BAB III

ANALISA MASALAH DAN RANCANGAN PROGRAM

III.1. Analisa 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*.

Perancangan aplikasi *game Sudoku* menggunakan *Eclipse Galileo* sebagai desain pengembang aplikasi. *Eclipse* memiliki sifat *Multi-platform* (dapat dijalankan di semua *platform*), *Multi-language* (mendukung pengembangan aplikasi beberapa bahasa pemrograman) dan *Multi-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* seperti : *Samsung Galaxy Mini, Nexian Journey*.

Pada *game* logika ini, *user* harus menyusun angka 1 – 9 pada blok – blok yang sudah disediakan. Blok – blok tersebut disusun ke dalam 9 kotak lagi. Permainan selesai bila semua blok pada kotak telah terisi semua dengan angka 1 – 9. Tujuan permainan Sudoku adalah mengisi sel-sel yang kosong dengan

angka antara 1 dan 9 (setiap sel hanya 1 angka) sesuai dengan persyaratan yaitu angka hanya dapat muncul sekali dalam setiap baris, angka hanya dapat muncul sekali dalam setiap kolom dan angka hanya dapat muncul sekali dalam setiap area/kotak.

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.

III.2. Penerapan Metode Algoritma Genetika

Penulis menggunakan algoritma genetika untuk menyelesaikan permasalahan pada *game* Sudoku. Ini dikarenakan algoritma genetika adalah suatu algoritma optimasi yang meniru mekanisme dari genetika alam. Sebenarnya terdapat banyak sekali variasi dari algoritma genetik di banyak referensi. Pada dasarnya, untuk kasus permutasi, algoritma genetik memiliki enam komponen langkah utama, yaitu pembuatan populasi awal, pengkodean, penghitungan nilai fitness, seleksi, reproduksi, dan mutasi. Penggunaan algoritma genetik untuk menyelesaikan permasalahan *puzzle* sudoku telah banyak dilakukan. Penelitian-

penelitian tersebut merekomendasikan beberapa teknik crossover dan atau mutasi serta pembentukan kromosom.

Dengan menggunakan algoritma genetika, suatu solusi *puzzle* sudoku direpresentasikan dalam sebuah kromosom atau individu dengan struktur tertentu. Kromosom yang terkumpul dalam populasi mengalami berbagai proses, mulai dari seleksi, pindah silang, mutasi, hingga pergantian generasi. Kromosom yang terbaik merupakan solusi dari *puzzle* tersebut.

Aplikasi *game Sudoku* yang akan dirancang 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:

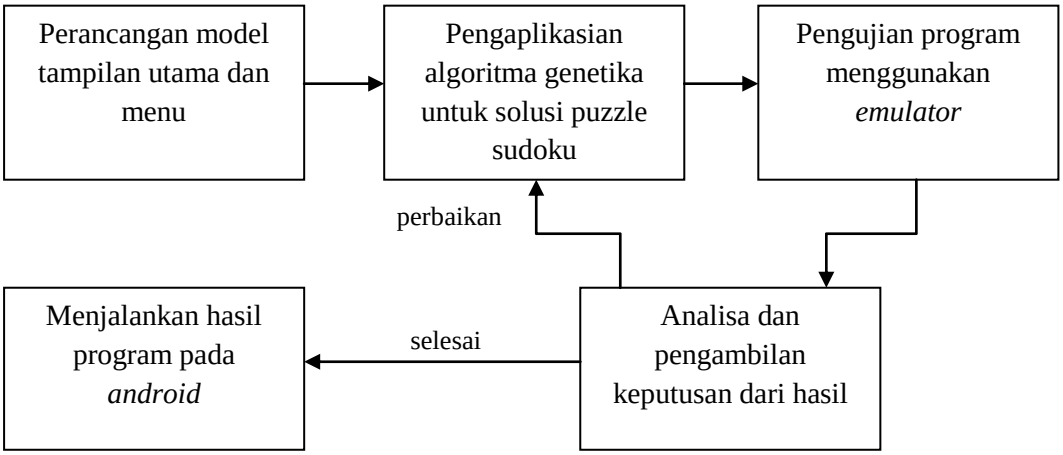
1. Desain Model visualisasi

   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. Solusi *Puzzle Game* Sudoku

