BAB 3

ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Sistem

Analisis sistem dapat didefinisikan sebagai penguraian dari suatu sistem informasi yang utuh ke dalam bagian-bagian komponennya untuk mengidentifikasikan masalah-masalah, dan hambatan-hambatan sehingga dapat diusulkan kebutuhan-kebutuhan untuk perbaikannya.

3.1.1 Analisis Masalah

Salah satu game dengan *genre casual game* adalah *flap flap bird* yang dibuat oleh *sprakelsoft*. Pada game ini pemain akan bermain sebagai seekor burung, burung tersebut akan terus terbang dan tugas pemain disini adalah untuk menghindar rintangan yang akan terus ada selama perjalanan. Pemain akan mati jika menabrak rintangan. Semakin lama pemain bisa menghindari rintangan semakin besar pula skor yang didapat.

Berdasarkan hasil review dan survey orang yang memainkan game *Flip flap bird* tersebut, terdapat beberapa kelemahan antara lain variasi rintangan yang sedikit yakni berupa balok saja, tidak adanya musuh, serta tidak ada sistem level sehingga tingkat kesulitannya sama dan terkesan monoton.

3.1.2 Analisis Game Sejenis

Analisis *game* sejenis dilakukan untuk memberikan gambaran sistem agar lebih mudah dalam pembangunan dan sebagai pembanding untuk kesempurnaan *game* yang akan dicapai.

3.1.2.1. Game *Flap flap bird*

Game *flap flap bird* Adalah salah satu game yang memiliki *genre casual game***.**

a. Gameplay *Flap flap bird*

Dalam game *flap flap bird* Ini pemain akan bermain sebagai seekor burung, burung itu akan terus terbang, disini pemain harus menghidari rintangan berupa balok. Jika pemain menabrak rintangan tersebut, burung akan mati dan permainan berakhir, selama burung belum menabrak rintangan maka game ini tidak akan pernah berakhir, semakin lama burung terbang maka semakin besar pula skor yang didapat pemain.



Gambar 3.1 Gameplay *Flap flap bird*

b. Skenario Game *Flap flap bird*

Pada saat aplikasi dimulai, pemain akan dibawa ke layar menu



Gambar 3.2 Menu *Flap flap bird*

Pada menu ini pemain dapat menekan 4 buah tombol yaitu *start game*, gambar note musik, gambar speaker, dan *other games*. Jika pemain menekan tombol *start game* maka permainan akan dimulai, jika pemain menekan gambar note musik maka lagu akan berhenti, jika pemain menekan tombol *other games* maka akan muncul layar pilihan beberapa game dari developer yang sama. Jika pemain gagal menghindari rintangan maka akan muncul layat *game over* yang menandakan permainan berakhir



Gambar 3.3 layar *game over Flap flap bird*

3.1.3 Perbandingan Game *Flap Flap Bird* dan Game *Fly Eagle*

Perbandingan antara game *flap flap bird* dan game *fly eagle* dapat dilihat di dalam tabel 3.1 berikut:

Tabel 3.1 Perbandingan Game *Flap flap bird* dan *Fly Eagle*

Keterangan	**Game *Flap Flap Bird***	**Game *Fly Eagle***
Storyline	Tidak Ada	Ada
Sistem level	Tidak Ada	Ada
Background music	Ada	Ada
Musuh	Tidak Ada	Ada

Kecerdasan Buatan	Tidak Ada	A*
Variasi rintangan	balok	17 rintangan
Menu petunjuk	Tidak Ada	Ada
Karakter utama	Burung	Elang

3.1.4 Analisis Game *Fly Eagle*

Game yang akan dibuat adalah *Fly Eagle* yang bergenre *Arcade, adventure*. *Gameplay Fly Eagle* mirip dengan *gamplay flap flap bird*

Game ini bergrafis 2D yang cara permainannya sediri masih menggunakan sistem single *Player*. AI (*Artificial Intelligence*) yang digunakan untuk musuh dalam *Game* ini adalah algoritma *A**

3.1.4.1. *Storyline*

Ceritanya elang ini terbang terlalu jauh dari hutan dan tersesat di kota, di tengah kesesatannya elang tersebut ditangkap. Dimulailah petualangan sang elang yang kabur dan akan kembali ke hutan, Sebelum sampai di hutan sang elang harus menghadapi beberapa rintangan di beberapa tempat seperti di dalam rumah, jalanan kota, bahkan setelah sampai di hutanpun ada rintangan yang menghadang.

3.1.4.2. Gameplay

Game ini dimainkan secara *single player* (perseorangan), pemain akan memainkan karakter seekor *eagle* atau elang. Pada awal permainan, elang akan terus bergerak dari kanan ke kiri, elang akan melompat atau terbang jika pemain mengklik tombol kiri *mouse*, akan ada musuh dan rintangan yang menghalangi elang. Pemain akan kalah jika elang menabrak musuh atau rintangan, ketika pemain kalah maka pemain akan diberi 2 pilihan yaitu mengulang permainan atau keluar, jika pemain memilih mengulang, maka pemain akan mengulangi permainan dari level yang

terakhir dimainkannya, pemain menyelesaikan permainan jika elang sudah sampai di bulatan berwarna biru.

3.1.4.3 Analisis Level

Saat memainkan game, pemain harus menyelesaikan beberapa level diantaranya :

1. Level 1
    1. Memiliki latar rumah
    2. Pemain memainkan karakter elang
    3. Pemain harus menghindari musuh dan rintangan
    4. Pemain dapat mengumpulkan daging, 1 daging berisi 1 poin untuk *score*
    5. Pemain juga dapat mengambil *itemspeed* untuk menambah kecepatan elang
    6. Pemain akan mati (kalah) jika bertabrakan dengan rintangan atau musuh
    7. Pemain akan menuju ke level berikutnya ketika sudah sampai di tujuan yang ditandakan dengan lingkaran berwarna biru
    8. Setelah sampai, poin yang dikumpulkan pemain akan ditambahkan ke dalam total *score* yang berisi akumulai poin dari level 1-3
2. Level 2
    1. Memiliki latar kota
    2. Pemain memainkan karakter elang
    3. Pemain harus menghindari musuh dan rintangan
    4. Pemain dapat mengumpulkan daging, 1 daging berisi 1 poin untuk *score*
    5. Pemain juga dapat mengambil *itemspeed* untuk menambah kecepatan elang
    6. Pemain akan mati (kalah) jika bertabrakan dengan rintangan atau msuh
    7. Pemain akan menuju ke level berikutnya ketika sudah sampai di tujuan yang ditandakan dengan lingkaran berwarna biru

8. Setelah sampai, poin yang dikumpulkan pemain akan ditambahkan ke dalam total *score* yang berisi akumulai poin dari level 1-3
9. Di level ini setiap kali pemain mati atau kalah, ketika pemain menekan tombol bermain lagi maka *total score* pemain akan dikurangi 10
10. Kecepatan elang lebih cepat 25% dari level 1
11. Musuh berjumlah 2 dan salah satu dari musuh bisa terbang.

3. Level 3
    1. Memiliki latar hutan
    2. Pemain memainkan karakter elang
    3. Pemain harus menghindari musuh dan rintangan
    4. Pemain dapat mengumpulkan daging, 1 daging berisi 1 poin untuk *score*
    5. Pemain juga dapat mengambil *itemspeed* untuk menambah kecepatan elang
    6. Pemain akan mati (kalah) jika bertabrakan dengan rintangan atau msuh
    7. Pemain akan menyelesaikan game ketika sudah sampai di tujuan yang ditandakan dengan lingkaran berwarna biru
    8. Setelah sampai, poin yang dikumpulkan pemain akan ditambahkan ke dalam total *score* yang berisi akumulai poin dari level 1-3
    9. Di level ini juga, setiap kali pemain mati atau kalah, ketika pemain menekan tombol bermain lagi maka *total score* pemain akan dikurangi 10
    10. Kecepatan elang lebih cepat 50% dari level 1
    11. Musuh berjumlah 3, 2 diantaranya adalah burung yang bisa terbang.

3.1.4.4 Analisis Tingkat kesulitan

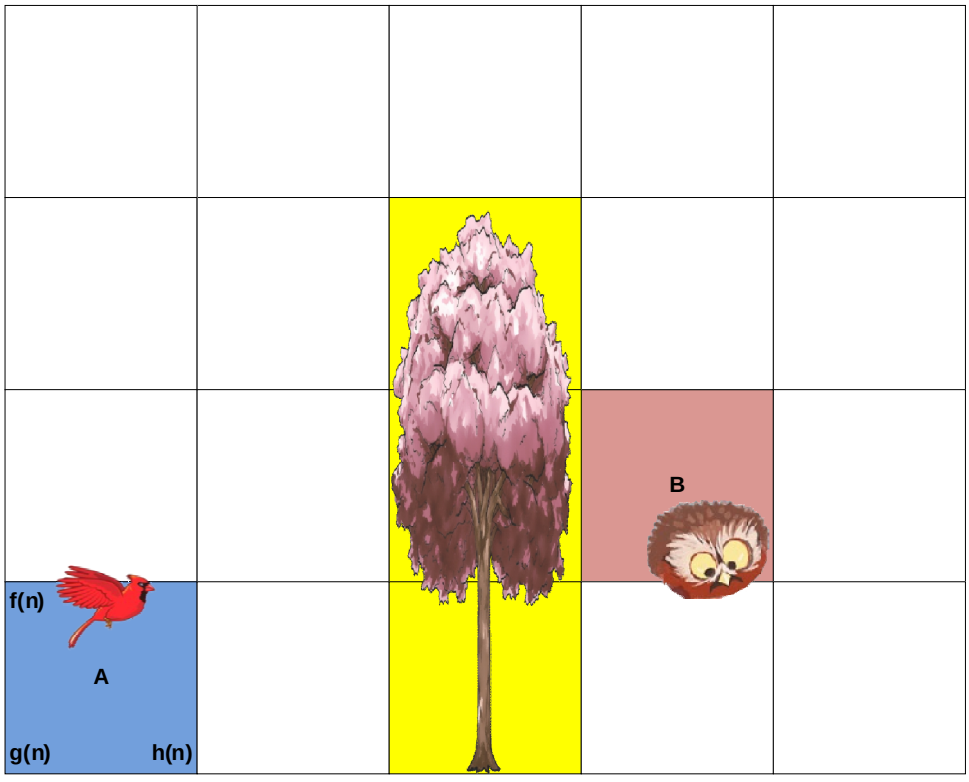
Tingkat kesulitan yang disediakan dalam game *fly eagle* terdiri dari tiga tingkat yang berdasarkan level permainan, yaitu level 1, level 2, dan level 3. Perbedaan kesulitan dalam game dapat dilihat pada table 3.2

Tabel 3.2 Perbandingan Tingkat kesulitan *Fly Eagle*

Perbedaan	Tingkat Kesulitan		
	Level 1	Level 2	Level 3
Elang	Kecepatan standar	Kecepatan meningkat 25% dari level 1	Kecepatan meningkat 50% dari level 1
Musuh	Berjumlah 1	Berjumlah 2	Berjumlah 3

3.1.5 Analisis Algoritma A*

Algoritma A* di dalam *game* akan diterapkan pada musuh sebagai penentuan jalur terpendek atau *pathfinding* menuju target yaitu, dalam hal ini *node* A merupakan Musuh dan *node* B adalah target tujuan atau player. Cara kerja algoritma A* di dalam *game* lebih jelasnya bisa dilihat pada gambar 3.4.



