# BAB III

# ANALISIS DAN DESAIN SISTEM

## III.1. Analisis Masalah

Perkembangan *game* dari skala kecil maupun besar sangat bervariasi yang dapat dimainkan oleh siapa saja tanpa memandang umur, dari anak – anak hingga orang dewasa. *Game* berkembang begitu pesat dengan jenis *platform* yang beragam mulai dari *console*, *mobile*, PC dan lain sebagainya, serta dapat dimainkan secara *online* maupun *offline*.

Sudoku adalah sebuah permainan teka-teki penempatan angka yang berbasiskan pada logika manusia. Tujuan dari permainan Sudoku sebenarnya sangat sederhana, yaitu mengisi suatu matriks yang berukuran 9x9 sehingga pada setiap kolom, setiap baris, dan setiap dari kotak yang berukuran 3x3 mengandung *digit* angka dari 1 sampai 9, dan pada setiap baris, kolom, dan kotak yang berukuran 3x3 tersebut tidak terdapat dua atau lebih kemunculan angka yang sama. Daya tarik dari permainan Sudoku ini ialah aturan permainan yang sederhana, tetapi penalaran yang dibutuhkan untuk menyelesaikan permainan ini dapat saja berubah menjadi kompleks.

Permainan ini jika diselesaikan secara manual akan memakan waktu cukup lama, sehingga mulai dilakukan penelitian untuk menyelesaikan *puzzle* Sudoku menggunakan beberapa algoritma yang dalam prosesnya diperlukan banyak iterasi. Telah banyak algoritma yang digunakan untuk menyelesaikan permasalahan ini, antara lain adalah *Unique Missing Candidate*, *Naked Singles*,

*Hidden Singles*, *Coloring and Multi-Coloring*, *Forcing Chains*, *Guessing*, dan lain-lain. Namun seluruh teknik tersebut memerlukan generasi yang sangat lama untuk sampai menemukan solusi yang tepat. Karena adanya alasan seperti yang telah disebutkan di atas maka banyak orang yang tidak bisa melanjutkan lagi permainannya menggunakan program-program komputer yang didesain khusus untuk menyelesaikan permainan Sudoku. Oleh karena itu timbullah kebutuhan akan adanya algoritma yang efektif dan efisien untuk menyelesaikan permainan Sudoku ini.

## III.2. Strategi Pemecahan Masalah

Adapun strategi pemecahan masalah dari analisa di atas, maka akan dirancang suatu aplikasi *game Sudoku* dengan mengggunakan algoritma *Brute Force* berbasis android. Perancangan aplikasi *game Sudoku* ini menggunakan *Eclipse Galileo* sebagai desain pengembang aplikasi. *Eclipse* memiliki sifat m*ulti-platform* (dapat dijalankan di semua *platform*), m*ulti-language* (mendukung pengembangan aplikasi beberapa bahasa pemrograman) dan m*ulti-role* (digunakan juga sebagai aktivitas dalam siklus pengambangan perangkat lunak), Sedangkan untuk *platform* yang digunakan pada aplikasi *game Sudoku* ialah *platform Android* 2.2 (*Froyo*) jenis yang merupakan generasi kedua dari Sistem Operasi *Android*. Aplikasi *games* yang akan dirancang hanya dapat dijalankan pada *handphone* yang memiliki sistem operasi *android*. Untuk pemecahan masalah (*problem solver*) menggunakan algoritma *Brute Force* agar permainan Sudoku ini diharapkan dapat berjalan dengan lebih optimal.

Aplikasi *game Sudoku* yang akan dirancang akan dijalankan pada *handphone* yang memiliki sistem operasi *android*, Setelah aplikasi selesai dirancang, maka penulis akan menjalankan aplikasi tersebut pada *emulator android*, *Emulator Android* atau *virtual* perangkat *mobile* adalah program yang menduplikasi fungsi-fungsi *smartphone* yang berjalan di atas *platform Android*. *Emulator* juga berfungsi sebagai pengujian aplikasi di komputer, pengujian diperlukan sebagai bahan masukan bagi penulis untuk mengetahui kekurangan dan kesalahan dari hasil aplikasi yang telah dirancang sebelum dijalankan langsung pada *handphone*.