Ada dua proses yang dilakukan oleh perangkat lunak ini, yaitu pembangkitan (*generate*) *puzzle* Sudoku dan penyelesaiannya menggunakan Algoritma Genetika. Proses yang pertama adalah proses membuat *puzzle* Sudoku yang valid, yang artinya memiliki solusi. Pengguna permainan kemudian dapat melakukan permainan untuk menyelesaikan *puzzle* tersebut.

Proses kedua adalah penyelesaian *puzzle* tersebut menggunakan Algoritma Genetika. Sebelum memasuki proses tersebut, terlebih dahulu dicari kandidat-

kandidat solusi untuk masing-masing kotak *puzzle*. Hal ini bertujuan untuk mengeliminasi kandidat-kandidat solusi yang tidak mungkin. Contoh, jika suatu kotak telah terisi dengan angka 9, maka angka 9 tersebut dieliminasi dari kandidat pada kotak – kotak di baris, kolom, dan *region* yang sama. Dengan langkah tersebut, diharapkan Algoritma Genetika akan memiliki konvergensi yang baik. Solusi dari suatu *puzzle* didapatkan jika suatu kromosom memiliki nilai *fitness* yang sempurna, karena kromosom dengan nilai *fitness* di bawah nilai *fitness* sempurna berarti bukan solusi dari *puzzle* Sudoku.

Pada Algoritma Genetika, satu kromosom adalah satu kandidat solusi *puzzle*. Karena nilai yang mungkin pada solusi adalah antara 1 sampai dengan 9, maka representasi kromosom yang digunakan adalah deretan bilangan bulat (integer) dengan panjang kromosom sama dengan jumlah kotak yang kosong pada *puzzle*. Nilai *fitness* diambil dari panjang kromosom dikurangi dengan jumlah *gen error*. *Gen error* adalah gen yang berisi angka yang berulang pada baris, kolom, atau *region*. Metode seleksi yang digunakan adalah *Roulette-Wheel*.

III.2.1.1. Pembangkitan Puzzle

Pembangkitan *puzzle* adalah proses pembuatan *puzzle* Sudoku yang valid. *Puzzle* sudoku direpresentasikan dengan *array* dua dimensi. *Puzzle* ini harus memenuhi aturan dalam permainan Sudoku, yaitu tidak boleh ada angka yang berulang pada baris, kolom, dan region. *Puzzle* ini kemudian dikosongkan beberapa kotaknya yang jumlahnya tergantung ada tingkat kesulitan permainan.

Tekniknya, kandidat-kandidat pada setiap blok direpresentasikan dengan sebuah *List*. Pada kondisi awal, *list* tersebut berisi angka 1 hingga 9. Karena papan berukuran 9x9, penyimpanan *cell-cell* tersebut dilakukan menggunakan *array* dua dimensi dengan ukuran 9x9. Kemudian *looping* dilakukan untuk semua *cell* tersebut, dari kiri ke kanan, atas ke bawah. Untuk masing-masing *cell*, diambil angka secara random dari kandidat-kandidat yang tersedia untuk *cell* tersebut. Kemudian angka yang terpilih tersebut dihapuskan dari kandidat pada *cell-cell* pada baris, kolom, dan *region* yang sama. Dengan teknik ini, satu *puzzle* Sudoku yang valid dapat dibangkitkan dalam waktu yang cukup cepat (< 1 detik).

Adapun pseudocode untuk permasalahan di atas adalah :

1. Lakukan langkah 2 hingga 9 selama tidak terjadi *error*.
2. Buat *array of List* berukuran 9x9.
3. Isi setiap list dengan angka 1 hingga 9.
4. Lakukan langkah 5 hingga 9 untuk semua *cell*.
5. Ambil satu angka dari kandidat secara acak.
6. Hapus angka yang terpilih dari kandidat pada baris yang sama.
7. Hapus angka yang terpilih dari kandidat pada kolom yang sama.
8. Hapus angka yang terpilih dari kandidat pada *region* yang sama.
9. Jika kandidat pada *cell* kosong, ulangi dari langkah 2 (terjadi *error*).
10. Kosongkan secara random sejumlah *cell* sesuai dengan tingkat kesulitan.