Gambar 3.4 Tahap Awal Pencarian Algoritma A*

Pada gambar 3.4 pencarian wilayah dibagi ke dalam *node-node* yang terdiri dari *node* berwarna biru merupakan *node* awal (*node* A atau musuh), *node* berwarna merah merupakan *node* akhir (*node* B atau *player*) dan yang berwarna kuning merupakan penghalang (*unwalkable node)*.

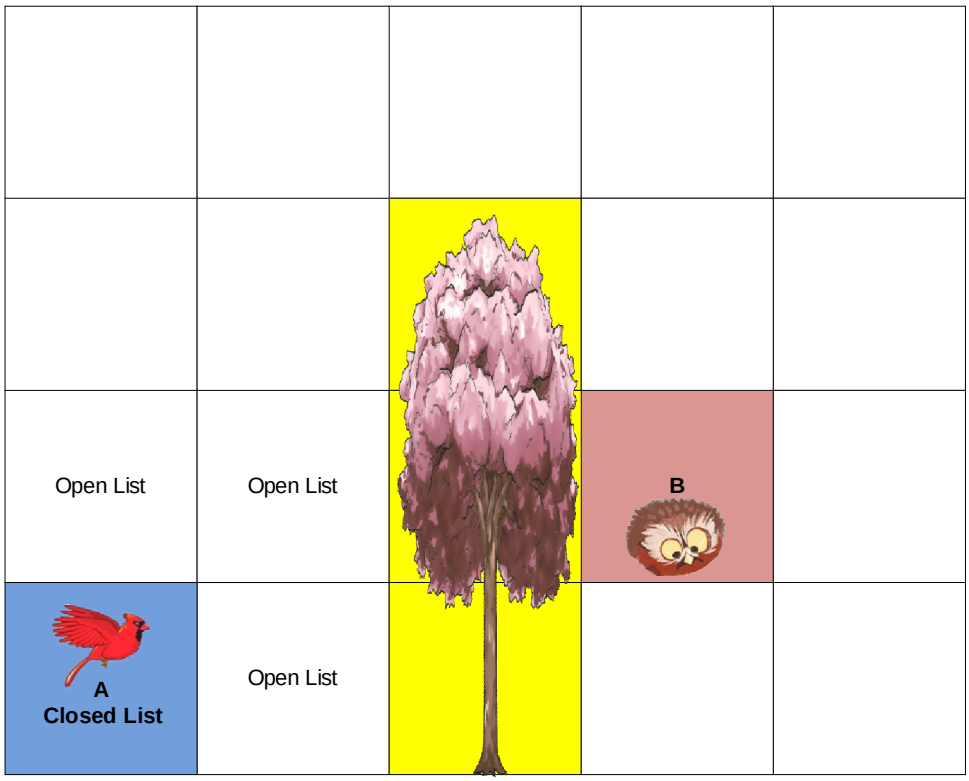
Selanjutnya adalah, melakukan pencarian untuk menemukan jalan terpendek dengan cara memeriksa *node-node* yang berdekatan dengan *node* awal untuk bisa sampai *node* B. Berikut adalah langkah pencarian yang dilakukan oleh *node* awal:

1. Langkah pertama, dimulai dari *node* awal yang ditambahkan ke dalam *Open list* yang merupakan *list* untuk *node-node* yang harus diperiksa.

2. Periksa semua *node* yang berdekatan dengan *node* awal apakah bisa dilalui atau tidak, *node* penghalang yang berwarna kuning diabaikan lalu tambahkan semua *node* tersebut ke dalam *open list*, untuk setiap *node* yang baru ditambahkan, simpan *node* awal sebagai "*parent*" yang nantinya akan digunakan untuk menelusuri jalan.

3. Hapus *node* awal dari *open list*, dan tambahkan ke dalam *closed list. Closed list* merupakan *list* untuk *node* yang tidak perlu diperiksa atau dalam artian merupakan bagian dari jalur terpendek yang sudah didapatkan.

Untuk lebih jelasnya *open list* dan *closed list* diilustrasikan pada gambar 3.5

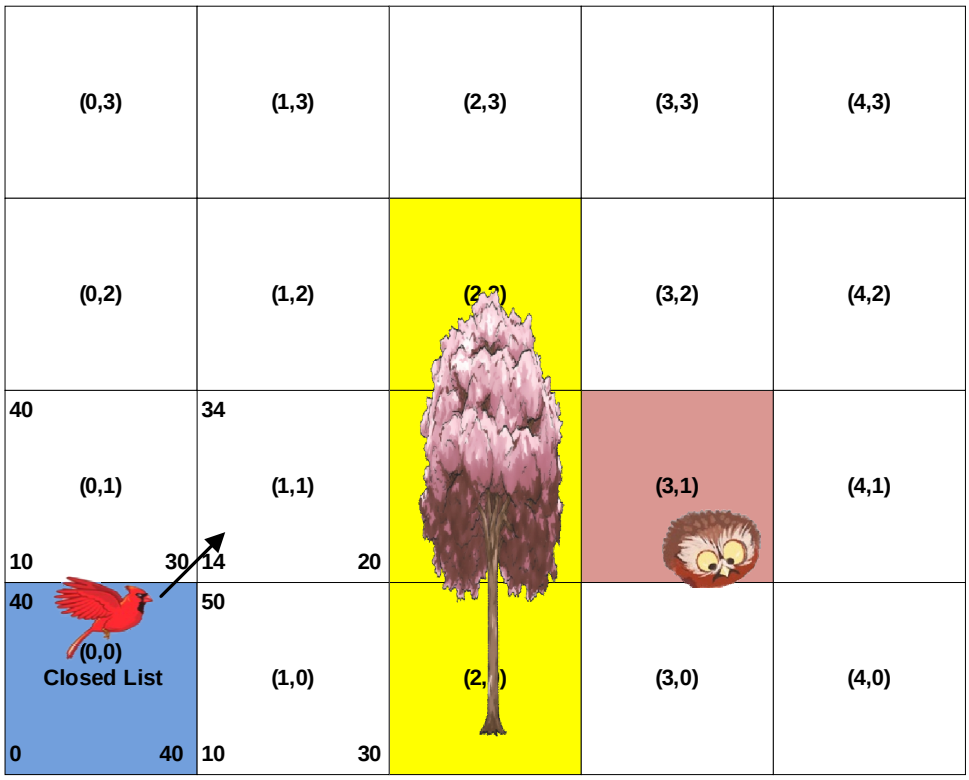


Gambar 3.5 Ilustrasi Open List dan Closed List

Pada gambar 3.5 diatas, *node* awal yang berwarna biru sudah ditambahkan ke dalam *closed list.* Sedangkan kotak yang berada di sekelilingnya adalah *open list* yang harus diperiksa satu persatu nilai dari masing-masing *node open list.*

Langkah berikutnya adalah memilih *node* yang memiliki nilai f terendah di dalam *open list* dengan menggunakan rumus (f = g + h). g adalah nilai yang dikeluarkan dari *node* A ke *node* yang ada disekitarnya. Sedangkan h adalah fungsi *heuristic* yang digunakan untuk menghitung perkiraan biaya yang dikeluarkan dari *node* awal menuju *node* tujuan.

Proses ini dilakukan secara berulang-ulang, dimulai dari *open list* dan menemukan *node* dengan nilai f paling rendah. Perhitungan persamaan untuk mendapatkan nilai f dijelaskan pada ilustrasi gambar 3.6.



Gambar 3.6 Perhitungan menentukan nilai F

Gambar 3.6 menunjukkan pada *open list* dari *node* awal yaitu NPC nilai f yang dihasilkan adalah 34 yang didapatkan dari menambahkan nilai g = 14 dan nilai h = 20. Di dalam contoh ini g akan diberi nilai 10 apabila *node* bergerak secara *vertical* atau *horizontal* dan diberi nilai 14 apabila *node* bergerak secara *diagonal*.

Perhitungan nilai h yaitu nilai estimasi jalur terpendek dari *node* awal ke *node* tujuan yaitu *player* dengan menggunakan rumus h = ((Xtujuan – Xstart) + (Ytujuan – Ystart) * 10). Sebagai contoh *node* yang terletak dibawah dari *node* awal diberikan nilai h = 30 yang didapatkan dari perhitungan sebagai berikut:

h = ((Xtujuan – Xstart) + (Ytujuan – Ystart) * 10)

= ((3 – 0) + (1 - 1) *10

= 3 * 10 = 30

Setelah didapatkan nilai f dari setiap *node* yang ada di *open list*, maka akan dipilih *node* yang memiliki nilai f paling rendah yang kemudian disebut dengan *current node*, langkah selanjutnya adalah sebagai berikut.

1. Hapus *current node* dari *open list* dan masukkan ke dalam *closed list.*

2. Periksa semua *node* yang berdekatan dengan *current node*, abaikan *node* penghalang. Jika tidak ada pada *open list,* tambahkan ke dalam *open list,* lalu tambahkan "*parent*" untuk *node* tersebut.

3. Jika *node* sudah ada pada *open list,* lalu bandingkan dengan *node* awal apakah jalur yang di ambil lebih baik dari *node* awal, dengan dengan memeriksa nilai G-nya apakah lebih rendah jika dalam menggunakan *node* ini, jika lebih rendah maka hitung ulang nilai f dan g serta merubah arah *pointer,* jika tidak maka jangan lakukan apa-apa.

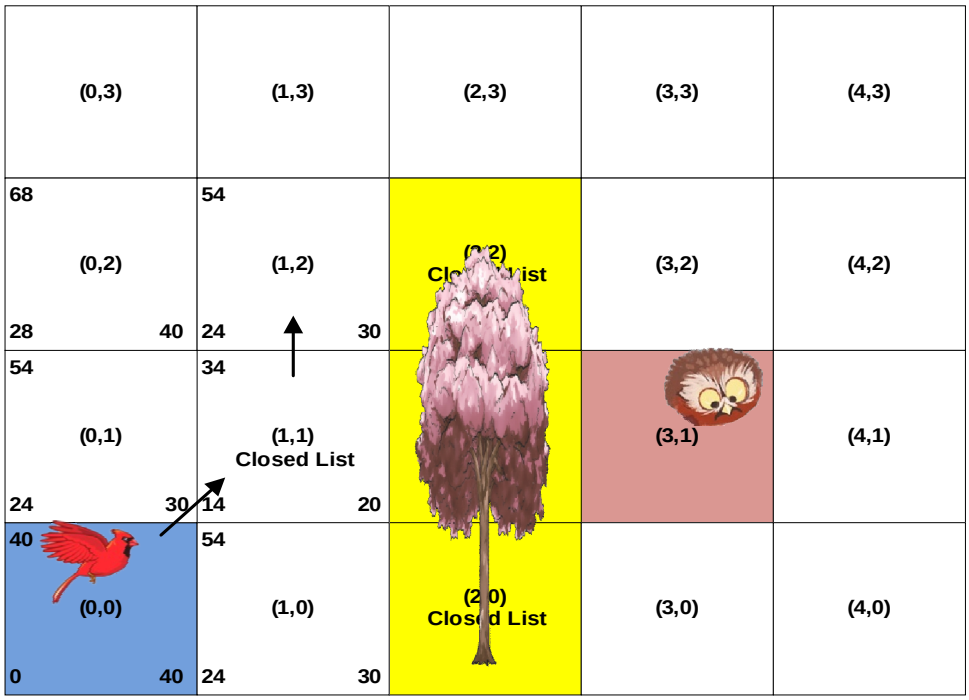
Langkah pertama menghasilkan *node-node* di dalam *open list* dan *closed list* dijelaskan pada tabel 3.3

Tabel 3.3 Perhitungan Langkah Pertama

Open List	(0,1) (1,1) (1,0)
Close List	(0,0)
Parent	(0,0)
Current Node	(1,1)

Gambar 3.6 merupakan penjelasan langkah pertama pencarian jalur dari algoritma *A**.

