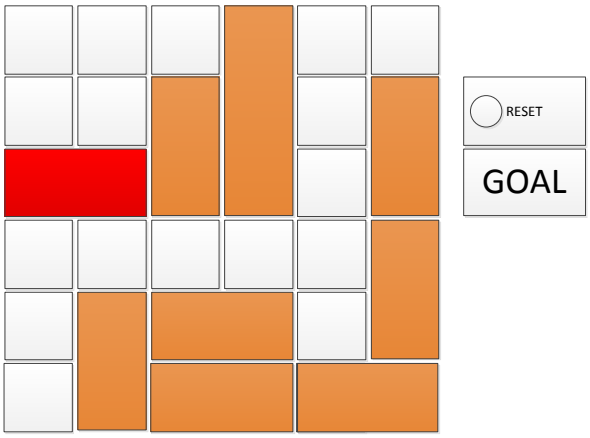
BAB 3
ANALISIS DAN PERANCANGAN SISTEM

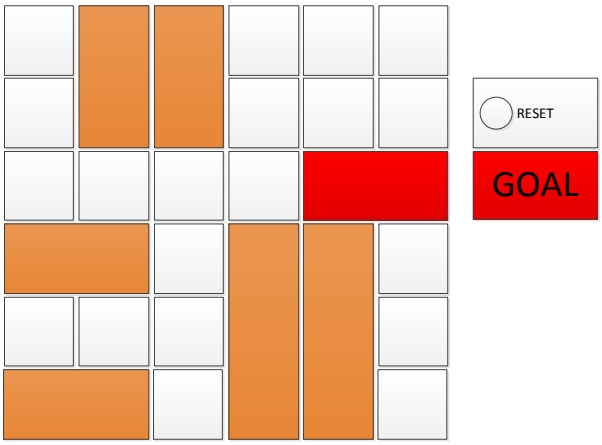
3.1 Analisis Masalah

Unblockme adalah permainan yang bersifat logika Permainan ini adalah permainan menemukan jalan atau solusi dengan cara menggeser balok penghalang pada kolom-kolom kosong sampai ditemukan solusi dari balok User menuju Goal. Permainan Unblockme bersifat unik ,karena setiap penggeseran balok saling berkaitan dan balok tersebut mempunyai gerak yang terbatas. Untuk menyelesaikan permainan unblockme ini, harus menggeser sejumlah balok penghalang (Cokelat) sampai terbentuk jalan dari balok tujuan (Merah) menuju Goal.



Gambar 3.1 Contoh penyelesaian Game *Unblockme*

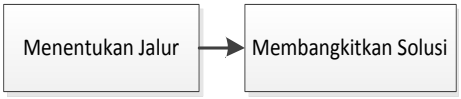
Oleh karena itu dibutuhkan teknik untuk menyelesaikan masalah yang bersifat unik dengan sistem *random* yang akan menghasilkan solusi yang akan saling terkait dari soal balok dengan balok lainnya dan mencari solusi dalam menyelesaikan jawaban pada permainan *Unblockme* dengan beberapa syarat didalamnya. Dengan menggunakan prinsip dasarnya, algoritma *backtracking* mencari segala kemungkinan solusi dengan menerapkan ruang keadaan dalam bentuk pohon solusi hingga menghasilkan solusi yang layak. Kelayakan solusi pada permainan *Unblockme* didefinisikan dengan sampainya balok User (Merah) pada Goal.



Gambar 3.2 State akhir dari Penyelesaian game *Unblockme.*

3.2 Analisis Data Masukan

Analisis *game* yang akan diteliti pada *game Unblockme* ini adalah dengan mencari rute atau solusi dan menyelesaikannya. Pada dasarnya *Unblockme* ini disetiap Balok saling terkait satu sama lainnya. Oleh karena itu, algoritrma *backtracking* akan diterapkan pada *Unblockme* ini. Berikut merupakan alur membangkitkan soal permainan serta penyelesaian Permainan :



Gambar 3.3 Proses menentukan jalur.

Berikut ini adalah langkah-langkah penelesaian dari game unblockme menggunakan algoritma backtracking, penjelasannya bisa dilihat dalam langkah sebagai berikut :

1. Menentukan jalur

Pada tahapan menentukan jalur ini, kotak pada papan diinisialisasikan terlebih dahulu kedalam bentuk matrix yang bertujuan mengidentifikasi balok pada papan. Tujuan dari inisialisasi ini adalah supaya mudah menidentifikasi atau menetukan apakah kotak pada papan berisi balok atau tidak, untuk lebih jelasnya bisa dilihat pada gambar berikut :

0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

Gambar 3.4 Kotak sebelum di inisialisasi.

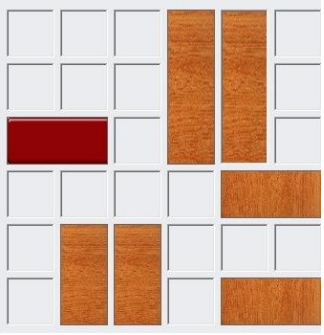
Pada gambar 3.4 diatas dapat dilihat nilai matrix kotak sebelum diinisialisasi, pada kotak tersebut diberi nilai nol (0) karena kotak tidak terisi balok atau bisa dikatakan kotak kosong.

Sebelum memberikan nilai inisialisasi pada kotak, akan di jelaskan inisial dari nilai yang dimasukan kedalam kotak dan inisialisasi tersebut dijabarkan kedalam bentuk matrix. Inisialisasi balok berdasarkan angka dapat dijabarkan sebagai berikut :

0 : Kotak kosong

1 : Balok User

2,3,4,5,6,7 : Balok penghalang



Gambar 3.5 kotak pada balok.

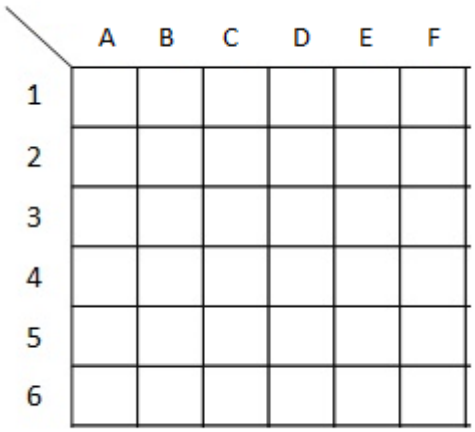
Pada penjelasan tersebut, dijelaskan keadaan pada kolom kotak setelah diberikan inisialisasi , dimana nilai 0 adalah kotak yang kosong, nilai 1 balok user dan 2,3,4,5,6,7 adalah balok penghalang. Pada gambar 3.5 masih dalam bentuk balok , jika diinisialisakan dalam bentuk matrik akan menjadi seperti gambar 3.6 sebagai berikut :

0	0	0	4	5	0
0	0	0	4	5	0
1	1	0	4	5	0
0	0	0	0	6	6
0	2	3	0	0	0
0	2	3	0	7	7

Gambar 3.6 inisialisasi matrix

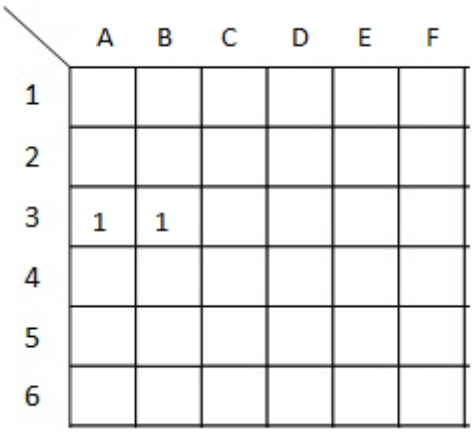
Gada gambar 3.6 di atas dijelaskan gambaran papan permainan setelah diisikan nilai matrix pada kolom inisialisasi.

