BAB III
ANALISIS DAN KEBUTUHAN ALGORITMA

III.1. Analisis Masalah

Analisis masalah pada pembangunan *game* penyelamatan korban banjir yaitu sebagai berikut:

1. Bagaimana cara menyelamatkan korban yang terjebak oleh banjir?
2. Bagaimana mengimplementasikan algoritma *Moving Target Search* pada game penyelamatan korban banjir?
3. Kurangnya pengetahuan dan keterampilan yang di miliki oleh masyarakat dalam bidang penyelamatan korban banjir.

III.2. Analisis *Game*

Pada bagian ini menjelaskan tentang game yang akan dikembangkan. Penjelasan terutama untuk bagian-bagian dalam *game* yang akan diterapkan algoritma. Dalam *game* ini terdapat beberapa karakter. Adapun karakter beserta perilakunya adalah sebagai berikut :

1. Tim Penyelamat
   a. Jumlah Karakter Tim Penyelamat

      Jumlah dari karakter tim penyelamat ada tiga orang.

   b. Perilaku Karakter Tim Penyelamat

      Perilaku dari karakter tim penyelamat yaitu dua orang bertugas untuk mendayung perahu karet dan satu orang bertugas untuk mengatur arah jalan yang akan ditempuh. Secara keseluruhan tim penyelamat bertugas untuk mencari dan menyelamatkan korban yang ada di tiap-tiap rumah mereka masing-masing kemudian membawa korban ke tempat yang aman.

2. Korban
    a. Jumlah Karakter Korban

       Jumlah korban yang terkena bencana banjir ada dua orang.
    b. Perilaku Karakter Korban

       Perilaku dari karakter koran yaitu hanya menunggu bantuan dari tim penyelamat dan mencari tempat aman agar tidak terbawa arus air yang deras.

III.3. Analisis Masukan

Pada bagian ini menjelaskan data masukan yang diperlukan algoritma yang akan diperoleh dari dalam *game*. Terdapat dua variabel pada *game* ini. Adapun variabel yang digunakan adalah :

a. Jarak

   Jarak adalah angka yang menunjukkan seberapa jauh suatu benda berubah posisi melalui suatu lintasan tertentu. Dalam *game* ini jarak yang dihitung yaitu jarak Xawal dan Yawal dari tim penyelamat sampai ketemu dengan Xakhir dan Yakhir dari kordinat tim penyelamat.

   Masing-masing dari karakter tersebut mempunyai titik kordinat tertentu, dimana korban mempunyai titik kordinat statik atau diam dan tim sars mempunyai titik kordinat yang dinamis atau bergerak mencari korban. Tim penyelamat akan mencapai *goal* jika titik kordinat Xawal = Xakhir dan Yawal = Yakhir, artinya tim penyelamat akan berhasil menemukan korban.

b. Jumlah Korban

   Dalam *game* ini jumlah korban yang akan dicari oleh tim penyelamat ada dua orang korban. Karakter dari korban yang pertama berada di atas atap rumah menunggu datangnya tim penyelamat, dan yang kedua terbawa hanyut oleh arus air. Korban yang terbawa hanyut oleh arus air akan di selamatkan terlebih dahulu oleh tim penyelamat.

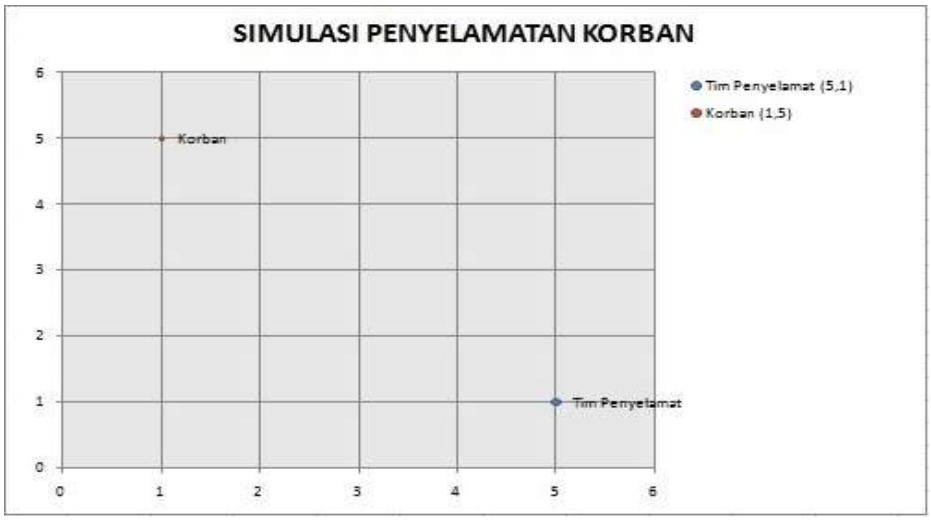
III.4. Analisis Metode / Algoritma

Algoritma *Moving Target Search* didefinisikan sebagai salah satu algoritma pemecahan masalah yang berbasis pencarian terhadap target dan juga memiliki jangkauan deteksi atau daerah dimana dapat mendeteksi target.

III.4.1. Penerapan Algoritma *Moving Target Search*

Penerapan Algortima *Moving Target Search* pada *Game* Penyelamatan Korban Banjir secara ringkas langkah demi langkahnya adalah sebagai berikut :

Algoritma *Moving Target Search* pada *game* Penyelamatan Korban Banjir diterapkan pada Tim Penyelamat atau disebut juga NPC (*Non Playable Character*) yaitu dimana algoritma tersebut tidak diterapkan pada pemain. Pada algoritma ini menerapkan Tim Penyelamat sebagai pencarian target yang bergerak untuk menyelamatkan target yaitu korban. Contoh penerapan Tim Penyelamat dan korban pada algoritma *Moving Target Search* bisa dilihat pada gambar dibawah ini.



Gambar 3.1. Penerapan Tim Penyelamat dan Korban pada Algoritma *MTS*

1. Diawali dengan menempatkan tim penyelamat atau titik biru pada starting point dengan koordinat X (5,1), kemudian memasukkan seluruh simpul yang bertetangga dan tidak memilik atribut penghalang dengan tim penyelamat ke dalam tujuan selanjutnya untuk bergerak.

Tetangga 1: koordinat X’ (4,1)

Tetangga 2: koordinat X’ (4,2)

Tetangga 3: koordinat X’ (5,2)

Tetangga 4: koordinat X’ (6,2)

Tetangga 5: koordinat X’ (6,1)

Tetangga 6: koordinat X’ (6,0)

Tetangga 7: koordinat X’ (5,0)

Tetangga 8: koordinat X’ (4,0)

2. Menempatkan korban atau target pada starting point dengan koordinat Y (1,5), dan koordinat korban yang sekarang akan menjadi *node* tujuan akhir. Bila korban bergerak maka koordinat tujuan akhirnya akan berubah menjadi koordinat Y’ yang akan dituju oleh korban.
3. Setelah menempatkan posisi awal dari masing-masing karakter, kemudian mencari nilai F terkecil dari *node-node* tetangga tim penyelamat untuk tujuan selanjutnya dengan menggunakan fungsi heuristic.

$$h_diagonal(n) = \min(abs(n.x - goal.x) + abs(n.y - goal.y))$$

$$h_orthogonal(n) = (abs(n.x - goal.x) + abs(n.y - goal.y))$$

$$h(n) = h_diagonal(n) + (h_orthogonal(n) - (2 * h_diagonal(n)))$$

4. Jika korban dalam keadaan diam, maka hitung semua tetangga yang akan dituju oleh tim penyelamat dengan menggunakan perhitungan 2.1.

$$F(x',y) = H(x',y) + 1$$

5. Jika korban dalam keadaan bergerak, maka hitung semua tetangga yang akan dituju oleh tim penyelamat dengan menggunakan perhitungan 2.2.

