MEMBANGUN APLIKASI *GAME PUZZLE RUBIK CUBE* TIGA DIMENSI

TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana Jurusan Teknik Informatika

Oleh :

Nama : Akhmad Rafiuddin
NIM : 02 523 025

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
YOGYAKARTA
2008

LEMBAR PENGESAHAN

MEMBANGUN APLIKASI *GAME PUZZLE RUBIK CUBE* TIGA DIMENSI

TUGAS AKHIR

Nama : [illegible] Rafiuddin
NIM : [illegible] 025

Yogyakarta, Oktober 2008

Pembimbing

(Yudi Prayudi, S.Si, M.Kom)

LEMBAR PENGESAHAN PENGUJI

MEMBANGUN APLIKASI *GAME PUZZLE RUBIK CUBE* TIGA DIMENSI

TUGAS AKHIR

Oleh :

Nama : Akhmad Rafiuddin
NIM : 02 523 025

Telah Dipertahankan di Depan Sidang Penguji Sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana Tekhik Informatika
Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, Desember 2008

Tim Penguji

Irving Vitra P, S.T _____

Yudi Prayudi, S.Si, M.Kom _____

_____ _____

Mengetahui,
Ketua Program Studi Teknik Informatika
Universitas Islam Indonesia

Yudi Prayudi, S.Si, M.Kom

LEMBAR KEASLIAN TUGAS AKHIR

Yang bertandatangan dibawah ini:

Nama : Akhmad Rafiuddin

NoMhs : 02 523 025

Tugas Akhir dengan Judul:

MEMBANGUN APLIKASI *GAME PUZZLE RUBIK CUBE* TIGA DIMENSI

Dengan ini menyatakan bahwa Tugas Akhir ini merupakan hasil kerja saya pribadi, adapun ide dari beberapa algoritmanya pada pemrograman ini merupakan ide saya sendiri dan ada beberapa yang merupakan hasil inspirasi dari beberapa aplikasi yang sudah ada. Apabila dikemudian hari terbukti bahwa Tugas Akhir ini bukan merupakan hasil kerja saya maka saya bersedia menanggung resiko dan konsekuensinya.

Yogyakarta, Desember 2008

Yang Membuat Pernyataan

Akhmad Rafiuddin

Saksi-saksi

Irving Vitra P, S.T _____

Yudi Prayudi, S.Si, M.Kom _____

HALAMAN PERSEMBAHAN

Tugas Akhir ini kupersmbahkan untuk orang tuaku, dan semua yang mendukungku selama ini

HALAMAN MOTTO

Katakanlah: "Dia-lah Allah, Yang Maha Esa

Allah adalah Tuhan yang bergantung kepada-Nya segala sesuatu

Dan tiada beranak dan tiada pula diperanakkan

Dan tiada ada seorangpun yang setara dengan Dia"

(QS: Al Ikhlas) 3x

"Sesungguhnya Allah dan Malaikat-malaikatNya bershalawat untuk Nabi.

Hai orang-orang yang beriman, bershalawatlah kamu untuk Nabi

dan ucapkanlah salam penghormatan kepadanya"

(QS: Al Ahzab 56)

.... 'Dunia ini terlaknat, terlaknat apa yang ada di dalamnya kecuali dzikir kepada Allah,

taat kepada-Nya, dan orang yang berilmu serta yang mencari ilmu.'(h.r Tirmidzi)

...'Barangsiapa belajar ilmu karena Allah, namun ia mempelajarinya melainkan hanya untuk

mendapatkan bagian dari dunia, maka ia tidak akan mendapatkan aroma surga

di hari kiamat kelak.' (h.r Abu Dawud)

KATA PENGANTAR

Syukur Alhamdulillah penulis ucapkan kehadirat Allah SWT, yang dengan izin, kuasa, kemurahan dan segala ke Maha an-Nya tugas akhir ini dapat penulis selesaikan dengan baik. Sholawat dan salam juga tidak lupa kami haturkan kepada junjungan Nabi besar Muhammad SAW. Tugas akhir ini dibuat sebagai salah satu syarat yang harus dipenuhi untuk memperoleh gelar sarjana di jurusan Teknik Informatika Universitas Islam Indonesia.

Dalam pelaksanaan kerja praktek ini, penulis mendapat bantuan dari beberapa pihak. Untuk itu pada kesempatan ini penulis ingin mengucapkan terima kasih yang sebesar-besarnya kepada :

1. **Bapak dan Ibu** ku atas segala dukungan, kasih sayang, do'a nya dan yang keikhlasan dalam mengasuh dan membesarkanku selama ini hingga saat ini bisa menyelesaikan tugas akhir ku.
2. Bapak Yudi Prayudi, S.Si, M.Kom selaku pembimbing dan juga ketua jurusan Teknik Informatika UII.
3. Simbah Sol dan simbah Mah yang juga sudah berdo'a untuk segala kebaikan yang insya Allah doanya dikabulkan Allah SWT, Amin.
4. Pakde Zuhron dan Paklek Yanto beserta keluarga besar, semua yang telah membantu dalam proses perkuliahanku selama ini yang telah aku repotin sampai akhirnya aku bisa sampai tahap akhir sebagai syarat kelulusanku ini yang semoga insya Allah bisa selesai kuliah di UII terima kasih.

5. Semua saudara dan orang-orang terdekatku yang dengan kehadiran kalian aku bisa menyelesaikan TA ku ini yang insya Allah sebagai syarat kelulusanku dari bangku kuliah.
6. Semua teman-teman, yang merasa pernah tak repotin dan tak mintain bantuan terima kasih juga.

Penulis menyadari bahwa tugas akhir ini masih jauh dari kesempurnaan sebuah tugas akhir. Untuk itu segala saran dan kritik yang sifatnya membangun sangat penulis harapkan. Semoga kerja praktek ini dapat bermanfaat bagi pihak yang memerlukannya.

Yogyakarta, Oktober 2008

Penulis

SARI

Keterbatasan kemampuan manusia dalam menghadapi tekanan hidup yang tinggi terutama pada kehidupan masyarakat modern perlu diimbangi dengan berbagai macam hiburan. Salah satu hiburan yang berbasis teknologi komputer yang saat ini sedang berkembang dan banyak digemari adalah *computer game*. *Computer game* saat ini telah menjadi bagian yang tidak bisa dipisahkan dari kehidupan masyarakat modern secara umum, dimana hampir setiap harinya berinteraksi dengan komputer. *Puzzle rubik cube* adalah contoh permainan yang banyak diadaptasi menjadi sebuah *computer game.* Tugas akhir ini bertujuan mengadopsi permainan *puzzle rubik cube* menjadi sebuah *computer game* yang dapat dimainkan pada sebuah *PC* (komputer).

Tugas akhir ini menggunakan OpenGL, openGL adalah librari *open source* yang dapat digunakan dalam pemrograman grafis dua dimensi maupun tiga dimensi guna membuat aplikasi berbasis dua maupun tiga dimensi.

Sebuah *game* berbasis tiga dimensi dalam tugas akhir ini yang dibuat menggunakan pemrograman OpengL ditujukan untuk membantu para pengguna komputer yang menginginkan bermain *puzzle rubik* pada sebuah Personal Computer.

keyword : hiburan, *computer game, puzzle, rubik cube,* OpenGL, tiga dimensi

TAKARIR

about	:	tentang
artificial intelegence	:	kecerdasan buatan
auto solving	:	penyelesaian otomatis
back	:	belakang
best time	:	waktu terbaik
computer game	:	permainan komputer
cube	:	kubus
difficult	:	sulit
down	:	bawah
exit	:	keluar
front	:	depan
hard	:	sulit
help	:	bantuan
keyboard	:	papan tuts
left	:	kiri
layer	:	lapisan
mechanical puzzle	:	puzzle mekanik
mouse	:	tetikus (alat navigasi pada komputer)
new game	:	permainan baru
object oriented analysis	:	analisa berarah obyek
object oriented design	:	desain berarah obyek
options	:	pilihan-pilihan
personal computer	:	komputer pribadi
processor	:	pemroses
puzzle	:	puzzle
radio box	:	kotak radio
record time	:	waktu rekor
right	:	kanan

rubik cube	:	kubus rubik
shared	:	berbagi
software	:	perangkat lunak
sound	:	suara
speed cubing	:	bermain kubik dengan cepat
timer	:	penghitung waktu
up	:	atas
use case	:	kasus pengguna

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

1.1 Latar Belakang

Sejak pertama kali diciptakan pada era 1960-an dan hanya dimainkan oleh beberapa kalangan saja, *computer game* saat ini telah menjadi bagian yang tidak bisa dipisahkan dari kehidupan masyarakat modern secara umum, dimana hampir setiap harinya berinteraksi dengan komputer. *Computer game* memiliki efek yang dapat mengakibatkan para pemainnnya merasakan berbagai sensasi, seperti rasa penasaran, puas, kesal, dapat memicu adrenalin pemainnya bahkan hingga menyebabkan ketagihan. Hal ini merupakan salah satu penyebab banyaknya orang menjadi hobi bermain *computer game*. Alasan tersebut diatas adalah salah satu penyebab dari pesatnya perkembangan industri *computer game* saat ini.

Tercatat pada tahun 2004 industri penjualan *software computer game* dan *video game* telah mencapai angka penjualan sebesar USD 7,3 Milyar [ANO08b], dan pada penjualan tahun 2007 mencapai USD 9,5 Milyar [ANO08c]. Hal inilah yang membuktikan akan besarnya perkembangan industri *computer game* sejak awal diciptakan.

Rubik cube adalah permainan berjenis *puzzle game*, lebih tepatnya adalah jenis *mechanical puzzle* yaitu *puzzle* yang dibuat dengan menyatukan bagian-bagiannya secara mekanis. Tujuan akhir dari permainan *puzzle rubik cube* ini adalah menyamakan warna pada tiap-tiap sisinya. Jenis permainan ini membutuhkan ketekunan, ketelitian dan juga kecerdasan dalam memainkannya.

Permainan *rubik cube* dapat mengakibatkan rasa penasaran dan keasyikan tersendiri bagi pemainnya, tidak heran jika permainan ini menjadi permainan paling sukses pada periode 1980-1982.

Seperti dikatakan dalam website resminya [ANO08a] : " (1982) *The first annual International Rubik's Championships are held in Budapest. More than 100 million cubes have now been sold and Rubik's enters the Oxford English Dictionary* ". Pernyataan tersebut diatas menunjukkan bahwa pada tahun 1982 *game puzzle rubik cube* ini terjual lebih dari 100 juta buah.

Bagi para penggemar berat permainan *rubik cube* tidaklah cukup bagi mereka jika hanya menyelesaikan ke enam sisi *rubik* secara keseluruhan, mereka menambahkan unsur waktu atau kecepatan (menambah tantangan) dalam menyusun keenam sisinya, hal semacam ini sering juga disebut dengan istilah *speed cubing.*

Meskipun kejayaan permainan (*toy*) *rubik cube* pada masanya (era tahun 80an) sudah lama berlalu dan sudah berkurang ketenarannya, namun didasari dengan fakta bahwa saat ini teknologi informasi serta industri dan pemrograman *game* berkembang sangat pesat, penulis mencoba untuk membuat sebuah aplikasi yang berbasiskan teknologi informasi (*computer game*) dengan mengadaptasi permainan *rubik cube*.

1.2 Rumusan Masalah

Latar belakang yang telah dijabarkan diatas, memberikan dasar tentang rumusan masalah yang akan dibuat yaitu '*bagaimanakah membuat sebuah aplikasi game puzzle rubik cube yang tiga dimensi ?*'

1.3 Batasan Masalah

Guna mengatasi meluasnya pembahasan, Tugas Akhir ini mempunyai beberapa batasan masalah antara lain :

1. Tidak memasukkan unsur-unsur AI *(Artificial Intelegence)*, sehingga tidak ada *auto solving* maupun pergerakan-pergerakan otomatis yang dijalankan oleh sistem.
2. *Rubik cube* yang diaplikasikan *cube* berdimensi 3x3x3.
3. *Rubik cube* yang diaplikasikan memiliki enam warna sisi (merah, hijau, biru kuning, putih, oranye) yang berbeda disesuaikan dengan standard pewarnaan *puzzle rubik* pada umumnya.
4. Pemrograman OpenGL tidak memasukkan unsur pencahayaan (*lighting*), bayangan (*shadow*) serta efek-efek yang lainnya.

1.4 Tujuan Penelitian

Adapun tujuan yang ingin dicapai dari penelitian tugas akhir ini adalah membangun suatu aplikasi *game puzzle rubik cube* bagi siapapun (secara khusus bagi penggemar *puzzle rubik cube*) yang ingin bermain *puzzle rubik cube* pada sebuah *PC* (komputer).

1.5 Manfaat Penelitian

Ada beberapa manfaat yang penulis harapkan dari penelitian ini antara lain :

1. Memberikan fasilitas bagi pengguna komputer untuk dapat bermain *game puzzle rubik cube* menggunakan *PC* (komputer pribadi) nya.
2. Memberi alternatif pilihan permainan bagi penggemar *computer game.*
3. Dapat menambah wawasan penulis tentang pemrograman tiga dimensi khususnya pemrograman OpenGL.

1.6 Metodologi Penelitian

Sebuah penelitian yang dibuat haruslah melalui suatu aturan perancangan yang berurutan serta memenuhi beberapa tahapan.

1.6.1 Pengumpulan Data

Pada penelitian ini, metode pengumpulan data yang digunakan adalah sebagai berikut :

a. Metode studi pustaka

   Metode studi pustaka, yaitu metode mengumpulkan data dengan mencari referensi dari berbagai buku.

b. Referensi internet

Mengumpulkan materi-materi dengan cara mengunjungi berbagai website yang berkaitan dengan tugas akhir.

1.6.2 Pengembangan Sistem

Setelah seluruh data dikumpulkan meka dilakukan tahapan-tahapan sebagai berikut guna mendapatkan perangkat lunak yang sesuai:

a. Analisa data

Mengumpulkan berbagai data tentang rubik, pemrograman Visual C++ khususnya yang menggunakan librari OpenGL.

b. Desain

Merancang dan membuat diagram UML sebagai media dalam pengimplementasian perancangan berorientasi objek. Membuat dasar-dasar tampilan antarmuka (*interface*).

c. Pengkodean

Membuat pemrograman menggunakan microsoft Visual C++ 6.0 dan menggunakan librari OpenGL.

d. Pengujian

Mengadakan pengujian dan menganalisa software seperti menguji cobakan pada beberapa *user* dan meminta kuisioner atas ujicoba tersebut.

1.7 Sistematika Penulisan

Penulisan laporan tugas akhir ini disusun kedalam **5** Bab, dengan maksud untuk mempermudah pembacaan yang lebih akurat. Garis besar isinya adalah sebagai berikut:

BAB I Pendahuluan

Pada Bab ini dibahas latar belakang masalah, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian dan sistematika penulisan.

BAB II Landasan Teori

Pada Bab ini dibahas teori-teori yang berhubungan dengan penelitian tentang pembuatan *game* komputer *puzzle rubik cube* tiga dimensi yang menggunakan pemrograman C++ dan OpenGL.

BAB III Metodologi

Pada Bab ini dibahas tentang metode analisis pembuatan perangkat lunak, serta perancangan perangkat lunak. Mencakup metode perancangan perangkat lunak yang menggunakan perancangan berorientasi objek (*Object Oriented Design*) dengan menggunakan standard UML.

BAB IV Hasil dan Pembahasan

Pada Bab ini dibahas hasil dari analisis kinerja perangkat lunak yang berisi mengenai dokumentasi kinerja, implementasi serta pengujian yang sesuai dengan kebutuhan yang diharapkan, seperti kebutuhan perangkat lunak dan perangkat keras.

BAB V Simpulan dan Saran

Pada Bab ini memuat kesimpulan yang merangkum hasil analisis kinerja pada bagian sebelumnya. Bab ini juga memuat saran dari penulis yang diperlukan guna pengembangan penelitian kedepan.

BAB II

LANDASAN TEORI

2.1 *Computer Game*

Computer game atau juga *game* komputer adalah *video game* yang dimainkan pada sebuah *Personal Computer* (PC). Menurut Mark J.P Wolf dalam buku nya "*The Medium of the Video Game*" , permainan dalam *video* dan *computer game* dapat diklasifikan menjadi 42 tipe [LIO08],[1] yaitu :

1. *Abstrac*

   Jenis *game* yang di presentasikan secara sederhana. Contoh : Tetris, Arkanoid.

2. *Adaptation*

   Jenis *game* yang diadaptasi dari aktifitas sehari-hari. Contoh: Solitaire, Black Jack (adaptasi permainan kartu), Rubik Cube.

3. *Adventure*

   Jenis *game* yang bersetting "dunia", biasanya dibuat untuk bermain secara *multiplayer*. Contoh: Tomb Raider.