Perhitungan pada langkah kedua yang berada pada *node* dengan koordinat (1,1) dapat dilihat pada gambar 3.7



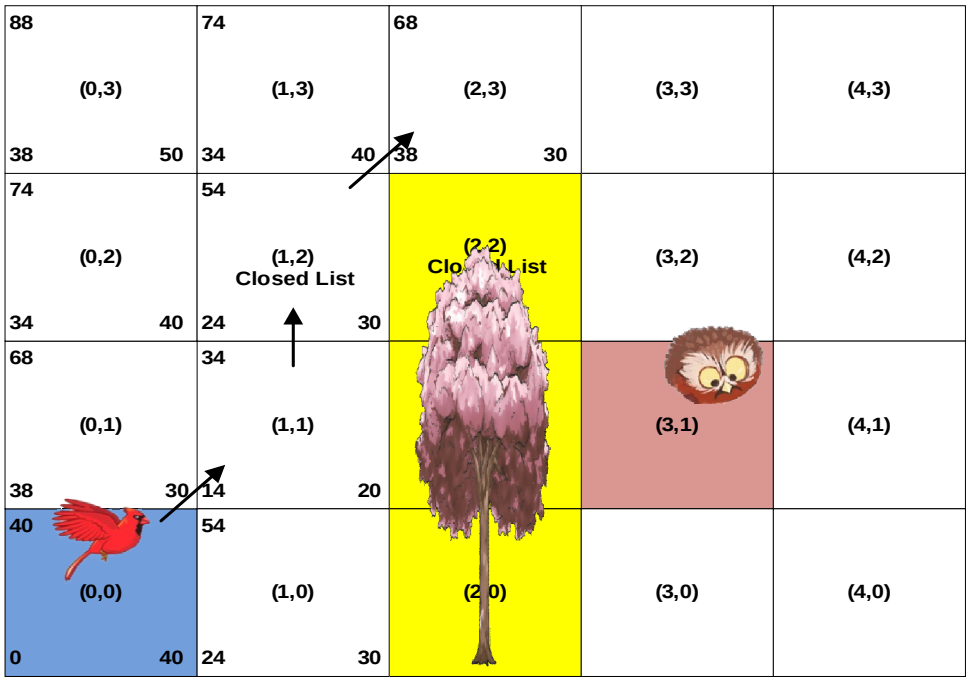
Gambar 3.7 Penjelasan A* Pada Langkah Kedua

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

Tabel 3.4 Perhitungan Langkah Kedua

Open List	(0,2) (1,2) (0,1) (1,0)
Close List	(0,0) (1,1) (2,0) (2,1) (2,2)
Parent	(1,1)
Current Node	(1,2)

Perhitungan langkah ketiga yang berada pada *node* dengan koordinat (1,2) dapat dilihat pada gambar 3.8.



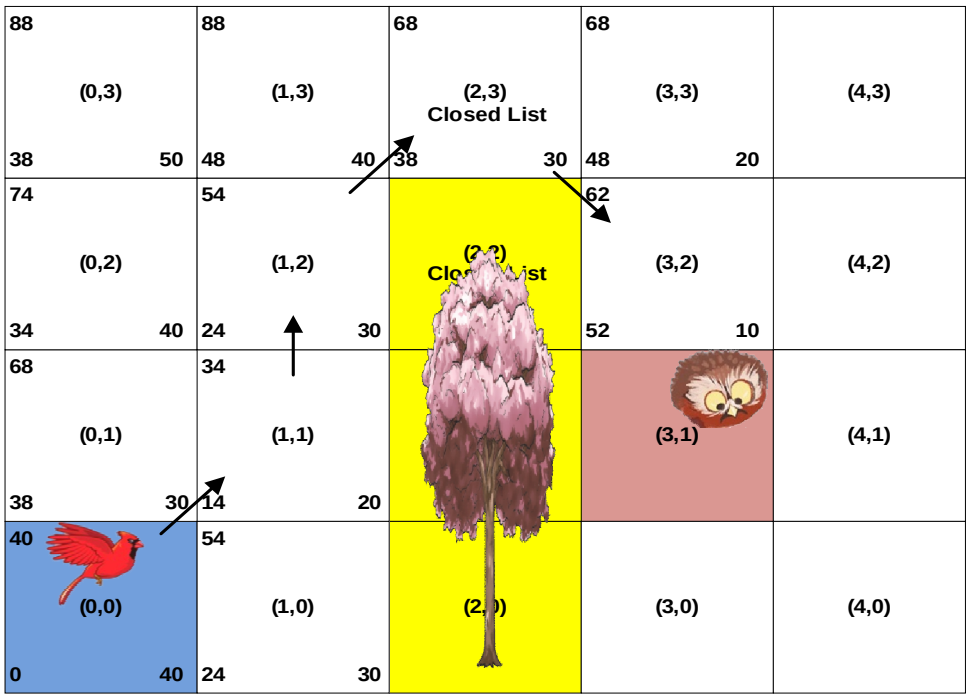
Gambar 3.8 Penjelasan A* Pada Langkah Ketiga

Tabel 3.5 menunjukan penjelasan tentang pencarian jalur terpendek untuk langkah ketiga dapat dilihat pada tabel 3.5.

Tabel 3.5 Perhitungan Langkah Ketiga

Open List	(0,2) (0,1) (0,3) (1,3) (2,3)
Close List	(1,2) (2,2) (1,1) (2,1)
Parent	(1,2)
Current Node	(2,3)

Perhitungan langkah keempat yang berada pada *node* dengan kooedinat (2,3) dapat dilihat pada gambar 3.9.



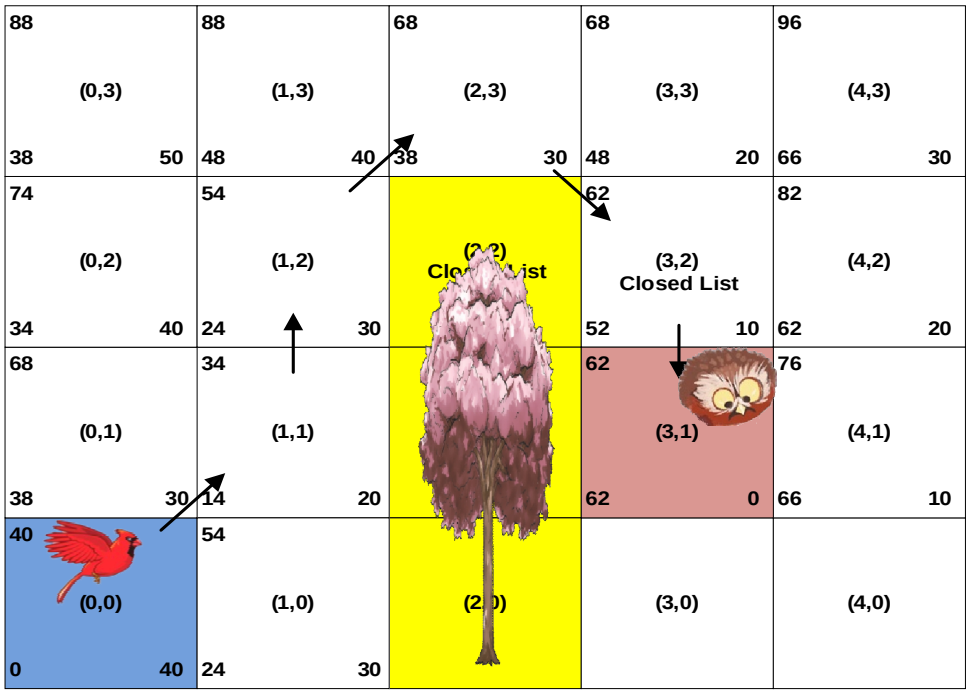
Gambar 3.9 Penjelasan A* Pada Langkah Keempat

Tabel 3.6 menunjukan penjelasan tentang pencarian jalur terpendek untuk langkah keempat.

Tabel 3.6 Perhitungan Langkah Keempat

Open List	(1,3) (3,2) (3,3)
Close List	(1,2) (2,2)(2,3)
Parent	(2,3)
Current Node	(3,2)

Perhitungan pada langkah kelima yang berada pada *node* dengan koordinat (3,2) yang memiliki nilai f = 68 bisa dilihat pada gambar 3.10.



Gambar 3.10 Penjelasan A* Pada Langkah Kelima

Pada tabel 3.7 penjelasan melalui tabel untuk pencarian jalur terpendek pada langkah kelima.

Tabel 3.7 Perhitungan Langkah Kelima

Open List	(3,3) (4,3) (4,2) (4,1) (3,1)
Close List	(2,1) (2,2) (2,3) (3,2)
Parent	(3,2)
Current Node	(3,1)

3.1.6 Analisis Kebutuhan Non-Fungsional

Analisis kebutuhan *non*-fungsional menggambarkan kebutuhan sistem yang menitik beratkan pada properti prilaku yang dimiliki oleh sistem, diantaranya kebutuhan perangkat keras, perangkat lunak, serta user sebagai bahan analisis

kekurangan dan kebutuhan yang harus dipenuhi dalam perancangan sistem yang akan diterapkan.

3.1.6.1 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 system di dalamnya.

Pada game *Fly Eagle* ini, spesifikasi standar perangkat keras yang dibutuhkan adalah sebagai berikut :

1. CPU dengan kecepatan minimal 1.6 GHz
2. RAM 1 GB
3. VGA 512 MB
4. *HardDisk* 32 GB
5. Monitor standar
6. *Mouse, keyboard,* dan *speaker*

Sedangkan perangkat keras minimum yang dibutuhkan oleh pemain adalah :

1. *Processor* dengan kecepatan 1,6 GHz
2. RAM 512 Gb
3. *HardDisk* 8 GB
4. VGA *Card* 256 MB
5. Monitor
6. *Mouse, keyboard, speaker*

3.1.6.2 Analisis Kebutuhan Perangkat Lunak

Perangkat lunak digunakan dalam sebuah sistem, merupakan perintah-perintah yang diberikan kepada perangkat keras agar bisa berinteraksi diantara keduanya. Adapun perangkat lunak yang dibutuhkan dalam membangun aplikasi game. Adapun

perangkat lunak yang dibutuhkan dalam membangun aplikasi game *Fly Eagle* ini adalah sebagai berikut :

1. Sistem Operasi Windows 7
2. *Construct 2*
3. *Adobe Photosho CS6*

Sedangkan perangkat lunak yang dibutuhkan pengguna untuk menajalankan aplikasi ini adalah sistem operasi Windonws XP ke atas.