$$F(x,y') = H(x,y') - 1$$

Misalkan :

a. Koordinat X’ = (4,1)

   Nilai n.x = 4

Nilai n.y = 1

node tujuan (1,5), sehingga goal.x = 1, goal.y = 5

h_diagonal(n) = min (abs(n.x - goal.x) + abs(n.y – goal.y))
= min (abs(4 – 1) + abs(1 – 5))
= min (abs(3) + (-4))
= -3 + -4
= -7

h_orthogonal(n) = (abs(n.x - goal.x) + abs(n.y – goal.y))
= (abs(4 – 1) + abs(1 – 5))
= ((3) + abs(-4))
= 3 + 4
= 7

h(n) = h_diagonal(n) + (h_orthogonal (n) – (2 * h_diagonal(n)))
= -7 + (7 – (2 * (-7)))
= -7 + (7 – (-14))
= -7 + 21
= 14

F(x,y') = h(x,y') – 1
= 14 – 1
= 13

b. Koordinat X' = (4,2)

Nilai n.x = 4

Nilai n.y = 2

node tujuan (1,5), sehingga goal.x = 1, goal.y = 5

h_diagonal(n)= min (abs(n.x - goal.x) + abs(n.y – goal.y))
= min (abs(4 – 1) + abs(2 – 5))
= min (abs(3) + (-3))
= -3 + -3
= -6

h_orthogonal(n) = (abs(n.x - goal.x) + abs(n.y – goal.y))
= (abs(4 – 1) + abs(2 – 5))
= ((3) + abs(-3))
= 3 + 3
= 6

h(n) = h_diagonal(n) + (h_orthogonal (n) – (2 * h_diagonal(n)))
= -6 + (6 – (2 * (-6)))
= -6 + (6 – (-12))
= -6 + 18
= 12

F(x,y') = h(x,y') – 1
= 12 – 1
= 11

c. Koordinat X' = (5,2)

Nilai n.x = 5

Nilai n.y = 2

node tujuan (1,5), sehingga goal.x = 1, goal.y = 5

h_diagonal(n) = min (abs(n.x - goal.x) + abs(n.y – goal.y))
= min (abs(5 – 1) + abs(2 – 5))
= min (abs(4) + (-3))
= -4 + -3
= -7

h_orthogonal(n) = (abs(n.x - goal.x) + abs(n.y – goal.y))
= (abs(5 – 1) + abs(2 – 5))
= ((4) + abs(-3))
= 4 + 3
= 7

h(n) = h_diagonal(n) + (h_orthogonal (n) – (2 * h_diagonal(n)))
= -7 + (7 – (2 * (-7)))
= -7 + (7 – (-14))

= -7 + 21

= 14

F(x,y') = h(x,y') – 1

= 14 – 1

= 13

d. Koordinat X' = (6,2)

Nilai n.x = 6

Nilai n.y = 2

node tujuan (1,5), sehingga goal.x = 1, goal.y = 5

h_diagonal(n) = min (abs(n.x - goal.x) + abs(n.y – goal.y))

= min (abs(6 – 1) + abs(2 – 5))

= min (abs(5) + (-3))

= -5 + -3

= -8

h_orthogonal(n) = (abs(n.x - goal.x) + abs(n.y – goal.y))

= (abs(6 – 1) + abs(2 – 5))

= ((5) + abs(-3))

= 5 + 3

= 8

h(n) = h_diagonal(n) + (h_orthogonal (n) – (2 * h_diagonal(n)))

= -8 + (8 – (2 * (-8)))

= -8 + (8 – (-16))

= -8 + 24

= 16

F(x,y') = h(x,y') – 1

= 16 – 1

= 15

e. Koordinat X’ = (6,1)

Nilai n.x = 6

Nilai n.y = 1

node tujuan (1,5), sehingga goal.x = 1, goal.y = 5

h_diagonal(n) = min (abs(n.x - goal.x) + abs(n.y – goal.y))

= min (abs(6 – 1) + abs(1 – 5))

= min (abs(5) + (-4))

= -5 + -4

= -9

h_orthogonal(n) = (abs(n.x - goal.x) + abs(n.y – goal.y))

= (abs(6 – 1) + abs(1 – 5))

= ((5) + abs(-4))

= 5 + 4

= 9

h(n) = h_diagonal(n) + (h_orthogonal (n) – (2 * h_diagonal(n)))

= -9 + (9 – (2 * (-9)))

= -9 + (9 – (-18))

= -9 + 27

= 18

F(x,y’) = h(x,y’) – 1

= 18 – 1

= 17

f. Koordinat X’ = (6,0)

Nilai n.x = 6

Nilai n.y = 0

node tujuan (1,5), sehingga goal.x = 1, goal.y = 5

h_diagonal(n) = min (abs(n.x - goal.x) + abs(n.y – goal.y))

= min (abs(6 – 1) + abs(0 – 5))

= min (abs(5) + (-5))

= -5 + -5

= -10

h_orthogonal(n) = (abs(n.x - goal.x) + abs(n.y – goal.y))

= (abs(6 – 1) + abs(0 – 5))

= ((5) + abs(-5))

= 5 + 5

= 10

h(n) = h_diagonal(n) + (h_orthogonal (n) – (2 * h_diagonal(n)))

= -10 + (10 – (2 * (-10)))

= -10 + (10 – (-20))

= -10 + 30

= 20

F(x,y') = h(x,y') – 1

= 20 – 1

= 19

g. Koordinat X' = (5,0)

Nilai n.x = 5

Nilai n.y = 0

node tujuan (1,5), sehingga goal.x = 1, goal.y = 5

h_diagonal(n) = min (abs(n.x - goal.x) + abs(n.y – goal.y))

= min (abs(5 – 1) + abs(0 – 5))

= min (abs(4) + (-5))

= -4 + -5

= -9

h_orthogonal(n) = (abs(n.x - goal.x) + abs(n.y – goal.y))

= (abs(5 – 1) + abs(0 – 5))

= ((4) + abs(-5))

= 4 + 5

= 9

h(n) = h_diagonal(n) + (h_orthogonal (n) – (2 * h_diagonal(n)))

= -9 + (9 – (2 * (-9)))

= -9 + (9 – (-18))

= -9 + 27

= 18

F(x,y') = h(x,y') – 1

= 18 – 1

= 17

h. Koordinat X' = (4,0)

Nilai n.x = 4

Nilai n.y = 0

node tujuan (1,5), sehingga goal.x = 1, goal.y = 5

h_diagonal(n) = min (abs(n.x - goal.x) + abs(n.y – goal.y))

= min (abs(4 – 1) + abs(0 – 5))

= min (abs(3) + (-5))

= -3 + -5

= -8

h_orthogonal(n) = (abs(n.x - goal.x) + abs(n.y – goal.y))

= (abs(4 – 1) + abs(0 – 5))

= ((3) + abs(-5))

= 3 + 5

= 8

h(n) = h_diagonal(n) + (h_orthogonal (n) – (2 * h_diagonal(n)))

= -8 + (8 – (2 * (-8)))

= -8 + (8 – (-16))

= -8 + 24

= 16

F(x,y') = h(x,y') – 1

= 16 – 1

= 15



Gambar 3.2. Hasil Perhitungan Langkah Pertama

Sebagaimana terlihat pada gambar diatas, *node* yang akan dijadikan sebagai *node* tujuan selanjutnya adalah *node* yang memiliki nilai f paling kecil, dan terpilihlah *node* dengan nilai f = 11 dari koordinat (4,2). Selanjutnya *node* tetangga dari koordinat (4,2) akan dievaluasi dan dihitung seperti perhitungan yang sebelumnya. Langkah pertama menghasilkan *node-node* di dalam tujuan selanjutnya dijelaskan pada tabel 3.1.

