

Rapport Résolution de Rubik's Cube

Breck, Yann

Zimmermann, Quentin

22 décembre 2013

Table des matières

1	Introduction	2
1.1	Sujet numéro 7	2
1.1.1	Contraintes	2
2	Le rubik's cube	3
2.1	Généralités	3
2.2	Fonctionnement	4
2.2.1	La structure	4
2.2.2	Les mouvements	5
3	Réalisation	6
3.1	La modélisation	6
3.1.1	Les centres	6
3.1.2	Les arêtes	6
3.1.3	Les coins	7
3.1.4	Le cube	7
3.1.5	Les mouvements	8
3.2	La résolution : L'algorithme de Petrus	9
3.2.1	Étape 1 : cube $2*2*2$	9
3.2.2	Étape 2 : cube $2*2*3$	9
3.2.3	Étape 3 : remise en place des arêtes	10
3.2.4	Étape 4 : cube $2*3*3$	11
3.2.5	Étape 5 : placement des coins supérieur	11
3.2.6	Étape 6 : finir les coins supérieur	12
3.2.7	Étape 7 : finir la croix	12
4	Conclusion	13
4.1	Manuel	13
4.1.1	Commandes	13
4.2	Bilan	14
4.2.1	Langages	14
4.2.2	Optimisation	14
A	Sources	15
A.1	Sites internet :	15
A.1.1	Rubik's cube	15
A.1.2	Ocaml	15
A.1.3	Latex	15

Chapitre 1

Introduction

1.1 Sujet numéro 7

Appliquez des rotations aléatoires sur le cube de Rubik. Puis remontez le. Utilisez une autre méthode que celle des 3 étages.

Le rapport écrit doit être réalisé en Latex et long de 10 à 15 pages.

Ni organigramme, ni pseudo code.

Donnez les références bibliographiques, les adresses des sites web que vous avez utilisés. Pensez à commenter vos sources, sans excès. Il n'est pas imposé d'utiliser ocaml/doc (similaire à javadoc).

Vous créerez une archive de votre projet, contenant les sources, le makefile, les fichiers d'exemples, votre rapport, et éventuellement le fichier de votre présentation orale, avec la commande :

```
tar cvzf VOTRENOM.tgz votrerepertoire
```

1.1.1 Contraintes

- Ocaml,
- un mélangeur aléatoire,
- un résolveur pouvant prendre n'importe quel positionnement et résoudre,
- un code lisible et commenté,
- une interface graphique OpenGL,
- un rapport LaTex.

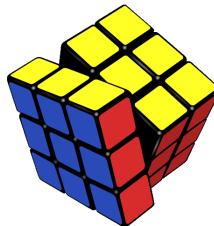
Chapitre 2

Le rubik's cube

2.1 Généralités

Inventé en 1974 par le Hongrois Erno Rubik, le rubik's cube est un casse-tête populaire basé sur un cube subdivisé en $3 \times 3 \times 3$ petit cube. Les faces du grand cube possédant chacune une couleur formée par les faces colorées des petits cubes.

Le créateur étant sculpteur/enseignant désirait à l'origine faire travailler ses étudiants sur le mécanisme interne pour développer leurs perception de la 3D. Suite à la suggestion d'un ami de colorer les faces rendant la remise à l'état initial "difficile".



Aujourd'hui de nombreuses solutions pour résoudre le cube existent, cependant à l'origine, même son créateur ignorait si il existait une solution réelle autre que retrouver théoriquement les mouvements executés. Le développement de l'informatique a grandement facilité la découverte de nouveaux algorithmes pour la résolution. De plus, l'encrage de la 3D dans ce "casse-tête" pousse de nombreuses personnes à se pencher sur sa résolution et avec différentes approches.

Ainsi, tout l'intérêt porté à ce cube a permis l'émergence de disciplines comme le speed-cubing¹, blindfold² ou encore en FMC³.