Didalam proses penginisialisasian ,setiap kotak akan diberikan nilai indeks yang akan di gambarkan dalam koordinat dalam papan permainan. Titik koordinat pada papan akan digambarkan sebagai berikut :



Gambar 3.7koordinat Sumbu.

Pada gambar 3.7 menjelaskan keadaan awal kotak yang masih kosong sebelum terjadinya inisialisasi balok permainan yang akan input dengan angka. Contoh, pada gambar berikut ini :



Gambar 3.8 Angka dimasukan sebagai inisialisasi balok

Pada gambar 3.8 , dapat dilihat sumbu (A,3),(B,3) diisikan oleh nilai 1 yang diinisialisasikan sebagai balok yang terdapat dalam kotak. Untuk lebih jelasnya masukan nilai yang menyatakan balok bisa dilihat pada tabel berikut :

Tabel 3.1 Tabel nilai input koordinat.

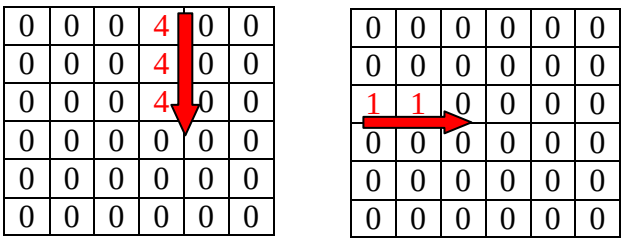
Koordinat	Nilai	Koordinat	Nilai	Koordinat	Nilai
A1	0	C1	0	E1	5
A2	0	C2	0	E2	5
A3	1	C3	0	E3	5
A4	0	C4	0	E4	6
A5	0	C5	3	E5	0
A6	0	C6	3	E6	7
Koordinat	Nilai	Koordinat	Nilai	Koordinat	Nilai
B1	0	D1	4	F1	0
B2	0	D2	4	F2	0
B3	1	D3	4	F3	0
B4	0	D4	0	F4	6
B5	2	D5	0	F5	0
B6	2	D6	0	F6	7

Pada tabel diatas dapat dilihat nilai-nilai koordinat pada papan yang diisi dengan angka-angka yang diinisialisasikan sebagai balok-balok pada papan permainan. Setelah nilai-nilai tersebut dimasukan kedalam papan sesuai koordinat yang sudah di tentukan. Pada gambar 3.9 terlihat hasil dari pemetaan nilai sesuai koordinat yang telah ditentukan Maka didapatkan hasil sebagai berikut :

	A	B	C	D	E	F
1	0	0	0	4	5	0
2	0	0	0	4	5	0
3	1	1	0	4	5	0
4	0	0	0	0	6	6
5	0	2	3	0	0	0
6	0	2	3	0	7	7

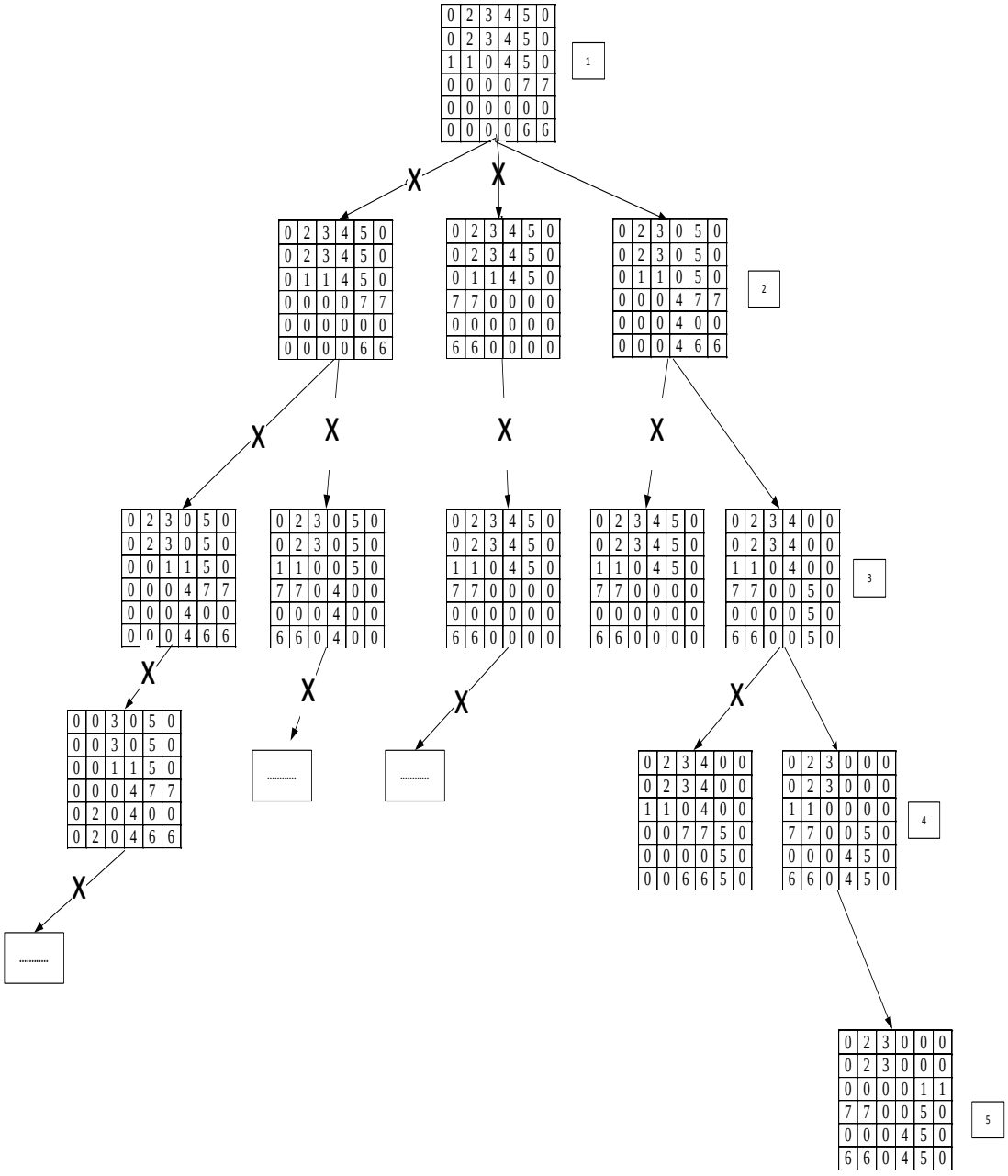
Gambar 3.9 Pemetaan nilai sesuai Koordinat.