Tabel 3.1. Perhitungan Langkah Pertama

Node-node Tetangga	(4,1) (4,2) (5,2) (6,2) (6,1) (6,0) (5,0) (4,0)
Node Awal	(5,1)
Node Selanjutnya	(4,2)

Pada perhitungan selanjutnya, korban bergerak dan *node* tujuan akhir pun berubah menjadi *node* tujuan korban yang akan dituju. Berikut adalah langkah kedua yang berada pada *node* dengan koordinat (4,2) dapat dilihat pada gambar 3.3.



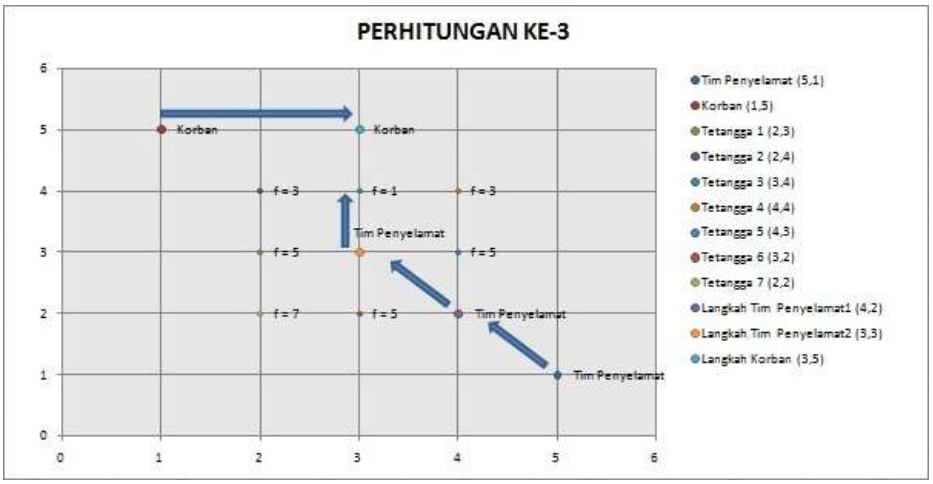
Gambar 3.3. Hasil Perhitungan Kedua

Tabel 3.2 menunjukan penjelasan tentang pencarian jalur terpendek untuk langkah kedua.

Tabel 3.2. Perhitungan Langkah Kedua

Node-node Tetangga	(3,2) (3,3) (4,3) (5,3) (5,2) (4,1) (3,1)
Node Awal	(4,2)
Node Selanjutnya	(3,3)

Perhitungan pada langkah ketiga yang berada pada *node* dengan koordinat (3,3) dapat dilihat pada gambar 3.4.



Gambar 3.4. Hasil Perhitungan Ketiga

Tabel 3.3 menunjukan penjelasan tentang pencarian jalur terpendek untuk langkah ketiga.

Tabel 3.3. Perhitungan Langkah Ketiga

Node-node Tetangga	(2,3) (2,4) (3,4) (4,4) (4,3) (3,2) (2,2)
Node Awal	(3,3)
Node Selanjutnya	(3,4)

Perhitungan pada langkah keempat yang berada pada *node* dengan koordinat (3,4) dapat dilihat pada gambar 3.5.



Gambar 3.5. Hasil Akhir Perhitungan

Tabel 3.4 menunjukan penjelasan tentang pencarian jalur terpendek untuk langkah keempat.

Tabel 3.4. Perhitungan Langkah Keempat

Node-node Tetangga	(2,4) (2,5) (3,5) (4,5) (4,4) (4,3) (2,3)
Node Awal	(3,4)
Node Selanjutnya	(4,5)

6. Kesimpulan

   Berdasarkan hasil akhir dari perhitungan di atas, korban dapat ditemukan dan diselamatkan. Ini membuktikan bahwa algoritma *Moving Target Search* dapat diterapkan pada masalah pencarian target dan pengambilan jalur terdekat atau tercepat agar dapat sampai ke tempat target.

III.5. Analisis Kebutuhan Non-Fungsional

Analisis non-fungsional merupakan analisis yang dibutuhkan untuk menentukan spesifikasi kebutuhan sistem. Spesifikasi ini juga meliputi elemen atau komponen-komponen apa saja yang dibutuhkan untuk sistem yang akan dibangun sampai dengan sistem tersebut diimplementasikan. Pada analisis kebutuhan sistem non-fungsioal ini dijelaskan analisis kebutuhan perangkat lunak, analisis kebutuhan perangkat keras, dan analisis pengguna.

III.5.1. Analisis Kebutuhan Perangkat Lunak

Kebutuhan perangkat lunak yang dibutuhkan untuk mengembangkan *game* Penyelamatan Korban Banjir yaitu:

a. Sistem Operasi Windows Xp dan Windows 7.
b. Adobe Flash CS 5.

III.5.2. Analisis Kebutuhan Perangkat Keras

Agar *game* penyelamatan korban banjir berjalan dengan baik, maka diperlukan spesfikasi kebutuhan perangkat keras yang sesuai dengan *game* yang akan dibangun. Spesifikasi minimum perangkat keras yang dibutuhkan yaitu:

1. Memori minimal 1 Gb.
2. *VGA* minimal 128 Mb.
3. *Processor* minimal 1 Ghz.
4. *Hardisk* minimal 40 Gb.
5. Keyboard : standar.
6. Mouse : standar.

III.5.3. Analisis Penggguna

Analisis pengguna diperlukan untuk mengetahui spesifikasi pengguna untuk dapat memainkan *game* penyelamatan korban banjir. Spesifikasinya bisa dilihat pada tabel 3.5.

Tabel 3.5. Spesifikasi Pengguna

Karakter Fisik Pengguna	
Umur	18 tahun keatas
Gender	Laki-laki dan Perempuan
Disabilities	Tidak buta dan tidak mempunyai cacat fisik seperti cacat tangan
Pengetahuan dan Pengalaman Pengguna	
Computer literacy	Dapat menggunakan dan mengoperasikan Komputer
Education	18 tahun keatas

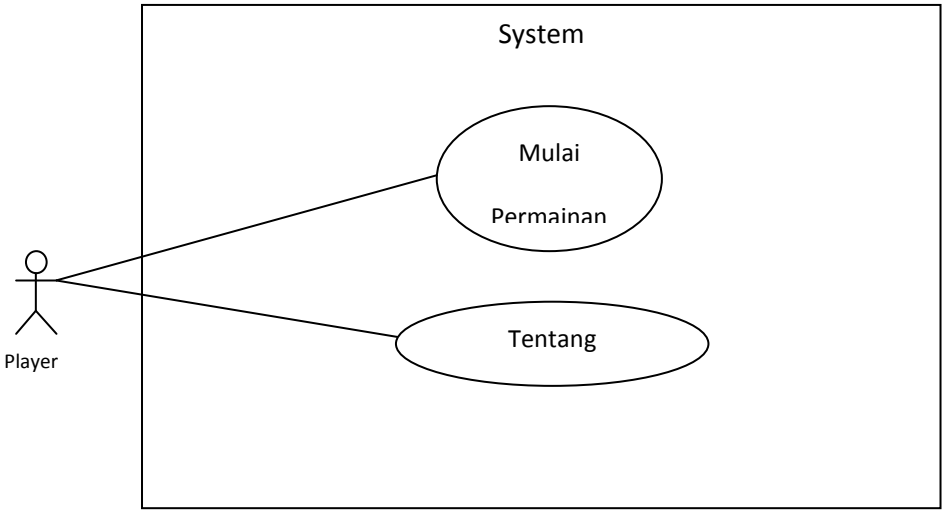
III.6. Analisis Kebutuhan Fungsional

Analisis kebutuhan fungsional menggambarkan proses kegiatan yang akan diterapkan dalam sebuah sistem dan menjelaskan kebutuhan yang diperlukan sistem agar sistem dapat berjalan dengan baik sesuai kebutuhan.