1. résolution la plus rapide possible
2. résolution en aveugle après avoir regardé le cube quelques instants
3. Fewest Moves Challenge, soit le minimum de mouvement possible

2.2 Fonctionnement

2.2.1 La structure

Comme le montre la Figure on peut déplacer les petits cubes pour mélanger le grand cube. L'enjeu étant bien évidemment de remettre en place les couleurs des 6 faces.

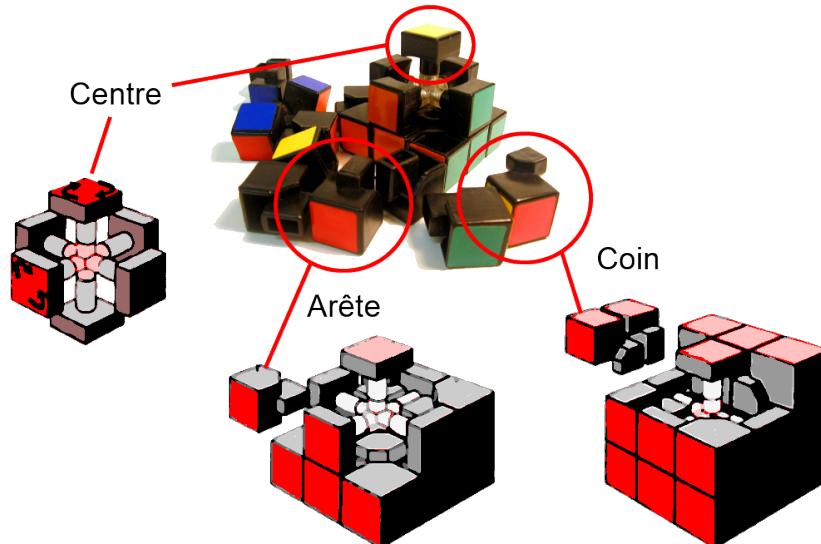
Rouge / Bleu / Jaune / Vert / Orange / Blanc

Toute l'ingéniosité du système réside dans des centres fixes pouvant tourner sur eux-mêmes. Ainsi leurs rotations entraînent celles des coins et des côtés. En effet, le cube est composé de 3 types de "mini-cube" :

	Nombre(s) de face	Occurrences dans le cube
Centre	1	6
Côté	2	16
Coin	3	8

Comme le montre l'image suivante on constate bien que l'intérieur du cube n'est pas la composition de $3 \times 3 \times 3$ petits cubes mais bien un noyau de centre complété par :

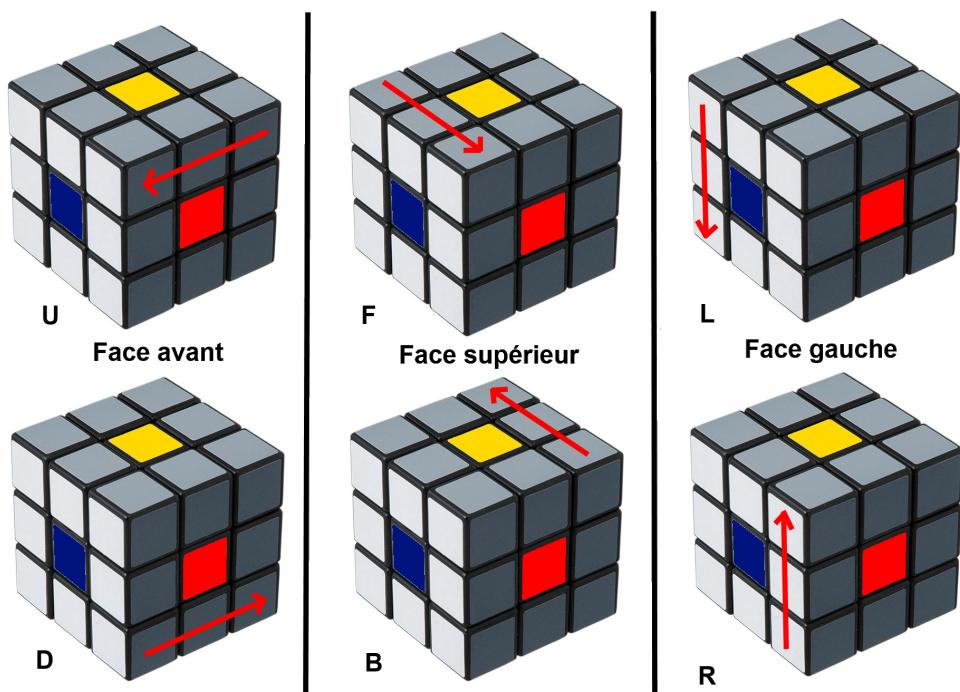
- une couche inférieure de 4 coins et 4 côtés,
- une couche de 4 côtés,
- une couche supérieure similaire à la première, soit 4 coins + 4 côtés.