Dalam pembuatan *game* untuk *handphone*, ada 2 hal penting yang harus dilakukan yaitu:

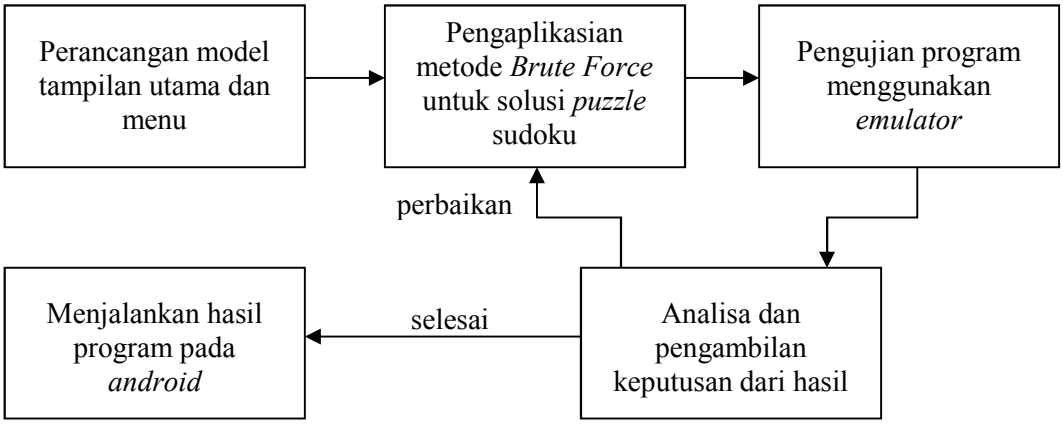
1. Desain Model visualisasi

   Dalam desain model visualisasi (tampilan pada layar *handphone*) kita harus harus merancang sebuah aplikasi yang memenuhi aspek kemudahan dalam menggunakan aplikasi, berarti aplikasi dapat bekerja secara efektif dan efisien dengan mengoptimalkan ukuran layar *handphone* yang sangat terbatas.

2. Pilihan menu yang disediakan

   Dalam pembuatan aplikasi kita juga harus memperhatikan menu yang disajikan dalam aplikasi.

Berikut penulis akan menggambarkan tahapan dalam pengerjaan pembuatan aplikasi *game Sudoku*, adapun tahapan tersebut dapat dilihat pada gambar III.1. berikut.




**Gambar III.1. Tahapan Pembuatan *Game Sudoku***

### III.2.1. Penerapan Metode *Brute Force*

Beberapa karakteristik dari algoritma *Brute Force* dapat dijelaskan sebagai berikut :

1. Membutuhkan jumlah langkah yang banyak dalam menyelesaikan suatu permasalahan sehingga jika diterapkan menjadi suatu algoritma program aplikasi akan membutuhkan banyak memori.
2. Digunakan sebagai dasar dalam menemukan suatu solusi yang lebih efektif.
3. Banyak dipilih dalam penyelesaian sebuah permasalahan yang sederhana karena kemudahan cara berpikirnya

Berikut ini adalah beberapa keunggulan algoritma *Brute Force* :

1. *Brute Force* umumnya tidak "cerdas" dan tidak mangkus, karena ia membutuhkan jumlah langkah yang besar dalam penyelesaiannya.
2. Kadang-kadang algoritma *Brute Force* disebut juga algoritma naif (*naive algorithm*).
3. Algoritma *Brute Force* seringkali merupakan pilihan yang kurang disukai karena ketidakmangkusannya itu, tetapi dengan mencari pola-pola yang

mendasar, keteraturan, atau trik – trik khusus, biasanya akan membantu kita menemukan algoritma yang lebih cerdas dan lebih mangkus.