Untuk tahap selanjutnya kotak-kotak yang diinisialisasi tersebut ditentukan alur vertikal atau horizontal pergerakannya. Untuk mengetahui pergerakannya dilihat dari urutan digit yang sama, contohnya pada papan permainan digit 4,4,4 terletak horizontal, maka hanya berlaku penggeseran horizontal, dan begitu juga dengan dengan digit yang berurutan secara vertikal ,maka pergerakan atau pergeseran yang berlaku vertikal. Pergeseran digit yang telah diinisialisasi tersebut dapat digambarkan dalam tabel sebagai berikut :



Gambar 3.10 arah pergerakan Horizontal (kiri) dan Vertikal (kanan).

Tujuan dari gambar 3.10 adalah untuk memperlihatkan arah pergerakan balok berbentuk matrix pada papan permainan. Berikut ini adalah contoh kasus pada penyelesaian game Unblockme dalam bentuk pohon solusi, bisa dilihat pada gambar berikut :



Gambar 3.11 Solusi layak dalam pohon solusi .

Pada gambar 3.11 di atas dalam menemukan solusi yang layak digambarkan kedalam pohon solusi. Tujuan dari tahapan-tahapan tersebut adalah untuk mengeluarkan balok user (balok merah), dengan kata lain jalur mana yg di tempuh oleh balok tersebut sehingga bisa keluar. Tahapan dalam inisialisasi ini bisa di jelaskan dengan tahapan pada gambar berikut:

0	0	0	4	5	0
0	0	0	4	5	0
1	1	0	4	5	0
0	0	0	0	6	6
0	2	3	0	0	0
0	2	3	0	7	7

Gambar 3.12 Menentukan Jalur.

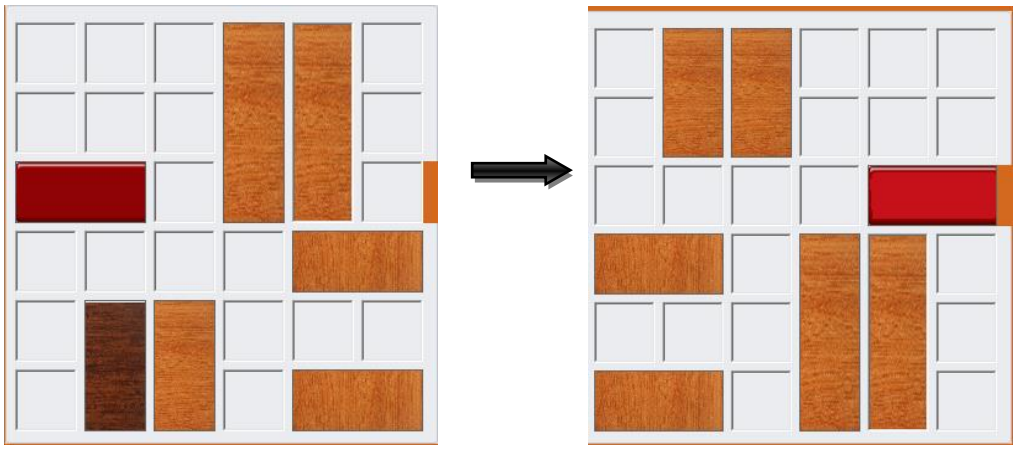
Gambar 3.12 menjelaskan tentang menentukan jalur, pada kasus ini balok yang akan dipindahkan berada pada koordinat (A3,B3) diinisialisasikan dengan angka 1, oleh karena itu koordinat yang harus dikosongkan adalah koordinat (C3,D3,E3,F3) yang bertujuan untuk menentukan jalur mana yang harus dikosongkan untuk dilewati oleh balok User.

0	0	0	4	5	0
0	0	0	4	5	0
1	1	0	4	5	0
0	0	0	0	6	6
0	2	3	0	0	0
0	2	3	0	7	7

→

0	2	3	0	0	0
0	2	3	0	0	0
1	1	0	0	0	0
6	6	0	4	5	0
0	0	0	4	5	0
7	7	0	4	5	0

Gambar 3.13 State awal (kiri) dan akhir (kanan) dalam bentuk matrix.



Gambar 3.14 State awal (kiri) dan akhir (kanan) dalam bentu balok.

Gambar diatas menjelaskan kotak setelah jalur ditentukan dan kemungkinan dibangkitkan. Pada setiap balok telah ditetapkan arah pergerakannya horizontal atau vertikal jalurnya tersebut..pada balok user atau balok merah telah ditetapkan goal statenya pada papan permainan. Pada penjelasan gambar di atas juga dapat dilihat titik koordinat yang berubah pada state state akhir.

3.3 Analisis Algoritma

Analisis algoritma yang akan diteliti pada *Unblockme* ini adalah untuk menganalisis cara kerja algoritma *backtracking* terhadap kasus permainan *Unblockme*, bagaimana cara kerja algoritma *backtracking* dalam mencari solusi serta penyelesaian tahap penyelesaian pada *game Unblockme*.

3.3.1 Analisis Algoritma *Backtracking* Penyelesaian Game

Sebelum melakukan pencarian solusi dengan graf, permainan ini harus dimodelkan menjadi matrix berukuran 6 × 6. Dimana tempat kosong ditandai dengan angka 0, balok merah (balok yang ingin dikeluarkan) ditandai dengan angka 1, dan balok lainnya ditandai dengan angka 2 s/d 18. Terdapat batasan pada banyak balok yaitu sebanyak 18. Dengan asumsi kasus terburuk, setiap balok merupakan balok horizontal berukuran 2 × 1 maka banyak balok maksimal adalah sebanyak luas permainan dibagi luas balok (sebanyak 36/2 atau 18) balok, termasuk dengan balok yang ingin dikeluarkan. Sebagai contoh, untuk merepresentasi keadaan permainan seperti pada Gambar menggunakan matriks dapat dilihat sebagai berikut :