III.6.1. *Use Case Diagram*

Use Case Diagram merupakan pemodelan yang menunjukkan interaksi antara sistem dan lingkungannya. *Use Case Diagram* memberikan gambaran yang cukup sederhana dari interaksi sehingga harus memberikan lebih rinci untuk memahami apa yang terlibat [8].

Berikut adalah perancangan proses-proses yang terdapat pada *game* penyelamatan korban banjir, yang digambarkan dengan *use case diagram* yang dapat dilihat pada gambar 3.6.



Gambar 3.6. *Use Case Diagram* Game Penyelamatan Korban Banjir

Use case terdiri dari tiga bagian yaitu definisi *Actor,* definisi *use case,* dan skenario *use case*.

1. Definisi *Actor*

   Actor yaitu pihak yang mengakses *use case.* Hanya ada satu *actor* yang berperan yaitu *player* yang akan memainkan *game.*

2. Definisi *Use Case*

   Use case berfungsi untuk mewakili apa yang sistem bisa lakukan. Definisi *use case* bisa dilihat pada tabel 3.6.

Tabel 3.6. Definisi *Use Case*

No	*Use Case*	Deskripsi
1	Mulai Permainan	Untuk memulai simulasi penyelamatan korban banjir: 1. *Player* menekan tombol Mulai Permainan. 2. Simulasi permainan dimulai dengan menempatkan posisi awal dari tim penyelamat dan korban. 3. Tim penyelamat kemudian menghitung seluruh nilai tetangganya yang berjumlah 8 tetangga dari posisi awal yang telah

		ditentukan. 4. Perhitungan tetangga menggunakan fungsi Heuristik. Dalam kasus *game* simulasi ini fungsi Heuristik yamg digunakan yaitu *Non-Manhattan Distance* (8 arah). 5. Setelah fungsi Heuristik dihitung, selanjutnya menghitung nilai F dengan memasukan hasil perhitungan Heuristik. Nilai F digunakan untuk menentukan arah selanjutnya dari tim penyelamat, nilai F yang dipilih oleh tim penyelamat adalah nilai F terkecil dari tetangga-tetangga tim pemyelamat. 6. Tim Penyelamat mulai bergerak untuk mencari korban dengan perhitungan yang sudah dihitung. 7. Selama posisi tim penyelamat tidak sama dengan posisi korban, maka perhitungan Heuristik akan terus berulang sampai posisi tim penyelamat sama dengan posisi korban. 8. Jika korban sudah ditemukan, maka tim penyelamat memasukan korban ke dalam perahu karet. 9. Kemudian korban dibawa ke tempat yang aman.
2	Tentang Pembuat	Menampilkan profil atau data diri.

3. Skenario *Use Case*

Skenario *use case* merupakan hasil instansiasi dari setiap *use case*. Berikut adalah skenario jalannya masing-masing *use case game* penyelamatan korban banjir yang telah didefinisikan sebelumnya:

Nama *use case*: Mulai Permainan

Skenario:

Tabel 3.7. Skenario *Use Case* Mulai Permainan

Identifikasi	
No	1
Nama	Mulai Permainan
Tujuan	Memulai permainan
Deskripsi	Proses untuk memulai simulasi penyelamatan korban
Aktor	*Player*
Skenario Utama	
Kondisi Awal	*Player* berada di menu permainan

Aksi Aktor	Reaksi Sistem
1. *Player* memilih menu memulai permainan.	
	2. Menampilkan *map* simulasi penyelamatan korban.
	3. Tim penyelamat dan korban sudah dalam posisi awal yang telah ditentukan.
	4. Tim penyelamat kemudian mengecek dan menghitung seluruh nilai tetangganya yang berjumlah 8 tetangga dari posisi awal yang telah ditentukan. Perhitungannya menggunakan fungsi Heuristik *Non-Manhattan Distance.*
	5. Menghitung nilai F dengan memasukan hasil perhitungan Heuristik dari masing-masing tetangga. Nilai F terkecil akan menjadi tujuan selanjutnya dari tim penyelamat.
	6. Tim penyelamat mulai bergerak

	mencari korban dengan memperhatikan perhitungan yang sudah dihitung.
	7. Selama posisi tim penyelamat tidak sama dengan posisi korban, maka perhitungan Heuristik akan terus berulang sampai posisi tim penyelamat sama dengan posisi korban.
	8. Jika korban sudah ditemukan, maka tim penyelamat memasukan korban ke dalam perahu karet.
	9. Kemudian korban dibawa ke tempat aman.
Kondisi akhir	Korban berhasil diselamatkan

Nama *use case*: Tentang Pembuat

Skenario:

Tabel 3.8. Skenario *Use Case* Tentang Pembuat

Identifikasi	
No	2
Nama	Tentang Pembuat
Tujuan	Menampilkan data diri
Deskripsi	Proses untuk menampilkan data diri
Aktor	*Player*
Skenario Utama	
Kondisi Awal	*Player* berada di menu permainan
Aksi Aktor	Reaksi Sistem
1. *Player* memilih menu Tentang Pembuat	

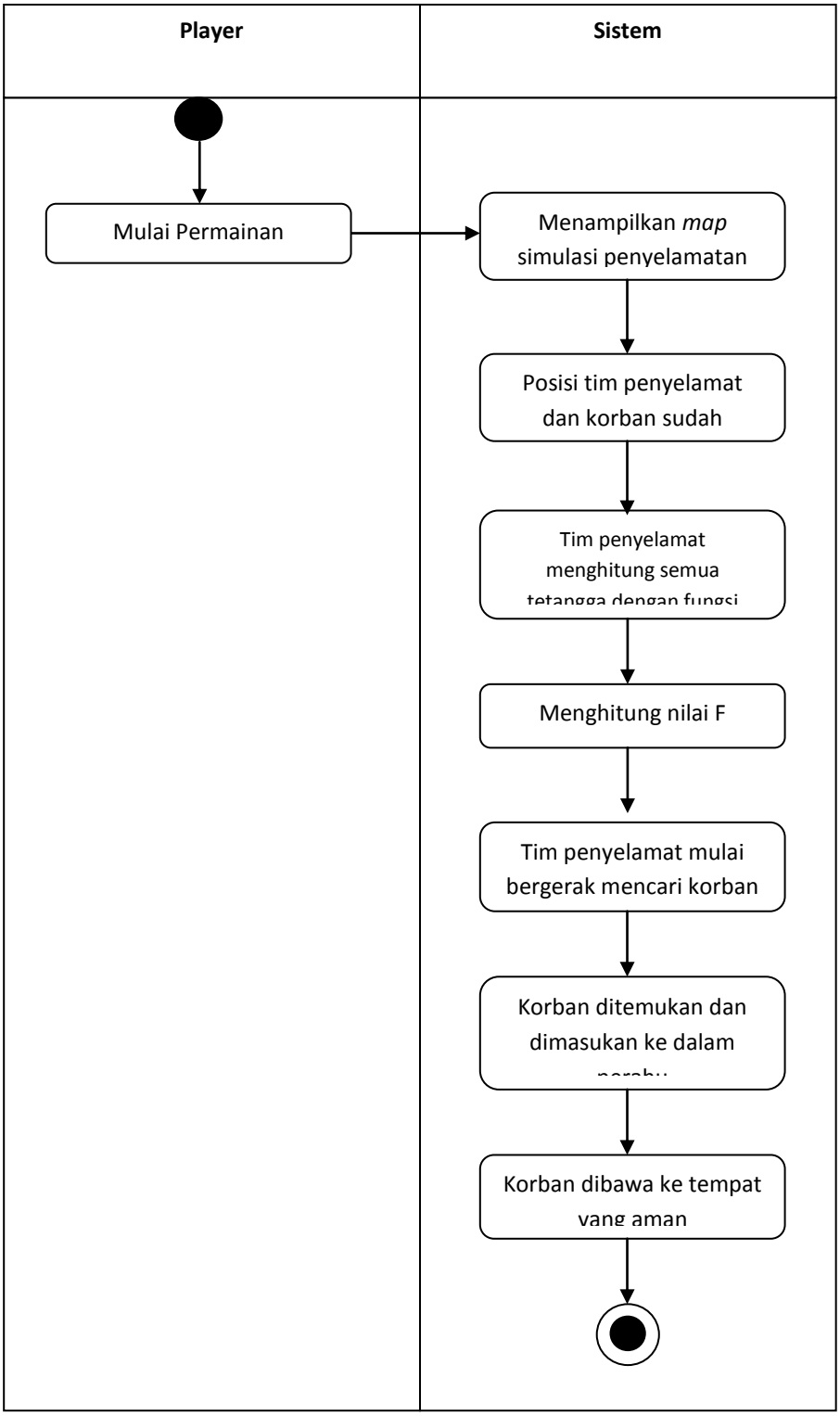
	2. Menampilkan menu Tentang Pembuat
3. *Player* menekan tombol kembali	
	4. Kembali ke Menu awal permainan
Kondisi akhir	Menampilkan data diri pembuat

