BAB 3
ANALISIS DAN KEBUTUHAN ALGORITMA

3.1 Analisis Sistem

Pada bagian analisis sistem ini, akan dilakukan berbagai macam analisis yang berhubungan dengan *game* yang akan dikembangkan, mulai dari analisis masalah, analisis *game*, analisis data masukan, analisis algoritma, analisis kebutuhan perangkat lunak, analisis kebutuhan non-fungsional dan analisis kebutuhan fungsional.

3.1.1 Analisis Masalah

Game S.O.S merupakan *game simulation* bergrafik 2D (dua dimensi), yang mempunyai misi utama untuk menyelamatkan korban dari tengah laut. Di dalam *game* tersebut terdapat kekurangan, diantaranya yaitu dari segi karakter NPC (*Non-Playable Character*) korban yang kurang bervariasi sehingga membuat *game* menjadi kurang menantang untuk dimainkan.

Game penyelamatan korban banjir merupakan *game* yang berjenis simulasi yang bergrafik 2D (dua dimensi). *Game* ini mempunyai solusi dari kekurangan pada *game* S.O.S, yaitu dengan menambahkan tingkah laku pada karakter NPC (*Non-Playable Character)* pada korban banjir. Misi utama dari *game* ini adalah untuk menyelamatkan korban banjir dari pemukiman masyarakat.

3.2 Analisis Game

Game Rescue Flood Victim adalah *game simulation* yang bergrafik 2 dimensi. Misi utama dan *gameplay* dari *game* ini adalah menyelamatkan korban banjir dari banjir yang berada pemukiman masyarakat dengan menggunakan perahu sebagai alat penolong. Pemain harus menyelamatkan korban banjir dari tingginya air banjir.

Game Rescue Flood Victim mengimplementasi algoritma logika *fuzzy* untuk mengatur tingkah laku serta membuat kecerdasan buatan pada karakter NPC (*Non-*

Playable Character) korban banjir. Tingkah laku pada korban banjir yaitu sebagai berikut :

1. Korban banjir bertahan untuk cepat diselamatkan.
2. Korban banjir menyelamatkan diri ketika korban lama menunggu untuk diselamatkan.

3.2.1 Gameplay Game Rescue Flood Victim

Gameplay dari *game Rescue Flood Victim* yaitu relawan menyelamatkan korban banjir yang masih selamat di atap rumah menggunakan perahu. Adapun rintangan-rintangan yang terdapat dalam *gameplay* pada *game Rescue Flood Victim* ini adalah :

1. Tinggi air

   Air dapat bertambah besar ketika hujan. Jika tidak hujan, maka air berkurang.
2. Stamina

   Stamina dapat berkurang ketika memulai melakukan penyelamatan.
3. Waktu

   Waktu dihitung ketika telah memulai permainan.
4. Jumlah korban yang harus diselamatkan.

   Misi mempunyai batas jumlah korban yang harus diselamatkan.
5. Jumlah korban yang dibawa oleh perahu

   Perahu mempunyai batas jumlah korban yang dibawa.



Gambar 3.1 ***Gameplay Game Rescue Flood Victim***

3.3 Analisis Masukan

Analisis masukan yang dilakukan dalam penelitian ini adalah menganalisis parameter dalam algoritma logika *fuzzy* pada *game Rescue Flood Victim*, parameter tersebut antara lain air, stamina dan jarak.

1. Parameter Air

   Parameter air ini dapat diperoleh dari ketetapan aturan. Ketika dimulai, cuaca hujan air akan mengalami kenaikan atau bertambah. Jika cuaca tidak hujan, maka air akan mengalami penurunan atau berkurang.

   -Nilai Air pertama kali mulai = 50

   -Air naik = + 5 (dalam waktu 10-12 detik)

   -Air turun = – 5 (dalam waktu 9 detik)

2. Parameter Stamina

   Parameter stamina ini diperoleh ketika korban mulai bertahan.

   -Nilai Stamina pertama kali mulai = 100

   -Stamina turun = -1 (dalam waktu 1 detik)

3. Parameter Jarak

   Parameter jarak ini diperoleh dari tempat korban untuk melakukan pertolongan dan penyelamatan kepada korban.

   -Nilai Jarak pertama kali mulai = 100 (korban sedang jauh)

3.4 Analisis Algoritma

Analisis algoritma yang dilakukan dalam penelitian ini adalah meneliti bagaimana cara kerja algoritma logika *fuzzy* pada NPC korban banjir di dalam *game Rescue Flood Victim*.

3.4.1 Analisis Algoritma Logika Fuzzy

Logika *Fuzzy* di dalam membangun *game* ini digunakan untuk menentukan variasi perilaku yang akan dihasilkan oleh NPC (*non playable character*). Dengan adanya logika *fuzzy*, masing-masing NPC dapat merubah perilaku berbasis perubahan variabel masukan menjadi perilaku yang akan dirancang menggunakan logika *fuzzy*.

Metode logika *fuzzy* yang digunakan adalah metode Sugeno, karena metode ini menghasilkan keluaran yang berupa konstanta tegas, sehingga dapat mewakili nilai perilaku yang sudah dirancang sebelumnya.

1. **Nilai Linguistic Pada Parameter**

   Berikut ini merupakan nilai *linguistic* pada parameter yang digunakan :

   1) Parameter Jarak memiliki 3 nilai *linguistic* yaitu :
      a. Dekat : memiliki nilai 0 – 40.
      b. Sedang : memiliki nilai 30 - 70.
      c. Jauh : memiliki nilai 60 – 100.
   2) Parameter Air memiliki 3 nilai *linguistic* yaitu :
      a. Rendah : memiliki nilai 0 – 80.
      b. Sedang : memiliki nilai 70 - 140.
      c. Tinggi : memiliki nilai 130 – 200.
   3) Parameter Stamina memiliki 3 nilai *linguistic* yaitu :
      a. Lemah : memiliki nilai 0 – 40.
      b. Sedang : memiliki nilai 30 - 70.
      c. Kuat : memiliki nilai 60 – 100.

2. **Perilaku NPC Korban Banjir**

   Terdapat 2 perilaku NPC korban banjir beserta nilai *linguistic* pada tabel 3.1

Tabel 3.1 Perilaku NPC

No	Perilaku	Nilai
1	Bertahan	0 - 50
2	Menyelamatkan diri	51 - 100

3. *Fuzzy's Rule*

Fuzzy (rules) atau aturan-aturan yang diterapkan dalam penentuan perilaku NPC korban banjir terdapat pada tabel 3.2.

Tabel 3.2 *Fuzzy Rule*

NO		Jarak		Stamina		Air		Perilaku NPC
1	IF	Dekat	AND	Lemah	AND	Rendah	THEN	Bertahan
2	IF	Dekat	AND	Lemah	AND	Sedang	THEN	Bertahan
3	IF	Dekat	AND	Lemah	AND	Tinggi	THEN	Menyelamatkan diri
4	IF	Dekat	AND	Sedang	AND	Rendah	THEN	Bertahan
5	IF	Dekat	AND	Sedang	AND	Sedang	THEN	Bertahan
6	IF	Dekat	AND	Sedang	AND	Tinggi	THEN	Menyelamatkan diri
7	IF	Dekat	AND	Kuat	AND	Rendah	THEN	Bertahan
8	IF	Dekat	AND	Kuat	AND	Sedang	THEN	Bertahan
9	IF	Dekat	AND	Kuat	AND	Tinggi	THEN	Menyelamatkan diri
10	IF	Sedang	AND	Lemah	AND	Rendah	THEN	Menyelamatkan diri
11	IF	Sedang	AND	Lemah	AND	Sedang	THEN	Menyelamatkan diri
12	IF	Sedang	AND	Lemah	AND	Tinggi	THEN	Menyelamatkan diri
13	IF	Sedang	AND	Sedang	AND	Rendah	THEN	Bertahan
14	IF	Sedang	AND	Sedang	AND	Sedang	THEN	Bertahan
15	IF	Sedang	AND	Sedang	AND	Tinggi	THEN	Menyelamatkan diri
16	IF	Sedang	AND	Kuat	AND	Rendah	THEN	Bertahan
17	IF	Sedang	AND	Kuat	AND	Sedang	THEN	Bertahan
18	IF	Sedang	AND	Kuat	AND	Tinggi	THEN	Menyelamatkan diri
19	IF	Jauh	AND	Lemah	AND	Rendah	THEN	Menyelamatkan diri
20	IF	Jauh	AND	Lemah	AND	Sedang	THEN	Menyelamatkan diri
21	IF	Jauh	AND	Lemah	AND	Tinggi	THEN	Menyelamatkan diri
22	IF	Jauh	AND	Sedang	AND	Rendah	THEN	Bertahan
23	IF	Jauh	AND	Sedang	AND	Sedang	THEN	Bertahan
24	IF	Jauh	AND	Sedang	AND	Tinggi	THEN	Menyelamatkan diri
25	IF	Jauh	AND	Kuat	AND	Rendah	THEN	Bertahan
26	IF	Jauh	AND	Kuat	AND	Sedang	THEN	Bertahan
27	IF	Jauh	AND	Kuat	AND	Tinggi	THEN	Menyelamatkan diri

4. Contoh Kasus

Korban banjir ingin diselamatkan dengan jarak 35 pixel, stamina korban banjir 65, dan tinggi air 135 cm.

Diketahui :

Jarak = 35

Stamina = 65

Air = 135

Ada 3 tahapan yang harus dilewati untuk dapat menyelesaikan kasus diatas.

a. *Fuzzification*

b. *Inference*

c. *Defuzzification*

A. Tahapan *Fuzzification*

Fuzzification adalah proses perubahan suatu nilai *crisp* ke dalam variabel *fuzzy* yang berupa variabel linguistik yang nantinya akan dikelompokkan menjadi himpunan *fuzzy*.

Berikut ini variabel linguistik yang diperoleh dari parameter yang sudah diketahui yaitu jarak, stamina dan air.

1. Jarak terdiri dari 3 variabel linguistik yaitu Dekat, Sedang, Jauh.

Fungsi Keanggotaan variabel jarak dari kurva trapesium adalah sebagai berikut :

$$\mu Dekat[x] = \begin{cases} 0; x \leq 0, x \geq 40 \\ \frac{x-0}{10}; 0 \leq x \leq 10 \\ 1; 10 \leq x \leq 30 \\ \frac{40-x}{10}; x \geq 40 \end{cases} \quad (3.1)$$

$$\mu Sedang[x] = \begin{cases} 0; x \leq 30, x \geq 70 \\ \frac{x-30}{10}; 30 \leq x \leq 40 \\ 1; 40 \leq x \leq 60 \\ \frac{70-x}{10}; x \geq 70 \end{cases} \quad (3.2)$$

$$\mu Jauh[x] = \begin{cases} 0; x \leq 60, x \geq 100 \\ \frac{x-60}{10}; 60 \leq x \leq 70 \\ 1; 70 \leq x \leq 90 \\ \frac{100-x}{10}; x \geq 100 \end{cases} \quad (3.3)$$

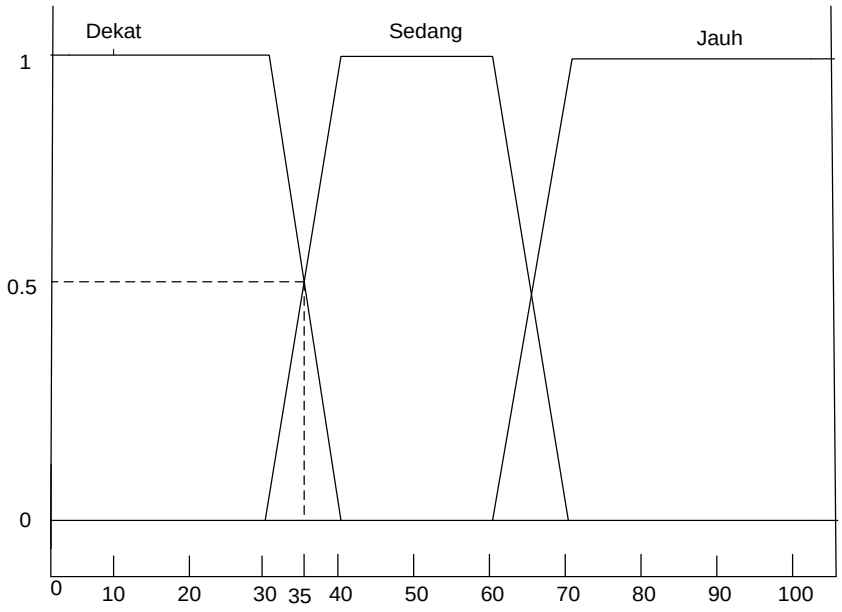
Derajat keanggotaan untuk Jarak 35 adalah :

$$\mu\text{Dekat}[35] = \frac{40-35}{10} = 0.5$$

$$\mu\text{Sedang}[35] = \frac{35-30}{10} = 0.5$$

$$\mu\text{Jauh}[35] = 0$$

Maka akan didapat fungsi keanggotaan trapesium dari jarak yang dapat dilihat pada gambar 3.2.



