BAB IV

IMPLEMENTASI DAN PENGUJIAN

Pada bab ini akan dilakukan implementasi dan pengujian terhadap aplikasi yang dibangun. Tahapan ini dilakukan setelah analisis dan perancangan selesai dilakukan dan selanjutnya akan diimplementasikan kedalam bahasa pemograman. Setelah implementasi maka dilakukan pengujian terhadap aplikasi. Aplikasi yang telah dibangun akan diimplementasikan untuk mengetahui apakah sistem tersebut dapat berjalan sesuai dengan tujuannya atau tidak.

4.1 Implementasi

Tujuan implementasi adalah untuk menerapkan perancangan yang telah dilakukan terhadap sistem sehingga *user* dapat memberi masukan demi berkembangnya sistem yang telah dibangun sebagai simulasi dari aplikasi penerapan algoritma *layer by layer* untuk menyelesaikan permainan rubik. Perangkat keras dan perangkat lunak yang digunakan untuk melakukan implementasi dan pengujian aplikasi yaitu sebagai berikut :

a. Perangkat keras

Perangkat keras yang digunakan untuk pembuatan aplikasi dijelaskan pada tabel 4.1.

Tabel 4.1. Perangkat keras yang digunakan

Komputer	Spesifikasi perangkat keras
Prosesor	Intel core i3 2.27 Ghz
Layar	Resolusi 1378 x 768
Monitor	LCD 14 inc
Memori	Memori 2 GB
Harddisk	Harddisk 160 GB
Keyboard dan mouse	-

b. Perangkat lunak

Perangkat lunak yang digunakan untuk mengimplementasikan sistem dijelaskan pada tabel 4.2.

Tabel 4.2. Perangkat lunak yang digunakan

Komputer	Spesifikasi perangkat lunak
Sistem Operasi	Windows XP
Bahasa Pemograman	Macromedia Flash 8.0

4.1.1 Implementasi Antarmuka

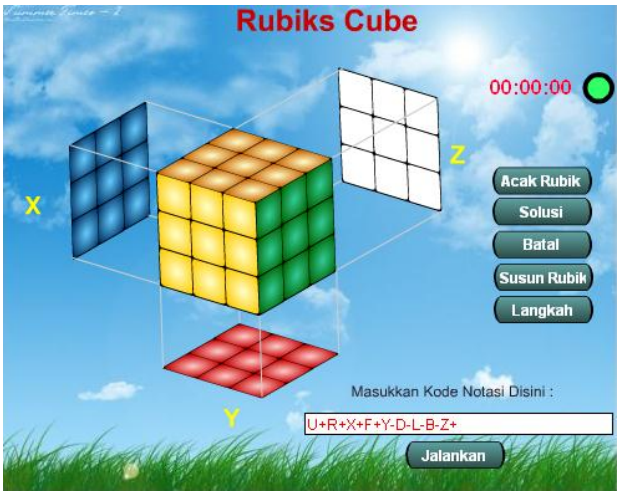
Implementasi antarmuka untuk aplikasi tampilan utama yang dibangun dapat dilihat dari gambar 4.1.



Gambar 4.1. Implementasi Tampilan Utama

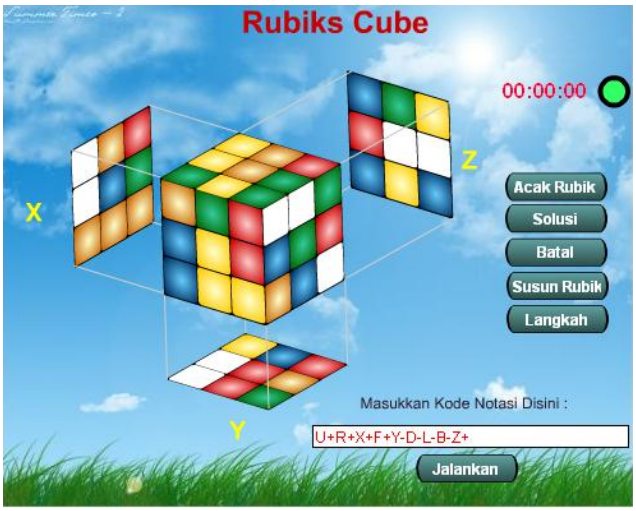
Didalam aplikasi ini terdapat beberapa menu, menu utama yang mempunyai fungsi seperti menampilkan halaman main, halaman bantuan, dan halaman profil. Menu yang terdapat didalam aplikasi terdiri dari:

