BAB III
ANALISIS MASALAH DAN RANCANGAN PROGRAM

III.1 Pembahasan

Permainan Halma adalah salah satu permainan papan yang bertujuan memindahkan bidak-bidak dari satu area ke area lain yang sama warna dan susunannya, dengan mengatur strategi langkah yang tepat untuk melalui daerah permainan dan mendahului lawan. Permainan halma hanya dapat dimainkan minimal dua pemain dan maksimal tiga pemain. Program aplikasi permainan halma multiplayer ini memiliki empat modul, yaitu modul pembuka, modul pendaftaran, modul permainan dan modul help. Modul-modul tersebut dirancang dengan menggunakan perangkat lunak *Visual Basic* dan menggunakan komponen *Winsock* dalam berkomunikasi antar jaringan. Jaringan yang digunakan dalam aplikasi ini adalah jaringan LAN. Pengujian program aplikasi ini dilakukan dengan metode *Blackbox Testing* dan juga dilakukan pengujian terhadap pengguna aplikasi. Pengujian Blackbox Testing dilakukan dengan memberikan sejumlah *input* pada program aplikasi. *Input* yang dilakukan adalah melakukan beberapa langkah dalam permainan dan juga memasukkan *input* teks. *Input* tersebut kemudian diproses untuk menghasilkan *output* yang diinginkan. Hasil pengujian membuktikan bahwa program aplikasi dapat berjalan dengan benar, sesuai dengan kebutuhan fungsionalitasnya. Hasil pengujian dengan pengguna membuktikan bahwa selain menjadi sarana hiburan, aplikasi ini juga dapat menjaddi sarana komunikasi.

III.1.1 Analisis dan Perancangan Sistem

Masa permainan tradisonal kini telah hilang ditelan waktu. Semua berubah serba digital. Anak-anak bahkan orang dewasa dianggap tidak modern jika tidak mengikuti segala yang berbau digital. Jika dulu anak-anak cukup bermain kelereng, petak umpet atau main tembak-tembakan dengan teman sepermainannya, sekarang sudah dianggap tidak seru lagi. Permainan kini sudah beralih pada pola permainan *virtual*, seperti yang disuguhkan *PlayStation* atau jejaring sosial *Facebook.*"

III.1.2 Analisa Kebutuhan Sistem

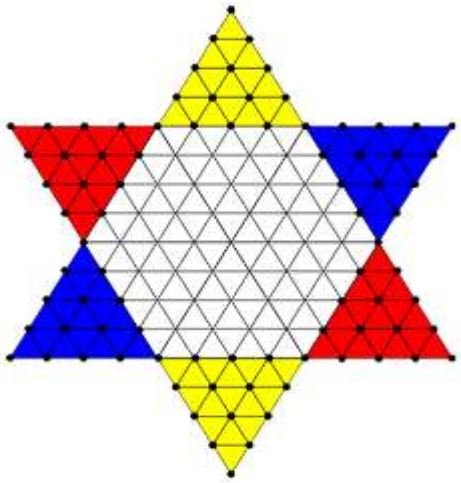
Tujuan dari fase analisis adalah memahami dengan sebenarnya kebutuhan dari sistem baru dan mengembangkan sebuah sistem yang mewadahi kebutuhan tersebut. Oleh karena itu, Analisis kebutuhan sistem (*system requirement*) sebagai salah satu dari fase analisis sistem sangat berperan penting untuk merumuskan tentang apa yang harus dimiliki dan dikerjakan oleh suatu sistem informasi.

III.1.3 Perencanaan

Perencanaan dilakukan setelah tahap analisis selesai. Tahap ini bertujuan untuk memenuhi kebutuhan penggunaan aplikasi dan memberikan gambaran mengenai aplikasi kepada *user*. Kegiatan yang dilakukan pada tahap ini adalah :

Proses perancangan halma melalui beberapa tahapan sebagai berikut:

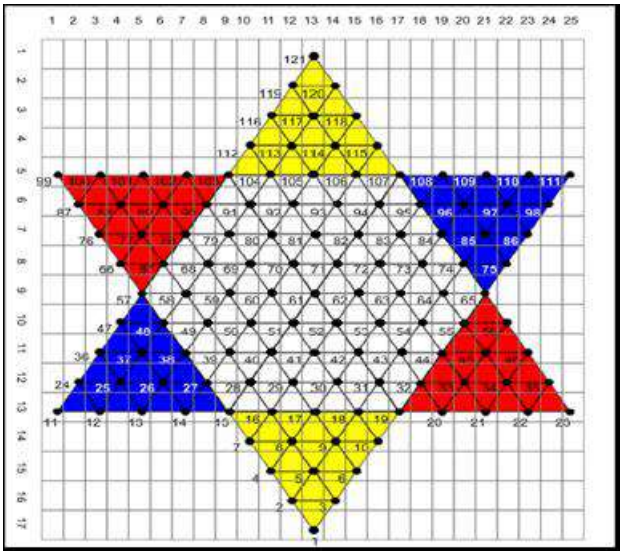
1. Proses Perancangan Gambar Papan Permainan Halma.