Penulis menggunakan *Brute Force* untuk menyelesaikan permasalahan pada *game* Sudoku. Ini dikarenakan *Brute Force* adalah suatu algoritma yang memecahkan persoalan dengan cara yang *straightforward*. Secara garis besar cara kerja algoritma ini ialah dengan membangkitkan semua permutasi dari komponen vektor solusi, kemudian setelah dibangkitkan maka akan diperiksa himpunan mana yang merupakan solusi dari sebuah persoalan.

Berikut contoh dari *game* Sudoku yang akan dirancang :

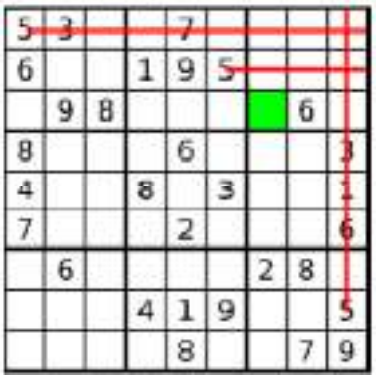
**Gambar III.2. Contoh *Game Sudoku***

### III.2.2. Solusi *Puzzle* Game Sudoku

Strategi di dalam menyelesaikan persoalan Sudoku secara manual dapat dianggap sebagai gabungan dari 3 proses yang dikerjakan, yaitu memeriksa, menandai, dan menganalisis. Proses memeriksa dilakukan pada awal dan saat permainan Sudoku berlangsung. Proses ini sendiri terdiri dari 2 teknik yaitu *cross hatching* dan perhitungan.

Pada proses *cross hatching* dilakukan pemeriksaan tiap baris, dipilih sel mana yang dapat diisi oleh angka tertentu melalui proses eliminasi. Hal yang sama juga dilakukan pada tiap kolom. Untuk hasil yang lebih baik lagi, pemeriksaan dilakukan secara sekuensial terhadap frekuensi kemunculan angka tersebut.

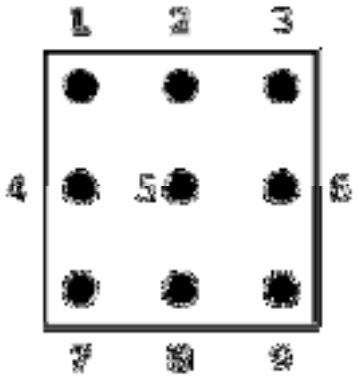
Proses perhitungan sendiri adalah sebuah proses dimana dilakukan perhitungan 1 sampai 9 pada baris, kolom, dan sub-matriks untuk menemukan angka yang "hilang". Perhitungan yang dimulai dari angka terakhir yang ditemukan dapat lebih meningkatkan kecepatan proses ini. Pada beberapa soal yang memiliki tingkat kerumitan tinggi, akan lebih baik jika perhitungan dilakukan terhadap angka yang tidak mungkin ada di baris, kolom, atau sub-matriks tersebut.




**Gambar III.3. Proses Pemeriksaan pada Sudoku**

Proses pemeriksaan dihentikan ketika tidak ditemukan lagi angka – angka lebih lanjut. Suatu metode lanjutan dari proses ini adalah memberikan tanda berupa angka – angka yang mungkin pada sel – sel yang masih kosong. Pemberian tanda ini pun terdiri dari 2 macam, yaitu pemberian tanda berupa notasi titik dan pemberian tanda berupa angka. Pada pemberian tanda berupa

notasi titik, lokasi titik yang terdapat pada sel yang kosong menunjukkan angka mulai dari 1 hingga 9.




**Gambar III.4. Pemberian Tanda Berupa Titik**

Sedangkan pada pemberian tanda berupa angka, angka – angka yang menjadi kandidat sel kosong tersebut dituliskan sebagai *subscript* pada sel kosong tersebut. Penggunaan 2 warna yang berbeda sangatlah dianjurkan.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 46 | 1469 | 3 | 69 | 7 | 2 | 48 | 34589 |

**Gambar III.5. Pemberian Tanda Berupa Angka**

Pendekatan utama pada proses selanjutnya adalah eliminasi kandidat dan skema “apa yang terjadi jika”. Pada proses eliminasi kandidiat, berlangsung eliminasi kandidat – kandidat angka pada sel kosong tersebut hingga hanya meninggalkan satu kandidat yang layak untuk sel tersebut. Pada skema “apa yang terjadi jika” biasanya terdapat dua pilihan angka dan dilakukan penebakan pada kedua angka tersebut.