1. Main
2. Bantuan
3. Profil



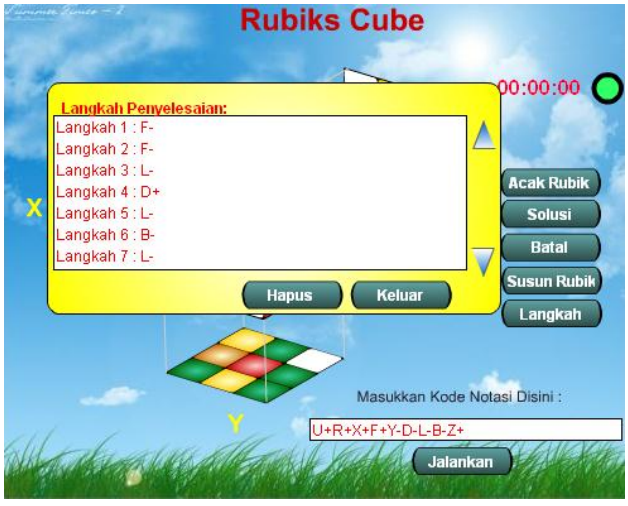
Gambar 4.2. Implementasi Tampilan Menu Main

Tampilan antarmuka aplikasi dibawah ini merupakan implementasi dari kegunaan tombol acak rubik. Pada tampilan terlihat setelah tombol acak rubik diklik maka rubik akan bergerak untuk mengacak rubik. Hasil implmentasi dapat dilihat pada gambar 4.3 dibawah ini.



Gambar 4.3. Implementasi Tampilan Menu Acak Rubik

Tampilan antarmuka aplikasi dibawah ini merupakan implementasi dari kegunaan tombol langkah. Tombol langkah merupakan perintah yang digunakan untuk menampilkan kotak pesan langkah-langkah gerakan rubik.



Gambar 4.3. Implementasi Tampilan Menu Langkah

Menu bantuan yaitu bantuan untuk pengguna melakukan semua proses yang ada pada aplikasi antarmukanya dapat dilihat pada gambar 4.5, sedangkan tentang aplikasi dapat dilihat pada gambar 4.4.



Gambar 4.4. Implementasi Tampilan Menu Profil



Gambar 4.5. Implementasi Tampilan Menu Bantuan

4.2 Implementasi Contoh Kasus *Layer by Layer*

Salah satu contoh implementasi simulasi yang akan digunakan untuk menyelesaikan *rubiks cube* yang teracak yaitu dengan menggunakan metode *layer by layer*. Langkah pertama yaitu menyelesaikan *layer* pertama, setelah *layer* pertama selesai kemudian langkah kedua menyelesaikan *layer* kedua dan tahap yang terakhir setelah *layer* kedua selesai kemudian penyelesaian *layer* ketiga. Pemain dapat menempatkan titik awal dan titik akhir pengacakan pada posisi yang diinginkan, dan apabila acakan tersebut ingin diselesaikan pemain dapat menggunakan tombol solving. Implementasi penyelesaian dengan algoritma *layer by layer* dapat dilihat pada penjelasan dibawah ini. Implementasi penyelesaian yang dimulai dari *layer* pertama sampai menyelesaikan *layer* yang terakhir hingga sisi – sisi rubik tersusun sesuai dengan warnanya.

1. Layer Pertama

Langkah pertama dalam menyelesaikan *layer* pertama/atas adalah membentuk sebuah *cross* disisi atas, seperti pada gambar ini.



Gambar 4.6. *Cross* yang sukses dibentuk

Setelah *cross* berhasil dibentuk yang perlu dibentuk lagi yaitu warna di samping keempat buah ujung *cross* tersebut harus serupa dengan warna dibawahnya. Langkah selanjutnya adalah mengisi sudut-sudut bagian atas dengan kubus yang sesuai tanpa merusak *cross* tersebut. Untuk bagian ini, pertama-tama

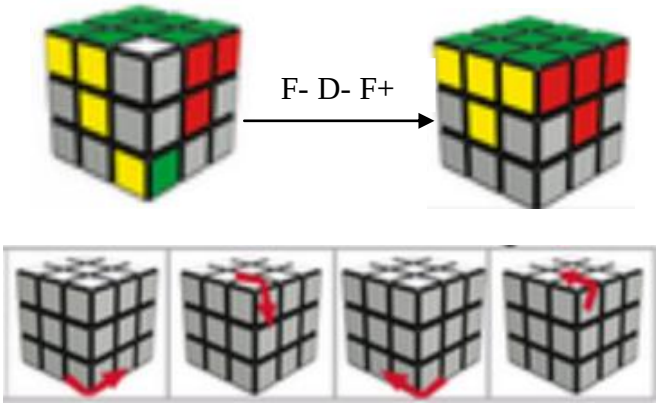
menempatkan kubus yang seharusnya menjadi sudut tersebut dibagian bawahnya dan putar *Rubik's Cube* hingga kubus tersebut berada dibawah kiri depan, contoh:



Gambar 4.7. Tempatkan sudut di bawah tempat sebenarnya

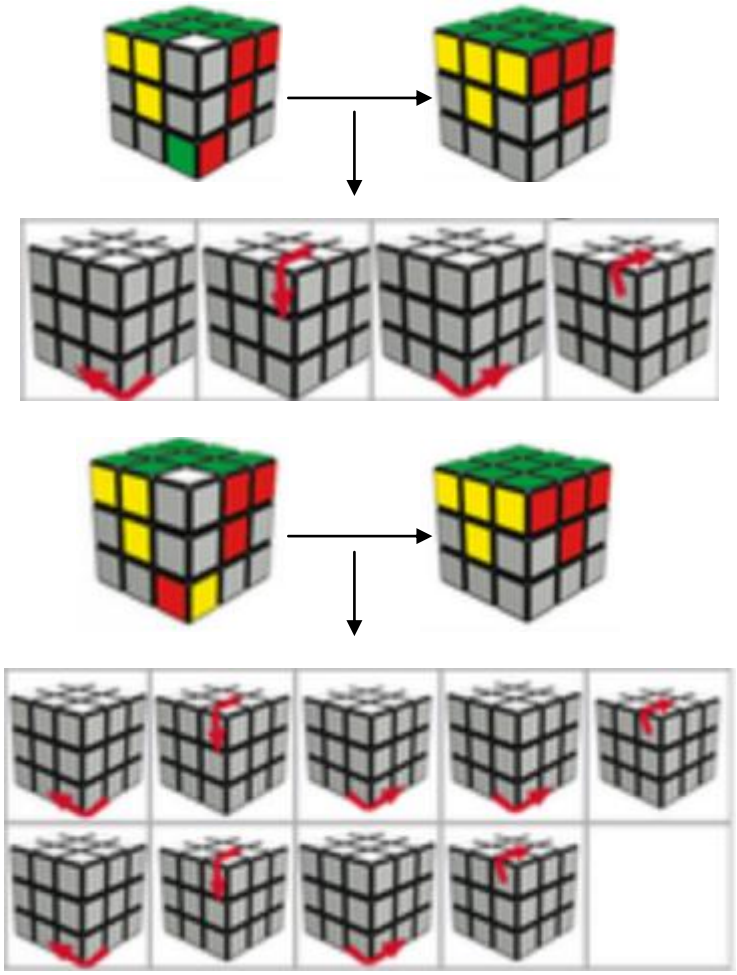
Lalu lakukan algoritma berikut berdasarkan letak sisi yang warnanya sama dengan warna *cross*:

- jika menghadap ke depan, F- D- F+
- jika menghadap ke kiri, L+ D+ L-
- jika menghadap ke bawah, F- D+ D+ F+ D+ F- D- F+



Gambar 4.8. Notasi Penyelesaian Rubik

Lakukan langkah ini hingga keempat sudut tersebut benar pada tempatnya dan selesaikan *layer* pertama.

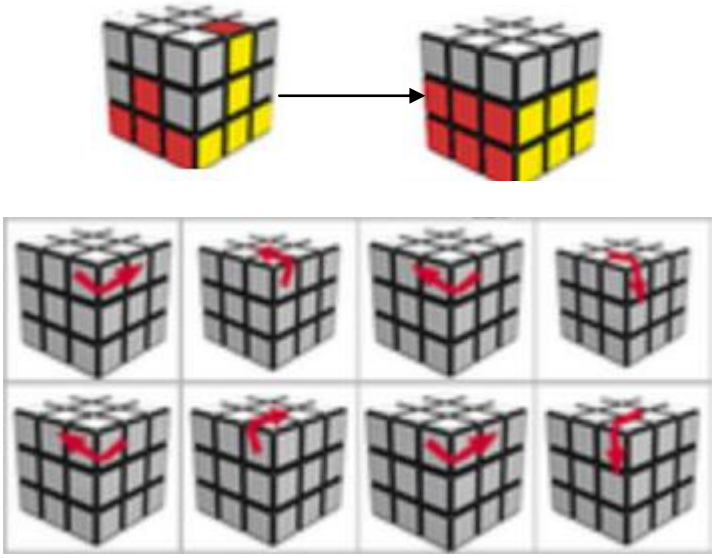


Gambar 4.9. Gerakan notasi yang dilakukan

Pada tahap ini *layer* pertama sudah dapat diselesaikan. *Layer* bagian atas sudah satu warna, dan disemua sisi samping rubik membentuk pola huruf "T".

2. Layer Kedua

Untuk setiap kubus yang tidak benar letaknya (pada *layer* kedua), tempatkan kubus yang seharusnya berada pada tempat tersebut dibawah salah satu sisi *Rubik's Cube* dimana warna sisi kubus tersebut serupa dengan warna sisi kubus diatasnya. Lalu, putar *Rubik's Cube* hingga sisi tersebut menjadi sisi depan. Seperti pada gambar dibawah ini:

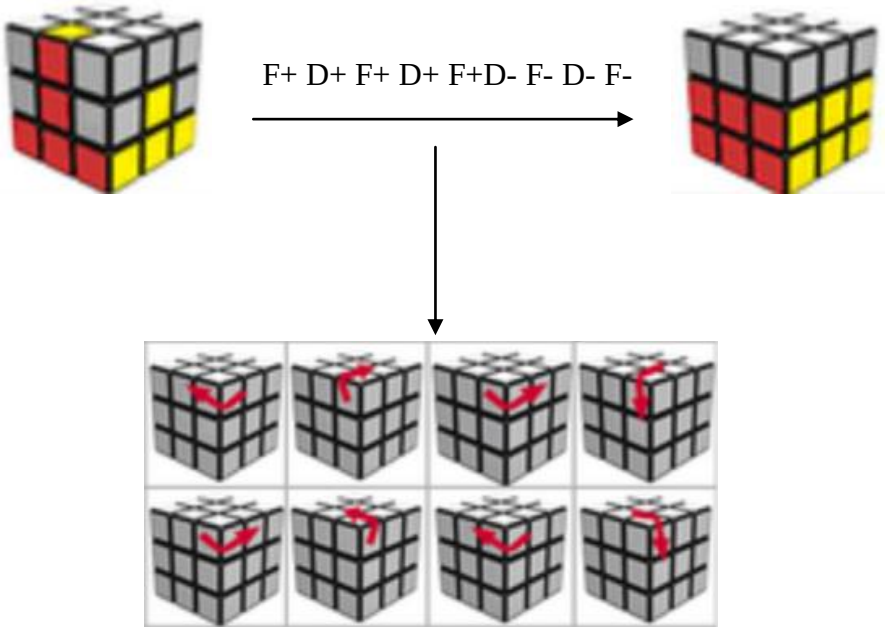


Gambar 4.10. Letakkan di bawah salah satu sisi

Lakukan algoritma berikut berdasarkan dimana seharusnya kubus bawah tersebut diletakkan:

- jika diletakkan di kiri, F+ D+ F+ D+ F+ D- F- D- F-

- jika diletakkan di kanan, F- D- F- D- F- D+ F+ D+ F+



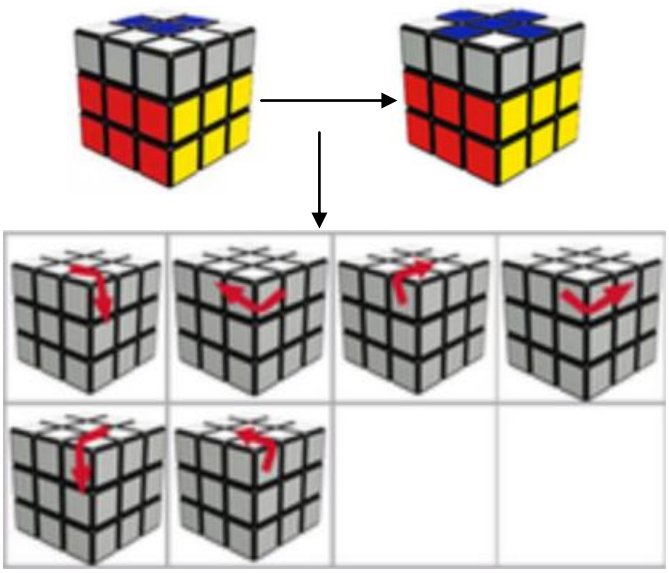
Gambar 4.11. Notasi Penyelesaian Layer Kedua

Lakukan langkah ini hingga keempat kubus pada *layer* kedua berada pada tempat sebenarnya. Seandainya ada kubus yang terbalik atau salah tempat pada *layer* kedua, tempatkan kubus tersebut pada *layer* ketiga dengan cara menukarnya dengan salah satu kubus di *layer* ketiga, setelah itu lakukan algoritma diatas seperti biasa.

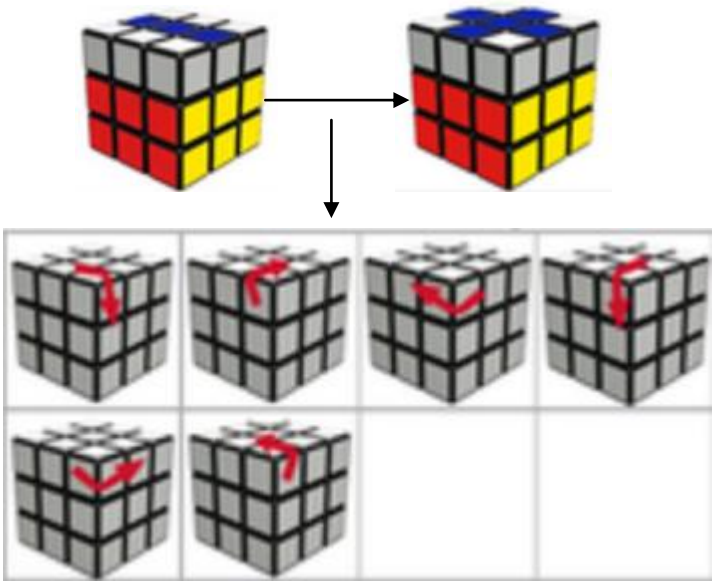
3. Layer Ketiga

Untuk menyelesaiakan *Layer* ketiga kedua layer yang telah selesai tidak boleh dirusak. Untuk itu, cukup banyak algoritma yang harus dihapal untuk menyelesaikan *layer* terakhir ini. Langkah pertama mirip dengan langkah pertama untuk *layer* pertama, yaitu membentuk *cross* pada sisi bawah. Jika pada sisi bawah belum terbentuk sebuah *cross*, ada tiga kondisi yang mungkin terjadi, yaitu:

a. Kondisi Satu



b. Kondisi Dua



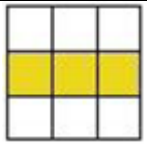
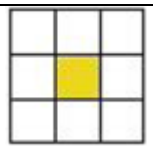
c. Kondisi Tiga



