BAB III
ANALISIS DAN PERANCANGAN SISTEM

III.1 Analisis Sistem

Analisis sistem adalah teknik pemecahan masalah yang menguraikan bagian-bagian komponen dengan mempelajari seberapa bagus bagian-bagian komponen tersebut berkerja dan berinteraksi untuk mencapai tujuan mereka.

III.1.1 Analisis Masalah

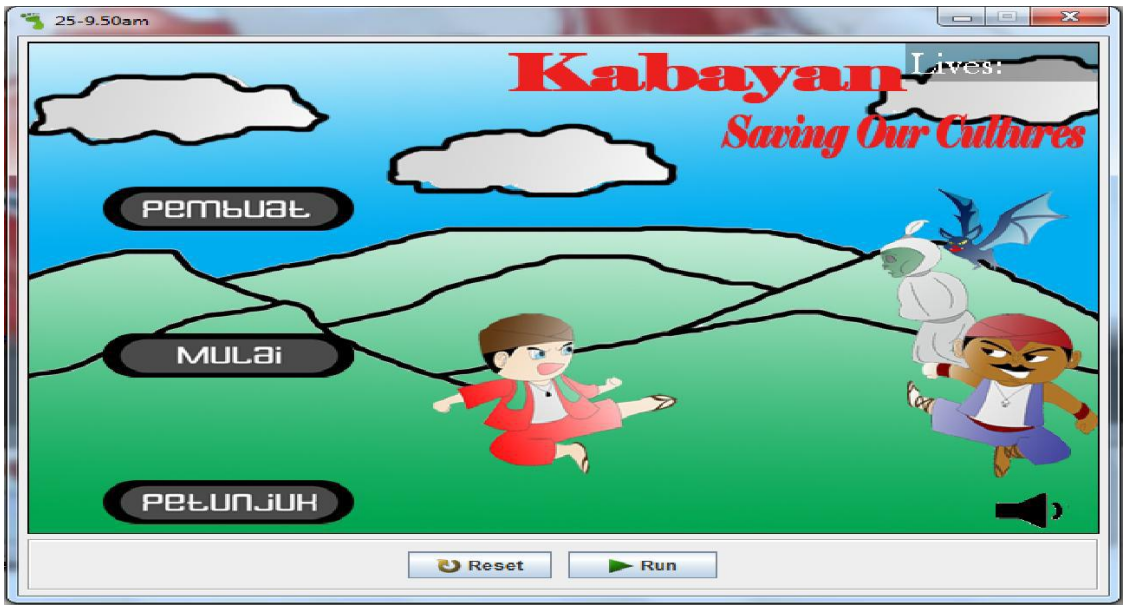
Mengajarkan anak tentang pentingnya menjaga kelestarian lingkungan bukanlah hal yang mudah, diperlukan sebuah media pengajaran yang menyenangkan dan mudah dipelajari untuk anak-anak. Anak-anak memiliki kecendurangan lebih suka bermain daripada belajar, oleh karena itu *game* edukasi bisa menjadi sebuah media yang tepat untuk mengajarkan anak-anak tentang betapa pentingnya menjaga kelestarian lingkungan hidup. Dengan *game* edukasi anak-anak akan bermain sambil belajar, selain itu juga *game* dapat membantu koordinasi visual motorik, mengembangkan visual motorik halus, dan mendorong anak mencapai target tertentu. *Video games* juga membentuk sikap sportif, melatih daya ingat, konsentrasi, kemampuan berlogika dan menyelasaikan masalah. Namun video *games* juga bisa memberi efek buruk, bila dikonsumsi berlebihan dan mengandung materi yang tidak sesuai dengan usianya. Maka dari perancangan *game* "*Save Our Forest*" unsur kekerasan dalam *game* akan diminimalisir, untuk menghindari anak-anak yang cenderung suka meniru sikap-sikap kekerasan yang ada dalam *video games.*

III.1.2 Analisis *Game* Sejenis

Pada analisis *game* sejenis, akan dilakukan observasi terhadap *game* yang memiliki kesamaan *gameplay* dan *genre* dengan *game* yang akan dibangun. Tujuan dari analisis ini adalah untuk mendapatkan konsep permainan dan

perbandingan *game* yang telah ada. *Game* yang akan dibandingkan adalah *Game Kabayan Saving Our Culture*:

III.1.2.1 *Game* 2D Kabayan *Saving Our Cultures*



Gambar III.1 Tampilan *Game* 2D Kabayan *Saving Our Cultures*

Game Kabayan *Saving Our Cultures* menceritakan perjalanan seorang pemuda dari suku Sunda yaitu Kabayan yang harus bertualang untuk menyelamatkan alat musik tradisional yaitu angklung yang telah dicuri oleh seorang penjahat dari Negara Misterius yang bernama Juragan Asep.

Dalam *game* ini, Kabayan harus mengalami berbagai macam petualangan dalam usahanya untuk mengejar Juragan Asep yang kabur ke negeri seberang dan berniat untuk membawa Angklung ke Negaranya. Kabayan harus berjuang mati-matian untuk merebut kembali kesenian asli Jawa Barat tersebut.