4. *Artificial Life*

   Jenis game yang mengajak kita untuk dapat merawat kehidupan (pertumbuhan, kematian) suatu karakter digital. Contoh: Tamagochi, The Sims.

---

[1] Artikel yang diambil dari website ini hanya mengutip (bab 6 : *Genre and The Video Game*) dari buku Mark J.P Wolf *"The Medium of The Video Games"*.

5. *Board Games*

   Jenis *game* yang diadaptasi dari permainan papan (*board game)* atau *game* yang mirip seperti *board game.* Contoh: Monopoli, Othello.

6. *Capturing*

   Jenis *game* yang bertujuan menangkap suatu karakter yang secara aktif berusaha melarikan diri dan menghilang dari karakter pemain. Contoh: Hole Hunter, Gopher.

7. *Card Games*

   Jenis *game* yang berdasarkan dari berbagai permainan kartu yang ada. Contoh: Black Jack, Poker.

8. *Catching*

   Sama seperti *capturing* hanya karakter yang dikejar tidak secara aktif berusaha menghilang dari kejaran karakter pemain. Contoh: Big Bird's Egg Catch.

9. *Chase* (Lihat *Catching, Capturing, Driving, Escape, Flying,* dan *Racing*).

10. *Collecting*

    Jenis *game* yang bertujuan mengumpulkan benda-benda yang tidak bergerak. Contoh: PacMan, Snake.

11. *Combat*

    Jenis *game* yang terdiri dua pemain atau lebih yang saling bertempur untuk saling mengalahkan. Contoh: Warlords.

12. *Demo*

    Jenis *game* yang khusus dikeluarkan untuk demo atau promosi.

13. *Diagnostic*

    Jenis *game* yang khusus dikeluarkan untuk mencoba dari sistem yang akan dipakai.

14. *Dodging*

    Jenis *game* yang tujuan utamanya menghindari benda-benda tertentu yang bergerak. Contoh: Frogging.

15. *Driving*

    Jenis *game* yang berdasar pada kemampuan dasar berkendara (menyetir, manuver, kontrol kecepatan dan lain-lain). Contoh: Night Driver, Pole Position.

16. *Educational*

    Jenis *game* yang dibuat untuk memberikan pengajaran. Contoh: Spelling Game, Word Games.

17. *Escape*

    Jenis *game* yang tujuannya melarikan diri dari pengejar. Contoh: Pac-Man, Ms.Pac-Man.

18. *Fighting*

    Jenis *game* yang melibatkan karakter-karakter yang saling berhadapan dan berkelahi. Contoh: Mortal Kombat, Street Fighter.

19. *Flying*

    Jenis *game* yang berdasar pada kemampuan untuk mengendarai pesawat terbang (lepas landas, mendarat, manuver, kontrol kecepatan dan lain-lain. Contoh: Flight Unlimited.

20. *Gambling*

    Jenis *game* yang mengajak pemain untuk mempertaruhkan "harta" miliknya (judi). Contoh: Black Jack, Poker.

21. *Interactive Movie*

    Jenis *game* yang jalan ceritanya dapat bercabang dan ditentukan oleh pemain. Contoh: Dragon's Lair, Johny Mnemonic.

22. *Management Simulation*

    Jenis *game* dimana pemain diharuskan untuk mengatur sumber daya yang akan digunakan untuk membangun semacam komunitas, institusi atau kerajaan. Contoh: Sim City.

23. *Maze*

    Jenis *game* yang tujuan akhirnya membutuhkan penguasaan arah pada sebuah *maze* (lorong). Contoh: Pac-Man, Ms.Pac-Man, Doom.

24. *Obstacle Course*

    Jenis *game* yang tujuannya menelusuri jalur yang sulit.

25. *Pencil and Paper Games*

    Jenis *game* yang diadaptasi dari permainan yang dimainkan dengan pensil dan kertas. Contoh: Tic-Tac-Toe, Hangman.

26. *Pinball*

    Jenis *game* yang mensimulasikan permainan pinball. Contoh: Arcade Pinball, Astrocade Pinball, Electronic Pinball.

27. *Platform*

Jenis *game* yang tujuan akhirnya dicapai dengan bergerak melalui beberapa jalan bertingkat. Contoh: SpiderMan, Donkey Kong.

28. *Programming Games*

Jenis *game* dimana pemainnya menuliskan program singkat untuk mengontrol karakter dalam game tersebut. Contoh: AI Wars.

29. *Puzzle*

Jenis *game* yang tidak menonjolkan konflik antar pemain namun lebih kepada mencari tahu solusi atau metode penyelesaiannya. Contoh: Rubik Cube, Sokoban, Tetris.

30. *Quiz*

Jenis *game* yang tujuan akhirnya adalah menjawab suatu pertanyaan dengan benar. Contoh: Video Trivia,Wizz Quiz.

31. *Racing*

Jenis *game* yang tujuan utamanya adalah memenangkan balapan. Contoh: Gran Tourismo, Need For Speed.

32. *Role Playing*

Jenis *game* dimana tiap pemain menciptakan atau memiliki karakter, dimana karakter masing-masing memiliki kemampuan yang berbeda berdasar statistik. Contoh: Anvil of Dawn, Diablo.

33. *Rhythm and Dance*

Jenis *game* dimana pemain harus menselaraskan antara waktu dan ritme musik. Contoh: Guitar Hero, VOS.

34. *Shoot 'Em Up (Shooter)*

Jenis *game* yang bertujuan berperang (tembak menembak) melawan musuh yang biasanya cenderung lebih banyak dari pemain. Contoh: Doom, Freedom Fighter.

35. *Simulation* (Lihat *Management Simulation and Training Simulation*).

36. *Sports*

Jenis *game* yang diadaptasi dari permainan olahraga di dunia nyata. Contoh: Winning Eleven, PES, American Football.

37. *Strategy*

Jenis *game* yang menekankan pemakaian strategi. Contoh: Chess.

38. *Table Top Games*

Jenis *game* yang diadaptasi dari permainan *Table Top* (di atas meja). Contoh: Battle Ping Pong, Cool Pool.

39. *Target*

Jenis *game* yang bertujuan membidik dan menembak target. Contoh: Skeet Shoot, Wabbit.

40. *Text Adventure*

Jenis game yang menggunakan text sebagai media perantaranya. Contoh: Planetfall, Zork.

41. *Training Simulation*

Jenis *game* yang menggambarkan keadaan seperti di dunia nyata dengan tujuan melatih kemampuan (menyetir atau menerbangkan). Contoh: A-10 Attack, Comanche 3.

42. *Utility*

Program yang memiliki tujuan atau fungsi tertentu yang dibuat sedemikian rupa seperti *game.*Contoh: Mario Teaches Typing.

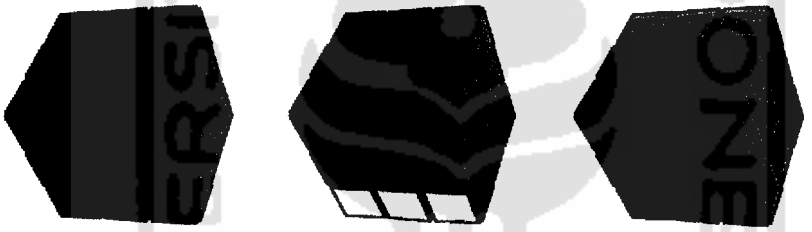
Dari pembagian klasifikasi menurut J.P Wolf diatas terlihat bahwa satu atau lebih *game* masuk dalam lebih dari satu tipe klasifikasi (lihat Tetris pada *puzzle* dan *abstract*, rubik cube pada *puzzle* dan *adaptation*). Hal ini dikarenakan tidak ada pembagian tipe *game* yang menjadi standar. Jadi dapat disimpulkan bahwa sesungguhnya pembagian tipe *game* bisa bervariasi.

2.2 *Rubik Cube*

Rubik cube adalah permainan berjenis *puzzle* yang diciptakan pada tahun 1974 oleh seorang pemahat yang juga profesor arsitektur bernama Erno Rubik yang berasal dari Hungaria. Pada awal diciptakannya, permainan ini bernama *magic cube,* yang kemudian pada tahun 1980 nama tersebut diubah menjadi *rubik cube* oleh sebuah perusahaan mainan bernama Ideal Toys.

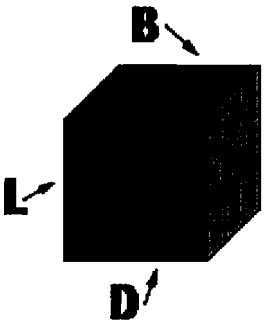
Gambar 2.1 Prototipe Awal *Rubik Cube*

Rubik cube tersusun atas dua puluh tujuh buah kubus yang saling berkaitan, namun pada kenyataannya sisi pusat kubus bukanlah kubus namun sebuah pengait yang berfungsi untuk menyatukan ke dua puluh enam kubus-kubus lainnya, yang juga berfungsi sebagai pusat dari perputaran kubus lainnya. Sebuah *rubik cube* memiliki enam warna berbeda pada tiap-tiap sisinya (merah, oranye, putih, kuning, biru dan hijau), dimana pada umumnya susunan warnanya adalah (merah berlawanan arah dengan oranye, putih berlawanan arah dengan kuning, biru berlawanan arah dengan hijau) seperti terlihat pada gambar berikut.



Gambar 2.2 Susunan Warna-Warna Pada *Rubik Cube*

Hingga kini telah dikenal standar notasi yang sering digunakan dalam permainan ini, yaitu standar yang dibuat oleh David Singmaster. Standar notasi ini digunakan untuk menandai sisi maupun arah perputaran sisi-sisi *rubik.*



Gambar 2.3 Sisi-sisi *Rubik*

R (Right) : sisi yang berlawanan dengan sisi *Left.*

L (Left) : sisi yang berada di sebelah kiri pemain (saat *rubik* lurus).

U (Up) : sisi yang berada pada bagian atas *rubik.*

D (Down) : sisi yang berlawanan dengan sisi *Up.*

F (Front) : sisi yang menghadap ke pemain.

B (Back) : sisi yang berlawanan dengan sisi *Front.*

Sementara itu terdapat tiga aturan perputaran *rubik* yang menjadi standar yaitu :

( ) : CW atau searah perputaran jarum jam (90°), tanpa simbol apapun. Contoh: F B L R.

(') : CCW berlawanan perputaran jarum jam (90°), disimbolkan dengan sebuah tanda petik . Contoh: F' B' L' R'

('') : Dua kali putaran *rubik* pada arah yang sama (180°), disimbolkan dengan dua tanda petik. Contoh: F'' B'' L'' R''.

(**2**) atau (^{2}) : Beda bentuk penulisan dari simbol ('') dua tanda kutip, yang sama saja maksudn yaitu dua kali perputaran *rubik* pada arah yang sama (180°). Contoh: F2 B2 L2 R2 atau F^2 B^2 L^2 R^2

Gambar 2.4 Arah Perputaran Rubik

Dapat dilihat pada Gambar 2.4 diatas (lihat dari kiri ke kanan) terdapat tiga macam perputaran *rubik* adalah:

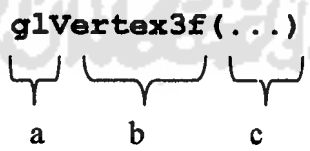
F – Sisi depan diputar 90° CW (searah jarum jam).

F' – Sisi depan diputar 90° CCW (berlawanan arah jarum jam).

F2 – Sisi depan diputar 180° CW (searah jarum jam).

2.3 OpenGL

''OpenGL is a three dimensional system. From application programmer's perspective, OpenGL primitives describe three dimensional objects that exist in a three dimensional world'', dari kutipan buku OpenGL A Primer [ANG05] tersebut dapat diberikan penjelasan bahwa OpenGL merupakan sebuah sistem tiga dimensi. Dimana dari sisi pemrogram, OpenGL dapat diartikan sebagai rutin-rutin dasar (primitif) yang menggambarkan obyek-obyek tiga dimensi pada dunia nyata (tiga dimensi), dimana hal tersebut diwujudkan dengan API (*Application Programming Interface*).



Diatas merupakan contoh salah satu fungsi dasar dari OpenGL, fungsi tersebut berguna untuk membuat lokasi sebuah titik, keterangan berikut ini:

a = kepustakaan GL (fungsi tersebut masuk dalam kepustakaan atau library opengl).

b = perintah dasar (fungsi tersebut merupakan fungsi penentu lokasi titik 3dimensi dengan argumen yang diisikan bertipe float).

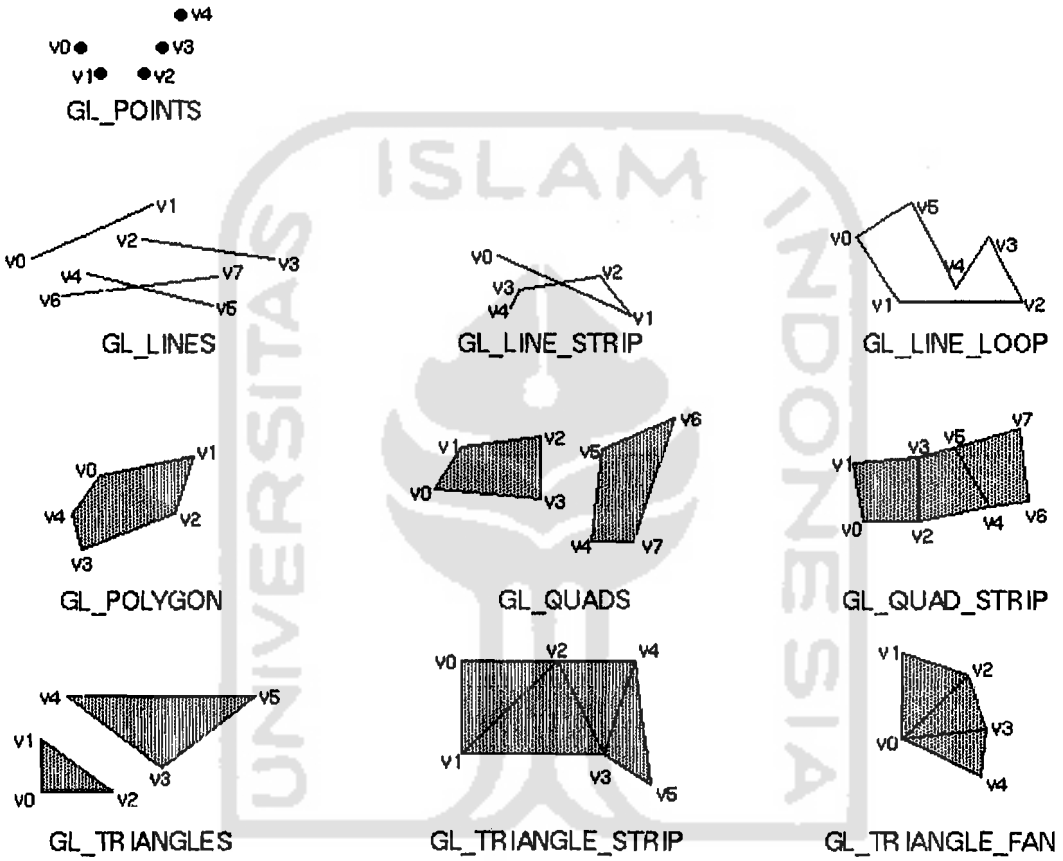
c = isian argumen atau variabel yang diisi (koordinat titik x,y dan z).

Tabel 2.1 Perintah-Perintah pada OpenGL

Perintah	**Arti**	**Keterangan**
glVertex2i(x,y)	Lokasi titik berada di (x,y)	Tipe argumen adalah integer dan 2Dimensi yaitu x dan y
glVertex2f(x,y)	Lokasi titik berada di (x,y)	Tipe argumen adalah float dan 2Dimensi yaitu x dan y
glVertex3i(x,y,z)	Lokasi titik berada di (x,y,z)	Tipe argumen adalah integer dan 3Dimensi yaitu x,y,z.
glVertex3f(x,y)	Lokasi titik berada di (x,y,z)	Tipe argumen adalah float dan 3Dimensi yaitu x dan y
glClearColor(R,G,B,α)	Warna latar belakang	Komponen warna Red, Green, Blue dan alpha
glColor3f(R,G,B)	Warna latar depan	Komponen warna Red, Green dan Blue
glColor4f(R,G,B,α)	Warna latar depan	Komponen warna Red, Green, Blue dan alpha
glPointSize(k)	Ukuran titik k piksel	k adalah bilangan integer
glBegin(GL_POINTS)	Titik	Titik individu

glBegin(GL_LINES)	Garis	Sepasang titik yang direpresentasikan sebagai sebuah garis
glBegin(GL_LINE_STRIP)	Poligaris	Beberapa garis yang terhubung
glBegin(GL_LINE_LOOP)	Poligaris tertutup (poligon)	Sama seperti diatas namun ditambahkan sebuah garis yang menghubungkan titik awal dan titik akhir
glBegin(GL_TRIANGLES)	Segitiga	Tiga buah titik yang diwujudkan dengan sebuah segitiga
glBegin(GL_TRIANGLE_STRIP)	Segitiga	Segitiga yang saling terkait
glBegin(GL_TRIANGLE_FAN)	Segitiga	Segitiga yang saling terkait dan membentuk kipas
glBegin(GL_QUADS)	Segiempat	Empat titik yang ditampilkan dalam bentuk sebuah poligon segi empat
glBegin(GL_QUAD_STRIP)	Segiempat	Empat titik yang ditampilkan dalam bentuk sebuah poligon segi empat yang saling terkait
glEnd()	Akhir perintah OpenGL	Menutup OpenGL

Dapat dilihat pada Tabel 2.1 diatas adalah perintah-perintah dasar yang digunakan pada pemrograman OpenGL, sedangkan pada Gambar 2.5 berikut adalah gambar objek-objek primitif yang digunakan dalam OpenGL.