### III.2.3. Solusi *Puzzle* Game Sudoku Menggunakan *Brute Force*

Penerapan algoritma *Brute Force* ini dalam permainan Sudoku ialah dengan mencoba seluruh kemungkinan isi kotak elemen dari matriks. Tahapan langkahnya adalah dengan mengisi semua sel yang kosong dan mencoba mengisinya dengan nomor dari pilihan yang tersedia. Jika tidak ada pelanggaran yang ditemukan setelah penyisipan, maka akan berlanjut ke sel kosong berikutnya dan diulang dari awal kembali. Bila terdapat pelanggaran terhadap aturan Sudoku ditemukan, algoritma akan mundur dan menambah sel sebelumnya.

Misalkan solusi dari suatu permainan Sudoku dinyatakan sebagai:

$$\mathbf{X=(x_1,x_2,x_3,\ldots..,x_9\ ^{81});}$$

Dari vektor solusi di atas dapat kita simpulkan bahwa dengan menggunakan algoritma *Brute Force* kita harus membangkitkan seluruh kemungkinan vektor solusi yang berjumlah $9^{81}$ =1,99x1077 kemungkinan.

*Pseudocode* dari algoritma *Brute Force* ini berupa *loop* berkalang sebanyak 81 buah, dimana di setiap *loop* dienumerasi seluruh kemungkinan angka yang dapat dimiliki oleh setiap elemen matriks Sudoku, yaitu :

```
x1,x2,x3,x4,x5,x6,x7,x8,x9,... :
integer
        for(x1=1 to 9)
                {
                for(x2=1 to 9)
                {
                for(x3=1 to 9)
                {
                for(x4=1 to 9)
                {
                for(x5=1 to 9)
                {
//semua loop berkalang
for(x81=1 to 9)
                {
        if
```

```
        (Solusi(x1,x2,x3,x4,x5,x6,x7,x8,x9))
        {
        }
        }
        }
        }
}
```

Kompleksitas asimptotik dari algoritma *Brute Force* murni di atas, dilihat dari banyaknya operasi perbandingan dilakukan, dapat disimpulkan bernilai $O(n^n)$ yang termasuk kategori *non polynomial complexity*, sehingga penggunaan algoritma ini sangatlah tidak menguntungkan jika dilihat dari waktu dan banyaknya komputasi yang dilakukan oleh komputer. Dari hasil yang telah kita dapatkan di atas maka dibutuhkan algoritma lain yang lebih mangkus daripada algoritma *Brute Force* murni di atas untuk menyelesaikan masalah permainan Sudoku, yaitu teknik *Brute Force Constraint*.

Pada teknik ini, dilakukan proses eliminasi terhadap semua kemungkinan susunan angka yang melanggar aturan dari permainan Sudoku. Sehingga semua kemungkinan yang diperhitungkan hanyalah kemungkinan yang mengandung angka unik pada satu baris, kolom dan sub matriks yang ada.

Dengan menggunakan teknik *Brute Force Constraint* ini, dapat dilakukan eliminasi kemungkinan hingga hanya tersisakan sekitar $6{,}67 \times 10^{21}$ kemungkinan yang tidak melanggar aturan permainan Sudoku. Hal ini menunjukkan bahwa penggunaan *Brute Force Constraint* mampu menekan waktu dan jumlah pengecekan yang muncul, sehingga algoritma ini dianggap memiliki tingkat kemangkusan yang lebih bila dibandingkan dengan algoritma *Brute Force* murni.