2.2.2 Les mouvements

Toutes les transformations que l'on peut faire subir aux Rubik's Cube sont basées sur le même mouvement élémentaire : une rotation de 90°, soit un quart de tour, dans le sens des aiguilles d'une montre ou dans le sens contraire, imprimée à l'une des six faces. Comme chacun de ces mouvements élémentaires modifie la configuration du cube, et que l'on peut réaliser des séries de tels mouvements aussi longues que l'on veut, le nombre de configurations possibles est immense.

L'ensemble des mouvements possibles sont tous représentés par des lettres, l'inverse d'un mouvement est représenté par la lettre accompagné d'un ' :



Comme on peut le constater il existe 6 mouvements de base et les 6 mouvements inverses liés. Le centre de la face avant et de la face gauche (ou supérieur) permet de déterminer leurs noms mais pourrait être changé. Cependant pour l'ensemble du projet, le centre de la face avant du cube sera toujours le centre rouge et le centre de la face gauche, le centre bleu.

Chapitre 3

Réalisation

3.1 La modélisation

Pour la modélisation du rubik's cube nous avons décidé d'opter pour de l'objet afin d'approfondir nos compétences en Objective Caml. Nous avons donc créé des objets centre / arête / coin ainsi qu'un objet Cube conservant l'ensemble de ces objets.

3.1.1 Les centres

Les centres (center) sont aux nombre de 6. Chaque centre comporte une face correspondant à un entier allant de 1 à 6 et modélisant la couleur du centre. Ainsi :

Identifiant	Couleur représenté
1	Rouge
2	Vert
3	Orange
4	Bleu
5	Blanc
6	Jaune

3.1.2 Les arêtes

Les arêtes (side) sont au nombre de 12 pour l'étage supérieur, 8 pour l'étage du milieu et 8 pour l'étage inférieur. Chaque arête comporte 2 couleurs distinctes mais chaque arête est unique. Ces deux couleurs sont la surface a et b de chaque objet Side. On peut déterminer si la pièce est inversée dans le rubik's cube quand les valeurs a et b sont inversées d'où l'importance du tableau suivant contenant l'ensemble des références et les "bonnes" positions des arêtes.

Ci-dessous la matrice des arêtes utilisé pour l'objet cube :

Étage des arêtes	Colonne des arêtes			
	0	1	2	3
0	Rouge-Jaune (1-6)	Vert-Jaune (2-6)	Orange-Jaune (3-6)	Bleu-Jaune (4-6)
1	Bleu-Rouge (4-1)	Rouge-VERT (1-2)	Vert-Orange (2-3)	Orange-Bleu(3-4)
2	Rouge-Blanc (1-5)	Vert-Blanc (2-5)	Orange-Blanc (3-5)	Bleu-Blanc (4-5)

3.1.3 Les coins

Les 2×4 coins du cubes comportent chacun 3 valeurs distinctes représentant une couleur et nommé surface a, b et c. Un coin peut être dans 6 positions différentes dont la "bonne" position.

