# BAB 3
# ANALISIS DAN KEBUTUHAN ALGORITMA

## 3.1 Analisis Masalah

Analisis masalah adalah langkah awal dari analisis sistem. Langkah ini diperlukan untuk mengetahui permasalahan apa saja yang terjadi pada sistem, oleh karena itu langkah pertama adalah menganalisis masalah yang muncul. Masalah yang muncul yaitu :

1. Bagaimana mengimplementasikan algoritma *boids* untuk perilaku ikan berkelompok pada permainan menjaring ikan?
2. Bagaimana pengaruh pengaturan perilaku yang diterapkan pada perilaku ikan berkelompok pada permainan menjaring ikan ini?

## 3.2 Analisis Perilaku Ikan

Perilaku ikan berkelompok merupakan perilaku sosial yang dipengaruhi oleh beberapa faktor. Salah satu faktornya adalah perlindungan diri dari pemangsa atau predator. Perilaku sosial ikan berkelompok tidak lepas dari salah satu perilaku alami dari ikan yaitu ikan akan bergerak menjauh atau menghindar dari pemangsa atau predator, maupun jaring yang merupakan ancaman bagi ikan.

Ikan yang bergerak secara berkelompok bergerak kearah yang sama bersama ikan lainnya, baik itu berputar di satu titik ataupun berpindah tempat. Dalam pergerakannya, ikan akan menjaga keteraturan geraknya untuk menghindari terjadinya tabrakan dengan ikan lain.

### 3.2.1 Analisis Perilaku Ikan dalam *Game*

Perilaku ikan berkelompok dalam permainan menjaring ikan ini merupakan implementasi dari algoritma *boids*. Perilaku ikan dalam permainan menjaring ikan ini dipengaruhi oleh tiga aturan dari algoritma *boids* yaitu *alignment*, *separation* dan *cohesion*.

Perilaku ikan yang dapat mensejajarkan diri dan bergerak menuju ke arah yang sama merupakan pengaruh dari aturan *alignment*. Perilaku ikan yang dapat mempertahankan atau memelihara jarak agar tidak terjadi benturan atau kepadatan

merupakan pengaruh dari aturan *separation*. Perilaku ikan yang dapat bergerak menuju posisi rata-rata dari kawanannya merupakan pengaruh dari aturan *cohesion*.

Perilaku ikan yang dapat bergerak menghindari jaring ketika jaring digerakan oleh pemain merupakan pengaruh dari pengaturan perilaku *obstacle avoidance*.

## 3.3 Analisis *Game*

Permainan yang dibangun ini adalah permainan berbasis *desktop* yang berjudul menjaring ikan dengan *genre arcade*, permainan ini merupakan permainan yang akan dibuat berupa *prototype*.

### 3.3.1 *Gameplay*

Dalam permainan ini, ikan akan berenang secara berkelompok. Pemain harus menjaring ikan untuk mendapatkan nilai. Pergerakan jaring dapat dikontrol dengan menggunakan *mouse*, pemain harus mengumpulkan poin sebanyak-banyakya sebelum permainan berakhir. Permainan akan berakhir jika waktu yang tersedia sudah habis.

### 3.3.2 Waktu

Waktu yang akan disediakan di dalam *prototype* permainan menjaring ikan ini adalah 30 detik. Waktu akan mulai dihitung mundur saat pemain menekan tombol mulai.

### 3.3.3 Skor

Aturan penilaian dalam *prototype* permainan menjaring ikan ini adalah pemain akan mendapatkan 5 poin untuk setiap ikan yang ditangkap, tidak ada nilai maksimum untuk pemain karena pemain harus mengumpulkan poin sebanyak-banyaknya.

## 3.4 Analisis Masukan

Algoritma *boids* yang menggabungkan tiga pengaturan perilaku sederhana (*separation*, *alignment* dan *cohesion*) memiliki beberapa masukan dasar, yaitu :

1. Kecepatan

   Merupakan nilai untuk kecepatan untuk setiap ikan dengan satuan px/s Kecepatan minimun untuk ikan bergerak adalah 2.0 px/s dan kecepatan maksimum untuk ikan bergerak adalah 8.0 px/s.

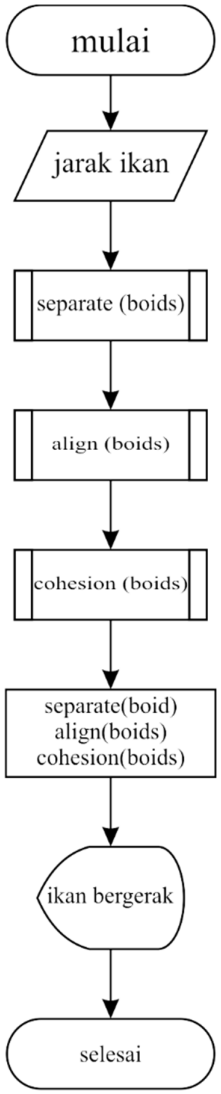
2. Jarak

   Merupakan nilai jarak antara ikan dengan tetangganya dan jarak antara ikan dengan jaring (*obstacle*).

## 3.5 *Flowchart* Algoritma *Boids*

Dalam game yang akan dibangun, penggunaan algoritma *boids* hanya untuk pergerakan ikan secara berkelompok. Berikut adalah *flowchart* algoritma *boids*.