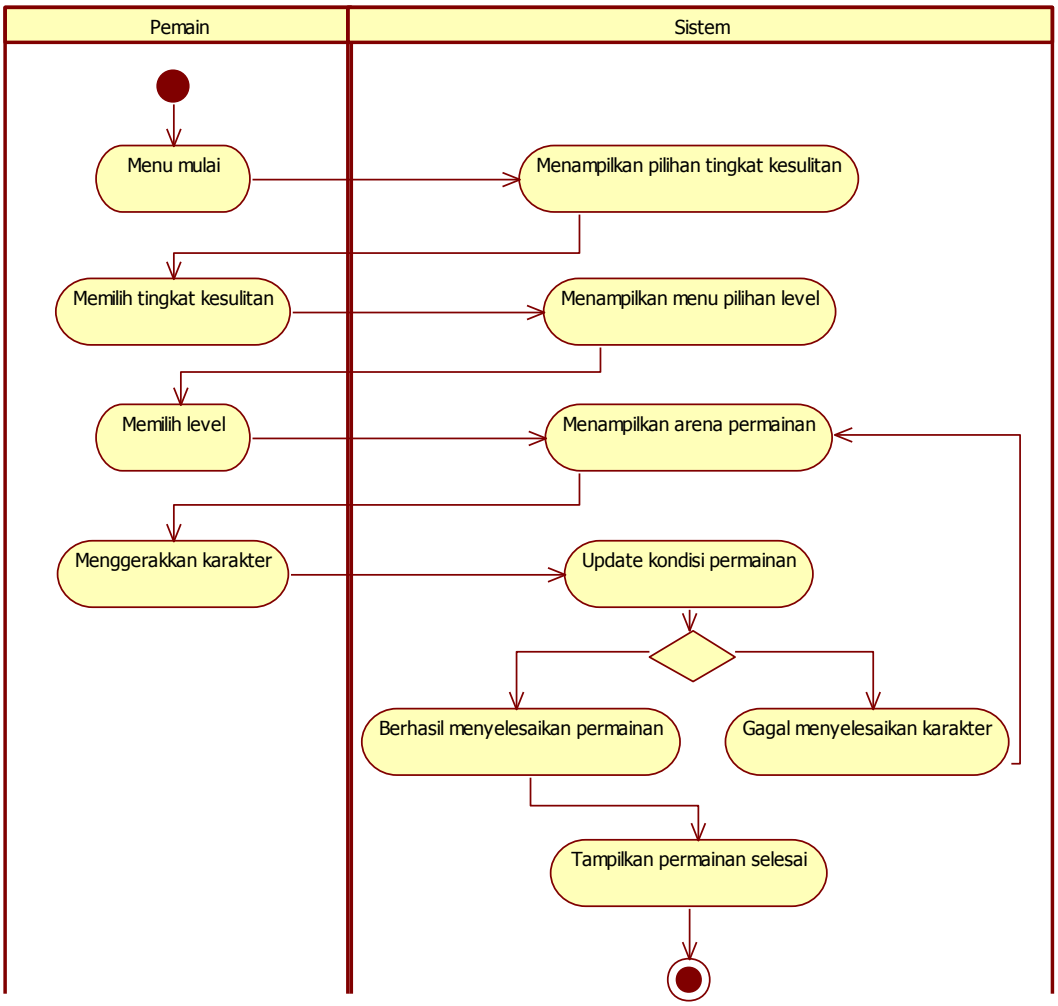
A. Gameplay

Dalam *Game Kabayan Saving Our Culture*, Pemain berperan sebagai Kabayan dan diharuskan melalui seluruh rintangan yang ada untuk menyelamatkan alat musik yang telah dicuri oleh Negara misterius. Pemain dapat memilih 2 tingkat yang tersedia yaitu “Mudah” dan “Sulit”, setelah memilih salah satu pemain diharuskan memilih level. Pada awal permainan, pemain diberikan 3 nyawa.

B. Skenario *game*

1. Pemain memilih menu mulai
2. Pemain memilih tingkat kesulitan
3. Pemain harus menghindari rintangan dan melawan musuh yang ada
4. Jika pemain terkena serangan, maka darah pemain akan berkurang

Alur sistem dari permainan game *Kabayan Saving Our Culture* dapat digambarkan melalui *activity diagram* yang dapat dilihat pada gambar III.2



Gambar III.2 Alur Permainan *game* 2D Kabayan *Saving Our Cultures*

Kelebihan dari *game* 2D Kabayan *Saving Our Cultures* adalah :

1. Diterapkannya kecerdasan buatan pada musuh yang membuat permainan semakin menantang.
2. Pemain dapat memilih tingkat kesulitan permainan.

Kekurangan dari *game* 2D Kabayan *Saving Our Cultures* adalah :

1. Pengangkatan latar belakang cerita yang mengisahkan tentang pencurian alat musik kurang.
2. Unsur edukasi masih kurang karena tidak terdapat penjelasan mengenai alat musik calung yang menjadi latar belakang *game* ini.

3. Desain kurang menarik.

III.1.3 Analisis *Game* yang Akan Dikembangkan

Pada bagian analisis game *Save Our Forest* ini akan dibahas mengenai storyline, analisis tingkat kesulitan, *gameplay* dan *scoring*.

III.1.3.1 *Storyline*

Save Our Forest adalah *game* yang bertujuan memberikan edukasi bahaya kerusakan lingkungan khususnya hutan terhadap anak-anak. *Game* ini menceritakan petualangan seorang polisi hutan yang ditugaskan untuk menghentikan pembalakan liar di kawasan hutan lindung, berbekal pengalaman dan keahliannya sebagai polisi hutan, ia pun berusaha menghentikan pembalakan liar dan berusaha mencari tahu dalang di balik pembalakan liar ini.

III.1.3.2 Analisis Tingkat Kesulitan

Dalam *game* ini, terdapat 3 *level* yang masing-masing *level* dibedakan berdasarkan alur cerita, tingkat kesulitan rintangan yang harus dihadapi, jumlah, kekuatan dan jenis musuh yang bertambah. Di setiap peningkatan *level*, *user* akan dihadapkan pada musuh dengan tingkat kesulitan yang akan bertambah, mulai dari kecepatan dan kesukaran musuh tersebut. Analisis tingkat kesulitan akan dijelaskan di tabel III.1

Tabel III.1 Analisis Tingkat Kesulitan

Level	Jumlah musuh	Total poin yang tersebar di game	Boss
Level 1	12	15	Tidak ada
Level 2	15	25	Tidak ada
Level 3	17	25	Ada

III.1.3.3 *Gameplay*

Tugas utama pemain (*player*) dalam memainkan aplikasi *game* ini yaitu melalui *level-level*, dimana setiap *level* menyediakan petualangan dengan menampilkan beberapa rintangan, karakter *level* yang berbeda-beda maupun jenis musuh yang harus dilawan. Pemain (*player*) memulai permainan dari level 1. Di

setiap *level* memiliki rintangan yang berbeda-beda dimana setiap level memiliki tingkat kesulitan yang semakin sulit untuk di lewati.

1. *Level* 1

   Di *level* 1, pemain diharuskan mencari petunjuk tentang lokasi keberadaan penebangan hutan liar terjadi. Pada *level* ini pemain dibekali dengan 100 nyawa dan total 40 peluru. Pemain dapat bergerak ke kiri, ke kanan, melompat, menembak dan mengisi ulang peluru. Pemain diharuskan untuk melewati rintangan dan musuh yang menghalangi jalannya misi. Kumpulkan kantung bibit untuk menambah skor permainan, kotak P3K untuk menambah nyawa. Pemain harus mendapatkan gulungan peta yang berisikan informasi tentang lokasi penebang liar.