Identifiant :	1	2	3	4	5	6
Ordonnancement des surfaces :	a b c	b c a	c a b	c b a	b a c	a c b

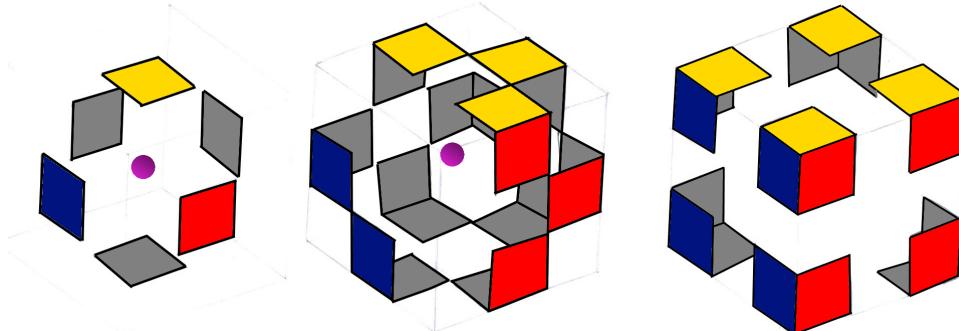
Ci-dessous la matrice des coins utilisé pour l'objet cube :

Etage des coins	Colonne des coins			
	0	1	2	3
0	R-V-J (1-2-6)	V-O-J (2-3-6)	O-Bleu-J (3-4-6)	Bleu-R-J (4-1-6)
0	R-V-Blanc (1-2-5)	V-O-Blanc (2-3-5)	O-Bleu-Blanc (3-4-5)	Bleu-R-Blanc (4-1-5)

3.1.4 Le cube

Le cube comporte le regroupement des différents centres/arêtes/coins. Ainsi il comporte :

- un tableau de 6 centres,
- une matrice de 3 étages (0,1,2) comportant chacun 4 arêtes,
- une matrice de 2 étages comportant chacun 4 coins.



Comme on peut voir sur mon magnifique dessin il ne reste alors plus que des surfaces, l'ensemble formant alors le rubik's cube.

Pour consolider la modélisation et rendre lisible la structure voici le tableau qui nous permet de savoir comment interroger le cube pour obtenir une de ses surfaces (les couleurs symbolisant les liens entre case) :

			0.2.c	0.2.b	0.1.c								
			0.3.b	5	0.1.b								
			0.3.c	0.0.b	0.0.c								
0.2.b	0.3.a	0.3.a	0.3.b	0.0.a	0.0.a	0.0.b	0.1.a	0.1.a	0.1.b	0.2.a	0.2.a		
1.3.b	3	1.0.a	1.0.b	0	1.1.a	1.1.b	1	1.2.a	1.2.b	2	1.3.a		
1.2.b	2.3.a	1.3.a	1.3.b	2.0.a	1.0.a	1.0.b	2.1.a	1.1.a	1.1.b	2.2.a	1.2.a		
			1.3.c	2.0.b	1.0.c								
			2.3.b	4	2.1.b								
			1.2.c	2.2.b	1.1.c								

3.1.5 Les mouvements

Les mouvements sont décomposés par catégorie d'objet. Ainsi chaque mouvement fait tourner les objets dans le sens indiqué comme un jeu de chaise musicale. Seuls les arêtes et coins sont bougés afin de préserver les milieux. En effet, les mouvements des centres ne sont pas requis quand on prend le temps de décortiquer les algorithme de mouvement.

Concept de mouvement d'arête (mouvement R) :

$$R \left| \begin{array}{c} B \\ V \\ J \end{array} \right| \rightarrow \text{devient} \rightarrow J \left| \begin{array}{c} R \\ V \\ B \end{array} \right|$$

Concept de mouvement de coins (mouvement R) :

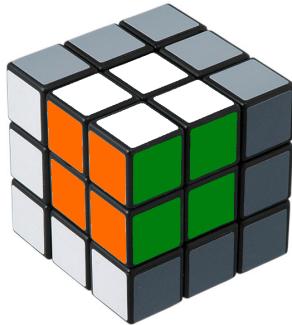
$$R \left| \begin{array}{c} B \\ V \end{array} \right| \rightarrow \text{devient} \rightarrow J \left| \begin{array}{c} R \\ V \\ B \end{array} \right|$$

3.2 La résolution : L'algorithme de Petrus

Lars Petrus né en 1960 est à l'origine d'un des plus rapides speedcubing en 1982. Sa technique est de limité la résolution du cube dans les dernières étapes à deux mouvements (U et F) ou très peu de mouvements dans des techniques basiques.