III.6.2. *Activity Diagram*

Activity diagram menggambarkan berbagai alur aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alur berawal, decision yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

1. *Activity Diagram* Mulai Permainan

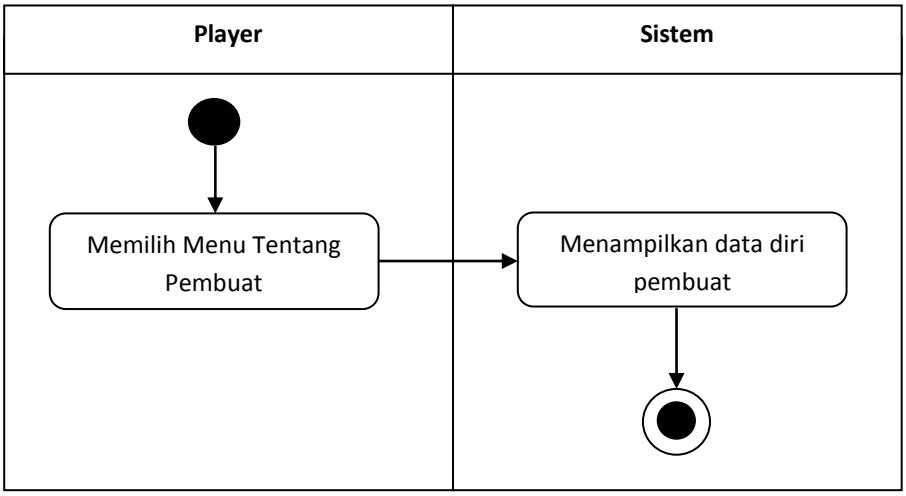
Berikut ini adalah diagram yang menunjukan alur aksi pada aktivitas Mulai Permaian yang dapat dilihat pada gambar 3.7.



Gambar 3.7. *Activity Diagram* Mulai Permainan

2. *Activity Diagram* Tentang Pembuat

Berikut ini adalah diagram yang menunjukan alur aksi pada aktivitas Tentang Pembuat yang dapat dilihat pada gambar 3.8.

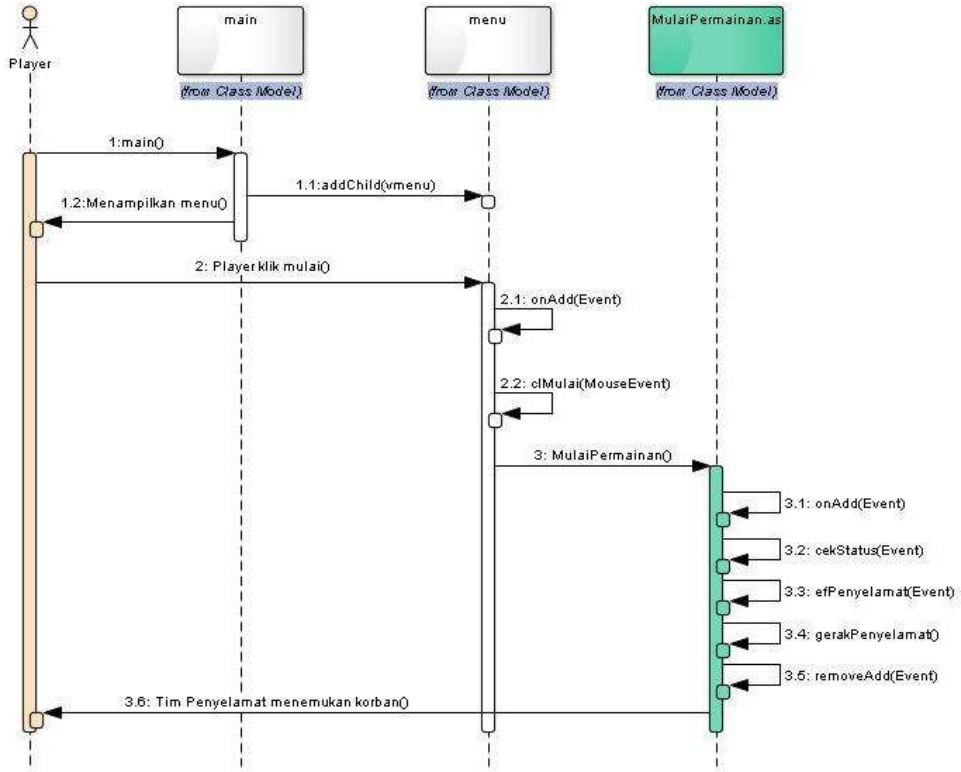


Gambar 3.8. *Activity Diagram* Tentang Pembuat

III.6.3. *Sequence Diagram*

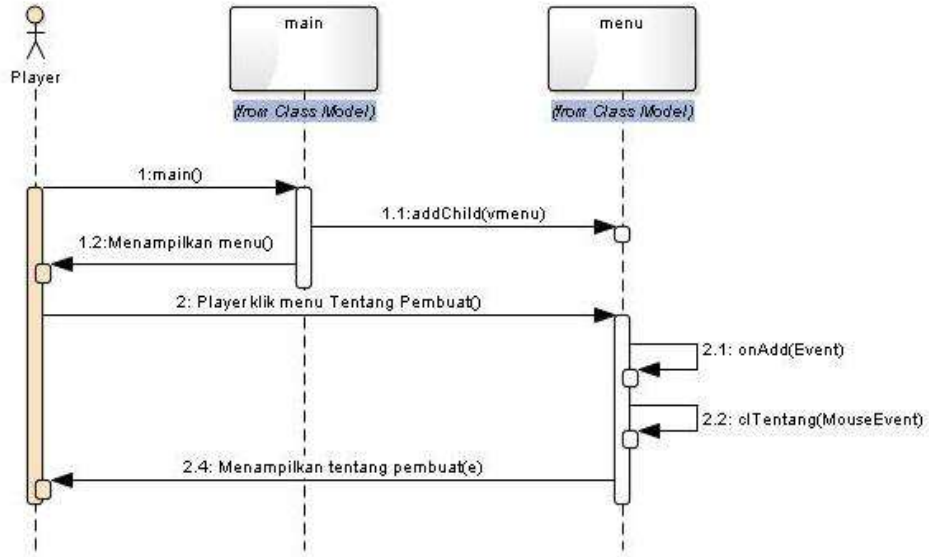
Sequence diagram menggambarkan interaksi antar objek di dalam dan disekitar sistem (termasuk pengguna, display, dan sebagainya) berupa message yang digambarkan terhadap waktu. *Sequence diagram* terdiri atas dimensi vertical (waktu) dan dimensi horizontal (objek-objek yang terkait). Berikut *sequence diagram* pada *game* Penyelamatan Korban Banjir.