Gambar 4.12. Rubik penyelesaian layer ketiga kondisi tiga

Kalau terjadi kondisi tiga, maka bisa diselesaikan dengan memakai kondisi satu kemudian dilanjutkan dengan kondisi dua, atau dibalik juga bisa (kondisi dua kemudian kondisi satu). Lebih lengkapnya ketiga kondisi untuk membentuk *cross* pada *layer* ketiga dijelaskan pada tabel 4.3.

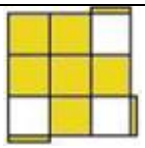
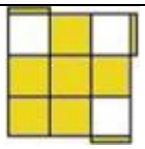
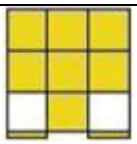
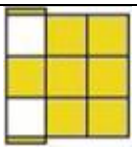
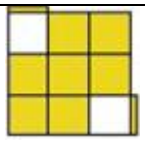
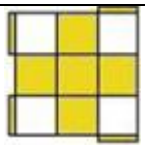
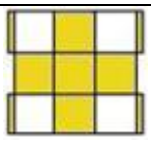
Tabel 4.3. Kondisi dari *cross* pada sisi bawah dan solusi

Kondisi	Gambar
Dua kubus bersebelahan belum membentuk *cross* Solusi: putar sisi bawah hingga berbentuk seperti gambar, lalu lakukan F+D+L+D-L-F-	
Dua kubus berseberangan belum membentuk *cross* Solusi: putar sisi bawah hingga berbentuk seperti gambar, lalu lakukan F+L+D+L-D-F-	
Empat kubus belum membentuk *cross* Solusi: lakukan salah satu algoritma di atas, setelah itu, salah satu kondisi di atas akan terpenuhi.	

Setelah *cross* terbentuk, letakkan keempat sudut pada *layer* bawah di tempat yang seharusnya, tetapi tidak perlu menghadap arah yang sebenarnya. Sebagai contoh, sudut merah-putih-biru harus berada diantara sisi merah, putih, dan biru, tetapi ketiga sisi sudut ini tidak perlu menghadap arah yang benar. Cara untuk membenarkan letak keempat sudut ini adalah dengan menukarkan letak dua buah sudut bawah. Hanya diperlukan satu buah algoritma untuk hal ini, algoritma ini menukar letak sudut depan-kiri-bawah dengan sudut belakang-kiri-bawah tanpa merusak posisi layer pertama dan kedua. Algoritma tersebut adalah R+ D- L- D+ R- D- L+ D+ D+. Setelah keempat sudut berada pada posisi sebenarnya, langkah selanjutnya adalah membuat sisi bawah berwarna serupa. Untuk mempermudah, putar *Rubik's Cube* keatas dua kali sehingga sisi bawah kini menjadi sisi atas. Jika sisi atas (tadinya sisi bawah) *Rubik's Cube* belum berwarna serupa, ada tujuh kasus yang mungkin terjadi, seperti dapat dilihat pada tabel di

bawah ini. Jika tidak ada kondisi yang cocok, putar *Rubik's Cube* ke kiri atau ke kanan hingga sisi atasnya cocok dengan salah satu kondisi di bawah ini.

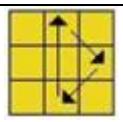
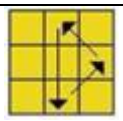
Tabel 4.4. Kondisi dari sisi atas dan solusi

Kondisi	Solusi
	R-U-R+U-R-U+U+R+U+U+ Ini adalah kondisi 1, setelah algoritma ini dilakukan, sisi atas akan langsung serupa.
	R+U+R-U+R+U+U+R-U+U+ Ini adalah kondisi 2, setelah algoritma ini dilakukan, sisi atas akan langsung serupa.
	R+U+R-U+R+U+U+R-U+U+ Setelah itu sisi atas akan menjadi kondisi 1 atau 2
	R+U+R-U+R+U+U+R-U+U+ Setelah itu sisi atas akan menjadi kondisi 1 atau 2
	R-U-R+U-R-U+U+R+U+U+ Setelah itu sisi atas akan menjadi kondisi 1 atau 2
	R+U+R-U+R+U+U+R-U+U+ Setelah itu sisi atas akan menjadi kondisi 1 atau 2
	R+U+R-U+R+U+U+R-U+U+ Setelah itu sisi atas akan menjadi kondisi 1 atau 2

Langkah terakhir dalam menyelesaikan *layer* ketiga (yang kini berada di atas) adalah menukar posisi kubus yang masih tidak berada pada tempatnya. Jika masih ada kubus yang salah tempat, ada dua kondisi yang mungkin, yaitu tiga kubus salah tempat atau empat kubus salah tempat. Jika ada tiga kubus salah tempat, ada dua kondisi yang mungkin terjadi, seperti digambarkan pada tabel di bawah. Jika

sisi atas tidak cocok dengan kedua gambar ini, putar *Rubik's Cube* ke kiri atau kanan hingga ada gambar yang cocok.

