Institut
Teknologi
Sepuluh Nopember

PROYEK AKHIR

APLIKASI GAME VIRTUAL RUBIK'S CUBE DENGAN MATA TERTUTUP (BLINDFOLDED SOLVING) MENGGUNAKAN ALGORITMA OLD POCHMAN

Yudanis Taqwin Rohman
NRP.7410040502

Dosen Pembimbing:

Arna Fariza, S.Kom., M.Kom
NIP.197107081999032001

Yuliana Setyowati,S.Kom
NIP.197807062002122003

JURUSAN TEKNIK INFORMATIKA
POLITEKNIK ELEKTRONIKA NEGERI SURABAYA
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA

2012

PROYEK AKHIR

APLIKASI GAME VIRTUAL RUBIK'S CUBE DENGAN MATA TERTUTUP (BLINDFOLDED SOLVING) MENGGUNAKAN ALGORITMA OLD POCHMAN

Yudanis Taqwin Rohman
NRP.7410040502

Dosen Pembimbing:

Arna Fariza, S.Kom., M.Kom
NIP.197107081999032001

Yuliana Setyowati,S.Kom
NIP.197807062002122003

JURUSAN TEKNIK INFORMATIKA
POLITEKNIK ELEKTRONIKA NEGERI SURABAYA
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2012

APLIKASI GAME VIRTUAL RUBIK'S CUBE DENGAN MATA TERTUTUP(BLINDFOLDED SOLVING) MENGGUNAKAN ALGORITMA OLD POCHMAN

oleh :

Yudanis Taqwin Rohman

7410040502

Proyek Akhir ini Diajukan Sebagai Salah Satu Syarat Untuk
Memperoleh Gelar Sarjana Sains Terapan (SST)
Di
Politeknik Elektronika Negeri Surabaya
Institut Teknologi Sepuluh Nopember

Surabaya, Januari 2012

Penguji I	**Pembimbing I**
Rizky Yuniar H, S.Kom	**Arna Fariza, S.Kom., M.Kom**
NIP.198106222008121003	**NIP.197107081999032001**
Penguji II	**Pembimbing II**
Tita Karlita, S.Kom., M.Kom	**Yuliana Setyowati,S.Kom., M.Kom**
NIP.197910142002122002	**NIP.197807062002122003**
Penguji III	
Ira Prasetyaningrum, S.Si, MT	
NIP.198005292008122001	

Mengetahui,
Ketua Jurusan Teknik Informatika

Arna Fariza, S.Kom., M.Kom
NIP.197107081999032001

ABSTRAK

Saat ini komputer bukan merupakan barang mewah yang hanya bisa dimiliki oleh kalangan tertentu. Komputer dapat digunakan dalam berbagai bidang. Bukan hanya untuk bisnis, pendidikan bahkan juga hiburan (hobi). Komputer sangat bermanfaat untuk hobi khususnya yang dilakukan secara manual sekarang telah dibuat versi virtualnya. Sehingga untuk melakukan hobi tersebut tidak dperlukan lagi banyak waktu dan biaya.

Begitu banyak permainan yang dahulu hanya bisa dimainkan dengan cara manual sekarang bisa dimainkan dengan komputer. Rubik's cube adalah sebuah permainan yang hanya bisa dilakukan dengan cara manual. Tetapi untuk pemula jika menggunakan cara manual maka memerlukan waktu dan biaya yang cukup besar. Oleh karena itu, perlu dibuat sebuah terobosan baru yang dapat mengatasi kelemahan-kelemahan dari metode pembelajaran manual. Salah satu cara untuk mengatasi permasalahan tersebut adalah dengan membuat sebuah virtual rubik's yang dapat menerjemahkan berbagai algoritma sehingga lebih mudah dipahami oleh pemula daripada menggunakan buku. Dalam pembuatan aplikasi virtual rubik's ini penulis menggunakan *Microsoft Visual Studio 2008* untuk membuat animasi dan menghubungkan aplikasi dengan *database*.

Aplikasi ini menggunakan metode Old Pochmann yang sangat tepat digunakan oleh pemula yang ingin mempelajari metode menyelesaikan Rubiks dengan mata tertutup, Algoritma Old Pochmann dapat diterapkan dengan estimasi gerakan antara 250 sampai 350 gerakan. Selain itu pada aplikasi ini juga dilengkapi dengan tutorial yang cukup lengkap dan disisipkan pula kasus – kasus dasar yang sederhana sehingga mempermudah user untuk memahami konsep dari metode Old Pochmann.

Kata kunci: Rubiks, cube, blindfolded, Old Pochmann, C#

ABSTRACT

Today computers are not a luxury that can only be owned by certain circles. Computers can be used in various fields. Not just for business, education and even entertainment (hobby). Computers are very useful for the hobby particularly those done manually now been made virtual versions. So for hobbies such that no longer much time and cost.

So many games that previously could only be played by hand, now can be played with a computer. Rubik's cube is a game that can only be done by hand. But for beginners if you use the manual method requires time and considerable expense. Therefore, need to be made a new breakthrough that could overcome the drawbacks of the method of learning the manual. One way to overcome these problems is to create a virtual Rubik's who can translate a variety of algorithms that is more easily understood by beginners instead of using the book. In making this application a virtual Rubik's author using Microsoft Visual Studio 2008 to create animations and connect applications to databases.

This application use method of Old very precise Pochmann used by beginner which wish to study method finish Rubiks with eye closed, Algorithm of Old Pochmann can be applied with movement estimation [among/between] 250 until 350 movement. Besides [at] this application [is] also provided with tutorial which complete enough and inserted also case – simple elementary case so that water down user to comprehend concept of method of Old Pochmann.

Keyword: Rubiks, cube, blindfolded, Old Pochmann, C#

KATA PENGANTAR

Alhamdulillah, segala puji bagi Allah SWT, Tuhan Semesta Alam yang telah melimpahkan segala rahmat dan kasih-Nya kepada seluruh makhluk-Nya di bumi. Sholawat dan salam terhatur kepada *qudwah* kita Rasulullah Muhammad SAW, semoga kita selalu *istiqamah* dalam meneladaninya. Karena hanya dengan pertolongan Allah SWT semata, yang berupa nikmat kesempatan, kesehatan dan rizki, penulis akhirnya dapat menyelesaikan Tugas Akhir ini dengan baik.

Tugas akhir ini merupakan salah satu rangkaian mata kuliah dan syarat kelulusan yang harus ditempuh oleh mahasiswa sebagai parameter pengaplikasikan dari beberapa mata kuliah yang telah dipelajari di bangku kuliah.

Penulis mengucapkan terima kasih yang sebesar-besarnya kepada pihak-pihak yang membantu terselesainya laporan Tugas Akhir ini, yaitu kepada:

1. Arna Fariza, S.Kom., M.Kom selaku pembimbing I yang telah bersedia membimbing dan memberikan masukan bagi penulis.
2. Yuliana Setyowati,S.Kom., M.Kom selaku pembimbing II yang telah meluangkan waktu untuk memberikan pengarahan dan masukan bagi penulis.
3. Bapak dan Ibu Dosen yang telah memberikan ilmu dan wawasan baru kepada kami sebagai anak didik, semoga ilmu yang diberikan bermanfaat.
4. Ibu beserta Bapak tersayang, terima kasih atas curahan doa yang tak pernah ada hentinya, kasih sayang yang tulus serta dukungan yang telah diberikan.
5. Teman-teman di D4 IT LJ khusunya angkatan 2010, terima kasih atas dukungannya.

6. Dan berbagai pihak yang tidak dapat disebut secara langsung yang telah memberikan bantuan dan dukungannya.

Penulis yakin bahwa Tugas Akhir ini masih sangat jauh dari kesempurnaan, karena itu penulis sangat berharap kritik dan saran terhadap pengembangan Tugas Akhir ini. Akhirnya, Penulis berharap agar laporan ini menjadi suatu referensi yang baru dan bermanfaat bagi semua pihak maupun Pembaca baik untuk masa sekarang dan yang akan datang semoga Tugas Akhir ini dapat memberikan manfaat bagi *civitas academica* Diploma IV Teknik Informatika Politeknik Elektronika Negeri Surabaya khususnya dan Institut Teknologi Sepuluh Nopember pada umumnya.

Surabaya, Januari 2012

Penulis

DAFTAR ISI

Halaman

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 LATAR BELAKANG

Dunia *Speedsolving* atau lebih dikenal dengan nama rubik saat ini mulai berkembang dengan pesat khususnya di Indonesia.

Blindfoldedsolving (menyelesaikan Rubik's cube dengan mata tertutup) yang merupakan salah satu cabang dari *Speedsolving* masih sedikit peminatnya di Indonesia.

Indonesia menempatkan salah satu putra bangsanya sebagai juara dunia *Mulitiple Blindfolded*(MBLD), namun hal ini tidak diimbangi dengan jumlah *cuber* (pemain Rubik's) yang menguasai metode mata tertutup. Dengan permasalahan tersebut maka perlu dilakukan berbagai perubahan untuk mengembangkan dunia Rubik's khususnya untuk mata tertutup. Sehingga diharapkan akan bermunculan *cuber* kelas dunia yang berasal dari Indonesia. Salah satu cara untuk mengatasi permasalahan tersebut adalah dengan membuat sebuah virtual Rubik's yang dapat menerjemahkan berbagai algoritma mata tertutup sehingga lebih mudah dipahami oleh pemula daripada menggunakan buku.

Aplikasi yang akan penuls buat adalah sebuah virtual rubik's yang berfungsi selain sebagai sarana pembelajaran yang efektif juga sebagai motivator untuk para pemula sehingga tidak mudah menyerah dalam belajar menyelesaikan rubik dengan mata tertutup. Oleh karena itu, di dalam aplikasi yang akan penuls buat juga dilengkapi dengan musuh virtual yang diharapkan lebih memacu user untuk menyelesaikan rubik dengan mata tertutup lebih cepat dan efisien. Dalam pembuatan aplikasi virtual rubik's ini penuls menggunakan aplikasi dengan *Microsoft Visual Studio 2008* dan *Microsoft Acces 2003.*

1.2 PERUMUSAN MASALAH

Rumusan masalah yang akan dibahas dalam laporan tugas akhir ini adalah

1.2.1. Merancang dan membuat program animasi Blindfolded Rubik's cube dengan *Microsoft Visual Studio 2008* dan *Microsoft Acces 2003* .

1.2.2. Mengimplementasikan algoritma Old Pochman kedalam anmasi Rubik's cube yang telah dibuat.

1.2.3. Membuat media pembelajaran atau tutorial untuk menjelaskan metode Old Pochmann kepada user

1.3 BATASAN MASALAH

Agar penulisan tugas akhir ini tidak keluar dari masalah yang telah dirumuskan, maka batasan masalah yang akan dibahas yaitu :

1.3.1 Virtual Rubiks yang digunakan adalah Rubiks 3x3.

1.3.2 Pergerakan rubik dilakukan dengan 1 jari (*single move*).

1.3.3 Metode yang digunakan adalah Old Pochmann.

Penerapan perancangan sisten ke dalam aplikasi dengan menggunakan bahasa pemrograman *Microsoft Visual Studio 2008* dan *Database Microsoft Acces 2003.*

1.4 TUJUAN

Tujuan penulisan tugas akhir ini adalah :

1. Merancang, membangun dan mengimplementasikan sebuah database yang mampu menunjang aplikasi yang akan dibuat.
2. Membangun sebuah aplikasi *Object Oriented* sesuai dengan mata kuliah yang telah dipelajari.
3. Merancang dan membuat aplikasi database yang mampu menerjemahkan algoritma Old Pochman ke dalam sebuah virtual rubik sehingga memudahkan para *cuber* untuk menyelesaikan rubik dengan mata tertutup.
4. Membuat sebuah aplikasi yang efektif untuk memantau perkembangan teknis dari *cuber* dari waktu ke waktu.

1.5 METODOLOGI

Pada penelitian tugas akhir ini, metodologi yang digunakan adalah sebagai berikut:

1. Metode Pengumpulan data
   a. *Interview*, yaitu dengan melakukan wawancara *on-line* kepada master rubik Indonesia yang tergabung dalam organisasi NSA (Nusantara Speedcubing Association).
   b. *Observasi*, yaitu dengan melakukan pengamatan secara langsung.
2. Jenis Penelitian
   a. Penelitian Perpustakaan
      Jenis penelitian yang sifatnya teoritis dengan membaca buku-buku atau literatur yang ada kaitanya dengan judul tugas akhir.
   b. Penelitian Lapangan
      Meneliti secara langsung dan mengambil data yang diperlukan yang ada kaitannya dengan judul tugas akhir.

1.6 SISTEMATIKA PENULISAN

Sistematika penulisan dalam tugas akhir ini adalah sebagai berikut:

BAB I PENDAHULUAN

Dalam bab ini berisi latar belakang penulisan, rumusan masalah, batasan masalah, tujuan penulisan, manfaat, metodologi penelitian dan sistematika penulisan tugas akhir yang berjudul "APLIKASI GAME VIRTUAL RUBIK'S CUBE DENGAN MATA TERTUTUP (BLINDFOLDED SOLVING) MENGGUNAKAN ALGORITMA OLD POCHMAN ."

BAB II TINJAUAN PUSTAKA

Dalam bab ini diuraikan tentang tinjauan pustaka yang mendukung penulisan tugas akhir.

BAB III PERENCANAAN DAN PERANCANGAN SISTEM

Bab ini berisi tentang perancangan sistem dan perancangan database tugas akhir yang akan diimplementasikan dalam bab selanjutnya.

BAB IV UJI COBA DAN ANALISI

Bab ini membahas tentang permasalahan yang diangkat dalam laporan tugas akhir berupa implementasi tabel dan pembuatan program aplikasi.

BAB V PENUTUP

Dalam bab ini berisi kesimpulan dan saran tentang hasil perancangan dan implementasi program.

--Halaman ini sengaja dikosongkan--

BAB II
TEORI PENUNJANG

2.1 RUBIK'S CUBE

Kubus Rubik atau yang lebih dikenal sebagai **Rubik's Cube** tercatat sebagai mainan yang terlaris dan terpopuler sepanjang masa serta hampir dimainkan diseluruh belahan bumi. Berdasarkan penelitian tentang **Rubik's Cube** diperoleh fakta yang mencengangkan.

- 1 dari 5 orang dewasa di Benua Eropa dan Amerika pernah memainkan Rubik's Cube
- Hanya 2 diantara 100 orang yang pernah memainkan **Rubik's Cube** mampu menyelesaikannya.
- Terdapat **43.252.003.274.489.856.000** (4.3×10^{19}) kombinasi kemungkinan dalam sebuah Kubus Rubik standar 3x3x3 namun hanya terdapat 1 penyelesaian.

Rubik's Cube diciptakan oleh ***Erno Rubik*** seorang Dosen di Akademi Seni Terapan dan Kerajinan di Budapest, Hungaria pada tahun 1974. Latar belakang Erno Rubik sebagai Arsitek dan ketertarikan nya akan bentuk geometry 3D membuatnya menciptakan Rubik Cube yang sangat terkait erat dengan struktur dan bentuk geometry [2].

Gambar 2.1 macam Rubik's

Sepanjang catatan sejarah, telah diciptakan Kubus rubik dengan berbagai bentuk dan kombinasi, diawali dengan rancangan 3x3x3 kubus fleksibel yang dipatenkan pada tahun 1977, Di Jerman, pada tahun 1980 **Rubik's Cube** mendapat penghargaan sebagai *Games of The Year* , pada tahun 1982 Rubik's Cube menjadi kata tersendiri yang masuk dalam kamus *the Oxf* hingga pada tahun 2007 **Rubik's Cube** dinobatkan sebagai *The Coolest Brand in the World*.

Hingga kini Secara reguler *World Cube Association* (**WCA**) menyelenggarakan Kejuaraan Dunia Kubus RUbik (*World Rubik's Cube Championship*) untuk mengumpulkan para penggermar **Rubik's Cube** di seluruh dunia dan untuk mencari pemecah rekor baru dari beberapa kategori pertandingan berdasarkan cara penyelesaiannya; kategori biasa (*reguler*), dengan satu tangan (*one handed*), dengan mata tertutup (*blind folded*) dan kategori penyelesaian dengan kaki (*with feet*).

Rubik's Cube saat ini banyak digunakan di sekolah dan universitas untuk membantu menjelaskan partikel komplek *fisika* dan *alogaritma matematika* yang rumit, dan dengan berjalannya waktu, telah tercipta Kubus rubik dengan berbagai bentuk dan ukuran yang menakjubkan.

2.2 METODE OLD POCHMANN

[5] Metode BLD yang penulis gunakan ini adalah metode Old Pochmann. Dikatakan old bukan berarti Pochmann sudah tua, tapi ini adalah metodenya yang lama, sekaligus paling mudah dimengerti. yang dibutuhkan dalam BLD antara lain :

1. Sebuah rubik 3x3x3, diusahakan yang ada pegasnya sehingga bisa cutting corner. (cutting corner adalah kemampuan sebuah kubus rubik untuk diputar sementara sisi di sebelahnya masih bergeser beberapa derajat dari tempatnya)
2. penghalang penglihatan antara mata dan rubik..

Skill yang dibutuhkan :

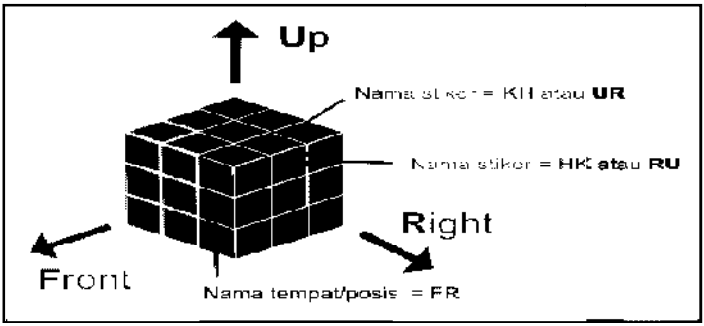
1. Pengetahuan dasar tentang rubik, termasuk di dalamnya tentang notasi - notasi dan lambing - lambang dalam menggunakan rubik.
2. Kemampuan berkonsentrasi, berkaitan erat dengan proses memorisasi.
3. Hafalan sedikit algoritma tambahan
4. Fingertick (memutar rubik yang lebih efesien dengan jari-jari, bukan genggaman tangan), ini benar-benar membantu agar rubik tidak berubah arah di tangan kita. tapi jika anda yakin bisa menjaga posisi rubik dengan putaran pada genggaman tangan, juga tidak apa.