Gambar 2.5 Objek-Objek Primitif OpenGL

2.4 Transformasi

Dalam buku OpenGL A Primer [ANG05] transformasi dinyatakan sebagai berikut: "*Transformation are the key to manipulating geometric objects, to animating scenes, and to obtaining the desired views*", yang kurang lebih dapat diartikan bahwa transformasi adalah kunci untuk memanipulasi objek-objek

geometris, membuat tampilan lebih hidup dan mendapatkan tampilan yang diinginkan. Pada dasarnya transformasi adalah memindahkan objek tanpa merusak bentuk dari objek tersebut.

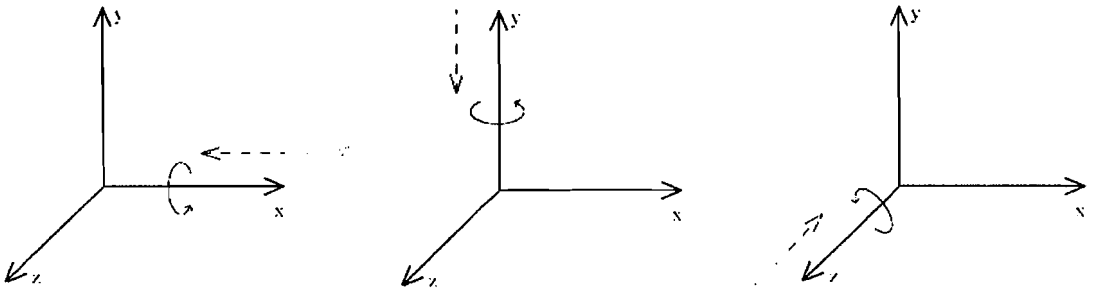
Sedangkan tujuan dari transformasi adalah [SUY03] :

1. Merubah atau menyesuaikan komposisi pandangan.
2. Memudahkan membuat objek yang simetris.
3. Melihat objek dari sudut pandang yang berbeda.
4. Memindahkan satu atau beberapa objek dari satu tempat ke tempat lain.

Transformasi meliputi translasi, penskalaan, rotasi (putaran) dan *shearing*. Dari beberapa proses transformasi tersebut disini penulis hanya akan membahas tentang proses rotasi (putaran), ini dikarenakan proses rotasi berkaitan langsung dengan proses perputaran sisi pada *rubik cube* (perangkat lunak yang akan dibuat).

2.5 Rotasi

Pada pemrograman grafika komputer baik 2 dimensi maupun 3 dimensi rotasi adalah salah satu jenis transformasi yang sering digunakan, rotasi menyebabkan suatu objek bergerak berputar pada titik pusat atau pada sumbu putar tertentu. Putaran dapat dilakukan pada sumbu-*x* (*roll*), sumbu-*y* (*pitch*), sumbu-*z* (*yaw*) maupun sumbu putar lainnya. Menurut aturan geometri perputaran sudut positif adalah berlawanan dengan arah jarum jam, seperti dapat dilihat pada Gambar 2.6 berikut.



Gambar 2.6 Sumbu Putar Sudut Positif (Berlawanan Arah Jarum Jam)

Rotasi terhadap sumbu-*x* (*roll*) didefinisikan dengan persamaan sebagai berikut:

$$x' = x$$
$$y' = y\cos\theta - z\sin\theta$$
$$z' = y\sin\theta + z\cos\theta$$

ekuivalen ...

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Rotasi terhadap sumbu-*y* (*pitch*) didefinisikan dengan persamaan sebagai berikut:

$$x' = z\sin\theta + x\cos\theta$$
$$y' = y$$
$$z' = z\cos\theta - x\sin\theta$$

ekuivalen ...

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Rotasi terhadap sumbu-*z* (*yaw*) didefinisikan dengan persamaan sebagai berikut:

$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta$$
$$z' = z$$

ekuivalen ...

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Dimana x', y' dan z' adalah koordinat titik yang baru setelah diputar sebesar sudut θ, dan x, y dan z adalah koordinat titik awal.

2.6 Microsoft Visual C++

Microsoft Visual C++ merupakan salah satu perangkat lunak yang digunakan dalam pengembangan (perangkat pengembangan) aplikasi yang menggunakan bahasa C++. Visual C++ merupakan sebuah IDE (*Integrated Development Environment*) atau lingkungan pengembangan terintegrasi yang menyediakan beragam fasilitas bagi pengembang aplikasi, seperti membuat kode program, mengedit aplikasi, mengkompilasi maupun menguji aplikasi secara lebih cepat dan lebih mudah. Beberapa komponen utama dalam Visual C++ adalah sebagai berikut [KAD04]:

- **Editor**

  Editor menyediakan sarana bagi pemrogram untuk menliskan program. Yang menarik, editor yang disediakan mampu mengenali kata-kata tercadang C++ dan akan memberi warna tersendiri terhadap kata-kata seperti itu. Keuntungannya, program menjadi lebih mudah dibaca dan sekiranya anda melakukan kesalahan dalam menuliskan kata-kata itu maka akan lebih cepat terdeteksi.

- **Kompiler**

  Kompiler adalah perangkat lunak yang berfungsi untuk menerjemahkan kode sumber (*source code*) kedalam bentuk bahasa mesin. Tentu saja piranti ini dapat memberikan pesan-pesan kesalahan jika terjadi kesalahan kaidah penulisan program yang terdeteksi pada tahap proses kompilasi. Hasil kompilasi berupa kode objek (*object code*) yang disimpan dalam berkas berekstensi (***.obj**).

- **Linker**

  Linker adalah perangkat lunak yang berfungsi menggabungkan berbagai modul yang dihasilkan oleh kompiler dan modul kode dari berbagai pustaka C++, serta membentuk menjadi kode yang dapat dieksekusi. Sebagaimana kompiler, linker juga dapat mendeteksi kesalahan. Kesalahan yang terjadi sewaktu proses *linking* bisa disebabkan karena ada bagian pustaka atau bagian program yang tidak ditemukan.

- **Pustaka**

  Visual C++ menyediakan berbagai pustaka (*library*) yang memudahkan pemrogram dalam melakukan berbagai operasi seperti melakukan berbagai operasi seperti menghitung akar kuadrat dan mengakses *database*. Pustaka-pustaka yang tersedia antara lain berupa:

  - *Standard C++ library* (berisi semua rutin yang tersedi pada kebanyakan kompiler C++)
  - *Microsoft Foundation Classes and Templates* (MFC&T), yang berkaitan dengan pemrograman Windows.

- **AppWizard**

  Perangkat ini bermanfaat untuk membangkitkan suatu kerangka dasar aplikasi Windows yang sangat memudahkan pemrogram untuk membuat aplikasi Windows.

- **ClassWizard**

  Perangkat ini bermanfaat untuk mengedit kelas-kelas yang dibangkitkan oleh AppWizard.

BAB III

METODOLOGI

3.1 Metode Analisis

Dalam proses pengembangan sebuah sistem perangkat lunak, sebelum kita memasuki tahap perancangan sistem dan tahap pengkodean (implementasi) kita diharuskan melakukan analisa terhadap sistem yang akan dibuat. Analisis diperlukan agar sistem yang dibuat kelak dapat menjadi sebuah sistem yang memenuhi dan sesuai kebutuhan. Analisis sistem adalah sebuah tehnik pemecahan masalah dengan menguraikan sebuah sistem secara utuh menjadi bagian-bagian komponen penyusun sistem yang lebih kecil, guna mempelajari kinerja dari komponen-komponen dalam sistem tersebut.

Metode analisis yang digunakan dalam pengembangan perangkat lunak pada penelitian ini adalah metode analisis berorientasi objek (*Object Oriented Analysis*). Analisis berorientasi objek (*Object Oriented Analysis*) adalah tahapan perangkat lunak dengan menentukan SRS (*Sytem Requirement Spesification*), identifikasi kelas-kelas serta hubungannya satu terhadap yang lain [NUG05]. Dalam hal ini kita harus melihat sistem sebagai kumpulan objek-objek yang saling berinteraksi.

Aplikasi *game puzzle rubik cube* merupakan versi komputer implementasi dari permainan *puzzle rubik cube* yang sudah dikenal selama ini, dimana para pemain ditantang untuk menyamakan seluruh sisi kubus yang terdiri dari enam sisi warna yang berbeda dengan cara memutar setiap sisinya, yang sebelumnya ke

enam sisi tersebut diputar secara acak. Untuk menyamakan seluruh sisi tersebut pemain dapat memutar seluruh sisi kubus baik searah maupun berlawanan arah dengan arah jarum jam. Tidak ada batasan waktu dalam menyelesaikan *game* ini, pemain dapat bermain sampai selesai maupun memutuskan untuk menyerah.

3.2 Hasil Analisis

Berdasarkan deskripsi dan teori-teori pada bagian terdahulu maka didapatkan beberapa hasil analisis kebutuhan yang berorientasi objek, antara lain:

3.2.2 Identifikasi Aktor

Aktor adalah orang atau sistem yang berhubungan dengan sistem serta berada diluar sistem. Dari definisi tersebut didapatkan aktor sebagai berikut:

- **Aktor** pada *game* ini adalah **pemain** (yang bermain *game puzzle rubik cube).*

3.2.3 Identifikasi *Use case*

Use case dapat didefinisikan sebagai hal-hal atau tindakan-tindakan yang dilakukan oleh aktor pada sistem. Dapat dianalisis beberapa use case pada aplikasi ini, antara lain:

- *Use case* **pemanggilan game**, pada *use case* ini pemain memanggil aplikasi untuk dijalankan.
- *Use case* ***new game***, pada *use case* ini pemain memulai permainan baru.
- *Use case* ***options***, pada *use case* ini pemain menentukan pengaturan pada permainan. Mencakup menentukan tekstur *rubik*, dan kecepatan animasi putaran sisi *rubik.*

- *Use case* ***besttime***, pada *use case* ini pemain dapat melihat catatan rekor waktu yang ada.
- *Use case* ***help***, pada *use case* ini pemain membuka file bantuan yang ada.
- *Use case* ***about***, pada *use case* ini pemain membuka form about yang berisi keterangan tentang pembuat *game*.
- *Use case* ***exit***, pemain menutup dan mengakhiri aplikasi *game*.

3.2.4 Identifikasi Objek-objek

Dalam menentukan objek-objek pada sebuah sistem hendaknya menggunakan kata benda sebagai basis untuk penentuan kelas atau objek [NUG05]. Dapat dianalisis objek-objek antara lain:

- Cubik (tersusun atas dua puluh tujuh kubus).
- Kubus (tersusun atas enam sisi atau enam buah persegi).
- Sisi (tersusun atas empat titik).
- Titik (tersusun atas koordinat : x,y dan z).
- *Window* atau Jendela tempat menampilkan hasil render Rubik pada layar monitor.

3.2.5 Identifikasi Metode atau Fungsi

Pada *game* ini dibutuhkan beberapa metode agar *game* dapat berjalan dengan baik, beberapa metode yang dapat dianalisis antara lain:

- Menggambar rubik, metode ini akan menjalankan metode penggambaran kubus sebanyak dua puluh tujuh kali.
- Menggambar kubus, metode ini akan menjalankan metode penggambaran sisi sebanyak enam kali.

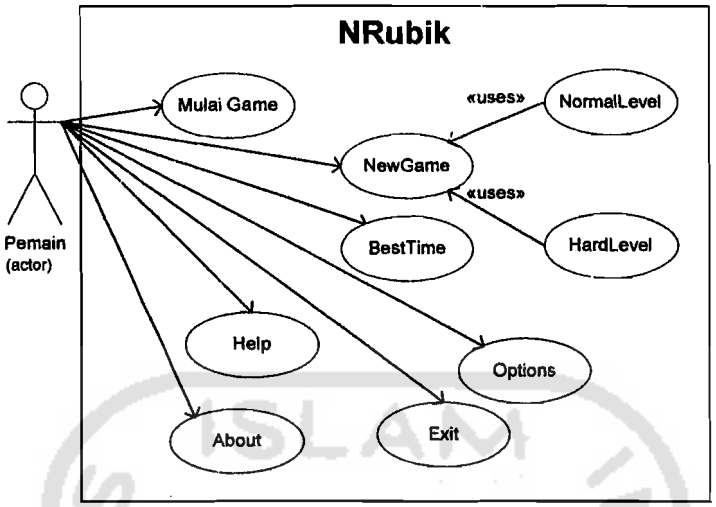
- Menggambar sisi, metode ini akan menggambar sebuah persegi (empat buah titik).
- Menentukan koordinat, metode ini akan memberikan koordinat awal posisi titik-titik (*verteks*) pada keempat titik sisi-sisi rubik.
- Memutar sisi, metode ini akan memutar sisi-sisi rubik.
- Mengalikan koordinat awal dengan rumus rotasi agar koordinat titik-titik pada sisi rubik dapat berubah, sehingga sisi rubik dapat berputar.

3.3 Metode Perancangan

Setelah mendapatkan kebutuhan perangkat lunak pada tahap analisis, maka kita dapat melakukan perancangan sebelum masuk ke tahap implementasi. Beberapa tahap perancangan yang akan dilaksanakan dengan membuat diagram-diagram UML antara lain:

3.3.1 Diagram Use case

Diagram *use case* menunjukkan interaksi antara sistem dengan aktor ataupun sistem eksternal lainnya. Berdasarkan pada analisis yang telah dilakukan maka dapat digambarkan diagram *use case* sebagai berikut:



Gambar 3.1 Diagram *Use Case*

3.3.2 Diagram Kelas

Diagram kelas menggambarkan perangkat lunak secara statis, beserta hubungan (relasi-relasi) kelas yang satu dengan kelas-kelas lainnya. Disini juga digambarkan operasi-operasi yang di lakukan sebuah kelas beserta atribut-atributnya. Kelas-kelas yang digunakan pada pembuatan *game* Nrubik diantaranya:

a. Kelas CNRubikDlg

Kelas ini adalah kelas yang digunakan sebagai tampilan utama (form menu utama) pada kelas ini terdapat metode-metode yang berkaitan dengan menu-menu yang akan kita pilih (misal: metode OnMenuAbout yang akan dijalankan jika kita memilih menu *about* pada menu utama atau kelas ini, OnMenuHelp yang akan dijalankan jika kita memilih menu *help,* OnMenuTimer yang akan menampilkan catatan waktu permainan pada menu utama ini dan lain-lain).

b. Kelas COpenGL

Kelas ini adalah kelas yang digunakan sebagai "kanvas" bagi obyek OpenGL yang akan kita buat atau kita gambar, objek dari kelas ini kelak akan ditempelkan pada objek dari kelas CNRubikDlg (form utama). Pada kelas ini juga menangani objek cubik (rubik) sebagai sebuah objek utuh (terdiri dari dua puluh tujuh kubus).

c. Kelas RKubus

Kelas ini adalah kelas yang akan menangani objek kubus (yang terdiri atas enam buah sisi).

d. Kelas RSisi

Kelas ini adalah kelas yang akan menangani objek sisi (yang terdiri atas empat buah titik pada setiap sisinya).

e. Kelas RTitik

Kelas ini adalah kelas yang merupakan realisasi dari keseluruhan objek Rubik (cubik), karena kelas inilah yang sesungguhnya yang menyusun seluruh objek dengan koordinat-koordinat 3D (x, y, z).

f. Kelas CBestTimeDlg

Kelas ini adalah kelas yang akan menampilkan form waktu rekor (*record time*).

g. Kelas COptionsDlg

Kelas ini adalah kelas yang akan menampilkan form pilihan seting *game* (*options*). Kelas ini akan memberikan pilihan pengubahan tekstur maupub kecepatan putar rubik.

h. Kelas CAboutDlg

Kelas ini adalah kelas yang akan menampilkan form tentang (*about*).

i. Kelas CFillRecDlg

Kelas ini adalah kelas yang akan menampilkan form untuk pengisian nama bagi pemecah waktu rekor (*record time*).