3.1.6.3 Analisis Pengguna

Analisis pengguna digunakan untuk mengetahui spesifikasi pengguna untuk dapat memainkan aplikasi permainan *Fly Eagle*, seperti pada tabel 3.8

Tabel 3.8 Analisis Pengguna

Ketentuan	Spesifikasi
Peran	Pemain
Usia	6 tahun ke atas
Kondisi Fisik	Tidak buta, memiliki gangguan fisik yang tidak memungkinkan pengguna untuk mengoperasikan komputer
Pengalaman	Pernah memainkan permainan yang sejenis maupun game dengan jenis permainan yang berbeda pada komputer

3.1.7 Analisis Kebutuhan Fungsional

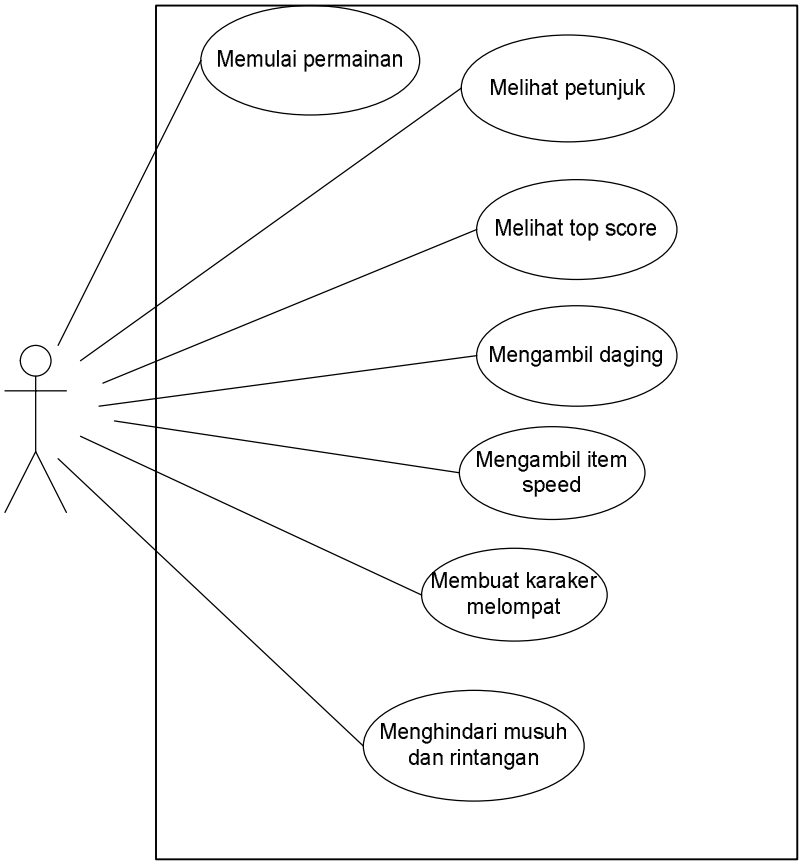
Analisis kebutuhan fungsional menggambarkan proses kegiatan yang akan diterapkan dalam sebuah sistem dan menjelaskan kebutuhan yang diperlukan sistem agar sistem dapat berjalan dengan baik serta sesuai dengan kebutuhan.

3.1.7.1. Identifikasi *Actor*

Tahap Pertama yang dilakukan dalam melakukan analisis berorientasi objek menggunakan UML adalah menentukan aktor atau pengguna sistem. Kata aktor dalam kontek UML, menampilkan peran (*roles*) yang pengguna (sesuatu diluar sistem yang dikembangkan yang dapat berupa perangkat keras, *end user*, sistem yang lain dan sebagainya).

3.1.7.2. Diagram *Use Case*

Diagram Use case merupakan konstruksi untuk mendeskripsikan hubungan-hubungan yang terjadi antar aktor dengan aktivitas yang terdapat pada sistem. Sasaran pemodelan *usecase* diantaranya adalah mendefinisikan kebutuhan fungsional dan operasional sistem dengan mendefinisikan skenario penggunaan sistem yang akan dibangun. Dari hasil analisis aplikasi yang ada maka diagram *use case* untuk aplikasi *game fly eagle* dapat dilihat pada gambar 3.11.



Gambar 3.11 Diagram *Use Case* aplikasi *Fly Eagle*

3.1.7.3 Definisi *Actor*

Definisi *Actor* berfungsi untuk menjelaskan *Actor* yang terdapat pada *Diagram*. Definisi *Actor* diterangkan pada Tabel 3.9.

Tabel 3.9 Definisi *Actor*

Actor	**Deskripsi**
Pemain	Orang yang memainkan game

3.1.7.4 Definisi *Use Case*

Definisi *Use case* berfungsi untuk menjelaskan proses yang terdapat pada setiap *Use case*. Definisinya dapat dilihat pada tabel 3.10.

Tabel 3.10 Definisi *Use case*

No	Use Case	Deskripsi
1	Memulai permainan	Proses memulai permainan dalam game
2	Melihat Petunjuk	Proses melihat petunjuk permainan dalam game
3	Melihat *Top Score*	Proses melihat *top score* dalam game
4	Mengambil daging	Proses mengambil daging
5	Mengambil item speed	Proses mengambil item speed
6	Membuat karakter melompat	Proses mebuat elang melompat
7	Menghindari musuh dan rintangan	Proses menghindari musuh dan rintangan

3.1.7.5 Skenario *Use Case*

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

Adapun skenario *use case* yang terdapat pada aplikasi permainan *Fly Eagle* ini adalah sebagai berikut :

1. Skenario *Use Case* Memulai Permainan

Skenario *use case* ini menjelaskan interaksi antara aktor, yaitu pemain, dengan *use case* memulai permainan yang dijelaskan pada tabel 3.11.

Tabel 3.11 Skenario *Use Case* Memulai Permainan

Identifikasi
Nama Use Case : Memulai permainan
Deskripsi : Proses untuk memulai permainan
Aktor : Pemain
Kondisi awal : Pemain berada di menu utama permainan
Pemicu : Pemain memilih menu mulai permainan dari menu utama
Skenario utama: 1. Sistem memainkan slide intro permainan 2. Sistem menampilkan permainan level 1
Skenario alternatif: Tidak ada

2. Skeario *Use Case* melihat petunjuk

Skenario *Use Case* ini menjelaskan interaksi antara aktor, yaitu pemain, dengan *Use Case* melihat petunjuk yang dijelaskan pada tabel 3.12.

Tabel 3.12 Skenario *Use Case* melihat petunjuk

Identifikasi
Nama Use Case : Melihat petunjuk
Deskripsi : Proses untuk menampilkan petunjuk permainan
Aktor : Pemain
Kondisi awal : Pemain berada di menu utama permainan
Pemicu : Pemain memilih menu petunjuk dari menu utama
Skenario utama: 1. Sistem menampilkan petunjuk permainan
Skenario alternatif: Tidak ada

3. Skeario *Use Case* melihat *top score*

Skenario *Use Case* ini menjelaskan interaksi antara aktor, yaitu pemain, dengan *Use Case* melihat petunjuk yang dijelaskan pada tabel 3.13

Tabel 3.13 Skenario *Use Case* melihat *top score*

Identifikasi
Nama Use Case : Melihat *top score*
Deskripsi : Proses untuk menampilkan *top score* permainan
Aktor : Pemain
Kondisi awal : Pemain berada di menu utama permainan
Pemicu : Pemain memilih menu *top score* dari menu utama
Skenario utama: 1. Sistem menampilkan *top score* permainan
Skenario alternatif: Tidak ada

4. Skeario *Use Case* mengambil daging

Skenario *Use Case* ini menjelaskan interaksi antara aktor, yaitu pemain, dengan *Use Case* mengambil daging yang dijelaskan pada tabel 3.14.

Tabel 3.14 Skenario *Use Case* mengambil daging

Identifikasi
Nama Use Case : Mengambil daging
Deskripsi : Proses untuk mengambil daging
Aktor : Pemain
Kondisi awal : Pemain berada di dalam permainan
Pemicu : Pemain bergerak mendekati daging
Skenario utama: 1. Karakter menyentuh daging 2. Sistem menambahkan poin yang telah terkumpul
Skenario alternatif:

1a. Karakter tidak menyentuh daging 2a. Sistem tidak menambahkan poin yang telah terkumpul

5. Skeario *Use Case* mengambil *item speed*

Skenario *Use Case* ini menjelaskan interaksi antara aktor, yaitu pemain, dengan *Use Case* mengambil *item speed* yang dijelaskan pada tabel 3.15.

Tabel 3.15 Skenario *Use Case* mengambil *item speed*

Identifikasi
Nama Use Case : Mengambil *item speed*
Deskripsi : Proses untuk mengambil *item speed*
Aktor : Pemain
Kondisi awal : Pemain berada di dalam permainan
Pemicu : Pemain bergerak mendekati *item speed*
Skenario utama: 1. Karakter menyentuh *itemspeed* 2. Sistem menambahkan kecepatan karakter
Skenario alternatif: 1a. Karakter tidak menyentuh *itemspeed* 2a. Sistem tidak menambahkan kecepatan karakter

6. Skeario *Use Case* membuat karakter melompat

Skenario *Use Case* ini menjelaskan interaksi antara aktor, yaitu pemain, dengan *Use Case* membuat karakter melompat yang dijelaskan pada tabel 3.16.

Tabel 3.16 Skenario *Use Case* membuat karakter melompat

Identifikasi
Nama Use Case : Membuat karakter melompat
Deskripsi : Proses membuat karakter melompat
Aktor : Pemain
Kondisi awal : Pemain berada di dalam permainan
Pemicu : Pemain menekan tombol *left* pada *mouse*
Skenario utama: 1. Sistem mebuat karakter melompat
Skenario alternatif: Tidak ada

7. Skeario *Use Case* menghindari musuh dan rintangan

Skenario *Use Case* ini menjelaskan interaksi antara aktor, yaitu pemain, dengan *Use Case* membuat karakter melompat yang dijelaskan pada tabel 3.17.

