



20/02/2013

## CubeSTA – Projet d'ISN

AUXENCE ARAUJO – JULIEN GARDET – YANN DROY

Notre projet : **CubeSTA** (Cube Scramble Timer Average), est porté sur le Rubik's Cube. Nous l'avons programmé avec l'IDE Netbeans. Le projet sera composé de fonctions qui permettront premièrement de générer une série de mouvement à faire pour pouvoir mélanger le Rubik's Cube, une fois mélangé, l'utilisateur, prêt à débiter la résolution du casse-tête, appuiera sur la touche espace pour pouvoir lancer une autre fonction *Timer* qui lancera un chronomètre et s'arrêtera avec un second appui sur la touche espace. Enfin, une dernière fonction nommée *Average* permettra de faire une moyenne élaguée des temps réalisés, c'est-à-dire que l'on supprime le meilleur temps réalisé et le moins bon pour faire une moyenne des temps restants. On proposera à l'utilisateur de faire un average de 5 ou de 12. Nous envisageons de faire un affichage graphique complet (voir annexe n°1). De plus, pour le travail en équipe, nous avons utilisé le site *GitHub* où nous pouvons enregistrer et charger toutes les modifications réalisées afin de faciliter le travail de groupe.

Pour commencer, l'ensemble du programme est constitué de plusieurs classes, elles mêmes rangées dans des packages (groupes de classes, on peut aussi les associer à des dossiers). Premièrement, il y a le package du mélange nommé : *edu.cubesta.scramble* où se situe les classes (On peut comparer une classe à un moule, c'est-à-dire qu'on appel une classe en lui renvoyant des données, « la pâte » et celui-ci permet de donner un résultat « la forme du moule ») *AlgoMaker*, *CubeGUI* et *TurnCube*. Ensuite, il y a le package intitulé *edu.cubesta.windows* qui, elle, permet un affichage graphique : la fenêtre du programme. Enfin, nous avons le package nommé *edu.cubesta* qui lui est composé de la classe nommée *CubeSTA* qui elle appelle les autres classes. C'est la classe principale.

Ici on peut distinguer les différents Packages et classes présents dans le programme.

Les « dossiers » sont les packages et les « fichiers » sont les classes.



Pour le bon fonctionnement et l'efficacité de notre travail d'équipe, nous utilisons donc le site *GitHub* (voir annexe n°2), qui est plus précisément un gestionnaire de versions collaboratives. Nous y avons créé la page de notre projet ainsi que toute une arborescence de pages : celles-ci nous servent à l'organisation. Notre projet peut être consulté par tout le monde et téléchargé par tout le monde. Le site dispose aussi d'un gestionnaire d'issues qui nous permet de pouvoir signaler les bugs, ou des innovations dans le programme. Il y a aussi une fonctionnalité permettant d'assigner un travail à une personne et aussi de pouvoir grouper les issues sous forme de "*milestone*" (jalon).

De plus nous utilisons la *javadoc* (voir annexe n°3) pour générer (sous forme HTML) depuis les commentaires du code source une documentation sur le fonctionnement des fonctions du programme.

Enfin, il existe une page web sur le projet qui permet d'accéder à la *javadoc* du projet ainsi que sa licence et le fichier .zip ou .tar.gz du projet.

Voici le lien vers la page web du projet: <http://cubesta-project.github.com/CubeSTA>

Les trois premières classes que nous avons codé sont celles qui génèrent le cube, l'algorithme et le mélange du cube. L'initialisation du cube se trouve dans la classe « *CubeGUI* », la création de l'algorithme se situe dans la classe « *AlgoMaker* » et le mélange dans « *TurnCube* ».

→ La classe *AlgoMaker* :

1. Tout d'abord, l'algorithme initialise une chaîne de caractères (R, L, U,...) qui indique les mouvements à exécuter pour mélanger le Rubik's Cube, ensuite, une nouvelle chaîne de caractère indique le sens de ce mouvement par exemple le « 2 » qui veut dire le faire deux fois.

2. Les variables temporaires permettent qu'un mouvement ne se répète plusieurs fois ce qui annulerait le mouvement précédent.

