IHM : Projet 1 : Interface d'optimisation du choix de couleurs

binôme : Benjamin Ruytoor et Aurore Allart
date : 21 mars 2013

Introduction

Dans ce projet, il nous a été demandé une interface permettant à un utilisateur d'obtenir des couleurs ayant un niveau de gris différent l'un avec les autres. Un des scénarios possible pour cette interface est de pouvoir imprimer sur une imprimante en niveau de gris.

Dans ce rapport, nous allons expliquer et justifier le choix de notre interface ainsi que les différentes techniques qui ont été implémentées.

1. Les interfaces

Tout d'abord, voici un aperçu de l'interface pour un choix de 6 couleurs. La première fenêtre nous permet de choisir le nombre de couleur que l'on désire obtenir. On a le choix entre 2 à 10 couleurs incluses. Par défaut, la valeur est au minimum, c'est à dire 2.

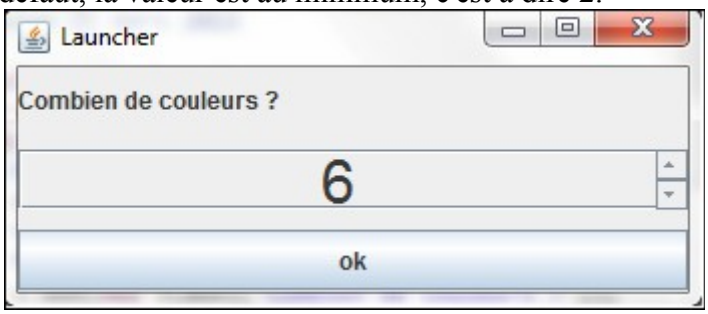

Figure 1 : launcher permettant de choisir le nombre de couleur.

Avec les flèches à droite de la fenêtre, il est possible de choisir un nombre limité de couleur (de 2 à 10). Ensuite, en appuyant sur le bouton « ok », on arrive sur la deuxième fenêtre, qui est le choix des couleurs.

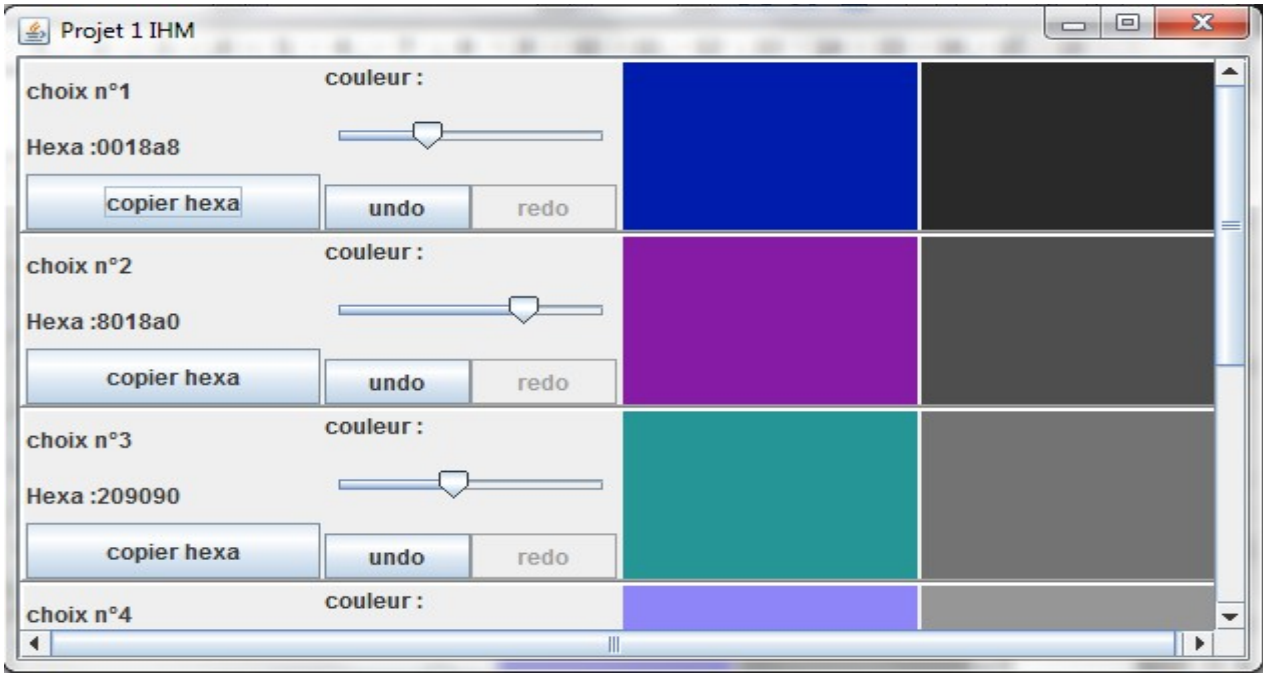

Figure 2 : choix des couleurs.