Mengapa “mampu menyelesaikan rubik” tidak termasuk dalam skill yang dibutuhkan? karena cara menyelesaikan rubik dengan mata terbuka berbeda dengan melakukan dengan mata tertutup. anda tidak harus bisa menyelesaikan rubik dengan mata terbuka untuk bisa melakukan BLD.

Konsep dalam metode ini adalah menembak. Caranya bukanlah menembak piece ke piece lain, tapi kita menembak stiker ke stiker yang lain. lalu ada istilah setup move, maksudnya adalah gerakan untuk meletakkan sebuah stiker pada sasaran tembak, setelah melakukan algoritma utama, harus dilakukan undo setup move agar stiker tersebut kembali ke tempat yang benar setelah ditembak (ditukar). Parity eror terjadi jika kita menyelesaikan edge dan corner, yang tersisa masih ada edge-edge yang tertukar di empat tempat. hal ini terjadi karena jumlah langkah yang kita tempuh berjumlah ganjil. perlu sebuah algoritma untuk menyelesaikan parity.

Bagaimana mengenal masing-masing potongan kubus. Penuls menyebut sisi-sisi kubus sebagai K (kuning), M (Merah), B (Biru), H (Hijau), O (Oranye), P (putih). penuls meletakkan K (kuning) sebagai U (Up, atas) dan O sebagai F (Front, depan). otomatis P berada dibawah, B berada di kanan, H berada di kiri, dan M berada di belakang. anda harus mengingat warna-warna posisi ini. Gunakan Kuning sebagai atas dan Oranye sebagai depan, anda harus memegangnya demikian.

MEMBERI NAMA



Gambar 2.2 Penamaan Piece

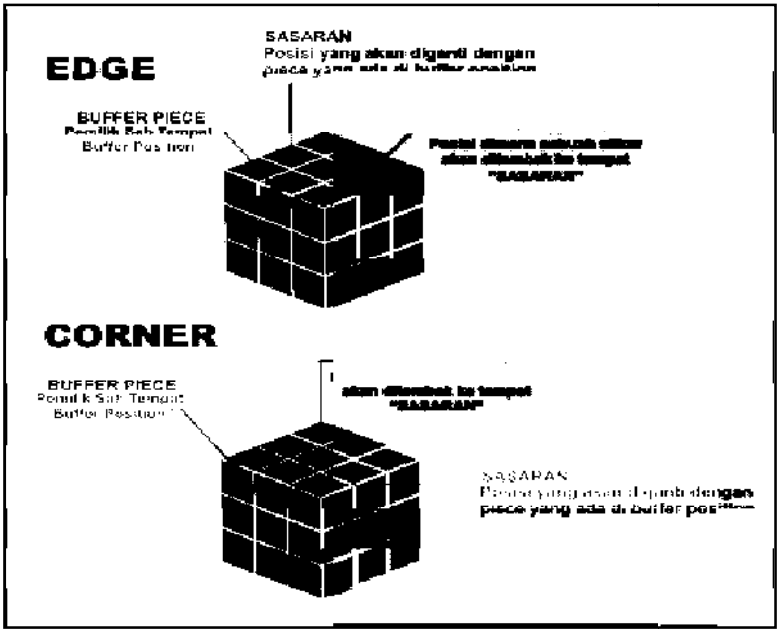
Perhatikan gambar. Penuls menyentuh piece edge yang stikernya berwarna Kuning dan Hijau. penuls tidak bisa memberinya nama, kecuali jika penuls tahu mana yang lebih penting antara warna Kuning dan Hijau di piece edge tersebut. Warna stiker yang lebih penting adalah warna stiker yang akan MENEMBAK atau DITEMBAK. Jika melihat gambar, kita tahu bahwa piece edge yang penuls sentuh tadi akan menembak ke seberang. Stiker yang menembak adalah Stiker kuning, dan yang ditembak adalah stiker kuning di seberang (UL). jadi pada piece yang penuls sentuh tadi (berwarna Kuning dan Hijau), warna stiker yang lebih penting adalah warna Kuning. Jadi penuls memberi

nama piece ini sebagai KH, bukan HK. Jika warna hijau lebih penting (sebagai stiker penembak) penuls akan memberinya nama HK.

Untuk sebuah posisi, kita memberi nama dengan Up, Down, Left, Right, Back, Front. Misal, jika pada sebuah rubik yang solved, posisi UR adalah KB, posisi RU adalah BK. dan begitu seterusnya. Lihat pada gambar, posisi yang ditunjuk disebut FR. Jika posisi yang ingin disebut adalah yang warna hijau, maka di sebut RF. ingat, stiker berbeda dengan Piece.

2.2.1 TENTANG BUFFER dan BUFFER POSITION

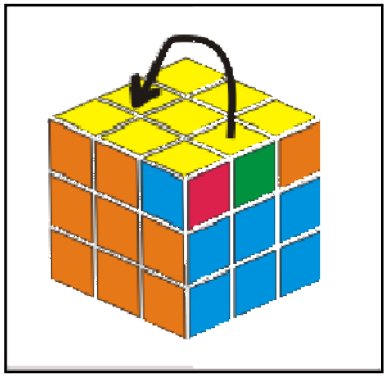
Buffer position adalah sebuah tempat dimana sebuah piece akan ditembakkan ke tempat sasaran. Dalam penyelesaian EDGE, buffer positionnya adalah edge antara sisi kanan dan sisi atas rubik atau disebut sebagai posisi UR (Up Right). dalam latihan ini (dengan Kuning sebagai atas dan oranye sebagai depan), buffer position edge-nya adalah sebuah tempat (edge) di antara sisi kuning dan sisi biru, yaitu UR. Sedangkan dalam penyelesaian CORNER, buffer positionnya adalah corner di antara Kuning, Hijau, dan Merah, yakni di belakang-atas-kiri.



Gambar 2.3 Istilah dalam Blindfolded solving

Buffer atau Buffer piece adalah sebuah piece (edge maupun corner) yang merupakan piece yang benar untuk menempati buffer position. Jika definisinya seperti ini, bukan berarti buffer harus segera diletakkan ke buffer position. piece lain harus diberikan kesempatan menempati buffer position untuk menembak ke tempat yang benar, barulah buffer piece menempati buffer position pada saat terakhir.

2.2.2 Menyelesaikan EDGE (Menembak ke UL (UP – LEFT))



Gambar 2.4 Shooting edge

Tembak EDGE = RUR'U'R'FR2U'R'U'RUR'F'

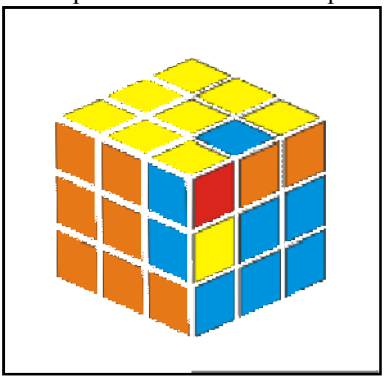
Seperti yang terlihat pada gambar, algoritma tersebut akan menembakkan piece KH (Kuning Hijau) ke posisi SASARAN. Jika sudah mengerti, kita singgung sedikit mengenai parity error.

Algoritma tersebut diterapkan pada rubik yang sudah solved. kita akan menemukan buffer piece dan piece sasaran tertukar. tapi jika kita melakukan algoritma tersebut dalam jumlah genap, cornernya kembali seperti semula. kesimpulannya, jika algoritma yang kita lakukan jumlahnya ganjil (sampai semua edge selesai), maka akan ada parity error. jika genap, maka parity pun lenyap.

2.2.3Tentang Setup Move

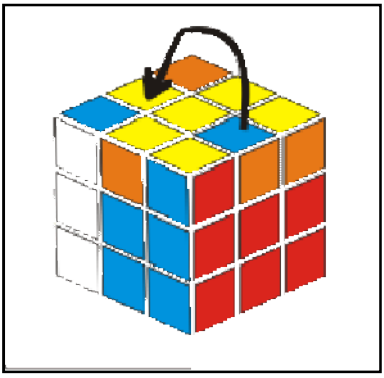
Pada sebuah rubik solved, lakukan algoritma berikut
d'L'RUR'U'R'FR2U'R'U'RUR'F'Ld
(huruf kecil di sini berarti memutar dua layer)

kita akan mendapatkan rubik tersebut seperti ini.



Gambar 2.5 Rubik Teracak

Stiker biru pada UR (Biru Oranye) atau di buffer position perlu ditukar ke RF (bukan FR). tapi yang kita punya hanyalah algoritma untuk menembak ke target, yakni ke UL, bukan ke RF. jadi, yang perlu dilakukan adalah membawa stiker RF (berwarna kuning) ke posisi UL. Gerakan membawa sebuah piece ke tempat SASARAN inilah yang disebut Setup Move.
pada kasus ini, untuk membawa stiker RF ke UL, setup movenya adalah d'L'. Lakukan setup move tersebut dan kita akan mendapatkan rubiknya akan jadi seperti ini.



Gambar 2.6 Setup Move

Langkah selanjutnya adalah menembak EDGE (shoot to UL). Setelah tertukar. selanjutnya harus mengembalikan stiker yang sudah ditukar ke tempat semula, yakni ke RF. gerakan mengembalikan-stiker-setelah-ditembak ini disebut undo setup move. undo setup move adalah kebalikan dari setup move sebelumnya. pada kasus ini setup movenya d'L', maka undo setup movenya adalah Ld. lakukan Ld maka rubik pun menjadi solved. algoritma penembakan dari buffer position ke RF disusun menjadi seperti ini.

d'L-'RUR'U'R'FR2U'R'U'RUR'F-'Ld

Tanda strip memisahkan antara setup move, algoritma, dan undo setup move.

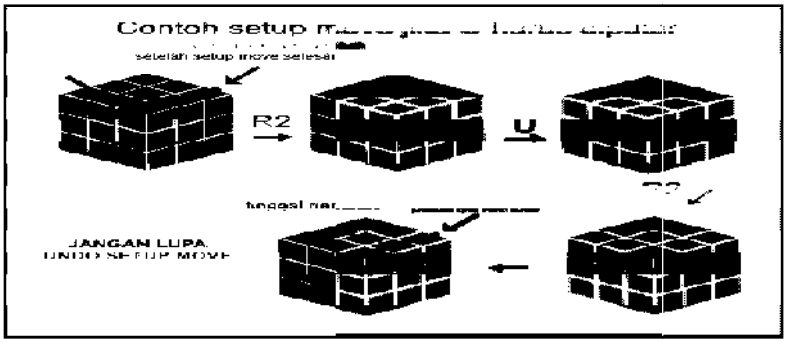
Berkiut ini adalah contoh setup move yang lain.

Ld'L-'RUR'U'R'FR2U'R'U'RUR'F-L'dL'

Untuk kasus di atas setup movenya adalah Ld'L. Target atau sasarannya adalah menembak stiker UR (Hijau) ke LU (bukan UL). Gerakan setup yang digunakan untuk membawa stiker LU ke UL adalah Ld'L. Kasus ini merupakan sebuah kasus sederhana atau kasus dasar yang wajib diketahui. Contoh kasus tersebut dapat dikembangkan sendiri sesuai keinginan untuk mendapatkan gerakan yang paling efektif dalam melakukan gerakan setup.

Algoritma nembak EDGE akan menukarkan corner di samping buffer position. artinya selama eksekusi edge dengan mata tertutup, kedua corner di samping buffer position tidak boleh berpindah tempat atau berpindah tanpa dikembalikan ke tempat semula. corner-corner tersebut terletak di U, F, B, R. jadi, selama melakukan setup move, kita tidak boleh melakukan gerakan U,F,B,R, apalagi u,f,b,r (yang sama sekali tidak berguna). Melakukan gerakan terlarang berarti merusak letak corner di samping buffer position.

Berikut ini adalah contoh kasus setup move jika gerakan upper layer (U) diperlukan.



Gambar 2.7 Setup Move dengan gerakan U

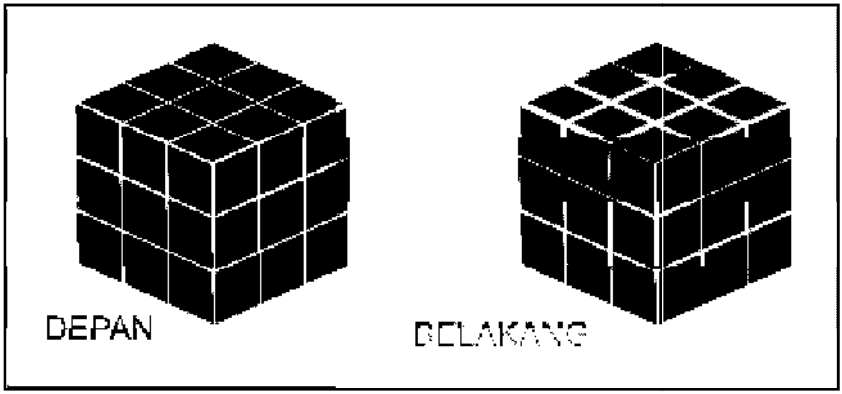
Permasalahan untuk kasus di atas adalah bagaimana jika piece sasaran berada di UF seperti digambar. Hal yang bisa dilakukan adalah gerakan terlarang yaitu U dengan syarat R harus diputar 2 kali untuk, membawa buffer position ke bawah. Buffer position beserta dua corner di sampingnya harus dibawa ke tempat dimana mereka tidak terpengaruh oleh putaran U, yakni dengan memutar R dua kali tadi dan gerakan U dilakukan seperlunya untuk membawa piece sasaran ke UL, kemudian dilakukan lagi R2.

2.2.4 Memorisasi

Cycle adalah urutan-urutan target yang perlu kita tembak, dan urutan-urutan ini perlu diingat.

Berikut ini adalah scramble untuk kasus memorisasi.

D2 R' U' B' R' U B U R U B' D2



Gambar 2.8 Cycle Edge

Hal yang harus dilakukan adalah menentukan target yang ada pada buffer position ditembak kemana, kemudian kemana, dan seterusnya. Dimulai pada piece di buffer position, yakni stiker UR. Stiker UR berwarna kuning. Artinya, kita harus membawa stiker UR ke suatu tempat di sisi U. Piece yang berwarnanya hijau, yang harus dibawa ke sisi L. jadi, stiker UR harus ditembak ke UL.

Jika target adalah piece pada posisi UL, maka akan diketahui bahwa stiker UR kuning akan tertukar dengan stiker UL hijau. Sehingga , nantinya yang akan menempati stiker UR adalah warna hijau. selanjutnya stiker UR harus dibawa ke suatu tempat di sisi L (karena ia berwarna hijau). Piece apa yang menjadi tetangga stiker UR hijau warnanya merah, dan sisi merah ada di B. jadi, stiker UR kali ini harus dibawa ke LB.

Stiker UR Hijau akan tertukar dengan stiker LB putih yang berarti Stiker UR putih ini (yang kini sudah di buffer position) harus dibawa ke D (sisi putih). sisi oranye ada di F. jadi, stiker UR putih harus ditukar ke stiker DF.

Karena stiker DF adalah oranye, maka stiker ini akan menempati stiker UR. Stiker UR oranye harus di bawa ke F dimana sisi biru berada. siapa tetangga stiker UR oranye yang memiliki warna biru yang mana sisi biru ada di R. maka stiker UR oranye harus di bawa ke FR.

Sehingga urutan cyclenya adalah UL-LB-DF-FR.

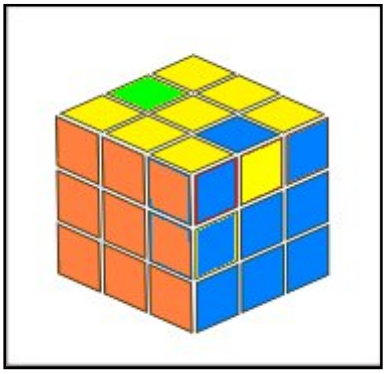
[shoot to UL] R U R' U' R' F R2 U' R' U' R U R' F' (tanpa setup move)
[shoot to LB] d L' – R U R' U' R' F R2 U' R' U' R U R' F' – L d'
[shoot to DF] D' L2 – R U R' U' R' F R2 U' R' U' R U R' F' – L2 D
[shoot to FR] d2 L – R U R' U' R' F R2 U' R' U' R U R' F' – L' d2

Contoh Cycle yang lain.
algoritma pengacakan = R2 U R D' U F' U' F' D U' R'
jika anda berpikir dengan benar, maka cyclenya adalah = UB LF RD FL
anda harus mengingat cycle tersebut.
berikut setup move-nembak EDGE-undo setup move-nya.
[shoot to UB] R2 U' R2 – R U R' U' R' F R2 U' R' U' R U R' F' – R2 U R2
[shoot to FL] L' – R U R' U' R' F R2 U' R' U' R U R' F' – L
[shoot to RD] D' F L' F' – R U R' U' R' F R2 U' R' U' R U R' F' – F L F' D
[shoot to LF] d' L – R U R' U' R' F R2 U' R' U' R U R' F' – L' d

2.2.5 Flip Edge

Flip Edge adalah kondisi dimana edge telah berada pada posisinya yang benar namun dalam orientasi yang salah sehingga Rubiks belum bisa dikatakan solve. Oleh karena itu diperlukan sebuah algoritma yang dapat merubah orientasi edge tersebut. Selain pada edge kondisi flip juga mungkin terjadi pada corner piece namun algoritma yang digunakan berbeda dengan algoritna untuk flip edge.