2. *Level* 2

   Di *level* 2, setelah pemain mendapatkan informasi tentang lokasi kelompok penebang hutan liar, pemain melanjutkan perjalanan menuju lokasi penebang liar. Dalam perjalanannya pemain dihadang oleh sekelompok anggota penebang liar tersebut. Pada *level* ini pemain dibekali dengan 100 nyawa dan 40 peluru. Pemain dapat bergerak ke kiri, ke kanan dan melompat. Pemain diharuskan untuk melewati rintangan dan musuh yang menghalangi jalannya misi. Kumpulkan kantung bibit untuk menambah skor permainan.

3. *Level* 3

   Di *level* 3, pemain telah sampai di kamp penebang liar dan pada akhir permainan, pemain harus berhadapan dengan *boss* yang harus di kalahkan untuk mengakhiri permainan.

III.1.3.4 *Scoring*

Dalam *game* ini, pemain harus mengumpulkan kantung bibit tanaman yang tersebar di setiap *level*. Untuk setiap bibit memiliki nilai 10 poin dan poin tersebut akan habis atau kembali ke keadaan awal jika pemain mati atau meninggalkan permainan.

Jika pemain dapat mengumpulkan sampai 30% poin dari total poin di setiap level, di akhir *level*, pemain akan mendapatkan 1 bintang. Jika pemain dapat mengumpulkan sampai 70% poin dari total poin di setiap level, di akhir *level*, pemain akan mendapatkan 2 bintang, dan jika dapat mengumpulkan lebih dari 70% akan mendapatkan 3 bintang.

III.1.4 Analisis Algoritma A*

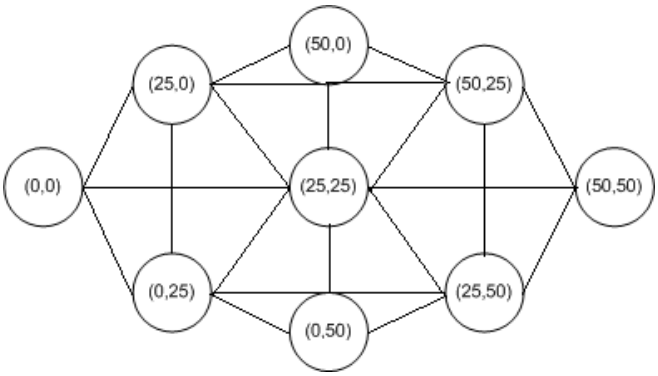
Pada bagian ini akan dibahas analisis algoritma A*, algoritma ini menjadi *complete* dan optimal dikarenakan algoritma ini menggabungkan informasi dari algoritma Djikstra (lebih memilih *node* yang dekat dengan *node* awal) dan informasi yang digunakan algoritma *Best First Search* (lebih memilih *node* yang dekat dengan *node* tujuan). Algoritma A* akan diterapkan pada objek *boss* dalam melakukan pencarian rute terpendek lokasi karakter pemain di arena permainan. Berikut ini adalah istilah-istilah yang akan dibahas:

1. *Open List* adalah *list* yang menyimpan kemungkinan *path* yang akan diperiksa. *Open list* digunakan untuk menentukan secara selektif (berdasarkan nilai f) jalan yang dikira lebih dekat menuju tujuan. *Open list* berisi *node* yang masih memiliki peluang untuk terpilih sebagai *best node*.
2. *Closed list* adalah *list* untuk menyimpan simpul-simpul yang sudah pernah dibangkitkan dan sudah pernah terpilih sebagai *best node*. *Closed list* dan *open list* bertujuan untuk menghindari penulusuran berulang rute agar tidak masuk kembali ke *open list*.
3. Nilai F adalah nilai yang menyatakan seberapa bagus sebuah *node*. Nilai F merupakan hasil dari f(n).
4. Nilai G adalah nilai dari fungsi g(n) adalah banyaknya langkah untuk menuju *path* saat ini.
5. Nilai H adalah nilai estimasi untuk sampai pada suatu tujuan mulai dari n.

Diasumsikan arena permainan dibagi-bagi kedalam sebuah *tilemap* berbentuk *array* dua dimensi, setiap *tile* memiliki koordinat X dan Y dan dapat

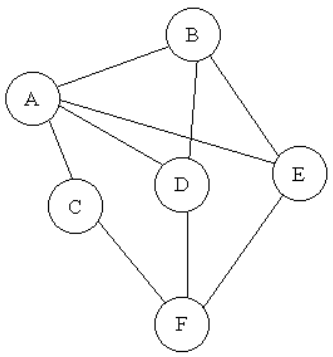
digambarkan melalui struktur data *graph*. Struktur data *graph* adalah struktur data yang berbentuk jaringan, hubungan antar elemennya adalah *many- to-many*. Berikut ini adalah contoh arena permainan yang terdapat pada *game Save Our Forest* dapat dilihat pada gambar III.3

	0	1	2
0	(0,0)	(25,0)	(50,0)
1	(0,25)	(25,25)	(50,25)
2	(0,50)	(25,50)	(50,50)



Gambar III.3 Arena Permainan *Game Save Our Forest*

Secara umum terdapat dua macam representasi dari struktur data graf yang dapat diimplementasikan. Pertama, disebut *adjacency list*, yaitu *list* yang diimplementasikan dengan menampilkan masing-masing simpul sebagai sebuah struktur data yang mengandung senarai dari semua simpul yang saling berhubungan. Yang kedua adalah representasi berupa *adjacency matrix* dimana baris dan kolom dari matriks tersebut merepresentasikan simpul awal dan simpul tujuan dan sebuah entri didalam senarai yang menyatakan apakah terdapat sisi di antara kedua simpul tersebut. Dalam pembangungan *game Save Our Forest* akan digunakan representasi berupa *adjacency list*. Berikut ini adalah contoh *adjacency list* dapat dilihat pada gambar III.4




```
A  B C D E
B  A D E
C  A F
D  A B F
E  A B F
F  C D E
```

Gambar III.4 Contoh *Adjacency List*

Pada Algoritma A* terdapat beberapa prosedur didalamnya, Berikut ini adalah kamus global algoritma A*:

{Kamus Global}
openList : linked list
closedList : linked list
path : boolean
directPath : double
actualPath : double
F : double
closed : Boolean

Berikut ini adalah *pseudocode* pembuatan *node* dapat dilihat pada tabel III.2

Tabel III.2 *Pseudocode* Prosedur *Node*

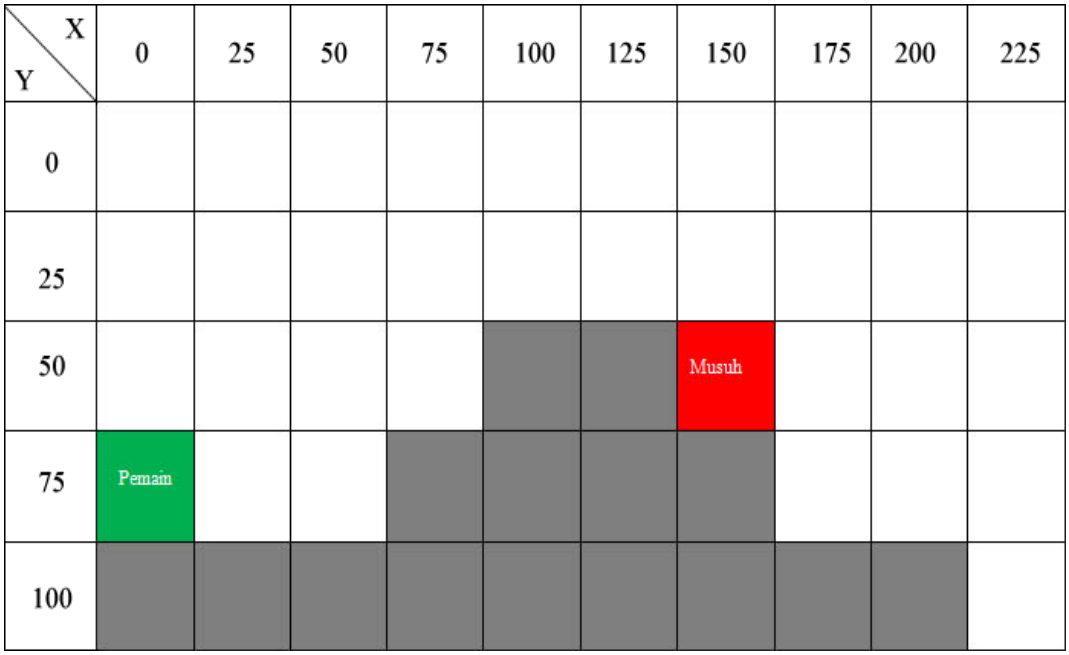
	Isi Pseudocode
1	Procedure Node (input : x, y, cost : integer)
2	{I.S : koordinat node sudah terdefinisi}
3	{F.S : menghasilkan node}
4	
5	Kamus :
6	xCoor : integer
7	yCoor : integer
8	nodeCost : integer
9	examinedNode : integer
10	neighbor : List
11	directPath : double
12	actualPath : double
13	F : double
14	
15	Algoritma :
16	xCoor : x
17	yCoor : y
18	nodeCost : cost
19	directPath = 2*(Math.sqrt(Math.pow((xCoor-
20	(world.getTargetXcoor())), 2) + Math.pow((yCoor-
21	(world.getTargetYcoor())), 2)))
22	actualPath = 2*(Math.sqrt(Math.pow((xCoor-
23	(world.getStartXcoor())), 2) + Math.pow((yCoor-
24	(world.getStartYcoor())), 2)))
25	F = directPath + actualPath + cost

Berikut ini adalah *pseudocode* pembangkitan suksesor dapat dilihat pada tabel III.3

Tabel III.3 *Pseudocode* Prosedur *checkNode*

	Isi Pseudocode
1	`Procedure checkNode`
2	`{I.S : Node sudah terdefinisi}`
3	`{F.S : Membangkitkan node sekitar}`
4	
5	`Kamus :`
6	`  Path : boolean`
7	`  openList : Linkedlist`
8	`  closedList : Linkedlist`
9	`  fls : Linkedlist`
10	`  tryNum : integer`
11	
12	`Algoritma :`
13	`  iter = openList.iterator()`
14	`  lastCheckedF : double`
15	`  while (iter.hasNext())`
16	`  {`
17	`    lastCheckNode = iter.next()`
18	`    lastCheckedF = lastCheckedNode.F;`
19	`    if(lastCheckedF <= smallestFfound) {F lebih kecil}`
20	`    {`
21	`      If(!closeList.contains(lastCheckedNode)){`
22	`      {tidak berada di close list}`
23	`        smallestFfound = lastCheckedF`
24	`      }`
25	`  }`

Pada *game Save Our Forest*, arena permainan dibagi kedalam *tilemap* yang terdiri dari *node* dengan orde X dan Y. tiap *node* memiliki koordinat X dan Y atau berbentuk *array* dua dimensi. Berikut ini adalah contoh kasus algoritma A* pada *game Save Our Forest* dapat dilihat pada gambar III.5



Gambar III.4 Contoh Kasus Algoritma A* pada *game Save Our Forest*

Tile atau *node* tujuan atau pada kasus ini disebut *node* pemain direpresentasikan dengan warna hijau dan berada di koordinat (0,75) sedangkan *node* awal atau pada kasus ini disebut musuh direpresentasikan dengan warna merah dan berada di *node* (150,50), sedangkan *node* abu-abu merupakan penghalang atau dinding. Musuh dapat bergerak ke semua *node* yang berwarna putih tetapi musuh tidak dapat bergerak ke arah *node* yang berwarna abu-abu. Sedangkan untuk penentuan nilai heuristik dapat dilihat di gambar III.5

X / Y	0	25	50	75	100	125	150	175	200	225
0	H=75	H=100	H=125	H=150	H=175	H=200	H=225	H=250	H=275	H=300
25	H=50	H=75	H=100	H=125	H=150	H=175	H=200	H=225	H=250	H=275
50	H=25	H=50	H=75	H=100			Musuh	H=200	H=225	H=250
75	Pemain	H=25	H=50						H=200	H=225
100										H=200