Tabel 3.17 Skenario *Use Case* menghindari musuh dan rintangan

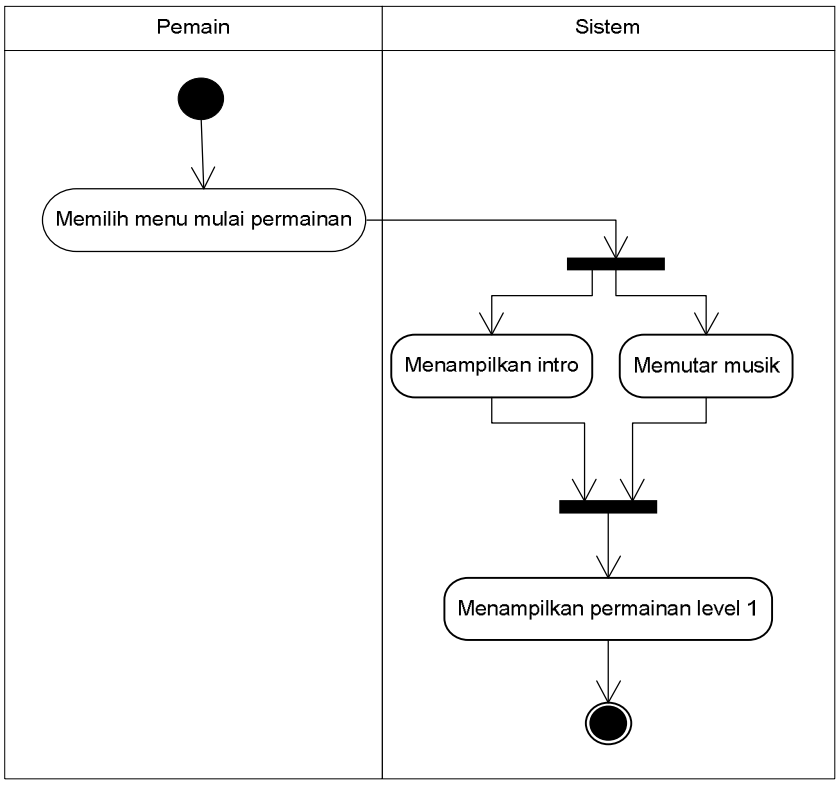
Identifikasi
Nama Use Case : Menghindari musuh dan rintangan
Deskripsi : Proses membuat karakter melompat
Aktor : Pemain
Kondisi awal : Pemain berada di dalam permainan
Pemicu : Pemain mendekati musuh dan rintangan
Skenario utama: 1. Pemain menghindari musuh dan rintangan 2. Sistem memeriksa apakah karakter menabrak musuh atau rintangan
Skenario alternatif: 2a. Karakter menabrak musuh atau rintangan, permainan berakhir pemain kalah 2b. Karakter tidak menabrak musuh atau rintangan, karakter terus bergerak

3.1.7.6 *Activity* Diagram

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

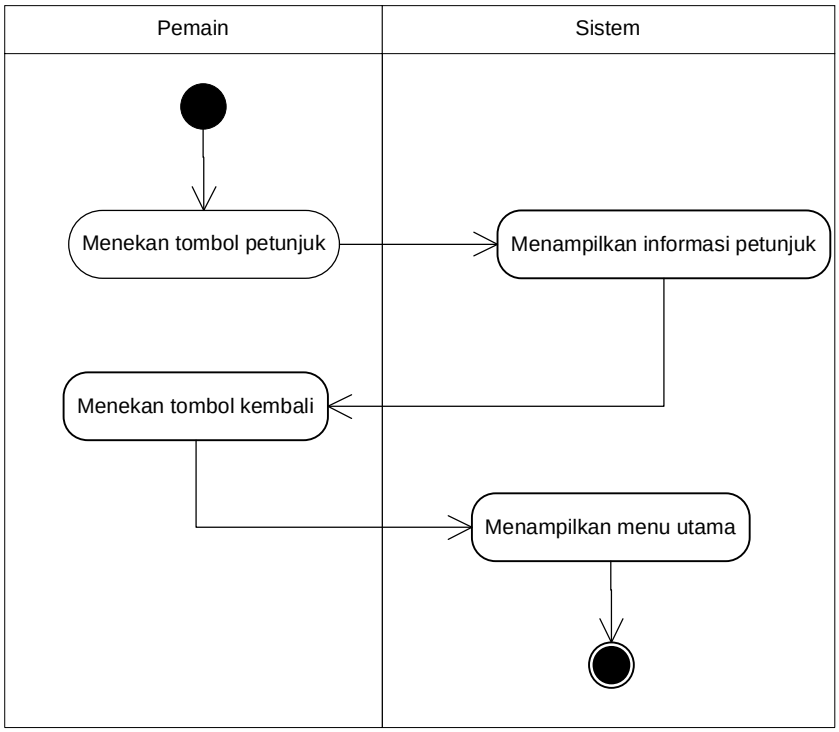
Activity diagram merupakan *state* diagram khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *Activity* diagram tidak menggambarkan behaviour internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

1. Diagram *Activity* Memulai Permainan



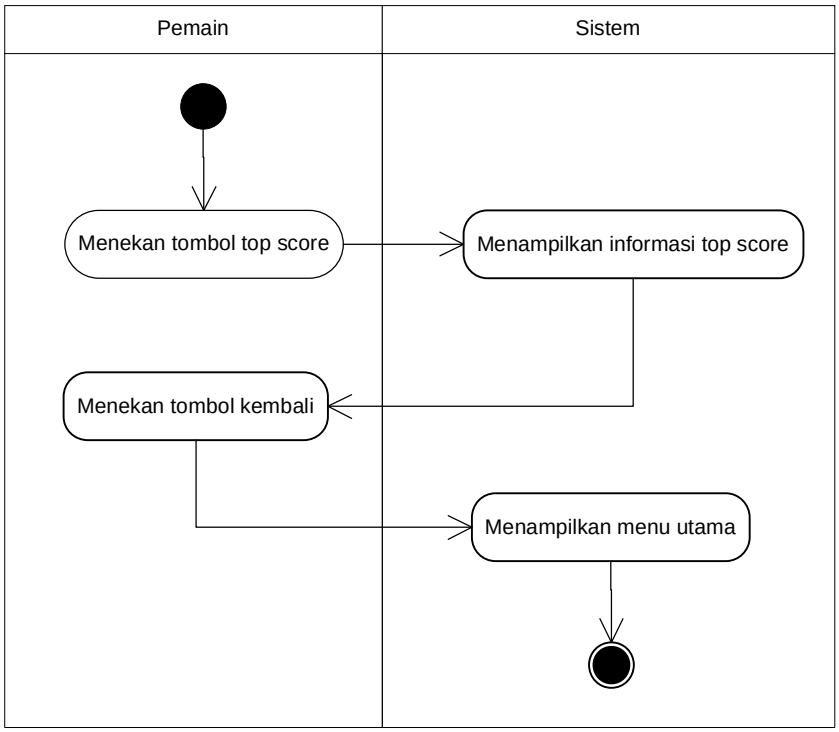
Gambar 3.12 Diagram *Activity* Memulai Permainan Pada *Fly Eagle*

2. Diagram *Activity* Melihat Petunjuk



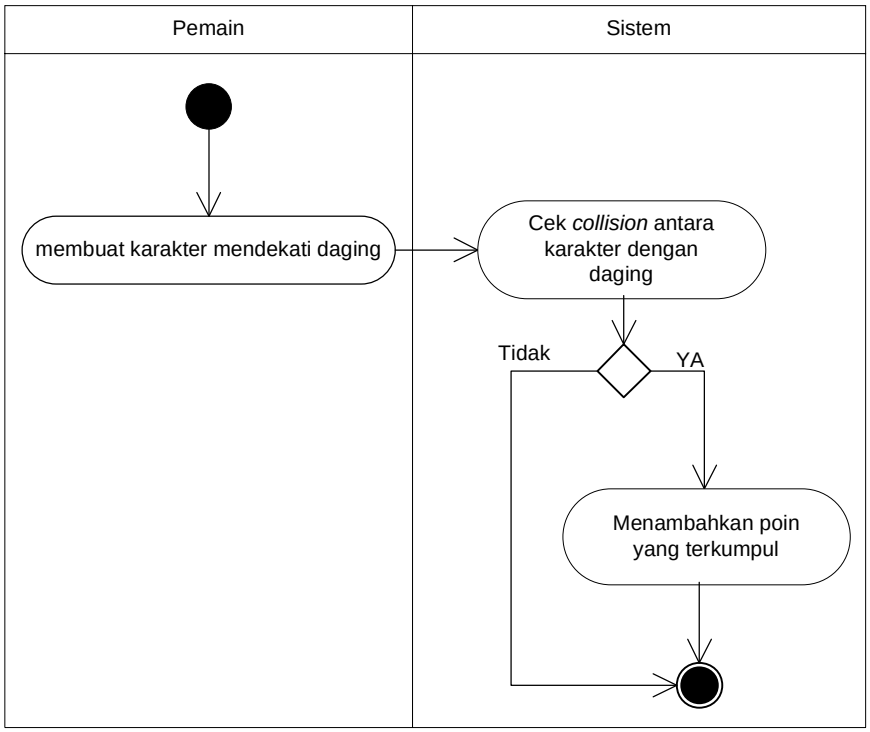
Gambar 3.13 Diagram *Activity* Melihat Petunjuk Pada *Fly Eagle*

3. Diagram *Activity* Melihat *Top Score*



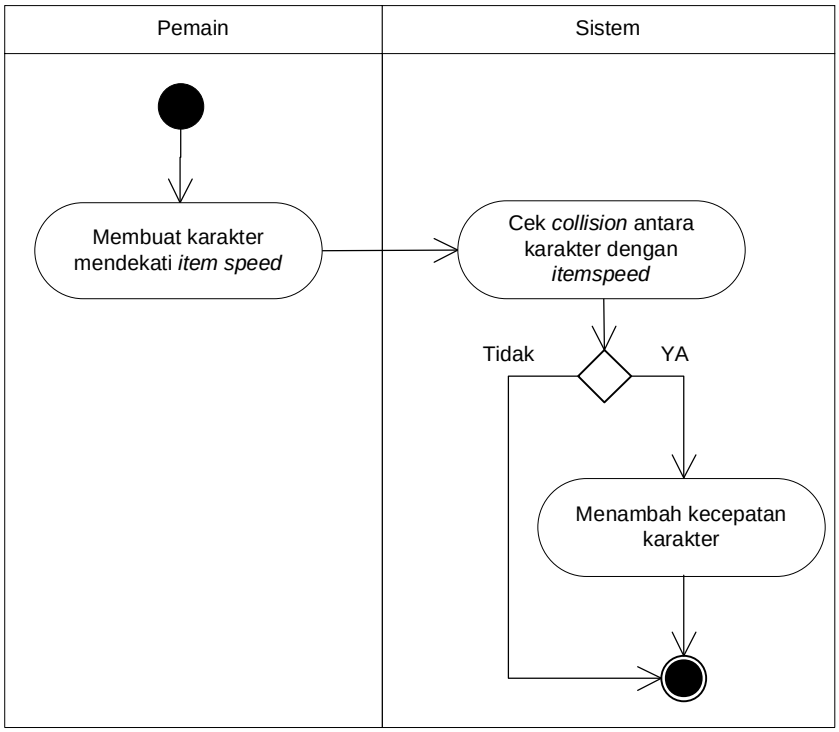
Gambar 3.14 Diagram *Activity* Melihat *Top Score* Pada *Fly Eagle*

4. Diagram *Activity* Mengambil daging



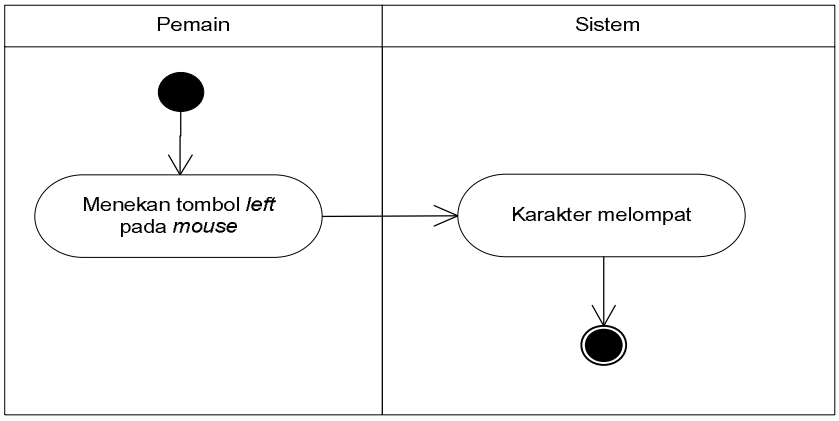
Gambar 3.15 Diagram *Activity* Mengambil daging Pada *Fly Eagle*