Gambar III.1 Papan Permainan Halma

Papan permainan Halma dirancang dengan menggunakan *ellipse tool* untuk menggambar lingkaran kecil dan dilakukan proses *fill color* dengan warna hitam untuk menghasilkan lingkaran hitam kecil. Sedangkan garis-garis pada papan permainan dirancang dengan menggunakan *line tool*. Proses terakhir, dirancang tiga buah segitiga sama sisi dengan cara menggambarkan garis-garis yang berhubungan secara berturut-turut hingga membentuk sebuah segitiga dan dilakukan proses *fill color* dengan warna kuning, merah dan biru. Kemudian tiga buah segitiga sama sisi yang dihasilkan tersebut diduplikasi dan dilakukan proses *rotate* hingga didapatkan posisi yang diinginkan

2. Proses Initialisasi Gambar Papan Halma.



Gambar III.2 Inisialisasi Papan Permainan Halma

Rancangan Papan Halma pada Gambar III.1 diberi inisialisasi. Gambar III.2 memperlihatkan bahwa terdapat matrik posisi berukuran 17 X 25 dimulai dari posisi [1,1] sampai posisi [17,25]. Papan Halma yang diberi tanda bulat ● disebut *node* yang jumlahnya 121 buah. Setiap *node* terletak pada sebuah matrik posisi. Pada *node* 1 berada pada matrik [17,13] dan *node* 61 berada pada matrik [9,13]. Perhatikan *node* 70 dimana memiliki tetangga *node* 80, 81, 69, 71, 60 dan 61. Jadi setiap *node* masing-masing memiliki tetangga. Sebuah *node* terdapat maksimum 6 buah *node* tetangga. Namun terdapat *node* yang tetangganya lebih kecil dari 6 *node* tetangga artinya ada *node* tetangga tidak berada di area papan halma diberi nilai 0

3. Proses Pengesetan *Board* Halma.

Pada saat sebelum permainan dimulai maka terlebih dahulu kita harus mengeset pion masing-masing ke tempat yang sudah ditentukan yaitu di posisi

awal rumah. Sehingga apabila setiap permainan dimulai maka pion pemain akan terletak pada rumah masing-masing.

4. Proses Pengecekan Langkah Yang Dapat Dijalankan Oleh Pion.

   Setiap pemain dapat menggerakkan pionnya ke posisi yang diinginkan. Namun, posisi yang diinginkan tersebut harus dapat dijalankan. Jika tidak, maka pergerakan pion tidak diperbolehkan

5. Proses Pencarian Langkah Terpendek.

   Pada proses pencarian langkah terpendek digunakan metode *Depth First Search* (DFS), proses akan dilakukan pada semua anaknya sebelum dilakukan pencarian ke titik (*node*) yang selevel. Pencarian dimulai dari *node* akar ke level yang lebih tinggi. Proses ini diulangi terus hingga ditemukannya solusi *path* terpendek.

6. Proses Pengecekan Pemenang.

   Pada proses ini akan dilakukan pengecekan terhadap pion yang telah masuk ke daerah tujuan rumah apakah semuanya sudah masuk atau tidak dengan cara menyimpan array posisi tujuan rumah. Pemain yang dahulu memasukkan semua pionnya ke daerah tujuan rumah dinyatakan sebagai pemenang.

III.2 Proses Pengembangan Sistem Menggunakan Algoritma *Greedy*

Algoritma *greedy* merupakan metode yang paling populer untuk memecahkan persoalan optimasi. Optimasi adalah persoalan yang tidak hanya mencari sekedar solusi namun juga mencari solusi terbaik. Algoritma *greedy* menggunakan prinsip "*take what you can get now*!". Algoritma *greedy* adalah

algoritma yang membentuk solusi langkah per langkah. Pada setiap langkah, terdapat banyak pilihan yang perlu dieksplorasi. Oleh karena itu, pada setiap langkah harus dibuat keputusan yang terbaik dalam menentukan pilihan. Keputusan yang telah diambil pada suatu langkah tidak dapat diubah lagi pada langkah selanjutnya. Pada setiap langkah kita membuat pilihan optimum lokal dengan harapan bahwa langkah sisanya mengarah ke solusi optimum global.

Elemen-elemen algoritma *greedy*:

1. Himpunan kandidat, *C*

   Himpunan kandidat adalah himpunan yang berisi elemen-elemen pembentuk solusi.

2. Himpunan solusi, *S*

   Himpunan solusi adalah himpunan yang berisi kandidat-kandidat yang terpilih sebagai solusi persoalan.

3. Fungsi seleksi (*selection function*)

   Fungsi seleksi adalah fungsi yang memilih kandidat-kandidat yang paling memungkinkan mencapai solusi optimal. Kandidat yang telah dipilih pada suatu langkah tidak pernah dipertimbangkan lagi pada langkah selanjutnya.

4. Fungsi kelayakan (*feasible*)

   Fungsi ini memeriksa apakah suatu kandidat yang telah dipilih dapat memberikan solusi yang layak, yakni kandidat tersebut bersama-sama dengan himpunan solusi yang sudah terbentuk tidak melanggar kendala yang ada. Kandidat yang layak dimasukkan ke dalam himpunan solusi, sedangkan kandidat yang tidak layak dibuang dan tidak pernah dipertimbangkan lagi

5. . Fungsi obyektif

Fungsi ini memaksimumkan atau meminimumkan nilai solusi.

Dengan kata lain algoritma *greedy* melibatkan pencarian sebuah himpunan bagian dari himpunan kandidat, dimana yang dalam hal ini, himpunan solusi harus me*menu*hi beberapa kriteria yang ditentukan, yaitu menyatakan suatu solusi dan himpunan solusi dioptimisasi oleh fungsi obyektif. Adapun *pseudo-code* untuk algoritma *greedy* adalah sebagai berikut:

```
function greedy(input C:
himpunan_kandidat)→ himpunan_kandidat
{Mengembalikan solusi dari persoalan optimasi dengan
algoritma greedy
Masukan : himpunan kandidat C
Keluaran: himpunan solusi yang bertipe himpunan_kandidat
}
Deklarasi
x : kandidat
S : himpunan_kandidat
Algoritma:
S ← {} {inisialisasi S dengan kosong}
while (not SOLUSI(S)) and (C !={} ) do
x ← SELEKSI(C) { pilih sebuah kandidat dari C}
C ← C - {x} { elemen himpunan kandidat berkurang satu }
if LAYAK(S ∪{x}) then
      S ← S ∪ {x}

endif
endwhile
{SOLUSI(S) or C = {} }
if SOLUSI(S) then
return S
else
write('tidak ada solusi')
endif
```