1. *Flowchart* algoritma *boids*



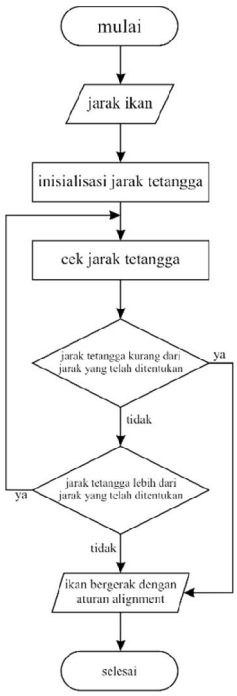

Gambar 3.1 *Flowchart* Algoritma *Boids*

2. *Flowchart separation rule*



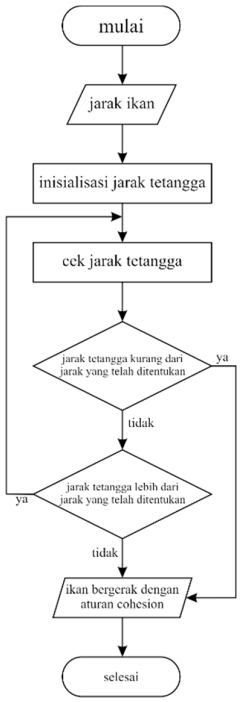

Gambar 3.2 *Flowchart Separation Rule*

3. *Flowchart alignment rule*




Gambar 3.3 *Flowchart Alignment Rule*

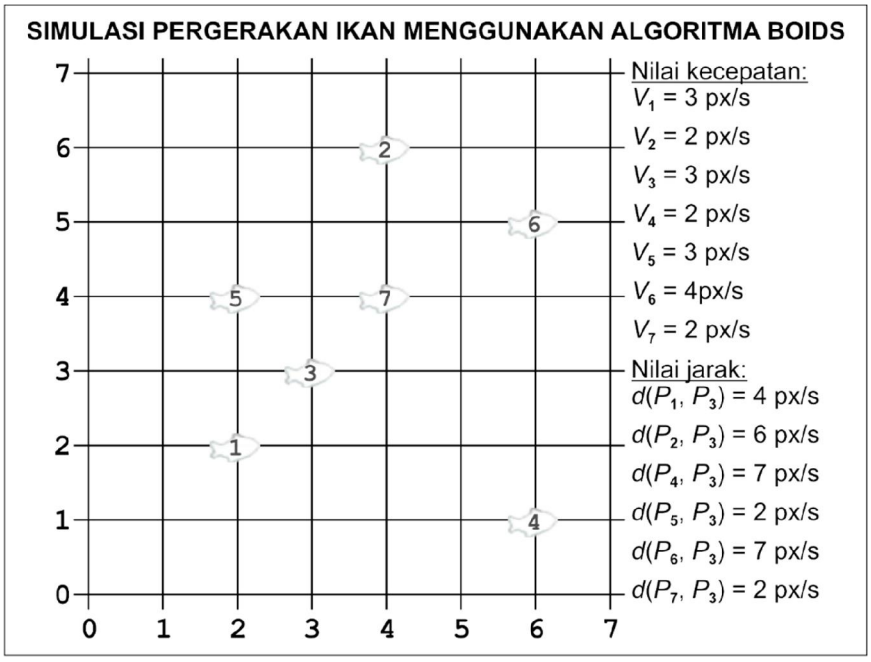
4. *Flowchart cohesion rule*




Gambar 3.4 *Flowchart Cohesion Rule*

### 3.6 Analisis Algoritma *Boids*

Algoritma *boids* dalam permainan menjaring ikan ini digunakan untuk mensimulasikan perilaku berkelompok pada ikan. Algoritma *boids* pada permainan menjaring ikan ini diterapkan pada ikan selaku NPC (*Non Playable Character*). Pada awal kemunculan ikan, ikan akan muncul pada posisi acak dengan nilai kecepatan yang berbeda-beda. Contoh kemunculan ikan dengan nilai kecepatan acak dapat dilihat pada gambar 3.5 dibawah ini.




Gambar 3.5 Posisi ikan saat pertama kali muncul dengan kecepatan acak.

Selanjutnya pergerakan ikan akan dipengaruhi oleh tiga aturan algoritma *boids* yaitu *separation, alignment* dan *cohesion*. Masing-masing dari tiga aturan ini akan saling mempengaruhi kecepatan dan arah gerak dari setiap ikan. Berikut adalah contoh kasus dari algoritma *boids* :

**1. *Alignment rule***

Di dalam aturan *alignment*, ikan akan menyelaraskan kecepatannya dengan ikan lain yang berada di dekatnya. Dalam contoh kasus *alignment rule* ini, *ikan 3* akan menyelaraskan kecepatan dengan tetangganya yaitu *ikan 1*, *ikan 5* dan *ikan 7*. Maka pergerakan *ikan 3* selanjutnya akan dipengaruhi oleh aturan *alignment*, aturan *alignment* dapat dihitung dengan menggunakan persamaan 2.1 yaitu :

$$d(Px, Pb) \leq d1 \cap d(Px, Pb) \geq d2 \Rightarrow Var = \frac{1}{n}\sum_{x}^{n} vx$$

Diketahui : $d(P_1, P_3) = 4$ px $V_1 = 3$ px/s $V_7 = 2$ px/s

$d(P_2, P_3) = 6$ px $V_2 = 2$ px/s $d_1 = 5$ px

$d(P_4, P_3) = 7$ px $V_3 = 3$ px/s $d_2 = 0$ px

$d(P_5, P_3) = 2$ px $V_4 = 2$ px/s

$d(P_6, P_3) = 7$ px $V_5 = 3$ px/s

$d(P_7, P_3) = 2$ px $V_6 = 4$ px/s

Ditanyakan : $Var$ = … ?

Jawab :

Diketahui bahwa ikan yang memenuhi syarat untuk dilakukan perhitungan *alignment* adalah *ikan 1*, *ikan 5* dan *ikan 7*, yaitu :

Ikan 1 : 4px ≤ 5px ∩ 4px ≥ 0px

Ikan 5 : 2px ≤ 5px ∩ 2px ≥ 0px

Ikan 7 : 2px ≤ 5px ∩ 2px ≥ 0px

maka dapat dilakukan perhitungan :

$$Var = \frac{1}{n}\sum_{x}^{n} vx$$

$$Var = \frac{1}{3}\ V_1 + V_5 + V_7$$

$$Var = \frac{1}{3}3 + 3 + 2$$

$$Var = \frac{1}{3}8$$

$$\underline{\boldsymbol{Var = 2.67\ px/s}}$$

Perubahan kecepatan *ikan 3* yang dipengaruhi oleh aturan *alignment* dapat dilihat pada tabel 3.1.

Tabel 3.1 Perubahan kecepatan setelah dipengaruhi aturan *alignment*

| | ***Vg*** |
|---|---|
| ***V awal*** | 3 px/s |
| ***Var*** | 2,67 px/s |

Maka kecepatan *ikan 3* yang baru setelah dipengaruhi oleh aturan *alignment* adalah 2.67px/s yang asalnya adalah 3px/s.

**2. *Separation rule***

Di dalam aturan *separation*, ikan akan mempertahankan atau memelihara jarak agar tidak terjadi benturan atau kepadatan. Dalam contoh kasus alignment rule ini, *ikan 3* akan mempertahankan atau memelihara jarak dari tetangganya yaitu *ikan 1*, *ikan 5* dan *ikan 7*. Maka pergerakan *ikan 3* selanjutnya akan dipengaruhi oleh aturan *separation*, aturan *separation* dapat dihitung dengan menggunakan persamaan 2.2 yaitu :

$$d(Px, Pb) \leq d1 \cap d(Px, Pb) \geq d2 \Rightarrow Vsr = \sum_{x}^{n} \frac{Vx + Vb}{d(Px, Pb)}$$

Diketahui : $d(P_1, P_3) = 4$ px $V_1 = 3$ px/s $V_7 = 2$ px/s

$d(P_2, P_3) = 6$ px $V_2 = 2$ px/s $d_1 = 5$ px

$d(P_4, P_3) = 7$ px $V_3 = 3$ px/s $d_2 = 0$ px

$d(P_5, P_3) = 2$ px $V_4 = 2$ px/s

$d(P_6, P_3) = 7$ px $V_5 = 3$ px/s

$d(P_7, P_3) = 2$ px $V_6 = 4$ px/s

Ditanyakan : $Vsr$ = … ?

Jawab :

Diketahui bahwa ikan yang memenuhi syarat untuk dilakukan perhitungan *separation* adalah *ikan 1*, *ikan 5* dan *ikan 7*, yaitu :

Ikan 1 : 4px ≤ 5px ∩ 4px ≥ 0px

Ikan 5 : 2px ≤ 5px ∩ 2px ≥ 0px

Ikan 7 : 2px ≤ 5px ∩ 2px ≥ 0px

maka dapat dilakukan perhitungan :

$$Vsr = \sum_{x}^{n} \frac{Vx + Vb}{d(Px, Pb)}$$

$$Vsr = \frac{V5 + V3}{d(P5, P3)} + \frac{V1 + V3}{d(P1, P3)} + \frac{V7 + V3}{d(P7, P3)}$$

$$Vsr = \frac{3 + 3}{2} + \frac{3 + 3}{4} + \frac{2 + 3}{2}$$

$$Vsr = \frac{6}{4} + \frac{6}{2} + \frac{5}{2}$$

$$Vsr = \frac{6}{4} + \frac{12}{4} + \frac{10}{4}$$

$$Vsr = \frac{28}{4}$$

$$\underline{\boldsymbol{Vsr = 7\ px/s}}$$

Perubahan kecepatan *ikan 3* yang dipengaruhi oleh aturan *separation* dapat dilihat pada tabel 3.2.

Tabel 3.2 Perubahan kecepatan setelah dipengaruhi aturan *separation*

| | ***Vg*** |
|---|---|
| ***V awal*** | 3 px/s |
| ***Var*** | 7 px/s |

Maka kecepatan *ikan 3* untuk menghindari benturan atau kepadatan setelah dipengaruhi oleh aturan *separation* adalah 7px/s yang asalnya adalah 3px/s.