1. *Sequence Diagram* Mulai Permainan



Gambar 3.9. *Sequence Diagram* Mulai Permainan

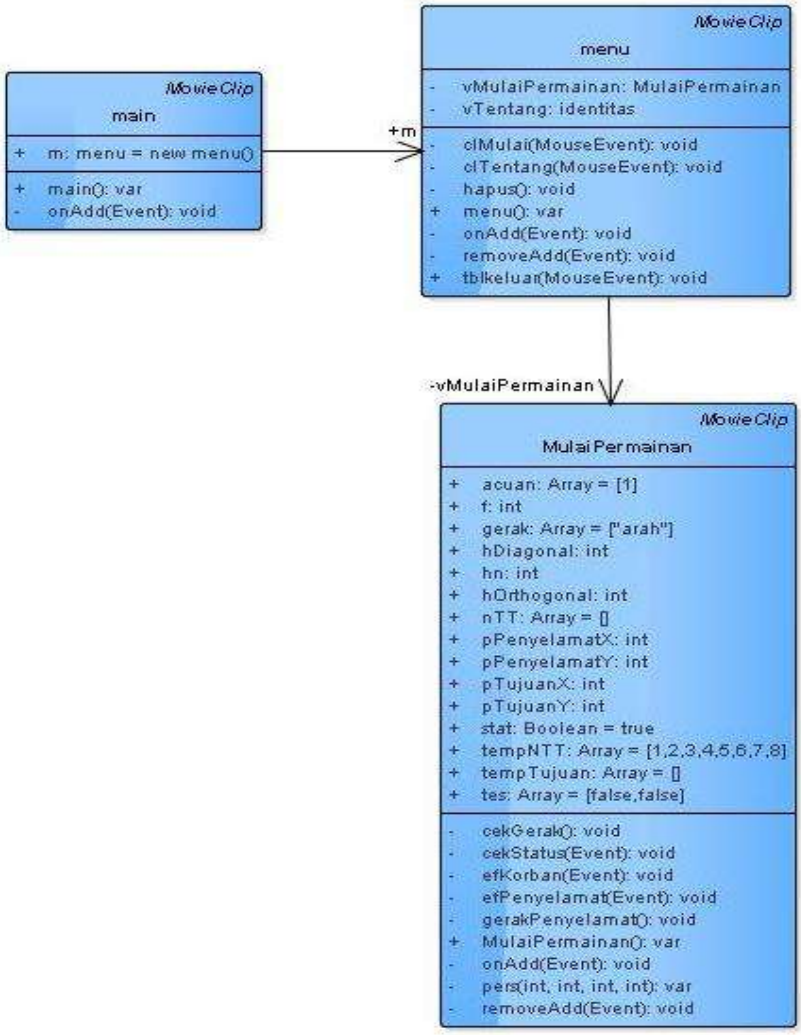
2. *Sequence Diagram* Tentang Pembuat



Gambar 3.10. *Sequence Diagram* Tentang Pembuat

III.6.4. *Class Diagram*

Class diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi [3].



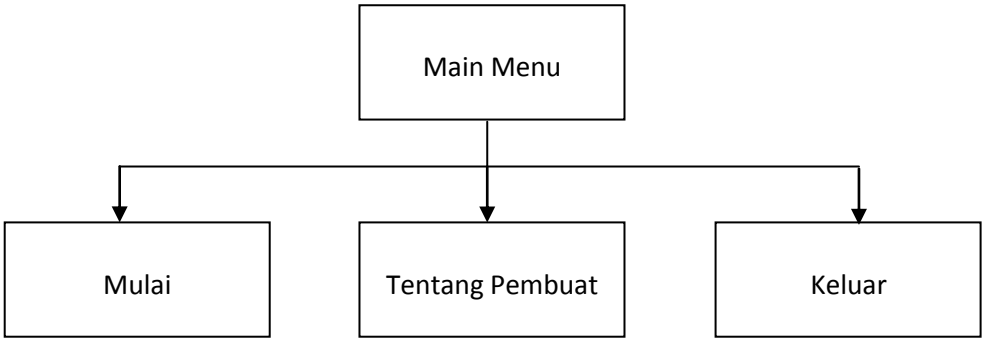
Gambar 3.11. *Class Diagram*

III.7. Perancangan Sistem

Perancangan sistem merupakan kegiatan untuk merancang aplikasi yang akan di bangun, tahapannya dimulai dari perancangan struktur menu, perancangan antarmuka, perancangan pesan dan jaringan semantik dari *game* penyelamatan korban banjir.

III.7.1 Perancangan Struktur Menu

Perancangan struktur menu pada *game* penyelamatan korban banjir adalah sebagai berikut :