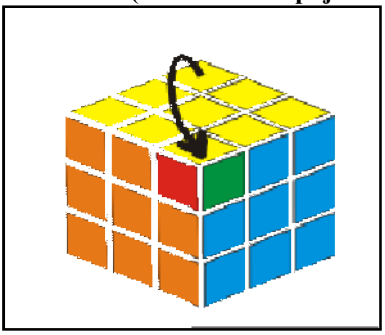
Berikut ini adalah gambar tentang flip edge yang terjadi pada Rubiks 3x3.



Gambar 2.9 Flip Edge

Untuk kasus seperti gambar di atas algoritmanya adalah :y-M' U M' U M' U2 M U M U M U2

2.2.6 Menyelesaikan Corner (Nembak dari pojok ke pojok)



Gambar 2.10 Shooting Corner

algoritma nembak corner adalah sebagai berikut :

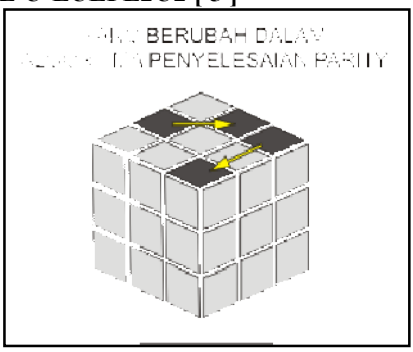
FRU'R'U'RUR'F'RUR'U'R'FR'F

Scramble tersebut dilakukan pada rubik yang solved, maka pojok atas akan tertukar, seperti pada gamabar, dan edge di atas di samping buffer position juga tertukar. Jika ganjil, maka parity, jika genap, parity pun lenyap.

2.2.7 Menyelesaikan Parity

Parity akan muncul bila jumlah langkahnya ganjil.
berikut algoritma penyelesaian parity :

U'LU2L'U2LF'L'U'LULFL2U2 [5]



Gambar 2.11 Parity

2.3 METODE M2R2 DAN BAYER HARDWICK

Metode untuk menyelesaikan Rubik's dengan mata tertutup sangat beragam. Namun hanya 3 metode yang paling sering digunakan oleh cuber di seluruh dunia. 3 metode tersebut adalah metode Old Pochmann, metode M2R2 dan metode Bayer-Hardwick atau lebih dikenal dengan sebutan *BH method*.

. Metode Old Pochmann adalah metode yang sangat terkenal dikalangan cuber karena metode ini adalah metode untuk menyelesaiakan Rubik's dengan mata tertutup yang ditemukan pertama kali di dunia oleh Steffan Pochmann.

Metode ini dikenal sebagai metode yang paling mudah untuk dipelajari karena algoritma yang dgunakan tidak terlalu banyak yaitu hanya 5 algoritma utama dan yang lain adalah penyesuaian dari algoritma tersebut dan kelima algoritma tersebut merupakan algoritma Fridrich yaitu algoritma untuk menyelesaikan Rubiks dengan mata terbuka sehingga hamper semua cuber yang bisa menyelesaikan Rubik's dengan mata terbuka akan sangat familiar dengan lima algoritma tersebut.

Metode ini yang penuls gunakan sebagai dasar pembuatan tugas akhir penuls karena kelebihan-kelebihannya tersebut. Namun metode ini juga memiliki kekurangan yaitu jumlah gerakannya sangat banyak untuk menyelesaikan Rubiks dengan metode ini memerlukan kurang lebih 200 sampai 250 gerakan. Hal ini terjadi karena jumlah algoritma nya yang sedikit sehingga sering kali mengulang algortima yang sama.

Berikut ini adalah penjelasan dari metode M2R2 dan Bayer Hardwick:

2.3.1. Metode M2R2

Metode M2R2 adalah metode yang juga ditemukan oleh Steffan Pochmann namun metode ini tidak lagi menggunakan algoritma yang ada pada metode Old Pochmann. Semua algoritma yang digunakan dalam metode ini murni untuk menyelesaikan dengan mata tertutup dan tidak mengadopsi atau mengambil algoritma Fridrich.

Pada dasarnya konsep dari metode ini sama dengan Old Pochmann yaitu menembak satu demi satu piece untuk diletakkan pada posisinya yang seharusnya. Perbedaannya terletak pada posisi buffer, pada Old Pochmann buffer edge terletak pada posisi UR (penjelelasan penamaan posisi ada pada bab II) dan untuk corner pada posisi UBL. Namun pada metode M2R2 buffer terletak pada posisi DF dan buffer untuk corner pada posisi DRF [1].

Metode ini sagat efektif karena untuk meletakkan piece pada posisinya yang seharusnya cukup menggunakan satu gerakan yaitu gerakan M2 (menggerakkan middle layer pada Rubik sehingga berputar 180 derajat) untuk edge dan R2 untuk corner. Sehingga dengan metode ini untuk menyelesaikan Rubiks diperlukan 150 sampai dengan 200 gerakan saja. Namun yang membuat sulit adalah gerakan *setup* dan *undo setup* karena semua setup dan undo setup harus dihafalkan dan tidak bisa dikembangkan secara intuitif seperti halnya pada Old Pochmann.

2.3.2. Metode Bayer Hardwick

Metode ini adalah metode yang paling efektif untuk menyelesaikan Rubiks dengan mata tertutup yang ditemukan oleh dua orang yaitu Bayer dan Criss Hardwick. Pada dasarnya konsep dari metode ini adalah hamper sama dengan Old Pochmann yaitu menggunakan buffer pada posisi UR untuk edge namun untuk corner berbeda yaitu buffer pada posisi URB.

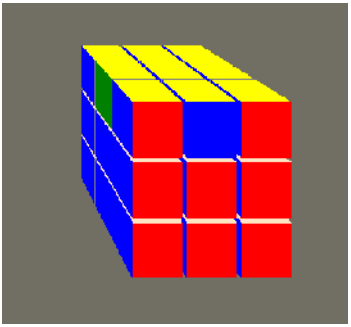
Berbeda dengan 2 metode sebelumnya (Old Pochmann dan M2R2) metode ini sangat efektif karena algoritma yang digunakan dapat

menembak atau meletakkan 2 piece sekaligus pada tempatnya yang seharusnya. Sementara pada dua metode sebelumnya hanya dapat meletakkan satu demi satu piece ke posisinya yang seharusnya. Sehingga untuk menyelesaikan Rubiks hanya diperlukan sekitar 100 sampai 150 gerakan saja. Namun untuk menguasai metode ini diperlukan kemampuan dan pengalaman yang cukup tinggi karena jumlah algoritma yang harus dihafalkan adalah 150 algoritma untuk edge dan 378 algoritma untuk corner. Oleh karena itu metode ini tidak dianjurkan untuk pemula karena sangat sulit dan perlu waktu yang lama untuk mempelajarinya.

Berikut ini adalah perbandingan jumlah gerakan dari masing – masing metode dengan menggunakan acakan yang sama yaitu R U' R U R U R U' R' U' R2. Rubiks akan tampak seperti dibawah ini.

Tabel 2.1 Perbandingan metode Old Pochmann, M2R2 dan Bayer HaSrdwick

Metode	Target	Setup	Shoot	Undo Setup	Total gerakan
Old Pochmann	UF UL	- -	B2 L U L' B2 R D' R D R2 R U R' U' R' F R2 U' R' U' R U R' F'	- -	24 moves
M2R2	UL UR UF UL	LU'L'U R'URU' - LU'L'U	M2 M2 U2M'U2M' M2	U'LUL' UR'U'R - U'LUL'	31 moves
Bayer Hardwick	UF-UL	-	F L E' L' U2 L E L' U2 F'		10 moves



Gambar 2.12 pola warna Rubiks

Ketiga metode tersebut masing – masing memiliki kelebihan dan kekurangan seperti yang telah dijelaskan di atas. Namun dari ketiga metode tersebut yang paling cocok untuk tugas akhir ini adalah metode Old Pochmann. Metode Old Pochmann memiliki konsep yang sederhana dan mudah dimengerti sehingga sangat cocok diterapkan oleh para pemula yang ingin belajar menyelesaikan Rubiks dengan mata tertutup walaupun memiliki gerakan yang cukup banyak namun gerakannya mudah dihafalkan.

BAB III
PERENCANAAN DAN PERANCANGAN SISTEM

Pada bab perancangan dan perencanaan sistem ini akan dibahas tentang proses skenario apa saja yang ada dalam aplikasi ini dan bagaimana cara menerapkan algoritma Old Pochmann atau algoritma menyelesaikan Rubik's dengan mata tertutup.

3.1 PERENCANAAN SISTEM

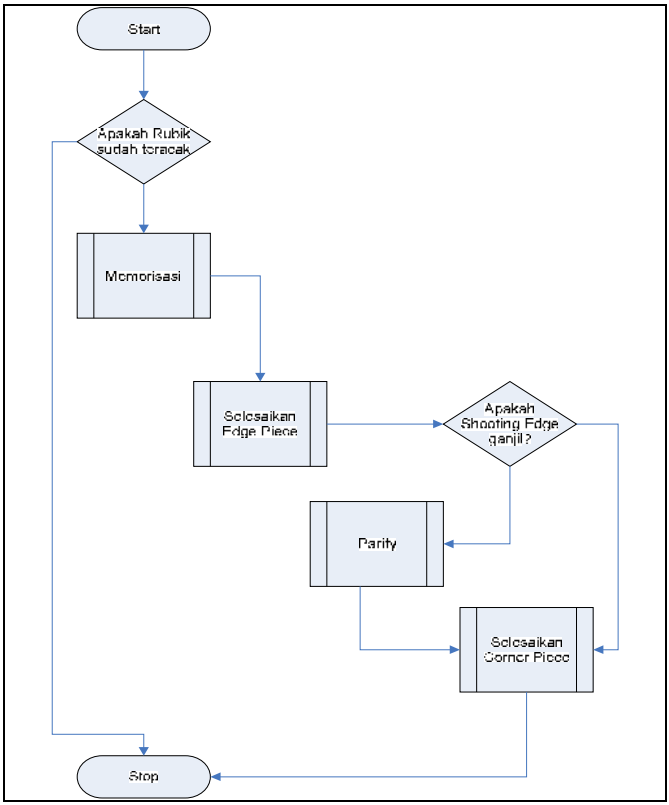
Disni akan dijelaskan mengenai perancangan system. Pada perancangan sistem ini ada 2 aspek yaitu Perencanaan metode menyelesaikan Rubiks dengan mata tertutup dan perencanaan database.

3.1.1 Perencanaan Metode Menyelesaikan Rubiks dengan Mata Mertutup.

Untuk menyelesaikan Rubik's dengan mata tertutup memiliki tahapan yang berbeda dengan mata terbuka. Adapun tahapan – tahapan dalam menyelesaikan Rubik dengan mata tertutup adalah memorisasi, menyelesaikan edge, menyelesaikan parity jika diperlukan, menyelesaikan corner.

Memorisasi adalah tahapan dimana user harus menghafalkan kombinasi warna yang ada pada virtual Rubiks cube. Setelah melakukan memorisasi user menyelesaikan edge piece yaitu piece yang memiliki 2 kombinasi warna terlebih dahulu. Selain menyelesaikan edge user juga harus menghitung jumlah target yang diselesaikan, hal ini dikarenakan jika jumlah target yang di selesaikan adalah ganjil maka user harus melakukan parity terlebih dahulu kemudian menyelesaika corner dan jika jumlah target genap maka user bisa langsung menuju tahap selanjutnya yaitu menyelesaikan corner piece. Corner piece adalah piece yang berada pada posisi pojok atau piece yang memiliki tiga kombinasi warna.

Untuk lebih jelasnya penuls gambarkan pada *Flow Chart* . Berikut ini adala *flow chart* untuk menyelesaian Rubiks cube dengan mata tertutup:



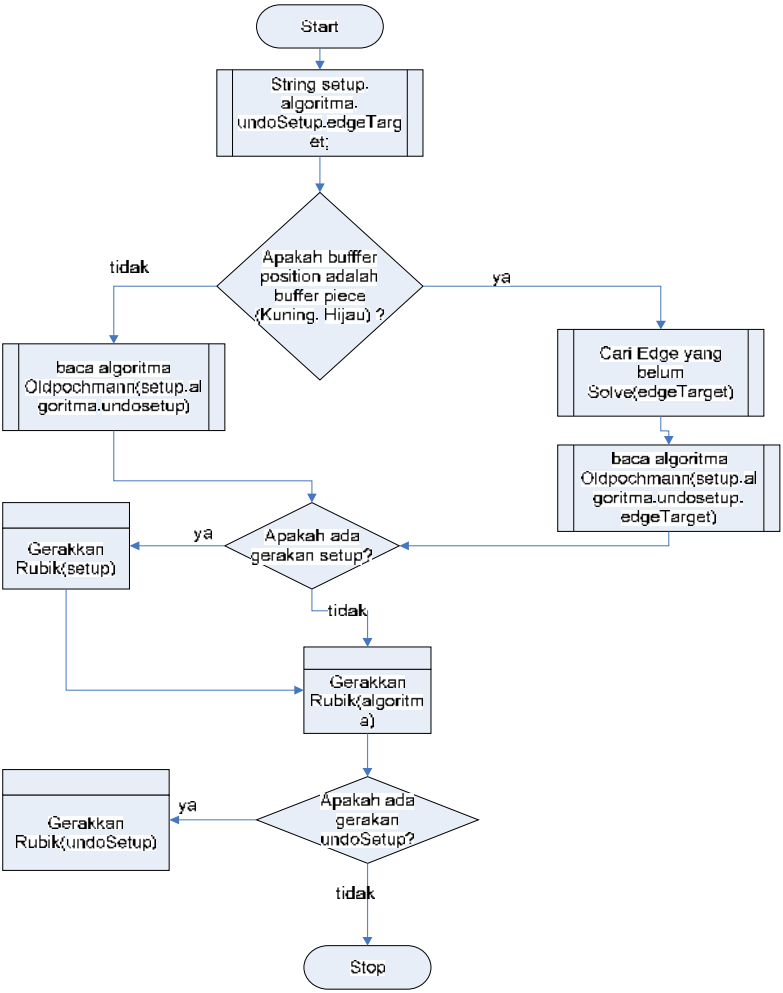
Gambar 3.1 Flowchart menyelesaikan Rubik mata tertutup

3.1.1.1 Memorisasi

Memorisasi adalah tahap awal yang harus dilakukan dalam menyelesaikan Rubik's dengan mata tertutup. Pada tahap ini sistem akan membuat siklus yang akan dgunakan untuk tahap selanjutnya. Hal pertama yang dilakukan sistem adalah membernama untuk masing-masing piece.

3.1.1.2 Menyelesaikan Edge

Edge adalah piece pada Rubik's yang memiliki dua warna atau stiker. Untuk menyelesaikan edge, sistem akan mengacu pada buffer position. Buffer position terletak pada edge (atas, kanan) atau UR atau piece yang memiliki warna kuning hijau.



Gambar 3.2 Flow chart Menyelesaikan Edge

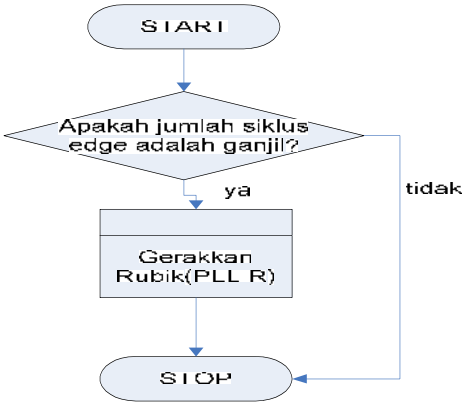
Berikut ini adalah kode untuk mengimplementasikan Flow Chart di atas:

```
  public void solveEdgeBLD()
        {
            string bufferTop = "", bufferRight = "", setup
= "", undosetup = "", idalgo = "", algo = "";
            bufferTop = "" + indexcolor[5, 1, 2];
            bufferRight = "" + indexcolor[1, 0, 1];
            targetEdge = bufferTop + " " + bufferRight;
            if ((bufferRight != "5" || bufferTop != "2") &&
(bufferRight != "2" || bufferTop != "5"))
            {
                dbrubik.command("select
setup,undo_setup,id_algo from pochman_algorithm where
buffer='" + bufferTop + bufferRight + "'");
                DataSet ds = dbrubik.select();
                //
MessageBox.Show(""+ds.Tables[0].Rows[0][0]);
                if (ds.Tables[0].Rows.Count > 0)
                {
                    setup =
ds.Tables[0].Rows[0][0].ToString();
                    undosetup =
ds.Tables[0].Rows[0][1].ToString();
                    idalgo =
ds.Tables[0].Rows[0][2].ToString();
                    if (setup != "")
                    {
                        generateScamble(setup);
                    }
                    dbrubik.command("select algoritma from
fridrich_algorithm where id_algo=" + idalgo);
                    DataSet ds2 = dbrubik.select();


generateScamble(ds2.Tables[0].Rows[0][0].ToString());
                    algo =
ds2.Tables[0].Rows[0][0].ToString();
                    if (undosetup != "")
                        generateScamble(undosetup);
                    algoBLD = "" + setup + algo +
undosetup;
                }
            }
            else
            {
                string[] temp = new string[11];
                temp[0] = "010 312
L3R1U1R3U3R3F1R2U3R3U3R1U1R3F3L1";
                temp[1] = "510 301
```

```
                int layer1 = 0, row1 = 0, column1 = 0;
                int layer2 = 0, row2 = 0, column2 = 0;
                int i = 0;
                for (int k = 0; k < 11; k++)
                {
                    layer1 =
Convert.ToInt16(temp[k].Substring(0, 1));
                     row1 =
Convert.ToInt16(temp[k].Substring(1, 1));
                   column1 =
Convert.ToInt16(temp[k].Substring(2, 1));
                    layer2 =
Convert.ToInt16(temp[k].Substring(4, 1));
                     row2 =
Convert.ToInt16(temp[k].Substring(5, 1));
                   column2 =
Convert.ToInt16(temp[k].Substring(6, 1));
                     i++;
       if (indexcolor[layer1, row1, column1] !=
indexcolor[layer1, 1, 1] || indexcolor[layer2, row2,
column2] != indexcolor[layer2, 1, 1])
                     {

        generateScamble(temp[k].Substring(7).Trim());
                         break;
                 }
```