Tabel 4.5. Kondisi dari sisi atas dengan tiga kubus salah tempat dan solusi

Kondisi	Solusi
	R+R+U+F+B-R+R+F-B+U+R+R+
	R+R+U-F+B-R+R+F-B+U-R+R+

Jika ada empat kubus yang salah tempat, lakukan salah satu dari algoritma di atas (untuk tiga kubus) dan setelah itu kubus yang salah tempat akan menjadi tiga sehingga algoritma diatas dapat digunakan. Penyelesaian *layer* ketiga pada rubik lebih jelas akan dijelaskan pada gambar dibawah ini.

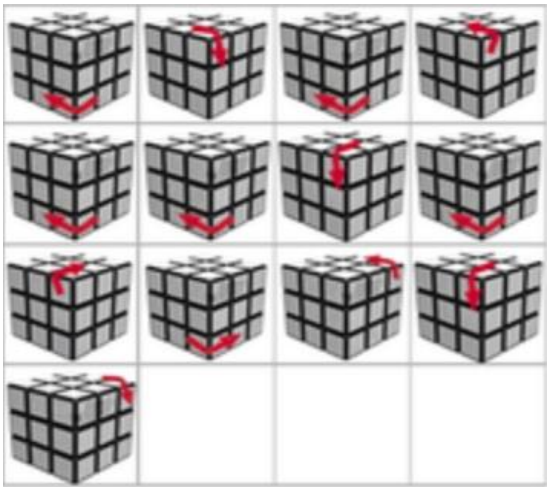
2 edge switch



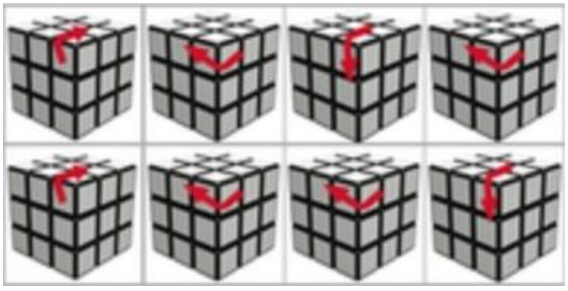
3 edge switch



2 edge switch



3 edge switch



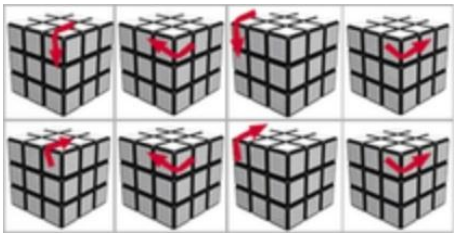
Disini penggunaan dari "*2 edge switch*" dan "*3 edge switch*" tergantung dari kondisi rubik. Setelah membenarkan sisi-sisi maka rubik akan menjadi seperti dibawah ini.



Untuk memperbaiki *corner*, langkah pertama adalah menempatkan *corner* di tempatnya. Caranya dengan melihat pada *corner*-nya ada warna apa saja, dan jadikan warna *center* sebagai acuan. Gunakan "*switch 3 corner*" untuk memindahkan *corner*. Tempatnya sudah benar, ada kemungkinan warnanya yang tidak benar. Untuk itu gunakan "*turn corner left*" dan "*turn corner right*".

Switch 3 corner

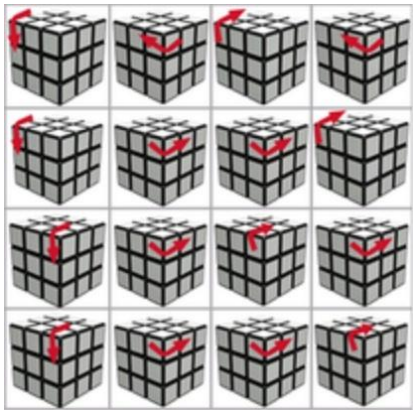




Turn Corner

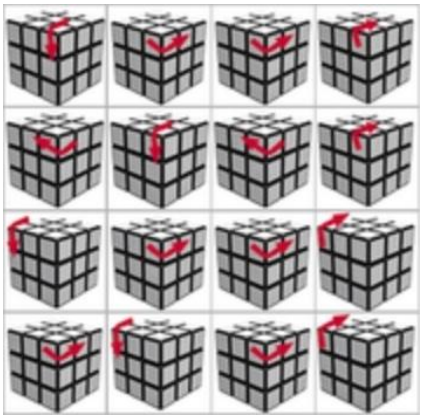


Turn Corner Right



Turn Corner left bisa diganti dengan dua kali *turn corner right*. Tapi lebih baik kalo bisa menghafal keduanya.