Gambar 3.2 Fungsi Keanggotaan Trapesium Variabel Jarak

2. Stamina terdiri dari 3 variabel linguistik yaitu Lemah, Sedang, Kuat.

Fungsi Keanggotaan variabel stamina dari kurva trapesium adalah sebagai berikut:

$$\mu Lemah[x] = \begin{cases} 0; x \leq 0, x \geq 40 \\ \frac{x-0}{10}; 0 \leq x \leq 10 \\ 1; 10 \leq x \leq 30 \\ \frac{40-x}{10}; x \geq 40 \end{cases} \quad (3.4)$$

$$\mu Sedang[x] = \begin{cases} 0; x \leq 30, x \geq 70 \\ \frac{x-30}{10}; 30 \leq x \leq 40 \\ 1; 40 \leq x \leq 60 \\ \frac{70-x}{10}; x \geq 70 \end{cases} \quad (3.5)$$

$$\mu Kuat[x] = \begin{cases} 0; x \leq 60, x \geq 100 \\ \frac{x-60}{10}; 60 \leq x \leq 70 \\ 1; 70 \leq x \leq 90 \\ \frac{100-x}{10}; x \geq 100 \end{cases} \quad (3.6)$$

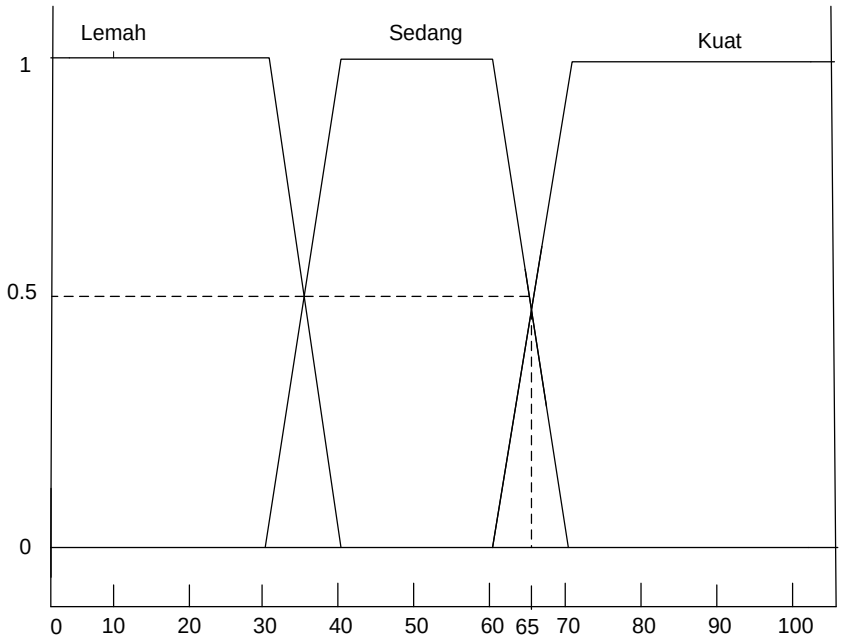
Derajat keanggotaan untuk Stamina 65 adalah :

μLemah$[65] = 0$

μSedang$[65] = 0.5$

μKuat$[65] = 0.5$

Maka akan didapat fungsi keanggotaan trapesium dari stamina yang dapat dilihat pada gambar 3.3.



Gambar 3.3 Fungsi Keanggotaan Trapesium Variabel Stamina

3. Air terdiri dari 3 variabel linguistik yaitu Rendah, Sedang, Tinggi.

Fungsi Keanggotaan variabel air dari kurva trapesium adalah sebagai berikut :

$$\mu Rendah[x] = \begin{cases} 0; x \leq 0, x \geq 80 \\ \frac{x-0}{10}; 0 \leq x \leq 10 \\ 1; 10 \leq x \leq 70 \\ \frac{80-x}{10}; x \geq 80 \end{cases} \quad (3.7)$$

$$\mu Sedang[x] = \begin{cases} 0; x \leq 70, x \geq 140 \\ \frac{x-70}{10}; 70 \leq x \leq 80 \\ 1; 80 \leq x \leq 130 \\ \frac{140-x}{10}; x \geq 140 \end{cases} \quad (3.8)$$

$$\mu Tinggi[x] = \begin{cases} 0; x \leq 130, x \geq 200 \\ \frac{x-130}{10}; 130 \leq x \leq 140 \\ 1; 140 \leq x \leq 190 \\ \frac{200-x}{10}; x \geq 200 \end{cases} \quad (3.9)$$

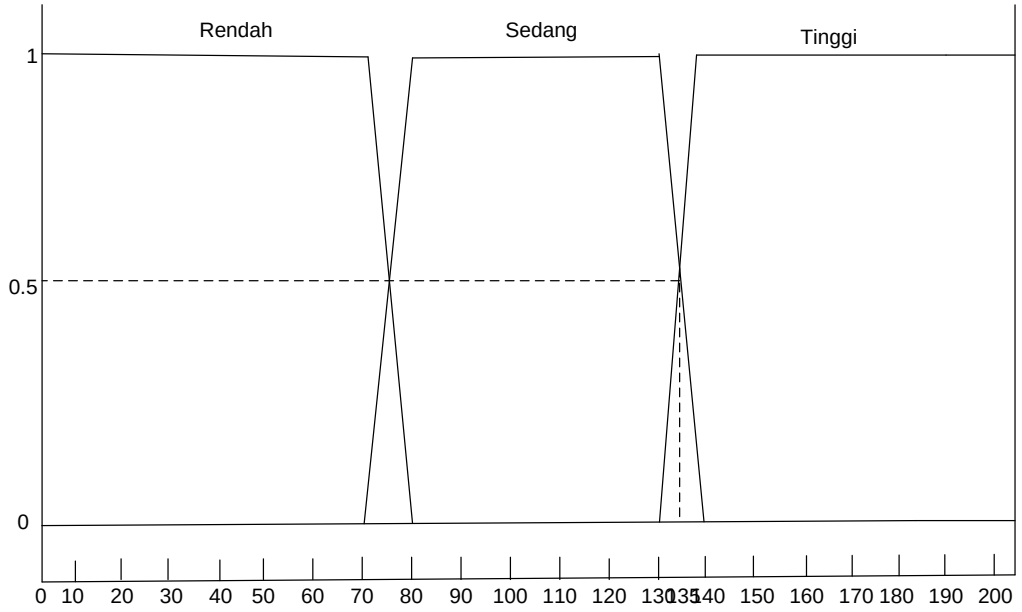
Derajat keanggotaan untuk Air 135 adalah :

μRendah$[135] = 0$

μSedang$[135] = 0.5$

μTinggi$[135] = 0.5$

Maka akan didapat fungsi keanggotaan trapesium dari air yang dapat dilihat pada gambar 3.4.



Gambar 3.4 Fungsi Keanggotaan Trapesium Variabel Air

Sehingga pada *fuzzyfication* menghasilkan *fuzzy* input dari fungsi keanggotaan :

a. Jarak = dekat (0.5), sedang (0.5) dan jauh (0)
b. Stamina = lemah (0), sedang (0.5) dan kuat (0.5)
c. Air = rendah (0), sedang (0.5) dan tinggi (0.5)

Dari *fuzzy* input tersebut maka akan dimasukan kedalam 27 *rule* yang dapat di aplikasikan.

B. Proses *Inference*

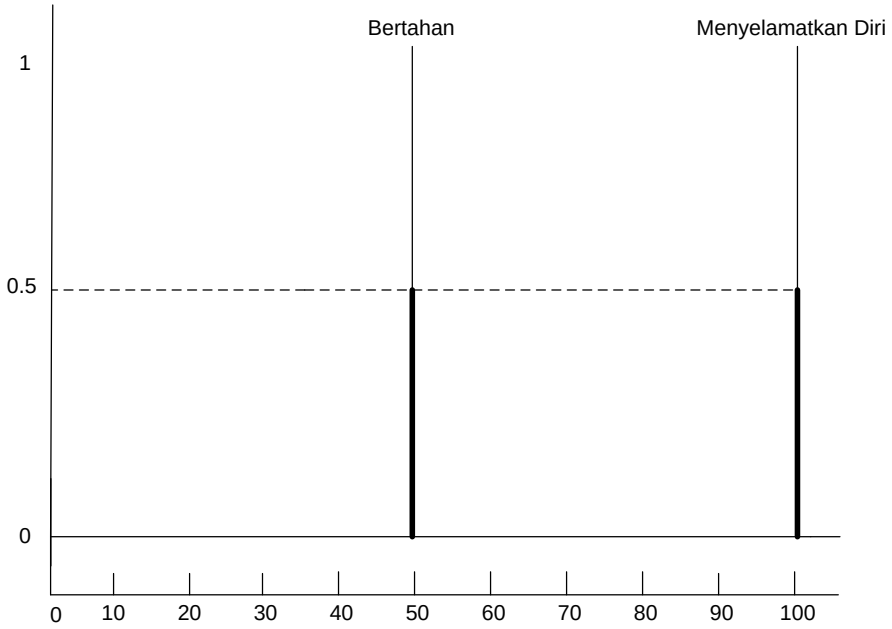
Dari 27 *rule* dan *fuzzy* input yang sudah diketahui tersebut maka proses *inference* yang terjadi adalah menggunakan aturan *Conjunction* (∧) dengan memilih derajat keanggotaan minimum sehingga di peroleh:

1. *If* Jarak *is* Dekat (0.5) *and* Stamina *is* Lemah (0) *and* Air *is* Rendah (0) *then* Perilaku NPC *is* Bertahan (0).
2. *If* Jarak *is* Dekat (0.5) *and* Stamina *is* Lemah (0) *and* Air *is* Sedang (0.5) *then* Perilaku NPC *is* Bertahan (0).
3. *If* Jarak *is* Dekat (0.5) *and* Stamina *is* Lemah (0) *and* Air *is* Tinggi (0.5) *then* Perilaku NPC *is* Menyelamatkan diri (0).
4. *If* Jarak *is* Dekat (0.5) *and* Stamina *is* Sedang (0.5) *and* Air *is* Rendah (0) *then* Perilaku NPC *is* Bertahan (0).
5. *If* Jarak *is* Dekat (0.5) *and* Stamina *is* Sedang (0.5) *and* Air *is* Sedang (0.5) *then* Perilaku NPC *is* Bertahan (0.5).
6. *If* Jarak *is* Dekat (0.5) *and* Stamina *is* Sedang (0.5) *and* Air *is* Tinggi (0.5) *then* Perilaku NPC *is* Menyelamatkan diri (0.5).
7. *If* Jarak *is* Dekat (0.5) *and* Stamina *is* Kuat (0.5) *and* Air *is* Rendah (0) *then* Perilaku NPC *is* Bertahan (0).
8. *If* Jarak *is* Dekat (0.5) *and* Stamina *is* Kuat (0.5) *and* Air *is* Sedang (0.5) *then* Perilaku NPC *is* Bertahan (0.5).
9. *If* Jarak *is* Dekat (0.5) *and* Stamina *is* Kuat (0.5) *and* Air *is* Tinggi (0.5) *then* Perilaku NPC *is* Menyelamatkan diri (0.5).
10. *If* Jarak *is* Sedang (0.5) *and* Stamina *is* Lemah (0) *and* Air *is* Rendah (0) *then* Perilaku NPC *is* Menyelamatkan diri (0).
11. *If* Jarak *is* Sedang (0.5) *and* Stamina *is* Lemah (0) *and* Air *is* Sedang (0.5) *then* Perilaku NPC *is* Menyelamatkan diri (0).
12. *If* Jarak *is* Sedang (0.5) *and* Stamina *is* Lemah (0) *and* Air *is* Tinggi (0.5) *then* Perilaku NPC *is* Menyelamatkan diri (0).
13. *If* Jarak *is* Sedang (0.5) *and* Stamina *is* Sedang (0.5) *and* Air *is* Rendah (0) *then* Perilaku NPC *is* Bertahan (0).