3.1.1.3 Menyelesaikan Parity

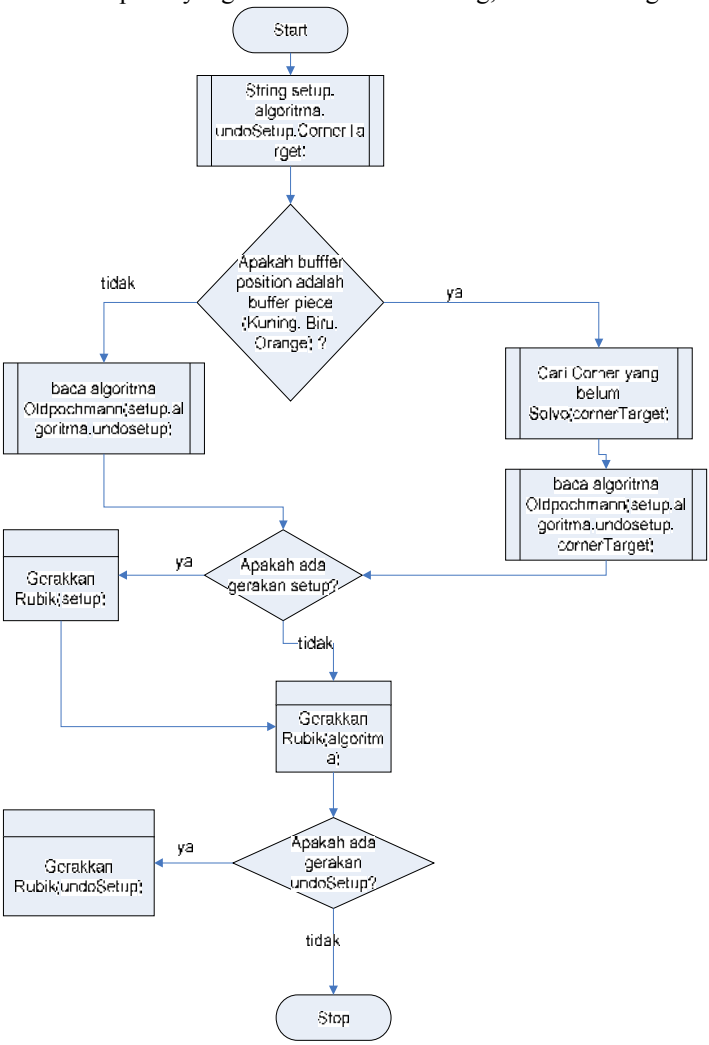
Parity akan terjadi jika siklus edge berjumlah ganjil dan untuk menyelesaikan parity hanya dilakukan satu algoritma fridrich yaitu algoritma yang bernama PLL R seperti yang telah dijelaskan pada Bab II. Sistem akan menjalankan algoritnma tersebut dengan perintah sebagai berikut:



Gambar 3.3 Flowchart menyelesaikan Parity

3.1.1.4 Menyelesaikan Corner

Corner adalah piece yang memiliki tiga warna atau stiker. Untuk menyelesaikan corner, sistem akan mengacu pada buffer position corner. Buffer position corner terletak pada piece (atas, kiri, belakang) atau URB atau piece yang memiliki warna kuning, biru dan orange.



Gambar 3.4 Flowchart menyelesaikan corner

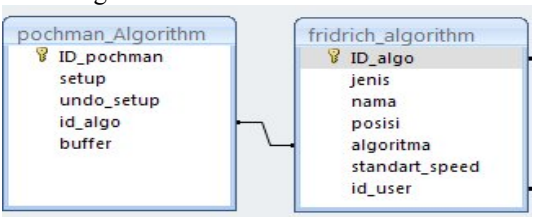
Berikut ini adalah kode untuk mengimplementasikan Flow Chart di atas:

```
public void solveCornerBLD()
        {
            string bufferTop = "", bufferBack = "",
bufferLeft = "", setup = "", undosetup = "", idalgo =
"", algo = "";
            bufferTop = "" + indexcolor[5, 0, 0];
            bufferBack = "" + indexcolor[2, 2, 0];
            bufferLeft = "" + indexcolor[3, 0, 0];
            string PLLy =
"F1R1U3R3U3R1U1R3F3R1U1R3U3R3F1R1F3";

            if ((bufferTop != "5" || bufferBack != "3"
|| bufferLeft != "0") && (bufferTop != "5" || bufferBack
!= "0" || bufferLeft != "3") && (bufferTop != "3" ||
bufferBack != "5" || bufferLeft != "0") && (bufferTop !=
"3" || bufferBack != "0" || bufferLeft != "5") &&
(bufferTop != "0" || bufferBack != "3" || bufferLeft !=
"5") && (bufferTop != "0" || bufferBack != "5" ||
bufferLeft != "3"))
            {
                dbrubik.command("select
setup,undo_setup,id_algo from pochman_algorithm where
buffer='" + bufferTop + bufferLeft + bufferBack + "'");
                DataSet ds = dbrubik.select();

          if (ds.Tables[0].Rows.Count > 0)
                {

         setup = ds.Tables[0].Rows[0][0].ToString();
         undosetup = ds.Tables[0].Rows[0][1].ToString();
         idalgo = ds.Tables[0].Rows[0][2].ToString();
                    if (setup != "")
                    {
                        generateScamble(setup);
                    }
         dbrubik.command("select algoritma from
fridrich_algorithm where id_algo=" + idalgo);
         DataSet ds2 = dbrubik.select();


generateScamble(ds2.Tables[0].Rows[0][0].ToString());
algo = ds2.Tables[0].Rows[0][0].ToString();
          if (undosetup != "")
          generateScamble(undosetup);
          algoBLD = "" + Convert.ToChar(13) +
Convert.ToChar(10) + "setup= " + setup +
Convert.ToChar(13) + Convert.ToChar(10) + " algo= " +
algo + Convert.ToChar(13) + Convert.ToChar(10) + " undo
```

```
    else
    {
if (indexcolor[3, 2, 0] != indexcolor[3, 1, 1])
    {
      setup = "D2F3"; undosetup = "F1D2";
    }
else if (indexcolor[0, 2, 0] != indexcolor[0, 1, 1])
    {
      setup = "D1F3"; undosetup = "F1D3";
    }
else if (indexcolor[1, 2, 0] != indexcolor[1, 1, 1])
    {
       setup = "F3"; undosetup = "F1";
    }
else if (indexcolor[2, 0, 2] != indexcolor[2, 1, 1])
    {
       setup = "D3F3"; undosetup = "F1D1";
    }
else if (indexcolor[3, 0, 2] != indexcolor[3, 1, 1])
    {
       setup = "F1"; undosetup = "F3";
    }
else if (indexcolor[2, 2, 2] != indexcolor[2, 1, 1])
    {
        setup = "R3"; undosetup = "R1";
    }
        generateScamble(setup + PLLy + undosetup);
    }
         algoBLD = setup + PLLy + undosetup;}
```

3.1.2 Perancangan database

Pada aplikasi ini memerlukan database untuk menyimpan algoritma yang akan digunakan dalam menyelesaikan Rubik's cube dengan mata tertutup yaitu algoritma Old Pochmann. Namun algoritma Old Pochmann sendiri pada dasarnya mengacu pada beberapa algoritma Fridrich yang merupakan algoritma yang digunakan untuk menyelesaikan Rubik's dengan mata terbuka. Sehingga database pada aplikasi ini memiliki dua tabel yang saling berhubungan yaitu tabel Pochmann_algorithm dan Fridrch_algorithm. Hubungan antar kedua tabel tersebut adalah sebagai berikut:



`Gambar 3.5 ERD

Game menu	
Challenge	Cube Timer
Tutorial	

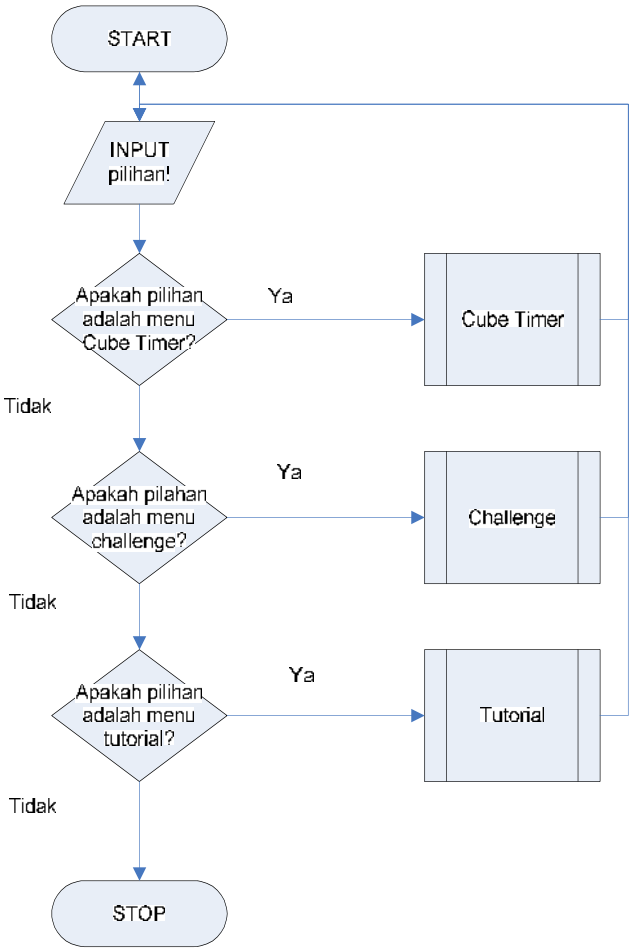
Chalenge	
Scramble	auto
	manual
Give Up	Solve edge
	Fix Parity
	solve corner

Tutorial	
Solving Edge	Shoot 1 edge
	Shoot 2 Edge
	Fix Parity
Solving Corner	Shoot 1 corner
	Shoot 2 Corner
Notation	
Memorization	

Struktur Rubik	
Size	Index Color
Create new Cube	Location
Rotate cube	Camera

Gambar 3.6 Rancangan Sistem Secara Umum

Untuk menjelaskan alur pada aplikasi ini, berikut adalah gambaran alur aplikasi dalam bentuk flow chart diagram.

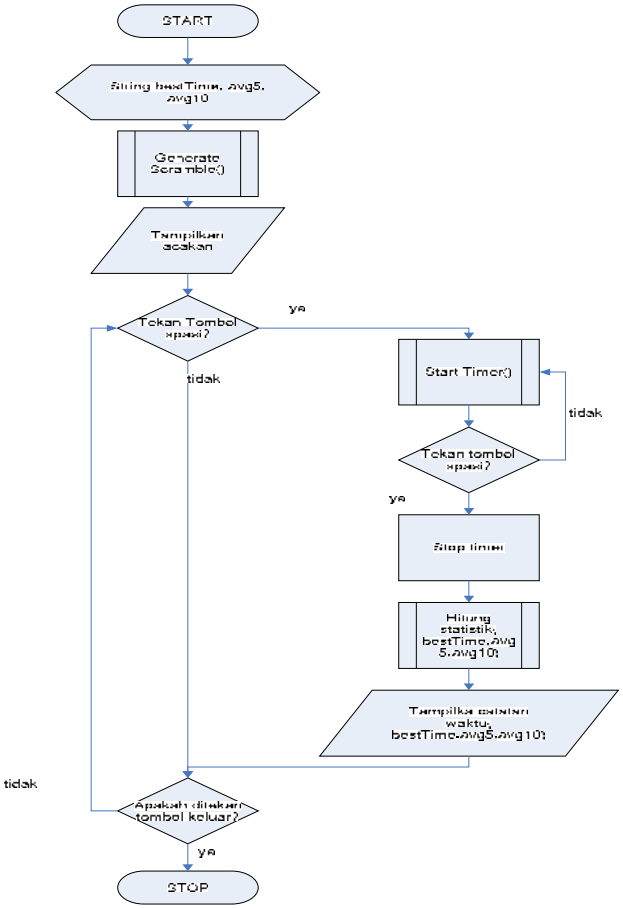


Gambar 3.7 Flow Chart Sistem

Pada flow chart di atas terdapat beberapa prosedur yang perlu dijelaskan lebih lanjut yaitu proses Cube Timer, proses Challenge dan Tutorial. Berikut ini adalah penjelasan dari ketiga prosedur tersebut.

3.2.1. Cube Timer

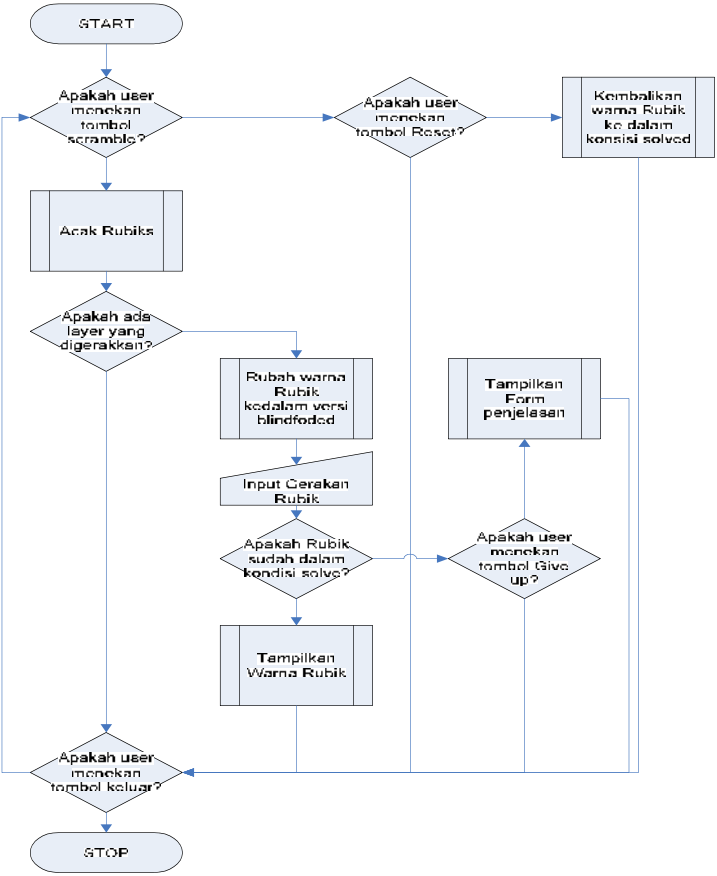
Cube timer adalah sebuah menu atau form yang dibuat untuk menghitung catatan waktu dari user dalam menyelsaikan Rubiks dengan menggunakan scramble atau acakan standart WCA (World Cube Accosiation) yang ditampilkan pada menu ini. Berikut ini adalah flow chart untuk sub menu cube timer.



Gambar 3.8 flow chart Cube Timer

3.2.2. Challenge

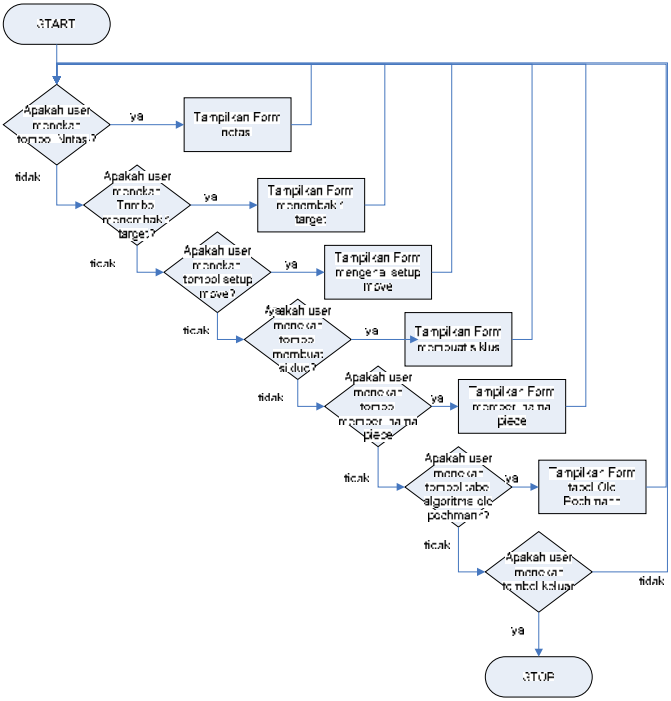
Challenge adalah sub menu atau form yang digunakan user untuk menyelsaikan virtual Rubiks dengan menggunakan keyboard. Jika user tidak dapat menyelesaikan Rubiks tersebut maka disediakan tombol give up yang berfungsi untuk menjelaskan bagaimana langkah yang seharusnya dilakukan untuk menyelesaikan Rubiks tersebut. Berikut ini adalah flow chart untuk sub menu challenge.



Gambar 3.9 Flow Chart Challenge

3.2.3. Tutorial

Tutorial adalah sub menu atau form yang berisi tutorial untuk menjelaskan metode Old Pochmann. Berikut ini adalah flow chart untuk sub menu tutorial.



Gambar 3.10 Flow chart tutorial

3.3 Membuat Rubik's 3 dimensi

Pada aplikasi ini terdapat 2 class utama yaitu class Rubik.cs yang berfungsi untuk mengelola warna dan algoritma serta class Rubikscube.cs unt

3.3.1 Membuat class Rubik's

Class Rubik digunakan untuk mengelola wana Rubik's dan mengimplementasikan algoritma Old Pochmann yang telah disimpan pada database. Desain class Rubik's yang penuls buat adalah sebagai berikut.