Gambar III.5 Penentuan nilai heuristik

Nilai untuk pergerakan secara horizontal dan vertikal adalah 25, sedangkan untuk nilai untuk pergerakan diagonal adalah 35. Untuk mendapatkan nilai heuristik menggunakan *diagonal distance* yang memungkinkan pergerakan ke 8 arah maka digunakan notasi matematika

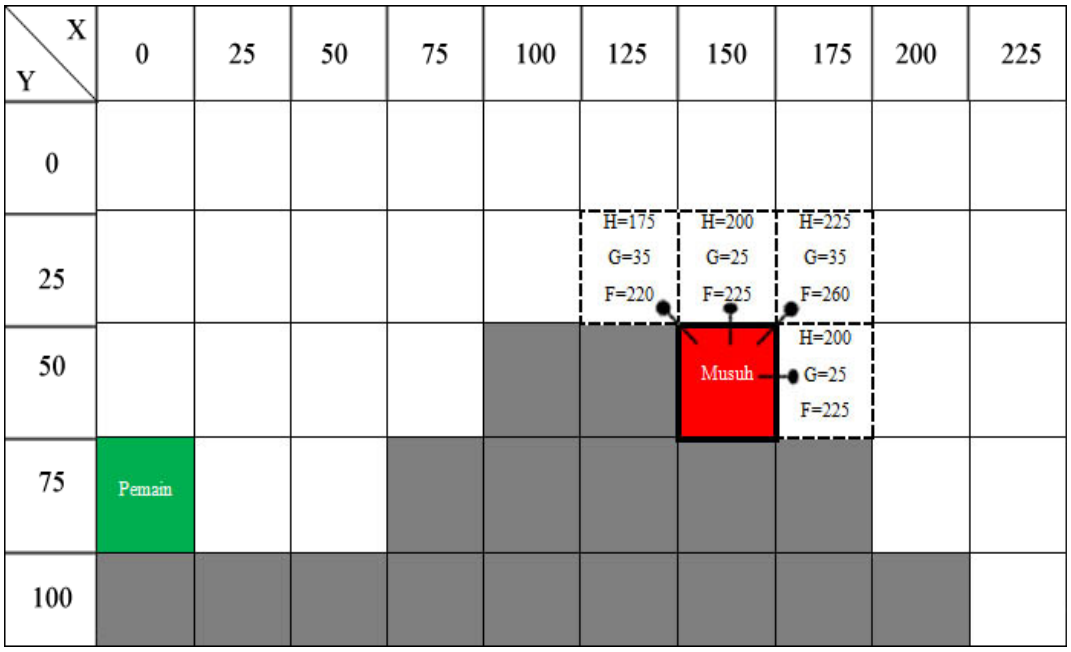
H diagonal(n) = (abs(n.x-goal.x) + abs(n.y-goal.y)) **(III.1)**

Selanjutnya *node* musuh akan melakukan pencarian untuk menemukan jalur terpendek dengan cara memeriksa *node* disekitarnya. Langkah-langkah pencarian jalur terpendek adalah sebagai berikut:

1. *Node* musuh dimasukkan ke dalam *open list*.
2. Periksa semua *node* yang berdekatan dengan *node* musuh apakah *node* tersebut merupakan dinding atau tidak, abaikan *node* yang tidak bisa dilalui, lalu masukkan *node* tersebut kedalam *open list*. Dan untuk setiap *node* baru yang ditambahkan kedalam *open list*, simpan *node* lama sebagai "*parent*" yang akan digunakan untuk menelusuri jalan.
3. *Node* dengan nilai f terkecil akan terpilih sebagai *best node* dan selanjutnya pencarian akan dimulai dari *node* ini.

4. Hapus *node* musuh atau *node* sebelumnya dari *open list* dan masukkan ke dalam *closed list*.

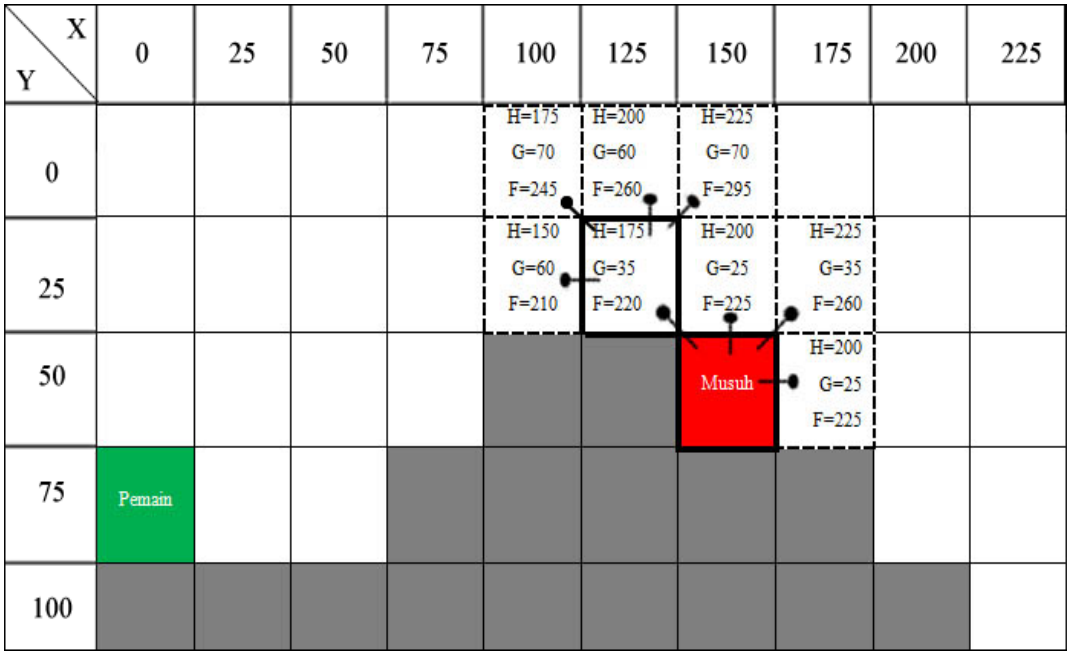
Berikut ini adalah gambar membangkitkan suksesor *node* musuh dapat dilihat di gambar III.6



Gambar III.6 Pembangkitan *node* suksesor musuh

Node dengan garis putus-putus merupakan *node* yang sudah dimasukkan kedalam *open list* sedangkan, *node* dengan garis tegas merupakan *node* yang sudah dimasukkan kedalam *closed list*. Masing-masing *node* yang berada di *open list* memiliki *pointer* yang mengarah ke *parent*nya masing-masing.