Berikut langkah – langkah proses penyelesaian masalah *game* Sudoku (*Sudoku Solver*), yaitu :

1. Tempatkan angka “1” pada sel pertama. Periksa apakah penempatan “1” dibolehkan (dengan memeriksa baris, kolom, dan kotak).
2. Jika tidak ada pelanggaran, maju ke sel berikutnya. Tempatkan “1” pada sel tersebut dan periksa apakah ada pelanggaran.
3. Jika pada pemeriksaan ditemukan pelanggaran, yaitu penempatan “1” tidak dibolehkan, maka coba dengan menempatkan “2”.
4. Jika pada proses penempatan ditemukan bahwa tidak satupun dari 9 angka diperbolehkan, maka tinggalkan sel tersebut dalam keadaan kosong, lalu mundur satu langkah ke sel sebelumnya. Nilai di sel tersebut dinaikkan 1.
5. Ulangi sampai 81 buah sel sudah terisi solusi yang benar.

### III.3. Desain Sistem

Bentuk rancangan sistem yang akan dibuat menggunakan beberapa bentuk diagram dari *Unified Modeling Language* (UML) yaitu *Use Cace Diagram*, *Sequence Diagram* dan *Activity Diagram*. Perancangan aplikasi *game* ini meliputi rancangan menu utama, yang di dalamnya terdapat *list* untuk Permainan Baru, Keterangan dan Keluar.

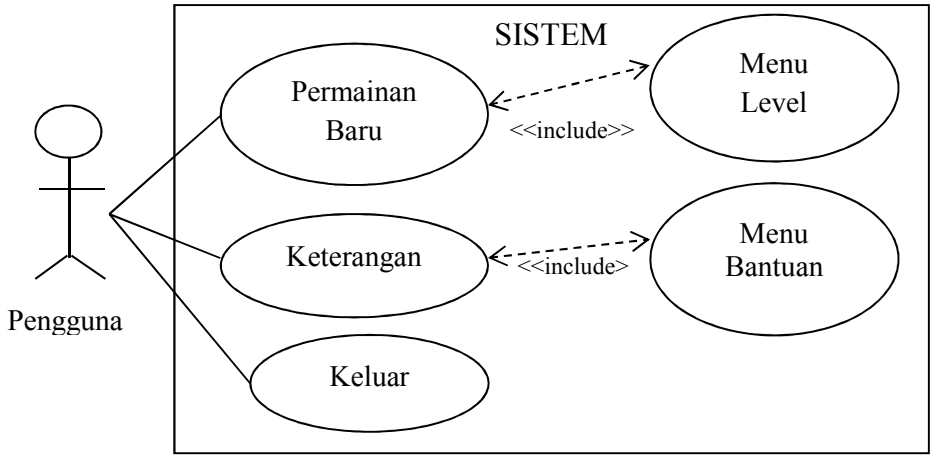
#### III.3.1. Use Case Diagram

*Use Case Diagram* adalah gambaran *graphical* dari beberapa atau semua *actor*, *use case*, dan interaksi diantara komponen-komponen tersebut yang memperkenalkan suatu sistem yang akan dibangun digunakan untuk menjelaskan bagaimana langkah-langkah yang seharusnya dikerjakan oleh sistem. *Use Case Diagram* menjelaskan manfaat suatu sistem jika dilihat menurut pandangan orang

yang berada di luar sistem. Diagram ini menunjukkan fungsionalitas suatu sistem atau kelas dan bagaimana sistem tersebut berinteraksi dengan dunia luar.

Adapun *Use Case Diagram* dari aplikasi yang dirancang dapat dilihat pada gambar III.5. Dari gambar tersebut dapat dilihat bahwa pengguna pada tampilan awal aplikasi akan melihat beberapa menu yaitu menu Permainan Baru, menu Keterangan dan menu Keluar. Jika pengguna memilih menu Permainan Baru akana menu ini digunakan untuk memulai memainkan *game*, lalu pengguna akan memilih menu *level* dari *game* yang dirancang sesuai keinginan dari pengguna. Pada menu Keterangan berfungsi menampilkan petunjuk / cara memainkan *game* Sudoku ini, jika pengguna memilih menu ini maka dapat melihat cara untuk bermain *game* sudoku, dan menu Keluar digunakan untuk keluar dari *game*.