3. La boucle For indique la condition que pour *i* commençant à 1 et allant à *number* (le nombre de mouvement du scramble) il faut générer un nombre aléatoire qui permet de sélectionner au hasard un caractère des chaînes de caractère initialisées précédemment. La fonction *RandomBW* :

```
public static int randomBW (int min, int max){  
    int x = (int) (min+Math.random()*(max-min+1));  
    return x;  
}
```

permet de générer l'entier aléatoire correspondant à un caractère de la chaîne de caractère.

4. Enfin, la boucle While permet d'éviter la répétition d'un mouvement, et rappelle les variables temporaires pour cela.

→ La classe *CubeGUI* :

1. Cette classe permet, dans un premier temps d'initialiser la tableau de la variable d'affichage du cube, c'est-à-dire, qu'il va positionner les couleurs à leur position initial (la face blanche complète, puis la face bleu, etc ...).

```
public CubeGUI() {
    cubeGUI = new char[Character.MAX_VALUE][10];
    char[] colorList;
    colorList = new char[]{'G','W','Y','O','R','B'};
    for(int i = 0; i <= 5; i++){
        for(int j = 1; j <= 9; j++){
            cubeGUI[colorList[i]][j] = colorList[i];
        }
    }
}
```

2. Ensuite la classe va exécuter l'algorithme. Celui-ci va envoyer l'affichage du cube initial vers la classe *TurnCube* puis va décoder les mouvement envoyés depuis *AlgoMaker*. Et va demander à *TurnCube* de les exécuter.

```
public void scrambleCubeGUI(char[][] scramble) {
    turn = new TurnCube(cubeGUI);
    for(int i = 0; i < scramble[0].length; i++){
        selectedFace(scramble[0][i], scramble[1][i]);
    }
    setCubeGUI(turn.getCubeGUI());
}
```

3. Enfin le tableau des couleurs du cube peut être renvoyé via des accesseurs et des mutateurs.

→ La classe *TurnCube* :

1. La classe va tout d'abord recueillir le mélange sous la forme d'un tableau à deux dimensions dans lequel se trouve : en première ligne la face à tourner, en deuxième le sens. C'est en analysant la face et le sens que le programme va appeler différentes fonctions.

2. La fonction *turnNumber* va définir le nombre de fois que la face doit être tournée : 90°, 180° ou -90°, selon ce qu'il y a dans l'algorithme de mélange, par exemple : F, F' ou F2.

```

public int turnNumber(char direction){
    int N;
    if(direction == '\'){
        N = 3;
    }else if(direction == '2'){
        N = 2;
    }else{
        N = 1;
    }
    return N;
}

```

3. En fonction de la face, le programme va exécuter 2 fonctions *turnAxes* et *turnFace*, pour tourner la face sur elle-même et les pièces autour. Pour cela, il va falloir renvoyer les bons paramètres à ces fonctions (couleur associée, coordonnées de chaque case colorée), qui sont donnés par la fonction suivante, pour le mouvement R par exemple :

```

public void R(char direction){
    for(int i = 1; i<= turnNumber(direction); i++){
        turnFace('R');
        turnAxes('G',new int[]{3,6,9},'W',new int[]
{7,4,1},'B',new int[]{3,6,9},'Y',new int[]{3,6,9});
    }
}

```

4. Ensuite donc, les fonctions *turnFace* et *turnAxes* vont être exécutées N fois pour effectuer tous les changements de variables de chaque couleur du cube. Tout ce processus sera ré exécuté pour chaque mouvement et son sens.

→ Les classes *Graphs* et *Windows* :

La classe *Windows* permet de créer une fenêtre pour afficher le patron du mélange avec un titre et un logo (Logo en annexe n°6).

```

public Windows(char[][] cubeGUI, char[][] algo){
    this.setTitle("CubeSTA");
    this.setSize(520, 400);
    this.setLocationRelativeTo(null);
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    Image icon;
    icon =
Toolkit.getDefaultToolkit().getImage(getClass().getClassLoader().getResource("edu/cubesta/resources/favicon.png"));
    this.setIconImage(icon);
    this.setContentPane(new Graphs(cubeGUI, algo));
    this.setVisible(true);
}

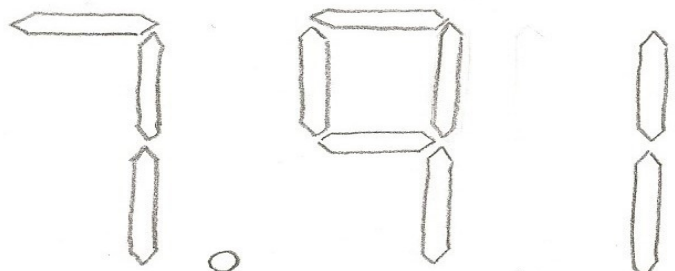
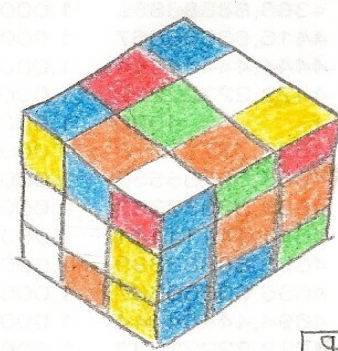
```