14. *If* Jarak *is* Sedang (0.5) *and* Stamina *is* Sedang (0.5) *and* Air *is* Sedang (0.5) *then* Perilaku NPC *is* Bertahan (0.5).
15. *If* Jarak *is* Sedang (0.5) *and* Stamina *is* Sedang (0.5) *and* Air *is* Tinggi (0.5) *then* (Perilaku NPC *is* Menyelamatkan diri (0.5).
16. *If* Jarak *is* Sedang (0.5) *and* Stamina *is* Kuat (0.5) *and* Air *is* Rendah (0) *then* Perilaku NPC *is* Bertahan (0).
17. *If* Jarak *is* Sedang (0.5) *and* Stamina *is* Kuat (0.5) *and* Air *is* Sedang (0.5) *then* Perilaku NPC *is* Bertahan (0.5).
18. *If* Jarak *is* Sedang (0.5) *and* Stamina *is* Kuat (0.5) *and* Air *is* Tinggi (0.5) *then* Perilaku NPC *is* Menyelamatkan diri (0.5).
19. *If* Jarak *is* Jauh (0) *and* Stamina *is* Lemah (0) *and* Air *is* Rendah (0) *then* Perilaku NPC *is* Menyelamatkan diri (0).
20. *If* Jarak *is* Jauh (0) *and* Stamina *is* Lemah (0) *and* Air *is* Sedang (0.5) *then* Perilaku NPC *is* Menyelamatkan diri (0).
21. *If* Jarak *is* Jauh (0) *and* Stamina *is* Lemah (0) *and* Air *is* Tinggi (0.5) *then* Perilaku NPC *is* Menyelamatkan diri (0).
22. *If* Jarak *is* Jauh (0) *and* Stamina *is* Sedang (0.5) *and* Air *is* Rendah (0) *then* Perilaku NPC *is* Bertahan (0).
23. *If* Jarak *is* Jauh (0) *and* Stamina *is* Sedang (0.5) *and* Air *is* Sedang (0.5) *then* Perilaku NPC *is* Bertahan (0).
24. *If* Jarak *is* Jauh (0) *and* Stamina *is* Sedang (0.5) *and* Air *is* Tinggi (0.5) *then* Perilaku NPC *is* Menyelamatkan diri (0).
25. *If* Jarak *is* Jauh (0) *and* Stamina *is* Kuat (0.5) *and* Air *is* Rendah *then* Perilaku NPC *is* Bertahan (0).
26. *If* Jarak *is* Jauh (0) *and* Stamina *is* Kuat (0.5) *and* Air *is* Sedang (0.5) *then* Perilaku NPC *is* Bertahan (0).
27. *If* Jarak *is* Jauh (0) *and* Stamina *is* Kuat (0.5) *and* Air *is* Tinggi (0.5) *then* Perilaku NPC *is* Menyelamatkan diri (0).

Dari hasil diatas maka akan dilanjutkan menggunakan aturan *Disjunction* (∨) dengan memilih derajat keangotaan maksimum. Sehingga dihasikan perilaku NPC is bertahan (0.5) ∨ perilaku NPC is menyelamatkan diri (0.5).

Maka akan di dapat grafik derajat keanggotaan model sugeno pada Gambar 3.5



Gambar 3.5 Grafik Derajat Keanggotaan Model Sugeno

C. *Defuzzification*

Defuzzification merupakan proses mengubah output *fuzzy* yang diperoleh dari mesin inferensi menjadi nilai tegas menggunakan fungsi keanggotaan yang sesuai dengan saat dilakukan *fuzzyfikasi*.

Berikut ini adalah rumus untuk mendapat nilai defuzzifikasi :

$$z = \frac{\sum \alpha i \; Zi}{\alpha i} \qquad (3.10)$$

αi = Nilai maksimal dari derajat keanggotaan pada aturan *disjunction*

zi = Nilai domain pada variabel linguistik perilaku

z = Nilai defuzzifikasi

Maka didapatkan hasil perhitungan sebagai berikut

$$z = \frac{0.5\ (50) + 0.5\ (100)}{0.5 + 0.5} = 75$$

Kesimpulan

Nilai yang didapat dari proses *defuzzyfication* tersebut selanjutnya digunakan untuk penentuan keputusan dengan aturan dari tabel 3.1 sebagai berikut :

1. Bertahan 0 – 50
2. Menyelamatkan diri 51 – 100

Maka hasil nilai dari 75 adalah menyelamatkan diri

3.5 Analisis Kebutuhan Non Fungsional

Analisis kebutuhan 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. Analisis kebutuhan ini juga menentukan spesifikasi masukan yang diperlukan sistem, keluaran yang akan dihasilkan sistem dan proses yang dibutuhkan untuk mengolah masukan sehingga menghasilkan suatu keluaran yang diinginkan. Pada analisis kebutuhan sistem non fungsional ini dijelaskan analisis kebutuhan perangkat lunak, analisis kebutuhan perangkat keras, dan analisis pengguna.

3.5.1 Analisis Kebutuhan Perangkat Lunak

Perangkat lunak yang dibutuhkan untuk implementasi algoritma pada *Game Rescue Flood Victim* ini adalah :

1. Sistem Operasi *Microsoft Windows 8.*
2. *Unity version 4.3.4.*
3. *Adobe Photoshop CS3.*

3.5.2 Analisis Kebutuhan Perangkat Keras

Komputer terdiri dari perangkat keras dan perangkat lunak yang saling berinteraksi. Perangkat lunak memberikan instruksi-instruksi kepada perangkat keras untuk melakukan suatu tugas tertentu, sehingga dapat menjalankan suatu sistem di dalamnya.

Perangkat keras minimum bagi pengguna untuk dapat memainkan *game Rescue Flood Victim* ini dengan baik adalah sebagai berikut :

1. *Processor Pentium IV* 1,50 Ghz
2. RAM 256 MB
3. VGA 128 MB
4. *Harddisk* kosong tersedia 250 MB

3.5.3 Analisis Pengguna

Adapun kriteria pengguna yang dapat menguji *game* ini adalah sebagai berikut:

1. Pengguna tidak buta angka.
2. Pengguna tidak buta huruf.
3. Pengguna dapat membaca.

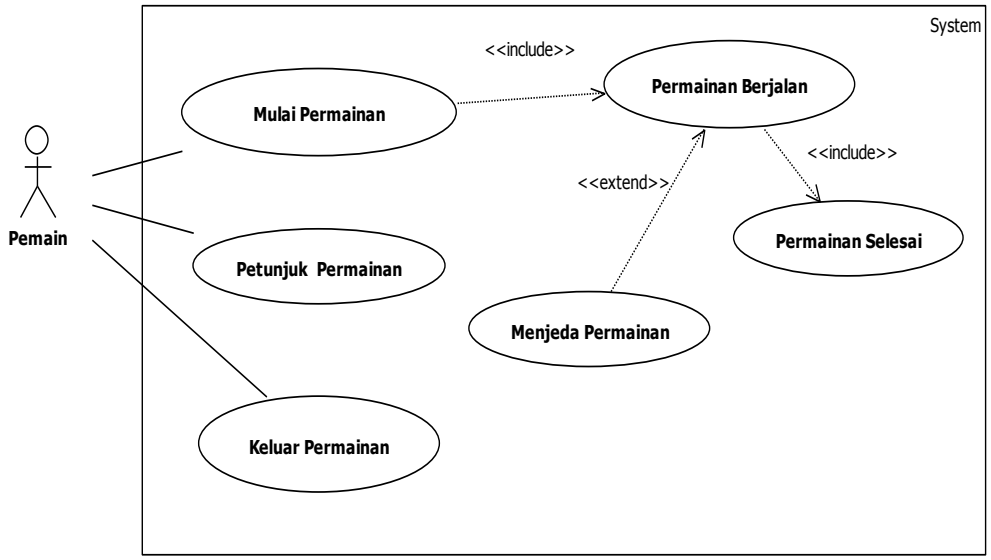
3.6 Analisis Kebutuhan Fungsional

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.

1. *Use Case* Diagram

Use Case atau diagram *use case* merupakan pemodelan untuk kelakuan (behavior) sistem informasi yang akan dibuat. *Use Case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat [6].

Berikut ini adalah perancangan proses-proses yang terdapat pada *game Rescue Flood Victim*, yang digambarkan dengan *Use Case Diagram* yang dapat dilihat 5



Gambar 3.6 *Use Case Diagram*

Use Case terdiri dari tiga bagian yaitu definisi *actor*, definisi *Use Case* dan skenario *Use Case*.

1.1 Definisi *Actor*

Definisi *Actor* berfungsi untuk menjelaskan *Actor* yang terdapat pada *Use Case Diagram*. Definisi *Actor* diterangkan pada Tabel 3.3.

Tabel 3.3 Definisi *Actor*

No	*Actor*	Deskripsi
1	Pemain	Orang yang memainkan *game*

1.2 Definisi *Use Case*

Definisi *Use Case* berfungsi untuk menjelaskan fungsi *Use Case* yang terdapat pada *Use Case Diagram*. Definisi *Use Case* diterangkan pada Tabel 3.4.

Tabel 3.4 Definisi *Use Case*

No	*Use Case*	Deskripsi
1	Mulai Permainan	Proses untuk memulai permainan
2	Permainan Berjalan	Proses untuk memainkan permainan
3	Menjeda Permainan	Proses untuk menjeda permainan
4	Permainan Selesai	Proses untuk memberhentikan permainan
5	Petunjuk Permainan	Proses untuk menampilkan petunjuk permainan
6	Keluar permainan	Proses untuk keluar dari permainan

1.3 Skenario *Use Case*

Skenario setiap bagian pada *use case* menunjukkan proses apa yang terjadi pada setiap bagian didalam *use case* tersebut, dimana pengguna memberikan perintah pada setiap bagian dan respon apa yang diberikan oleh sistem kepada pengguna setelah pengguna memberikan perintah pada setiap bagian - bagian *use case*.

Tabel 3.5 Skenario Memulai Permainan

Indentifikasi	
No	1
Nama	Mulai Permainan
Tujuan	Untuk memulai permainan
Deskripsi	Proses untuk memulai permainan
Aktor	Pemain
Skenario Utama	

Kondisi Awal	Aktor berada di menu utama
Aksi Aktor	**Reaksi Sistem**
	1. Sistem menampilkan menu utama
2. Pemain memilih menu mulai	3. Sistem memulai permainan
	4. Sistem menampilkan layar permainan
Kondisi Akhir	Permainan di mainkan

Tabel 3.6 Skenario Permainan Berjalan

Indentifikasi	
No	2
Nama	Permainan Berjalan
Tujuan	Untuk menjalankan permainan dan menghitung data algoritma
Deskripsi	Proses untuk memainkan permainan dengan menghitung data pada algoritma *fuzzy*.
Aktor	Pemain
Skenario Utama	
Kondisi Awal	Aktor berada di dalam permainan
Aksi Aktor	**Reaksi Sistem**
1. Pemain menekan tombol A	2. Sistem menggerakan perahu ke kiri
3. Pemain menekan tombol D	4. Sistem menggerakan perahu ke kanan
	5. Sistem menampilkan proses logika fuzzy
	6. Sistem menghitung jarak
	7. Sistem menghitung stamina
	8. Sistem menghitung air
	9. Sistem menampilkan hasil keputusan

10. Pemain melanjutkan permainan	
Kondisi Akhir	Permainan di mainkan

Tabel 3.7 Skenario Menjeda Permainan

Indentifikasi	
No	4
Nama	Menjeda Permainan
Tujuan	Untuk menjeda permainan
Deskripsi	Proses untuk menjeda permainan
Aktor	Pemain
Skenario Utama	
Kondisi Awal	Aktor berada di dalam permainan
Aksi Aktor	**Reaksi Sistem**
	1. Sistem menampilkan layar permainan
2. Pemain menekan tombol ESC	3. Sistem menjeda permainan
4. Pemain pilih menu lanjut	5. Sistem menampilkan kembali layar permainan
6. Pemain pilih menu ulangi	7. Sistem menampilkan kembali layar permainan
8. Pemain pilih menu ulangi	9. Sistem kembali ke menu utama
Kondisi Akhir	Permainan dimainkan

Tabel 3.8 Skenario Permainan Selesai

Indentifikasi	
No	4
Nama	Permainan Selesai
Tujuan	Untuk menyelesaikan permainan

Deskripsi	Proses untuk memberhentikan permainan
Aktor	Pemain
Skenario Utama	
Kondisi Awal	Aktor berada di dalam permainan
Aksi Aktor	**Reaksi Sistem**
1. Pemain melanjutkan permainan	2. Sistem menampilkan misi selesai
3. Pemain memilih menu ulangi	4. Sistem menampilkan kembali layar permainan
5. Pemain memilih menu keluar	6. Sistem kembali ke menu utama
Kondisi Akhir	Permainan selesai

Tabel 3.9 Skenario Petunjuk Permainan

Indentifikasi	
No	5
Nama	Petunjuk permainan
Tujuan	Untuk keluar dari permainan
Deskripsi	Proses untuk menampilkan petunjuk permainan
Aktor	Pemain
Skenario Utama	
Kondisi Awal	Aktor berada di menu utama
Aksi Aktor	**Reaksi Sistem**
	1. Sistem menampilkan menu utama
2. Pemain pilih menu petunjuk	3. Sistem menampilkan petunjuk permainan
Kondisi Akhir	Petunjuk ditampilkan