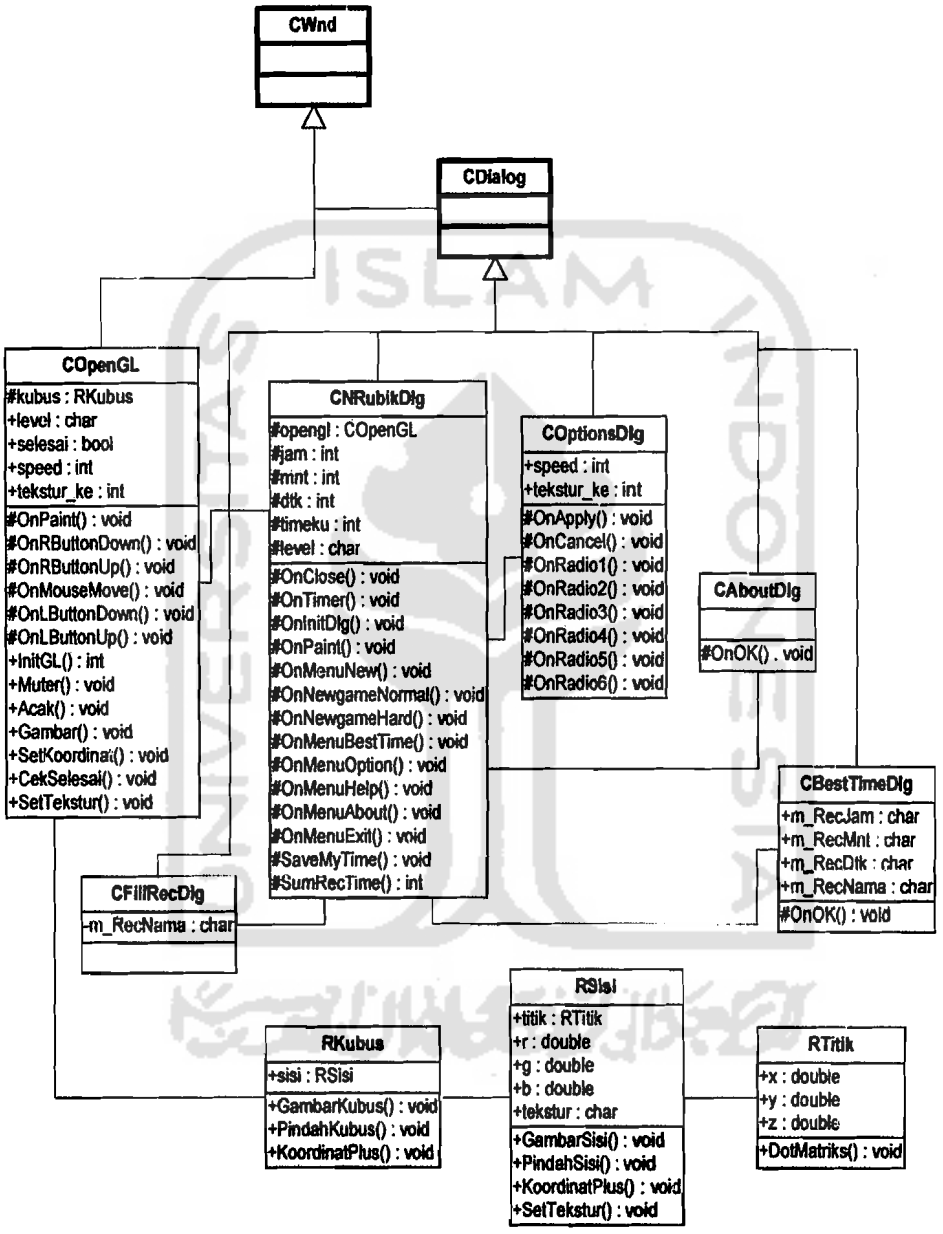
j. CWnd

Kelas bawaan Visual C++ (MFC) yang menangani aplikasi window.

k. CDialog

Kelas bawaan Visual C++ (MFC) yang digunakan sebagai pembentuk form (dialog). Kelas ini akan menjadi turunan bagi kelas-kelas dialog (form-form) yang akan diciptakan pada sebuah aplikasi MFC yang berbasis dialog.

Berdasarkan pada analisis yang telah dilakukan maka dapat digambarkan diagram kelas seperti pada gambar 3.2 berikut :



Gambar 3.2 Diagram Kelas

\# : **protected** (hanya dapat diakses kelas tersebut beserta turunannya).

\+ : **public** (dapat diakses seluruh kelas).

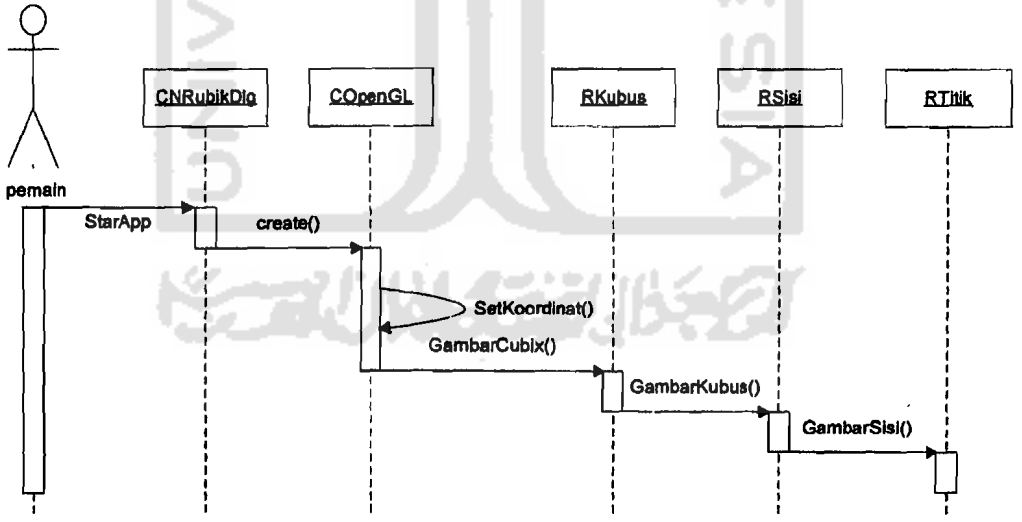
Kelas **CWnd** dan **CDialog** (cetak tebal), adalah kelas bawaan Visual C++.

3.3.3 Diagram Sekuen

Diagram sekuen atau yang sering disebut *sequence diagram* adalah salah satu diagram UML yang memodelkan logika sebuah *use case*, dengan kata lain diagram ini menggambarkan sebuah *use case*. Diagram sekuen menggambarkan bagaimana objek berinteraksi dengan satu sama lain melalui pesan pada eksekusi sebuah *use case* [WHI04]. Diagram ini terdiri atas dimensi vertikal (atas-bawah) yang menggambarkan waktu atau masa aktif objek, sedangkan dimensi horisontal (datar) menggambarkan pesan (interaksi) antara objek-objek terkait.

1. Diagram Sekuen Mulai*Game*

Diagram sekuen untuk *use case* Mulai*Game* dapat dilihat pada Gambar 3.3 sebagai berikut :

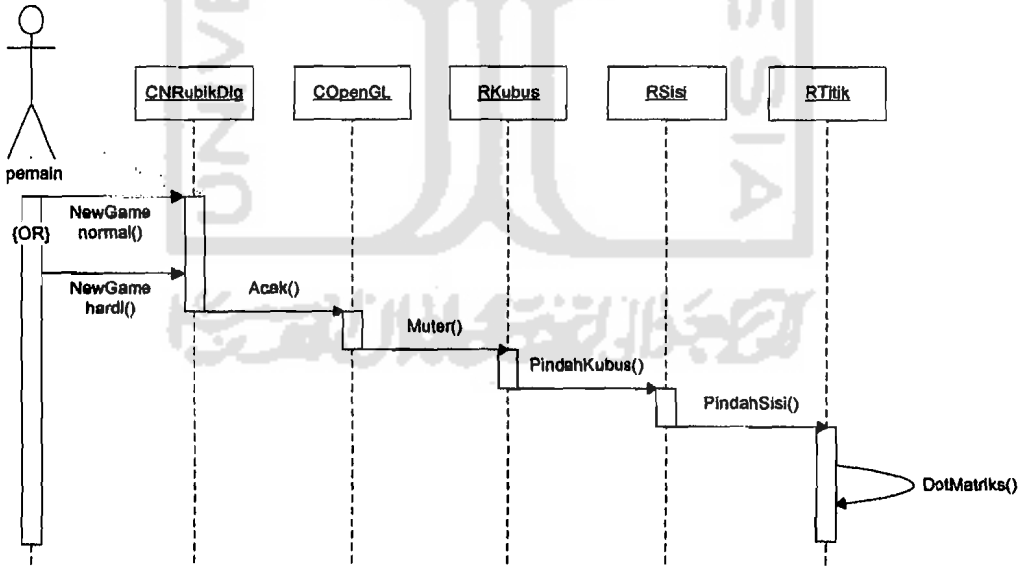


Gambar 3.3 Diagram Sekuen Mulai*Game*

Dari gambar diatas dapat dilihat bahwa ketika pemain menjalankan aplikasi untuk pertama kali maka kelas yang dipanggil pertama adalah kelas CNRubikDlg kelas tersebut akan memunculkan *main form* (form utama) dari game ini, kemudian kelas tersebut akan menciptakan objek COpenGL. Kelas COpenGL sendiri akan mengeset koordinat lalu memanggil fungsi GambarCubik dan memanggil kelas RKubus yang akan memanggil fungsi gambar kubus dan seterusnya hingga tergambar sebuah rubik cube utuh.

2. Diagram Sekuen *NewGame*

Diagram sekuen untuk *use case NewGame* dapat dilihat pada Gambar 3.4 sebagai berikut :



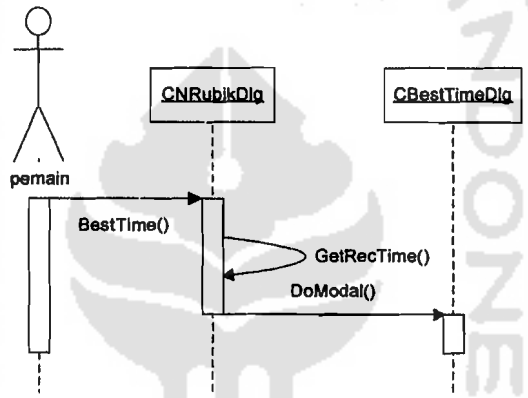
Gambar 3.4 Diagram Sekuen *NewGame*

Pada gambar diatas dapat dilihat pemain dapat memilih permainan baru dengan level normal atau hard. Lalu sistem berturut-turut akan melakukan

pengacakan terhadap koordinat rubik dengan menjalankan metode-metode yang mendukungnya.

3. Diagram Sekuen *BestTime*

Diagram sekuen untuk *use case BestTime* dapat dilihat pada Gambar 3.5 sebagai berikut :

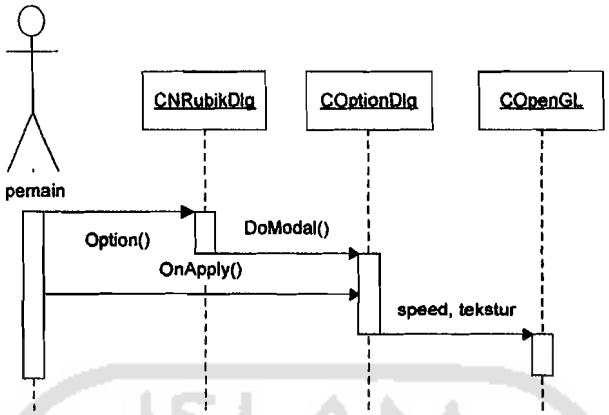


Gambar 3.5 Diagram Sekuen *BestTime*

Pada gambar diatas pemain membuka menu *BestTime (record time),* kelas NRubikDlg (menu utama) akan meload catatan waktu rekor yang ada lalu menampilkan form *best time (record time).*

4. Diagram Sekuen *Options*

Diagram sekuen untuk *use case Options* dapat dilihat pada Gambar 3.6 sebagai berikut :

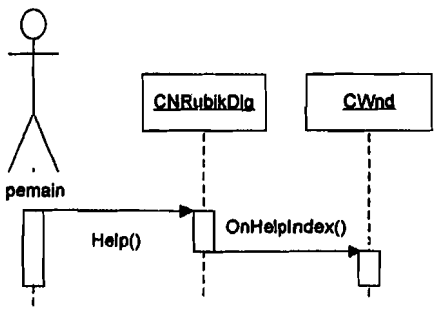


Gambar 3.6 Diagram Sekuen *Options*

Pada gambar diatas pemain membuka menu *Options* yang kemudian kelas NRubikDlg (menu utama) akan menampilkan form *option* (kelas COptionDlg) setelah menu *options* tampil pemain bisa memilih setting tekstur dan kecepatan. Pada akhirnya settingan pengguna akan di transfer ke kelas COpenGL untuk disetting ke *game*.

5. Diagram Sekuen *Help*

Diagram sekuen untuk *use case Help* dapat dilihat pada Gambar 3.7 sebagai berikut :

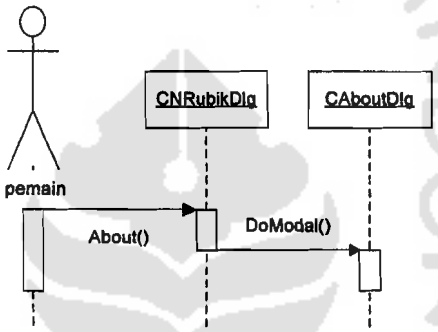


Gambar 3.7 Diagram Sekuen *Help*

Pada gambar diatas pemain membuka menu *Help* (bantuan), kelas NRubikDlg (menu utama) akan meminta metode *OnHelpIndex* dari kelas CWnd.

6. Diagram Sekuen *About*

Diagram sekuen untuk *use case About* dapat dilihat pada Gambar 3.8 sebagai berikut :

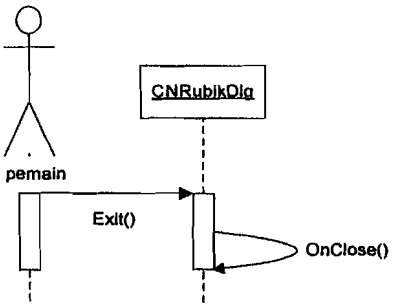


Gambar 3.8 Diagram Sekuen *About*

Pada gambar diatas pemain membuka menu *About* (tentang), kelas NRubikDlg (menu utama) akan menampilkan form *about* (kelas CaboutDlg).

7. Diagram Sekuen *Exit*

Diagram sekuen untuk *use case Exit* dapat dilihat pada Gambar 3.8 sebagai berikut :



Gambar 3.9 Diagram Sekuen *Exit*

3.3.4 Perancangan Antarmuka

1. Jendela Utama (CNRubikDlg)

Perancangan pada jendela utama memiliki dua menu utama dapat dilihat hierarkinya sebagai berikut:

▶ Game ▶▶ New game ▶▶▶ normal (memulai game level normal).

▶▶▶ hard (memulai game level hard).

▶ RecordTime (melihat catatan waktu terbaik).

▶ Options (memilih pengaturan *game*).

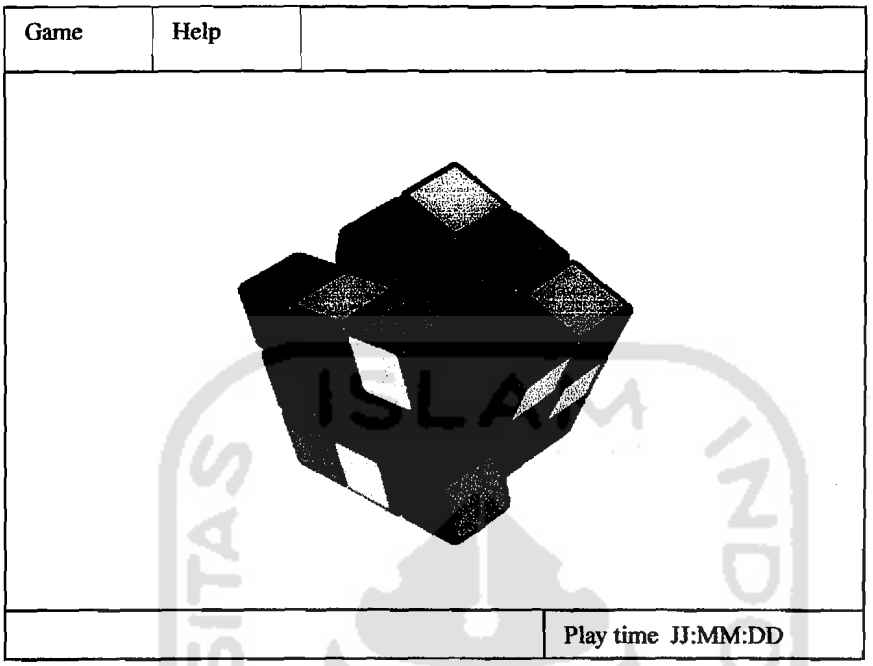
▶ Exit (keluar permainan).