0	**0**	**0**	**2**	**3**	**3**
0	**0**	**4**	**2**	**5**	**0**
1	**1**	**4**	**6**	**5**	**0**
7	**7**	**0**	**6**	**8**	**0**
9	**10**	**10**	**10**	**8**	**0**
9	**11**	**11**	**11**	**0**	**0**

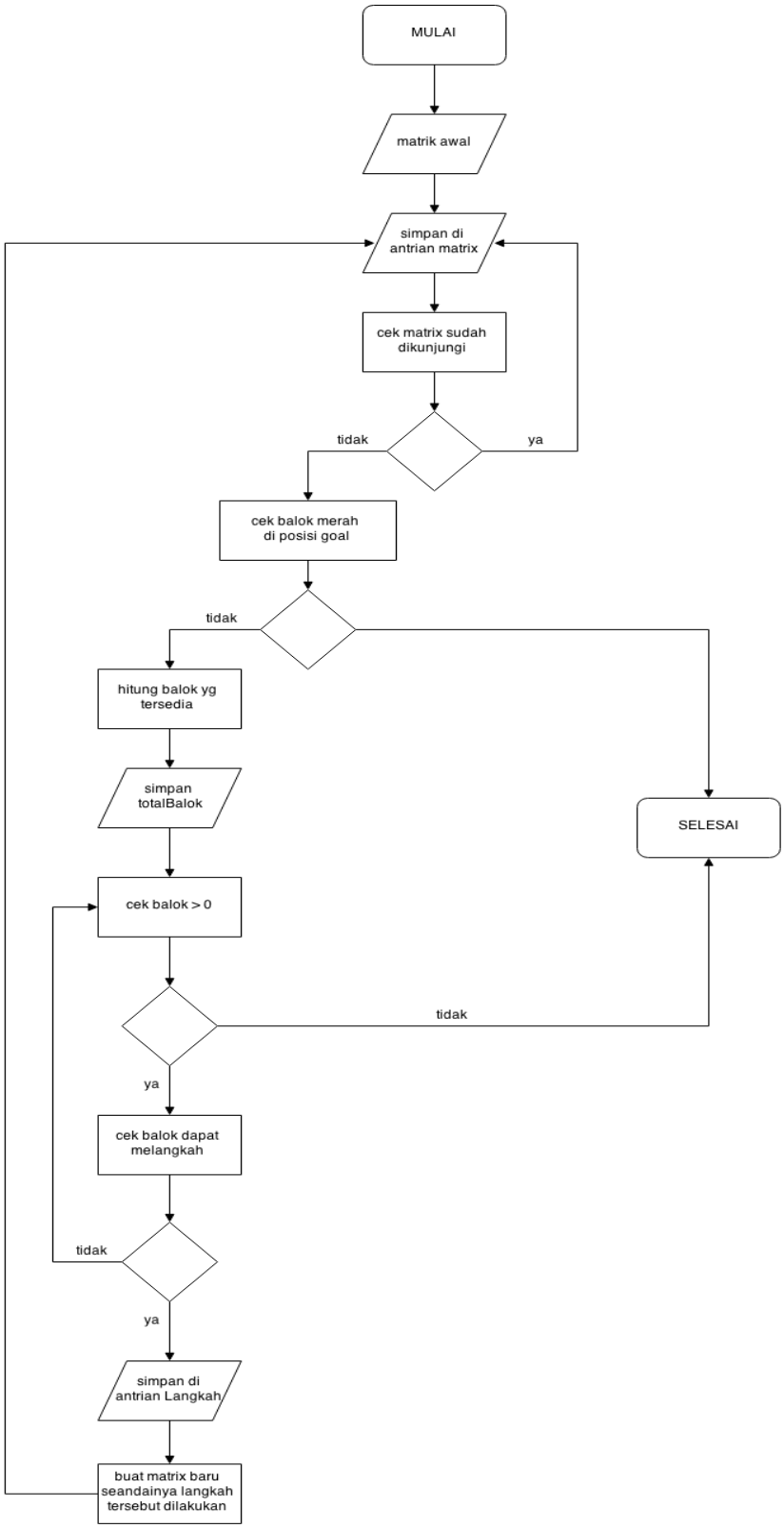
Gambar 3.15 Gambar kotak yang telah diinisialkan.

Pada gambar di atas akan di jelaskan langkah-langkah yang dilakukan dari awal permainan. Sebelum melakukan permainan , langkah-langkah harus dicatat dalam bentuk matrix agar terlihat pergerakan balok pada saat game dimainkan.

Untuk menandai apakah suatu matriks keadaan permainan sudah pernah dikunjungi atau belum bisa digunakan sebuah *Map* yang memetakan Matriks ke sebuah *Boolean*. *Map* memetakan Matriks ke *Boolean* dengan menyimpan Elemen dalam bentuk pasangan Matriks dan *Boolean*. Dapat dilihat apakah suatu Matriks sudah dipetakan ke sebuah *Boolean* dengan mudah. Jika sudah terdapat Pemeteaan dari Matriks ke *Boolean* maka Matriks tersebut sudah dikunjungi. *Boolean* dalam pemakaian ini selalu bernilai *true* jika ada. Selain cara di atas, ada cara lain untuk mengetahui apakah Matriks sudah pernah dikunjungi adalah dengan menggunakan *hash table* atau *set*. Pemakaian *hash table* akan membuat rumit persoalan karena harus diimplementasikan fungsi *hash* terlebih dahulu, sedangkan pemakaian *set* hanya dapat dilakukan untuk segelintir bahasa pemrograman. Setelah dihasilkan Node yang merupakan solusi dari permainan, digunakanlah langkah-langkah dari Node tersebut untuk mengkonstruksi ulang transisi matriks permainan dari awal sampai ditemukannya solusi.

3.3.2 Membangkitkan Solusi

Tiap 1 balok memiliki kemungkinan langkah yang banyak, bisa jadi >=1, dalam aplikasi semua kemungkinan-kemungkinan langkah di simpan dalam array. Contohnya pada matrix di posisi awal . Tiap-tiap balok dibuat kemungkinan langkahnya dan dimasukan kedalam satu daftar langkah-langkah .Setiap 1 langkah dari balok yg manapun, direkam keseluruhan posisi balok (matrix). 1 matrix ini yang disebut membangkitkan solusi. sampai didapat matrix dengan posisi dimana jalur yang telah di tentukan sama dengan jalur pada gambar 3.5 untuk balok user. Untuk tahapan lebih jelasnya dapat dilihat dari alur flowchat sebagai berikut :



Gambar 3.16 *Flowchart* pembangkitan solusi.