Constructor class Rubik adalah proses yang dilakukan ketika class pertama kali dipanggil atau di *Create*. Source code untuk constructor kelas Rubik adalah sebagai berikut:

```
  public rubik(int color1, int color2, int color3, int
color4, int color5, int color6)
        {

            indexcolor = new int[6, 3, 3];
            for (int j = 0; j < 3; j++)
            {
                for (int k = 0; k < 3; k++)
                {
                    indexcolor[0, j, k] = color1;
                    indexcolor[1, j, k] = color2;
                    indexcolor[2, j, k] = color3;
                    indexcolor[3, j, k] = color4;
```

Listing Code Constructor Kelas Rubik

3.3.1.1 Rotasi Cube

Ketika Rubik digerakkan maka komposisi warnanya akan berubah (bertukar posisi). Berikut ini adalah contoh kode untuk menukar warna Rubik pada layer kanan.

```
private void rotateRightLayer()
        {//up
            setTemporary(1);
            indexcolor[1, 0, 0] = temp[1, 2, 0];
            indexcolor[1, 0, 1] = temp[1, 1, 0];
            indexcolor[1, 0, 2] = temp[1, 0, 0];

            indexcolor[1, 1, 0] = temp[1, 2, 1];
            indexcolor[1, 1, 2] = temp[1, 0, 1];

            indexcolor[1, 2, 0] = temp[1, 2, 2];
            indexcolor[1, 2, 1] = temp[1, 1, 2];
            indexcolor[1, 2, 2] = temp[1, 0, 2];
          }
```

Listing Code Rotasi Cube

3.3.1.2 Scramble

Scamble adalah urutan gerakan untuk mengacak Rubik. Untuk mengeacak Rubik memiliki aturan yaitu tidak boleh melakukan gerakan yang sama. Contoh jika acakan pertama adalah R maka acakan selanjutnya tidak boleh R2 maupun R'. Berikut ini adalah kode untuk mendapatkan acakan yang benar pada Rubik 3x3.

```
public void scramble()
        {
            string[] move = new string[] { "R", "L", "F",
"B", "D", "U" };
            Cscramble = "";

            int[] randomInt = new int[25];
            Random r = new Random();
            for (int i = 0; i < 25; i++)
            {
                randomInt[i] = r.Next(0, 6);
                if (i > 0)
                {
          while (randomInt[i] ==randomInt[i - 1])
                {
                    randomInt[i] = r.Next(0, 6);
                }
                }
 Cscramble = Cscramble + move[randomInt[i]] + r.Next(1,
4);
```

Listing Code Scramble

3.3.1.3 CekSolved

Method cekSolved ini selalu digunakan untuk mengecek apakah Rubik sudah dalam konsisi solve (semua piece berada pada posisinya yang benar). Berikut ini adalah kode method ceksolved().

```
public string cekSolved(){        {
            string status = "";
            int truepiece = 0;
            for (int i = 0; i < 6; i++)
                for (int j = 0; j < 3; j++)
                    for (int k = 0; k < 3; k++)
          if (indexcolor[i, j, k] == indexcolor[i, 1, 1])
                        truepiece++;
            if (truepiece == 54)
                status = "solved";
            else
                status = "not solved";
            return status;
```

3.3.1.4 Generate Scramble

Setelah memiliki acakan yang benar maka langkah selanjutnya adalah menerapkan acakan tersebut pada Rubik yang dalam konsidi solved. Pada aplikasi ini menggunakan skema warna merah di depan dan kuning di atas. Berikut ini adalah kode untuk melakukan Scramble:

```
public void generateScamble()
        {
            string Tscramble;
            Tscramble = Cscramble;
            string moveg = "";
            for (int i = 0; i < 25; i++)
            {
                moveg = Tscramble.Substring(i * 2, 2);
                for (int j = 0; j <
Convert.ToInt16(Tscramble.Substring(i * 2 + 1, 1)); j++)
                {
                    if (moveg.Substring(0, 1) == "R")
                        right();
                    else if (moveg.Substring(0, 1) == "L")
                        left();
                    else if (moveg.Substring(0, 1) == "F")
                        front();
                    else if (moveg.Substring(0, 1) == "B")
                        back();
                    else if (moveg.Substring(0, 1) == "D")
                        down();
                    else if (moveg.Substring(0, 1) == "r")
                        rightSmall();
                    else if (moveg.Substring(0, 1) == "l")
```

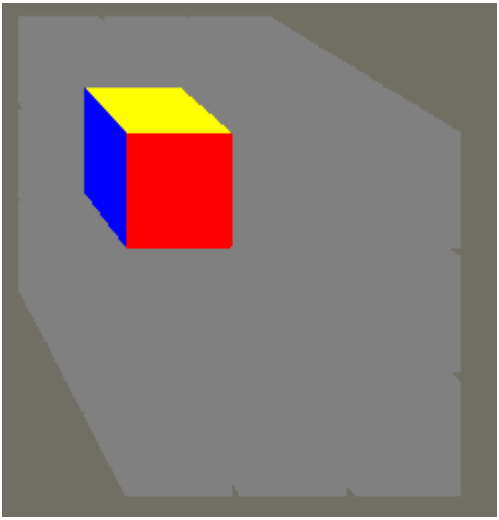
Listing Code generateScramble

3.4 Membuat class Rubik's 3 dimensi

Untuk membuat Tampilan Rubik's 3 dimensi dan animasinya penuls memanfaatkan class Rubik's yang penuls dapat kan dari www.codeproject.com. Pada class tersebut sudah terdapat fungsi-fungsi untuk menggerakan Rubik's dengan sempurna namun class tersebut masih memiliki beberapa kelemahan atau bug yaitu jika Rubik's digerakkan lebih dari 100 gerakan maka gerakan Rubik's akan menjadi sangat lambat. Oleh karena itu penuls menggabungkan 2 class ini untuk menyelesaikan masalah tersebut.

3.4.1 Pembuatan Kubus

Untuk membuat animasi Rubik's secara utuh pertama-tama di buat object kubus terlebih dahulu. Dalam kasus ini pembuatan kubus disimpan pada class Cube.cs. Berikut ini adalah gambar kubus yang terbentuk dari kelas Cube.



Gambar 3.11 Satu cube

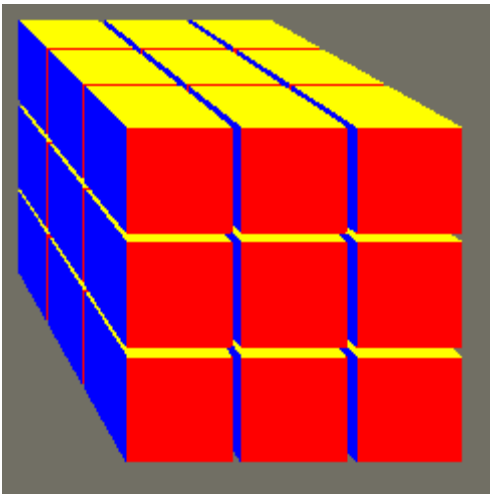
3.4.2 Penyusunan Kubus Menjadi Rubik's

Untuk membuat sebuah Rubiks yang utuh maka diperlukan 27 kubus yang memiliki skema warna yang sama. Berikut ini adalah kode untuk menata 27 Rubik tersebut pada class Rubikscube.

```
public RubikCube()
        {
            double xplus = 15, yplus = 15, zplus = 15;
            int offset = 0;
            double xspace = 1, yspace = 1, zspace = 1;
            for (int j = 0; j < 3; j++)
            {
                for (int k = 0; k < 3; k++)
                {
                    for (int i = 0; i < 3; i++)
                    {
     cubes[i + k * 3 + j * 3 * 3] = new cube(i + k * 3 + j *
3 * 3);

  CubesAtposition.Add(i + k * 3 + j * 3 * 3, i + k * 3 + j *
3 * 3);

                        xplus = 11;
                        yplus = 11;
                        zplus = 11;

         cubes[i + k * 3 + j * 3 * 3].POINTS[0] = new
Point3D(1, offset + i * xplus + i * xspace, yplus + j * yplus
+ j * yspace, zplus + k * zplus + k * zspace);
                        cubes[i + k * 3 + j * 3 *
3].POINTS[1] = new Point3D(2, xplus + i * xplus + i * xspace,
yplus + j * yplus + j * yspace, zplus + k * zplus + k *
zspace);
                        cubes[i + k * 3 + j * 3 *
3].POINTS[2] = new Point3D(3, xplus + i * xplus + i * xspace,
offset + j * yplus + j * yspace, zplus + k * zplus + k *
zspace);
                        cubes[i + k * 3 + j * 3 *
3].POINTS[3] = new Point3D(4, offset + i * xplus + i *
xspace, offset + j * yplus + j * yspace, zplus + k * zplus +
k * zspace);
cubes[i + k * 3 + j * 3 * 3].POINTS[4] = new Point3D(5,
offset + i * xplus + i * xspace, yplus + j * yplus + j *
yspace, offset + k * zplus + k * zspace);
                        cubes[i + k * 3 + j * 3 *
3].POINTS[5] = new Point3D(6, xplus + i * xplus + i * xspace,
yplus + j * yplus + j * yspace, offset + k * zplus + k *
zspace);
                        cubes[i + k * 3 + j * 3 *
3].POINTS[6] = new Point3D(7, xplus + i * xplus + i * xspace,
```

Listing Code Constructor class Rubikscube

Hasilnya pada layar akan terlihat seperti gambar berikut:



Gambar 3.12 susunan 27 cube menjadi Rubiks

3.4.3 Rotasi

Seperti halnya kelas Rubik pada sub bab sebelumnya, pada class Rubikscube ini juga memiliki method untuk melakukan rotasi. Berikut ini adalah kode untuk melakukan rotasi pada layer atas (*upper layer*).

```
    private void Up()
        {
         myRubik.Up();
         RubikCube.done = true;
         rubikCube.sidetorotate =
SideToRotate.TopSideAntiClockWise;
            for (int i = 1; i <= 6; i = i + 1)
            {
                rubikCube.SetLayerAngles(new Angles(0, 360
               - 15 * i, 0),
               SideToRotate.TopSideAntiClockWise, 2);
                this.Refresh();
            }
            refreshCube();
```

Listing Code gerakan Up

3.5 Menggabungkan 2 class

Untuk membuat Rubikscube dapat mengimplementasikan algoritma Old Pochmann maka perlu dilakukan penggabungan 2 class yaitu Class Rubik dan Class Rubikscube sebagai animasi. Selain itu penggabungan ini juga berfungsi untuk me-*refresh* animasi Rubik sehingga gerakannya tetap normal dan tidak semakin lambar. Berikut ini adalah kode untuk melakukan penggabungan tersebut.

```
public void refreshCube()
{
 rubikCube = new RubikCube("BLD");
            //bottom layer
 if (statusColor == "OPEN")
     {
rubikCube.cubes[0].Colors[2] = warna[myRubik.getcolor(2, 0, 0)];
rubikCube.cubes[0].Colors[4] = warna[myRubik.getcolor(3, 2, 0)];
rubikCube.cubes[0].Colors[5] = warna[myRubik.getcolor(4, 2, 0)];
rubikCube.cubes[1].Colors[4] = warna[myRubik.getcolor(3, 2, 1)];
rubikCube.cubes[1].Colors[5] = warna[myRubik.getcolor(4, 1, 0)];
rubikCube.cubes[2].Colors[3] = warna[myRubik.getcolor(0, 2, 0)];
rubikCube.cubes[2].Colors[4] = warna[myRubik.getcolor(3, 2, 2)];
rubikCube.cubes[2].Colors[5] = warna[myRubik.getcolor(4, 0, 0)];
rubikCube.cubes[3].Colors[2] = warna[myRubik.getcolor(2, 0, 1)];
rubikCube.cubes[3].Colors[5] = warna[myRubik.getcolor(4, 2, 1)];
rubikCube.cubes[4].Colors[5] = warna[myRubik.getcolor(4, 1, 1)];
rubikCube.cubes[5].Colors[3] = warna[myRubik.getcolor(0, 2, 1)];
rubikCube.cubes[5].Colors[5] = warna[myRubik.getcolor(4, 0, 1)];
rubikCube.cubes[6].Colors[0] = warna[myRubik.getcolor(1, 2, 2)];
rubikCube.cubes[6].Colors[2] = warna[myRubik.getcolor(2, 0, 2)];
rubikCube.cubes[6].Colors[5] = warna[myRubik.getcolor(4, 2, 2)];
rubikCube.cubes[7].Colors[0] = warna[myRubik.getcolor(1, 2, 1)];
rubikCube.cubes[7].Colors[5] = warna[myRubik.getcolor(4, 1, 2)];
rubikCube.cubes[8].Colors[0] = warna[myRubik.getcolor(1, 2, 0)];
rubikCube.cubes[8].Colors[3] = warna[myRubik.getcolor(0, 2, 2)];
rubikCube.cubes[8].Colors[5] = warna[myRubik.getcolor(4, 0, 2)];
               //middle layer
rubikCube.cubes[9].Colors[2] = warna[myRubik.getcolor(2, 1, 0)];
rubikCube.cubes[9].Colors[4] = warna[myRubik.getcolor(3, 1, 0)];
rubikCube.cubes[10].Colors[4] = warna[myRubik.getcolor(3, 1, 1)];
rubikCube.cubes[11].Colors[3] = warna[myRubik.getcolor(0, 1, 0)];
rubikCube.cubes[11].Colors[4] = warna[myRubik.getcolor(3, 1, 2)];
rubikCube.cubes[12].Colors[2] = warna[myRubik.getcolor(2, 1, 1)];
rubikCube.cubes[14].Colors[3] = warna[myRubik.getcolor(0, 1, 1)];
rubikCube.cubes[15].Colors[0] = warna[myRubik.getcolor(1, 1, 2)];
rubikCube.cubes[15].Colors[2] = warna[myRubik.getcolor(2, 1, 2)];
rubikCube.cubes[16].Colors[0] = warna[myRubik.getcolor(1, 1, 1)];
rubikCube.cubes[17].Colors[0] = warna[myRubik.getcolor(1, 1, 0)];
rubikCube.cubes[17].Colors[3] = warna[myRubik.getcolor(0, 1, 2)];
```

```
                //upper layer
    rubikCube.cubes[18].Colors[1] = warna[myRubik.getcolor(5,
0, 0)];
    rubikCube.cubes[18].Colors[2] = warna[myRubik.getcolor(2,
2, 0)];
    rubikCube.cubes[18].Colors[4] = warna[myRubik.getcolor(3,
0, 0)];
    rubikCube.cubes[19].Colors[1] = warna[myRubik.getcolor(5,
1, 0)];
    rubikCube.cubes[19].Colors[4] = warna[myRubik.getcolor(3,
0, 1)];
    rubikCube.cubes[20].Colors[1] = warna[myRubik.getcolor(5,
2, 0)];
    rubikCube.cubes[20].Colors[3] = warna[myRubik.getcolor(0,
0, 0)];
    rubikCube.cubes[20].Colors[4] = warna[myRubik.getcolor(3,
0, 2)];
    rubikCube.cubes[21].Colors[1] = warna[myRubik.getcolor(5,
0, 1)];
    rubikCube.cubes[21].Colors[2] = warna[myRubik.getcolor(2,
2, 1)];
    rubikCube.cubes[22].Colors[1] = warna[myRubik.getcolor(5,
1, 1)];
    rubikCube.cubes[23].Colors[1] = warna[myRubik.getcolor(5,
2, 1)];
    rubikCube.cubes[23].Colors[3] = warna[myRubik.getcolor(0,
0, 1)];
    rubikCube.cubes[24].Colors[0] = warna[myRubik.getcolor(1,
0, 2)];
    rubikCube.cubes[24].Colors[1] = warna[myRubik.getcolor(5,
0, 2)];
    rubikCube.cubes[24].Colors[2] = warna[myRubik.getcolor(2,
2, 2)];
    rubikCube.cubes[25].Colors[0] = warna[myRubik.getcolor(1,
0, 1)];
    rubikCube.cubes[25].Colors[1] = warna[myRubik.getcolor(5,
1, 2)];
    rubikCube.cubes[26].Colors[0] = warna[myRubik.getcolor(1,
0, 0)];
    rubikCube.cubes[26].Colors[1] = warna[myRubik.getcolor(5,
2, 2)];
    rubikCube.cubes[26].Colors[3] = warna[myRubik.getcolor(0,
0, 2)];
            }
        }
```

Listing Code lanjutan procedure refreshCube

BAB IV
UJI COBA DAN ANALISA

Setelah melalui tahapan perancangan, maka langkah selanjutnya adalah mengimplementasikan rancangan sistem ke dalam program dengan menggunakan Microsoft Visual Studio 2008 dan Microsoft Office Access 2003. Hal ini bertujuan untuk menilai apakah rancangan yang telah dibuat sudah sesuai dengan yang diharapkan.

Pengujian dalam tugas akhir ini meliputi pengujian aplikasi virtual Rubik's Cube dengan mata tertutup yaitu pengujian menu challenge, menu tutorial dan Cube timer.

Setelah dilakukan pengujian terhadap perangkat lunak yang di telah dibuat, kemudian akan dilakukan analisa terhadap hasil yang didapatkan. Hal ini bertujuan untuk mengetahui apakah metode yang digunakan sudah sesuai dengan hasil yang diharapkan.

4.1 PENGUJIAN PROGRAM

Pengujian program diperlukan untuk menunjukkan apakah aplikasi yang telah dibuat sesuai dengan apa yang diharapkan. Dalam pengujian program ini meliputi 2 hal yaitu pengujian Menu dan pengujian fungsi Tombol. Penjelasan untuk keduanya adalah sebagai berikut:

4.2.1. Pengujian Menu

Pengujian menu dperlukan untuk menguji apakah menu –menu yang dibuat berjalan sesuai dengan fungsinya. Pada aplikasi ini memiliki beberapa menu diantaranya adalah menu Utama dan Sub menu yang ada di dalamnya.