5. Diagram *Activity* mengambil *item speed*



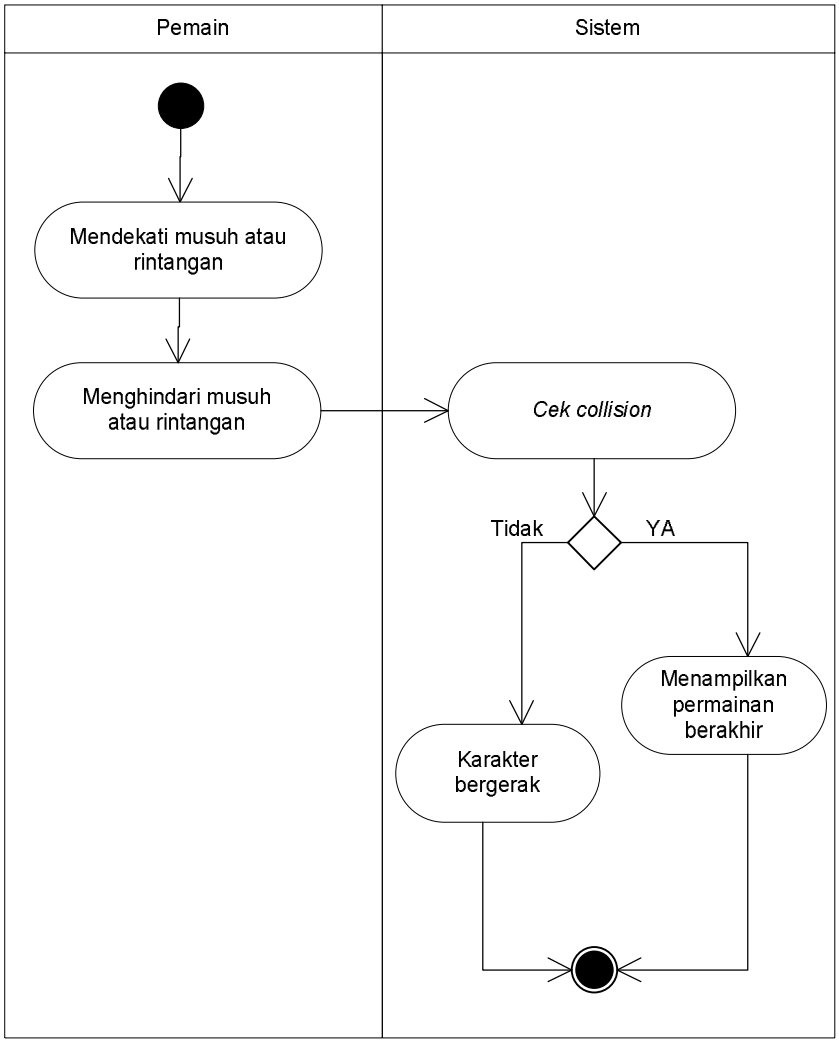
Gambar 3.16 Diagram *Activity* mengambil *item speed*

6. Diagram *Activity* membuat karakter melompat



Gambar 3.17 Diagram *Activity* membuat karakter melompat Pada *Fly Eagle*

7. Diagram *Activity* menghindari musuh dan rintangan



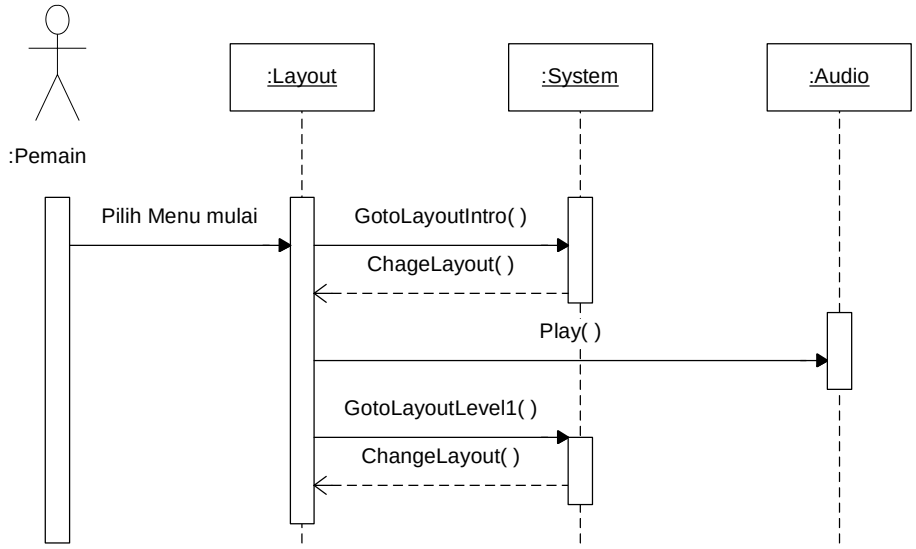
Gambar 3.18 Diagram *Activity* membuat menghindari musuh dan rintangan

3.1.7.7 *Sequence* Diagram

Sequence Diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem berupa message yang digambarkan terhadap waktu. Sequence diagram terdiri atas dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).

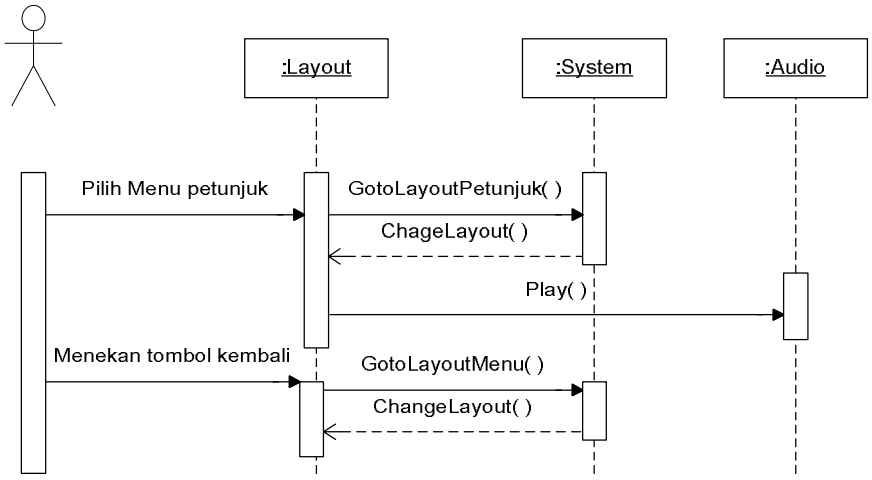
Sequence Diagram yang ada dalam game *Fly Eagle* adalah sebagai berikut :

1. *Sequence Diagram* memulai permainan



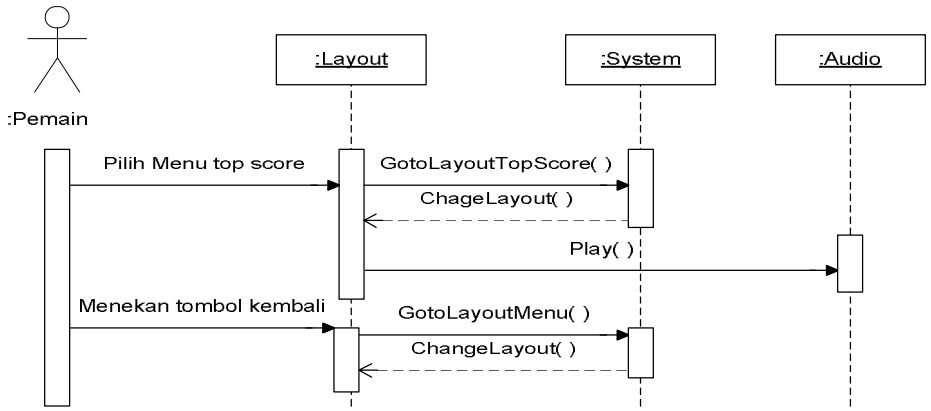
Gambar 3.19 *Sequence Diagram* Mulai Permainan Pada *Fly Eagle*

2. *Sequence Diagram* Melihat Petunjuk



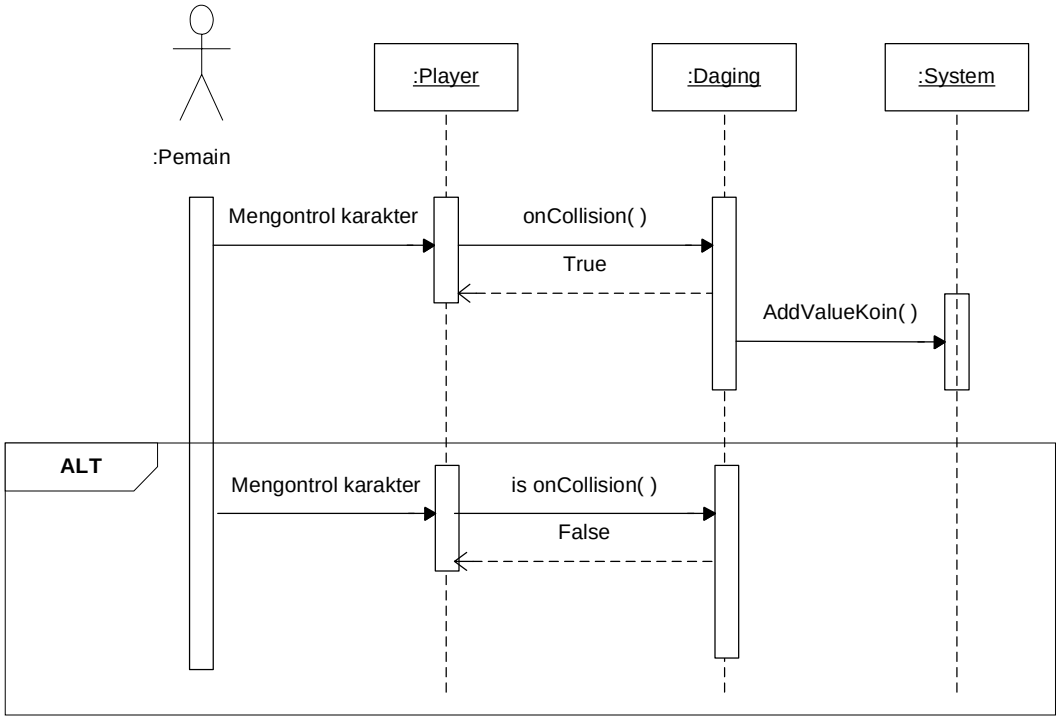
Gambar 3.20 *Sequence Diagram* Melihat Petunjuk Pada *Fly Eagle*