Flowchart tersebut merupakan alur dari proses pembangkitan solusi atau generate solution pada program. Alur flowchart tersebut dapat di tuliskan dengan langkah seperti berikut :

1. Mulai.
2. Inisialisasi matrix awal dari posisi awal balok.
3. Simpan pada antrian Pemetaan (Matrix).
4. Sebelum dilakukan pembangkitan langkah, di cek terlebih dahulu apakah matrix (pemetaan posisi balok) sudah pernah di lakukan proses pembangkitan atau belum. jika sudah kembali ke antrian pemetaan untuk ambil antrian selanjutnya.
5. Jika belum dilakukan, maka cek kembali apakah pada pemetaan matrix tersebut balok merah (di tandai dengan angka 1) sudah ada pada posisi goal/ belum).
6. Jika ya maka pencarian akan dinyatakan "Selesai".
7. Jika belum ,maka lakukan pembangkit solusi berupa langkah- langkah yang dapat dilakukan oleh tiap balok.
8. Masukan kedalam antrian langkah.
9. Setiap dilakukan 1 langkah kemungkinan di atas, petakan ke dalam matrix dari posisi awal. (yang berganti hanya balok yg melakukan langkah dari dan ke).
10. Masukan pemetaan tersebut kedalam antrian Pemetaan(Matrix).
11. Proses ini dilakukan berulang sampai mendapat posisi yg diinginkan (posisi goal).

Pada proses tersebut algoritma backtrackingnya terlihat dari mulai no :

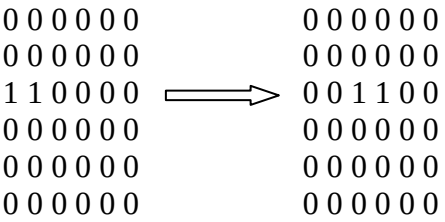
7 -> pencarian kemungkinan langkah / disebut membangkitkan solusi, (didapat banyak langkah)

8 -> simpan ke antrian langkah(Al)

9 -> lakukan langkah sebanyak antrian.

Contoh : 1 langkah (asumsikan angka 1 adalah balok merah)

matrix awal(Ma) matrik setelah dilakukan 1 langkah(Mf)



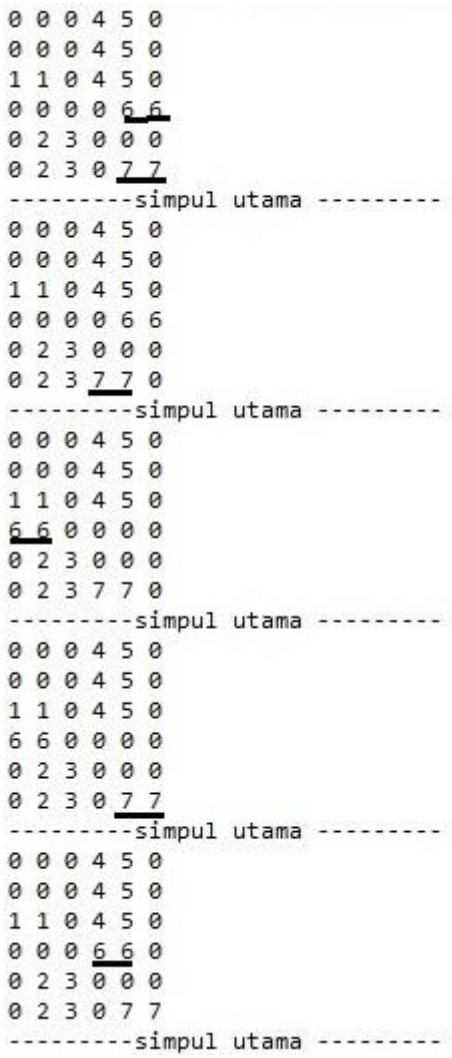
simpan kedalam antrian matrix.

antrian Matrix (Am) / jika d analogikan sebagai pohon Am ini adalah node nya. jadi 1 node = 1 matrix[a,b] .

setelah di lakukan langkah, akan dilakukan lagi pembangkitan langkah/solusi dari matrix yang baru (Mf), terus menerus dilakukan hingga d dapat matrix dengan posisi angka 1 (balok merah) berada pada posisi. setiap 1 langkah dilakukan, pada balok juga di lakukan perubahan pada titik koordinat baloknya.

Untuk aturannya dilakukan terlebih dahulu pada balok no 1. jadi langkah yg pertama di ambil oleh balok merah (angka 1 pda matrix). selanjutnya bebas atau acak (tanpa menggunakan random).

Pada proses tersebut terjadi pergeseran balok , pergeseran tersebut berupa inisialisasi dalam bentuk matrix. Pada proses tersebut dapat dilihat pada gambar berikut :



Gambar 3.17 Pergesaeran balok berupa Matrix

Pada gambar tersebut dapat dilihat pergerakan setiap balok , jadi setiap pergeseran atau pergerakan balok deirekam dan ditampilkan kedalam output seperti gambar 3.17.

3.4 Analisis Kebutuhan Perangkat Lunak

Perangkat lunak (*software*) merupakan hal terpenting dalam mendukung kinerja sebuah sistem. Perangkat lunak digunakan dalam sebuah sistem untuk memberikan perangkat keras (*hardware*) sebuah perintah agar perangkat keras melakukan apa yang diinginkan oleh pemain.

Berikut ini, kebutuhan perangkat lunak untuk menerapkan algoritma *backtracking* pada *game Unblockme* pada tabel 3.2.

Tabel 3.2 Spesifikasi perangkat lunak

Spesifikasi Perangkat Lunak
Windows XP, Windows 7, Windows 8
Visual Studio

3.4.1 Analisis Kebutuhan Perangkat Keras

Agar aplikasi dapat berjalan dengan baik, maka dibutuhkan perangkat keras (*hardware*) yang sesuai dengan kebutuhan aplikasi. Tabel 3.3 adalah spesifikasi minimum perangkat keras yang dibutuhkan agar dapat menjalankan aplikasi permainan *Unblockme* secara optimal.

Tabel 3.3 Spesifikasi perangkat keras

Komputer	**Spesifikasi Perangkat Keras**
Prosesor	Prosesor 1,8 Ghz
Resolusi Layar	800 x 600 *pixel*
Memori	Memori 128 Mb
Hardisk	*Hardisk* 20 Gb
Keyboard dan *mouse*	Standar
Sound card	Standar