Berikut ini gambar *Use Case* pada perancangan aplikasi ini :



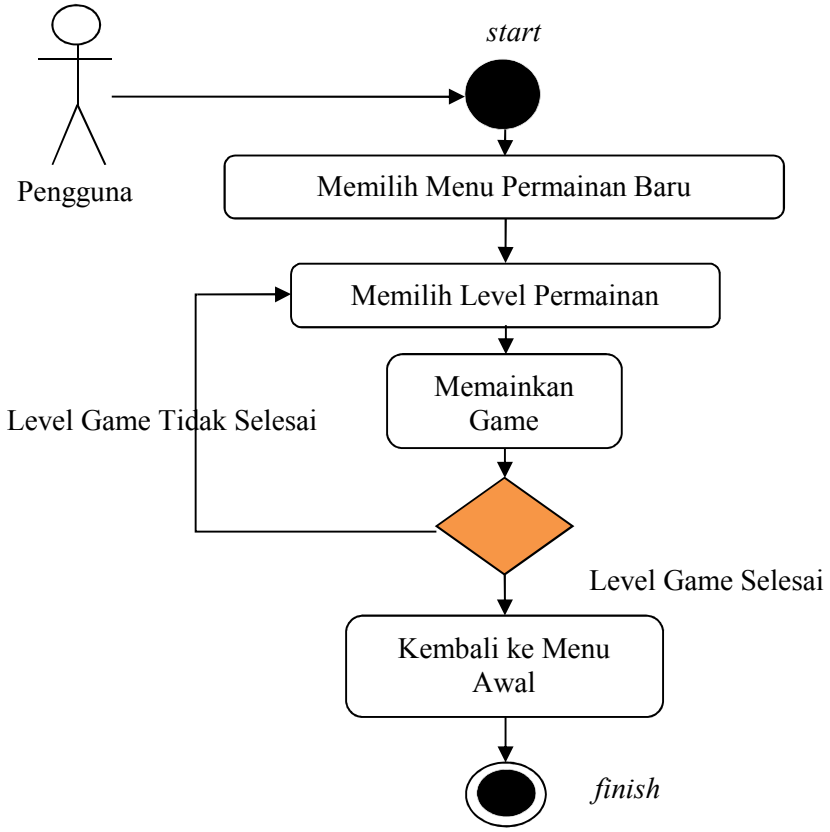

**Gambar III.6. *Use Case Diagram* Aplikasi *Game Sudoku***

### *III.3.2. Activity Diagram*

*Activity Diagram* menggambarkan aktifitas-aktifitas, objek, *state*, transisi *state* dan *event*. Dengan kata lain kegiatan diagram alur kerja menggambarkan

perilaku sistem untuk aktivitas. Berdasarkan *Use Case Diagram*, maka mulailah dibuat *Activity Diagram*. *Activity Diagram* juga sebagai teknik untuk mendiskripsikan logika prosedural, proses bisnis dan aliran kerja dalam banyak kasus. *Activity Diagram* mempunyai peran seperti halnya *flowchart*, akan tetapi perbedaannya dengan *flowchart* adalah *Activity Diagram* bisa mendukung perilaku paralel sedangkan *flowchart* tidak bisa.