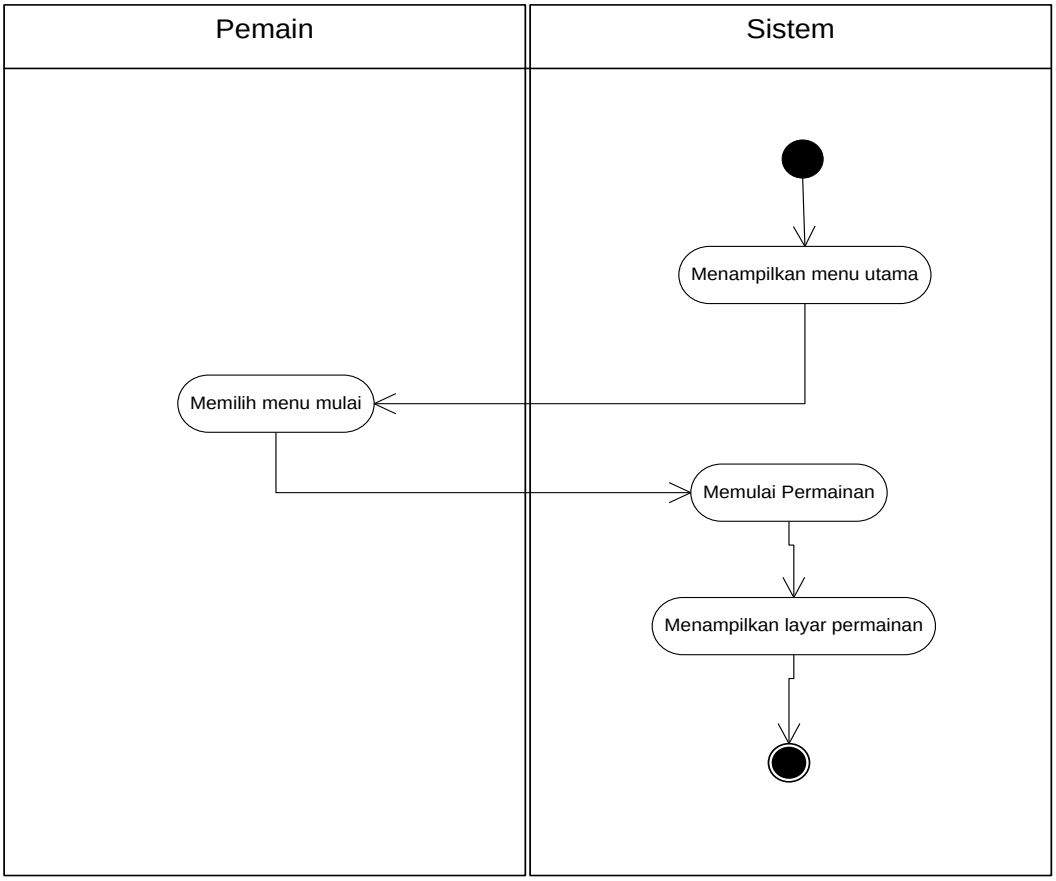
Tabel 3.10 Skenario Keluar Permainan

Indentifikasi		
No	6	
Nama	Keluar permainan	
Tujuan	Untuk keluar dari permainan	
Deskripsi	Proses untuk keluar permainan	
Aktor	Pemain	
Skenario Utama		
Kondisi Awal	Aktor berada di menu utama	
Aksi Aktor		**Reaksi Sistem**
		1. Sistem menampilkan menu utama
2. Pemain pilih menu keluar		3. Sistem keluar dari permainan
Kondisi Akhir	Keluar permainan	

2. *Activity* Diagram

Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir [6]. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. *Activity diagram* merupakan *state diagram* khusus, dimana sebagian besar *state* adalah action dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelum (*internal processing*).

1. *Activity Diagram* Proses Memulai Permainan

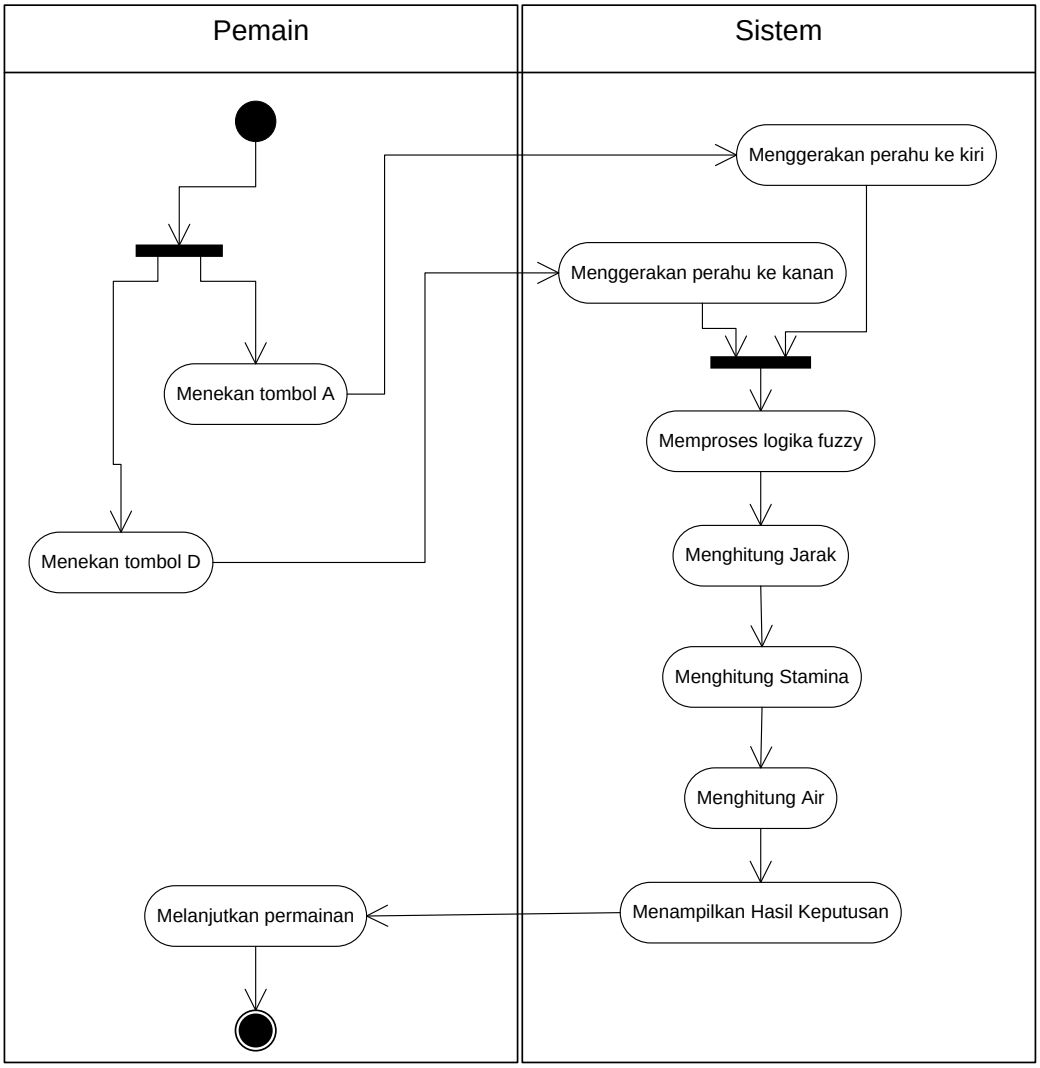


Gambar 3.7 *Activity Diagram* Proses Memulai Permainan

Tabel 3.11 Penjelasan Proses Memulai Permainan

Deskripsi	Proses Memulai Permainan
Normal Flow	1. Sistem menampilkan menu utama 2. Pemain memilih menu mulai 3. Sistem memulai permainan 4. Sistem menampilkan layar permainan

2. *Activity Diagram* Proses Permainan Berjalan



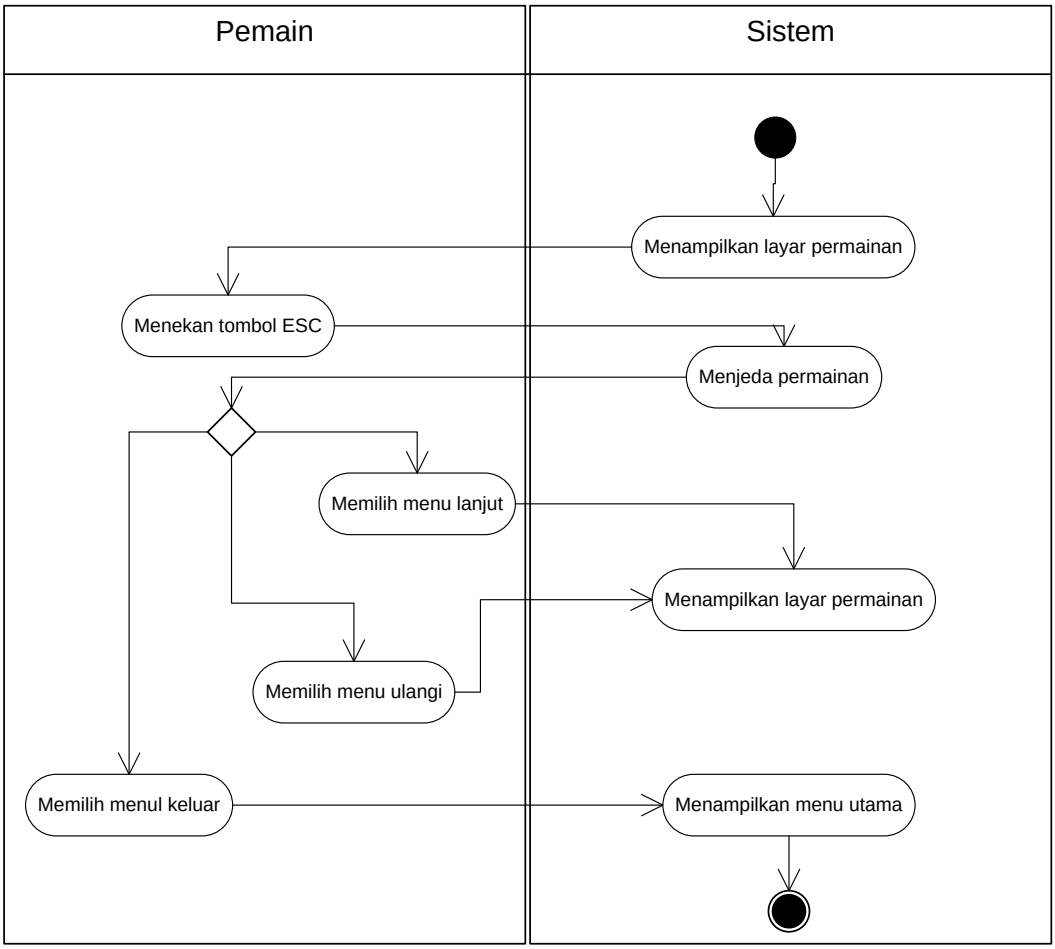
Gambar 3.8 *Activity Diagram* Proses Permainan Berjalan

Tabel 3.12 Penjelasan Proses Permainan Berjalan

Deskripsi	Proses Permainan Berjalan
Normal Flow	1. Pemain menekan tombol A 2. Sistem menggerakan perahu ke kiri 3. Pemain menekan tombol D 4. Sistem menggerakan perahu ke kanan 5. Sistem memproses logika fuzzy 6. Sistem menghitung jarak

	7. Sistem menghitung stamina 8. Sistem menghitung air 9. Sistem menampilkan hasil keputusan 10. Pemain melanjutkan permainan

3. *Activity Diagram* Proses Menjeda Permainan



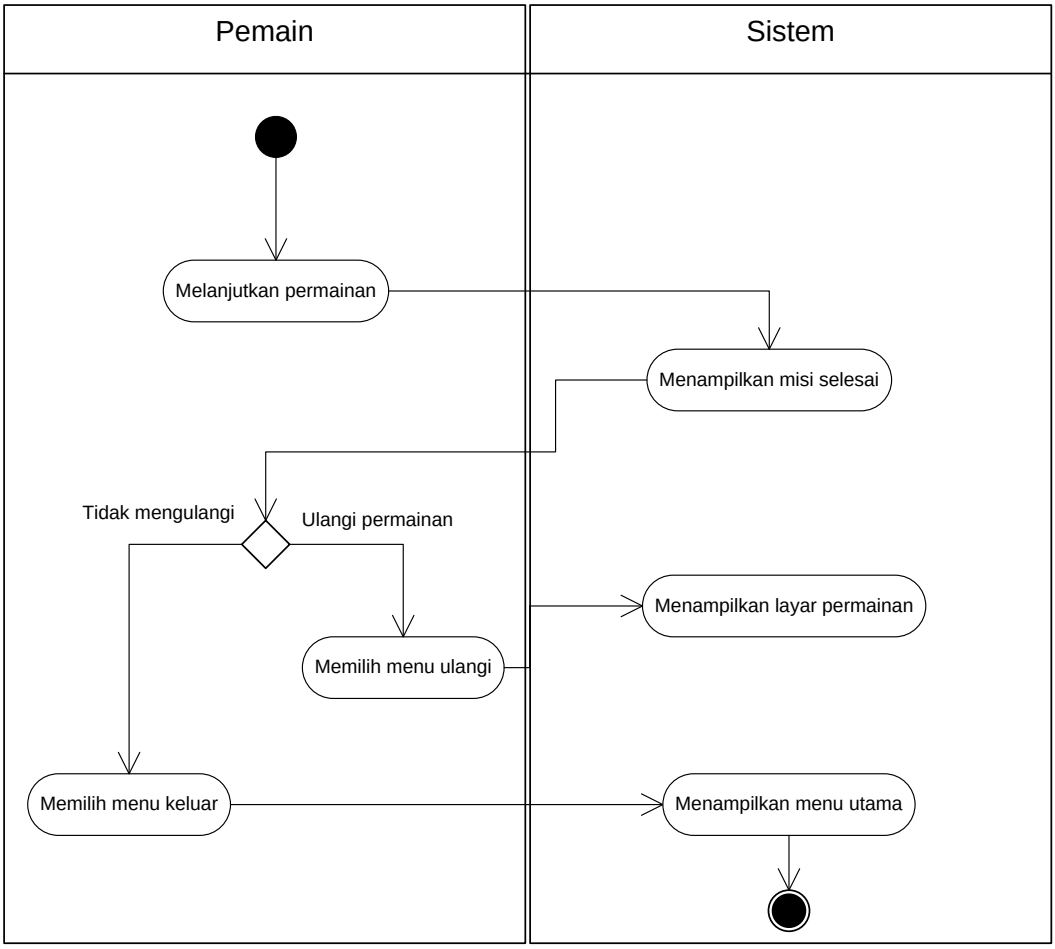
Gambar 3.9 *Activity Diagram* Proses Menjeda Permainan