3. *Sequence Diagram* Melihat *Top Score*



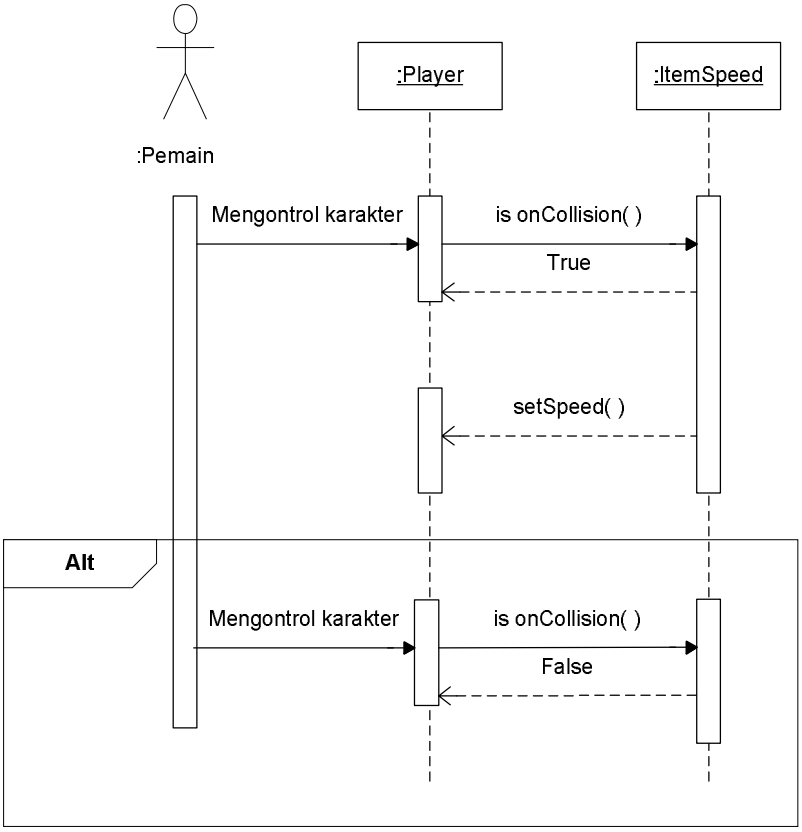
Gambar 3.21 *Sequence Diagram* Melihat *Top Score* Pada *Fly Eagle*

4. *Sequence Diagram* Mengambil daging



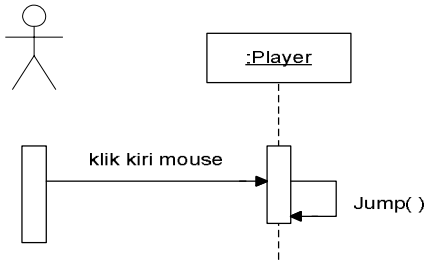
Gambar 3.22 *Sequence Diagram* Mengambil daging

5. *Sequence Diagram* mengambil *item speed*



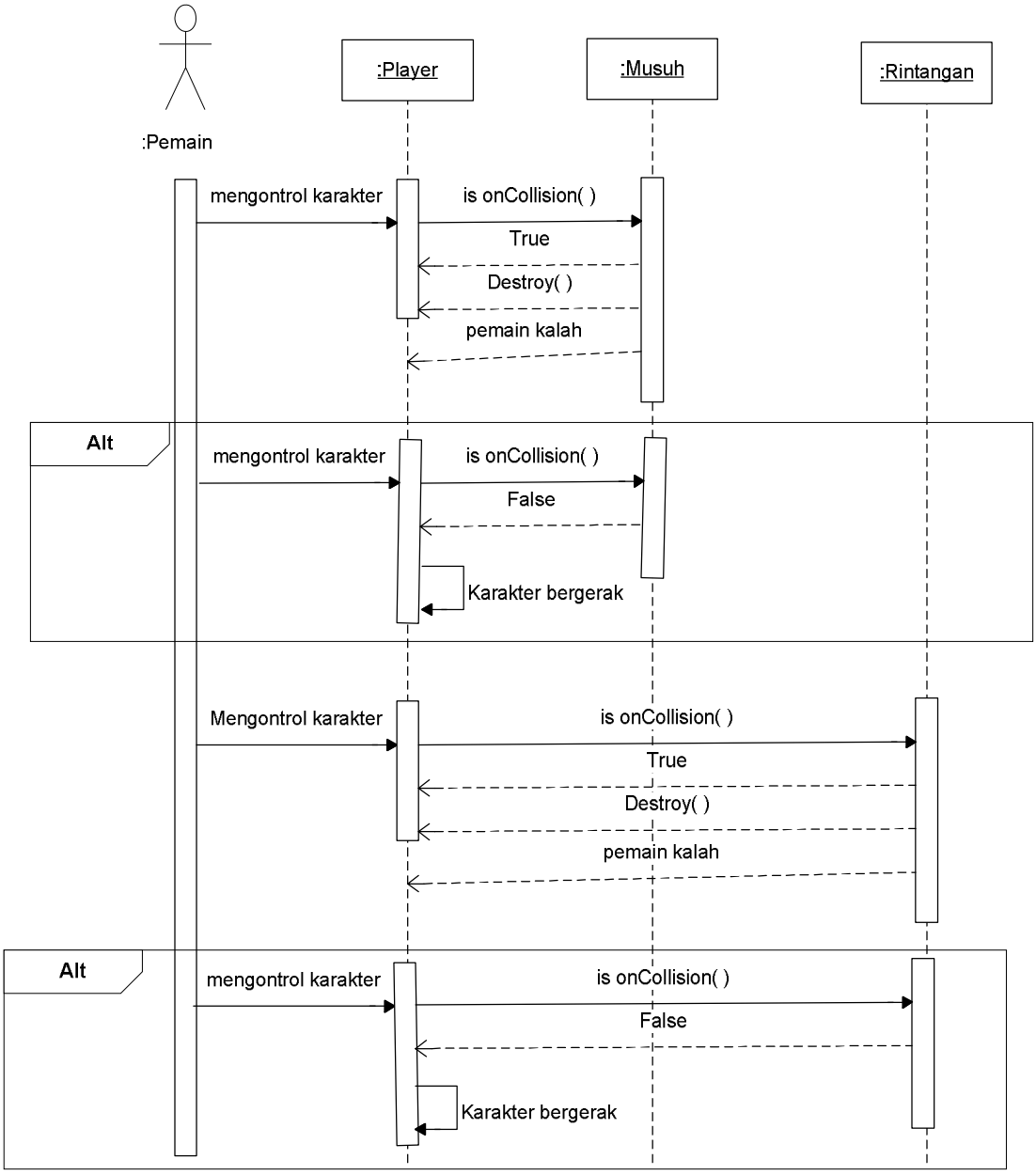
Gambar 3.23 *Sequence Diagram* mengambil *item speed*

6. *Sequence Diagram* membuat karakter melompat



Gambar 3.24 *Sequence Diagram* membuat karakter melompat

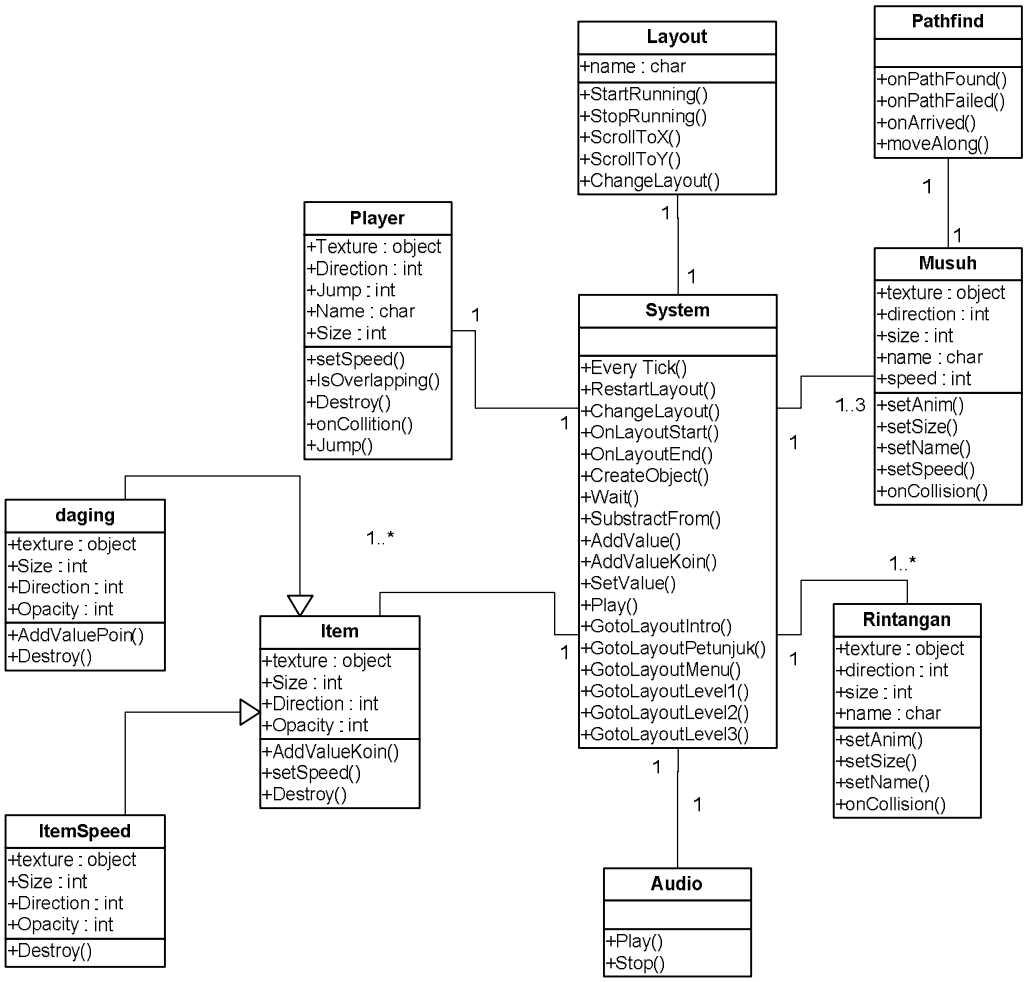
7. *Sequence Diagram* menghindari musuh dan rintangan



Gambar 3.25 *Sequence Diagram* menghindari musuh dan rintangan

3.1.7.8 *Class* Diagram

Class Diagram menggambarkan keadaan suatu sistem dengan menjelaskan keterhubungan antara suatu class dengan class yang lain yang terdapat pada sistem tersebut. *Class* Diagram dalam game *Fly Eagle* adalah sebagai berikut:



Gambar 3.26 *Class Diagram* pada game *Fly Eagle*

3.2 Perancangan Sistem

Perancangan sistem menjelaskan tentang rancangan game yang dibangun dimana perancangan ini meliputi perancangan menu, antarmuka, pesan (dialog), dan jaringan semantik.

3.2.1 Perancangan Komponen Game

Perancangan komponen permainan menjelaskan tentang rancangan komponen dalam game *Fly Eagle* yang mana terdiri dari karakter pemain, musuh, *item* serta *storyboard*.