▶ Help ▶ Help (membuka file bantuan).

▶ About (membuka file tentang).

Keterangan : ▶ Menu ▶ SubMenu ▶▶ Sub SubMenu

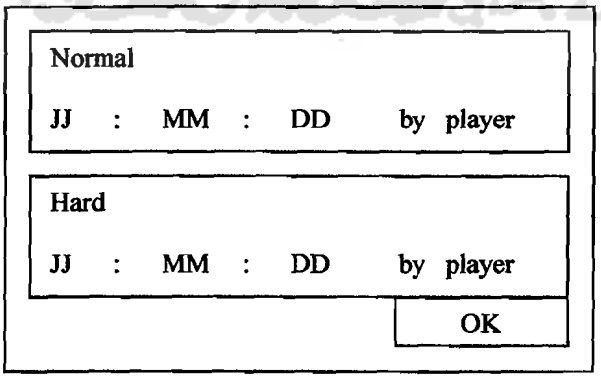
Gambar perancangan dari form menu utama (kelas CNRubikDlg) dapat dilihat pada gambar 3.10 berikut:



Gambar 3.10 Rancangan Form Menu Utama

2. *Record time (BestTime)*

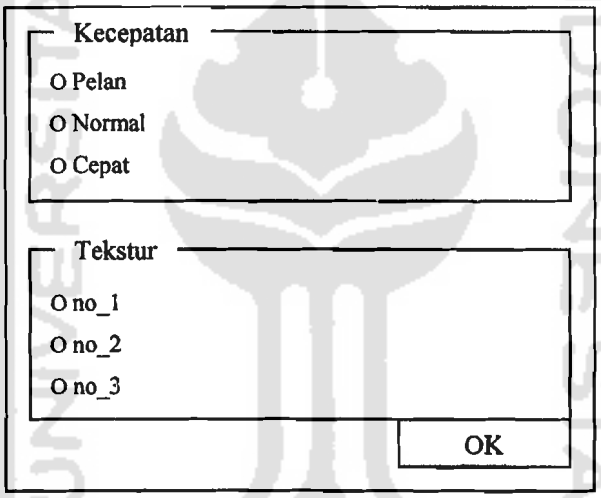
Pada antarmuka form *record time* memiliki dua tampilan catatan waktu rekor, yaitu catatan rekor level normal dan catatan waktu rekor level hard, masing masing catatan rekor memiliki format (jam : mnt : dtk). Perancangan antarmuka form *record time* dapat dilihat pada gambar 3.11 berikut :



Gambar 3.11 Rancangan Form *Record Time*

3. *Option*

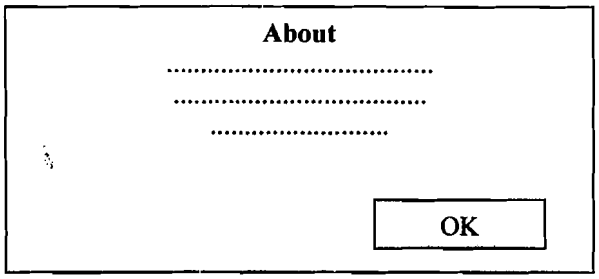
Pada antarmuka form *options* memiliki dua pilihan setingan pada *game,* yaitu seting kecepatan putar sisi rubik dan seting pilihan tekstur. Pada seting kecepatan menyediakan tiga pilihan kecepatan putar (pelan, normal, cepat) yang masing-masing dapat dipilih dengan *radio box*. Pada setingan tekstur memiliki tiga pilihan tekstur yang dipilih juga menggunakan radio box. Perancangan antarmuka form *options* dapat dilihat pada gambar 3.12 berikut :



Gambar 3.12 Rancangan Form Options

4. *About*

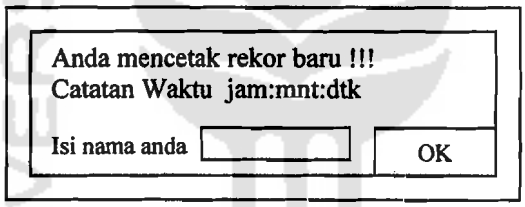
Form ini digunakan untuk melihat catatan about (tentang) *game* ini baik pembuat tahun pembuatanya dan data-data lainnya. Perancangan antarmuka formnya dapat dilihat pada gambar 3.13 berikut :



Gambar 3.13 Rancangan Form About

5. *Isi Pencetak Rekor*

Form ini digunakan jika kita menyelesaikan permainan dengan catatan waktu yang lebih baik dari catatan waktu yang sudah ada. Perancangan antarmuka formnya dapat dilihat pada gambar 3.14 berikut :



Gambar 3.14 Rancangan Form Isi Pencetak Rekor

BAB IV

HASIL DAN PEMBAHASAN

4.1 Implementasi

Setelah melalui tahap analisis dan perancangan maka kini kita telah sampai pada tahap implementasi.

4.1.1 Spesifikasi Sistem yang Digunakan

Implementasi hardware yang digunakan penulis dalam membuat, mengkompilasi hingga menjalankan perangkat lunak ini adalah:

1. *Processor Mobile* AMD Sempron 2800.
2. *Memory* 224MB
3. VGA *shared* 32 MB (resolusi 1024 x 768).
4. *Harddisk* 40 GB.
5. Monitor.
6. *Keyboard.*
7. *Mouse.*

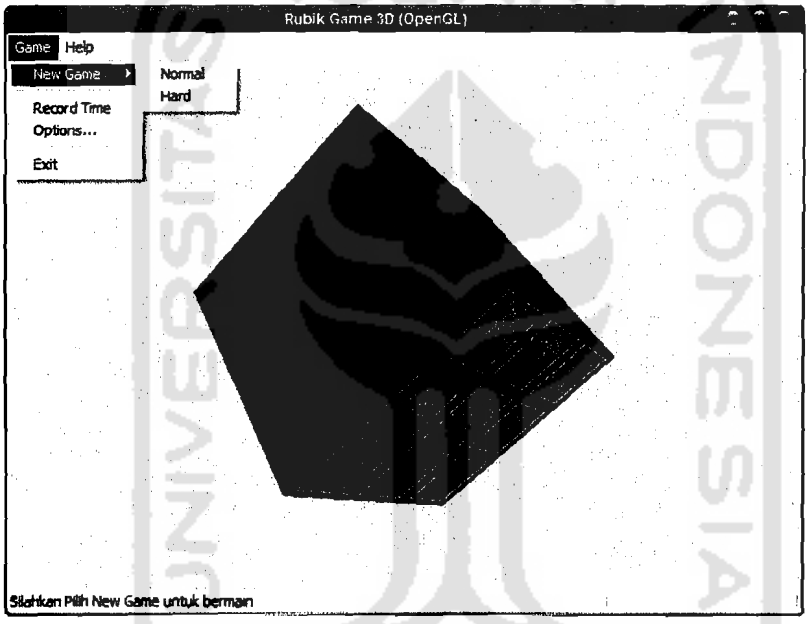
Implementasi software yang digunakan dalam membuat, mengkompilasi hingga menjalankan perangkat lunak ini adalah:

1. Microsoft Windows XP 2002 *Home Edition Service Pack 2* ***Original.***
2. Microsoft Visual C++ 6.0.
3. Tambahan librari OpenGL (glAux.lib) dan file header (glut.h).

4.1.2 Implementasi Antarmuka

1. Antarmuka Jendela Utama

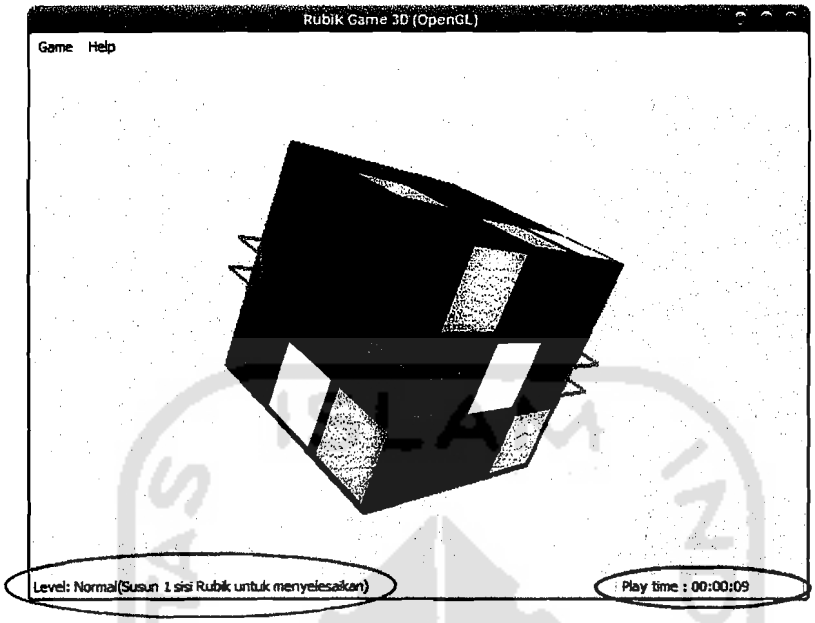
Jendela ini merupakan jendela utama atau tampilan awal ketika kita membuka aplikasi *game* ini. Pada tahap ini rubik akan memutar secara otomatis (animasi) selama kita belum memulai game (memilih submenu *NewGame* Normal atau submenu *NewGame Hard*).



Gambar 4.1 Form Menu Awal

2. *NewGame (hard* atau normal*)*

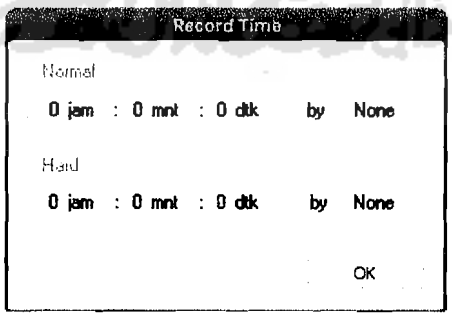
Jika pilihan pada salah satu submenu *NewGame* sudah dipilih maka tampilan akan seperti Gambar 4.2 dibawah, akan muncul informasi level yang kita pilih (lingkaran kiri bawah) level Normal atau *Hard*, dan juga akan muncul informasi waktu lama permainan (lingkaran kanan bawah) sampai kita bisa menyelesaikan permainan.



Gambar 4.2 *NewGame*

3. RecordTime

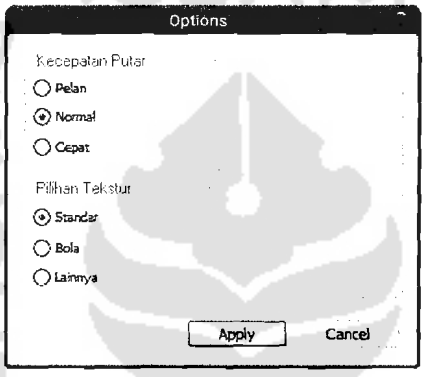
Form *Record Time* menampilkan menu waktu rekor (*record time*). Form ini menampilkan form *record time* seperti pada Gambar 4.3 berikut. Jika kita bisa melampaui rekor yang tercatat pada menu dibawah maka nama dan waktu (penyelesaian) kita yang akan menggantikan pemilik rekor waktu sebelumnya.



Gambar 4.3 Form *RecordTime*

4. *Options*

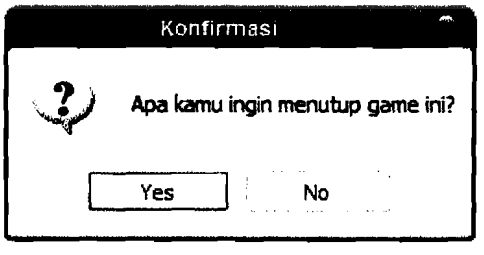
Form *options* ditampilkan jika kita ingin mengubah setingan (tekstur atau kecepatan putaran sisi rubik). Menu ini menampilkan form *options* seperti pada Gambar 4.4 berikut. Kita bisa memilih setingan kecepatan putar sisi-sisi rubik dan tekstur rubik. Jika kita menekan tombol *Apply* maka seting berubah jika kita tekan tombol *Cancel* maka seting tidak berubah.



Gambar 4.4 Form *Options*

5. *Exit*

Form *exit* dijalankan jika kita memilih submenu *exit* atau menekan tombol *close* (x) pada kanan atas *window game* ini. Seperti dilihat pada Gambar 4.5 berikut.



Gambar 4.5 Form *Exit*

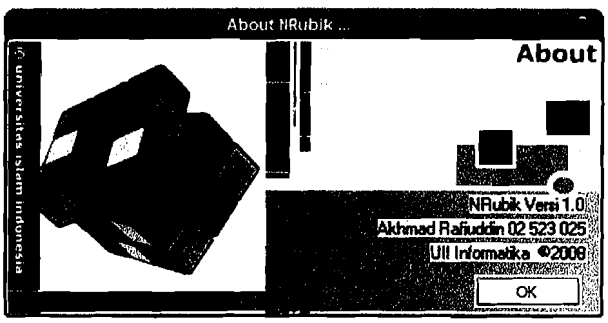
6. *Help*

Form ini merupakan file bantuan (keterangan-keterangan yang diperlukan pada program) seperti cara bermain dan *kontrol* permainan, seperti dapat dilihat pada Gambar 4.6 berikut.

Gambar 4.6 File *Help*

7. *About*

Form *about* berisi data-data tentang pembuat *game* ini. Ditampilkan jika kita menmilih sub menu About. Tampilannya dapat dilihat pada Gambar 4.7 berikut.



Gambar 4.7 Form *About*

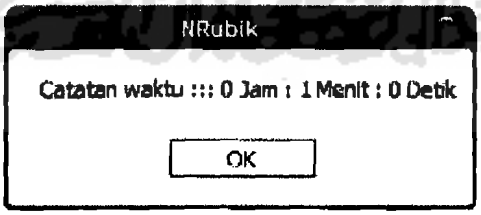
8. ***NewRecord***

Form *newrecord* ini akan keluar jika kita berhasil memecah rekor waktu permainan yang sudah ada (catatan kita lebih cepat dari pada catatan terdahulu yang tersimpan) atau belum ada yang pernah bisa menyelesaikan sehingga kita menjadi orang pertama yang akan mengisi catatan waktu. Tampilannya dapat dilihat pada Gambar 4.8 berikut.

Gambar 4.8 Form *NewRecord*

9. ***Game* selesai**

Form ini ditampilkan jika kita berhasil menyelesaikan *game* baik level normal maupun *hard* dengan tidak memecahkan catatan rekor yang ada. Tampilnnya dapat dilihat pada gambar 4.9 berikut.