Tabel 3.13 Penjelasan Proses Menjeda Permainan

Deskripsi	Proses Menjeda Permainan
Normal Flow	1. Sistem menampilkan layar permainan 2. Pemain menekan tombol ESC 3. Pemain memilih menu lanjut

	4. Sistem menampilkan layar permainan 5. Pemain memilih menu ulangi 6. Sistem menampilkan layar permainan 7. Pemain memilih menu keluar 8. Sistem kembali ke menu utama

4. *Activity Diagram* Proses Permainan Selesai



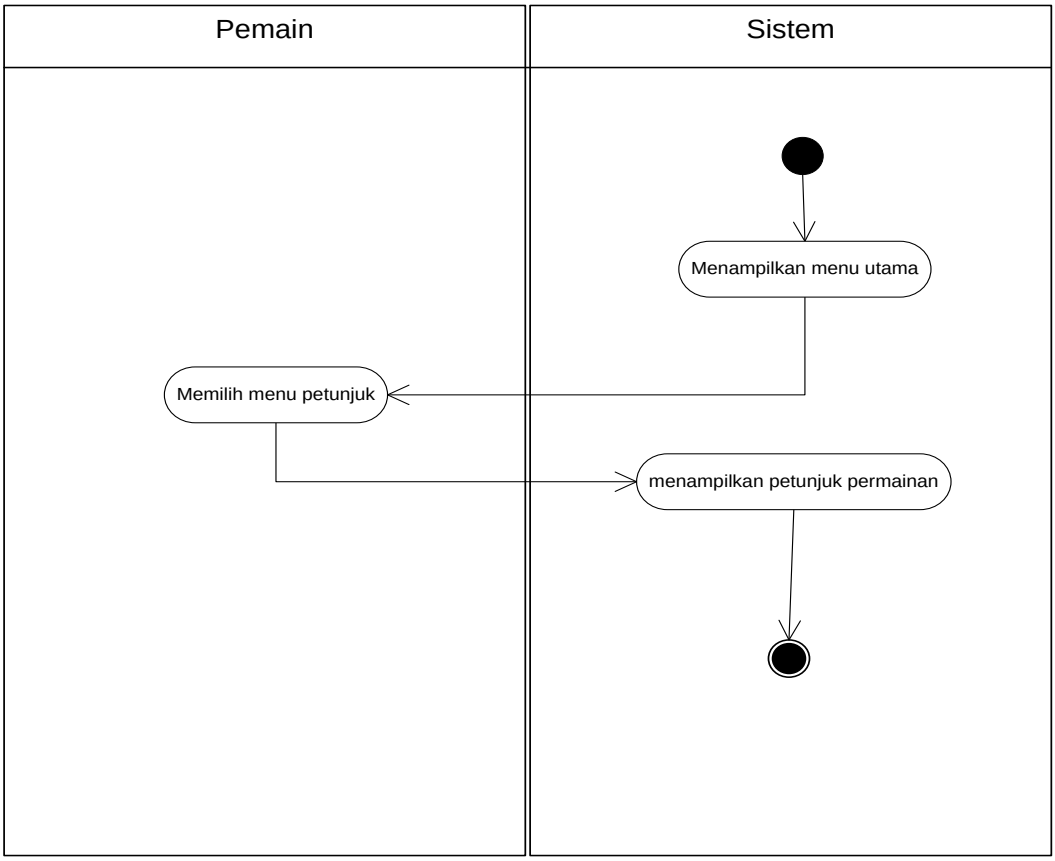
Gambar 3.10 *Activity Diagram* Proses Permainan Selesai

Tabel 3.14 Penjelasan Proses Permainan Selesai

Deskripsi	Proses Permainan Selesai
Normal Flow	1. Pemain melanjutkan permainan 2. Sistem menampilkan misi selesai

	3. Jika pemain ingin mengulangi permainan, maka pemain bisa menekan tombol ulangi dan sistem dapat menampilkan kembali layar permainan 4. Jika pemain tidak ingin mengulangi permainan, maka pemain bisa menekan tombol keluar dan sistem kembali ke tampilan menu utama

5. *Activity Diagram* Proses Petunjuk Permainan



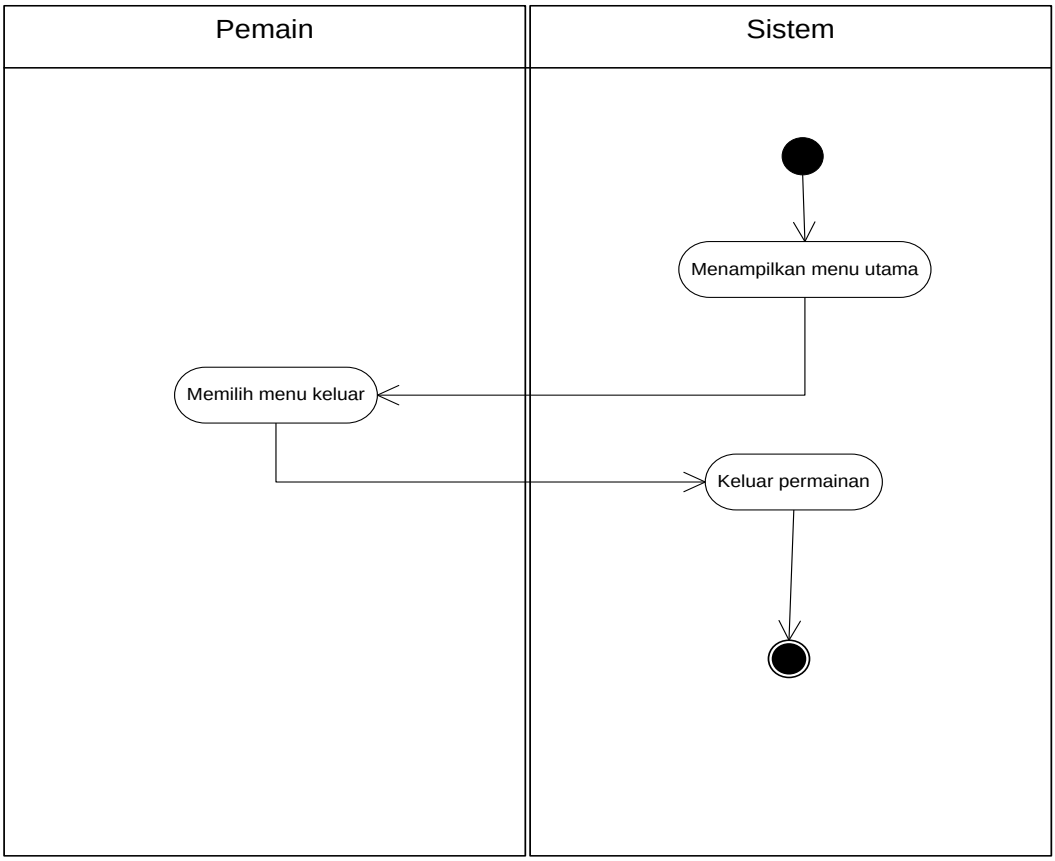
Gambar 3.11 *Activity Diagram* Proses Menampilkan Petunjuk

Tabel 3.15 Penjelasan Proses Menampilkan Petunjuk

Deskripsi	Proses menampilkan petunjuk
Normal Flow	1. Sistem menampilkan menu utama 2. Pemain memilih menu petunjuk

	3. Sistem menampilkan petunjuk permainan

6. *Activity Diagram* Proses Keluar Permainan



Gambar 3.12 *Activity Diagram* Proses Keluar Permainan

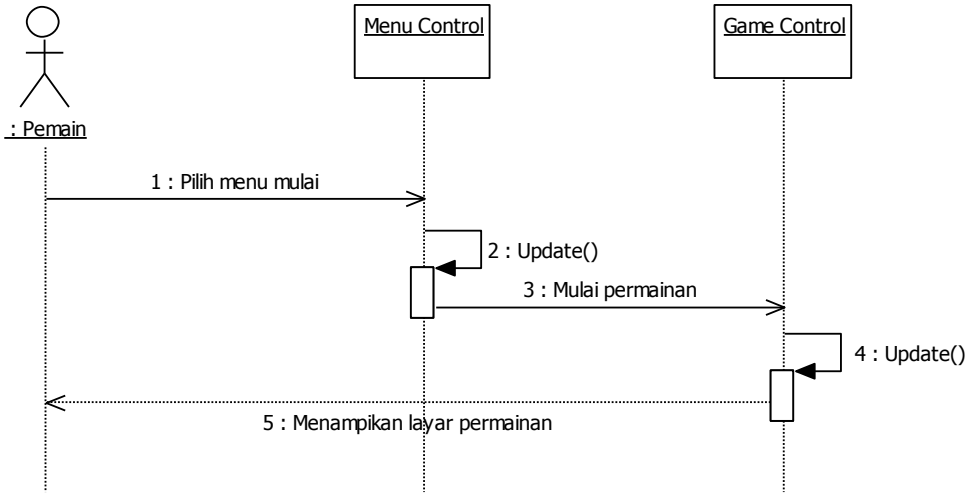
Tabel 3.16 Penjelasan Proses Keluar Permainan

Deskripsi	Proses Keluar Permainan
Normal Flow	1. Sistem menampilkan menu utama 2. Pemain memilih menu keluar 3. Sistem keluar permainan

3. *Sequence* Diagram

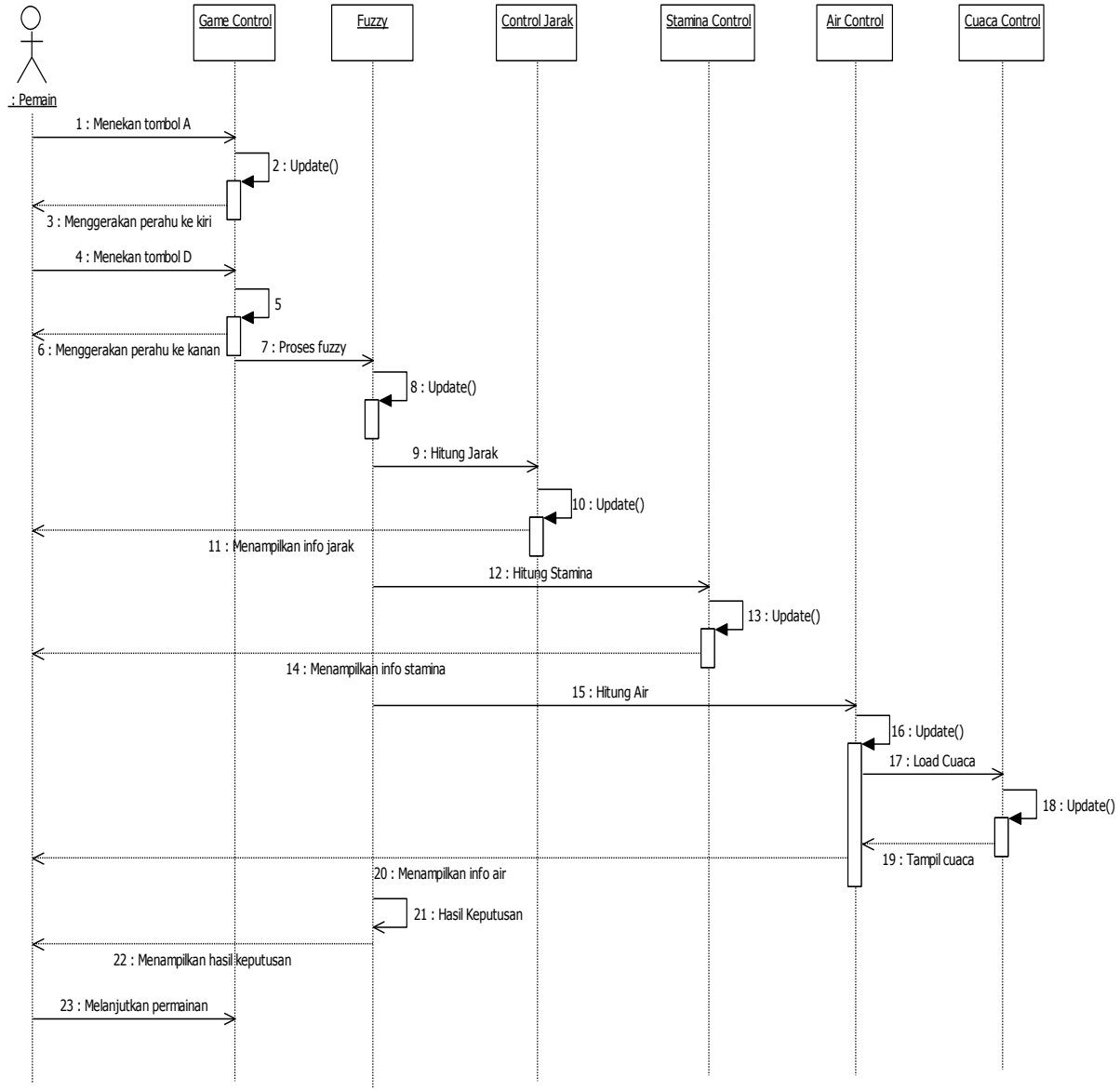
Diagram *sequence* atau *sequence diagram* menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek [6].