**3. *Cohesion rule***

Di dalam aturan *cohesion*, ikan akan bergerak menuju posisi rata-rata kawanan terdekatnya. Dalam contoh kasus *cohesion rule*, *ikan 3* akan bergerak menuju posisi rata-rata dari *ikan 1*, *ikan 5* dan *ikan 7*. Maka pergerakan *ikan 3* selanjutnya akan dipengaruhi aturan *cohesion*, aturan *cohesion* dapat dihitung dengan menggunakan persamaan 2.3 yaitu :

$$d(Px, Pb) \leq d1 \cap d(Px, Pb) \geq d2 \Rightarrow Pavg = \sum_{x}^{n} \frac{Px}{n}$$

Diketahui : $d(P_1, P_3) = 4$ px $V_1 = 3$ px/s $V_7 = 2$ px/s $P_4 = (6,1)$

$d(P_2, P_3) = 6$ px $V_2 = 2$ px/s $d_1 = 5$ px $P_5 = (2,4)$

$d(P_4, P_3) = 7$ px $V_3 = 3$ px/s $d_2 = 0$ px $P_6 = (6,5)$

$d(P_5, P_3) = 2$ px $V_4 = 2$ px/s $P_1 = (2,2)$ $P_7 = (4,4)$

$d(P_6, P_3) = 7$ px $V_5 = 3$ px/s $P_2 = (4,6)$

$d(P_7, P_3) = 2$ px $V_6 = 4$ px/s $P_3 = (3.3)$

Ditanyakan : $P_{avg}$ = … ?

Jawab :

Diketahui bahwa ikan yang memenuhi syarat untuk dilakukan perhitungan *cohesion* adalah *ikan 1*, *ikan 5* dan *ikan 7*, yaitu :

Ikan 1 : 4px ≤ 5px ∩ 4px ≥ 0px

Ikan 5 : 2px ≤ 5px ∩ 2px ≥ 0px

Ikan 7 : 2px ≤ 5px ∩ 2px ≥ 0px

maka dapat dilakukan perhitungan :

$$Pavg = \sum_{x}^{n} \frac{Px}{n}$$

$$Pavg = \frac{P5}{n} + \frac{P1}{n} + \frac{P7}{n}$$

$$Pavg = \frac{(2,4)}{3} + \frac{(2,2)}{3} + \frac{(4,4)}{3}$$

$$Pavg = \frac{(8,10)}{3}$$

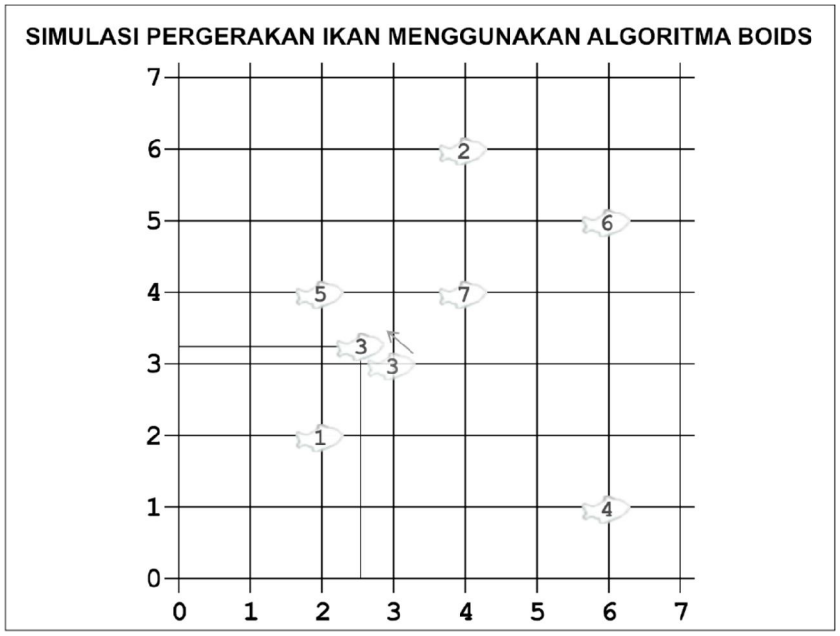
$$Pavg = (2.67, 3.33)$$

Perpindahan posisi *ikan 3* yang dipengaruhi oleh aturan *cohesion* dapat dilihat pada tabel 3.3.

Tabel 3.3 Perubahan posisi setelah dipengaruhi aturan *cohesion*

| | ***Pg*** |
|---|---|
| ***P awal*** | (3,3) |
| ***P avg*** | (2.67,3.33) |

Maka *ikan 3* akan bergerak menuju posisi rata-rata tetangganya terdekatnya yaitu ke titik (2.67, 3.33) yang asalnya berada pada titik (3,3).




Gambar 3.6 Ikan bergerak menuju posisi rata-rata tetangganya.

Pada dasarnya setiap ikan akan melakukan aturan *alignment, separation* dan *cohesion* secara terus menerus. Perhitungan diatas merupakan perhitungan dengan 1x iterasi.

**4. Kesimpulan**

Berdasarkan hasil analisis di atas, maka dapat disimpulkan bahwa pergerakan ikan akan dipengaruhi oleh tiga aturan dari algoritma *boids*, yaitu *alignment*, *separation* dan *cohesion*. Ini membuktikan bahwa algoritma *boids* dapat diterapkan pada simulasi pergerakan ikan secara berkelompok.

## 3.7 Analisis Kebutuhan Non Fungsional

Analisis kebutuhan non fungsional merupakan analisis untuk menentukan spesifikasi kebutuhan sistem. Spesifikasi kebutuhan sistem meliputi analisis perangkat keras atau hardware, analisis perangkat lunak atau software, dan analisis pengguna.

### 3.7.1 Analisis Kebutuhan Perangkat Lunak

Analisis perangkat lunak digunakan untuk mengetahui spesifikasi perangkat lunak yang digunakan untuk menjalankan permainan menjaring ikan, spesifikasi perangkat lunak yang digunakan untuk menjalankan permainan ini adalah sebagai berikut :

1. Sistem Operasi Windows 7 Ultimate 32 bit
2. Adobe Flash CS 4
3. Corel Draw X4

Sedangkan perangkat lunak yang dibutuhkan untuk memainkan permainan ini adalah *adobe flash player*, dengan spesifikasi sebagai berikut [14] :

1. Microsoft ® Windows ® XP (32 bit), Windows Server 2008 (32 bit), Windows Vista ® (32 bit), Windows 7 (32 bit dan 64 bit), Windows 8 (32 bit dan 64 bit), atau Windows Server 2012 (64 bit)
2. Internet Explorer 8.0 atau yang lebih baru, Mozilla Firefox 17 atau yang lebih baru, Google Chrome, atau Opera 11

### 3.7.2 Analisis Kebutuhan Perangkat Keras

Analisis perangkat keras digunakan untuk mengetahui spesifikasi perangkat keras yang digunakan untuk menjalankan permainan menjaring ikan, perangkat keras yang digunakan untuk menjalankan permainan ini adalah :

1. *Processor* @ 1.83GHz (2 CPUs)
2. *Memory* (RAM) 1536MB
3. *Display Memory* 256 MB
4. *Display Mode* 1280 x 800 (32 bit) (60Hz)
5. *Input Device Mouse* dan *Keyboard*

Sedangkan untuk spesifikasi perangkat keras yang dibutuhkan untuk dapat memainkan permainan ini adalah sebagai berikut [14] :

1. Procesor 2.33GHz atau prosesor x86-kompatibel lebih cepat.
2. 512MB RAM (1GB RAM direkomendasikan untuk netbook); 128MB memori grafis

### 3.7.3 Analisis Pengguna

Analisis pengguna digunakan untuk mengetahui spesifikasi pengguna agar dapat memainkan permainan ini. Spesifikasi minimal bagi pengguna dapat dilihat pada tabel 3.4.