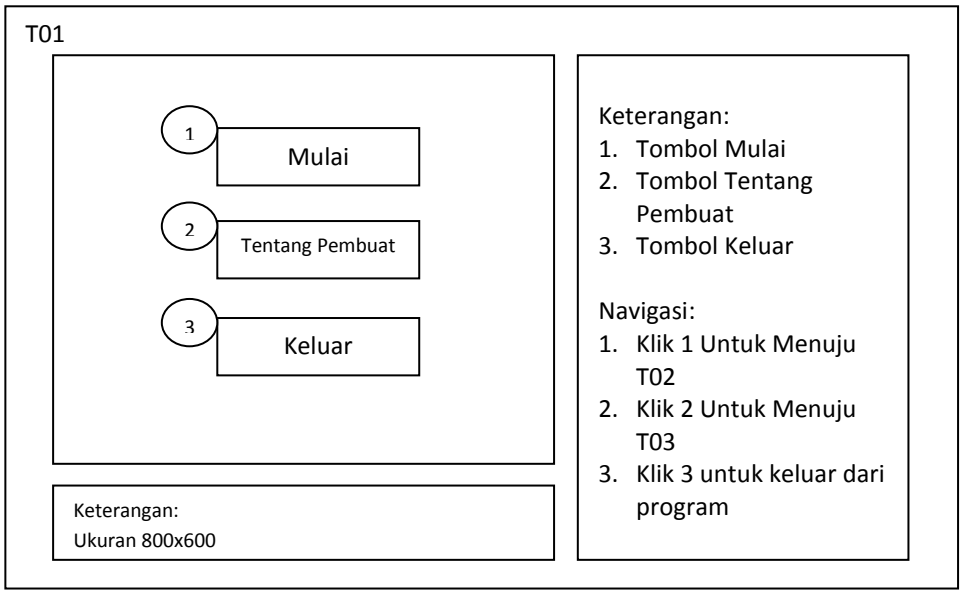


Gambar 3.12. Perancangan Struktur Menu

III.7.2 Perancangan Antarmuka

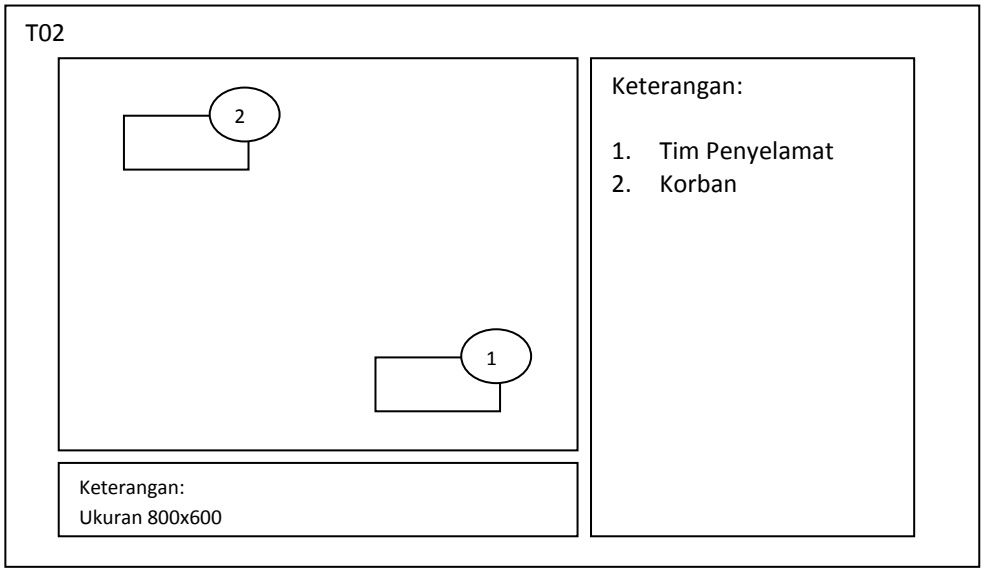
Perancangan antarmuka bertujuan untuk memberikan gambaran tentang aplikasi yang akan dibangun, sehingga akan mempermudah dalam mengimplementasikan aplikasi serta akan memudahkan pembuatan aplikasi.

1. Perancangan Antarmuka Menu



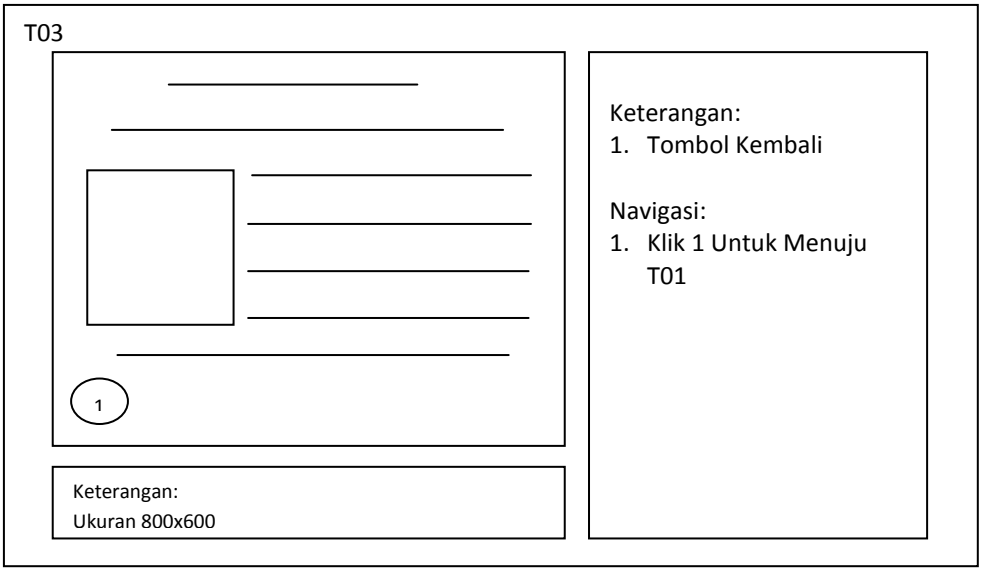
Gambar 3.13. Antarmuka Main Menu

2. Perancangan Antarmuka Mulai Permainan



Gambar 3.14. Antarmuka Mulai Permainan

3. Perancangan Antarmuka Tentang

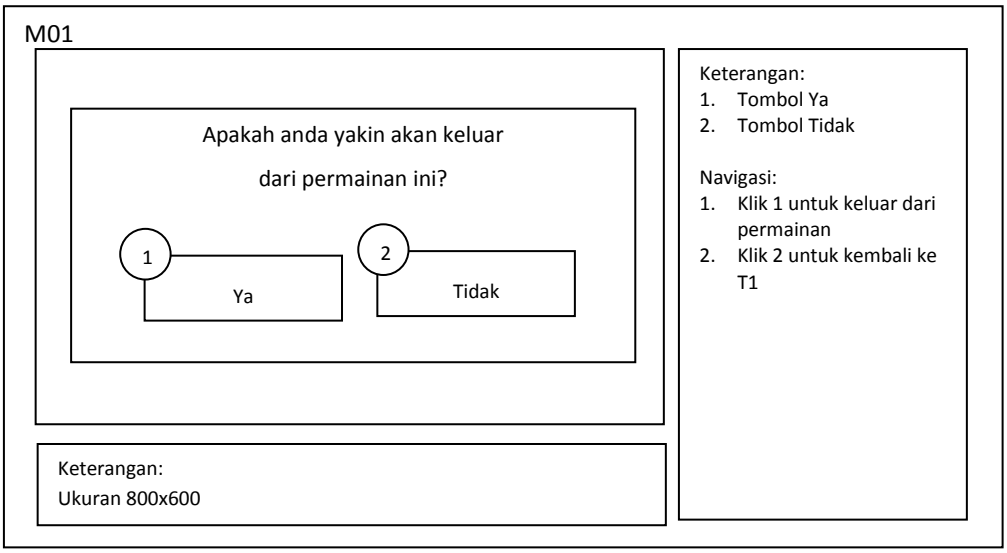


Gambar 3.15. Antarmuka Tentang

III.7.3 Perancangan Pesan

Pada perancangan pesan ini menjelaskan mengenai pilihan yang harus di ambil oleh *player* dalam sebuah intruksi. Perancangan pesan yang terdapat pada *game* penyelamatan korban banjir yaitu perancangan pesan keluar.

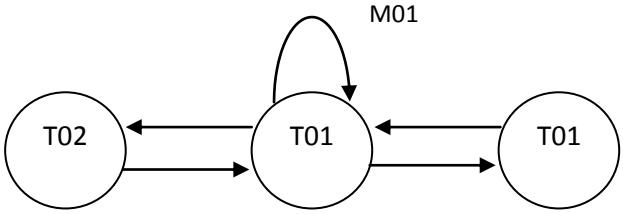
1. Perancangan Pesan Menu Keluar



Gambar 3.16. Antarmuka Keluar

III.7.4 Jaringan Semantik

Untuk mengetahui keterhubungan dari setiap tampilan yang dibuat dapat menggunakan jaringan semantik. Gambar di bawah ini adalah jaringan semantik pada *game* penyelamatan korban banjir.



Gambar 3.17. Jaringan Semantik

III.7.5 Perancangan Method

Perancangan method merupakan perancangan yang berfungsi untuk mendeskripsikan *method-method* yang berada di dalam aplikasi. Method

dapat dipanggil dengan menyertakan variabel, baik hanya satu variabel, banyak variabel atau bahkan tidak ada sama sekali.