Langkah pertama, *node* Musuh membangkitkan *node* di sekitarnya dan hasilnya membangkitkan 4 *node*, baik arah horizontal, vertical, maupun diagonal. *Node* pertama dengan koordinat (125,25) memiliki nilai g = 35, h = 175 sehingga f = g + h = 220. *Node* kedua dengan koordinat (150,25) memiliki nilai g = 25, h = 200 sehingga f = g + h = 225. *Node* ketiga dengan koordinat (175,25) memiliki nilai g = 35, h = 225 sehingga f = g + h = 260. *Node* keempat dengan koordinat (175,25) memiliki nilai g = 25, nilai h = 200, sehingga f = g + h = 225. Berikut ini adalah langkah pertama pencarian rute terpendek menggunakan algoritma A* dapat dilihat di gambar III.7



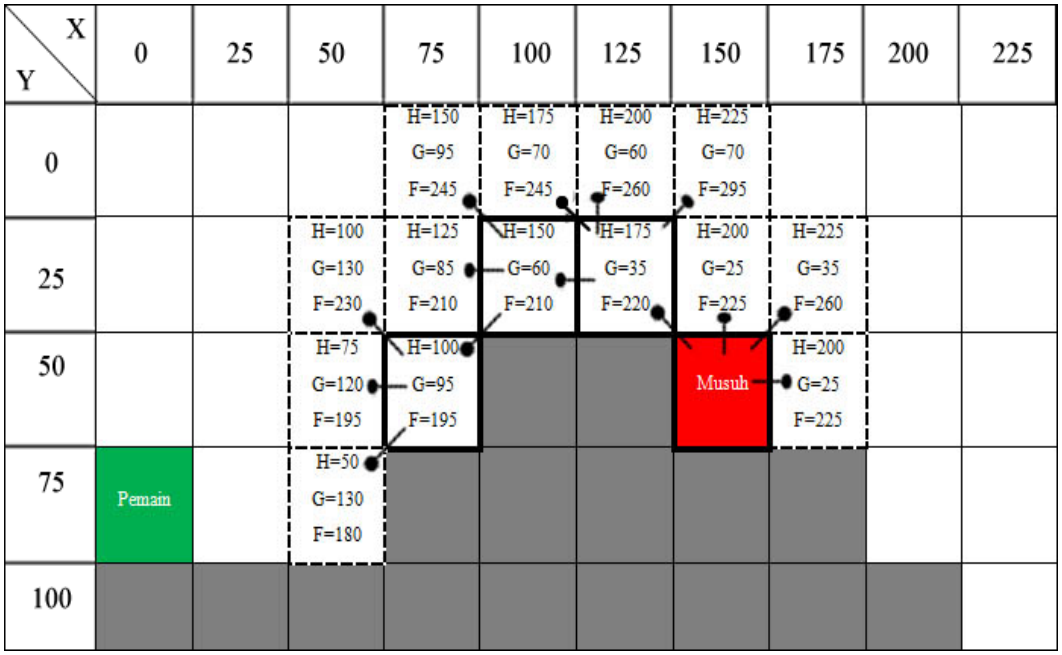
Gambar III.7 Langkah 1 pencarian rute terpendek menggunakan algoritma A* dalam *Tile Map*

Karena *node* dengan koordinat (125,25) memiliki biaya terkecil (f=220) terpilih sebagai *BestNode* dan pencarian selanjutnya dimulai dari *node* ini. Pencarian dimulai dengan menelusuri *node* pertama dengan koordinat (100,25) memiliki nilai g = 60, angka ini didapat berdasarkan g *node* sebelumnya ditambah 25, h = 150, sehingga f = g + h = 210. *Node* kedua dengan koordinat (100,0) memiliki nilai g = 70, angka ini didapat berdasarkan g *node* sebelumnya ditambah 35, h = 70, sehingga f = g + h = 245. *Node* ketiga dengan koordinat (125,0) memiliki nilai g = 60, angka ini didapat berdasarkan g *node* sebelumnya ditambah 25, h = 200, sehingga f = g + h = 260. *Node* keempat dengan koordinat (150,0) memiliki nilai g = 70, angka ini didapat berdasarkan g *node* sebelumnya ditambah 35, h = 225, sehingga f = g + h = 295. Berikut ini adalah langkah kedua pencarian rute terpendek menggunakan algoritma A* dapat dilihat di gambar III.8



Gambar III.8 Langkah 2 pencarian rute terpendek menggunakan algoritma A* dalam *Tile Map*

Karena *node* dengan arah horizontal kiri memiliki biaya terkecil (f=210) terpilih sebagai *BestNode* dan pencarian selanjutnya dimulai dari *node* ini. Pencarian dimulai dengan menelusuri *node* pertama dengan koordinat (75,50) memiliki nilai g = 95, angka ini didapat berdasarkan g *node* sebelumnya ditambah 35, h = 100, sehingga f = g + h = 195. *Node* kedua dengan koordinat (75,25) memiliki nilai g = 85, angka ini didapat berdasarkan g *node* sebelumnya ditambah 25, h = 125, sehingga f = g + h = 210. *Node* ketiga dengan koordinat (75,0) memiliki nilai g = 95, angka ini didapat berdasarkan g *node* sebelumnya ditambah 25, h = 150, sehingga f = g + h = 245. Berikut ini adalah langkah ketiga pencarian rute terpendek menggunakan algoritma A* dapat dilihat di gambar III.9



Gambar III.9 Langkah 3 pencarian rute terpendek menggunakan algoritma A* dalam *Tile Map*

Karena node dengan koordinat (75,50) memiliki biaya terkecil (f=195) terpilih sebagai *BestNode* dan pencarian selanjutnya dimulai dari *node* ini. Pencarian dimulai dengan menelusuri *node* pertama dengan arah koordinat (50,25) memiliki nilai g = 130, angka ini didapat berdasarkan g *node* sebelumnya ditambah 35, h = 100, sehingga f = g + h = 230. *Node* kedua dengan koordinat (50,50) memiliki nilai g = 120, angka ini didapat berdasarkan g *node* sebelumnya ditambah 25, h = 75, sehingga f = g + h = 195. *Node* ketiga dengan arah koordinat (50,75) memiliki nilai g = 130, angka ini didapat berdasarkan g *node* sebelumnya ditambah 35, h = 50, sehingga f = g + h = 180. Berikut ini adalah langkah keempat pencarian rute terpendek menggunakan algoritma A* dapat dilihat di gambar III.10