3.2.1.1 Karakter Pemain

Karakter pemain dalam game *Fly Eagle* beserta keterangannya dapat dilihat pada tabel 3.18 berikut :

Tabel 3.18 Karakter Pemain

No	Nama	Gambar	Keterangan	Status
1	**Elang**		Karakter utama dari game ini, dia tersesat dan ingin kembali ke hutan	kecepatan = 200

3.2.1.2 Karakter Musuh

Musuh dalam game *Fly Eagle* dapat dilihat pada table 3.19 berikut :

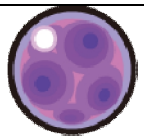
Tabel 3.19 Karakter Musuh

No	Nama	Gambar	Kemampuan
1	Anak Kecil		Kecepatan : 200
2	Turis		Kecepatan : 300
3	Pemburu		Kecepatan : 400
4	Burung		Bisa terbang, kecepatan : 300 AI menggunakan A*
5	Burung Merah		Bisa terbang, kecepatan : 400 AI menggunakan A*

3.2.1.3 Item

Item yang ada di dalam game dapat dilihat pada table 3.20 berikut :

Tabel 3.20 Item

No	Nama	Gambar	Kemampuan
1	*Item Speed*		Mengurangi kecepatan *player* elang
2	Daging		Menambahkan poin yang terkumpul

3.2.1.4 *Storyboard*

Story Board merupakan gambaran sketsa permainan secara umum yang disusun berdasarkan cerita atau *storyline* game. Berikut *storyboard* untuk game *Fly Eagle* :

1. Level 1

   Level 1 berlatar di rumah



Gambar 3.27 Storyboard level 1

2. Level 2

   Level 2 berlatar di kota



Gambar 3.28 Storyboard level 2

3. Level 3

   Level 3 berlatar di hutan

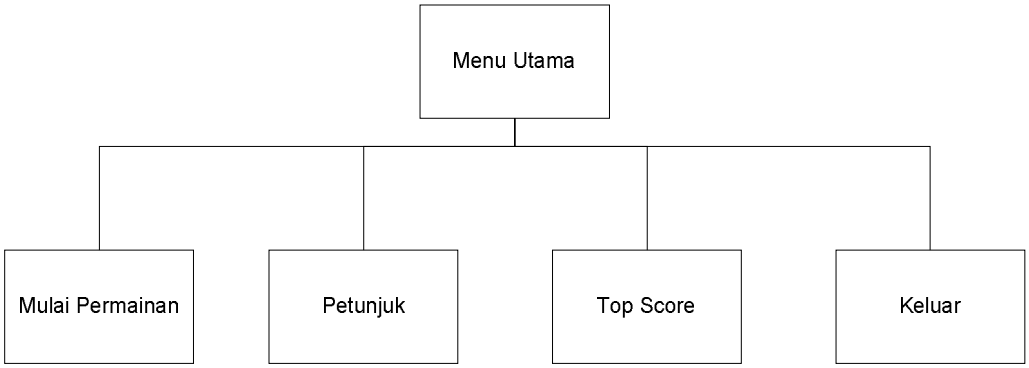


Gambar 3.29 Storyboard level 3

3.2.2 Perancangan Arsitektur Menu

Untuk mempermudah dalam 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 nantinya akan diakses terlebih dahulu setelah program

selesai dibuat. Perancangan arsitektur menu pada aplikasi permainan *Fly Eagle* dapat dilihat pada gambar 3.30.



Gambar 3.30 Arsitektur Menu *Fly Eagle*

a. Mulai Permainan : merupakan submenu untuk memulai permainan baru
b. Petunjuk : merupakan submenu untuk menampilkan informasi tentang cara memainkan aplikasi permainan *Fly Eagle*
c. *Top Score* : merupakan submenu menampilkan *top score*
d. Keluar : merupakan submenu untuk keluar dari aplikasi permainan *Fly Eagle*

3.2.3 Perancangan Antar Muka

Perancangan antarmuka dibutuhkan untuk mewakili keadaan sebenarnya dari aplikasi yang akan dibangun, berikut akan disajikan perancangan antarmuka dari aplikasi yang akan dibangun:

Perancangan antarmuka menu pada aplikasi permainan *Fly Eagle* dapat dilihat pada gambar

T01	
MENU **Mulai Permainan** **Petunjuk** **Top Score** **Keluar**	- klik mulai permainan untuk menuju T02 - klik Petunjuk untuk menuju T08 - klik *top score* untuk menuju T09 - Klik keluar keluar aplikasi
Ukuran layar : *Width* 800px (*default)* Besar font beragam disesuaikan	

Gambar 3.31 Perancangan Antarmuka Menu *Fly Eagle*

Perancangan antarmuka *intro* pada aplikasi permainan *Fly eagle* dapat dilihat pada gambar

T02	
Tampilan Intro **Screen**	
Ukuran layar : *Width* 800px (*default)* Besar font beragam disesuaikan	

Gambar 3.32 Perancangan Antarmuka *intro Fly Eagle*

Perancangan antarmuka tampilan level 1 pada aplikasi permainan *Fly eagle* dapat dilihat pada gambar

T03	
Tampilan Game **Level 1**	- Setelah pemain menyelesaikanya akan masuk ke T04
Ukuran layar : *Width* 800px (*default)* Besar font beragam disesuaikan	

Gambar 3.33 Perancangan Antarmuka level 1 pada *Fly Eagle*

Perancangan antarmuka tampilan level 2 pada aplikasi permainan *Fly Eagle* dapat dilihat pada gambar

<table>
<tr><td colspan="2">T04</td></tr>
<tr><td>Tampilan Game
Level 2</td><td>- Setelah pemain menyelesaikanya akan masuk ke T05</td></tr>
<tr><td colspan="2">Ukuran layar : Width 800px (default)
Besar font beragam disesuaikan</td></tr>
</table>

Gambar 3.34 Perancangan Antarmuka level 2 pada *Fly Eagle*

Perancangan antarmuka tampilan level 3 pada aplikasi permainan *Fly Eagle* dapat dilihat pada gambar

T05	
Tampilan Game **Level 3**	- Setelah pemain menyelesaikanya akan masuk ke T06
Ukuran layar : *Width* 800px (*default)* Besar font beragam disesuaikan	

Gambar 3.35 Perancangan Antarmuka level 3 pada *Fly Eagle*

Perancangan antarmuka tamat pada aplikasi permainan *Fly Eagle* dapat dilihat pada gambar

T06	
TAMAT !!	-
Ukuran layar : *Width* 800px (*default)* Besar font beragam disesuaikan	

Gambar 3.36 Perancangan Antarmuka tamat pada *Fly Eagle*

Perancangan antarmuka *game over* pada aplikasi permainan *Fly Eagle* dapat dilihat pada gambar

T07	
Game Over	- Tekan Ya pada M01 untuk bermain kembali pada level sebelumnya - Tekan tidak pada M01 untuk kembali ke T01
Ukuran layar : *Width* 800px (*default)* Besar font beragam disesuaikan	

Gambar 3.37 Perancangan Antarmuka *game over* pada *Fly Eagle*

Perancangan antarmuka petunjuk pada aplikasi permainan *Fly Eagle* dapat dilihat pada gambar

T08	
Petunjuk **Kembali**	- Tekan Kembali untuk menuju T01
Ukuran layar : *Width* 800px (*default)* Besar font beragam disesuaikan	

Gambar 3.38 Perancangan Antarmuka petunjuk pada *Fly Eagle*

Perancangan antarmuka *top score* pada aplikasi permainan *Fly Eagle* dapat dilihat pada gambar

<table>
<tr><td colspan="2">T09</td></tr>
<tr><td>Top Score

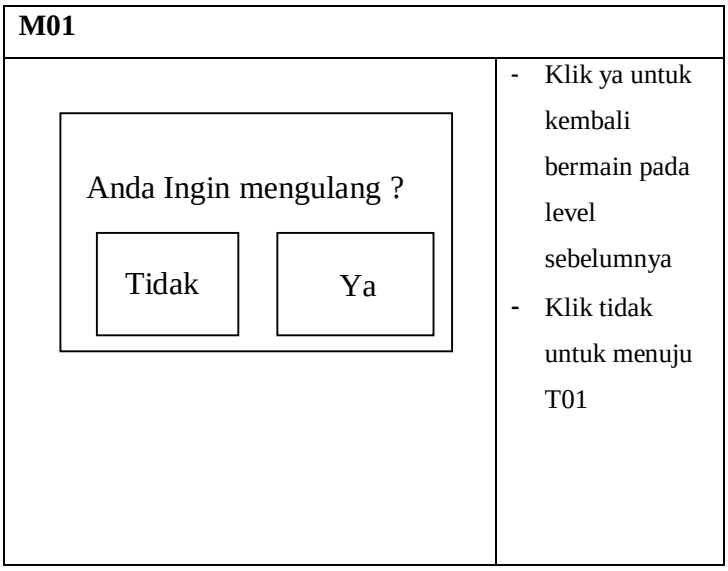
1. 6.
2. 7.
3. 8.
4. 9.
5. 10.

Kembali</td><td>- Tekan Kembali untuk menuju T01</td></tr>
<tr><td colspan="2">Ukuran layar : Width 800px (default)
Besar font beragam disesuaikan</td></tr>
</table>

Gambar 3.39 Perancangan Antarmuka *top score* pada *Fly Eagle*

3.2.4 Perancangan Pesan

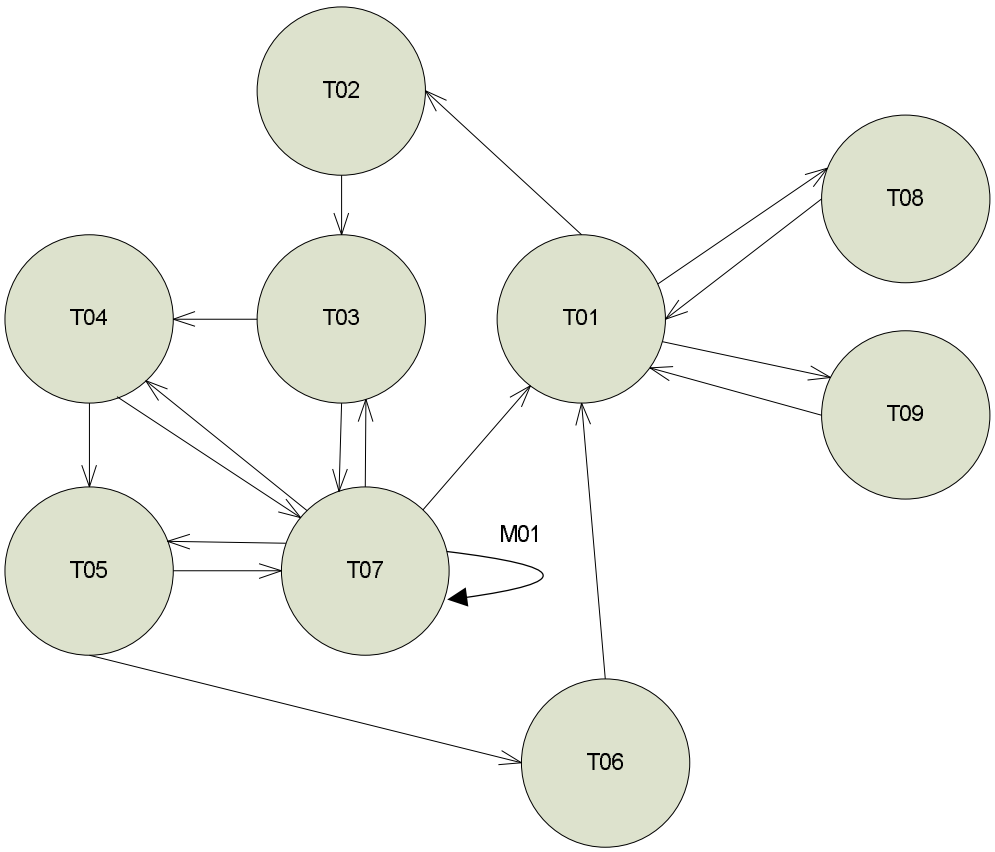
Pesan digunakan untuk memberi pemberitahuan, peringatan atau untuk meminta konfirmasi dari suatu aksi. Pesan dalam game *Fly Eagle* adalah sebagai berikut :



Gambar 3.40 Pesan Konfirmasi Mengulang Permainan

3.2.5 Jaringan Semantik

Jaringan semantik merupakan jaringan data yang menunjukan hubungan antar berbagai objek. Gambar 3.41 menunjukkan jaringan semantik dari game yang dibangun.



Gambar 3.41 Jaringan Semantik *Fly Eagle*