3.1 Diagram *Sequence* Memulai Permainan



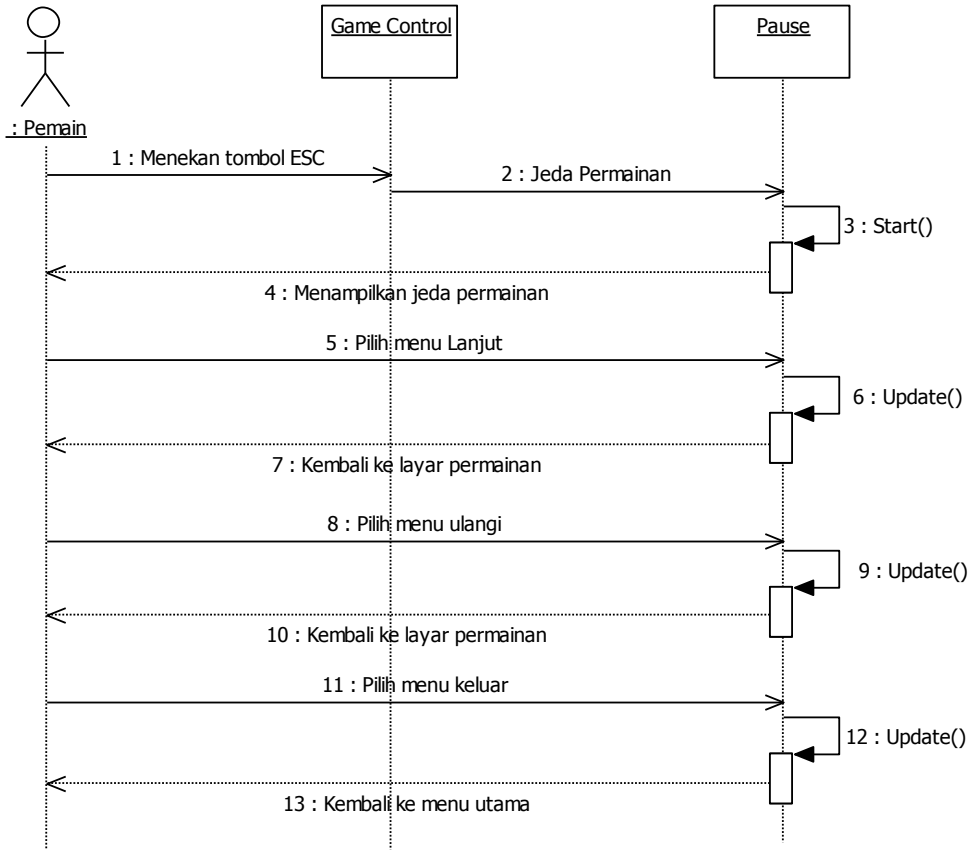
Gambar 3.13 Diagram *Sequence* Memulai Permainan

3.2 Diagram *Sequence* Permainan Berjalan



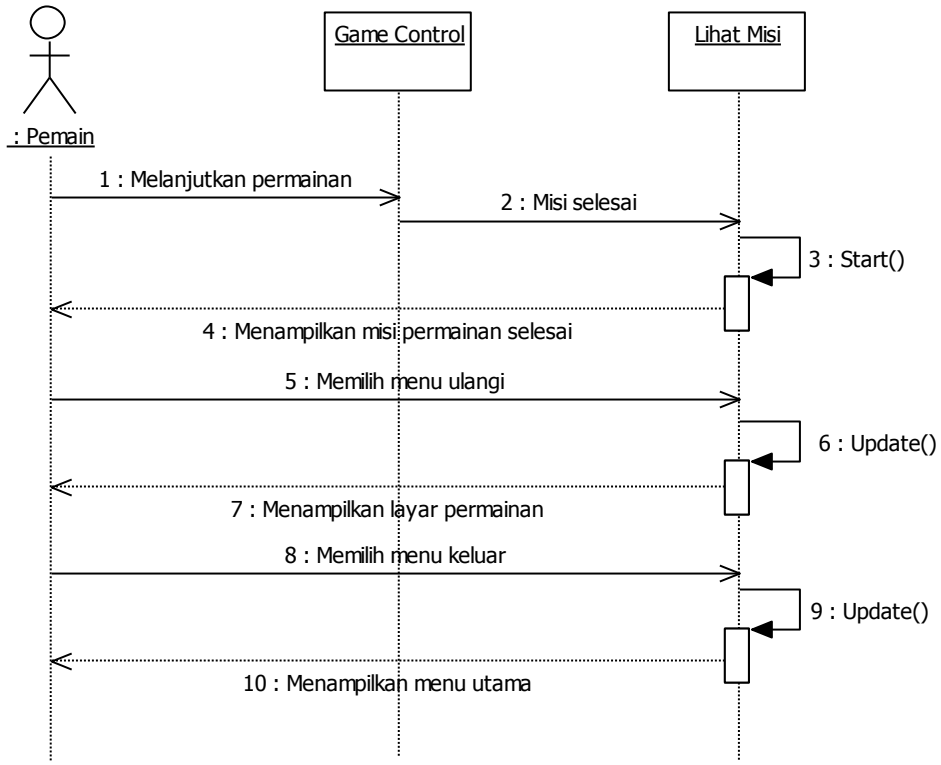
Gambar 3.14 Diagram *Sequence* Permainan Berjalan

3.3 Diagram *Sequence* Menjeda Permainan



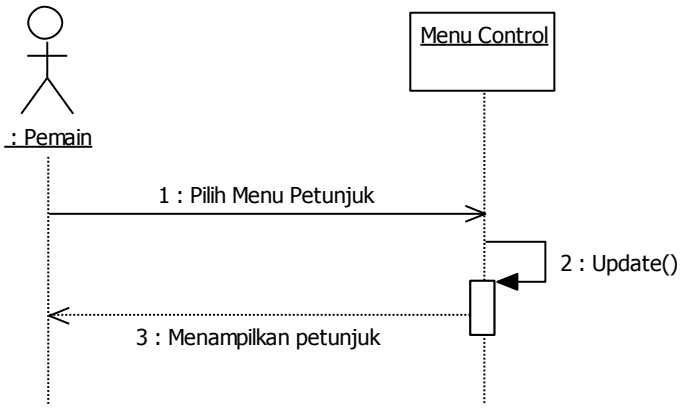
Gambar 3.15 Diagram *Sequence* Menjeda Permainan

3.4 Diagram *Sequence* Permainan Selesai



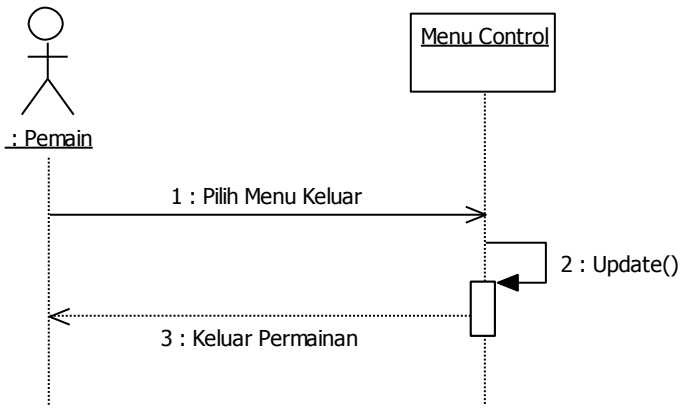
Gambar 3.16 Diagram *Sequence* Permainan Selesai

3.5 Diagram *Sequence* Petunjuk Permainan



Gambar 3.17 Diagram *Sequence* Petunjuk Permainan

3.6 Diagram *Sequence* Keluar Permainan

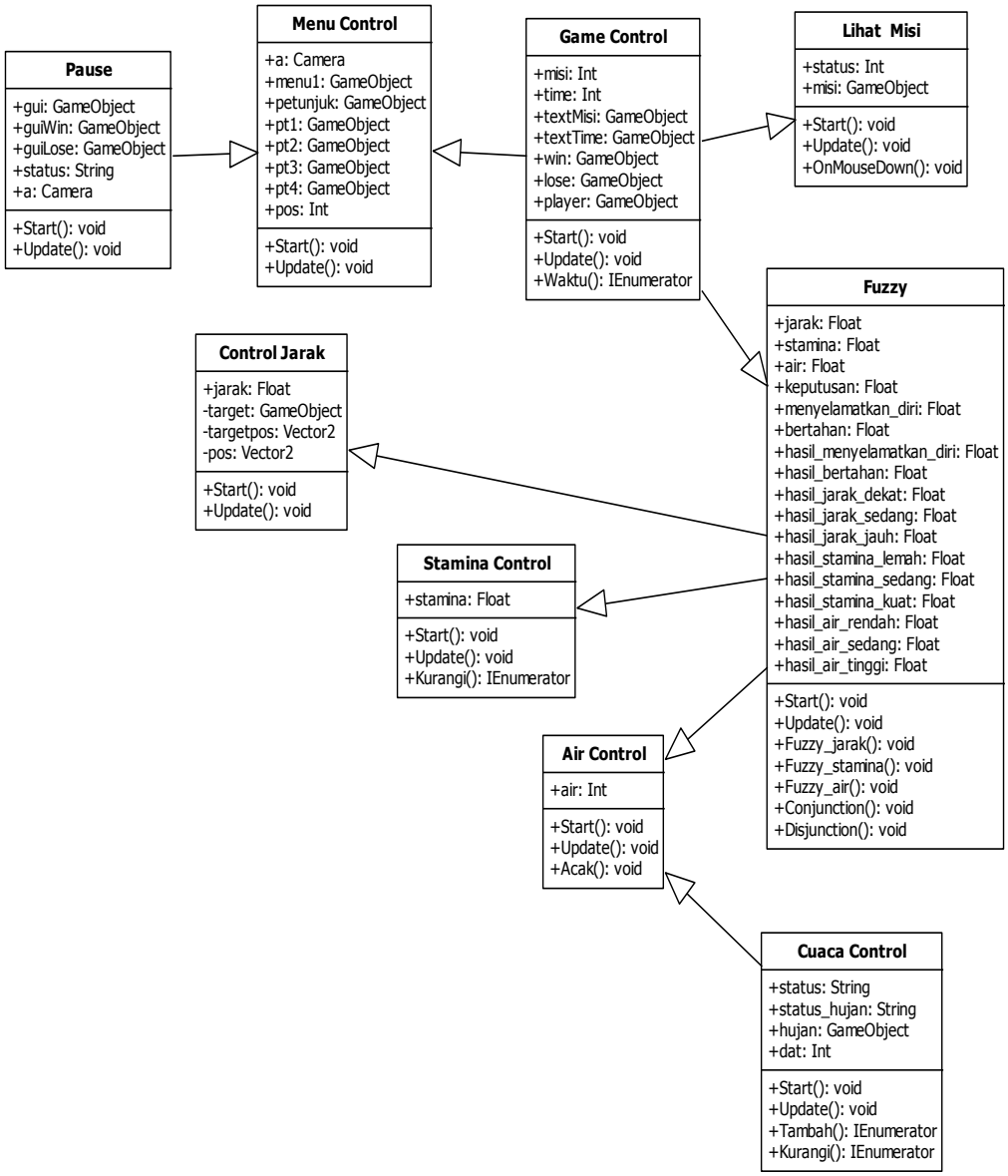


Gambar 3.18 Diagram *Sequence* Keluar Permainan

4. *Class* Diagram

Diagram kelas atau atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang dibuat untuk membangun sistem [6].

Berikut ini adalah perancangan struktur sistem yang terdapat pada *game Rescue Flood Victim*, yang digambarkan dengan class diagram yang dapat dilihat pada Gambar 3.19.