Gambar 4.9 Form Selesai (tidak cetak rekor)

4.1.3 Implementasi Prosedural

1. MulaiGame

Sesuai dengan perancangan pada bab sebelumnya proses mulai game melalui beberapa tahapan sebagai berikut:

- Create objek dari kelas COpenGL berikut scriptnya:

```
CRect rect;

GetClientRect(&rect);

int right = rect.Width();
int bottom    = rect.Height();

// SetRect(left,top,right,bottom)
rect.SetRect(0, 0, right, bottom-18);
opengl.Create(NULL,
              NULL,
              WS_CHILD|WS_CLIPSIBLINGS|WS_CLIPCHILDREN
              |
              WS_VISIBLE,
              rect,
              this,
              1);
```

- SetKoordinat mengisi koordinat x,y,z :

```
for(int k=0;k<27;k++)
     {
     // Set koordinat sisi kanan (kuning)
     kubus[k].sisi[0].titik[0].x = (x + 0.5);
     kubus[k].sisi[0].titik[0].y = (y - 0.5);
     kubus[k].sisi[0].titik[0].z = (z - 0.5);

     kubus[k].sisi[0].titik[1].x = (x + 0.5);
     kubus[k].sisi[0].titik[1].y = (y + 0.5);
     kubus[k].sisi[0].titik[1].z = (z - 0.5);

     kubus[k].sisi[0].titik[2].x = (x + 0.5);
     kubus[k].sisi[0].titik[2].y = (y + 0.5);
     kubus[k].sisi[0].titik[2].z = (z + 0.5);

     kubus[k].sisi[0].titik[3].x = (x + 0.5);
     kubus[k].sisi[0].titik[3].y = (y - 0.5);
     kubus[k].sisi[0].titik[3].z = (z + 0.5);
                         :
                         :
     }
```

- Proses penggambaran Objek (Cubik)

```
for (int k=0;k<27;k++)
{
     kubus[k].GambarKubus();
}
```

- Proses penggambaran Objek (Kubus)

```
sisi[0].GambarSisi();
sisi[1].GambarSisi();
sisi[2].GambarSisi();
sisi[3].GambarSisi();
sisi[4].GambarSisi();
sisi[5].GambarSisi();
```

- Proses penggambaran Objek (Sisi)

```
glEnable(GL_TEXTURE_2D);          // enable tekstur
glBegin(GL_POLYGON);
     glTexCoord2f(0.0f, 0.0f);
     glVertex3f(titik[0].x,titik[0].y,titik[0].z);
     glTexCoord2f(1.0f, 0.0f);
     glVertex3f(titik[1].x,titik[1].y,titik[1].z);
     glTexCoord2f(1.0f, 1.0f);
     glVertex3f(titik[2].x,titik[2].y,titik[2].z);
     glTexCoord2f(0.0f, 1.0f);
     glVertex3f(titik[3].x,titik[3].y,titik[3].z);
glEnd();
glDisable(GL_TEXTURE_2D);         // disable tekstur
```

2. *NewGame*

Setelah *game* dibuka maka kita melakukan proses newgame (memainkan permainan baru). Berikut langkah-langkahnya:

- *NewGame* normal()

```
opengl.level = 'N';
OnMenuNew();
rec_file = "Nrecord.dat";

m_bar.SetPaneText(0,"Level: Normal(Susun 1 sisi Rubik
untuk menyelesaikan)");
level          =    'N';
```

- *NewGame hard()*

```
opengl.level = 'H';
OnMenuNew();
rec_file = "Hrecord.dat";

m_bar.SetPaneText(0,"Level: Hard(Susun 6 sisi Rubik
untuk menyelesaikan)");
level          =    'H';
```

- Prossedur OnNewGame() yang dijalankan oleh kedua prosedur diatas:

```
if(opengl.speed <10)
     opengl.speed = 25;
else
     opengl.speed = opengl.speed;

if(opengl.tekstur_ke <1)
     opengl.tekstur_ke = 1;
else
     opengl.tekstur_ke = opengl.tekstur_ke;

// Setting awal game
opengl.ResetKamera();
opengl.SetTekstur(opengl.tekstur_ke);
opengl.kol = 0;
opengl.klik_kiri= TRUE;
opengl.selesai- FALSE;
opengl.mouse_on = TRUE;
opengl.anim_show= FALSE;

for(int n=0;n<75;n++)
{
     // Acak kubus
     opengl.Acak();
     if(n < 25)
          m_bar.SetPaneText(0,"Tunggu Sebentar .");
     else if(n > 25 && n < 50)
          m_bar.SetPaneText(0,"Tunggu Sebentar ..");
     else if(n > 50)
          m_bar.SetPaneText(0,"Tunggu Sebentar ...");
}

// timer di 'nol' posisi awal game
timeku    =    0;

// Set variabel2 pd OnTimer <== (zero : zero : zero)
jam = mnt = dtk = 0;

// Mulai timer
SetTimer(1,1000,NULL);
```

- Prosedur Acak() :

```
// membuat angka random berdasar waktu
CTime wkt_skg = CTime::GetCurrentTime();

// Muter(kol,speed,arah,n_muter)
// Prosedur Acak() adalah menjalankan metode muter
// secara random sebanyak n kali

Muter(1,1,1,n_muter);
int bil_acak  =  wkt_skg.GetSecond() % 6;
int bil_acak2 =  wkt_skg.GetMinute() % 6;

if(bil_acak == 0)
     bil_acak = 6;

for(int n=0;n<bil_acak;n++)
{
     Muter(n,1,1,n_muter);

     for(int n2=0;n2<6;n2++)
     {
          Muter(n,1,-1,n_muter);
          Muter(n+2,1,-1,n_muter);
          Muter(n-1,1,-1,n_muter);
          Muter(n-3,1,-1,n_muter);
          Muter(n2+bil_acak,1,1,n_muter);
          Muter(n2+bil_acak2,1,1,n_muter);
          Muter(n+n2,-1,-1,n_muter);
     }
}
```

- Prosedur Muter() :

```
TMatriks m;

int deg       =    arahmuter * 90;
// konversi deg to rad
double rad    =    (deg * 3.141592657) / 180;
double sdt    =    rad / speed;

// sdt yg digunakan dlm cos & sin (rad) bkn (deg)
// PUTARAN POSITIF @ derajat arahe CCW
// PUTARAN NEGATIF @ derajat arahe CW (SearahJam)
// sdt) disini sdt = radian
// radian = (derajat*phi)/180

// Isi (awal) semua brs & kol matriks = 0
for(int x=0;x<3;x++)
{
     for(int y=0;y<3;y++)
     {
          m.brs_klm[x][y] = 0;
     }
}
```

```
// Rumus perkalian matriks disesuaikan Pada BaB
// LANDASAN TEORI
// brs & kol pd matriks diisi matriks rotasi

if(kol<3)
{
     //   Putaran CW sisi    *1 / x
     //   LEFT to RIGHT
     m.brs_klm[0][0]    = 1;
     m.brs_klm[1][1]    = cos(sdt);
     m.brs_klm[1][2]    = -sin(sdt);
     m.brs_klm[2][1]    = sin(sdt);
     m.brs_klm[2][2]    = cos(sdt);
}
else if(kol>2 && kol<6)
{
     //   Putaran CW sisi    *2 / y
     //   DOWN to UP
     m.brs_klm[0][0]    =cos(sdt);
     m.brs_klm[0][2]    =sin(sdt);
     m.brs_klm[1][1]    =1;
     m.brs_klm[2][0]    =-sin(sdt);
     m.brs_klm[2][2]    =cos(sdt);
}
else
{
     //   Putaran CW sisi    *3 / z
     //   BACK to FRONT
     m.brs_klm[0][0]    =cos(sdt);
     m.brs_klm[0][1]    =-sin(sdt);
     m.brs_klm[1][0]    =sin(sdt);
     m.brs_klm[1][1]    =cos(sdt);
     m.brs_klm[2][2]    =1;
}

// Setelah matriks di set operasi (perkalian matriks)
// dilakukan
for(int k=0;k<27;k++)
{
     //   *1   sisi kiri ke kanan      (sb - x)
     //   *2   sisi bawah ke atas      (sb - y)
     //   *3   sisi depan ke belakang  (sb - z)

     // ====================================
     // Muter CW & CCW sisi kiri      *1

     if(kol==0)
     {
          if (kubus[k].KoordinatPlus().x < -1)
               kubus[k].PindahKubus(m);
     }
```

```
      // Muter CW & CCW sisi kanan *1
      else if(kol==2)
      {
            if(kubus[k].KoordinatPlus().x > 1)
                  kubus[k].PindahKubus(m);

      // ====================================

      // Muter CW & CCW sisi bawah *2
      else if(kol==3)
      {
            if(kubus[k].KoordinatPlus().y < -1)
                  kubus[k].PindahKubus(m);
      }

      // Muter CW & CCW sisi tengah*2
      else if(kol==4)
      {
            if(kubus[k].KoordinatPlus().y > -1 &&
            kubus[k].KoordinatPlus().y < 1)
                  kubus[k].PindahKubus(m);
      }
      // Muter CW & CCW sisi atas  *2
      else if(kol==5)
      {
            if(kubus[k].KoordinatPlus().y > 1)
                  kubus[k].PindahKubus(m);
      }

      // ====================================

      // Muter CW & CCW sisi depan *3
      else if(kol==6)
      {
            if(kubus[k].KoordinatPlus().z < -1)
                  kubus[k].PindahKubus(m);
      }
      // Muter CW & CCW sisi tengah*3
      else if(kol==7)
      {
            if(kubus[k].KoordinatPlus().z > -1 &&
            kubus[k].KoordinatPlus().z < 1)
                  kubus[k].PindahKubus(m);
      }
      // Muter CW & CCW sisi belakang*3
      else if(kol==8)
      {
            if(kubus[k].KoordinatPlus().z > 1)
                  kubus[k].PindahKubus(m);
      }
}
```

- PindahKubus() :

```
sisi[0].PindahSisi(m);
sisi[1].PindahSisi(m);
sisi[2].PindahSisi(m);
sisi[3].PindahSisi(m);
sisi[4].PindahSisi(m);
sisi[5].PindahSisi(m);
```

- Pindah Sisi:

```
titik[0].DotMatriks(m);
titik[1].DotMatriks(m);
titik[2].DotMatriks(m);
titik[3].DotMatriks(m);
```

- DotMatriks:

```
float x_aks,y_aks,z_aks;

x_aks     =   m.brs_klm[0][0]*x + m.brs_klm[0][1]*y
              + m.brs_klm[0][2]*z;
y_aks     =   m.brs_klm[1][0]*x + m.brs_klm[1][1]*y
              + m.brs_klm[1][2]*z;
z_aks     =   m.brs_klm[2][0]*x + m.brs_klm[2][1]*y
              + m.brs_klm[2][2]*z;
```

4.2 Analisis Kinerja Sistem

Analisis kinerja pada sistem diperlukan guna mengetahui kemampuan interaksi antara sistem dengan pemakai. Analisis yang akan digunakan adalah Analisis Penerapan (uji penerapan kunci rubik apakah bisa juga diterapkan pada aplikasi sama dan sesuai dengan *rubik cube* yang asli) dan Analisis perbandingan (menganalisa dengan membandingkan dengan perangkat lunak sejenis).

4.2.1 Uji Penerapan Kunci Rubik Cube

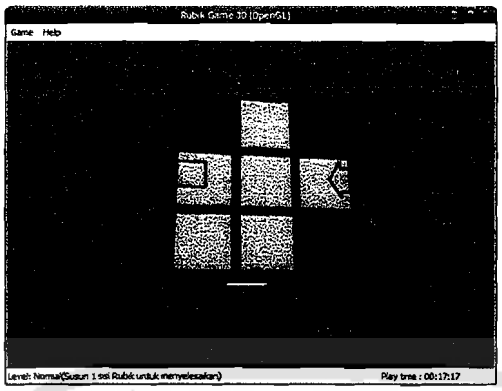
Pada proses ini penulis mencoba untuk menerapkan metode penyelesaian rubik cube pada level normal dan level *hard* (menggunakan metode solusi *beginner* Jasmine Lee)[2] guna mencoba apakah *game* sudah bisa digunakan dan diterapkan seperti *puzzle* rubik yang asli.[3]

1. Level Normal

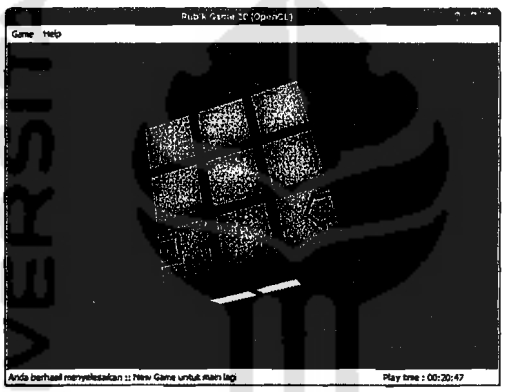
Pada level normal ini kita hanya diminta untuk menyelesaikan satu sisi warna dari *puzzle rubik*. Permainan di level ini selesai jika kita berhasil menyelesaikan salah satu sisi warna (merah, oranye, biru, hijau, kuning, maupun putih). Untuk menyelesaikannya pertama kita harus membuat pola plus pada sisi yang akan kita susun (gambar 4.10). Jika sudah maka baru kita susun bagian sudut-sudutnya satu persatu (gambar 4.11). Setelah semua sudut tersusun (gambar 4.12).

Gambar 4.10 Bentuk Plus

[2] Diambil dari http://www.geocities.com/jasmine_ellen/index.html.
[3] Untuk melihat metode penyelesaian secara lengkap lihat bagian lampiran laporan ini.



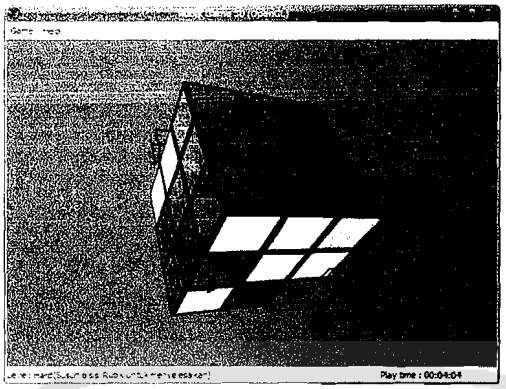
Gambar 4.11 Jadikan Bagian Pojok-Pojok



Gambar 4.12 Bentuk Jadi Mode Normal

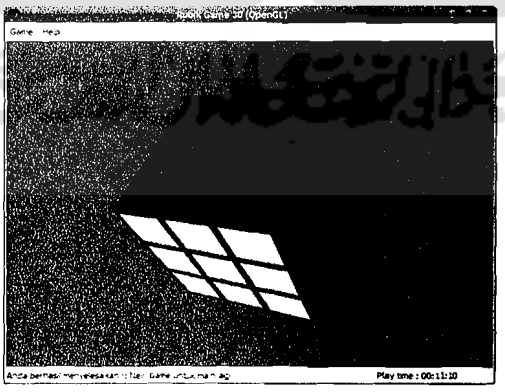
2. Level *Hard*

Pada level *hard* ini kita diminta untuk menyelesaikan seluruh sisi warna dari *puzzle rubik.* Permainan di level ini selesai jika kita berhasil menyelesaikan seluruh sisi warna (merah, oranye, biru, hijau, kuning, maupun putih). Untuk menyelesaikannya pertama kita harus menyusun salah satu lapisan rubik (gambar 4.13). Jika sudah maka kita susun lapisan kedua atau lapisan diatasnya (gambar 4.14), terakhir baru kita menyusun *layer* teratas (gambar 4.15).



Gambar 4.13 *Layer* Pertama Jadi

Gambar 4.14 *Layer* ke-Dua Jadi



Gambar 4.15 Bentuk Jadi Mode *Hard*

4.2.2 Analisis Perbandingan dengan *Game* Sejenis

Pada proses analisis perbandingan ini penulis menggunakan dua *game* yang sejenis dengan *game* yang dibuat pada Tugas akhir ini, diantaranya:

1. Rubiksim (Visual C++) lisensi *open source.*[4]
2. Rubik's Cube 2.3 (Borland Delphi) versi *trial.* [5]

Berikut tampilan dari kedua *game* diatas dapat dilihat pada gambar 4.16:

(a) (b)

Gambar 4.16 Tampilan *game* pembanding: (a) Rubiksim dan (b) Rubik's Cube

Berikut adalah perbandingan antara ketiga *game* yaitu NRubik (Tugas akhir), Rubiksim (*open source*) dan Rubik's Cube 2.3 (versi trial).

Tabel 4.1 Tabel Perbandingan Dengan *Game Sejenis*

Variabel Pembanding	**NRubik**	**Rubiksim**	**Rubik's Cube**
Waktu atau Skor	Ada	Tidak ada	Tidak ada
Save & Load Game	Tidak ada	Ada	Ada
Sound	Tidak ada	Ada	Tidak ada

[4] Diambil dari, http://sourceforge.net/projects/rubiksim. (c) 2003 *Licensed under* GPL.
[5] Diambil dari, http://necrosoft.ussr.to. (c) 2001 Necro$oft Inc.

Level	Ada (normal dan *hard*)	Tidak ada	Tidak ada
Set Tekstur	Ada	Tidak ada	Tidak ada
Set Kecepatan Putar	Ada	Ada	Tidak ada
Resize Jendela	Tidak bisa	Bisa	Bisa
Zoom	Tidak bisa	Bisa	Tidak bisa

Pada tabel 4.1 diatas dapat dilihat bahwa perangkat lunak yang dibuat (NRubik) memiliki keunggulan namun juga masih memiliki beberapa kekurangan.

1. Kelebihan sistem

Kelebihan-kelebihan dari sistem ini yang dapat dijadikan sebagai ciri khas antara lain:

- Sistem yang dibangun memiliki dua mode *level* (normal dan *hard*). Sehingga pengguna bisa memilih salah satu diantara keduanya, tergantung kemampuan.
- Sistem memiliki *timer* (penghitung waktu), sehingga pemain dapat termotivasi untuk memperoleh catatan waktu yang lebih baik dari rekor waktu sudah ada.
- Sistem menyediakan tiga macam tekstur yang berlainan (bagi rubik), sebagai alternatif tampilan jika pemain menginginkan suasana lain.