Turn Corner Left





Gambar 4.13. Implementasi Rubik Berhasil Diselesaikan

4.3 Pengujian Sistem

Pengujian sistem merupakan hal terpenting yang bertujuan untuk menemukan kesalahan–kesalahan atau kekurangan – kekurangan pada perangkat lunak yang diuji. Pengujian bermaksud untuk mengetahui perangkat lunak yang dibuat sudah memenuhi kriteria yang sesuai dengan tujuan perancangan perangkat lunak tersebut.

. Pengujian pada sistem yang baru dibangun dilakukan melalui dua tahap yaitu pengujian *alpha* (fungsional) dan pengujian *betha*.

a. Pengujian *Alpha*

Metode yang digunakan dalam pengujian *alpha* adalah metode *blackbox* yang fokus pada persyaratan fungsional dari perangkat lunak yang dibangun.

b. Rencana Pengujian Program Simulasi

Rencana pengujian yang akan dilakukan pada aplikasi dengan menggunakan metode *blackbox* .

Tabel 4.6. Rencana pengujian program simulasi

Item uji	Detail pengujian	Jenis Pengujian
Acak Rubik	Sumbu X, Y, Z	*Blackbox*
	Sisi atas, bawah teracak	*Blackbox*
	Sisi kanan, kiri teracak	*Blackbox*
	Sisi depan, belakang teracak	*Blackbox*
Solving	Sumbu X, Y, Z	*Blackbox*
	Selesaikan sisi atas, bawah	*Blackbox*
	Selesaikan sisi kanan, kiri	*Blackbox*
	Selesaikan sisi depan, belakang	*Blackbox*
Batal	Rubik berhenti	*Blackbox*
Ulang	Reset Rubik	*Blackbox*
Langkah	Lihat langkah penyelesaian	*Blackbox*
Hapus	Bersihkan *form* langkah	*Blackbox*
Profil	Lihat Profil	*Blackbox*
Bantuan	Lihat Bantuan	*Blackbox*
Jalankan	Sumbu X, Y, Z	*Blackbox*
	Sisi atas, bawah teracak	*Blackbox*
	Sisi kanan, kiri teracak	*Blackbox*
	Sisi depan, belakang teracak	*Blackbox*

c. Kasus dan Hasil Pengujian

Berdasarkan rencana pengujian, maka dapat dilakukan pengujian *alpha* dengan *blackbox* pada aplikasi penyelesaian permainan *rubiks cube* yang dijelaskan pada tabel 4.7.

Tabel 4.7. Kasus dan hasil uji (data normal)

Kasus dan hasil uji (data normal)			
Masukan	**Yang diharapkan**	**Pengamatan**	**Kesimpulan**
Data Acak Rubik	Untuk mengacak posisi titik awal dan titik tujuan secara bersamaan	Tombol dapat berfungsi sesuai yang diharapkan	Diterima
Data Solusi	Untuk menyelesaikan rubik yang teracak dengan algoritma *layer by layer*	Tombol dapat berfungsi sesuai yang diharapkan	Diterima
Klik tombol batal	Menghentikan gerakan rubik	Tombol dapat berfungsi sesuai yang diharapkan	Diterima
Klik tombol Susun Rubik	Menyusun dan menampilkan rubik dalam keadaan belum teracak	Tombol dapat berfungsi sesuai yang diharapkan	Diterima
Klik tombol langkah	Menampilkan tampilan untuk langkah-langkah penyelesaian dalam bentuk notasi gerak	Tombol dapat berfungsi sesuai yang diharapkan	Diterima
Klik tombol hapus	Dapat menghapus kode notasi pada tampilan langkah	Tombol dapat berfungsi sesuai yang diharapkan	Diterima
Klik tombol profil	Menampilkan profil dari penulis	Tombol dapat berfungsi sesuai yang diharapkan	Diterima
Klik tombol bantuan	Dapat menampilkan tampilan navigasi pada aplikasi	Tombol dapat berfungsi sesuai yang diharapkan	Diterima
Klik Tombol Main	Menampilkan *form* permainan rubik	Tombol dapat berfungsi sesuai yang diharapkan	Diterima
Data Jalankan	Untuk menjalankan kode notasi pada textbox dan menampilkannya	Tombol dapat berfungsi sesuai yang diharapkan	Diterima

Tabel 4.8. Kasus dan hasil uji (data tidak normal)

Kasus dan hasil uji (data tidak normal)			
Masukan	**Yang diharapkan**	**Pengamatan**	**Kesimpulan**
Data kode notasi salah input huruf	Muncul pesan konfirmasi bahwa kode notasi input gagal	Hasil sesuai dengan yang diharapkan	Diterima
Data kode notasi input 1 huruf	Muncul pesan konfirmasi bahwa harus memasukkan kode notasi dua digit nama sisi dan arahnya	Hasil sesuai dengan yang diharapkan	Diterima

d. Kesimpulan Pengujian *Alpha*

Hasil pengujian dari pengujian *alpha* yang telah dilakukan, menunjukkan bahwa aplikasi yang dibangun sudah memenuhi persyaratan fungsional. Secara fungsional sistem yang telah dibangun sudah dapat menghasilkan keluaran yang diharapkan, begitu juga secara perhitungan yang dilakukan dengan *layer by layer* telah dihasilkan dari titik awal ke titik tujuan dalam menyelesaikan rubik yang teracak. Langkah yang dihasilkan adalah langkah secara terstruktur untuk menyusun rubik.

e. Pengujian Beta

Pengujian beta merupakan pengujian yang dilakukan secara objektif dimana pengujian dilakukan secara langsung kepada responden dengan membuat kuesioner mengenai kepuasan pengguna dengan kandungan *point* syarat.