Tabel 3.4 Spesifikasi Pengguna

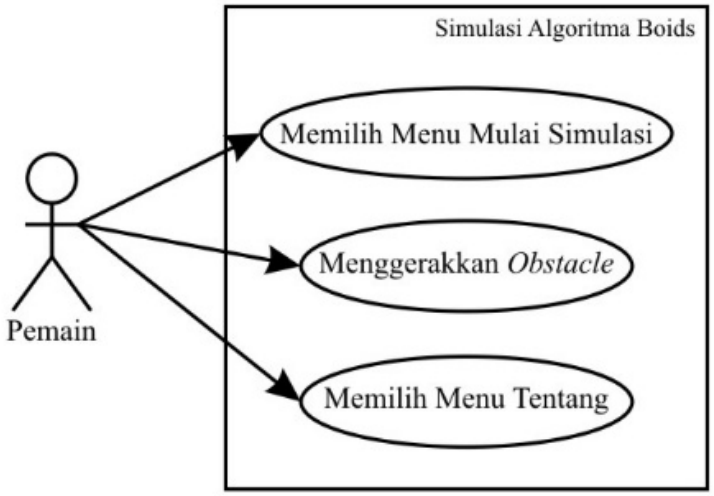
| **Karakteristik Fisik Pengguna** | |
|---|---|
| **Umur** | 18 tahun keatas |
| ***Gender*** | Laki-laki dan Perempuan |
| ***Disabilities*** | Tidak buta dan tidak mempunyai cacat fisik seperti cacat tangan |
| **Pengetahuan dan Pengalaman Pengguna** | |
| ***Computer literacy*** | Dapat menggunakan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

## 3.8 Analisis Kebutuhan Fungsional

Analisis kebutuhan fungsional merupakan analisis untuk menggambarkan sistem yang akan dibangun. Analisis kebutuhan fungsional dimodelkan dengan menggunakan UML, tahap-tahap pemodelannya antara lain adalah *use case diagram*, *use case scenario*, *activity diagram*, *class diagram* dan *sequence diagram*.

### 3.8.1 *Use Case Diagram*

*Use case diagram* merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem yang akan dibuat [2]. Berikut adalah *use case diagram prototype* permainan menjaring ikan dapat dilihat pada gambar 3.7.




Gambar 3.7 *Use Case Diagram Prototype* Permainan Menjaring Ikan

### 3.8.2 Definisi Aktor

Definisi aktor berfungsi untuk menjelaskan aktor yang terdapat pada *use case diagram*. Deskripsi pendefinisian aktor pada *prototype* permainan menjaring ikan dapat dilihat pada tabel 3.5.

Tabel 3.5 Tabel definisi aktor pada *prototype* permainan menjaring ikan

| No | Aktor | Deskripsi |
|---|---|---|
| 1 | Pemain | Orang yang memainkan permainan |

### 3.8.3 Definisi *Use Case*

Definisi *use case* berfungsi untuk menjelaskan proses yang terdapat pada *use case diagram*. Deskripsi pendefinisian *use case* pada *prototype* permainan menjaring ikan dapat dilihat pada tabel 3.6.

Tabel 3.6 Tabel definisi *use case* pada *prototype* permainan menjaring ikan

| No | Aktor | Deskripsi |
|---|---|---|
| 1 | Memilih Menu Mulai Simulasi | Proses untuk memulai simulasi dari pergerakan ikan |
| 2 | Menggerakan *Obstacle* | Proses untuk menggerakan *obstacle* |
| 3 | Memilih Menu Tentang | Proses untuk menampilkan informasi aplikasi |

### 3.8.4 *Use Case Scenario*

*Use case Scenario* mendeskripsikan urutan langkah-langkah dalam proses bisnis baik yang dilakukan aktor terhadap sistem maupun yang dilakukan oleh sistem terhadap aktor. Berdasarkan *use case* diagram, maka *use case scenario* untuk *prototype* permainan menjaring ikan terdiri dari mulai permainan, menggerakan *obstacle* dan tentang. Berikut adalah *use case scenario* pada *prototype* permainan menjaring ikan :

**1. *Use Case Scenario* Mulai Simulasi**

*Use case scenario* ini menjelaskan interaksi antara aktor pemain dengan *use case* mulai permainan yang dijelaskan pada tabel 3.7.

Tabel 3.7 Tabel *Use Case Scenario* Mulai Simulasi

| **Identifikasi** | | |
|---|---|---|
| **Nomor** | 1 | |
| **Nama** | Mulai Simulasi | |
| **Tujuan** | Memulai simulasi | |
| **Deskripsi** | Proses untuk memulai simulasi dari pergerakan ikan | |
| **Aktor** | Pemain | |
| **Skenario Utama** | | |
| **Kondisi awal** | Pemain berada di layar simulasi | |
| **Aksi Aktor** | | **Reaksi Sistem** |
| a. Memilih menu mulai simulasi | | |
| | | b. Menampilkan jendela simulasi<br>c. Ikan bergerak bebas di layar |

| | |
|---|---|
| | d. Ikan akan menghitung jarak dengan tetangganya sebagai kondisi untuk melakukan aturan *alignment*, *separation* dan *cohesion*.<br>e. Jika kondisi terpenuhi untuk melakukan aturan *alignment*, maka ikan akan melakukan perhitungan dari aturan alignment dengan cara menghitung jumlah tetangganya yang ada di dalam jarak yang telah ditentukan dan menghitung rata-rata kecepatan tetangga.<br>f. Ikan bergerak dengan menyelaraskan kecepatan (*alignment*) dengan tetangganya.<br>g. Jika kondisi tidak terpenuhi untuk melakukan perhitungan dari aturan *alignment*, maka ikan tidak akan melakukan aturan *alignment*.<br>h. Jika kondisi terpenuhi untuk melakukan aturan *separation*, maka ikan akan melakukan perhitungan dari aturan *separation* dengan cara menghitung jumlah tetangganya yang ada di dalam jarak yang telah ditentukan serta menghitung kecepatannya dengan kecepatan tetangganya dan menghitung jarak dengan tetangganya.<br>i. Ikan bergerak menjauh dari tetangganya (separation).<br>j. Jika kondisi tidak terpenuhi untuk melakukan perhitungan dari aturan *separation*, maka ikan tidak akan melakukan aturan *separation*.<br>k. Jika kondisi terpenuhi untuk melakukan aturan *cohesion*, maka ikan akan melakukan perhitungan dari aturan *cohesion* dengan cara |

| | |
|---|---|
| | menghitung jumlah tetangganya yang ada di dalam jarak yang telah ditentukan, lalu ikan akan menghitung posisi rata-rata dari tetangganya.<br>l. Ikan akan bergerak menuju posisi rata-rata tetangganya (*cohesion*).<br>m. Jika kondisi tidak terpenuhi untuk melakukan perhitungan dari aturan *cohesion*, maka ikan tidak akan melakukan aturan *cohesion*. |
| **Kondisi Akhir** | Ikan bergerak dengan aturan *alignment, separation* dan *cohesion*. |

**2. *Use Case Scenario* Menggerakan *Obstacle***

*Use case scenario* ini menjelaskan interaksi antara aktor pemain dengan *use case* menggerakan *obstacle* yang dijelaskan pada tabel 3.8.

Tabel 3.8 Tabel *Use Case Scenario* Menggerakan *Obstacle*

| **Identifikasi** | |
|---|---|
| **Nomor** | 2 |
| **Nama** | Menggerakan *Obstacle* |
| **Tujuan** | Menunjukan penerapan dari perilaku menghindari rintangan |
| **Deskripsi** | Proses untuk menggerakan *obstacle* |
| **Aktor** | Pemain |
| **Skenario Utama** | |
| **Kondisi awal** | Pemain berada di layar simulasi |
| **Aksi Aktor** | **Reaksi Sistem** |
| | a. Ikan bergerak bebas di layar dengan tiga aturan algoritma *boids* (*alignment*, *separation*, dan *cohesion*) |
| b. Menggerakan *Obstacle* | |

| | |
|---|---|
| | c. Ikan akan menghitung jarak dengan *obstacle* sebagai kondisi untuk dilakukan penghindaran.<br>d. Jika kondisi terpenuhi, maka ikan akan bergerak menjauh dari *obstacle.*<br>e. Jika kondisi tidak terpenuhi, maka ikan akan bergerak bebas kembali. |
| **Kondisi Akhir** | Ikan bergerak menghindari *obstacle* |

**3. *Use Case Scenario* Tentang**

*Use case scenario* ini menjelaskan interaksi antara aktor pemain dengan *use case* tentang yang dijelaskan pada tabel 3.9.