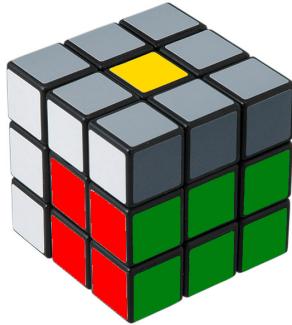
3.2.1 Étape 1 : cube 2*2*2

La première étape d'une résolution Petrus est la constitution d'un cube 2*2*2. Cette étape tient plus de l'intuition que de l'utilisation de véritable formule. Cependant nous avons réussi à rationaliser l'ensemble.



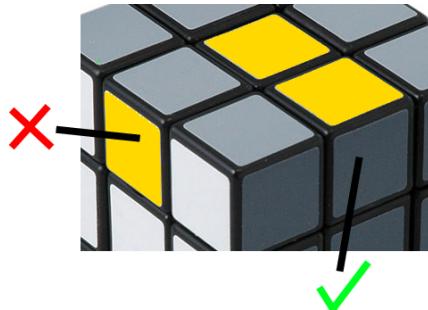
3.2.2 Étape 2 : cube 2*2*3

Une fois l'étape 1 terminée on étend notre cube résolu jusqu'à la face avant pour obtenir un cube 2*2*3. De même que pour l'étape précédente l'ensemble se fait intuitivement après avoir appréhendé le fonctionnement du cube. Le programme, lui, fait pièce par pièce en les retrouvant sur le cube et en les plaçant dans des positions qu'il connaît afin de résoudre.

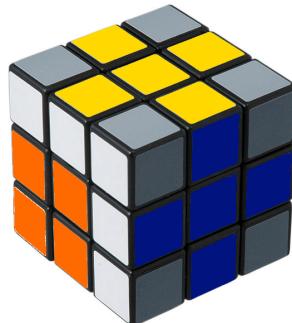


3.2.3 Étape 3 : remise en place des arêtes

Cette étape est très dur à saisir mais une fois comprise elle devient triviale pour un humain. En effet, chaque arête peut soit être "bonne" ou "mauvaise" (quand l'arête est sur la face où elle devrait être alors la couleur de sa face devrait être jointe au centre, voir exemple).



Les mauvaises vont par paire, ainsi on en a soit 0, 2, 4 ou 6. Il faut alors les appareiller en 1.3 et 0.3 et applique une formule minimale¹. Un fois le nombre d'arête mauvaise à 0 on peut repositionner les arêtes sur leurs faces respectivement on obtient alors les deux dernières croix. (Les faces annexes des arêtes n'importe pas pour l'instant).



Bien que l'étape est la plus facile pour un humain une fois comprise nous n'avons pas réussi à l'implémenter correctement. En effet, parfois le solveur se mettait à boucler indéfiniment. Nous avons donc décidé de revenir à notre première version que nous avions réalisé sans véritablement comprendre le fonctionnement et qui mettait parfois 3 min pour plus de 100 000 mouvements. Nous avons tout de même diminué son temps moyen d'exécution à 2 secondes.

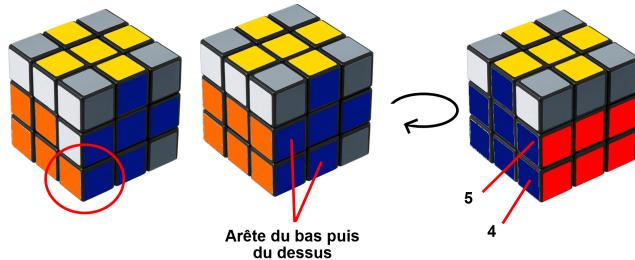
1. L U L'

3.2.4 Étape 4 : cube 2*3*3

Lors de cette étapes on peut détruire les croix tant que les arêtes restent "bonnes". On commence pas placer le coin 1.2 puis ces côtés adjacents (2.3 avec F/U/F/F/U'/F' et 1.3 avec F/F/U'/F'/U/F/F dans les deux cas après les avoir mis en 0.3).