Berdasarkan hasil pengujian beta, dicari persentase masing-masing jawaban dengan menggunakan rumus:

Y=P/Q*100%

Keterangan:

P = jumlah jawaban responden tiap soal

Q = jumlah responden

Y = nilai persentase

Berikut ini adalah hasil persentase masing-masing jawaban yang sudah dihitung nilainya dengan menggunakan rumus di atas. Kuesioner ini telah diujikan kepada 10 orang. Hasil perhitungan persentase masing-masing jawaban dapat dilihat pada tabel 4.9 – 4.13.

Untuk Pertanyaan No.1

"Apakah penyelesaian yang dihasilkan merupakan algoritma *layer by layer* pada permainan rubik ?"

Tabel 4.9. Hasil pengujian beta pertanyaan No.1

Kategori Jawaban	**Frekuensi Jawaban**	**Jumlah Populasi *Sample***	**Jumlah Persentase**
Sangat Setuju	5	10	50%
Setuju	5	10	50%
Ragu-ragu	0	10	0%
Tidak Setuju	0	10	0%
Sangat Tidak Setuju	0	10	0%

Untuk Pertanyaan No.2

“Apakah aplikasi penyelesaian *rubiks cube* dengan algoritma *layer by layer* mudah digunakan ?”

Tabel 4.10. Hasil pengujian beta pertanyaan No.2

Kategori Jawaban	**Frekuensi Jawaban**	**Jumlah Populasi *Sample***	**Jumlah Persentase**
Sangat Setuju	3	10	30%
Setuju	6	10	60%
Ragu-ragu	1	10	10%
Tidak Setuju	0	10	0%
Sangat Tidak Setuju	0	10	0%

Untuk Pertanyaan No.3

“Apakah penelusuran dengan algoritma *layer by layer* mudah dipahami ?”

Tabel 4.11. Hasil pengujian beta pertanyaan No.3

Kategori Jawaban	**Frekuensi Jawaban**	**Jumlah Populasi *Sample***	**Jumlah Persentase**
Sangat Setuju	4	10	40%
Setuju	6	10	60%
Ragu-ragu	0	10	0%
Tidak Setuju	0	10	0%
Sangat Tidak Setuju	0	10	0%

Untuk Pertanyaan No.4

“Apakah diperlukan fungsi manual untuk pengguna mencari penyelesaian sendiri?”

Tabel 4.12. Hasil pengujian beta pertanyaan No.4

Kategori Jawaban	**Frekuensi Jawaban**	**Jumlah Populasi *Sample***	**Jumlah Persentase**
Sangat Setuju	2	10	20%
Setuju	5	10	50%
Ragu-ragu	0	10	0%
Tidak Setuju	2	10	20%
Sangat Tidak Setuju	1	10	10%

Untuk Pertanyaan No.5

“Apakah aplikasi ini sudah sesuai dengan tujuan yang ingin dicapai ?”

Tabel 4.13. Hasil pengujian beta pertanyaan No.5

Kategori Jawaban	**Frekuensi Jawaban**	**Jumlah Populasi *Sample***	**Jumlah Persentase**
Sangat Setuju	5	10	50%
Setuju	5	10	50%
Ragu-ragu	0	10	0%
Tidak Setuju	0	10	0%
Sangat Tidak Setuju	0	10	0%

Saran dari responden dapat disimpulkan sebagai berikut :

1 Tambahkan algoritma lain yang efektif dan efisien untuk menyelesaikan rubik, karena rubik merupakan permainan yang terdiri dari banyak kondisi penyelesaian.
2 Memakai kata-kata yang lebih sesuai dengan pemainan, tidak terlalu banyak menggunakan bahasa baku untuk sebuah permainan.
3 Dibuat lebih interaktif lagi agar tampilan lebih menarik.
4 Tambahkan *backsound* permainan.

f. Kesimpulan Pengujian Beta

Berdasarkan dari jawaban yang diberikan oleh responden, bisa ditarik kesimpulan bahwa "Implementasi Algoritma *Layer by Layer* Untuk Menyelesaikan Permainan *Rubiks Cube*" ini dinilai dari penelusuran algoritma *layer by layer* dinyatakan mudah dipahami. Jalur yang dihasilkan merupakan jalan aplikasi sudah sesuai dengan tujuan, diperlukan pencarian secara manual bagi pengguna dan pengoperasian aplikasi mudah digunakan.