III.2.1.2. Representasi Kromosom

Kromosom adalah representasi solusi dari suatu masalah pada Algoritma Genetika. Pada Sudoku, kromosom adalah representasi suatu solusi dari *puzzle* yang berisi angka-angka yang akan diisikan pada kotak-kotak yang kosong sehingga panjang kromosom adalah sama dengan jumlah kotak yang kosong pada *puzzle*. Nilai untuk masing-masing gen pada kromosom diambil secara acak dari kandidat-kandidat pada setiap *cell*. Kandidat nilai untuk setiap kromosom adalah angka 1 hingga 9. Pada pemrograman, kromosom direpresentasikan dengan *array of integer (int[]).*

4	7	8	9	4	3	6	1	7	4	6	5	2	5	6	9	9	7	4	7

Gambar III.2. Contoh Bentuk Kromosom

III.2.1.3. Eliminasi Kandidat

Eliminasi kandidat dilakukan untuk menghilangkan kandidat-kandidat solusi yang tidak mungkin. Kandidat yang tidak mungkin adalah kandidat yang memiliki angka yang berulang pada baris, kolom, atau *region* yang sama. Adapun *pseduocode* untuk permasalahan ini adalah :

1. Lakukan langkah 2 hingga 8 untuk semua *cell* kosong.
2. Buat *List*, isi dengan angka 1 hingga 9.
3. Lakukan langkah 3 untuk *cell-cell* berisi pada baris yang sama.
4. Hapus angka yang ditemui dari *list*.
5. Lakukan langkah 6 untuk *cell-cell* berisi pada kolom yang sama.
6. Hapus angka yang ditemui dari *list*.

7. Lakukan langkah 8 untuk *cell-cell* berisi pada *region* yang sama.
8. Hapus angka yang ditemui dari *list*.

III.2.1.4. *Naked Single*

Naked Single adalah salah satu teknik dalam penyelesaian *puzzle* Sudoku secara manual. Tujuan dari teknik ini adalah mencari *cell-cell* kosong yang hanya memiliki satu kandidat (kandidat tunggal) yang artinya *cell* tersebut hanya mungkin diisikan dengan angka tersebut.

Teknik ini biasanya dilakukan dengan menuliskan kandidat-kandidat setiap *cell* secara manual. Teknik ini merupakan langkah awal dalam penyelesaian karena teknik ini adalah teknik yang paling sederhana. Pada perangkat lunak ini, pencarian dilakukan dengan memanfaatkan hasil dari eliminasi kandidat. *Cell-cell* dengan kandidat tunggal akan diisi langsung dengan nilai kandidat tunggal tersebut. Adapun *pseudocode* dari langkah di atas adalah :

1. Lakukan langkah 2 hingga untuk semua *cell* kosong.
2. Hitung jumlah kandidat.
3. Jika jumlah kandidat sama dengan 1, isi *cell* dengan kandidat tersebut
4. Jika tidak, maka *cell* tersebut tidak dalam kondisi *Naked Single*.

III.2.1.5. Fungsi *Fitness*

Untuk menghitung nilai *fitness* masing – masing kromosom, terlebih dahulu dihitung jumlah angka yang berulang pada baris, kolom, dan *region* pada *puzzle* setelah angka-angka pada kromosom diisikan pada kotak-kotak yang

kosong pada *puzzle*. Fungsi *fitness* yang digunakan cukup sederhana, yaitu adalah pengurangan panjang kromosom dengan jumlah angka yang berulang sehingga kromosom yang merupakan solusi *puzzle* adalah kromosom yang mempunyai nilai *fitness* sebesar panjang kromosom.