Gambar III.10 Langkah 4 pencarian rute terpendek menggunakan algoritma A* dalam *Tile Map*

Karena node dengan koordinat (50,75) memiliki biaya terkecil (f=180) terpilih sebagai *BestNode* dan pencarian selanjutnya dimulai dari *node* ini. Pencarian dimulai dengan menelusuri *node* pertama dengan koordinat (25,50) memiliki nilai g = 165, angka ini didapat berdasarkan g *node* sebelumnya ditambah 35, h = 50, sehingga f = g + h = 215. *Node* kedua dengan koordinat (25,75) memiliki nilai g = 155, angka ini didapat berdasarkan g *node* sebelumnya ditambah 25, h = 25, sehingga f = g + h = 180. Berikut ini adalah langkah kelima pencarian rute terpendek menggunakan algoritma A* dapat dilihat di gambar III.11



Gambar III.11 Langkah 5 pencarian rute terpendek menggunakan algoritma A* dalam *Tile Map*

Karena node dengan arah horizontal kiri memiliki biaya terkecil (f=180) terpilih sebagai *BestNode* dan pencarian selanjutnya dimulai dari *node* ini. Pencarian dimulai dengan menelusuri *node* pertama dengan koordinat (0,50) memiliki nilai g = 190, angka ini didapat berdasarkan g *node* sebelumnya ditambah 35, h = 25, sehingga f = g + h = 215. *Node* kedua dengan koordinat (0,75) memiliki nilai g = 180, angka ini didapat berdasarkan g *node* sebelumnya ditambah 25, h = 0, sehingga f = g + h = 180. Berikut ini adalah langkah keenam pencarian rute terpendek menggunakan algoritma A* dapat dilihat di gambar III.12



Gambar III.12 Langkah 6 pencarian rute terpendek menggunakan algoritma A* dalam *Tile Map*

Pencarian rute terpendek dari *node* musuh menuju *node* pemain menggunakan algoritma A* menghasilkan rute sebagai berikut:

1. Dari *node* musuh bergerak ke koordinat (125,25) dengan f = 220.
2. Dari *node* dengan f = 84 bergerak ke koordinat (100,25) dengan f = 210.
3. Dari *node* dengan f = 84 bergerak ke koordinat (75,50) dengan f = 195.
4. Dari *node* dengan f = 78 bergerak ke koordinat (50,75) bawah dengan f = 180.
5. Dari *node* dengan f = 72 bergerak ke koordinat (25,75) kiri dengan f = 180.
6. Dari *node* dengan f = 180 bergerak ke koordinat (0,75) dan menemukan *goal* (*node* pemain).

III.1.5 Analisis Kebutuhan Non Fungsional

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

III.1.5.1 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 sistem di dalamnya. Pada aplikasi *game* ini, perangkat keras yang digunakan adalah sebagai berikut :

1. Prosesor 1.00 Ghz
2. RAM 512 MB
3. VGA 64 MB
4. Hard Disk 10 GB
5. Monitor dengan resolusi minimal 800 x 600
6. Speaker atau Headset
7. Mouse
8. Keyboard

III.1.5.2 Analisis Kebutuhan Perangkat Lunak

Perangkat lunak digunakan dalam sebuah sistem merupakan perintah - perintah yang diberikan kepada perangkat keras agar bisa saling berinteraksi diantara keduanya. Perangkat lunak yang dibutuhkan untuk membangun aplikasi *game Save Our Forest* ini adalah sebagai berikut :

1. Sistem Operasi XP/Vista/7
2. Greenfoot versi 2.2.1
3. jdk-1_5_0_07-windows-i586-p ke atas (JDK 1.5 minimal).
4. CorelDRAW X4

Adapun perangkat lunak yang dibutuhkan dalam penerapan aplikasi *game* edukasi ini adalah sebagai berikut :

1. Sistem Operasi XP/Vista/7
2. jdk-1_5_0_07-windows-i586-p ke atas (JDK 1.5 minimal).

III.1.5.3 Analisis Pengguna

Game yang akan dibuat ditujukan khusus untuk anak-anak berusia 13 tahun ke atas, dan pengguna umum yaitu publik yang dapat mengerti dan memahami komputer sehingga dapat menggunakan aplikasi yang akan dibangun. Alasan pemilihan usia ini didasarkan pada rating *game* dari ESRB (*Entertainment Software Rating Board*) dikarenakan game ini berusaha menyisipkan pesan-pesan penyebab kerusakan hutan dan dampaknya terhadap lingkungan sekitar, tetapi melalui cara belajar sambil bermain, usia ini juga merupakan usia yang efektif untuk mempelajari berbagai hal dan usia yang kritis dalam hal tumbuh kembang fisik, mental dan psikososial.

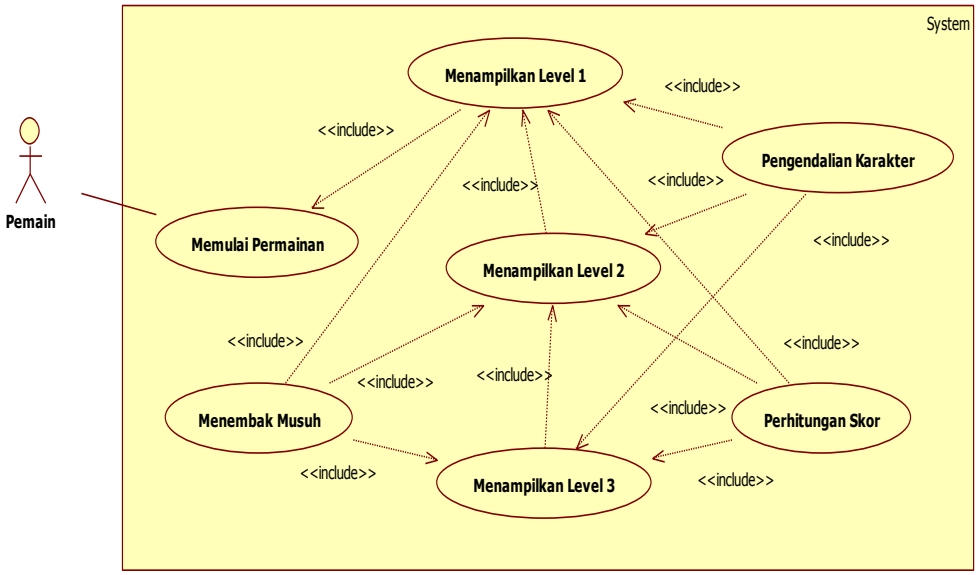
III.1.6 Analisis Kebutuhan Fungsional

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

III.1.6.1 *Use Case* Diagram

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

Berikut ini adalah perancangan proses-proses yang terdapat pada *game Save Our Forest* dapat dilihat pada gambar III.13