Adapun *method-method* yang terdapat dalam *game* penyelamatan korban banjir yang akan dibangun adalah sebagai berikut :

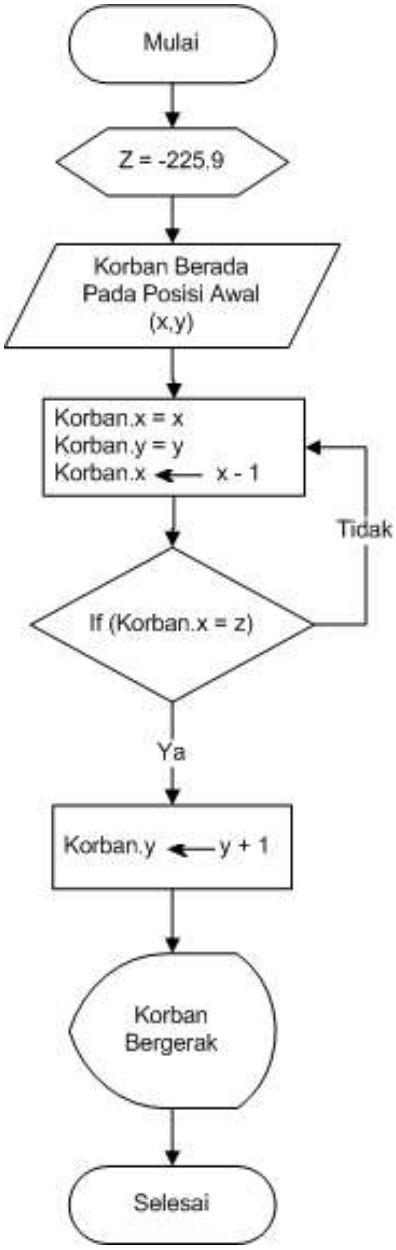
a. Perancangan *Method* yang Tersedia di Dalam *Tools* Pembangunan

Tabel 3.9. *Method* yang tersedia di dalam tools pembangunan

No	Nama Method	Keterangan
1	onAdd(Event):void	Untuk menampilkan fungsi atau procedure
2	removeAdd(Event):void	Untuk menghapus fungsi atau procedure
3	menu()	Untuk menampilkan menu
4	hapus()	Untuk menghapus *MovieClip*
5	tblKeluar(MouseEvent):void	Untuk keluar dari program
6	main()	Untuk menampilkan semua *Stage*
7	MulaiPermainan()	Untuk menampilkan *Stage* permainan
8	efPenyelamat(Event):void	Untuk memperbaharui nilai tim penyelamat
9	cekGerak()	Untuk mengecek gerak korban
10	pers()	Untuk perhitungan algoritma

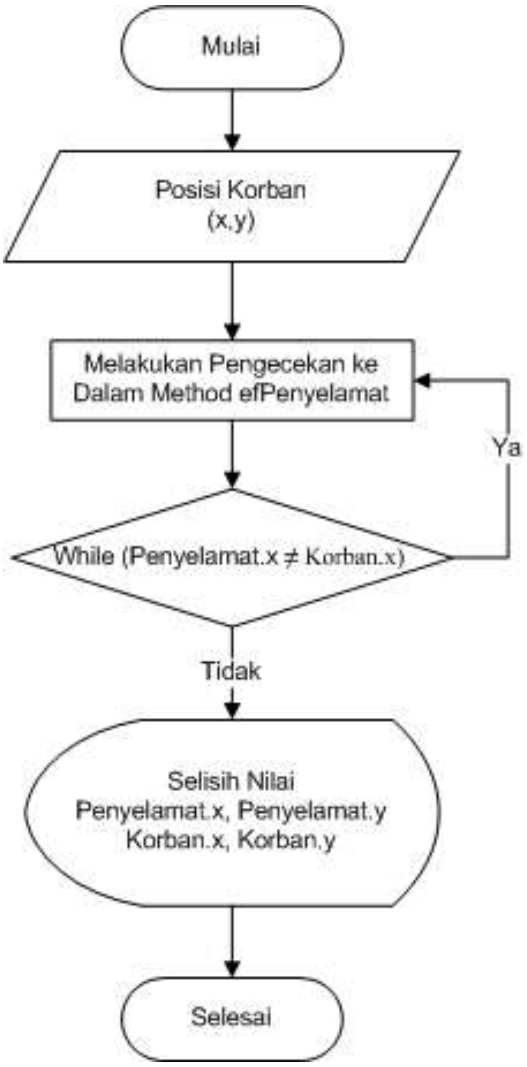
b. Perancangan *Method* yang Akan Digunakan di Dalam *Game*

1. Perancangan *Method* efKorban



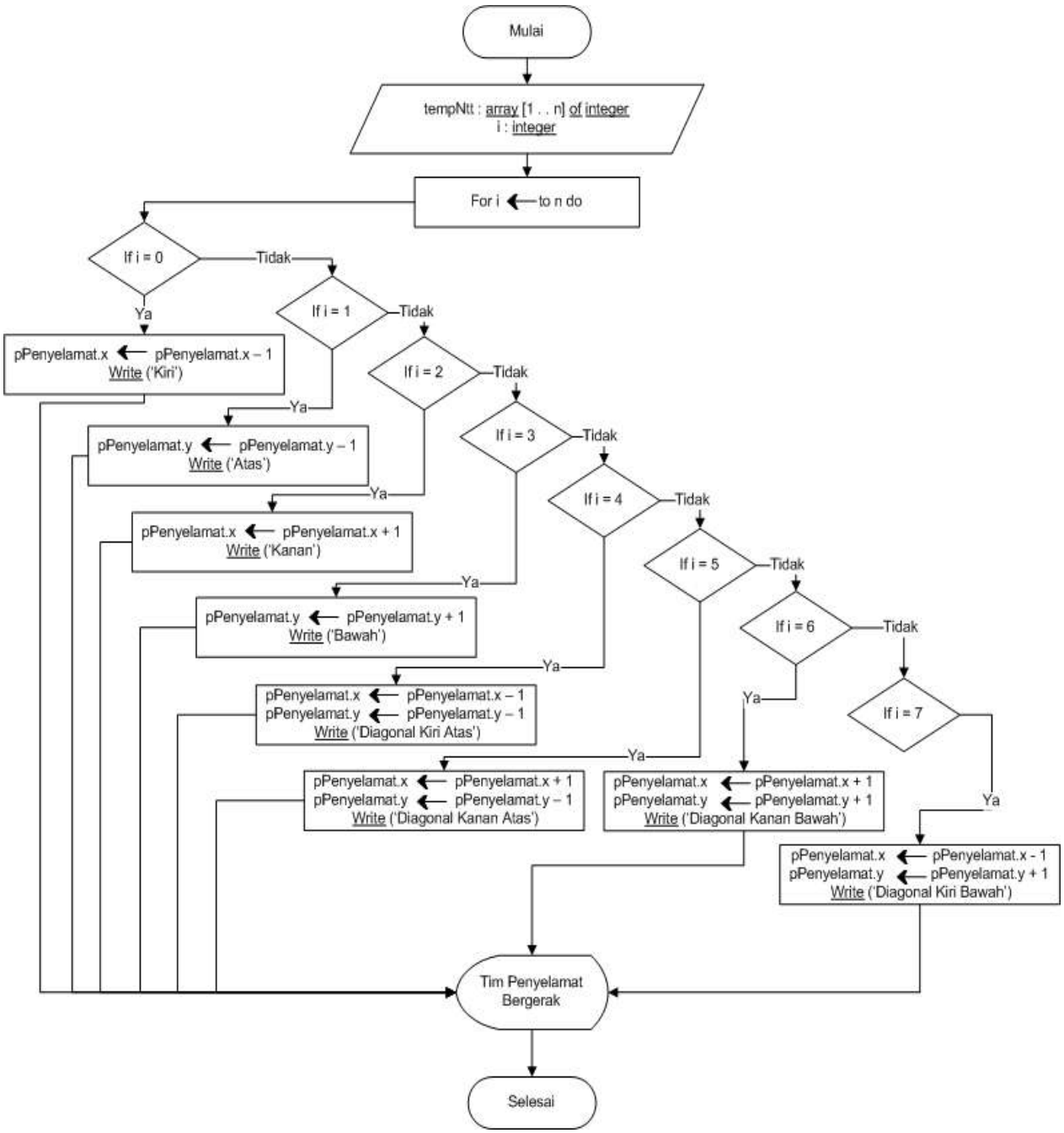
Gambar 3.18. Perancangan *Method* efKorban

2. Perancangan *Method* cekStatus



Gambar 3.19. Perancangan *Method* cekStatus

3. Perancangan *Method* gerakPenyelamat



Gambar 3.20. Perancangan *Method* gerakPenyelamat