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.*

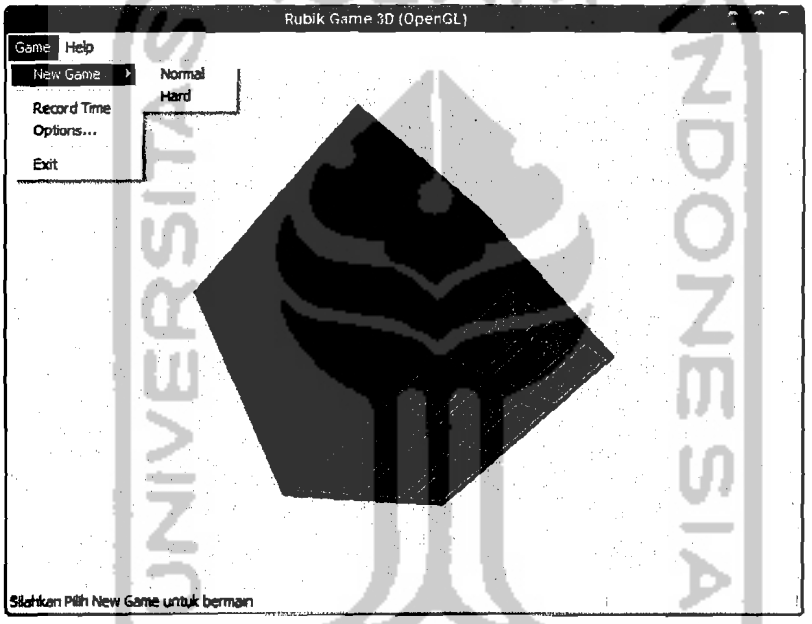
Implementasi software yang digunakan dalam membuat, mengkompilasi hingga menjalankan perangkat lunak ini adalah:

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

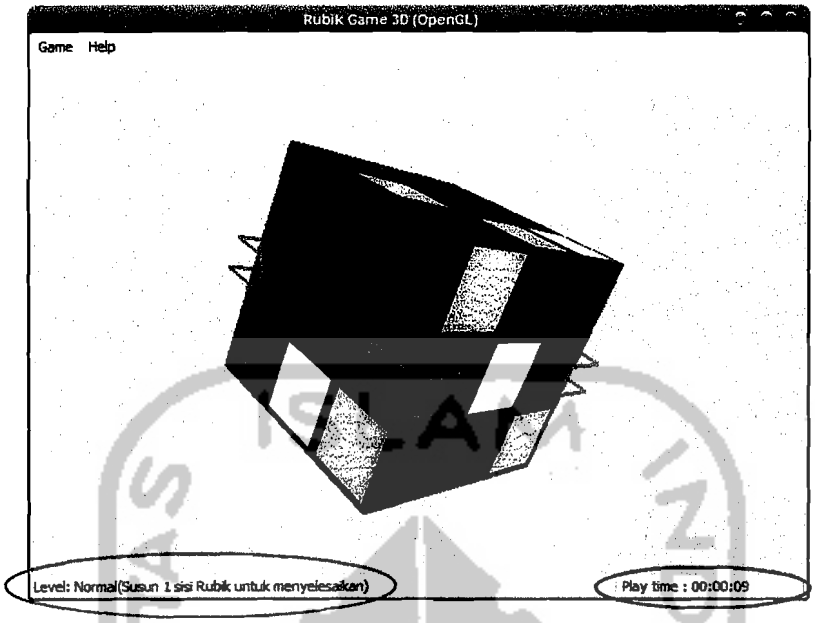
Jendela ini merupakan jendela utama atau tampilan awal ketika kita membuka aplikasi *game* ini. Pada tahap ini rubik akan memutar secara otomatis (animasi) selama kita belum memulai game (memilih submenu *NewGame* Normal atau submenu *NewGame Hard*).



Gambar 4.1 Form Menu Awal

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

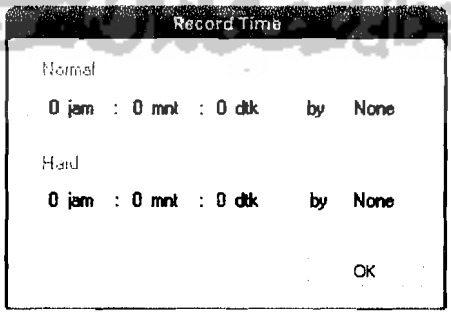
Jika pilihan pada salah satu submenu *NewGame* sudah dipilih maka tampilan akan seperti Gambar 4.2 dibawah, akan muncul informasi level yang kita pilih (lingkaran kiri bawah) level Normal atau *Hard*, dan juga akan muncul informasi waktu lama permainan (lingkaran kanan bawah) sampai kita bisa menyelesaikan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



Gambar 4.2 *NewGame*

3. RecordTime

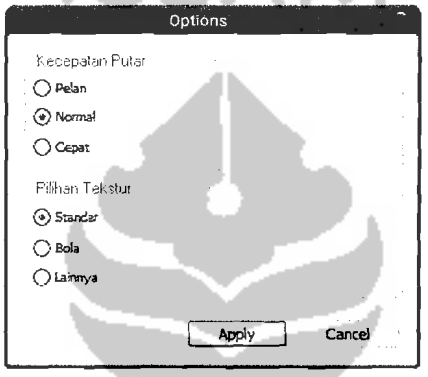
Form *Record Time* menampilkan menu waktu rekor (*record time*). Form ini menampilkan form *record time* seperti pada Gambar 4.3 berikut. Jika kita bisa melampaui rekor yang tercatat pada menu dibawah maka nama dan waktu (penyelesaian) kita yang akan menggantikan pemilik rekor waktu sebelumnya.