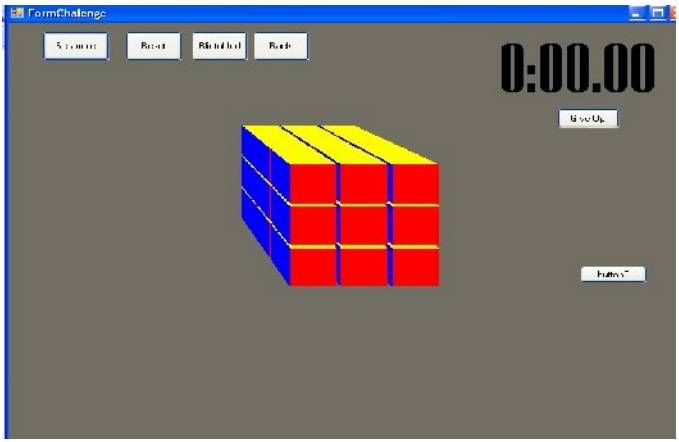
4.2.1.1. Menu Utama

Menu utama adalah menu yang pertama kali muncul ketika program dijalankan. Pada Menu ini terdapat 3 Tombol untuk menuju Sub menu lainnya yaitu sub menu Challenge, sub menu Tutorial dan sub menu Cube Timer. Berikut ini adalah screen shoot dari menu utama.

Gambar 4.1 Menu Utama

4.2.1.2. Sub Menu Challenge

Sub menu ini dibuat untuk menantang user menyelesaikan virtual Rubiks dengan mata tertutup. Jika ditengah solving user menyerah atau tidak dapat melnjutkan maka bias menekan tombol give up untuk menuju form berikutnya yang akan menjelaskan bagaimana menyelesaikan virtual cube tersebut. Berikut ini adalah screen shoot untuk sub menu Challenge.



Gambar 4.2 Sub menu Challenge

4.2.1.3. Sub Menu Tutorial

Sub menu tutorial adalah sub menu yang berfungi untuk media pembelajaran user yang ingin mengetahui bagaimana cara menyelesaikan Rubiks dengan mata tertutup menggunakan metode Old Pochmann. Didalam sub menu ini terdapat 6 tombol untuk menuju ke form lainnya yaitu tombol untuk menuju form notasi, menembak 1 target, mengenal setup move, membuat siklus, tabel algoritma Old Pochmann dan member nama piece. Berikut ini adalah screen shoot untuk sub menu tutorial.

Gambar 4.3 sub menu Tutorial

Berikut ini adalah penjelasan dari masing – masing form yang ada pada sub menu tutorial

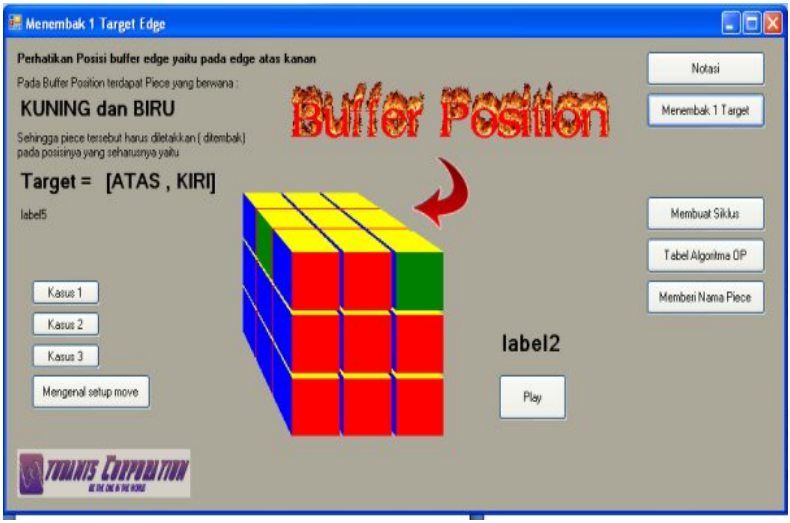
4.2.1.2.1. Notasi

Form notasi adalah form yang menunjukkan notasi yang standart digunakan dalam menyelesaikan Rubiks. Berikut ini adalah screen shoot dari form notasi.

Gambar 4.4 Form notasi

4.2.1.3.2. Menembak satu target

Form menembak satu target adalah form yang menunjukkan konsep dasar dalam menyelesaikan Rubiks dengan metode Old Pochmann. Dalam form ini terdapat kasus – kasus sederhana yang harus diketahui user untuk memahami metode Old Pochmann. Kasus – kasus tersebut sangat penting karena merupakan kasus yang paling dasar (*basic case*). Berikut ini adalah screen shoot untuk form menembak satu target.



Gambar 4.5 menembak satu target

4.2.1.3.3. Mengenal setup move

Setup move adalah konsep tingkat lanjut yang wajib dipahami dalam menyelesaikan Rubiks dengan mata tertutup. Pada form ini juga akan di tunjukkan beberapa kasus penting. Tidak semua kasus di jelaskan namun kasus yang di tampilkan dalam kasus ini adalah kasus yang sering muncul dan beberapa orang sedikit sulit untuk secara intuitif membuat gerakan setup dan undo setup- nya. Sehingga dengan melihat kasus tersebut user diharapkan dapat menyelesaikan kasus serupa. Berikut ini adalah screen shoot dari form mengenal setup move.

Gambar 4.6 mengenal setup move

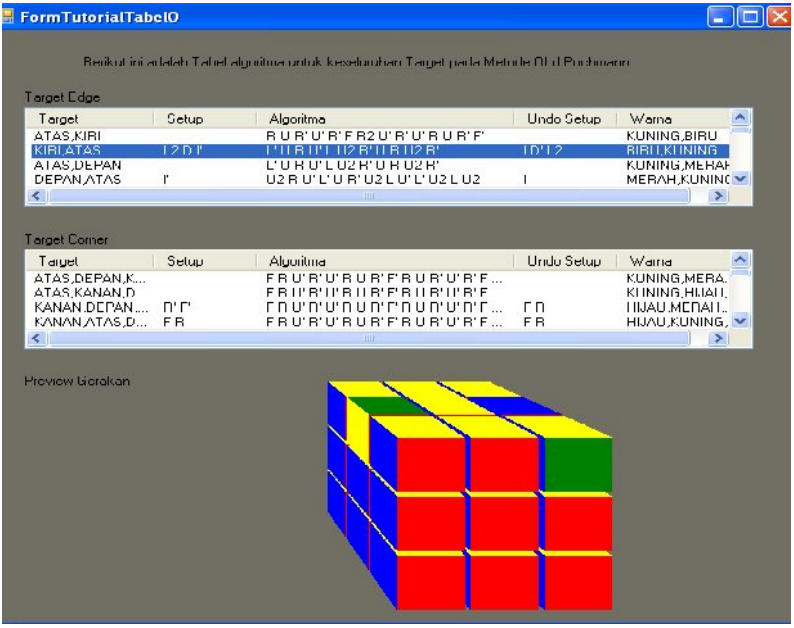
4.2.1.3.4. Membuat siklus

Siklus adalah urutan target uang harus diselesaikan oleh user. Konsep membuat siklus ini juga sangat penting karena ini adalah proses dimana kita menghafal kombinasi warna sebelum menutup mata. Dalam form ini akan dijelaskan siklus sederhana yaitu siklus yang tidak perlu melakukan breaking into new cycle yaitu kondisi dimana target yang kita dapatkan adalah buffer piece sementara buffer piece tidak boleh solve sebelum semua piece yang lain solve. Selain itu juga akan ditampilkan siklus yang memerlukan breaking into new cycle. Berikut ini adalah screen shoot untuk form membuat siklus.

Gambar 4.7 Form membuat siklus

4.2.1.3.5. Tabel algoritma Old Pochmann

Form tabel algoritma Old Pochmann adalah form yang berisi semua algoritma yang diperlukan dalam menyelesaikan Rubiks dengan metode ini. Dalam tabel tersebut dijelaskan lengkap mulai dari nama target, warna target dan algoritma setup undo setup yang digunakan. Selain menampilkan semua agoritma user juga dapat langsung melihat gerakan yang dilakukan pada algoritma tersebut dengan cara meng – klik salah satu algoritma kemudian secara otomatis virtual Rubiks yang telah disediakan akan menerjemahkan algoritma tersebut kedalam gerakan Rubiks 3 dimensi. Berikut ini adalah screen shoot untuk form tabel algoritma Old Pochmann.



Gambar 4.8 Form tabel algoritma Old Pochmann

4.2.1.3.6. Memberi nama piece

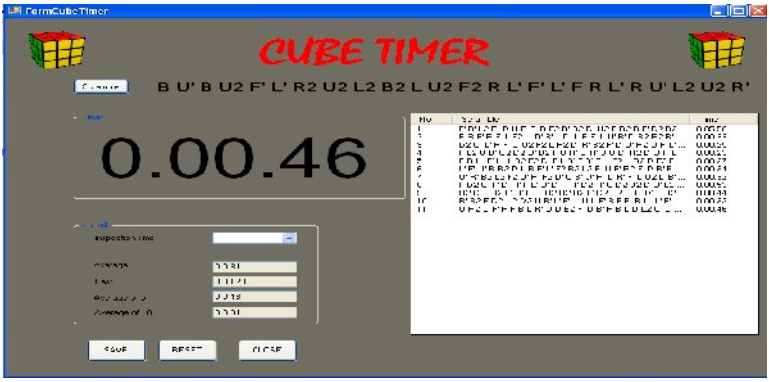
Member nama piece adalah tahap untuk mempermudah dalam menghafal kombinasi warna pada Rubiks. Pada form ini akan dijelaskan bagaimana memberi nama piece yang benar

sehingga dapat membantu dalam mengingat siklus yang harus diselesaikan oleh user. Berikut ini adalah screen shoot untuk form member nama piece.

Gambar 4.9 Form member nama piece

4.2.1.4. Sub Menu Cube Timer

Sub menu ini adalah sub menu yang digunakan sebagai timer untuk menyelesaikan Rubiks dengan scramble yang sah dan sesuai dengan standart WCA (World Cube Accosiation). Selain user dapat menghitung berapa catatan waktunya untuk menyelesaikan Rubiks, user juga dapat mengetahui berapa average untuk 5 kali solve, 10 kali solve dan berapa catatan waktu terbaiknya selama menyelesaikan Rubiks pada aplikasi ini. Berikut ini adalah screen shoot untu sub menu Cube Timer.



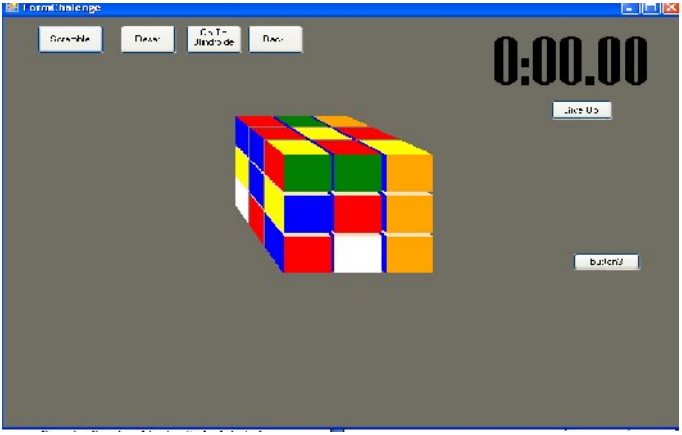
Gambar 4.10 Sub Menu Cube Timer

4.2.2. Pengujian Fungsi Tombol

Pengujian ini diperlukan untuk menguji apakah tombol yang telah dibuat fungsinya sudah sesuai dengan apa yang diharapkan. Berberapa tombol yang akan diuji diantaranya adalah tombol scramble, show keyboard, give up, solve edge, solve corner, tombol notasi, reset dan tombol pada cube timer. Berikut ini adalah penjelasan untuk masing-masing tombol tersebut.

4.2.2.1 Scramble

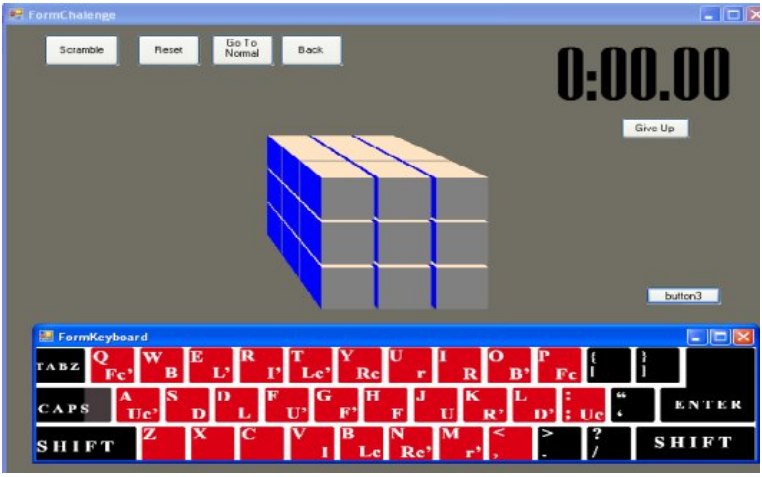
Tombol scramble adalah tombol yang digunakan untuk mengacak virtual Rubiks pada sub menu Challenge. Tombol ini akan mengaplikasikan scramble yang sesuai dengan standart WCA (World Cube Accosiation) dan sekaligus menampilkan gerakan rubik 3D sesuai dengan scramblenya. Berikut ini adalah screen shoot jika tombol scramble ditekan.



Gambar 4.11 Tombol Scramble

4.2.2.2 Show keyboard

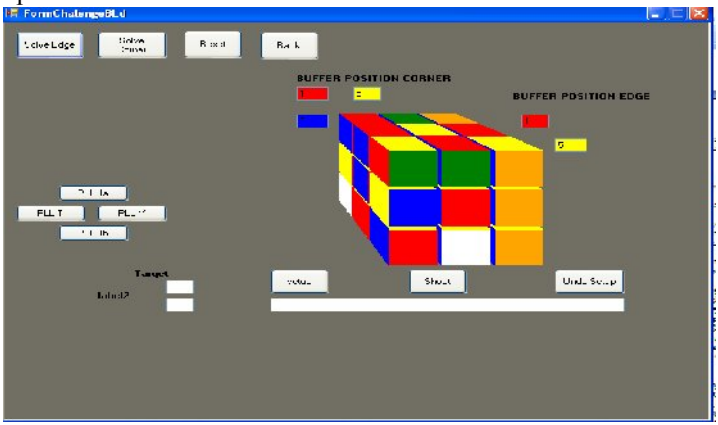
Tombol show keyboarh adalah tombol yang digunakan untuk menampilkan fungsi keyboard untuk menggerakkan Rubiks. Dalam tampilan keyboard tersebut terdapat notasi untuk menggerakkan setiap layer pada virtual Rubiks sehingga user dapat menggerakkan Rubiks sesuai dengan keiinginannya dengan melihat tampilan keyboard tersebut. Berikut ini adalah screen shoot jika tombol show keyboard ditekan.



Gambar 4.12Tombol Show Keyboard

4.2.2.3 Give Up

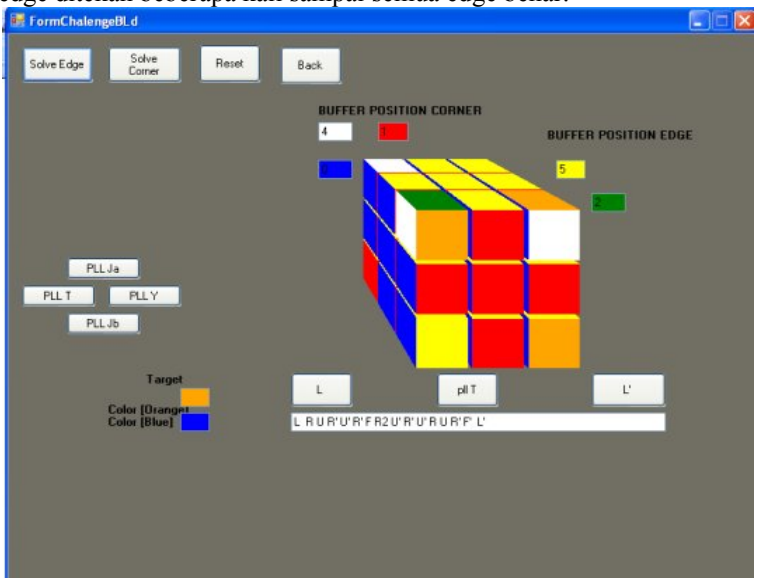
Tombol give up adalah tombol yang berfungsi jika user tidak dapat melanjutkan solving atau dalam kata user user menyerah. Tombol ini akan menuju pada form yang berfungsi untuk menunjukkan bagaimana menyelesaikan Rubiks tersebut dengan metode mata tertutup (metode Old Pochmann). Berikut ini adalah screen shoot jika tombol give up ditekan.



Gambar 4.13Tombol give up

4.2.2.4 Solve Edge

Tombol solve edge adalah tombol yang berfungsi untuk menunjukkan kepada user bagaimana menyelesaikan edge satu persatu (menempatkan edge pada posisinya yang seharusnya). User harus menekan tombol ini berberapa kali (10 sampai 15 tekanan) sampai semua edge berada pada posisinya yang bernar. Setelah menekan tombol ini user akan melihat gerakan yang seharusnya pada virtual Rubiks selain itu gerakan juga akan ditampilkan pada texbox yang ada pada bagian bawah form. Berikut ini adala screen shoot jika tombol solve edge ditekan beberapa kali sampai semua edge benar.