Ensuite la classe *Graphs* permet de faire afficher des rectangles de couleur sur la fenêtre en fonction des couleurs du cube donné (Voir annexe n°5).

## ANNEXES

- Annexe ①: Futur affichage du programme

STA CubeSTA  
v 1.0

Scramble: R2 U' L B' R U2 D2 F' U' L2 B R2 D F2 U' L

**START**

Scramble = 16 <sup>Δ</sup> <sub>▽</sub>	Cubes : 5	5 : 7,91	x
Autre option	Best : 7,91	4 : 8,04	x
Autre option	Worst : 9,21	3 : 9,13	x
	Average of 5 : 8,72	2 : 8,99	x
	Average of 12 : ----	1 : 9,21	x
	Average : 8,66		

**Reset**

• Annexe ②: *GitHub*

Firefox ▾ Your Dashboard x CubeSTA-Project/CubeSTA · GitHub x +

https://github.com/CubeSTA-Project/CubeSTA

Search or type a command ? ⚙ Explore Gist Blog Help the-PePito

PUBLIC CubeSTA-Project / CubeSTA Pull Request Watch Star 1 Fork 1

Code Network Pull Requests 0 Issues 24 Wiki Graphs Settings

ISN project — Read more

Clone in Windows ZIP HTTP SSH Git Read-Only git@github.com:CubeSTA-Project/CubeSTA.git Read+Write access

branch: master Files Commits Branches 2 Tags

CubeSTA / 29 commits

Merge branch 'master' of github.com:CubeSTA-Project/CubeSTA.git

auxence.araujo authored 7 days ago latest commit 0f4945846d

nbproject	14 days ago	Update nbproject/project.properties [J-J36]
src	7 days ago	Merge branch 'master' of github.com:CubeSTA-Project/CubeSTA.git [auxence.araujo]
.gitignore	11 days ago	Ajout d'un favicon pour la fenêtre [Jitrixis]
README.md	17 days ago	Initial commit [Jitrixis]
build.xml	17 days ago	Premier Commit du projet [Jitrixis]
licensing.txt	17 days ago	Create licensing.txt [Jitrixis]
manifest.mf	17 days ago	Premier Commit du projet [Jitrixis]

README.md

# CubeSTA

• Annexe ③: La javadoc

Firefox | Your Dashboard | CubeSTA-Project/CubeSTA - GitHub | CubeSTA (CubeSTA)

file:///C:/Users/PePito/Lycée/ISN/Projet/CubeSTA-master/dist/javadoc/index.html

Marque-pages

All Classes

**Packages**

- edu.cubesta
- edu.cubesta.average
- edu.cubesta.scramble
- edu.cubesta.timer
- edu.cubesta.windows

**All Classes**

- AlgoMaker
- Average
- CubeGUI
- CubeSTA
- Graphs
- Timer
- TurnCube
- Windows

Overview Package **Class** Use Tree Deprecated Index Help

Prev Class Next Class Frames No Frames

Summary: Nested | Field | Constr | Method Detail: Field | Constr | Method

edu.cubesta

### Class CubeSTA

java.lang.Object  
edu.cubesta.CubeSTA

---

```
public class CubeSTA
extends java.lang.Object
```

**Version:**

1.0

**Author:**

julien.gardet yann.droy auxence.araujo

#### Constructor Summary

**Constructors**

Constructor and Description
CubeSTA()

#### Method Summary

**Methods**

Modifier and Type	Method and Description
static void	main(java.lang.String[] args)