Contoh perhitungan untuk *puzzle* di bawah ini :

3	4			8	7		1	9
6		1		9		8	3	7
7	8		3	1	6		5	4
8	6	7			5		9	2
2		4	9		8	5		3
9	5			7	2	1	8	6
4		2	7		1		6	8
1		8	6	2	3	9	4	5
5	3	6	8	4	9	7	2	1

Gambar III.3. Contoh *Puzzle* Sudoku

Jika diisikan kromosom :

5	2	2	2	2	4	9	2	1	3	4	1	6	7	3	4	7	5	3	7

Gambar III.4. Contoh Kromosom *Puzzle* Sudoku

Maka akan didapatkan :

3	4	**5**	**2**	8	7	**2**	1	9
6	**2**	1	**2**	9	**4**	8	3	7
7	8	**9**	3	1	6	**2**	5	4
8	6	7	**1**	**3**	5	**4**	9	2
2	**1**	4	9	**6**	8	5	**7**	3
9	5	**3**	**4**	7	2	1	8	6
4	**7**	2	7	**5**	1	**3**	6	8
1	**1**	8	6	2	3	9	4	5
5	3	6	8	4	9	7	2	1

Gambar III.5. Kotak – Kotak yang *Error* pada *Puzzle*

Angka berwarna merah adalah *cell-cell* yang error. Maka nilai *fitness* untuk kromosom tersebut adalah :

Fitness = panjang kromosom – jumlah error

= 20 – 7

= 13

Solusi dari suatu *puzzle* Sudoku adalah kromosom dengan nilai *fitness* sempurna, yaitu sebanyak jumlah kotak yang kosong.

III.2.1.6. Seleksi

Metode seleksi yang digunakan adalah metode *Roulette-Wheel* yang membuat romosom-kromosom dengan nilai *fitness* tinggi memiliki kemungkinan terpilih yang tinggi pula.

Probabilitas keterpilihan setiap kromosom ditentukan dengan cara membagi nilai *fitness* kromosom tersebut dengan total nilai *fitness*. Untuk itu, yang terlebih dahulu harus dihitung adalah total nilai *fitness*. Kemudian dilakukan pembagian masing-masing nilai *fitness* dengan total *fitness* tersebut. Nilai tersebut adalah probabilitas setiap kromosom yang berkisar antara nol dan satu.

Kemudian pada proses pemutaran roda *roulette*, teknisnya juga sederhana. Suatu bilangan acak dibangkitkan, dengan jangkauan nilai antara nol dan satu. Nilai acak tersebut kemudian dibandingkan dengan nilai probabilitas masing-masing kromosom. Pembandingan dilakukan dari kromosom yang terkecil. Kromosom yang memiliki nilai fitness yang lebih besar atau sama dengan nilai acak akan menjadi individu yang terpilih.

Tabel III.1. Contoh Probabilitas Seleksi untuk Setiap Kromosom

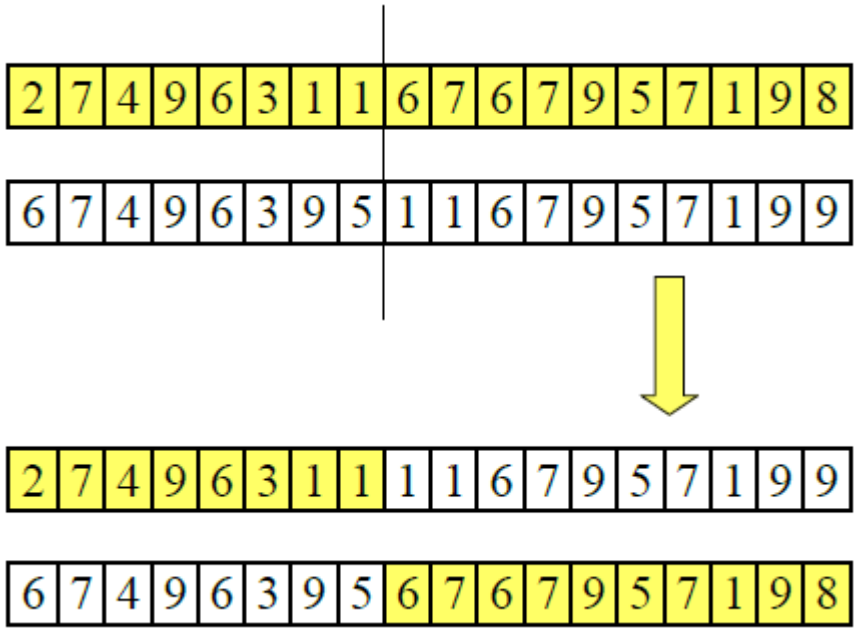
Kromosom	Fitness	Probabilitas	Nilai Probabilitas
C1	2	1/4	0,25
C2	1	1/8	0,125
C3	1	1/8	0,125
C4	4	1/2	0,5
Total	**8**	**1**	**1**