Tabel 3.9 Tabel *Use Case Scenario* Tentang

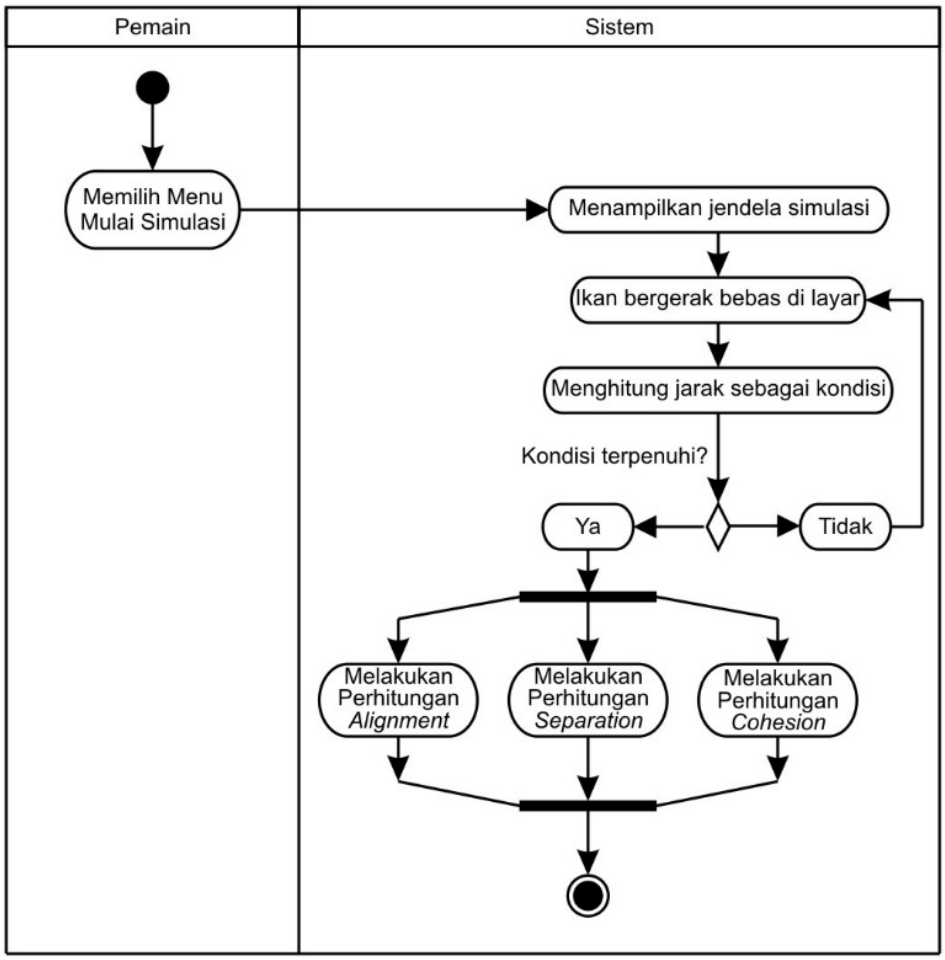
| **Identifikasi** | | |
|---|---|---|
| **Nomor** | 3 | |
| **Nama** | Tentang | |
| **Tujuan** | Menampilkan informasi aplikasi | |
| **Deskripsi** | Proses untuk menampilkan informasi aplikasi | |
| **Aktor** | Pemain | |
| **Skenario Utama** | | |
| **Kondisi awal** | Pemain berada di layar utama | |
| **Aksi Aktor** | | **Reaksi Sistem** |
| a. Memilih menu tentang | | |
| | | b. Menampilkan informasi aplikasi |
| **Kondisi Akhir** | Menampilkan informasi aplikasi | |

### 3.8.5 *Activity Diagram*

Diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak [2]. Berikut adalah *activity diagram* dari *prototype* permainan menjaring ikan:

**1. *Activity Diagram* Mulai**

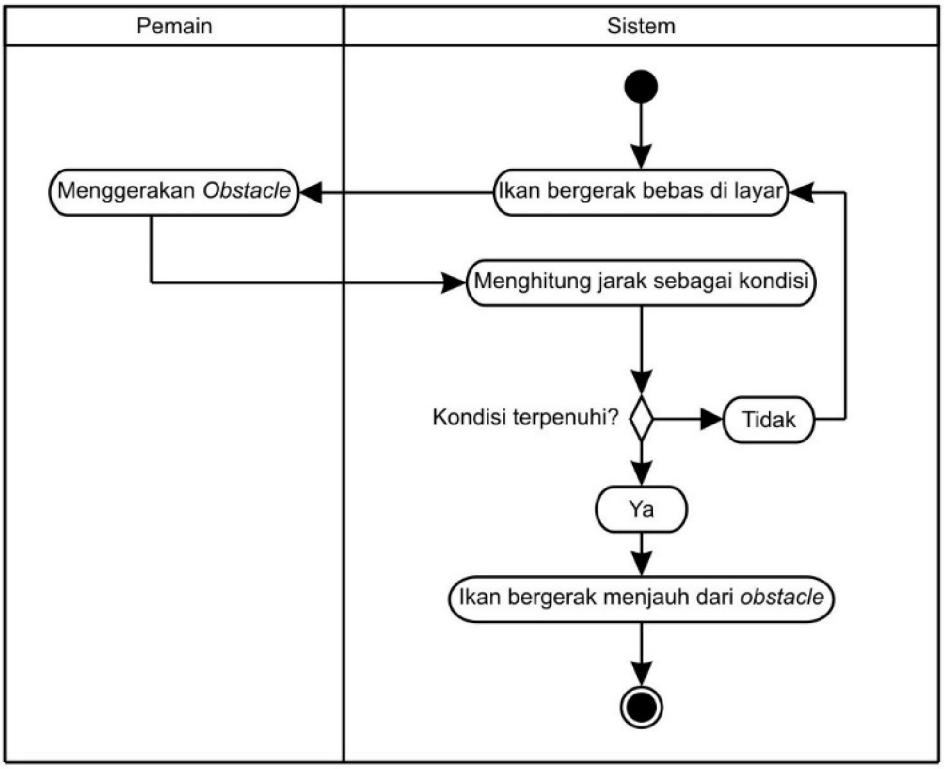
Pada proses ini sistem akan memulai simulasi pergerakan ikan. *Activity diagram* mulai dapat dilihat pada gambar 3.8.




Gambar 3.8 *Activity Diagram* Mulai

**2. *Activity Diagram* Menggerakan *Obstacle***

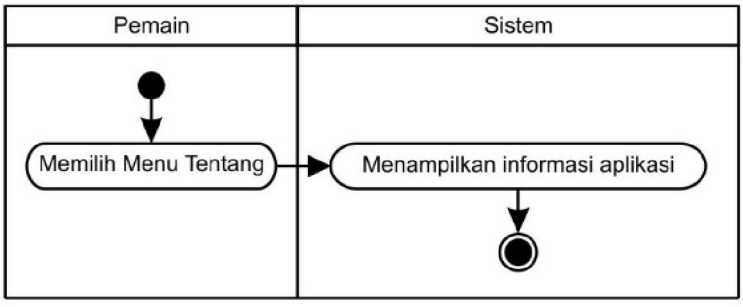
Berikut ini adalah diagram yang menunjukan alur aksi pada aktivitas menggerakan *obstacle* yang dapat dilihat pada gambar 3.9.




Gambar 3.9 *Activity Diagram* Menggerakan *Obstacle*

**3. *Activity Diagram* Keluar**

Berikut ini adalah diagram yang menunjukan alur aksi pada aktivitas tentang yang dapat dilihat pada gambar 3.10.