2. Kekurangan sistem

Sedangkan sistem ini juga memiliki berbagai kekurangan jika dibandingkan dengan dua *game* pembanding lainnya, antara lain:

- Sistem yang dibangun tidak memiliki fasilitas *save* dan *load* game sehingga permainan tidak bisa dilanjutkan di kesempatan berbeda.

- Jendela utama pada sistem tidak bisa di *resize* sehingga ukuran jendela (*window*) statis.
- Besar kecil ukuran rubik statis (tanpa *zoom*).

4.2.3 Analisis Pengguna

Analisis pengguna digunakan sebagai tolak ukur keberhasilan pembuatan program dinilai dari sisi pengguna (*user*). Penilaian menggunakan kuisioner dengan memberikan nilai rata-rata pada tiap kuisioner dengan kriteria sebagai berikut:

Nilai 1 = buruk

Nilai 2 = kurang baik

Nilai 3 = sedang

Nilai 4 = baik

Nilai 5 = sangat baik

Nilai rata-rata yang digunakan untuk menghitung nilai dari tiap-tiap pertanyaan kuisioner dihitung dengan rumus:

$$\textbf{Rata-rata} = \frac{\textbf{jumlah nilai jawaban}}{\textbf{jumah responden}}$$

Berikut hasil dari kuisioner yang dibagikan kepada 9 orang responden yang berusia 20 sampai 26 tahun.

Tabel 4.2 Tabel Hasil Kuisioner Responden

No	Pertanyaan	Pemilih Memilih (nilai 1 - 5) sebanyak n orang				
		1	2	3	4	5
1.	Bagaimana tampilan pada *game* ini?			2	7	
2.	Bagaimana kemudahan operasional (menu) pada *game* ini ?		1	4	4	
3.	Bagaimana pengendalian (kontrol) pada game rubik ini ?		1	4	2	2
4.	Bagaimanakah kesesuaian *game* ini dengan permainan asli nya ?		1	1	6	1
5.	Apakah anda terhibur dengan bermain *game* ini ?		1		8	

Dari data diatas dapat dilihat bahwa pada pertanyaan nomor 1 memiliki rata-rata nilai **3.8**. Pertanyaan nomor 2 memiliki rata – rata nilai **3.3**. Pertanyaan nomor 3 memiliki rata – rata nilai **3.5**. Pertanyaan nomor 4 memiliki rata – rata nilai **4.7**. Pertanyaan nomor 5 memiliki rata – rata nilai **3.8**. Dari hasil penilaian di atas dapat diambil hasil atas penilaian pengguna terhadap aplikasi ini yaitu:

- Tampilan untuk game ini sedang cenderung mendekati baik.
- Kemudahan operasional (menu) pada *game* ini lebih cenderung sedang (biasa).
- Pengendalian *game* ini sedang dan cukup baik.
- Kesesuaian game ini dengan game versi aslinya baik mendekati sangat baik.
- Perasaan terhibur pemain memiliki nilai sedang cenderung baik.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Setelah dilakukan serangkaian pengujian terhadap aplikasi *game* Nrubik (*puzzle* rubik cube) ini maka dapat diberikan beberapa kesimpulan, antara lain:

1. Aplikasi *game* **Nrubik** (*puzzle rubik cube*) ini dapat digunakan sebagai alternatif permainan yang dapat dimainkan pada sebuah komputer personal.
2. Penggunaan OpenGL sebagai media (pendukung) dalam pemrograman ini sudah tepat karena kita menginginkan sebuah aplikasi *game* yang bisa ditampilkan secara tiga dimensi.
3. Menambah kepopuleran permainan *puzzle rubik cube* dikalangan pecinta *game* komputer.

5.2 Saran

Setelah melihat hasil yang dicapai dalam tugas akhir ini, maka terdapat beberapa saran ataupun ide yang dapat kami kemukakan, antara lain:

1. Menambahkan fitur jaringan pada *game* yang akan dibuat (bermain dan bertanding secara *multi player*) untuk berlomba dalam kecepatan penyusunan sisi-sisinya, sehingga menambahkan unsur ketegangan dalam bermain.
2. Dimungkinkan untuk bermain melawan komputer yang menggunakan AI.

3. Dikemudian hari dimungkinkan untuk membuat aplikasi sejenis yang menekankan pada unsur AI nya (misal bagaimana menyelesaikan seluruh sisi rubik dengan metode AI).

DAFTAR PUSTAKA

BUKU TEKS

[ANG05] Angel, Edward. *OpenGL A Primer,* 2nd ed. Pearson Education, Inc. USA, 2005.

[KAD04] Kadir, Abdul. *Panduan Pemrograman Visual C++.* Yogyakarta: Penerbit ANDI, 2004.

[NUG05] Nugroho, Adi. 2005. *Analisis dan Perancangan Sistem Informasi dengan Metodologi Berorientasi Objek.* Bandung: Penerbit Informatika.

[SUY03] Suyoto. *Teori Dan Pemrograman Grafika Komputer dengan VisualC++ V.6 dan OpenGL.* Yogyakarta: Gava Media, 2003.

[WHI04] Whitten, L. J., *et. al. Metode Desain dan Analisis Sistem.* Alih Bahasa: Tim Penerjemah Penerbit Andi. Yogyakarta: Penerbit Andi, 2004.

WEB SITE

[ANO08a] Anonym. *Rubik's History.* http://www.rubiks.com/World/Rubiks history.aspx, diakses 2 Maret 2008.

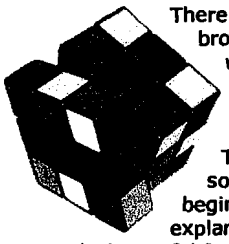
[ANO08b] Anonym. *Computer and Video Game Software Sales Reach Record $7.3 Billion in 2004.* http://www.theesa.com/archives/2005/02/computer_ and_vi.php, diakses 3 Maret 2008.

[ANO08c] Anonym. *Computer and Video Game Industry Reaches $18.85 Billion in 2007.* http://www.theesa.com/archives/2008/01/computer_and_vi_1.php, diakses 3 Maret 2008.

[LIO08] Lionheart, Robin. *Genre and the Video Game.* http://www.robinlionheart.com/genres.xhtml, diakses 3 Maret 2008.

Beginner Solution to the Rubik's Cube

Introduction



There are many different methods for solving the Rubik's cube. They can be divided into two broad categories - layer methods and corners first methods (and there are sub-categories within these broad categories). The method I use for speedsolving is a layer based method. More specifically, the method I currently use is: cross, F2L, 3-look LL. If you are a newbie cuber then this description may not mean much to you, so I should add that it's the 'Advanced Solution' I described in the Next Steps section at the end of this page.

There are many great websites that explain intermediate, advanced and expert methods for solving the cube (check out my Rubiks links page). However, there are very few that explain beginner methods, which is why I wrote this. It's not meant to be a totally comprehensive explanation, it's really just some notes I threw together for some friends I was teaching. I thought it might be useful for others, so I've turned it into a webpage.

This beginner method requires memorising only a few algorithms, and when done efficiently can achieve solves of 60 seconds or faster. I know people who can solve in 30s with a method like this. I, personally, have not done 30s with this method though, so don't be too distressed if you can't either. On the other hand, if you can do 30s solves with this method, then you are too good for this method and you should be learning an Advanced method!

Another benefit of this method is that it is fairly scalable, so more algorithms may be added later to develop it into an advanced method, or if you're really keen, an expert method.

Structure of the cube

We all know that 3x3x3=27, however, rather than thinking about the cube as 27 little "cubies", think about it as 6 fixed centres (that can rotate on their own axis) with 8 corners and 12 edges which rotate around it. As the centres are fixed, the centre colour defines the colour for the face. It's important to remember this otherwise you'll end up trying to do illogical (mechanically impossible!) things like wondering why you can't work out how to put a corner piece in an edge position, or assuming that you're looking at the blue face merely because 8 of the 9 cubies on it are blue (if the centre is white then it's the white face).

Terminology

When describing the solution for the 2nd and 3rd layers, standard cube notation will be used. Here's what you need to know to read it:

F = front face **B** = back face **R** = right face **L** = left face **U** = up face **D** = down face

In addition to a letter, each move may be accompanied by an apostrophe or the number two:
--> A letter by itself means turn that face 90 degrees clockwise (eg. **F**).
--> A letter followed by an apostrophe means turn that face 90 degrees anti-clockwise (eg. **F'**).
--> A letter followed by the number 2 means turn that face 180 degrees (direction is irrelevant), (eg. **F2**).

So **R U' L2** is shorthand for *"turn the right face 90 degrees clockwise, then turn the up face 90 degrees anti-clockwise, then turn the left face 180 degrees"*. When thinking whether to turn clockwise/anti-clockwise, imagine that you are looking directly at the particular face you are turning.

For each algorithm, the notation is written with the assumption that the core of the cube remains fixed throughout the whole algorithm, and the faces just turn around it. This means that you also need to know how to position the cube to start the algorithm.

For pictures and further detail about cube notation, have a look at Jon Morris' cube notation page.

The Solution

The First Layer

The first layer is solved in two stages:
(1) Form the cross
(2) Insert the 4 first layer corners (each corner is inserted individually)

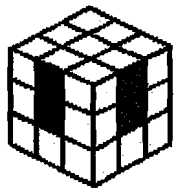
I believe that the first layer should be done intuitively. You need to understand it and solve it without learning algorithms. Until you can do this, I wouldn't bother attempting the rest of the cube! So, spend some time playing with the cube and familiarising yourself with how to move the pieces around the cube.

Now, here are some tips to get you started.

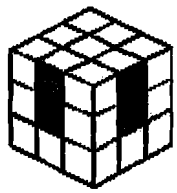
The Cross

I prefer to start with the white cross because I find white easier to quickly identify on a completely scrambled cube, however, you can use any colour.

There are 4 edge pieces with white (ie. the 4 arms of the cross) which have specific positions. You can't put any white edge piece in an arm of the cross because the other colour on the edge cubie must match up with it's centre on the middle layer.



Here is a pic of what a correctly formed cross looks like (grey denotes cubies that are irrelevant to the cross). Note that the **white/red** edge cubie matches up with the **white** centre and the **red** centre. Ditto re the **white**/blue cubie.



Here's a pic on an incorrectly formed cross. Looking at the **white** face we do indeed see a **white** cross, however the **white/red** edge cubie does not match up with the red centre. Ditto re the **white**/blue cubie. This is bad!

For a detailed explanation of the cross, check out Dan Harris' Solving the Cross page.

The First Layer Corners

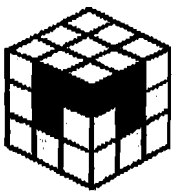
Once you have completed the cross, completing the first layer requires inserting each of the 4 corners in separately. The first thing to do is examine your cube and locate all of the top layer edge pieces - they will be sitting in either the top layer or the bottom layer. Inserting the top layer corners should be done intuitively, not by learning algorithms. To get you started, here's a step-by-step example of one way to insert a top layer corner.

Step 1
The blue/**red**/**white** corner is sitting in the bottom layer (the blue part is facing the bottom so we can't see it in this picture). Turn the blue face 90 degrees anticlockwise.

Step 2
Now your cube should look like this. Move the D face 90 degrees anti-clockwise to line up the blue/**white** edge with the blue/**white**/ **red** corner.

Step 3
Now that the blue/**white** edge and the blue/**white**/ **red** corner have been lined up, re-form the **white** cross by turning the blue face 90 degrees clockwise.

Step 4
Now the blue/**white**/**red** corner is in its correct place.



Here are some tips for inserting the top layer corners:

- Start with a top layer corner that is sitting in the bottom layer.
- If there are multiple top layer corners in the bottom layer (there usually will be), start with one that does not have thewhite part of the corner on the D face. Or, if you were using a different colour for the cross ('colour X'), start with a corner that does not have the 'colour X' part of the corner on the D face.
- When working with a top layer corner piece that is in the top layer (but in the wrong top layer corner position), you will need to get it out of the top layer into the bottom layer, then insert it into the correct top layer corner position. The same principle applies if a top layer corner piece is in the correct top layer corner position but needs to be flipped around. You need to get it out of the top layer (ie. into the bottom layer), and then re-insert it into the top layer the correct way around.

This is what the first layer should look like when finished.

The Middle Layer

The middle layer consists of one stage:

(1) Insert the 4 middle layer edges (each edge is inserted individually)

You only need to learn one algorithm (plus the mirror algorithm) for the second layer. There are many more algs, but let's just learn the essential one first.
First, locate a middle layer edge that is currently sitting in the last layer. I'm going to use the blue/red edge for this example.

This blue edge cubie in the last layer is the blue/red edge cubie.

In this picture, **U=white, L=red and F=**blue. We can't see the other three faces, but obviously the R face is the one opposite the L face, the D face is opposite the U face and the B face is opposite the F face.

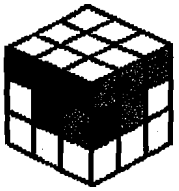
Now, position the blue/red edge piece so that the colour on the side of the cube (blue in this case) lines up with it's centre. Now perform the following algorithm: **D L D' L' D' F' D F**

If the blue/red edge piece was flipped the other way so that the blue was on the bottom rather than the **red**, you wouldposition the cubie under the red centre and perform the following alg: **D' F' D F D L D' L'**. This is the mirror of the previous algorithm. The axis of symmetry lies diagonally across the white face, and along the line which divides the blue face and the red face.

What if the edge piece is not in the last layer?

The instructions above assume that the middle layer edge piece you are inserting is sitting somewhere in the last layer. If some middle edges are in the last layer and some are in the middle layer in the wrong spot, always start working with the edge pieces that are in the last layer.
After you've done this, sometimes (but not too often) you'll be left with a middle layer edge piece that's in the middle layer but in the wrong spot. In this situation, you can use the same middle layer algorithms from above (**D L D' L' D' F' D F** or **D' F' D F D L D' L'**) to insert another edge piece into the middle layer edge position, thereby knocking the middle layer edge piece out of its spot and into the last layer. Once you've done this, the middle layer edge piece is in the last layer and you can deal with it in the usual way.

The red/blue middle layer edge piece is in the middle layer but not oriented correctly. It needs to be moved to the last layer, then put back into the middle layer in the right way.



The Last Layer

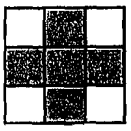
The last layer ("LL") is done in 4 steps:
(1) Orient the edges (2 algs) - ie. form a cross on the D face
(2) Permute the corners (1 alg) - ie. get the corners in the correct position in 3D space (don't worry if they still need to be rotated)
(3) Orient the corners (1 alg + mirror alg) - ie. flip the corners
(4) Permute the edges (1 alg) - ie. swap the edges around. The cube should now be solved! :)

All last layer algorithms are performed with the cross (i.e the first layer - white side in this example) on the bottom.

Orienting the LL Edges

Once you've completed the first two layers ("F2L"), hold the cube so that the white side is on the bottom. The white side will be on the bottom for the remainder of the solution. This means that the white side is the D side for all last layer algorithms.

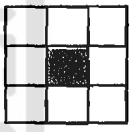
On my cube, white is opposite yellow, therefore yellow is the U face for all last layer algorithms on my cube. Note that your cube may have a different colour opposite white (eg. blue). Now have a look at your last layer, and in particular, look at the last layer face -- there are 4 possible patterns of LL edges that you may see.



State 1

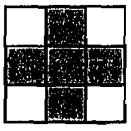
State 2

State 3

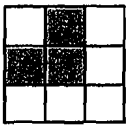


State 4

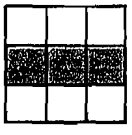
Unlike with the initial cross (where all the edges must match up with the white centre and with the centres on the middle layer), here all you need to worry about is getting all the last layer edges matching up with the last layer centre. It doesn't matter if the other colour on the LL edge piece does not match up with the colour on the middle layer centre. Also, ignore the LL corners too. It doesn't matter what they are doing at the moment. Now, let's consider each of these LL edge states separately.



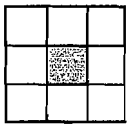
State 1
All the edges are already oriented correctly. Move on to permuting the corners.



State 2
We are going to re-orient our faces for this algorithm. The face you are looking directly at in this picture is now the U face (it was the D face for when you were doing the second layer edges). Perform the following algorithm: **F U R U' R' F'**



State 3
As with State 2, the face you are looking directly at in this picture is now the U face. Perform the following algorithm: **F R U R' U' F'**



State 4
State 4 is really a combination of States 2 and 3, so all you need to do is perform the algorithm for either State 2 or State 3. Once you've done this, you'll see that your LL edges now look like State 2 or State 3, so just perform the appropriate algorithm and you will have a cross on the LL.