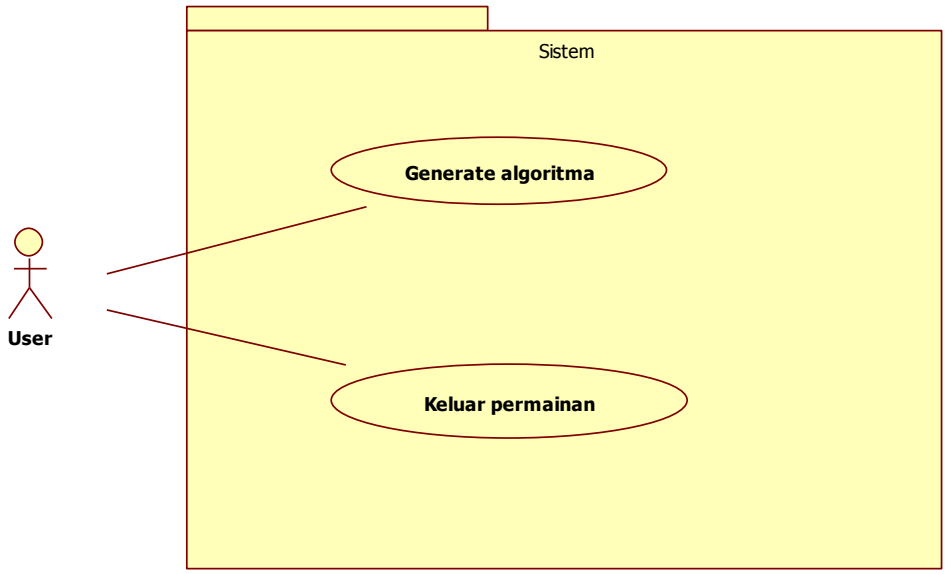
3.4.2 Analisis Kebutuhan Non Fungsional

Analisis kebutuhan non fungsional merupakan analisis yang dibutuhkan untuk menentukan spesifikasi kebutuhan sistem. Spesifikasi yang meliputi elemen atau komponen-komponen yang dibutuhkan untuk sistem yang akan dibangun hingga sistem tersebut diimplementasikan. Analisis kebutuhan ini menentukan spesifikasi masukan yang diperlukan sistem, keluaran yang akan dihasilkan sistem dan proses yang dibutuhkan untuk mengolah masukan sehingga menghasilkan suatu keluaran yang diinginkan.

3.4.3 Analisis Kebutuhan Fungsional

Pemodelan yang digunakan pada *game* yang akan dibuat adalah analisis perancangan dan pengembangan perangkat lunak berbasis objek, dimana alat yang akan digunakan untuk pengembangan perangkat lunak yaitu pemodelan *Unified Modeling Language* (UML).

3.4.3.1 *Use case Diagram*



Gambar 3.18 *Use case diagram*

Definisi *use case* berfungsi untuk menjelaskan fungsi dari *use case* yang terdapat pada diagram use case. Definisi *use case* dijelaskan pada tabel 3.3.

Tabel 3.3 Definisi *use case*

No	*Use Case*	Keterangan
1.	Generate Solution	Proses menyelesaikan game menggunakan algoritma.
2.	Keluar	Proses untuk keluar dari permainan

3.4.3.2 *Use Case Scenario*

Skenario *use case* merupakan penjelasan mengenai alur kejadian yang di dalamnya menjelaskan urutan interaksi antara aktor dengan *use case* tersebut. Adapun skenario *use case* digambarkan sebagai berikut.

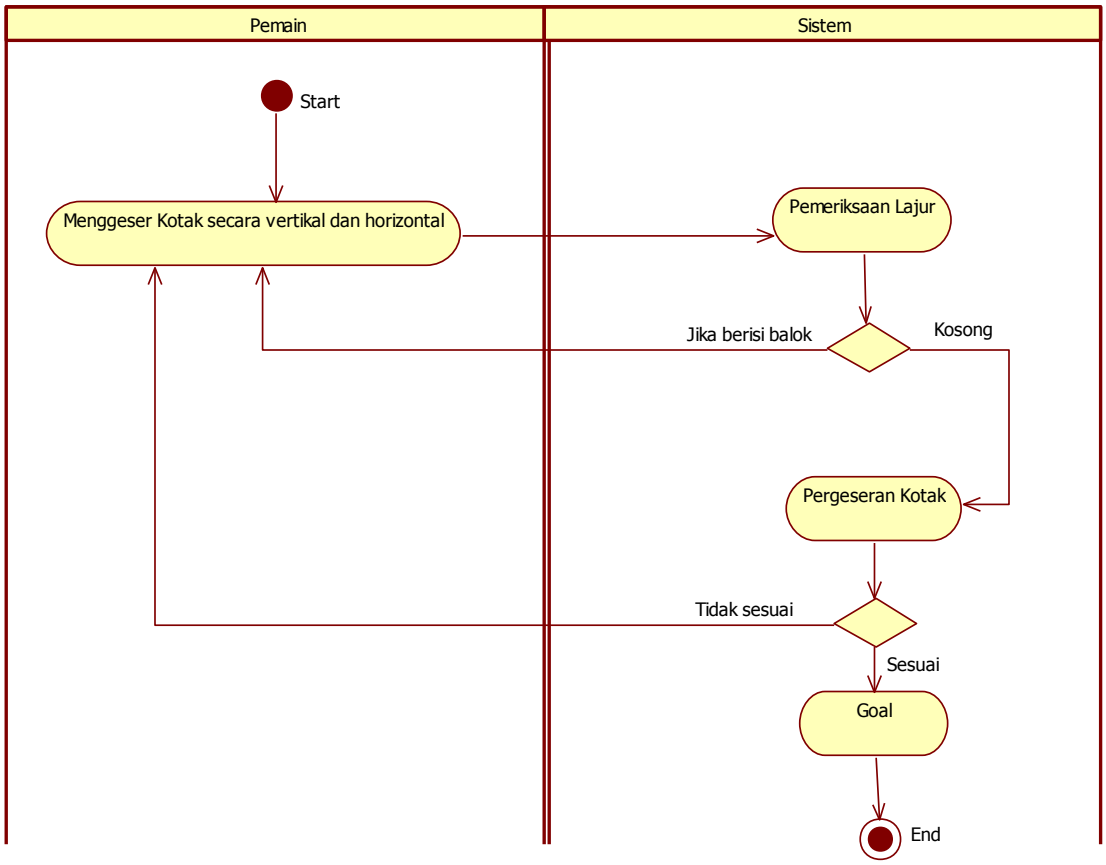
Tabel 3.4 Tabel memainkan permainan

Identifikasi	
Nomor	1
Nama	Memainkan Permainan
Tujuan	Memainkan permainan
Deskripsi	Proses untuk memainkan permainan
Aktor	Pemain
Skenario Utama	
Kondisi Awal	Pemain berada di permainan
Aksi Aktor	Reaksi Sistem
	1. Menggeser balok permainan
	2. Menginisialisasi Papan Permainan
	3. Melakukan *Backtracking*
	4. Melakukan *Cek Balok*
	5. Menemukan solusi
	6. Menampilkan sukses