Gambar 4.3 Form *RecordTime*

4. *Options*

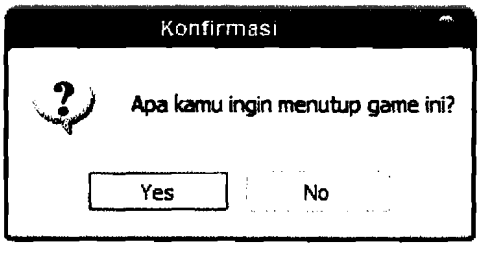
Form *options* ditampilkan jika kita ingin mengubah setingan (tekstur atau kecepatan putaran sisi rubik). Menu ini menampilkan form *options* seperti pada Gambar 4.4 berikut. Kita bisa memilih setingan kecepatan putar sisi-sisi rubik dan tekstur rubik. Jika kita menekan tombol *Apply* maka seting berubah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

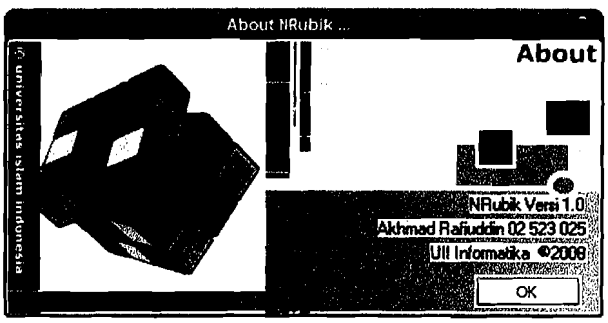
6. *Help*

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*

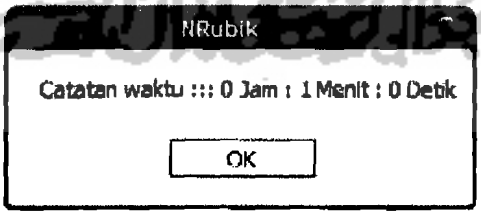
8. ***NewRecord***

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

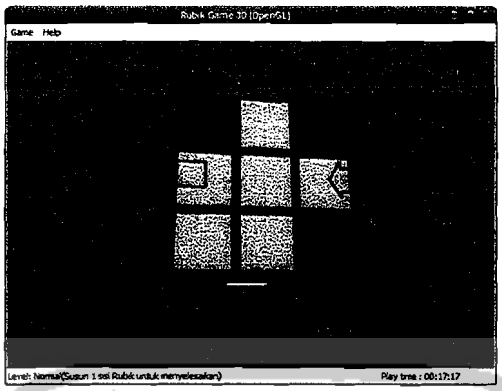
Pada proses ini penulis mencoba untuk menerapkan metode penyelesaian rubik cube pada level normal dan level *hard* (menggunakan metode solusi *beginner* Jasmine Lee)[2] guna mencoba apakah *game* sudah bisa digunakan dan diterapkan seperti *puzzle* rubik yang asli.[3]

1. Level Normal

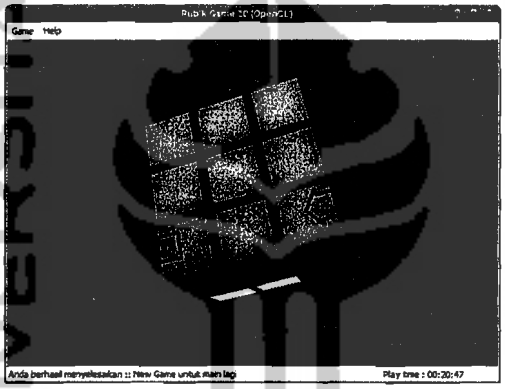
Pada level normal ini kita hanya diminta untuk menyelesaikan satu sisi warna dari *puzzle rubik*. Permainan di level ini selesai jika kita berhasil menyelesaikan salah satu sisi warna (merah, oranye, biru, hijau, kuning, maupun putih). Untuk menyelesaikannya pertama kita harus membuat pola plus pada 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.



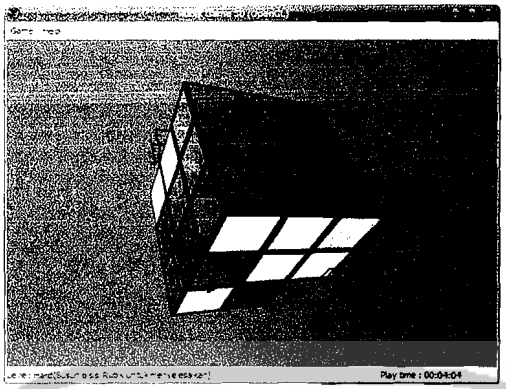
Gambar 4.11 Jadikan Bagian Pojok-Pojok



Gambar 4.12 Bentuk Jadi Mode Normal

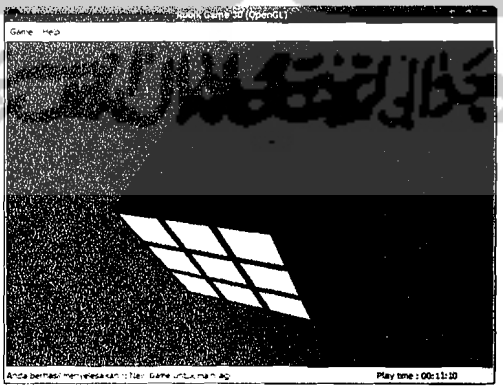
2. Level *Hard*

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.