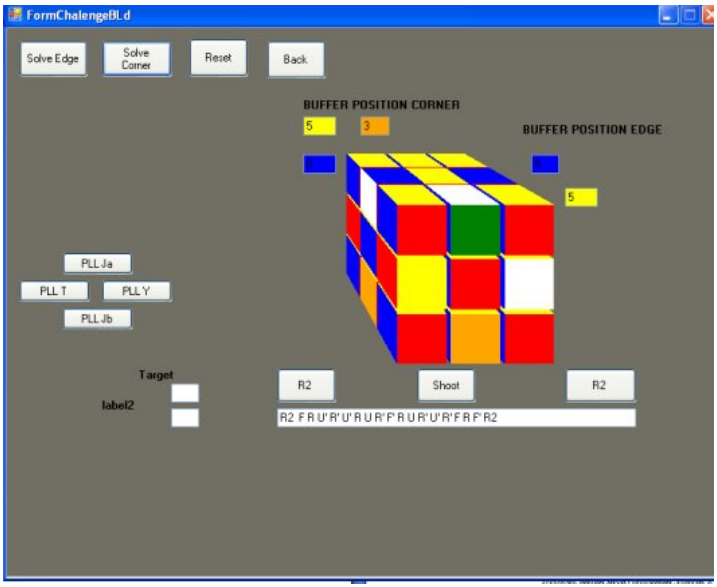


Gambar 4.14 Solve Edge

4.2.2.5 Solve Corner

Tombol solve corner adalah tombol yang berfungsi untuk menunjukkan kepada user bagaimana menyelesaikan corner satu persatu (menempatkan corner pada posisinya yang seharusnya). User harus menekan tombol ini berberapa kali (7 sampai 10 tekanan) sampai semua corner berada pada posisinya yang bernar. Setelah menekan tombol ini user akan melihat gerakan yang seharusnya pada virtual Rubiks selain itu gerakan juga akan ditampilkan pada texbox yang ada pada bagian

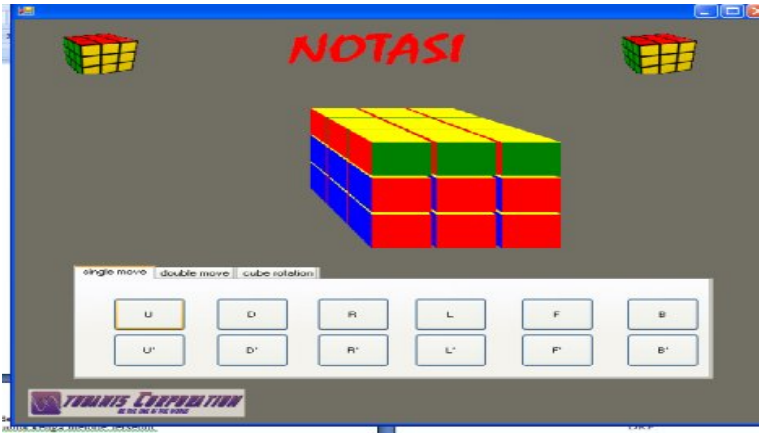
bawah form. Berikut ini adala screen shoot jika tombol solve corner ditekan beberapa kali sampai semua edge benar.



Gambar 4.15Tombol Solve corner

4.2.2.6 Notasi

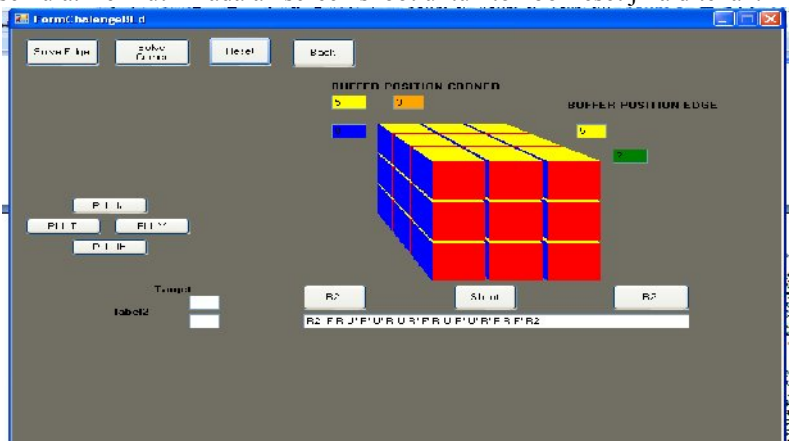
Tombol notasi adalah tombol yang berada pada form notasi pada sub menu tutorial. Tombol ini berfungsi untuk menunjukkan kepada user gerakan yang seharusnya dilakukan pada Rubiks jika menjumpai notasi seperti yang ditekan. Missal jika user menekan tombol notasi U maka virtual Rubiks akan menunjukkan bagaimana maksud dari gerakan U tersebut. Untuk lebih jelasnya berikut ini adalah screen shoot dari tombol pada form notasi.



Gambar 4.16 Form Notasi

4.2.2.7 Reset

Tombol reset adalah tombol yang diperlukan untuk memulai kembali rubiks yang sudah teracak sehingga jika tombol ini di tekan maka semua warna pada virtual rubiks akan kembali tertata seperti semula. Berikut ini adalah screen shoot untuk tombol reset jika ditekan.



Gambar 4.17 tombol reset

4.2.2.8 Tombol pada Cube Timer

Tombol pada cube timer adalah satu tombol yang berfungsi untuk menampilkan scramble yang benar dan sekaligus memulai atau

mengentikan timer untuk untuk melihat berapa lama waktu yang didapat user untuk menyelesaikan Rubiks dengan scramble atau acakan yang telah ditampilkn. Screen shoot untuk tombol pada cube timer seperti pada gambar 4.3.

Untuk pengujian menu dapat dikatakan bahwa semua menu yang ada pada aplikasi ini telah berjalan sesuai dengan harapan walaupun masih ada beberapa kekurangan yang harus diperbaiki untuk kedepannya.

4.2 ANALISA METODE PENYELESAIAN RUBIK

Pada analisa metode penyelesaian Rubik;s terdapat dua anlalisa yaitu pengujian dan analisa aplikasi virtual Rubik's menggunakan metode yang tepat dan pengujian dan analisa aplikasi virtual Rubik's sudah benar dalam menerapkan metode yang dipilih. Berikut ini adalah penjelasan terhadap dua analisa tersebut.

4.2.1 Pengujian dan analisa aplikasi virtual Rubik's menggunakan metode yang tepat.

Untuk menguji aplikasi ini menggunakan metode yang tepat yaitu metode yang memiliki jumlah gerakan dan variasi algoritma yang sedikit, aplikasi ini telah dicoba dengan 10 kasus yang akan dibandingkan dengan metode M2R2 yang terdapat pada buku "Rubik Gedhe Siapa Takut" yang ditulis oleh wicaksono adi. Berikut ini adalah 10 kasus yang digunakan untuk mencoba aplikasi ini.

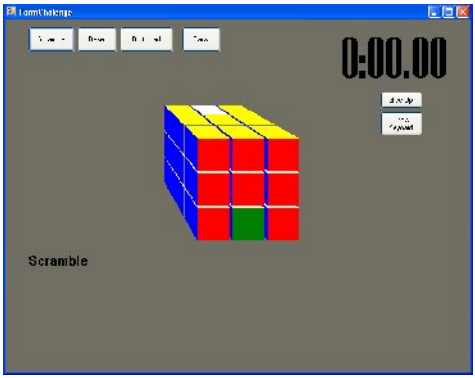
Tabel 4.1 Perbandingan algoritma Old Pochmann dan M2R2 [1]

No	Scramble	Old Pochmann	M2R2	Perbandingan gerakan
1	U2R2B2L2 DL2B2R2U'	(PLL Ja) l2 (PLL Ja) l2	R'URU'M2U R'U'RM2	22 : 10
2	UR2B2L2D' L2B2R2U2	l2 (PLL Ja) l2 (PLL Ja)	M2R'URU'M 2UR'U'R	22 : 10
3	R'UF U'RL'BU'B 'L	(PLL Jb) l' (PLL Jb) l'	B'RBM2B'R' BM2	22 : 8

4	UBR'LU'F' URL'B'	l' (PLL Jb) l' (PLL Jb)	M2B'RBM2B' R'B	22 : 8
5	U2R2F2L2D 'L2F2R2U	(PLL Jb) l2 (PLL Jb) l2	**R'URU'M2U R'U'R** M'F2M R'URU'M2U R'U'R	**22 :22**
6	U'R2F2L2D L2F2R2U2	l2 (PLL Jb) l2 (PLL Jb)	R'URU'M2U R'U'R F2M'F2M R'URU'M2U R'U'R	22:22
7	R2B'R'BR2 L2F'RFL2	l' (PLL Jb) l2 (PLL Jb) l'	M2 B'RBM2B'R' B	23 :8
8	F'R'FR2L2 B'RBR2L2	l (PLL Jb) l2 (PLL Jb) l'	B'RBM2B'R' B M2	23 :8
9	R'LDR'D'R L'FRF'	l' (PLL Ja) l2 (PLL Ja) L2	B'RBM2B'R' B M2FRUR' ERU'R'E'F'	23 : 18
10	RL'FR2F'R' LDR2D'	l2 (PLL Ja) l' (PLL Ja) l'	FERUR'E'RU 'R'F'M2	23 : 12

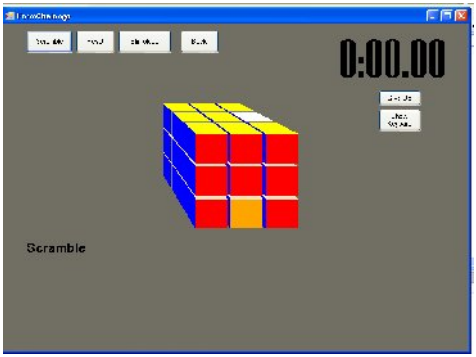
Ket:
PLL Ja= `F2L'U'LF2R'DR'D'R2 :10 moves`
PLL Jb = `B2LUL'B2RD'RDR2 :10 moves`

Berikut ini adalah gambar pada pengujian kasus 1 samapai dengan kasus 10.



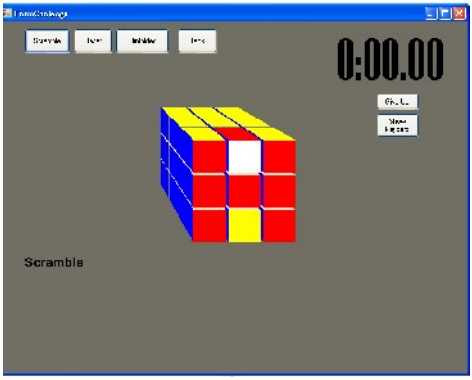
Gambar 4.18 Pengujian Kasus 1 U2R2B2L2DL2B2R2U'

Pada kasus 1, urutan target yang harus diselesaikan menggunakan Old Pochmann adalah UB-DF-UR sedangkan pada M2R2 UR-UB-DF.



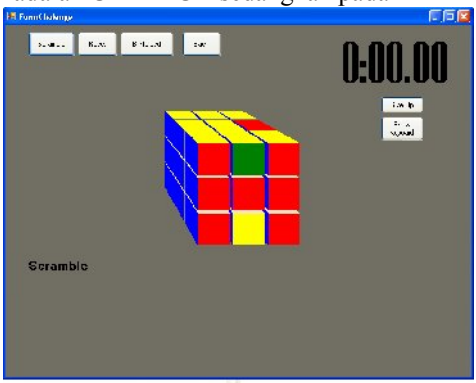
Gambar 4.19 Pengujian Kasus 2

Pada kasus 2, urutan target yang harus diselesaikan menggunakan Old Pochmann adalah DF-UB-UR sedangkan pada M2R2 UB-UR-DF.



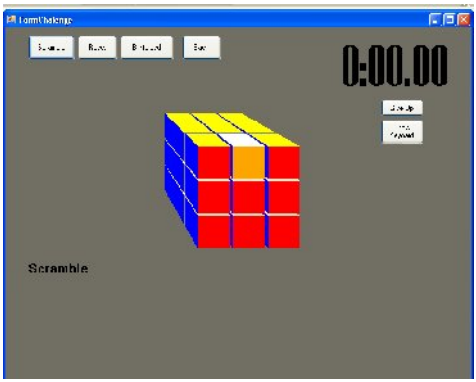
Gambar 4.20 Pengujian Kasus 3

Pada kasus 3, urutan target yang harus diselesaikan menggunakan Old Pochmann adalah UF-FD-UR sedangkan pada M2R2 RU-FU-DF.



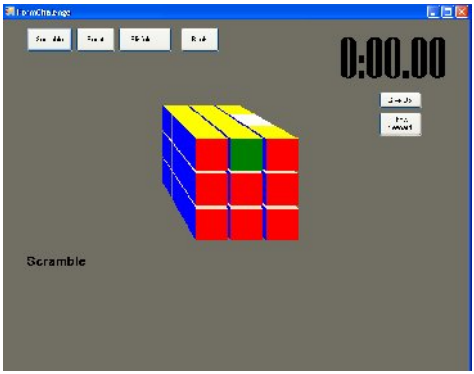
Gambar 4.21 Pengujian Kasus 4

Pada kasus 4, urutan target yang harus diselesaikan menggunakan Old Pochmann adalah FD-UF-UR sedangkan pada M2R2 FU-RU-DF.



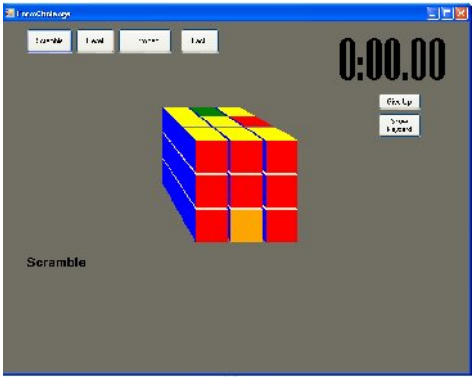
Gambar 4.22 Pengujian Kasus 5

Pada kasus 5, urutan target yang harus diselesaikan menggunakan Old Pochmann adalah UF-DB-UR sedangkan pada M2R2 UR-UF-DB-UR.



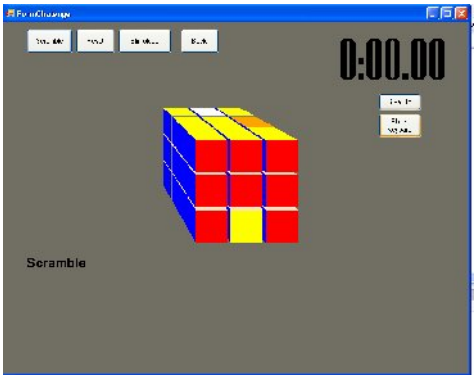
Gambar 4.23 Pengujian Kasus 6

Pada kasus 6, urutan target yang harus diselesaikan menggunakan Old Pochmann adalah DB-UF-UR sedangkan pada M2R2 UR-DB-UF-UR.



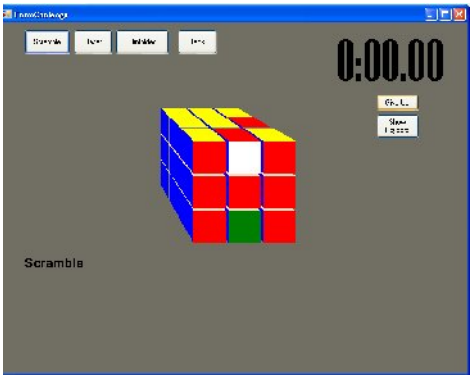
Gambar 4.24 Pengujian Kasus 7

Pada kasus 7, urutan target yang harus diselesaikan menggunakan Old Pochmann adalah FD-BU-UR sedangkan pada M2R2 UB-RU-DF.



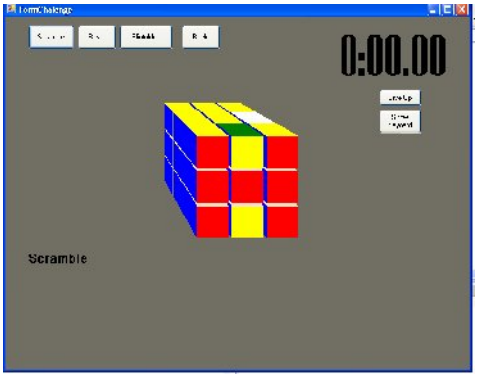
Gambar 4.25 Pengujian Kasus 8

Pada kasus 8, urutan target yang harus diselesaikan menggunakan Old Pochmann adalah BU-FD-UR sedangkan pada M2R2 RU-UB-DF.



Gambar 4.26 Pengujian Kasus 9

Pada kasus 9, urutan target yang harus diselesaikan menggunakan Old Pochmann adalah FU-DF-UR sedangkan pada M2R2 UR-FU-DF.

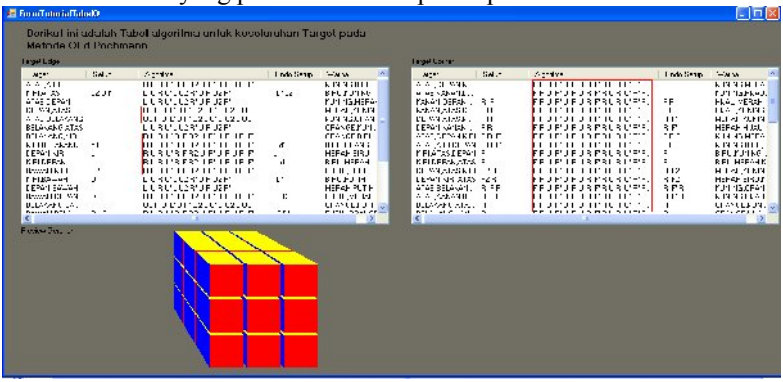


Gambar 4.27 Pengujian Kasus 10

Pada kasus 10, urutan target yang harus diselesaikan menggunakan Old Pochmann adalah DF-FU-UR sedangkan pada M2R2 FU-UR-DF.

Dari analisa diatas dapat diketahui masing – masing algoritma memiliki kelebihan dan kekurangan. Secara umum algoritma M2R2 memiliki jumlah gerakan yang lebih sedikit jika dibandingkan dengan Old Pochmann namun dari segi variasi algoritma Old Pochmann lebih unggul karena hanya menggunakan 5 variasi aslgoritma saja yang perlu dihafalkan yaitu PLL (Ja,Jb,T,Y,R) sementara M2R2 memiliki 26

variasi gerakan yang harus dihafalkan. Jadi pada aplikasi ini algoritma Old Pochmann lebih cocok digunakan karena kemudahannya dari segi variasi gerakan yang lebih sedikit. Berikut ini adalah gambar variasi algoritma Old Pochmann yang perlu dihafalkan pada aplikasi ini.