3.4.3.3 *Activity Diagram*

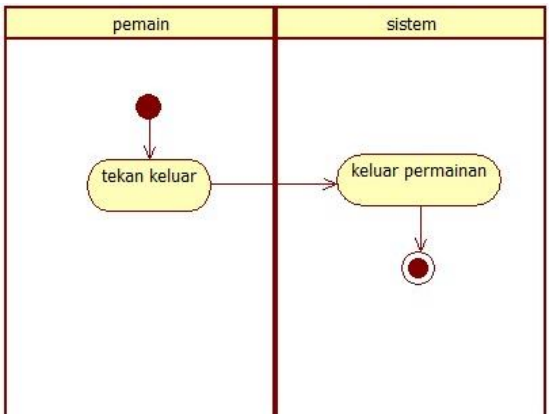
Activity diagram merupakan gambaran alur aktivitas pada permainan yang sedang dirancang, bagaimana masing-masing alur aktivitas berawal, *decision* yang mungkin terjadi, dan bagaimana berakhir. *Activity* diagram dijelaskan melalui gambar berikut:

1. *Activity* memainkan permainan



Gambar 3.19 *Activity* memainkan permainan

2. *Activity* keluar permainan



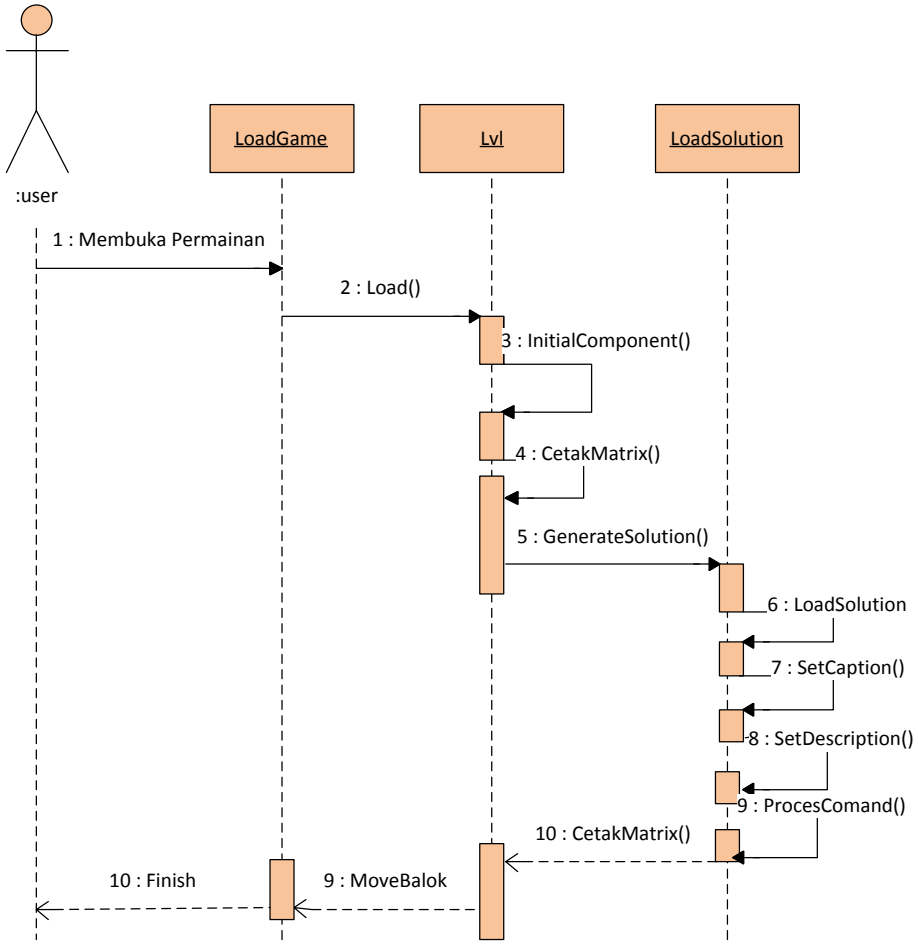
Gambar 3.20 *Activity* keluar permainan

3.4.3.4 *Sequence Diagram*

Sequence diagram menggambarkan tindakan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. *Sequence diagram* dapat dilihat pada gambar berikut.

1. *Squence diagram* memainkan permainan

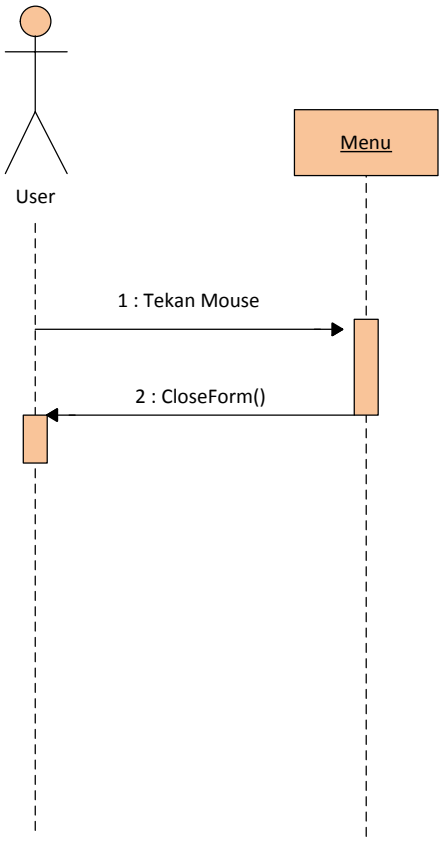
Sequence diagram memainkan permainan menunjukkan rangkaian interaksi, pesan yang dikirim antar objek pada proses dimana game dimulai.