On place ensuite (4) le coin 1.3 en utilisant U/F'/U/F en chaîne jusqu'à ce que le coin se place correctement.

On termine par placer l'arête 1.0 avec la formule F/U'/R'/L/F/F/R/L'/U'/F' quand on à réussi à le pré-placer en 0.2 (l'avantage d'avoir mis les arêtes correctement c'est que si il n'est pas à sa place alors il est sur l'anneau supérieur).

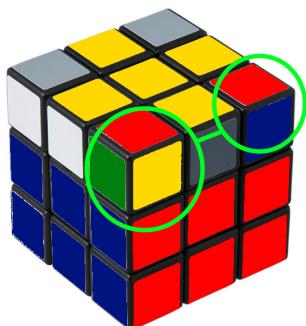


3.2.5 Étape 5 : placement des coins supérieurs

Algorithme de Niklas

Cette étape cherche à positionner les coins à leurs places respective sans regarder si ils se placent correctement. Pour cela on utilise la séquence Niklas² qui inverse les positions des deux coins de la face avant.

On commence par placer le coin 0.0 à sa place puis on regarde si les autres sont à leurs places et on agit en conséquence

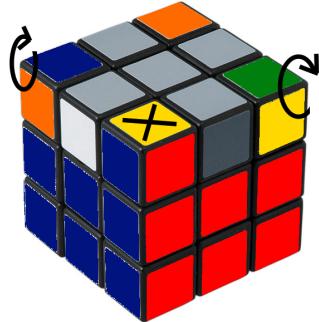


2. Niklas :L/U'/R'/U/L'/U'/R/U/U

3.2.6 Étape 6 : finir les coins supérieur

Algorithme de Sune

Lors de l'étape 6 on fait tourner sur eux même les coins pour qu'ils atteignent leurs position finale. On utilise alors Sune³ qui travail sur 3 côtés et bloque le coin avant gauche. On cherche donc un coin bien placé quand les autres sont "erronés" pour appliquer Sune. Dans le cas contraire on applique Sune sur un coin erroné afin d'éviter les désynchronisations. Celle-ci apparaissent si deux coins ou plus sont bien placés et que l'un d'entre eux est préservé pendant Sune.

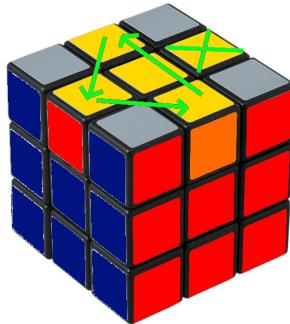


3.2.7 Étape 7 : finir la croix

Algorithme de Allan

Cette étape ressemble à l'étape 6 mais concerne le positionnement des arêtes. On utilise Allan⁴ qui préserve l'arête supérieure droite quand on l'applique et travail sur les 3 autres. Comme pour Sune on utilise Allan sur une arête bien placée quand les autres sont fausses ou l'inverse (une fausse quand 2 arêtes ou plus sont bonnes).

Allan fait tourner dans le sens inverse des aiguilles d'une montre les arrêtes visées (voir exemple).



3. Sune :B/U/B'/U/B/U/U/B'/U/U

4. Allan :F/F/U'/L/R'/F/F/L'/R/U'/F/F

Chapitre 4

Conclusion

4.1 Manuel

Le projet comporte 9 fichiers

- cubeObject.ml : qui permet d'instancier un cube en OCaml et ces commandes,
- les étapes 1 à 7 séparé entre elle pour pouvoir être mieux modifié/travaillé,
- resolveur.ml : qui tient le rôle d'interface et de coordinateur de l'objet cube et de ses étapes de résolution.