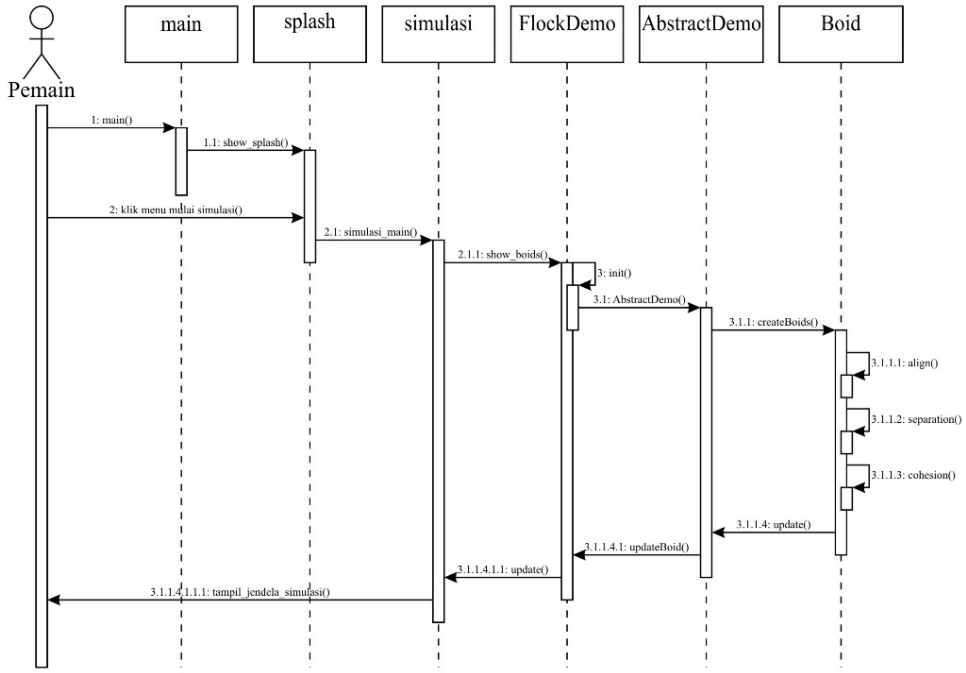



Gambar 3.10 *Activity Diagram* Keluar

### 3.8.6 *Sequence Diagram*

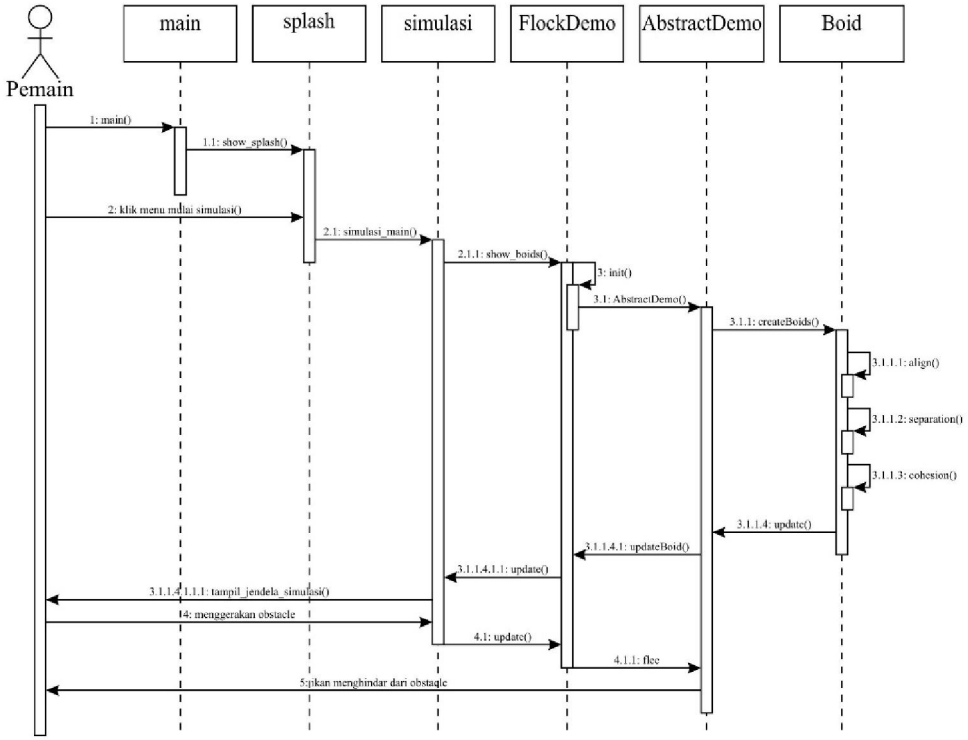
Diagram sekuen menggambarkan kelakuan objek pada use case dengan mendeskripsikan waktu hidup objek dan message yang dikirimkan dan diterima antar objek [2]. Berikut adalah *sequence diagram* dari *prototype* permainan menjaring ikan:

**1. *Sequence Diagram* Mulai**



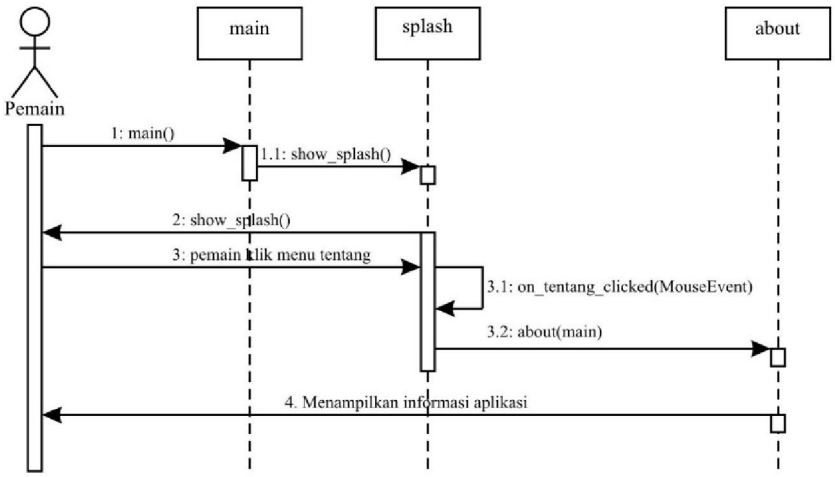

Gambar 3.11 *Sequence Diagram* Mulai

**2. *Sequence Diagram* Menggerakan *Obstacle***




Gambar 3.12 *Sequence Diagram* Menggerakan *Obstacle*

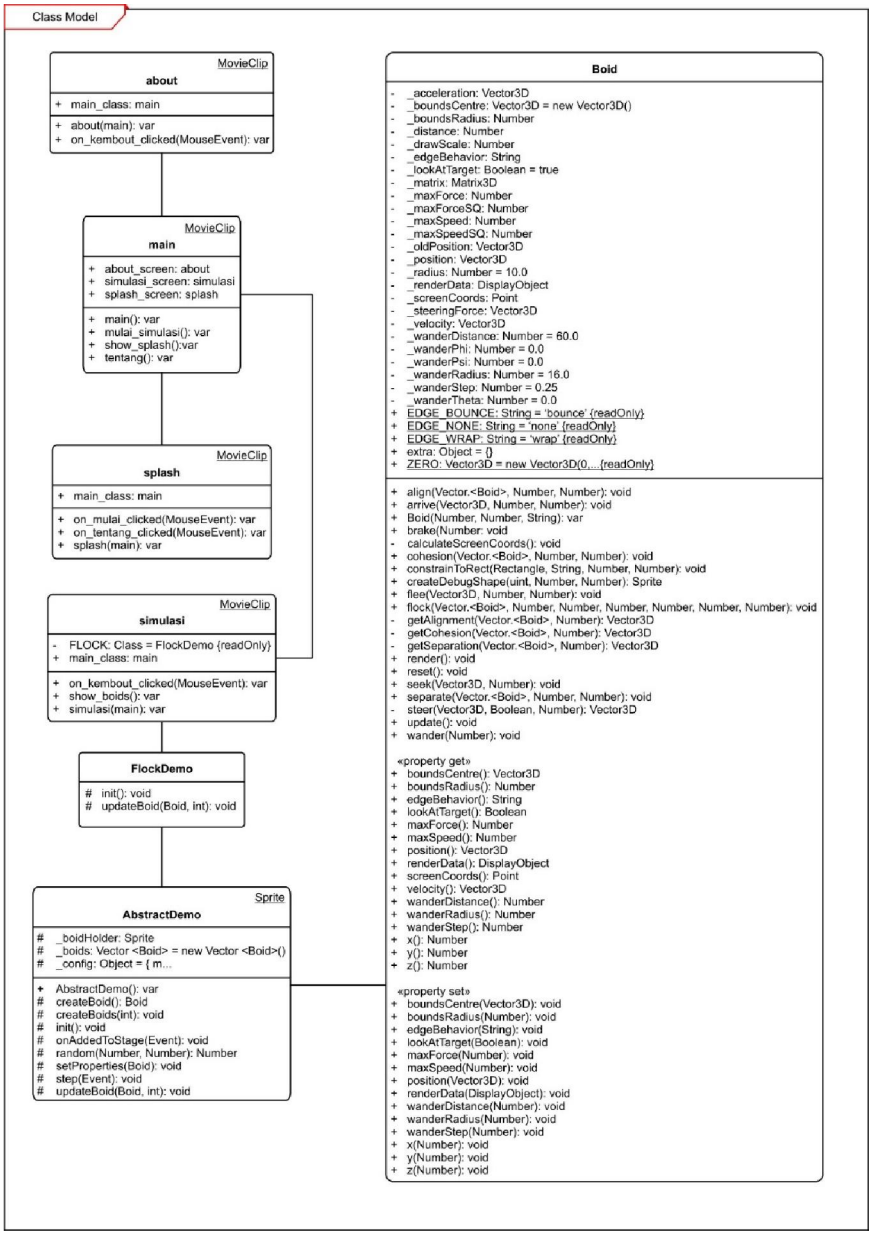
**3. *Sequence Diagram* Keluar**




Gambar 3.13 *Sequence Diagram* Keluar

### 3.8.7 *Class Diagram*

Diagram kelas atau 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. Berikut adalah *class diagram* dari *prototype* permainan menjaring ikan dapat dilihat pada gambar 3.14.




Gambar 3.14 *Class Diagram Prototype* Permainan Menjaring Ikan

## 3.9 Perancangan Sistem

Perancangan sistem merupakan kegiatan untuk merancang aplikasi yang akan dibangun, tahapannya dimulai dari perancangan struktur menu, perancangan antarmuka, perancangan pesan dan jaringan semantik dari permainan menjaring ikan.

### 3.9.1 Perancangan Kelas

Perancangan kelas merupakan hasil identifikasi dari kelas-kelas yang terdapat pada diagram kelas. Berikut perancangan kelas dari *prototype* permainan menjaring ikan dapat dilihat pada tabel 3.10.

Tabel 3.10 Perancangan kelas *prototype* permainan menjaring ikan

| No | Nama Kelas | Tipe Kelas | Deskripsi |
|---|---|---|---|
| **1** | main | *Control* | Kelas ini berfungsi untuk menangani kelas-kelas yang berfungsi membangkitkan halaman. |
| **2** | splash | *Boundary* | Kelas ini berfungsi untuk membangkitkan halaman menu utama sebagai jembatan interaksi antara pengguna dan aplikasi. |
| **3** | about | *Boundary* | Kelas ini berfungsi untuk membangkitkan halaman tentang. |
| **4** | simulasi | *Boundary* | Kelas in berfungsi untuk membangkitkan halaman simulasi. |
| **5** | FlockDemo | *Control* | Kelas ini berisi *method* yang berfungsi untuk menampilkan ikan dan *obstacle* ke halaman simulasi. |
| **6** | AbstractDemo | *Control* | Kelas ini berfungsi untuk menangai konfigurasi dari demo algoritma *boids*. |
| **7** | Boid | *Entity* | Kelas ini berisi *method* untuk menangani operasi dari algoritma *Boids*. |

### 3.9.2 Perancangan Stuktur Data

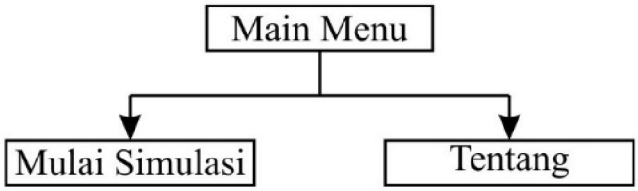
Perancangan struktur data merupakan hasil identifikasi dari variabel-variabel. Berikut perancangan stuktur data *entity boid* dari *prototype* permainan menjaring ikan dapat dilihat pada tabel 3.11.

Tabel 3.11 Perancangan Stuktur Data *Entity Boid*

| No | Variabel | Tipe Data | Keterangan |
|---|---|---|---|
| 1 | _acceleration | Vector3D | Variabel akselerasi dari ikan |
| 2 | _boundsCentre | Vector3D | Variabel titik pusat dari ikan |
| 3 | _boundsRadius | Number | Variabel jarak maksimum dari ikan untuk dapat bergerak dari variabel _boundsCentre |
| 4 | _distance | Number | Variabel jarak di sekitar ikan dimana ikan akan menghitung rotasi untuk bergerak |
| 5 | _drawScale | Number | Variabel untuk menciptakan grafis sederhana untuk mewakili *boids* |
| 6 | _edgeBehavior | String | Variabel untuk perilaku yang akan mengubah kemudi |
| 7 | _lookAtTarget | Boolean | Variabel untuk mengatur arah ikan untuk bergerak |
| 8 | _matrix | Matrix3D | Variabel yang digunakan untuk me*render* data ikan |
| 9 | _maxForce | Number | Variabel dari gaya maksimum ikan ketika menghitung kekuatan kemudi |
| 10 | _maxForceSQ | Number | Variabel dari nilai _maxForce dikali nilai _maxForce |
| 11 | _maxSpeed | Number | Variabel kecepatan maksimum dari ikan |
| 12 | _maxSpeedSQ | Number | Variabel dari nilai _maxSpeed dikali nilai _maxSpeed |
| 13 | _oldPosition | Vector3D | Variabel posisi dari ikan |
| 14 | _position | Vector3D | Variabel posisi dari ikan |
| 16 | _renderData | DisplayObject | Variabel yang digunakan untuk membuat ikan |
| 17 | _screenCoords | Point | Variabel dari sebuah objek yang mewakili posisi x dan y ikan pada layar |
| 18 | _steeringForce | Vector3D | Variabel pengaturan perilaku dari ikan |
| 19 | _velocity | Vector3D | Variabel kecepatan dari ikan saat ini |
| 20 | _wanderDistance | Number | Variabel jarak dari ikan untuk menghitung rotasi |
| 21 | _wanderPhi | Number | Variabel yang menghasilkan untuk ikan dapat bergerak secara bebas |
| 22 | _wanderPsi | Number | Variabel yang menghasilkan untuk ikan dapat bergerak secara bebas |
| 23 | _wanderRadius | Number | Variabel dari jari-jari lingkaran yang digunakan untuk menghitung perilaku dari ikan |
| 24 | _wanderStep | Number | Variabel dari ikan yang dapat berubah pada setiap langkah |
| 25 | _wanderTheta | Number | Variabel yang menghasilkan untuk ikan dapat bergerak secara bebas |

### 3.9.3 Perancangan Struktur Menu

Perancangan struktur menu dirancang untuk menyediakan fungsi-fungsi yang akan digunakan dalam permainan menjaring ikan dapat dilihat pada gambar 3.15.