Gambar 3.19 *Class Diagram*

3.7 Perancangan Sistem

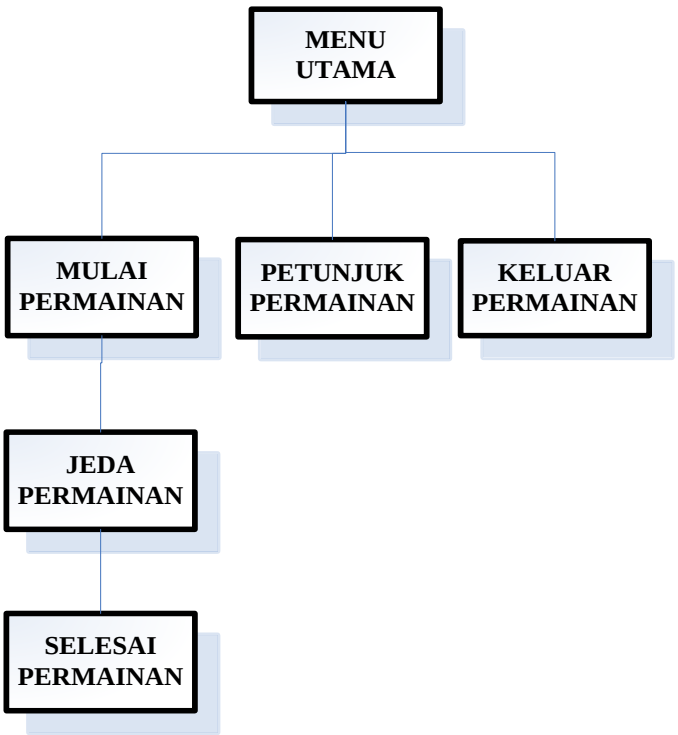
Perancangan adalah suatu bagian dari metodologi pengembangan suatu perangkat lunak yang dilakukan setelah tahapan untuk memberikan gambaran secara terperinci. Dalam tahap ini digambarkan rancangan sistem yang akan dibangun sebelum dilakukan pengkodean ke dalam suatu bahasa pemrograman.

3.7.1 Perancangan Perangkat Lunak

Perancangan perangkat lunak adalah kegiatan untuk merancang aplikasi yang akan di bangun, tahapannya dimulai dari perancangan arsitektur menu, lalu berlanjut ke perancangan antar muka dan jaringan semantik.

3.7.1.1 Perancangan Arsitektur Menu

Untuk mempermudah pembuatan sistem diperlukan perancangan struktur menu program yang akan dibangun. Perancangan struktur menu program ini membantu dalam merancang bagian-bagian dari sistem yang sebenarnya dan untuk mengetahui bagian mana yang terlebih dahulu nantinya yang akan diakses setelah program tersebut selesai. Berikut di bawah ini menunjukkan perancangan arsitektur menu dari sistem yang akan dibangun.

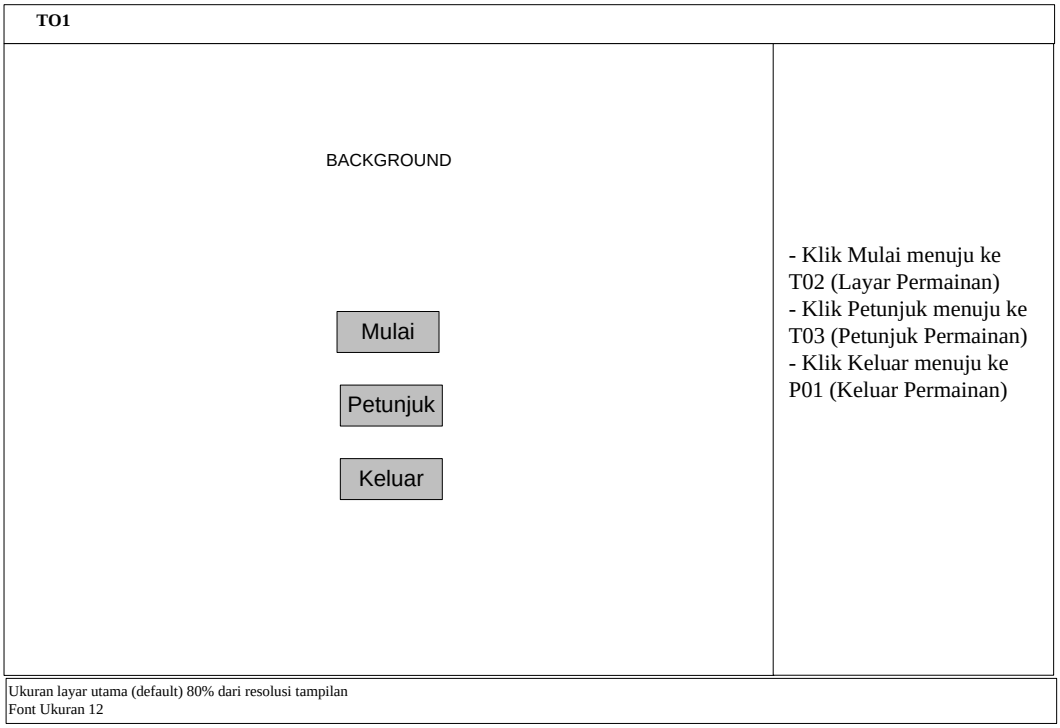


Gambar 3.20 Perancangan Arsitektur Menu

3.7.1.2 Perancangan Antarmuka

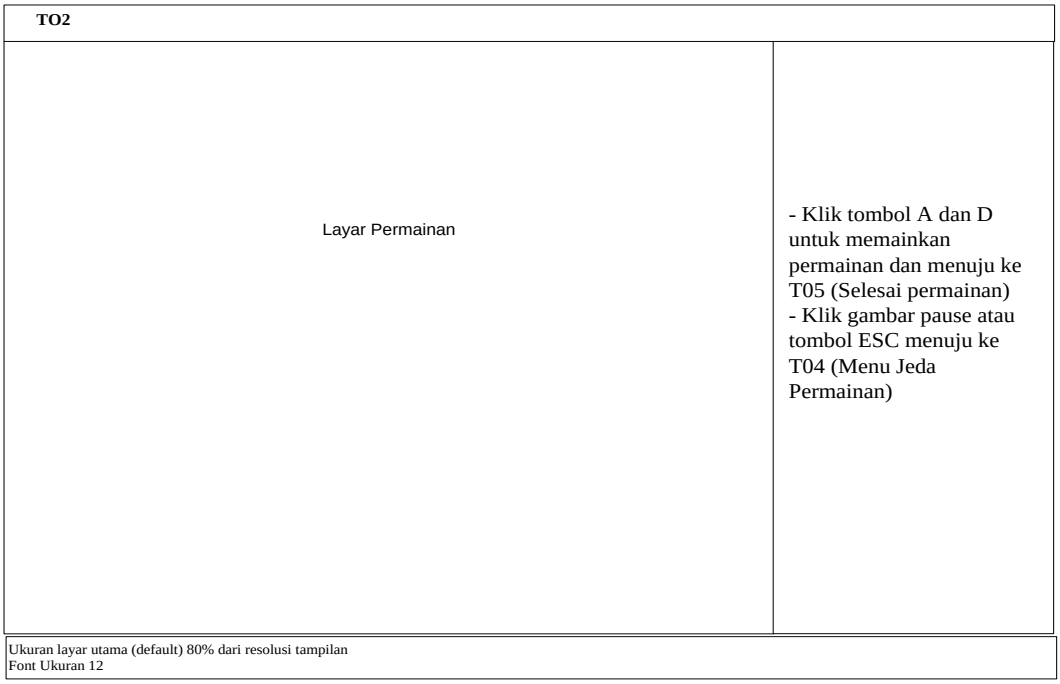
Perancangan antarmuka dilakukan untuk mempermudah dalam mengimplementasikan sistem yang akan dibangun. Berikut ini adalah antar muka pada *game Rescue Flood Victim*.

- Antarmuka Menu Utama (T01)