4.1.1 Commandes :

Une fois les fichiers réunis ensemble la commande `ocaml #use "resolveur.ml"` chargera l'ensemble. On pourra alors utiliser les commandes :

Commandes et Explications :

- **resolv (cube)** : résout le cube passé en paramètre et l'affiche.
- **shake (cube) (nombre de mouvement)** : mélange avec le nombre de mouvements aléatoires en paramètre.
- **init (cube)** : Remet à zéro le cube passé en paramètre.
- **next (cube)** : applique les étapes de résolution du cube, les mouvements qui ont été nécessaires et l'état du cube. (Next bloque à l'étape 7 quand le cube est résolu).
- **all (cube) (nombre de mouvement)** : mélange avec le nombre de mouvements aléatoires en paramètre puis utilise resolv.
- **how (cube) (numéro de l'étape)** : donne les mouvements effectués à l'étape passé en paramètre.
- **howMany (cube) (numéro de l'étape)** : donne le nombre de mouvements effectués à l'étape passé en paramètre (avec 0=l'ensemble des étapes).

4.2 Bilan

4.2.1 Langages

OCaml

OCaml c'est révélé très performant (même sur machine virtuelle les résultats ont été stupéfiants) mais malgré une première approche positive et aisée plusieurs problèmes sont apparus.

- Bien que le O signifie Objective ait été une volonté de l'inria d'ajouter une dimension P-O-Objet, je pense que ce choix est à fuir car extrêmement coûteux en temps de développement (car très exigeant) et le tout pour un résultat en dessous de ce qu'aurait pu permettre la programmation fonctionnelle "classique".
- Le peu de documentation à aussi été un frein. En effet les codes retour des erreurs étant faible ou approximatif la résolution de problème prend énormément de temps. De plus, la communauté de développeur n'est visiblement pas suffisante pour compenser cette lacune.
- Les exigences du langage sont aussi tellement strictes que seul quelqu'un d'expérimenté peut profité de la puissance de OCaml.

On reconnaîtra tout de même, sans tomber dans le patriotisme, la classe d'avoir un bon langage de programmation français.

Pour conclure, un langage avec un bon potentiel surtout quand on le maîtrise bien MAIS avec une faible documentation pour débutant, une communauté de développeur restreinte en comparaison au Java,C/C++,C#,Python et une inutilité de la partie object selon nous.

LaTex

Découverte complète et très positive même si toute les subtilités m'échappe encore. La mise en page est tellement propre et automatique sans de brusque saut d'humeur du logiciel. Un plaisir.

4.2.2 Optimisation

L'étape 3 utilise encore de 1000 à 10000 mouvements alors qu'elle pourrait tourner à une 30 50aines de mouvement au maximum. Actuellement, une partie aléatoire détermine les formules utilisées jusqu'à trouver le bon résultat. Bien que ça me déprime, le temps manque pour debug ce problème. La résolution utilise donc encore step3old.

Un affichage 3D avec openGl serait vraiment un gros plus à ce projet, à voir.

Annexe A

Sources

A.1 Sites internet :

A.1.1 Rubik's cube

- http://fr.wikipedia.org/wiki/Rubik's_Cube
- <http://lar5.com>
- <http://www.francocube.com/>
- <http://french.cars.free.fr/classroom/sv753.htm>
- http://images.productwiki.com/upload/images/grayscale_rubik_s_cube.jpg
- <http://chronomaton.apln-blog.fr/files/2012/11/rubik.png>

A.1.2 Ocaml

- <http://caml.inria.fr/pub/docs/manual-ocaml-4.00/manual005.html>
- http://form-ocaml.forge.ocamlcore.org/html/formation_ocaml.html

A.1.3 Latex

- <http://fr.openclassrooms.com/informatique/cours/redigez-des-documents-de-qualite-avec-latex/types-de-documents-et-caracteres-speciaux>
- <http://marin.jb.free.fr/latex/>
- <http://deuns.chez.com/latex/environ.html>