Algoritma pengembangan aplikasi halma multiplayer game yaitu :

1. Algoritma koneksi antar pemain

```
Private Sub Form_Unload(Cancel As Integer)

     Winsock(0),Close

     Winsock(1),Close

     Winsock(2),Close

End Sub
```

2. Algoritma pengesetan *board* halma

Pada saat sebelum permainan dimulai maka terlebih dahulu kita harus mengeset pion-pionnya masing-masing ke tempat yang sudah ditentukan yaitu di posisi awal rumah. Proses pengesetan ini menggunakan *array* untuk menyimpan posisi awal masing-masing setiap pion. Sehingga apabila setiap permainan dimulai maka pion-pion pemain akan terletak pada rumah masing-masing.

```
A = Array ((0,1,2,4,5,6,7,8,9,10,15,16,17,18,19
HomePosition(0)=A
GamePosition(1)=A
For   I1=1 To15
      I2=A(I1)
      If Pemain(0).WarnaBiji=”R”Then
            ImageYellow(I1).Left=1b1Drag(I2).Left
            ImageYellow(I1).Top=1b1Drag(I2).Top
            ImageYellow(I1).Tag=I2
            ISI(I2)=”K”
      Else If Pemain(0).WarnaBiji=”M” Then
            ImgRed (I1).Left=1b1Drag(I2).Left
            ImgRed (I1).Top=1b1Drag(I2).Top
            ImgRed (I1).Tag=I2
      ISI(I2)=”M”
      Else
            ImgBlue (I1).Left=1b1Drag(I2).Left
            ImgBlue (I1).Top=1b1Drag(I2).Top
            ImgBlue(I1).Tag=I2
            ISI(I2)=”R”
End If
```

3. Algoritma pencarian langkah terpendek

Metoda ini digunakan untuk mencari langkah terpendek yang ditempuh oleh pion pemain ke daerah tujuan.

```
For J = (i-1) To 1 Step -1
       If A(J) = PosisiAwal And IsValidMove2 (B(UBound(B)),A (J))
Then
         i = 1
         nLangkah A(1)
       ELself IsValidMove1(B(UBound(B)),,A (J)) Then
          i = i-1
         nLangkah A(J)
       End If
       f i = 1 Then Exit For
Next J
```

4. Proses Pengecekan Langkah – Langkah Yang Dapat Dijalankan Oleh Pion

```
       IsValidMove1 = False
       If ISI (POS(X),A1),A1) =""And ISI (POS(X),A1) <>""-
          And (pnPOSISITuuan = POS(POS(X),A1),A1) Then

       If ISI(POS(x),A1) <>""_
          And (pnPOSISITuuan = POS(POS(X),A1),A1) Then
```

Setiap pemain dapat menggerakkan pionnya ke posisi yang diinginkan. Namun, posisi yang diinginkan tersebut harus dapat dijalankan. Jika tidak, maka pergerakan pion tidak diperbolehkan. Proses pengecekan pergerakan pion yang diperbolehkan adalah sebagai berikut :

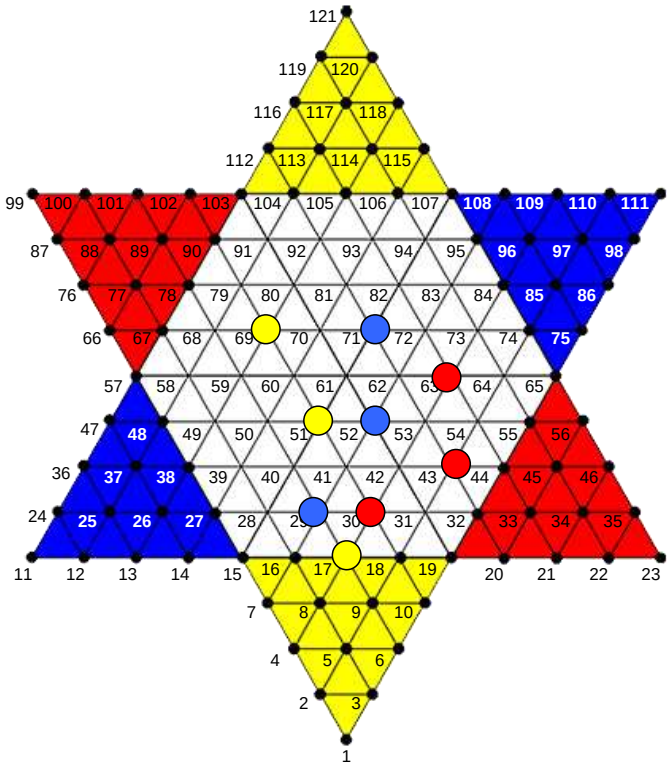
1. Pengecekan dimulai dari posisi awal pion dengan mengecek nilai setiap *pointer* dari posisi pion tersebut.
2. Nilai *pointer menu*njukkan posisi tujuan yang dapat digerakkan oleh pion tersebut. Jika nilai *pointer* bernilai 0, maka berarti pion tidak dapat digerakkan ke arah tersebut.