Gambar 4.28 Tabel Algoritma Old Pochmann

4.2.2 Pengujian dan analisa aplikasi virtual Rubik's dapat menerapkan metode yang dipilih.

Untuk menguji apakah aplikasi ini telah menerapakan metode yang dipilih, aplikasi ini dicoba kepada 10 orang diantaranya 5 orang yang telah menguasai metode ini dan 5 orang yang belum menguasai metode ini. Dari 10 orang tersebut semua dapat menyelesaikan virtual Rubik's dengan jumlah gerakan antara 250-350 gerakan. Berikut ini adalah data hasil pengujian tersebut.

Tabel 4.2 Hasil Survey terhadap 10 orang.

Nama	Jumlah Gerakan	WCA ID
Nanda Bhayu *	308	2010HARI01
Adam Rotal Y *	298	2011YULI01
Ilham Fikriya D *	318	2011DAR01
Stefanus Anggara *	320	2011ANGG03
Yudanis *	315	2011ROHM01
Pamana Yoga	303	
Fajar ade	331	
Ahmad Jaelani	279	
Agnes Maxelino	269	
M. Doni Setiawan	350	

Ket :

Tanda (*) adalah cuber yang sudah bisa menyelesaikan Rubiks dengan mata tertutup sedangkan yang tidak bertanda (*) adalah cuber yang belum bisa menyelesaikan Rubiks dengan mata tertutup.

WCA ID adalah World Cube Accosiation ID yang menunjukkan catatan waktu pada kompetisi resmi atau official.

4.3 ANALISA PROGRAM

Pada setiap game atau aplikasi memiliki beberapa aspek yang perlu diperhatikan yaitu:

4.3.1 User Interface

User interface dalam aplikasi ini sudah tepat sesuai dengan fungsinya masing – masing. Namun tidak ada background yang menarik, hal ini dikarenakan animasi virtual Rubiks ini menggunakan sistem double buffer yang memiliki kelemahan jika ditambahkan background makan akan berpengaruh pada pergerakan virtual Rubiks menjadi semakin lambat. Sehingga dikhawatirkan dapat membuat user merasa tidak nyaman.

4.3.2 Mekanisme Aplikasi

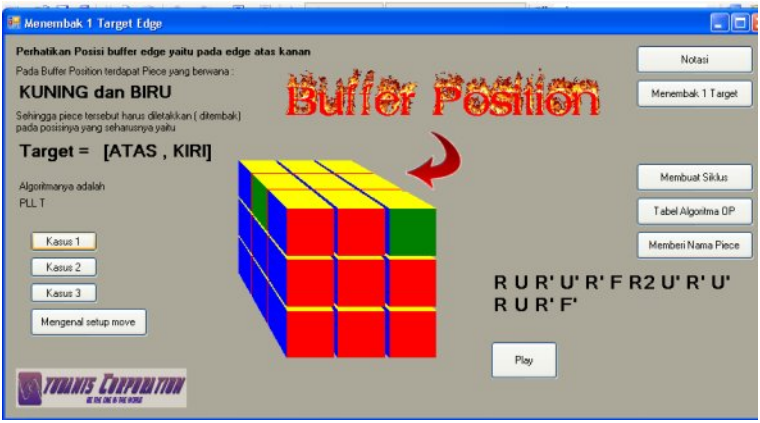
Mekanisme dalam aplikasi ini secara umum telah sesuai dengan yang diharapkan. Tidak adanya penunjuk waktu pada sub menu challenge yang menunjukkan waktu yang dicapai oleh user untuk menyelesaikan Rubiks dikarenakan sistem double buffer yang dipakai juga memiliki kelemahan hanya dapat melakukan satu proses sehingga jika letakkan timer pada virtual Rubiks maka waktu yang ditunjukkan tidak akurat. Namun hal ini telah di atasi dengan menambahkan sub menu Cube timer yang ditujukkan khusus untuk menghitung catatan waktu user dalam menyelesaikan Rubiks.

4.3.3 Analisa aplikasi memiliki Pembelajaran Menyelesaikan Rubiks dengan Mata Tertutup

Untuk menguji aplikasi ini memiliki pembelajaran atau tutorial, aplikasi memiliki sub menu tutorial didalam sub menu tersebut terdapat penjelesan lengkap mengenai metode Old Pochmann untuk menyelesaikan Rubiks dengan mata tertutup. Berikut ini adalah tampilan dalam sub menu tutorial:

Gambar 4.29 Sub menu Tutorial

Selain tutorial aplikasi ini juga menyediakan contoh kasus dasar yang harus diketahui oleh user yang ingin bisa menyelesaikan Rubiks dengan mata tertutup. Berikut ini adalah contoh kasus yang diberikan pada aplikasi ini.



Gambar 4.30 Contoh Kasus 1

Contoh kasus 1 adalah kasus dasar yang harus diketahui oleh user yaitu kasus menembak 1 target pada layer atas.

Gambar 4.31 Contoh kasus 2

Contoh kasus 2 juga merupakan kasus dasar yang harus diketahui oleh user yaitu kasus mengenal setup move untuk menembak target yang berada di selain layer atas (layer bawah dan layer tengah). Kasus ini cukup penting karena lebih dari 80% edge target berada pada layer bawah dan layer tengah.

Gambar 4.32 Contoh kasus 3

Contoh kasus 3 adalah kasus dasar yang sangat penting harus diketahui oleh user yaitu bagaimana membuat siklus yang benar dan kasus *breaking into new cycle* yaitu kondisi dimana buffer piece

(kuning, hijau) telah pada posisinya yang seharusnya namun edge yang lain belum berada pada posisi yang benar.

Pembelajaran atau tutorial pada aplikasi ini sudah cukup lengkap namun masing memiiki beberapa kekurangan yaitu tidak ada background yang menarik. Tetapi secara umum bisa dikatakan cukup untuk menjelaskan metode Old Pochmann dalam menyelesaikan Rubiks dengan mata tertutup.

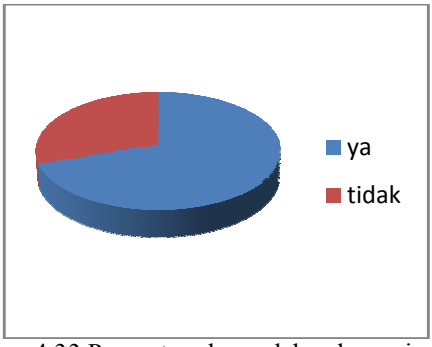
4.4 ANALISA QUESIONER

Selain diuji dan di analisa, aplikasi ini juga dinilai melalui survey melalui media quesioner kepada komunitas Rubik's cube yang ada di kota Kediri untuk mengetahui penilaian orang lain (sudah memahami konsep dasar menyelesaikan Rubik's cube) terhadap aplikasi ini. Berikut ini adalah aspek – aspek yang dinilai dalam aplikasi ini:

4.4.1 Kemudahan dalam bermain game

Kemudahan dalam bermain merupakan salah satu aspek yang perlu diperhatikan dalam setiap permainan atau game. Berikut ini adalah hasil penilaian tentang kemudahan dalam bermain game.

1. Ya = 6 Suara
2. Tidak = 4 Suara

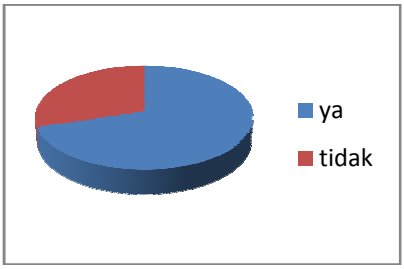


Gambar 4.33 Prosentase kemudahan bermain game

4.4.2 Kesesuaian algoritma yang ditawarkan

Kesesuaian algortima yang ditawarkan juga merupakan aspek penting khususnya dalam bermain Rubik's. Berikut ini adalah hasil penilaian tentang kesesuaian algoritma yang ditawarkan.

1. Ya = 7 Suara
2. Tidak = 3 Suara

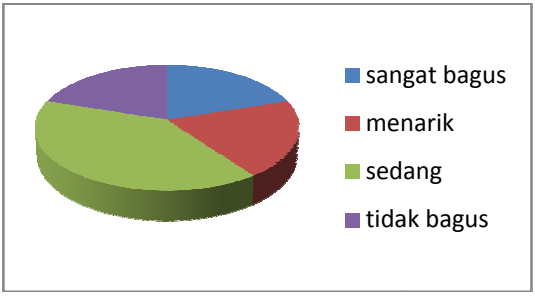


Gambar 4.34 prosentase kesesuaian algoritma yang ditawarkan

4.4.3 Tampilan grafik 3D

User interface merupakan media yang paling efektif untuk membuat user atau pemain merasa nyaman dan tidak bosan dalam memainkan game ini. Berikut ini adalah hasil penilaian terhadap tampilan grafik 3D dalam aplikasi ini.

1. Sangat bagus = 2 Suara
2. Menarik = 2 Suara
3. Sedang = 4 Suara
4. Tidak bagus = 2 Suara

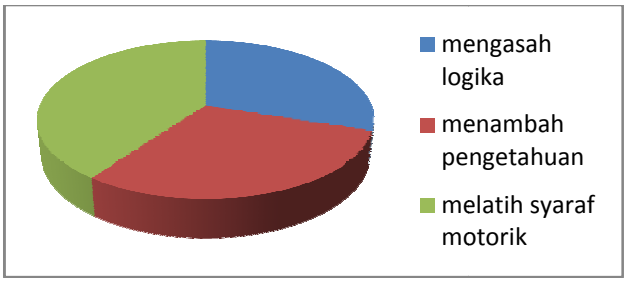


Gambar 4.35 Prosentase tampilan grafik 3D

4.4.4 Manfaat dari game.

Aspek yang paling penting dalam sebuah aplikasi adalah manfaatnya bagi user. Berikut ini adalah hasil penilaian pada aspek manfaat pada aplikasi ini.

1. Mengasah Logika = 3 Suara
2. Menambah Pengetahuan = 3 Suara
3. Melatih Syaraf Motorik = 4 Suara

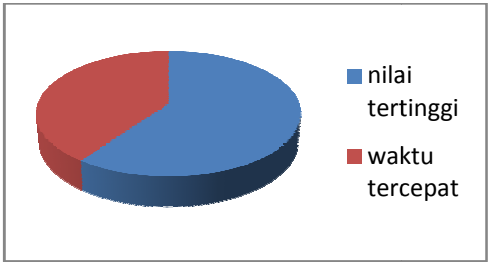


Gambar 4.36 Prosentase manfaat bermain game

4.4.5 Tantangan dalam game

Tantangan adalah aspek yang cukup penting untuk melatih user dalam menyelesaikan Rubiks cube dengan mata tertutup. Berikut ini adalah penilaian terhadap tantangan pada aplikasi ini.

1. Nilai tertinggi = 6 Suara
2. Waktu Tercepat = 4 Suara

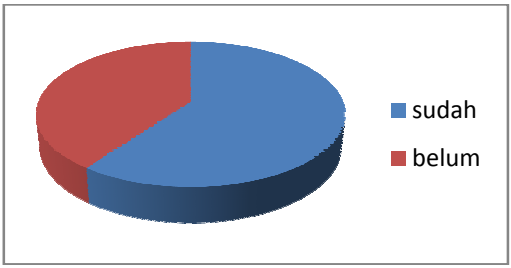


Gambar 4.37 Prosentase tantangan bermain game

4.4.6 Kelangkapan tutoraial atau media pembelajaran

Kelengkapan tutorial adalah aspek yang menunjang dalam pemahaman user untuk menyelesaikan Rubiks dengan mata tertutup sekaligus sebagai media pembelajaran. Berikut ini adalah penilaian terhadap kelengkapan tutorial pada aplikasi ini.

1. Sudah = 6 Suara
2. Belum = 4 Suara



Gambar 4.38 Prosentase kelengkapan tutorial

--Halaman ini sengaja dikosongkan--

BAB V
PENUTUP

5.1. Kesimpulan

Berdasarkan hasil perancangan dan implementasi sistem yang dilakukan pada proyek akhir ini, maka dapat diambil beberapa kesimpulan, yaitu :

1. Konsep yang telah direncanakan dapat diaplikasikan seperti perangkat lunak yang telah dibuat ini dan dapat memberikan sumbangsih bagi dunia permainan di Indonesia.
2. Aplikasi ini menggunakan metode yang tepat yaitu Old Pochmann yang memiliki variasi algoritma yang sedikit dan konsep yang mudah dimengerti oleh user pemula.
3. Algoritma Old Pochmann dapat diterapkan dengan baik dengan estimasi jumlah gerakan antara 250 sampai dengan 350 gerakan.
4. User interface dalam aplikasi ini sudah tepat sesuai dengan fungsinya masing – masing
5. Media pembelajaran pada aplikasi ini cukup lengkap dan disisipkan pula contoh – contoh kasus dasar dan sederhana untuk membantu pemahaman user terhadap metode Old Pochmann.
6. Berdasarkan hasil survey aplikasi ini mudah dijalankan, algoritma yang ditawarkan sesuai, grafik 3D yang digunakan untuk menampilkan virtual Rubiks cukup bagus, apklikasi ini juga memberikan manfaat untuk melatih syaraf motorik, tantangan yang paling menarik bagi user adalah mendapatkan nilai tertinggi, dan dari segi kelengkapan tutorial aplikasi ini sudah memiliki tutorial yang lengkap.

5.2 Saran

Berikut ini saran-saran untuk meningkatkan hasil yang telah dicapai :

1. Aplikasi sebaiknya digunakan oleh user yang telah bisa menyelesaikan Rubiks dengan mata terbuka terlebih dahulu.
2. User interface sebaiknya dibuat lebih mobile sehingga bisa digunakan dibanyak tempat.
3. Disediakan feed back kepada user seperti peringkat atau penghargaan terhadap user yang telah sukses menyelesaikan Rubiks dengan mata tertutup.

4. Selain metode Old Pochmann perlu ditambahkan metode yang lain sehingga user lebih leluasa memiih metode mana yang paling cocok untuk digunakan.

DAFTAR PUSTAKA

[1] Adi, Wicaksono. 2010. *RUBIK GEDE SIAPA TAKUT*. Gramedia mediatama. Jakarta

[2] Meriam. "sejarah Rubiks cube".http://www.meriam-sijagur.com/learning/41-history/547-sejarah-rubiks-cube.html (Diakses pada tanggal 2 Januari 2012)

[3] Pochmann, steffan. " Blindsolving the 3x3"*http://www.stefan-pochmann.info/spocc/blindsolving/3x3/old.php.* (Diakses pada tanggal 22 agustus 2011)

[4] Pochmann, steffan *"* M2/R2 blindcubing methods *"http://www.stefan-pochmann.info/spocc/blindsolving/M2R2/* (Diakses pada tanggal 22 agustus 2011)

[5] Pico, heri "How To : Menyelesaikan Rubik 3×3 dengan Mata Tertutup (Blindfold)."*http://pikopages.wordpress.com/2009/10/11/how-to-menyelesaikan-rubik-3x3-dengan-mata-tertutup-blindfold/* (Diakses pada tanggal 22 agustus 2011)

Kuesioner Evaluasi Aplikasi Game Virtual Rubik's Cube dengan Mata Tertutup

Nama :
Jenis Kelamin :
Usia :
Pertanyaan :

1. Menurut anda bagaimana *gameplay* pada aplikasi vitual Rubiks'cube ?
   a. Sangat Menarik
   b. Menarik
   c. Sedang
   d. Tidak Menarik
2. Apakah aplikasi ini mudah untuk dioperasikan ?
   a. Ya
   b. Tidak
3. Apakah kasus yang diberikan dan algoritma yang ditawarkan sudah sesuai?
   a. Ya
   b. Tidak
4. Menurut anda bagaimana tampilan grafik 3D pada aplikasi virtual Rubik's cube ?
   a. Sangat Bagus
   b. Menarik
   c. Sedang
   d. Tidak Bagus
5. Jenis game yang paling anda suka?
   a. Action
   b. Adventure
   c. Puzzle
   d. Racing
   e. Strategi
   f. RPG
   g. Sport
6. Alasan anda bermain game?
   a. Mengisi waktu luang
   b. Hobi
   c. Hiburan
   d. Lainnya (sebutkan)

   ……
7. Menurut anda manfaat bermain game?
   a. Mengasah logika
   b. Menambah pengetahuan
   c. Melatih saraf motorik
   d. Lainnya(sebutkan)

……

8. Apakah game ini mudah memberikan manfaat ?
   a. Ya b. Tidak
9. Unsur yang membuat anda tertarik bermain game?
   a. Grafik c. Music
   b. Game play d. Ketenaran
10. Tantangan yang paling menarik pada puzzle game?
   a. Nilai tertinggi c. Waktu tercepat
   b. Mendapatkan bonus d. Lainnya(sebutkan)

11. Apakah tutorial pada aplikasi ini sudah lengkap?
   a. Sudah b. belum
12. Sebutkan jumlah gerakan yang anda lakukan dalam menyelesaikan virtual Rubik's pada aplikasi ini!

……

DAFTAR RIWAYAT HIDUP

Nama	: Yudanis Taqwin Rohman
Tempat tanggal lahir	: Kediri, 24 November 1988
Alamat	: Jl. Batik Madrim 109 Kalirong-Tarokan-Kediri-Jawa Timur
Email	: yudanistaqwinrohman@yahoo.co.id
Telepon	: 085731117311
Hobi	: Rubik, Music
Cita-cita	: Menjadi orang yang bermanfaat
Motto	:Do The Best and Be The One in The world

Riwayat Pendidikan :

- SDN 1 Kalirong - Kediri 1995-2001
- SMPN 1 Ngadiluwih, 2001-2004
- SMAN 2 Kediri, 2004-2007
- D3 Manajemen Informatika Universitas Brawijaya, 2007-2010
- D4 PENS – ITS, 2010-2012