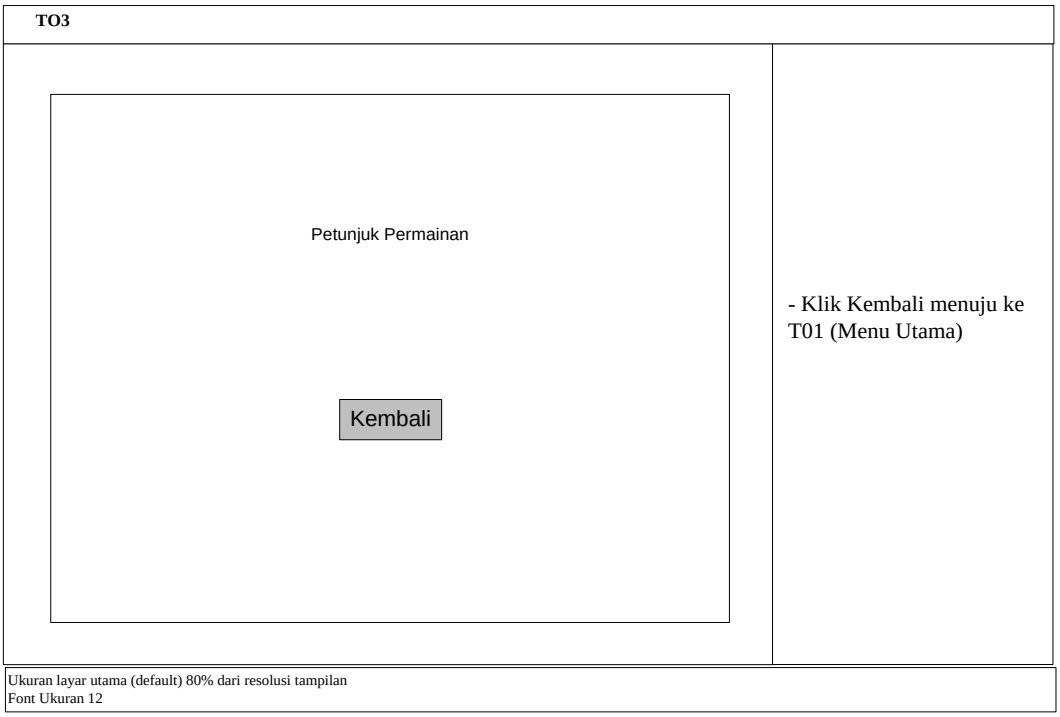
Gambar 3.21 Antarmuka Menu Utama

2. Antarmuka Layar Permainan (T02)



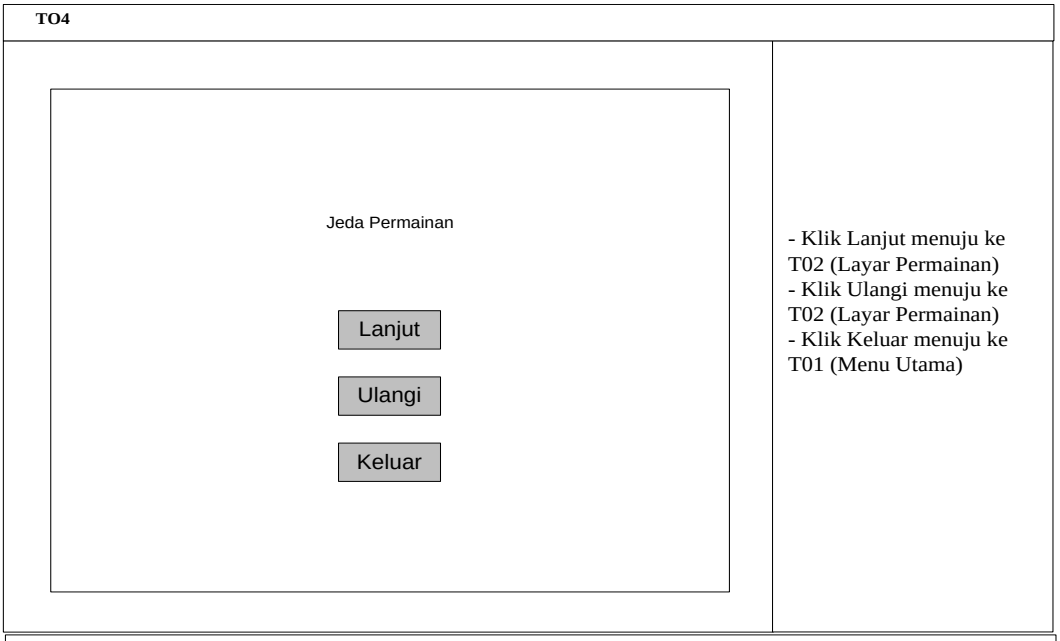
Gambar 3.22 Antarmuka Layar Permainan

3. Antarmuka Menu Petunjuk Permainan (T03)



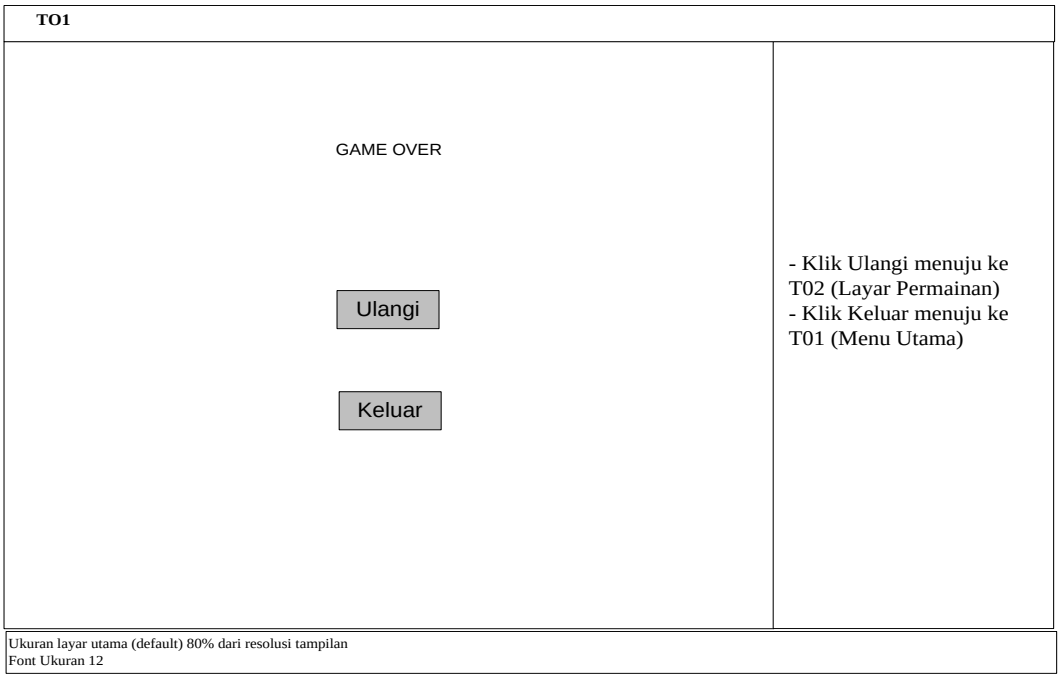
Gambar 3.23 Antarmuka Petunjuk Permainan

4. Antarmuka Menu Menjeda Permainan (T04)



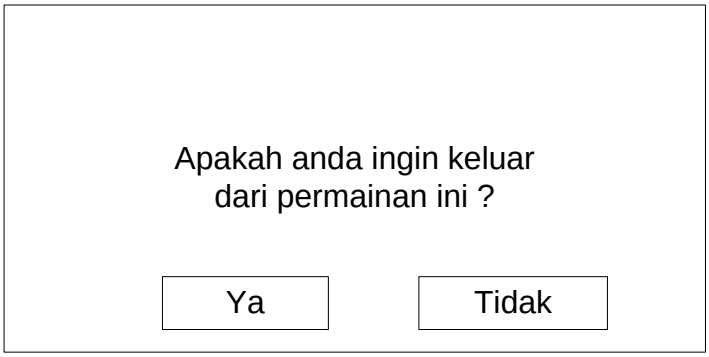
Gambar 3.24 Antarmuka Menu Jeda Permainan

5. Antarmuka Permainan Selesai (T05)



Gambar 3.25 Antarmuka Permainan Selesai

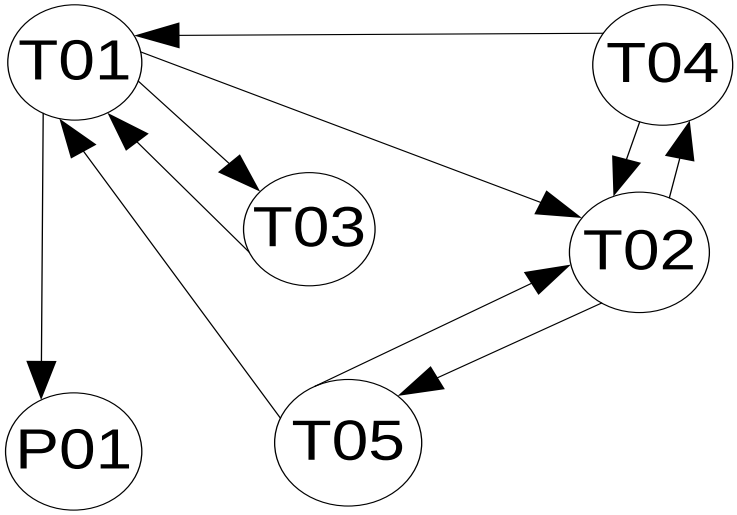
6. Peringatan Keluar Permainan (P01)



Gambar 3.26 Peringatan Keluar Permainan

3.7.1.3 Jaringan Semantik

Struktur aplikasi ini dibuat secara moduler, yaitu program dipecah menjadi modul-modul kecil yang mudah dibuat, mudah dites, dan mudah dimodifikasi. Dalam pembuatan *game* ini menggunakan alat bantu bagan struktural yang disebut juga jaringan semantik.



Gambar 3.27 Jaringan Semantik

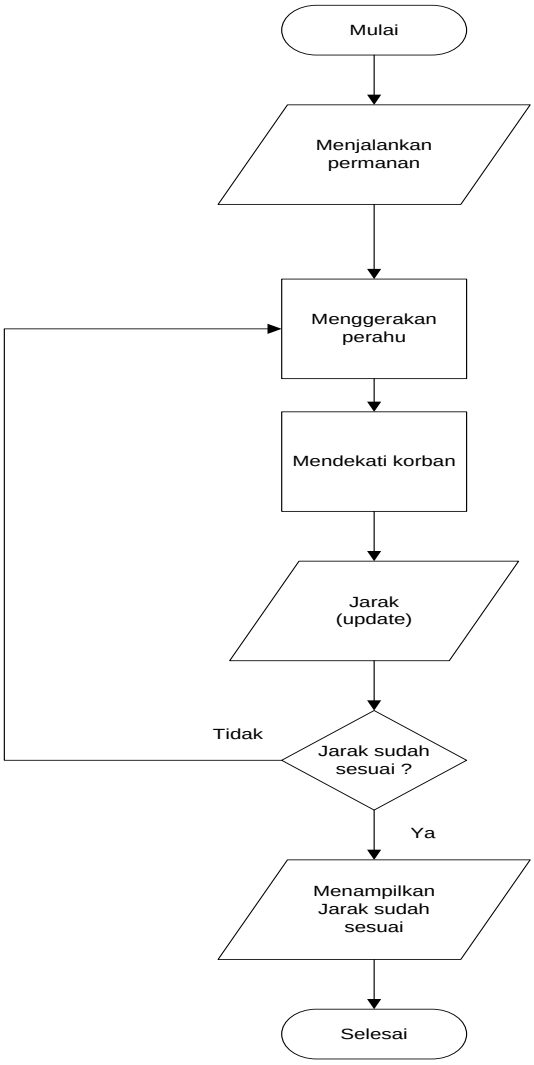
3.7.2 Perancangan Method

Perancangan komponen *method* merupakan perancangan yang dibuat setelah perancangan arsitektur menu, antar muka, dan jaringan semantik.

Perancangan ini berfungsi untuk mendeskripsikan method-method yang berada di dalam aplikasi. Adapun method-method yang terdapat dalam *game Rescue Flood Victim* yang akan dibangun adalah sebagai berikut.

a. *Method* Control Jarak

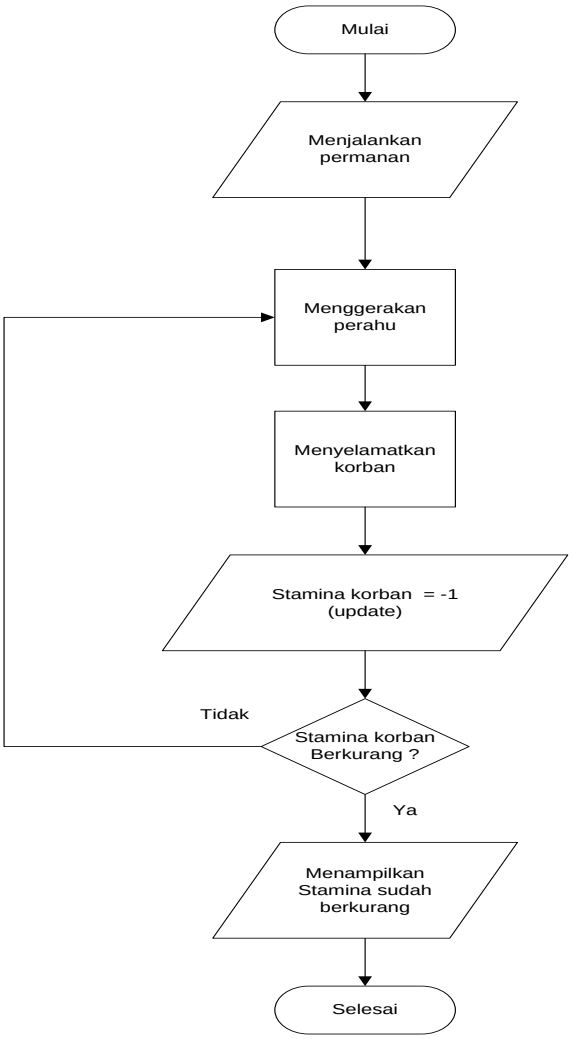
Method ini untuk menguji dan mengecek apakah jarak sudah sesuai dengan tempat penyelamatan korban atau tidak dan pengujian sudah tepat.



Gambar 3.28 *Flowchart Method* Control Jarak

b. *Method* Stamina Control

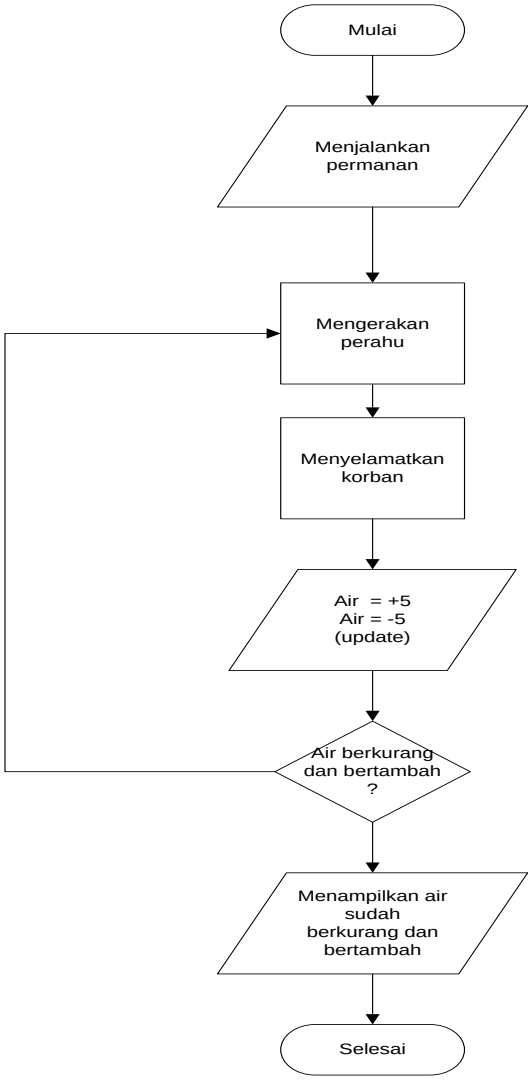
Method ini untuk menguji dan mengecek apakah stamina sudah berkurang pada korban atau tidak dan pengujian sudah tepat.



Gambar 3.29 *Flowchart Method* Stamina Control

c. *Method* Air Control

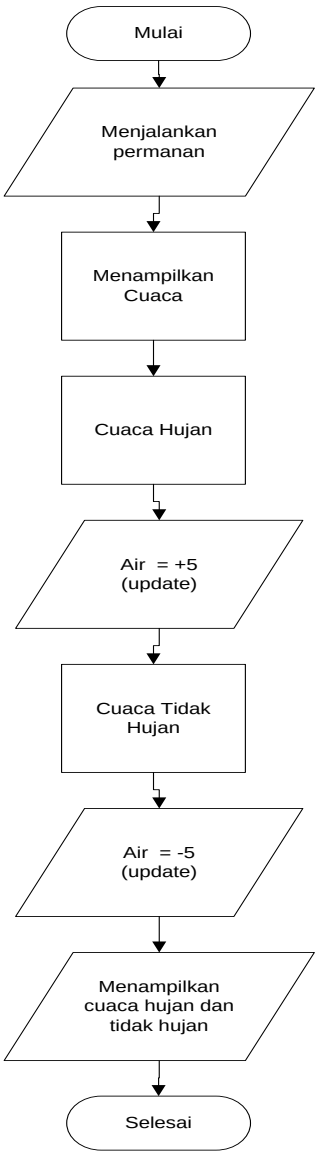
Method ini untuk menguji dan mengecek apakah air sudah bertambah atau tidak dan pengujian sudah tepat.



Gambar 3.30 *Flowchart Method* Air Control

d. *Method* Cuaca Control

Method ini untuk menguji dan mengecek apakah cuaca hujan atau tidak hujan berpengaruh terhadap air dan pengujian sudah tepat.



Gambar 3.31 *Flowchart Method* Cuaca Control