3. Jika posisi tujuan yang dapat digerakkan masih kosong (tidak ditempati oleh pion), maka pion berhenti di posisi tersebut dan tidak dapat digerakkan lagi.
4. Jika posisi tujuan yang dapat digerakkan tidak kosong (ditempati oleh pion), maka pion tersebut digerakkan ke posisi dengan arah *pointer* yang sesuai dengan arah posisi tujuan tersebut jika ditinjau sebagai nilai *pointer* dari posisi asal.
5. Proses pengecekan untuk langkah keempat dilakukan untuk semua nilai *pointer* dari posisi tujuan tersebut yang telah ditempati oleh pion hingga tidak terdapat nilai *pointer* dari posisi tujuan yang telah ditempati oleh pion.
6. Jika pada waktu proses pengecekan, didapat posisi tujuan yang telah diperoleh sebelumnya, maka proses pengecekan untuk posisi tujuan tersebut tidak perlu dilanjutkan lagi.
7. Pengecekan dimulai dari posisi awal pion dengan mengecek nilai setiap *pointer* dari posisi pion tersebut.
8. Nilai *pointer menu*njukkan posisi tujuan yang dapat digerakkan oleh pion tersebut. Jika nilai *pointer* bernilai 0, maka berarti pion tidak dapat digerakkan ke arah tersebut.
9. Jika posisi tujuan yang dapat digerakkan masih kosong (tidak ditempati oleh pion), maka pion berhenti di posisi tersebut dan tidak dapat digerakkan lagi.

10. Jika posisi tujuan yang dapat digerakkan tidak kosong (ditempati oleh pion), maka pion tersebut digerakkan ke posisi dengan arah *pointer* yang sesuai dengan arah posisi tujuan tersebut jika ditinjau sebagai nilai *pointer* dari posisi asal.
11. Proses pengecekan untuk langkah keempat dilakukan untuk semua nilai *pointer* dari posisi tujuan tersebut yang telah ditempati oleh pion hingga tidak terdapat nilai *pointer* dari posisi tujuan yang telah ditempati oleh pion.
12. Jika pada waktu proses pengecekan, didapat posisi tujuan yang telah diperoleh sebelumnya, maka proses pengecekan untuk posisi tujuan tersebut tidak perlu dilanjutkan lagi.

Agar lebih jelas, simaklah contoh berikut ini.

Misalkan ingin digerakkan pion kuning yang berada pada posisi 17 dengan posisi pion-pion lainnya seperti ditunjukkan oleh gambar berikut ini.



Gambar III.3 Contoh keadaan posisi pion-pion pada papan permainan halma

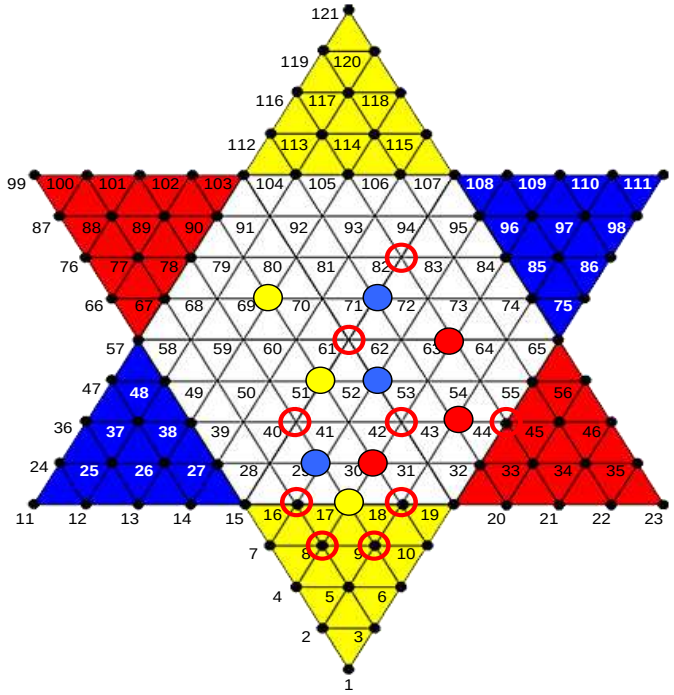
Posisi tujuan yang dapat dicapai oleh pion kuning pada posisi 17 tersebut adalah sebagai berikut :

1. **Posisi 8** yang merupakan nilai *pointer* B1 dari posisi 17.
2. **Posisi 9** yang merupakan nilai *pointer* B2 dari posisi 17.
3. **Posisi 16** yang merupakan nilai *pointer* S1 dari posisi 17.
4. **Posisi 18** yang merupakan nilai *pointer* S2 dari posisi 17.
5. Nilai *pointer* A1 dari posisi 17 yaitu posisi 29 telah ditempati oleh pion, maka posisi tujuan yang dapat ditempati oleh pion adalah sesuai dengan nilai *pointer* A1 dari posisi 29 yaitu **posisi 40**.