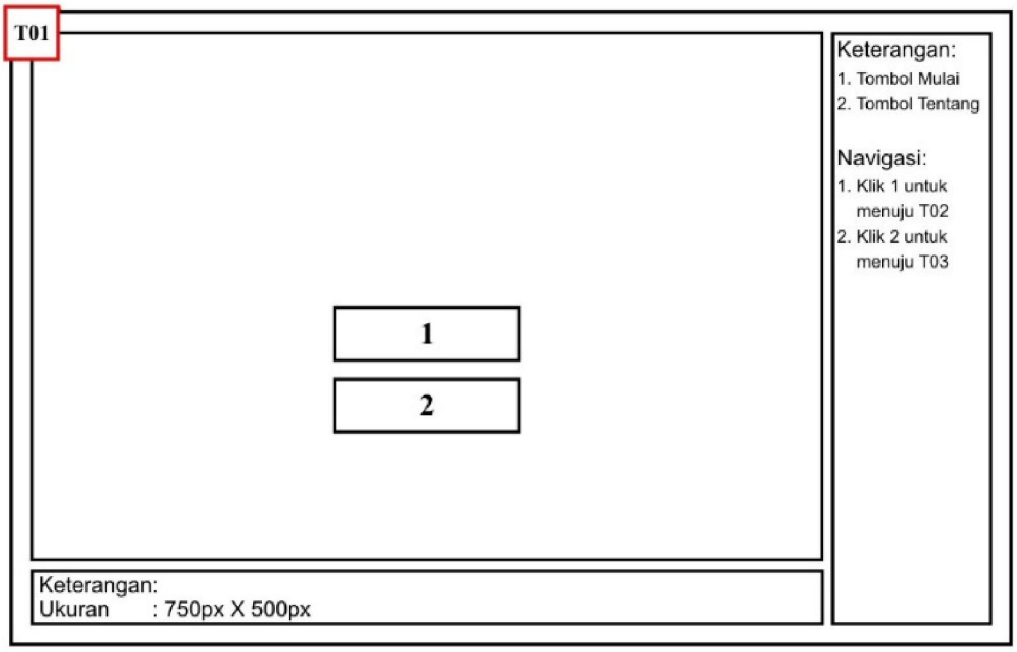



Gambar 3.15 Rancangan Struktur Menu

### 3.9.4 Perancangan Antarmuka

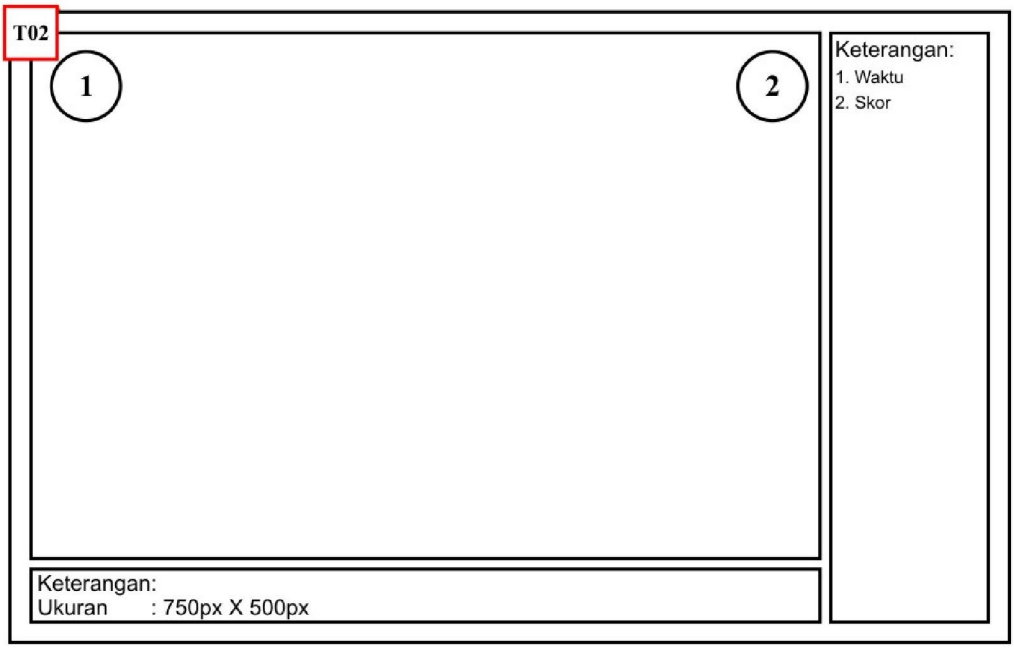
Perancangan antarmuka bertujuan untuk memberikan gambaran tentang aplikasi yang akan dibangun, sehingga akan mempermudah dalam mengimplementasikan aplikasi serta akan memudahkan pembuatan aplikasi.

1. Perancangan antarmuka menu



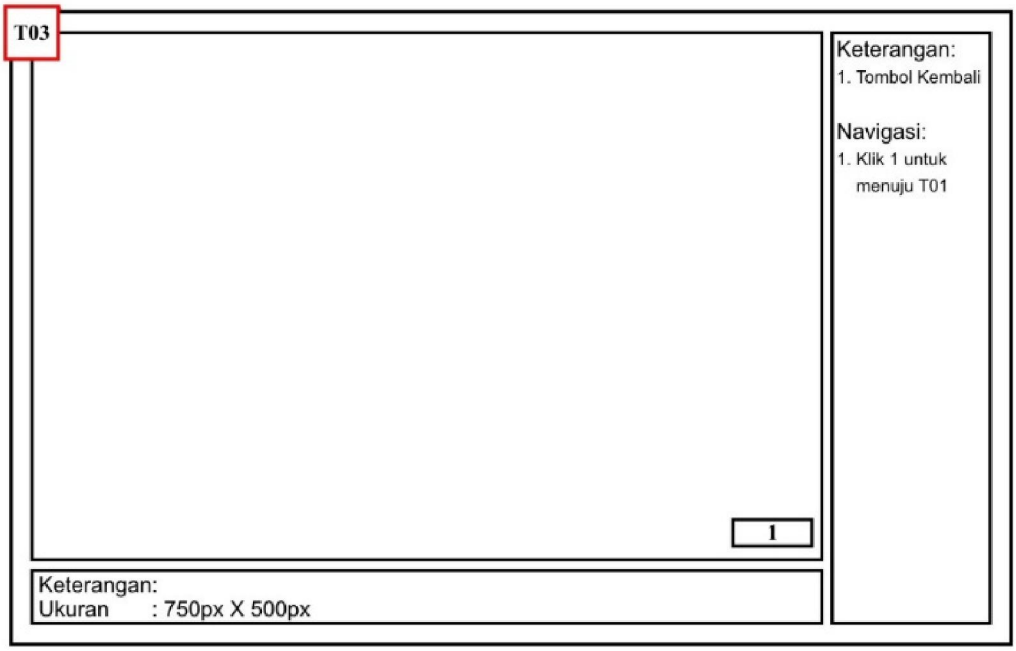

Gambar 3.16 Rancangan antarmuka menu

2. Perancangan antarmuka simulasi




Gambar 3.17 Rancangan antarmuka simulasi

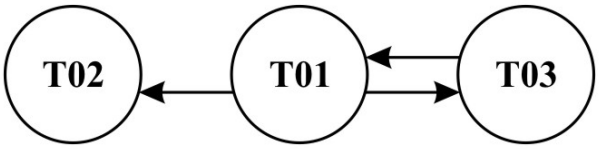
3. Perancangan antarmuka tentang




Gambar 3.18 Rancangan antarmuka tentang

### 3.9.5 Jaringan Semantik

Untuk mengetahui keterhubungan dari setiap tampilan yang dibuat dapat menggunakan jaringan semantik. Berikut adalah jaringan semantik pada *prototype* permainan menjaring ikan dapat dilihat pada gambar 3.19.




Gambar 3.19 Jaringan Semantik