0 || r > 1)
                        System.out.println("La composante n'est pas
valide ! (0-1)");
                }
                while (g < 0 || r > 1) {
                    System.out.print("Indiquez la composante n°2 de la
couleur a ajouter (0-1): ");
                    g = scanner.nextDouble();
                    if (g < 0 || g > 1)
                        System.out.println("La composante n'est pas
valide ! (0-1)");
                }
                while (b < 0 || b > 1) {
                    System.out.print("Indiquez la composante n°3 de la
couleur a ajouter (0-1): ");
                    b = scanner.nextDouble();
                    if (b < 0 || b > 1)
                        System.out.println("La composante n'est pas
valide ! (0-1)");
                }
                colorList.add(new Color(r, g, b));
            }
            return new InterpolatedPalette(colorList);
        default:
            return InterpolatedPalette.generateRGBPal();
        }
```

Bugs connus :

- Lorsqu'on supprime l'avant-dernière fractale Flame de la liste, le bouton supprimmer est désactivé, cependant, si on charge une nouvelle liste de Flames (Avec Ouvrir ou Par défaut par exemple), le bouton ne se réactive pas malgré que la taille de la liste soit plus grande que 1. Ceci aurait pu être corrigé en mettant le bouton remove en attribut car dans l'etat il est inatteignable par l'observateur puisque niché dans une méthode.