Permuting the LL Corners

The two possible states are: (i) two adjacent LL corners need to be swapped; or (ii) two diagonal LL corners need to be swapped. These are the only two possible states. If you cannot identify one of these two states with your LL corners then one of the following must be true (i) you have not finished the F2L; (ii) someone has ripped out a corner of your cube and put it in the wrong way; (iii) someone has ripped off some of your stickers and put them back in the wrong place; or (iv) you are not looking hard enough. ;)

Swapping adjacent corners

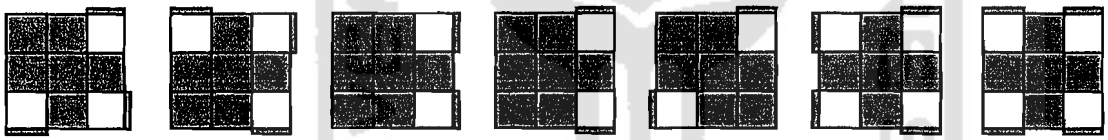
Hold the cube with the white side on the bottom, and the two corners to be swapped are in the front right top and the back right top positions. Perform the following algorithm: **L U' R' U L' U' R U2**

Swapping diagonal corners

Swapping diagonal corners can be done by executing the adjacent corner swap algorithm twice. Perform it once to swap any two LL corners. Re-examine you cube and you'll see that now there are just two LL corners that need to be swapped. Position it correctly for the final LL adjacent corner swap and perform the LL adjacent corner swap algorithm.

Orienting the LL Corners

There are 8 possible orientation states for the LL corners. One is where all 4 corners are correctly oriented. The other 7 look like this.



State 1 **State 2** **State 3** **State 4** **State 5** **State 6** **State 7**

State 1. Twisting three corners anti-clockwise

R' U' R U' R' U2 R U2

State 2. Twisting three corners clockwise

R U R' U R U2 R' U2

States 3-7

Once you know the algorithms for States 1 and 2, you can solve any LL orientation State. The remaining States can be orientedusing a maximum of 2 algorithms. You will need to do one of the following (i) the State 1 algorithm twice, (ii) the State 2 algorithm twice, (iii) the State 1 algorithm, then the State 2 algorithm, or (iv) the State 2 algorithm, then the State 1 algorithm.

I'm not going to tell you which of these options should be used for which remaining State because it's important that you try to understand how the State 1 and the State 2 algorithms work. Once you understand these algorithms you will be able to work out how to use them to solve all the States.

Permuting the LL Edges

There are 4 possible states permutation states for the LL edges. One is where all 4 edges are correctly permuted. The other 4 look like this.

State 1

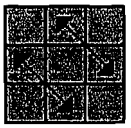
R2 U F B' R2 F' B U R2

State 2

R2 U' F B' R2 F' B U' R2

This is almost identical to the algorithm for State 1.
Only difference is the 2nd move and the 2nd last move.

State 3
Apply the algorithm for either State 1 or State 2. Reexamine your cube and it will now look like State 1 or State 2.



State 4
Apply the algorithm for either State 1 or State 2. Reexamine your cube and it will now look like State 1 or State 2.

And that's all you really need to know to solve the Rubik's Cube! With practice, you should eventually be able to achieve times of 60 seconds (or faster) using this method. Once your comfortable with this method and want to learn more, take a look at the following section.

KUISIONER

NRubik adalah Tugas Akhir yang berupa sebuah *game* (permainan) pada sebuah komputer yang mengadaptasi permainan (toy) rubik cube. Guna menganalisis hasil kinerja *game* ini maka dibutuhkan penilaian kinerja *game* ini dari sisi pengguna.

Petunjuk pengisian :

1. Responden diharapkan mengisi dengan jujur.
2. Jawaban jangan dikosongi (tidak diisi).
3. Berilah tanda centang (√) untuk jawaban yang menurut anda paling sesuai.

No	Pertanyaan	Nilai				
		1	2	3	4	5
1.	Bagaimana tampilan pada *game* ini?				✓	
2.	Bagaimana kemudahan operasional (menu) pada *game* ini ?				✓	
3.	Bagaimana pengendalian (kontrol) pada game rubik ini ?				✓	
4.	Bagaimanakah kesesuaian *game* ini dengan permainan asli nya ?				✓	
5.	Apakah anda terhibur dengan bermain *game* ini ?				✓	

Berapa usia anda ? (22 tahun)
Pernahkah anda bermain *puzzle* rubik cube ? (Ya / ~~Tidak~~)*
Apakah anda hobi bermain video *game* ? (Ya / ~~Tidak~~)*

**coret salah satu*

Kritik dan Saran:

_____

_____

Keterangan penilaian :

Nilai **1** = buruk
Nilai **2** = kurang baik
Nilai **3** = sedang
Nilai **4** = baik
Nilai **5** = sangat baik

KUISIONER

NRubik adalah Tugas Akhir yang berupa sebuah *game* (permainan) pada sebuah komputer yang mengadaptasi permainan (toy) rubik cube. Guna menganalisis hasil kinerja *game* ini maka dibutuhkan penilaian kinerja *game* ini dari sisi pengguna.

Petunjuk pengisian :

1. Responden diharapkan mengisi dengan jujur.
2. Jawaban jangan dikosongi (tidak diisi).
3. Berilah tanda centang (√) untuk jawaban yang menurut anda paling sesuai.

No	Pertanyaan	Nilai				
		1	2	3	4	5
1.	Bagaimana tampilan pada *game* ini?				✓	
2.	Bagaimana kemudahan operasional (menu) pada *game* ini ?				✓	
3.	Bagaimana pengendalian (kontrol) pada game rubik ini ?					✓
4.	Bagaimanakah kesesuaian *game* ini dengan permainan asli nya ?					✓
5.	Apakah anda terhibur dengan bermain *game* ini ?				✓	

Berapa usia anda ? (22... tahun)
Pernahkah anda bermain *puzzle* rubik cube ? (Ya / ~~Tidak~~)*
Apakah anda hobi bermain video *game* ? (Ya / ~~Tidak~~)*

**coret salah satu*

Kritik dan Saran:
lebih baik lagi apabila tiap sisi/muka. bisa diganti gambar/foto dengan begitu lebih menarik lagi

Keterangan penilaian :

Nilai **1** = buruk
Nilai **2** = kurang baik
Nilai **3** = sedang
Nilai **4** = baik
Nilai **5** = sangat baik

KUISIONER

NRubik adalah Tugas Akhir yang berupa sebuah *game* (permainan) pada sebuah komputer yang mengadaptasi permainan (toy) rubik cube. Guna menganalisis hasil kinerja *game* ini maka dibutuhkan penilaian kinerja *game* ini dari sisi pengguna.

Petunjuk pengisian :

1. Responden diharapkan mengisi dengan jujur.
2. Jawaban jangan dikosongi (tidak diisi).
3. Berilah tanda centang (√) untuk jawaban yang menurut anda paling sesuai.

No	Pertanyaan	Nilai				
		1	2	3	4	5
1.	Bagaimana tampilan pada *game* ini?				✓	
2.	Bagaimana kemudahan operasional (menu) pada *game* ini ?			✓		
3.	Bagaimana pengendalian (kontrol) pada game rubik ini ?				✓	
4.	Bagaimanakah kesesuaian *game* ini dengan permainan asli nya ?				✓	
5.	Apakah anda terhibur dengan bermain *game* ini ?				✓	

Berapa usia anda ? (21 tahun)
Pernahkah anda bermain *puzzle* rubik cube ? (Ya / ~~Tidak~~)*
Apakah anda hobi bermain video *game* ? (Ya / ~~Tidak~~)*

**coret salah satu*

Kritik dan Saran:
Buatlah Indonesia maju dalam bidang anda.

Keterangan penilaian :

Nilai **1** = **buruk**
Nilai **2** = **kurang baik**
Nilai **3** = **sedang**
Nilai **4** = **baik**
Nilai **5** = **sangat baik**

KUISIONER

NRubik adalah Tugas Akhir yang berupa sebuah *game* (permainan) pada sebuah komputer yang mengadaptasi permainan (toy) rubik cube. Guna menganalisis hasil kinerja *game* ini maka dibutuhkan penilaian kinerja *game* ini dari sisi pengguna.

Petunjuk pengisian :

1. Responden diharapkan mengisi dengan jujur.
2. Jawaban jangan dikosongi (tidak diisi).
3. Berilah tanda centang (√) untuk jawaban yang menurut anda paling sesuai.

No	Pertanyaan	Nilai				
		1	2	3	4	5
1.	Bagaimana tampilan pada *game* ini?				✓	
2.	Bagaimana kemudahan operasional (menu) pada *game* ini ?				✓	
3.	Bagaimana pengendalian (kontrol) pada game rubik ini ?			✓		
4.	Bagaimanakah kesesuaian *game* ini dengan permainan asli nya ?				✓	
5.	Apakah anda terhibur dengan bermain *game* ini ?				✓	

Berapa usia anda ? (21 tahun)
Pernahkah anda bermain *puzzle* rubik cube ? (Ya / ~~Tidak~~)*
Apakah anda hobi bermain video *game* ? (Ya / ~~Tidak~~)*

**coret salah satu*

Kritik dan Saran:
pengendalian pada game rubik ini kalau bisa di bikin lebih mudah lagi.

Keterangan penilaian :

Nilai **1** = buruk
Nilai **2** = kurang baik
Nilai **3** = sedang
Nilai **4** = baik
Nilai **5** = sangat baik

KUISIONER

NRubik adalah Tugas Akhir yang berupa sebuah *game* (permainan) pada sebuah komputer yang mengadaptasi permainan (toy) rubik cube. Guna menganalisis hasil kinerja *game* ini maka dibutuhkan penilaian kinerja *game* ini dari sisi pengguna.

Petunjuk pengisian :

1. Responden diharapkan mengisi dengan jujur.
2. Jawaban jangan dikosongi (tidak diisi).
3. Berilah tanda centang (√) untuk jawaban yang menurut anda paling sesuai.

No	Pertanyaan	Nilai				
		1	2	3	4	5
1.	Bagaimana tampilan pada *game* ini?				✓	
2.	Bagaimana kemudahan operasional (menu) pada *game* ini ?			✓		
3.	Bagaimana pengendalian (kontrol) pada game rubik ini ?			✓		
4.	Bagaimanakah kesesuaian *game* ini dengan permainan asli nya ?				✓	
5.	Apakah anda terhibur dengan bermain *game* ini ?				✓	

Berapa usia anda ? (~~2.0~~. tahun)
Pernahkah anda bermain *puzzle* rubik cube ? (Ya / ~~Tidak~~)*
Apakah anda hobi bermain video *game* ? (Ya / ~~Tidak~~)*

**coret salah satu*

Kritik dan Saran:
perlu di tingkatkan lagi cara penkertan
warna agar beda dengan game lainnya.

Keterangan penilaian :

Nilai **1** = buruk
Nilai **2** = kurang baik
Nilai **3** = sedang
Nilai **4** = baik
Nilai **5** = sangat baik

KUISIONER

NRubik adalah Tugas Akhir yang berupa sebuah *game* (permainan) pada sebuah komputer yang mengadaptasi permainan (toy) rubik cube. Guna menganalisis hasil kinerja *game* ini maka dibutuhkan penilaian kinerja *game* ini dari sisi pengguna.

Petunjuk pengisian :

1. Responden diharapkan mengisi dengan jujur.
2. Jawaban jangan dikosongi (tidak diisi).
3. Berilah tanda centang (√) untuk jawaban yang menurut anda paling sesuai.

No	Pertanyaan	Nilai				
		1	2	3	4	5
1.	Bagaimana tampilan pada *game* ini?			√		
2.	Bagaimana kemudahan operasional (menu) pada *game* ini ?		√			
3.	Bagaimana pengendalian (kontrol) pada game rubik ini ?			√		
4.	Bagaimanakah kesesuaian *game* ini dengan permainan asli nya ?			√		
5.	Apakah anda terhibur dengan bermain *game* ini ?				√	

Berapa usia anda ? (23 tahun)
Pernahkah anda bermain *puzzle* rubik cube ? (Ya / ~~Tidak~~)*
Apakah anda hobi bermain video *game* ? (Ya / ~~Tidak~~)*

**coret salah satu*

Kritik dan Saran:

_____

_____

Keterangan penilaian :

Nilai **1** = **buruk**
Nilai **2** = **kurang baik**
Nilai **3** = **sedang**
Nilai **4** = **baik**
Nilai **5** = **sangat baik**

KUISIONER

NRubik adalah Tugas Akhir yang berupa sebuah *game* (permainan) pada sebuah komputer yang mengadaptasi permainan (toy) rubik cube. Guna menganalisis hasil kinerja *game* ini maka dibutuhkan penilaian kinerja *game* ini dari sisi pengguna.

Petunjuk pengisian :

1. Responden diharapkan mengisi dengan jujur.
2. Jawaban jangan dikosongi (tidak diisi).
3. Berilah tanda centang (√) untuk jawaban yang menurut anda paling sesuai.

No	Pertanyaan	Nilai				
		1	2	3	4	5
1.	Bagaimana tampilan pada *game* ini?				✓	
2.	Bagaimana kemudahan operasional (menu) pada *game* ini ?			✓		
3.	Bagaimana pengendalian (kontrol) pada game rubik ini ?			✓		
4.	Bagaimanakah kesesuaian *game* ini dengan permainan asli nya ?				✓	
5.	Apakah anda terhibur dengan bermain *game* ini ?				✓	

Berapa usia anda ? (21 ... tahun)
Pernahkah anda bermain *puzzle* rubik cube ? (Ya / ~~Tidak~~)*
Apakah anda hobi bermain video *game* ? (Ya / ~~Tidak~~)*

**coret salah satu*

Kritik dan Saran:

_____

_____

Keterangan penilaian :

Nilai **1** = buruk
Nilai **2** = kurang baik
Nilai **3** = sedang
Nilai **4** = baik
Nilai **5** = sangat baik

KUISIONER

NRubik adalah Tugas Akhir yang berupa sebuah *game* (permainan) pada sebuah komputer yang mengadaptasi permainan (toy) rubik cube. Guna menganalisis hasil kinerja *game* ini maka dibutuhkan penilaian kinerja *game* ini dari sisi pengguna.

Petunjuk pengisian :

1. Responden diharapkan mengisi dengan jujur.
2. Jawaban jangan dikosongi (tidak diisi).
3. Berilah tanda centang (√) untuk jawaban yang menurut anda paling sesuai.

No	Pertanyaan	Nilai				
		1	2	3	4	5
1.	Bagaimana tampilan pada *game* ini?			✓		
2.	Bagaimana kemudahan operasional (menu) pada *game* ini ?			✓		
3.	Bagaimana pengendalian (kontrol) pada game rubik ini ?		✓			
4.	Bagaimanakah kesesuaian *game* ini dengan permainan asli nya ?		✓			
5.	Apakah anda terhibur dengan bermain *game* ini ?		✓			

Berapa usia anda ? (24 ..... tahun)
Pernahkah anda bermain *puzzle* rubik cube ? (Ya / ~~Tidak~~)*
Apakah anda hobi bermain video *game* ? (Ya / ~~Tidak~~)*

**coret salah satu*

Kritik dan Saran:
- Bisa dibuat layar penuh (full screen).
-

Keterangan penilaian :

Nilai **1** = buruk
Nilai **2** = kurang baik
Nilai **3** = sedang
Nilai **4** = baik
Nilai **5** = sangat baik

KUISIONER

NRubik adalah Tugas Akhir yang berupa sebuah *game* (permainan) pada sebuah komputer yang mengadaptasi permainan (toy) rubik cube. Guna menganalisis hasil kinerja *game* ini maka dibutuhkan penilaian kinerja *game* ini dari sisi pengguna.

Petunjuk pengisian :

1. Responden diharapkan mengisi dengan jujur.
2. Jawaban jangan dikosongi (tidak diisi).
3. Berilah tanda centang (√) untuk jawaban yang menurut anda paling sesuai.

No	Pertanyaan	Nilai				
		1	2	3	4	5
1.	Bagaimana tampilan pada *game* ini?				✓	
2.	Bagaimana kemudahan operasional (menu) pada *game* ini ?				✓	
3.	Bagaimana pengendalian (kontrol) pada game rubik ini ?					✓
4.	Bagaimanakah kesesuaian *game* ini dengan permainan asli nya ?				✓	
5.	Apakah anda terhibur dengan bermain *game* ini ?				✓	

Berapa usia anda ? (20 tahun)
Pernahkah anda bermain *puzzle* rubik cube ? (Ya / ~~Tidak~~)*
Apakah anda hobi bermain video *game* ? (Ya / ~~Tidak~~)*

**coret salah satu*

Kritik dan Saran:
Game ini sebaiknya levelnya di sesuaikan dg umur yg main game

Keterangan penilaian :

Nilai **1** = buruk
Nilai **2** = kurang baik
Nilai **3** = sedang
Nilai **4** = baik
Nilai **5** = sangat baik