6. Karena posisi 40 ditempati dengan melompati pion lainnya, maka pion masih dapat digerakkan lagi, namun harus dengan melakukan lompatan, yang berarti bahwa posisi tujuan sesuai dengan nilai *pointer* dari posisi tersebut harus telah ditempati oleh pion lainnya. Jika tidak, maka pion tidak dapat digerakkan lagi. Nilai *pointer* dari posisi 40 hanya *pointer* A2 yaitu posisi 51 dan B2 yaitu posisi 29 yang telah terisi, maka *pointer* A2 dari posisi 51 yaitu **posisi 61**. Pengecekan dilanjutkan untuk posisi 61. Nilai *pointer* dari posisi 61 hanya *pointer* A2 yaitu posisi 71 dan *pointer* B2 yaitu posisi 52 yang telah ditempati oleh pion maka *pointer* A2 dari posisi 71 yaitu **posisi 82** dan *pointer* B2 dari posisi 52 yaitu **posisi 42** merupakan posisi tujuan yang dapat ditempati oleh pion.
7. Pengecekan dilanjutkan untuk posisi 82 dan posisi 42. Nilai *pointer* dari posisi 82 hanya *pointer* B1 yaitu posisi 71 yang telah ditempati oleh pion, maka *pointer* B1 dari posisi 71 yaitu posisi 61 merupakan posisi tujuan yang dapat ditempati oleh pion. Namun, karena posisi 61 merupakan posisi asal sebelumnya dan telah dimasukkan sebagai posisi tujuan, maka proses pengecekan untuk posisi 61 dihentikan. Nilai *pointer* dari posisi 42 hanya *pointer* A1 yaitu posisi 52, *pointer* B1 yaitu posisi 30, dan *pointer* S2 yaitu posisi 43 yaitu posisi 44 merupakan posisi tujuan yang dapat ditempati oleh pion. Namun, karena posisi 17 merupakan posisi awal maka posisi 17 bukan merupakan posisi tujuan. **posisi 44**.
8. Nilai *pointer* A2 dari posisi 17 yaitu posisi 30 telah ditempati oleh pion, maka *pointer* A2 dari posisi 30 yaitu posisi 42 merupakan posisi tujuan

yang dapat ditempati oleh pion. Namun, karena posisi 42 sebelumnya telah dimasukkan sebagai posisi tujuan, maka proses pengecekan untuk posisi 42 dihentikan.

Maka, posisi tujuan yang dapat dicapai oleh posisi 17 adalah posisi 8, 9, 16, 18, 40, 42, 44, 61, dan 82 seperti ditunjukkan oleh tanda lingkaran merah pada gambar berikut ini.



Gambar III.4 Contoh posisi tujuan dari pion pada papan permainan halma

III.3 Aturan Permainan Halma

Permainan ini dimainkan dalam suatu daerah yang berbentuk bintang berkaki enam. Permainan ini dapat dimainkan oleh maksimal 3 pemain sekaligus dengan diwakili oleh 3 macam warna, yaitu warna merah, kuning dan biru. Setiap pemain memiliki 15 buah pion berwarna. Sasaran dari permainan ini adalah

memindahkan semua pion berwarna tersebut dari tempat (daerah) asal ke tempat (daerah) tujuan di seberang.

III.3.1 Aturan Tingkat *Beginner*

1. Jumlah pemain minimal 2 orang dan maksimal 3 orang.
2. Permainan dapat menggunakan batas waktu.
3. Apabila menggunakan batas waktu maka pemain yang kehabisan waktu sebelum melangkah tidak diperbolehkan jalan dan diganti pemain lain.
4. Pion dapat digeser satu langkah kedepan apabila posisi tujuan kosong.
5. Pion dapat melangkah melompati dengan syarat terdapat satu pion rintangan didepan jalurnya dan posisi tujuan kosong.
6. Pemain tidak diperbolehkan meng-undo langkahnya kembali.
7. Pemain yang duluan memindahkan semua pionnya ke posisi tujuan rumah dinyatakan menang.

III.3.2 Aturan Tingkat *Expert*

1. Jumlah Pemain minimal 2 orang dan maksimal 3 orang.
2. Permainan dapat menggunakan batas waktu.
3. Apabila menggunakan batas waktu maka pemain yang kehabisan waktu sebelum melangkah tidak diperbolehkan jalan dan diganti pemain lain.
4. Pion dapat digeser satu langkah kedepan apabila posisi tujuan kosong.
5. Pion dapat melangkah melompati dengan syarat terdapat satu pion rintangan didepan jalurnya dan posisi tujuan kosong.
6. Apabila ada pion yang belum keluar dari 4 tingkat posisi diawal rumah maka pion teman tidak dapat masuk ke posisi tujuan rumah dan pion yang

telah masuk didalam rumah tidak boleh digeser lagi dan kalau mati langkah maka diperbolehkan ditempatkan kembali ke posisi awal rumah pion tersebut.

7. Pemain tidak diperbolehkan meng-*undo* langkahnya kembali.
8. Pemain yang duluan memindahkan semua pionnya ke posisi tujuan rumah dinyatakan menang.

III.4 Perancangan

Perangkat lunak permainan Halma ini dirancang dengan menggunakan bahasa pemrograman *Microsoft Visual Basic* Komponen- komponen yang digunakan dalam pembuatan perangkat lunak ini sebagai berikut :

1. *Form* : merupakan lembaran kerja tempat meletakkan item dalam *Window Visual Basic*.
2. *Label* : unit ini digunakan untuk menampilkan teks, angka, atau simbol pada saat program dijalankan.
3. *Text Box* : unit ini digunakan untuk menampilkan teks pada *form* atau untuk menerima *input* dari pemakai pada saat program *Visual Basic* dijalankan.
4. *Command Button* : unit ini digunakan untuk memberikan suatu perintah atau tindakan ketika digunakan.
5. *Check Box* : unit ini digunakan untuk memilih satu atau beberapa syarat secara bersamaan.

6. *Combo Box* : unit ini digunakan untuk memilih item lewat *Drop-Down List*.
7. *Line* : unit ini memungkinkan pemakai membuat garis lurus.
8. *Picture Box* : unit ini untuk menampilkan file gambar (*Bitmaps, Icon, Gif, JPG* dan sebagainya).
9. *Image Box* : unit akan menampilkan gambar *Bitmaps, Windows, Metafile* dan *Icon*.
10. *Timer* : unit ini digunakan untuk mengoperasikan waktu kejadian pada rutin program termasuk internal waktu.
11. *Shape* : Unit ini membentuk objek dua dimensi (bujur sangkar, lingkaran, emapat persegi panjang dan *elips*).
12. *Progress Bar* : unit ini untuk menampilkan berapa lama suatu operasi berlangsung.
13. *Menu Editor* : unit ini digunakan untuk membuat *menu* aplikasi.
14. *MSFlexGrid* : unit ini digunakan untuk menampilkan informasi *database*.
15. *Winsock* : unit ini digunakan sebagai perantara koneksi antara sesama komputer.