III.2.1.7. Elitisme

Elitisme dilakukan untuk mempertahankan individu terbaik karena ada kemungkinan bahwa individu terbaik itu tidak terpilih pada proses seleksi. Jika itu terjadi, maka bisa saja Algoritma Genetika kehilangan konvergensinya. Implementasi mekanisme elitisme pada pemrograman cukup sederhana yaitu pilih dua individu dengan nilai *fitness* tertinggi.

III.2.1.8. Pindah Silang

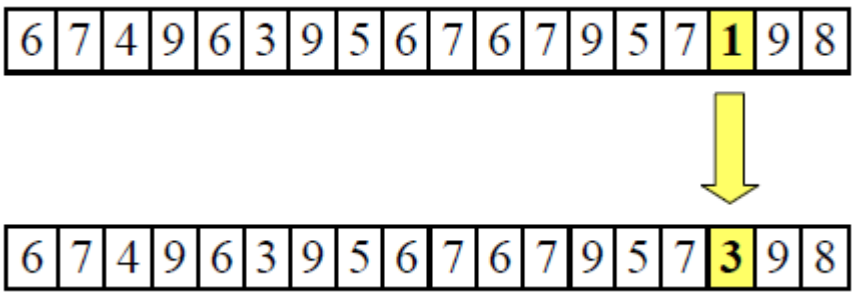
Metode pindah silang yang digunakan adalah pindah silang dengan satu titik. Pertukaran gen kemudian dilakukan pada kedua kromosom mulai titik gen tertentu sampai akhir kromosom.



Gambar III.6. *Crossover Single Point*

III.2.1.9. Mutasi

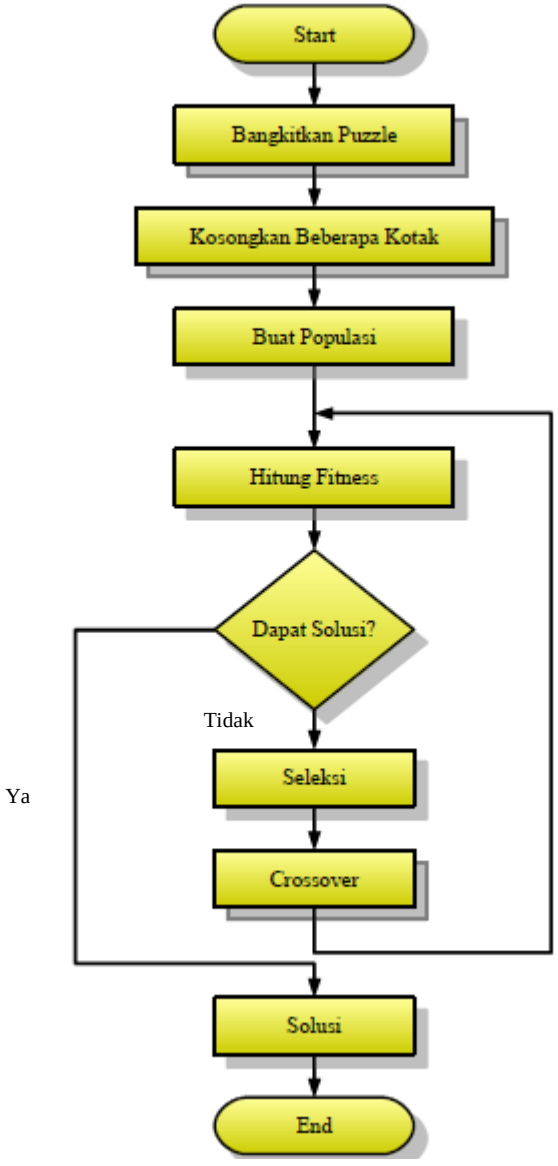
Mutasi dilakukan satu titik, yaitu dengan mengganti nilai pada gen tertentu dengan nilai yang lain yang mungkin diisikan pada kotak yang bersesuaian.