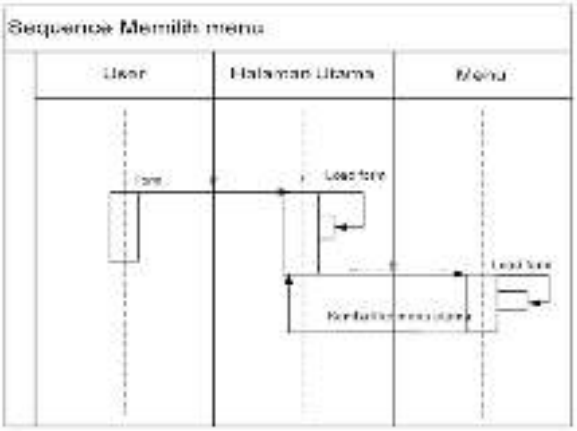
*Activity Diagram* menggambarkan aktivitas yang dilakukan oleh pengguna dalam memainkan *game* ini. Untuk memainkan *game* ini, pengguna harus menyusun angka atau menyelesaikan *puzzle* yang ada hingga seluruh blok di dalam kotak terisi semua dengan angka yang acak. *Activity diagram* dapat dilihat pada gambar III.7.




**Gambar III.7. *Activity Diagram* Memainkan *Game***

### III.3.3. Sequence Diagram

Adapun *Sequence Diagram* pada perancangan aplikasi ini terlihat pada Gambar III.8. berikut.



**Gambar III.8. *Sequence Diagram* Aplikasi *Game* Sudoku**

## III.4. Desain Sistem Secara Detail

### III.4.1. Rancangan *Form* Menu Utama

Rancangan Menu Utama menampilkan 3 (tiga) pilihan menu yaitu: Permainan Baru, Keterangan dan Keluar. Ketika menu *Permainan Baru* dipilih maka akan muncul *form level* permainan yang akan dimainkan, ketika menu *Keterangan* dipilih akan ditampilkan *form* pengaturan keterangan dari *game* dan ketika menu *Keluar* berguna untuk keluar dari program. Rancangan *Form* Menu Utama dapat dilihat pada Gambar III.9.


**Gambar III.9. Rancangan *Form* Menu Utama**

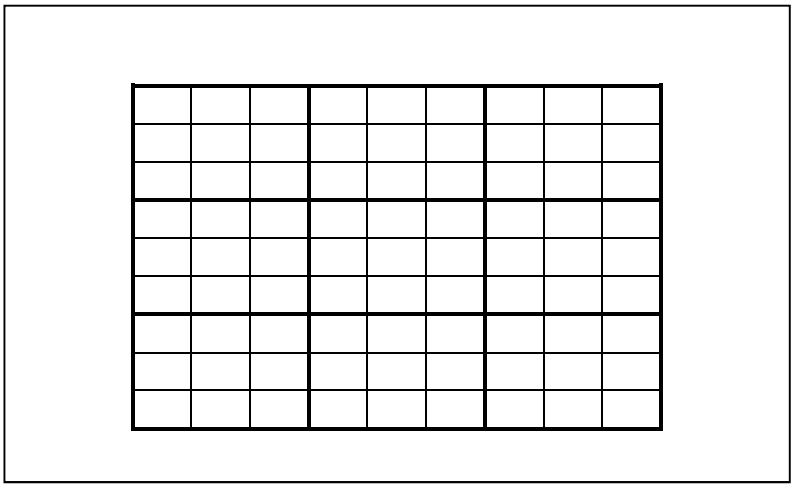
### III.4.2. Rancangan *Form* Keterangan

Rancangan *form Keterangan* berisi keterangan cara bermain *game* dan pembuat *game*. Rancangan *form Keterangan* dapat dilihat pada Gambar III.10.


**Gambar III.10. Rancangan *Form Keterangan***

### III.4.3. Rancangan *Form* Permainan

Rancangan *form* permainan berfungi layar / wadah memainkan *game* dengan menyusun angka acak 1 – 9 hingga semua kotak terisi. Rancangan *form* permainan dapat dilihat pada Gambar III.11.



**Gambar III.11. Rancangan *Form* Permainan**

### III.4.4. Rancangan *Form Level*

Rancangan *form level* menampilkan menu *level* dari permainan yaitu Mudah, Sedang dan Sulit dengan tingkat kesulitan yang berbeda. *Level*

selanjutnya akan muncul bila *level* sebelumnya telah selesai dimainkan. Rancangan *form level* dapat dilihat pada Gambar III.12.


**Gambar III.12. Rancangan *Form Level***