Perangkat lunak ini memiliki beberapa *form* yaitu :

1. *Form Splash Screen*.
2. *Form* Pengaturan Koneksi Jaringan.
3. *Form* Setting Permainan.
4. *Form* Permainan Halma.

5. *Form About*.

III.4.1 *Form Splash Screen*

Gambar III.5 Rancangan *Form Splash Screen*

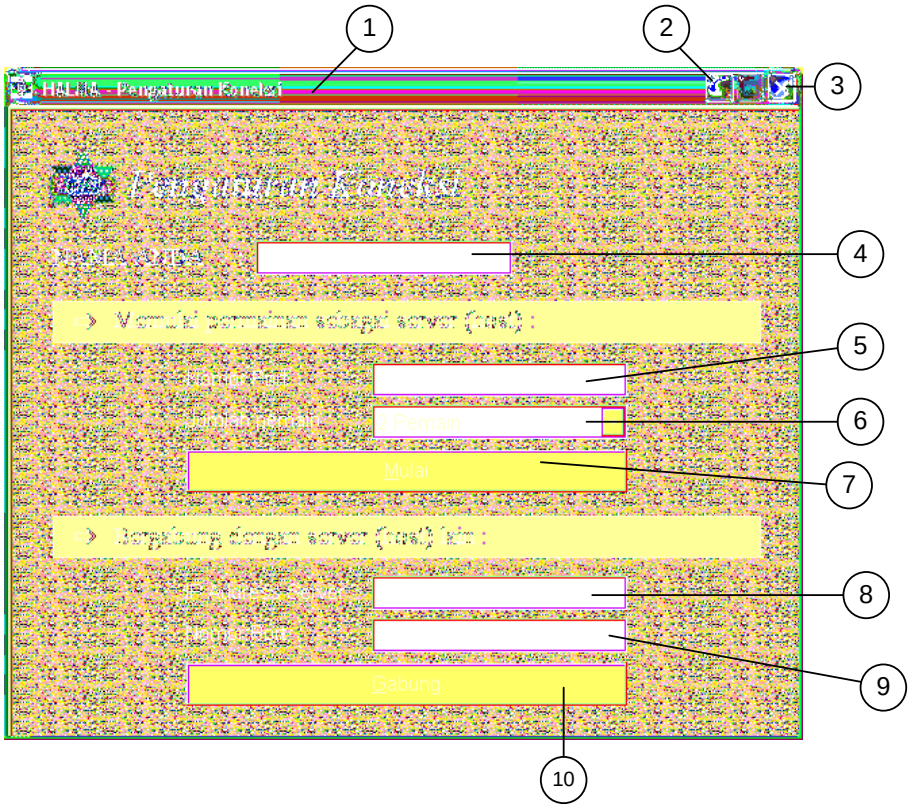
Keterangan :

1 : nama perangkat lunak.

2 : gambar logo dari perangkat lunak.

3 : data pembuat program.

III.4.2 Form Pengaturan Koneksi Jaringan



Gambar III.6 Rancangan *Form* Pengaturan Koneksi Jaringan

Keterangan :

1 : *title bar* dengan tulisan ‘HALMA - Pengaturan Koneksi’.

2 : tombol ‘*Minimize*’ untuk mengecilkan tampilan *form*.

3 : tombol ‘Close’ untuk *menu*tup *form*.

4 : *text box* untuk meng-*input* nama pemain.

5 : *text box* ‘Nomor *Port*’ untuk mengisi kode pengenal dari komputer *server*.

6 : *combo box* untuk memilih jumlah pemain yang diinginkan (2 atau 3 orang).

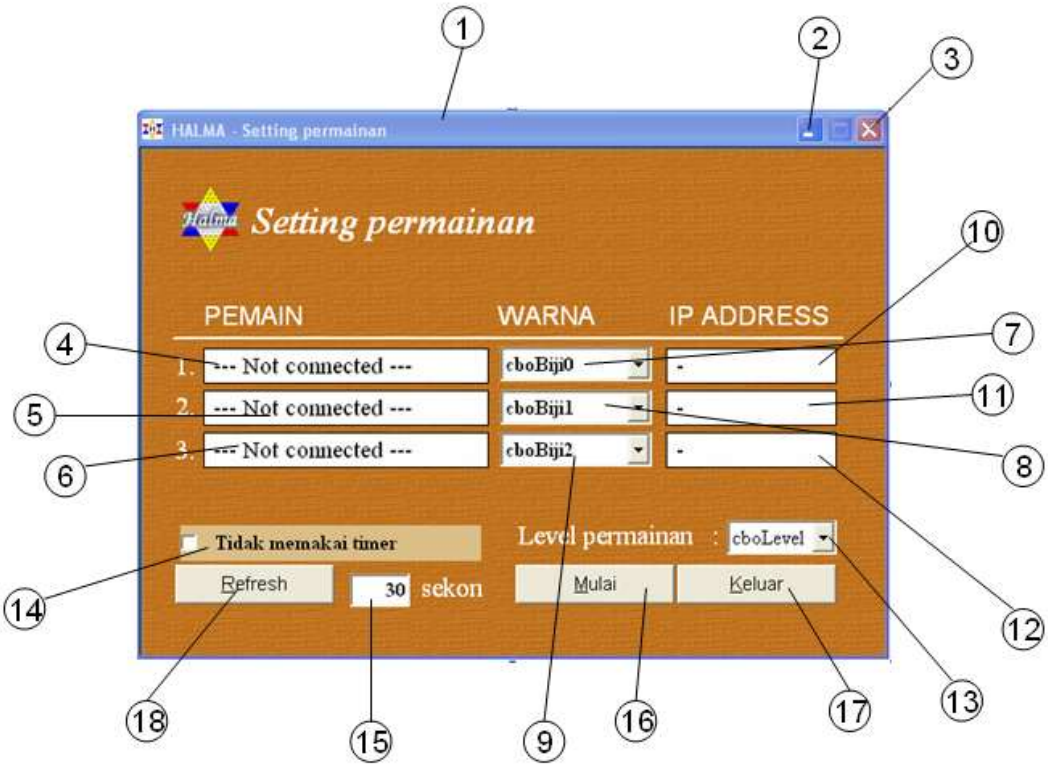
7 : tombol ‘Mulai’ untuk memulai permainan sebagai *server*.