Gambar III.7. Mutasi *Single Point*

III.2.1.10. *Flow Chart* Algoritma Genetika

Adapun *flow chart* Algoritma Genetika pada aplikasi ini adalah sebagai berikut :



Gambar III.8. *Flow Chart* Algoritma Genetika pada Apikasi *Game* Sudoku

III.3. Perancangan

Perancangan aplikasi *game* ini meliputi rancangan menu utama, yang didalamnya terdapat *list* untuk Permainan Baru, Keterangan dan Keluar.

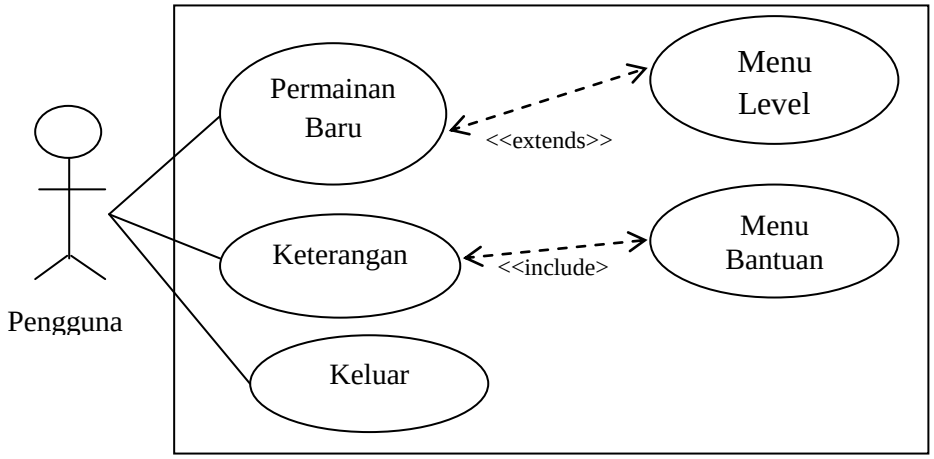
III.3.1. *Use Case Diagram*

Use Case merupakan permodelan untuk kelakukan (*behavior*) sistem. *Use case* digunakan untuk memodelkan dan menyatakan unit fungsi/layanan yang disediakan oleh sistem (bagian *system*, *subsistem* atau *class*) ke pemakai (*user*). Use case dapat dilingkupi dengan batasan sistem yang diberi *label* nama sistem dan dapat menyediakan hasil yang dapat diukur ke pemakai atau sistem eksternal.

Use case diagram adalah penggambaran sistem dari sudut pandang pengguna sistem tersebut (*user*), sehingga pembuatan *use case* lebih dititikberatkan pada fungsionalitas yang ada pada sistem, bukan berdasarkan alur atau urutan kejadian.