Tout d'abord, un choix de couleur nous est proposé par défaut. Ensuite, si les couleurs ne conviennent pas à l'utilisateur, il peut utiliser le coulisseau pour sélectionner une couleur qui lui convient. Cette nouvelle couleur sera de même niveau de gris que la précédente avec une marge de plus ou moins 5.

De plus, il est possible de revenir en arrière ou en avant par rapport à une couleur, grâce à la technique Undo/Redo. Enfin, il est possible de copier la valeur en hexadécimal de la couleur en cliquant sur le bouton « copier hexa ».

De base, on ne voit pas toutes les couleurs, mais il est possible d'agrandir la fenêtre. De plus, on peut cliquer une première fois sur le bouton Undo, cela, nous amène sur la couleur en niveau de gris.

Il est possible de choisir presque toutes les couleurs grâce à la représentation en HSB. Cette représentation est un espace colorimétrique définit en fonction de 3 composantes : la teinte suivant l'angle qui lui correspond sur le cercle des couleurs (par exemple pour le rouge, on a un angle de 0° ou 360°), la saturation c'est l'intensité de la couleur (varie entre 0 et 100%) et la brillance de la couleur (varie entre 0 et 100%).

Pour éviter trop de calcul, on ne prend pas en compte la brillance et on divise par 8 le nombre de possibilité de couleur. Maintenant, on dispose d'un grand nombre de possibilité (entre 300 000 à 1 millions).

2. Les technologies

Dans ce projet, nous avons implémenté les 2 techniques qui nous semblaient utile dans ce projet : le copier-coller et undo/redo. Pour les autres, nous allons vous expliquer pourquoi nous ne les avons pas pris.

- Drag and Drop : cette technique n'est pas mal, sauf qu'elle a un inconvénient principal, c'est qu'il faut que le logiciel receveur a déjà implémenté cette technique de la même façon qu'on l'a implémenté. Par exemple dans notre cas, il faut qu'il puisse recevoir une valeur en hexa, et non l'image de la couleur.
- Création de nouveaux composants : cette méthode peut être incorporé au projet, mais elle demandait beaucoup trop de travail, par rapport au temps imparti. Un exemple de composant sera proposé dans la 4^{e} partie de ce rapport.
- Reconnaissance de gestes : cette technique qui est principalement utilisé sur des tablettes, n'est pas utile dans notre interface car il serait difficile d'identifier la couleur que l'on désire en faisant des gestes avec la souris. Une possibilité aurait été de faire des gestes de passer à la couleur suivante ou précédente, mais le coulisseau le fait déjà et est beaucoup plus rapide.

3. Les difficultés rencontrées

Au cours de ce projet, nous avons rencontré une difficulté principale : celle de trouver la couleur associée au niveau de gris choisi. Au départ, on était parti sur 3 sliders pour les 3 composantes couleurs. Par exemple si on bougeait le rouge, on repartait du calcul du niveau de gris pour trouver les valeurs pour le vert et le bleu, en évitant de changer la valeur de gris. Nous avons constaté que cette méthode ne fonctionnait pas très bien.

C'est pourquoi nous nous sommes orientés sur le calcul avant l'affichage de la deuxième fenêtre, de tous les cas de couleurs avec les nombres de couleurs choisis.

Tout d'abord, on calcule pour chaque composante couleur avec un pas de 8, c'est à dire 3 boucles imbriquées de 0 à 256. Ensuite, on cherche à quelle liste de niveau de gris cette valeur, avec une marge de plus ou moins 5, appartient et enfin on la stocke dans cette liste. Cela nous donne ce code :

```
for(int r=0;r<256;r+=8){
	for(int g=0;g<256;g+=8){
		for(int b=0;b<256;b+=8){
			MyColor c=new MyColor(r, g, b);
			int gris=calculeGreyLvl(c);
			for(int i=1;i<nbCouleur-1;++i){
				if((i*255/(nbCouleur-1)<gris+5)&&(i*255/(nbCouleur-1)>gris-5)){
					tab.get(i).add(c);
				}
			}
		}
	}
}
```

4. Ce qui peut être amélioré

Tout d'abord, la technique de la création de nouveaux composants. Par exemple : on pourrait incorporer une fenêtre au dessus du slider, qui affiche le dégrader de couleur disponible pour ce niveau de gris.

Ensuite, on pourrait classer les couleurs du plus clair au plus foncé. Pour l'instant ce n'est pas le cas.

Enfin, on pourrait améliorer l'ergonomie de l'interface.

Conclusion

On peut dire que c'était plus difficile de créer une interface IHM qu'au premier abord.