8 : *text box* ’*IP Address Server*’ untuk mengisi *IP Address* dari *server* tujuan.

9 : *text box* ‘Nomor *Port*’ untuk mengisi kode pengenal dari *server* tujuan.

10 : tombol ‘Gabung’ untuk bergabung dengan *server* lain.

III.4.3 Form Setting Permainan



Gambar III.7 Rancangan *Form Setting* Permainan

Keterangan :

1 : *title bar* dengan tulisan ‘HALMA – Setting permainan’.

2 : tombol ‘*Minimize’* untuk mengecilkan tampilan *form*.

3 : tombol ‘*Close*’ untuk menutup *form*.

4 : daerah tampilan nama pemain 1 (*server*).

5 : daerah tampilan nama pemain 2 (*client*).

6 : daerah tampilan nama pemain 3 (*client*).

7 : *combo box* untuk memilih warna pion dari pemain 1 dan hanya dapat dipilih oleh *server*.

8 : *combo box* untuk memilih warna pion dari pemain 2 dan hanya dapat dipilih oleh *server*.

9 : *combo box* untuk memilih warna pion dari pemain 3 dan hanya dapat dipilih oleh *server*.

10 : daerah tampilan *IP Address* dari pemain 1.

11 : daerah tampilan *IP Address* dari pemain 2.

12 : daerah tampilan *IP Address* dari pemain 3.

13 : *combo box* '*Level* Permainan' untuk memilih *level* permainan yang diinginkan.

Terdapat 2 buah pilihan yaitu :

a. '*Beginner*', di mana permainan dapat dimainkan secara bebas dengan mengikuti aturan dasar permainan Halma.

b. '*Expert*', dengan aturan permainan bahwa pion yang telah menempati daerah tujuan tidak dapat digerakkan lagi

14 : *check box* untuk memilih apakah ingin menggunakan batas waktu (*timer*) untuk setiap giliran permainan atau tidak.

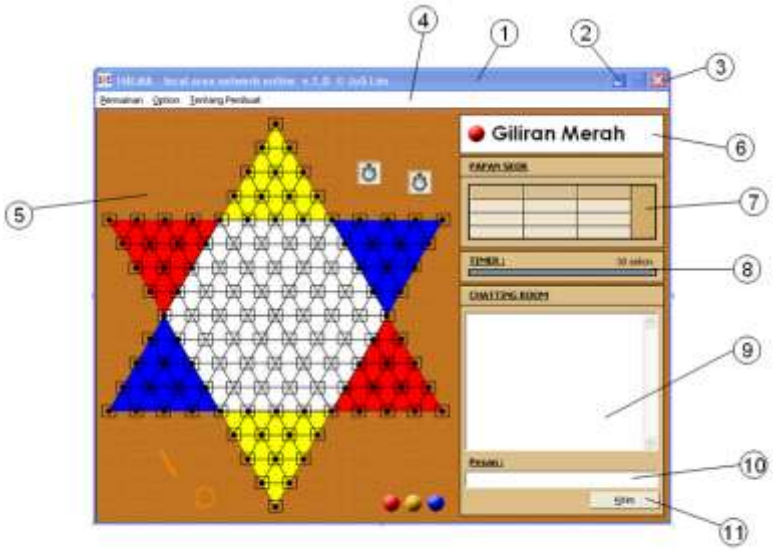
15 : *text box* untuk menginput lama batas waktu yang diinginkan.

16 : tombol 'Mulai' untuk memulai permainan.

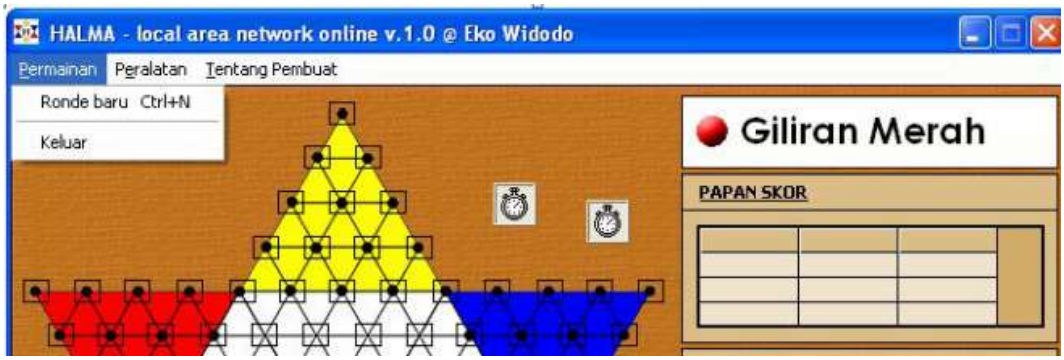
17 : tombol 'Keluar' untuk membatalkan proses.