Adapun *use case* dari aplikasi yang dirancang dapat dilihat pada gambar III.9. 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 :



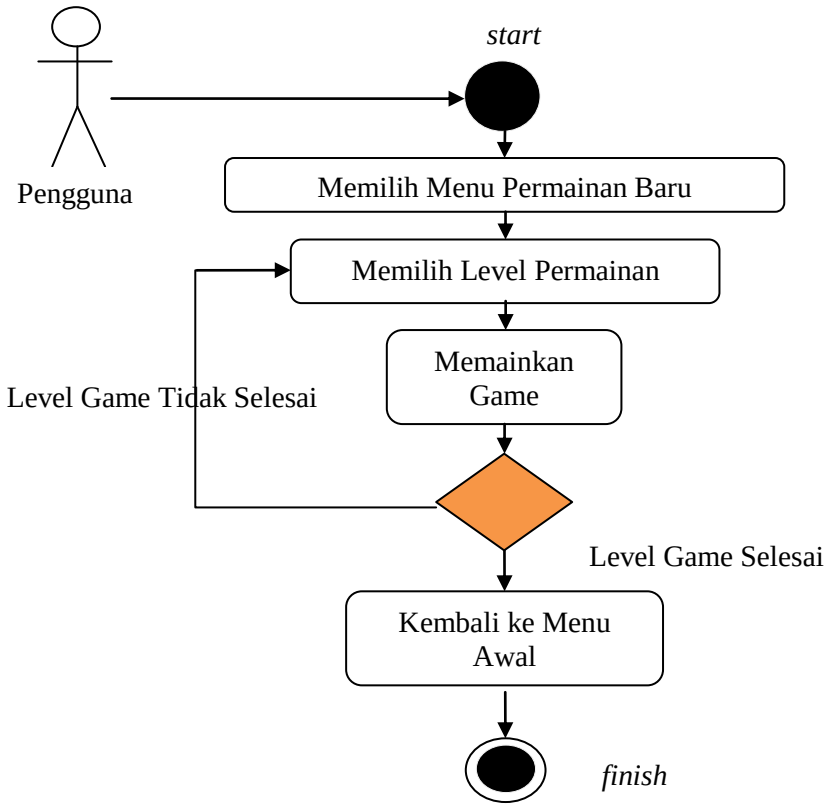
Gambar III.9. *Use Case Diagram* Aplikasi *Game Sudoku*

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.10.



Gambar III.10. *Activity Diagram* Memainkan *Game*

III.3.3. Perancangan Tampilan

Layar atau model dibuat untuk menggambarkan bentuk dari *game* yang akan dirancang. Model ataupun bentuk *game* yang akan dirancang dapat dilihat sebagai berikut :

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.11.

Gambar III.11. Rancangan Form Menu Utama

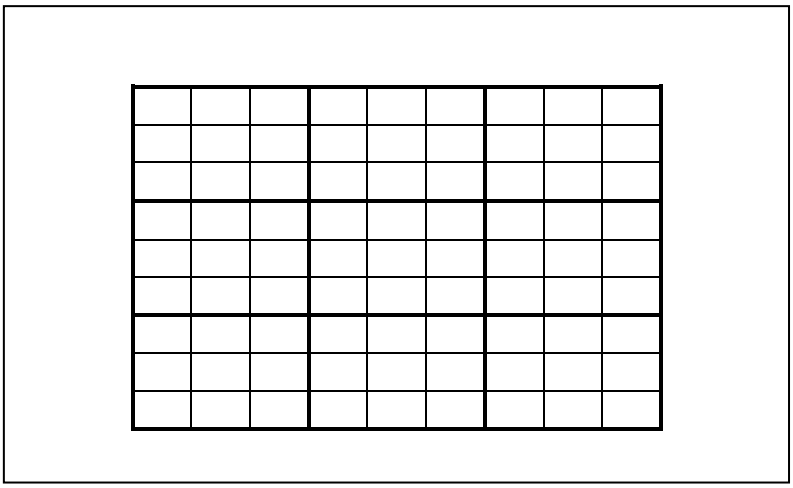
2. Rancangan *Form Keterangan*

Rancangan *form Keterangan* berisi keterangan cara bermain *game* dan pembuat *game*. Rancangan *form Keterangan* dapat dilihat pada Gambar III.12.

Gambar III.12. Rancangan *Form Keterangan*

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.13.



Gambar III.13. Rancangan *Form* Permainan

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.14.

Gambar III.14. Rancangan *Form Level*