#### Methods inherited from class java.lang.Object

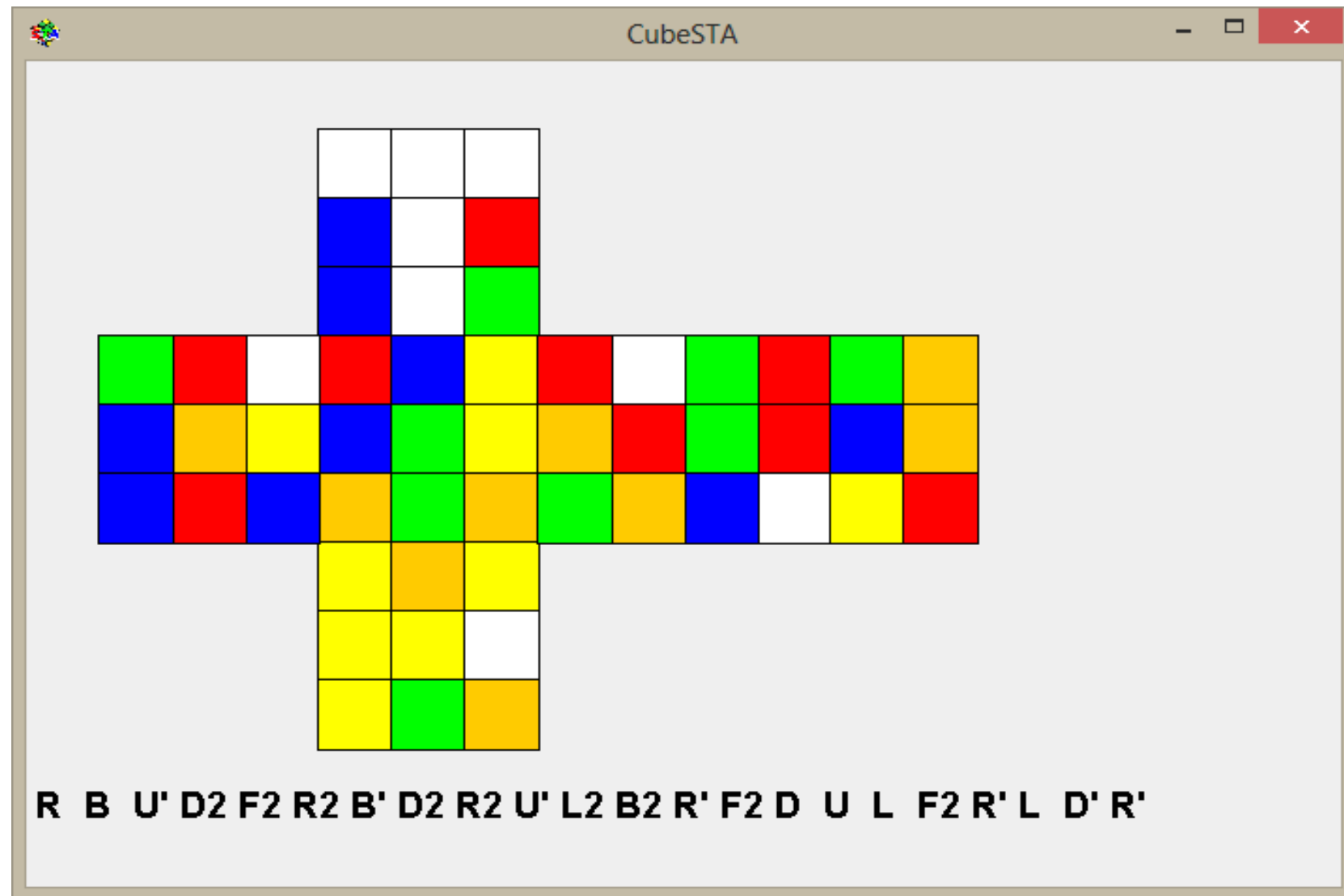
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

- Annexe ④ : Le patron du nom des variable du projet

			W9	W8	W7							
			W6		W4							
			W3	W2	W1							
O7	O4	O1	G1	G2	G3	R3	R6	R9	B9	B8	B7	
O8		O2	G4		G6	R2		R8	B6		B4	
O9	O6	O3	G7	G8	G9	R1	R4	R7	B3	B2	B1	
			Y1	Y2	Y3							
			Y4		Y6							
			Y7	Y8	Y9							



- Annexe ⑤: L'affichage temporaire du programme



- Annexe ⑥ : Le logo.