Gambar 3.21 *Squence diagram* memainkan permainan

2. *Squence diagram* keluar permainan

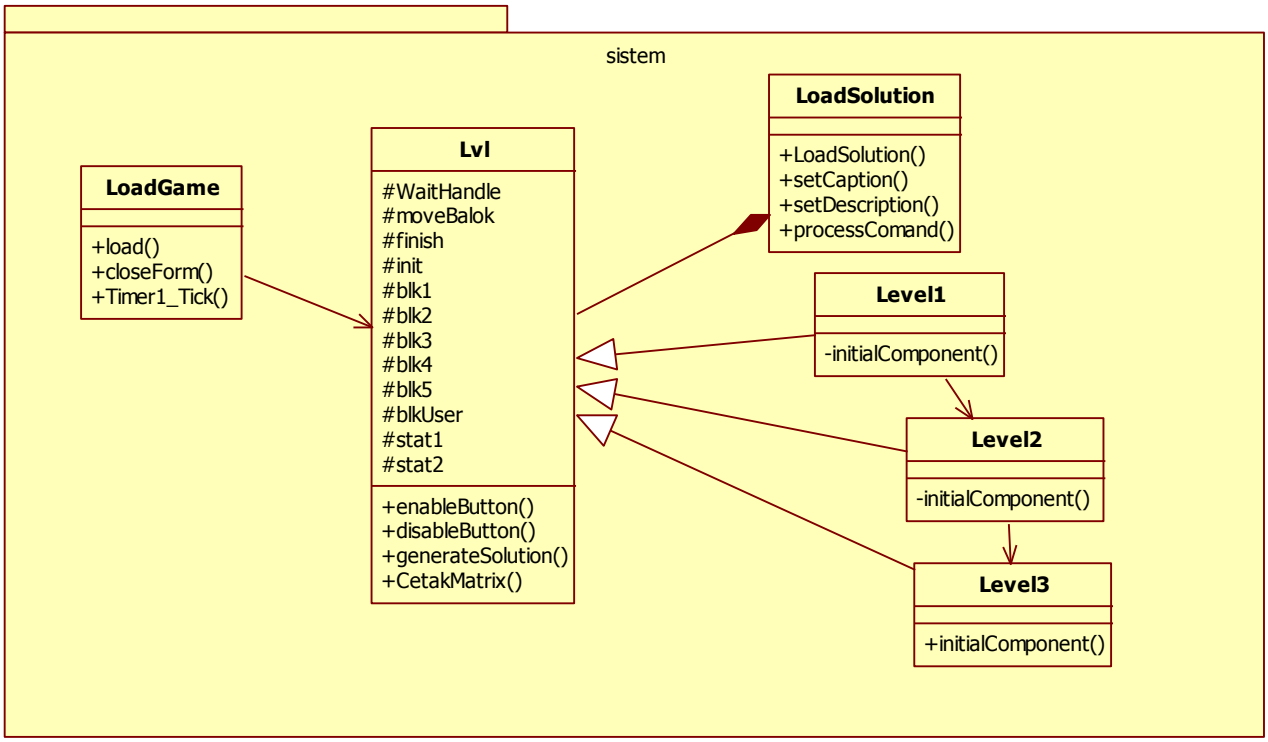
Sequence diagram keluar menunjukkan rangkaian interaksi, pesan yang dikirim antar objek pada proses dimana notifikasi akan ditampilkan.



Gambar 3.22 *Squence diagram* keluar permainan

3.4.3.5 *Class Diagram*

Class diagram merupakan gambaran sistem berupa relasi antar kelas-kelas ataupun paket-paketnya. *Class diagram* dijelaskan melalui gambar berikut.

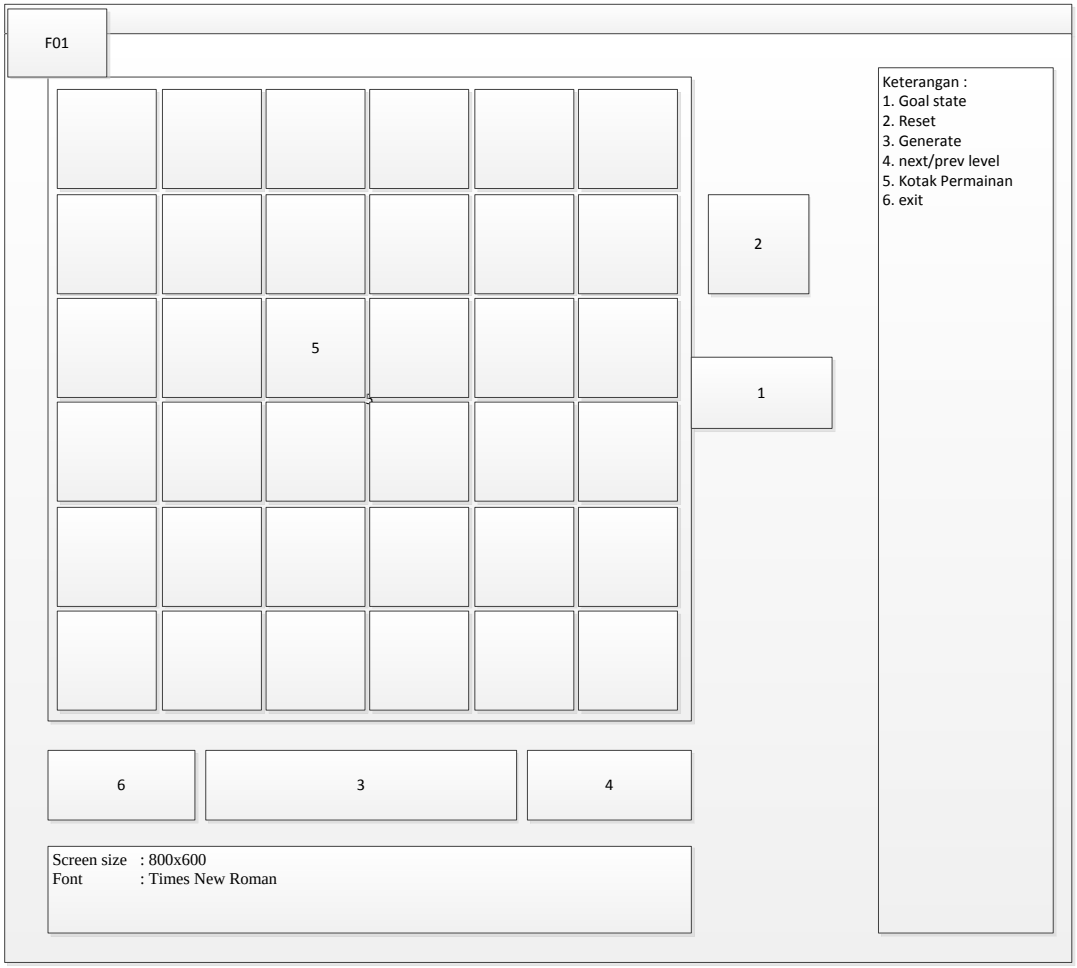


Gambar 3.23 *Class diagram Game Unblockme*

3.4.3.6 Perancangan Antarmuka

Perancangan antarmuka bertujuan untuk memberikan gambaran tentang aplikasi yang akan dibangun, sehingga akan mempermudah dalam mengimplementasikan aplikasi serta akan memudahkan pembuatan aplikasi. Antarmuka permainan merupakan halaman yang muncul pertama kali pada saat memainkan game. Berikut perancangan antarmuka permainan :

1. Perancangan antarmuka permainan.



Gambar 3.24 Antarmuka permainan

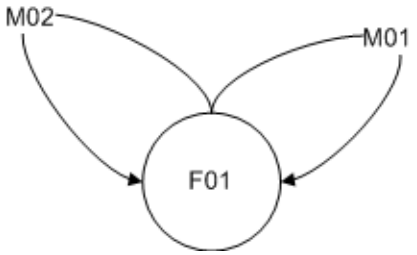
2. Pesan ketika permaianan terselesaikan.



Gambar 3.25 Pesan ketika permaianan terselesaikan

3.4.3.7 Jaringan Semantik

Untuk mengetahui keterhubungan dari setiap tampilan yang dibuat dapat menggunakan jaringan semantik.



Gambar 3.27 Jaringan semantic.