18 tombol '*Refresh'* untuk mengrefresh jaringan.

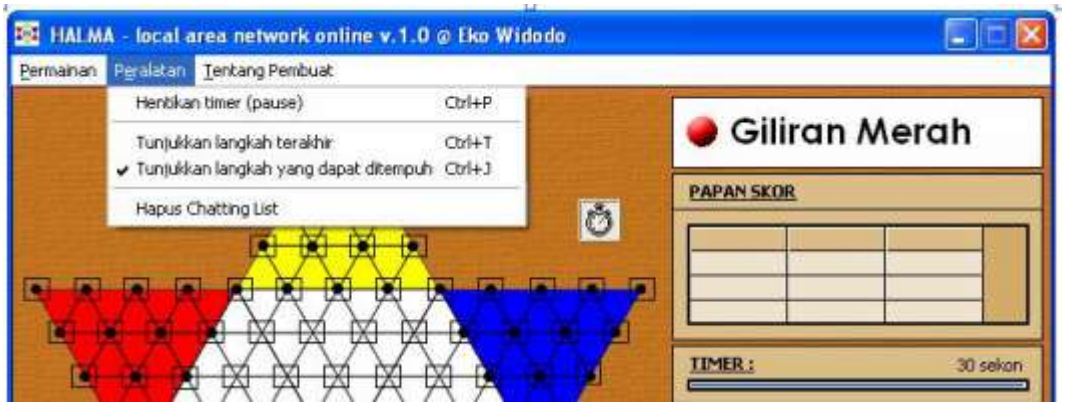
III.4.5 *Form* Permainan Halma



Gambar III.8 Rancangan *Form* Permainan Halma



Gambar III.9 Rancangan *Menu* Permainan Pada *Form* Permainan Halma



Gambar III.10 Rancangan *Menu Option* Pada *Form* Permainan Halma

1 : *title bar* dengan tulisan 'HALMA – *local area network online* v.1.0 © eko

2 : tombol '*Minimize'* untuk mengecilkan tampilan *form*.

3 : tombol '*Close'* untuk *menu*tup *form*.

4 : *menu bar* dengan rincian *menu* sebagai berikut :

a. *Menu* 'Permainan', terdiri dari beberapa sub *menu* yaitu :

i. 'Ronde Baru' untuk memulai permainan baru.

ii. 'Keluar' untuk keluar dari perangkat lunak.

b. *Menu* '*Option'*, terdiri dari beberapa sub *menu* yaitu :

i. 'Hentikan timer (*pause*)' untuk menghentikan *timer* berjalan.

ii. 'Tunjukkan langkah terakhir' untuk menunjukkan langkah terakhir yang dijalankan.

iii. 'Tunjukkan langkah yang dapat ditempuh' untuk menunjukkan posisi-posisi tujuan yang dapat dicapai.

c. *Menu* ‘Tentang Pembuat’ untuk menampilkan *form* ‘About’.

5 : papan permainan ‘Halma’.

6 : daerah tampilan giliran pemain.

7 : tabel yang berisi skor pemain.

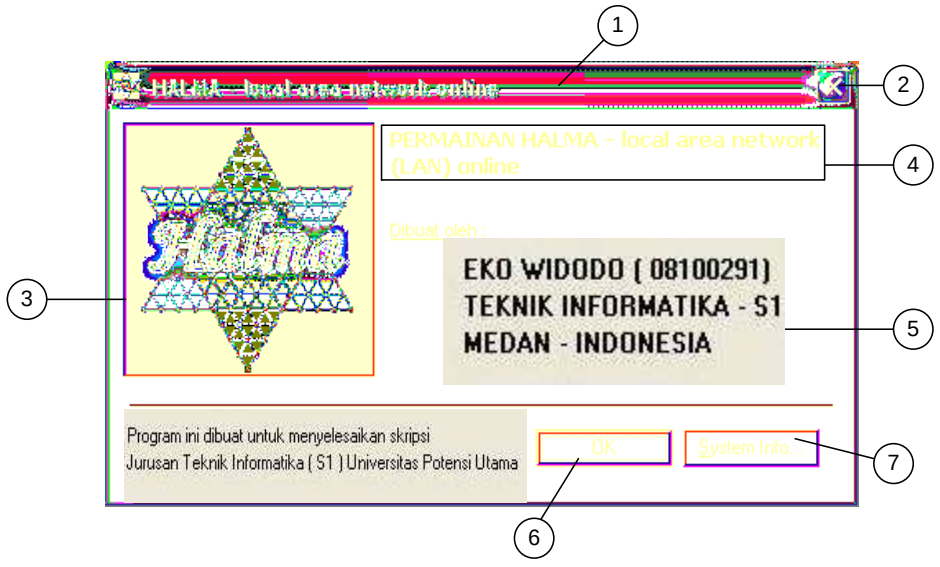
8 : *timer* berjalan.

9 : daerah tampilan kata-kata (pesan) yang dikirim oleh setiap komputer.

10 : *text box* untuk menginput kata (pesan).

11 : tombol ‘Kirim’ untuk mengirim pesan ke *server* untuk dikirim ke semua komputer yang tergabung dalam permainan.

III.4.6 *Form About*



Gambar III.11 Rancangan *Form About*

Keterangan,

1 : *title bar* dengan tulisan ‘HALMA – *local area network online’*.

2 : tombol ‘*Close*’ untuk menutup *form*.

3 : gambar logo dari perangkat lunak.

4 : nama perangkat lunak.

5 : data pembuat program.

6 : tombol ‘OK’ untuk keluar dari *form*.

7 : tombol ‘*System Info*’ untuk menampilkan informasi mengenai sistem operasi yang digunakan.