Gambar III.13 *Use Case* Diagram *Game*

III.1.6.2 Definisi Aktor

Definisi aktor berfungsi untuk menjelaskan peranan aktor dalam pengoperasian *game*. Dalam *game* ini, hanya terdapat seorang aktor yaitu pemain. Berikut ini adalah definisi dari aktor yang ada dalam perancangan *game*.

Tabel III.3 Definisi Aktor

No	Aktor	Deskripsi
1.	Pemain	Pemain merupakan orang yang memainkan *game* dan berinteraksi langsung dengan antar muka *game.*

III.1.6.3 Definisi *Use Case*

Definisi *Use Case* berfungsi untuk menjelaskan proses yang terdapat pada setiap *use case.* Definisi *Use Case* dapat dilihat di tabel III.3

Tabel III.4 Definisi *Use Case*

No	*Use Case*	Deskripsi
1.	Memulai permainan	Proses untuk memulai permainan
2.	Menampilkan *Level* 1	Proses untuk menampilkan *Level* 1
3.	Menampilkan *Level* 2	Proses untuk menampilkan *Level* 2
4.	Menampilkan *Level* 3	Proses untuk menampilkan *Level* 3
5.	Perhitungan skor	Proses pada permainan untuk menghitung skor yang didapatkan oleh pemain.
6.	Pengendalian karakter	Proses yang dilakukan pemain untuk menggerakkan karakter.
7.	Menembak musuh	Proses yang dilakukan oleh pemain untuk menembak musuh.

III.1.6.4 Skenario *Use Case*

Skenario *Use Case* menggambarkan alur penggunaan sistem dimana setiap skenario digambarkan dari sudut pandang aktor, seseorang atau piranti yang berinteraksi dengan perangkat lunak dalam berbagai cara.

1. Skenario *Use Case* Memulai Permainan

Skenario *use case* ini menjelaskan interaksi antara aktor yaitu pemain dengan *use case* memulai permainan yang akan dijelaskan pada tabel III.5

Tabel III.5 *Use Case Scenario* Memulai Permainan

Identifikasi Masalah	
Nama	Memulai Permainan
Tujuan	Masuk ke arena permainan dan memulai permainan
Deskripsi	Proses yang dilakukan oleh pemain untuk memulai permainan
Aktor	Pemain
Skenario Utama	
Kondisi Awal	Tampilkan menu utama pada *game*
Aksi Aktor	**Reaksi Sistem**
	1. Menampilkan menu utama
2. Memilih menu Mulai game	
	3. Menampilkan intro
Kondisi Akhir	Sistem menampilkan intro

2. Skenario *Use Case* Menampilkan *Level* 1

Skenario *use case* ini menjelaskan interaksi antara aktor yaitu pemain dengan *use case* memulai permainan yang akan dijelaskan pada tabel III.6

Tabel III.6 *Use Case Scenario* Menampilkan *Level* 1

Identifikasi Masalah	
Nama	Menampilkan *Level* 1
Tujuan	Menampilkan *Level* 1
Deskripsi	Proses untuk menampilkan arena permainan di *level* 1
Aktor	Pemain
Skenario Utama	
Kondisi Awal	*Level* 1
Aksi Aktor	**Reaksi Sistem**

	1. Menampilkan arena permainan
	2. Menempatkan *player* di arena permainan
	3. Menempatkan musuh di arena permainan
	4. Inisialisasi informasi permainan
5. Menyelesaikan permainan	
	6. Berpindah ke *level* selanjutnya
Skenario Alternatif	
	1. Menampilkan arena permainan
	2. Menempatkan *player* di arena permainan
	3. Menempatkan musuh di arena permainan
	4. Inisialisasi informasi permainan
5. Permainan tidak selesai	
	6. Kembali ke level 1

3. Skenario *Use Case* Menampilkan *Level* 2

Skenario *use case* ini menjelaskan interaksi antara aktor yaitu pemain dengan *use case* memulai permainan yang akan dijelaskan pada tabel III.7

Tabel III.7 *Use Case Scenario* Menampilkan *Level* 2

Identifikasi Masalah	
Nama	Menampilkan *Level* 2
Tujuan	Menampilkan *Level* 2
Deskripsi	Proses untuk menampilkan arena permainan di *level* 2
Aktor	Pemain
Skenario Utama	

Kondisi Awal	*Level* 2
Aksi Aktor	**Reaksi Sistem**
	1. Menampilkan arena permainan
	2. Menempatkan *player* di arena permainan
	3. Menempatkan musuh di arena permainan
	4. Inisialisasi informasi permainan
5. Menyelesaikan permainan	
	6. Berpindah ke *level* selanjutnya
Skenario Alternatif	
	1. Menampilkan arena permainan
	2. Menempatkan *player* di arena permainan
	3. Menempatkan musuh di arena permainan
	4. Inisialisasi informasi permainan
5. Permainan tidak selesai	
	6. Kembali ke level 2

4. Skenario *Use Case* Menampilkan *Level* 3

Skenario *use case* ini menjelaskan interaksi antara aktor yaitu pemain dengan *use case* memulai permainan yang akan dijelaskan pada tabel III.8

Tabel III.8 *Use Case Scenario* Menampilkan *Level* 3

Identifikasi Masalah	
Nama	Menampilkan *Level* 3
Tujuan	Menampilkan *Level* 3
Deskripsi	Proses untuk menampilkan arena permainan di *level* 3

Aktor	Pemain
Skenario Utama	
Kondisi Awal	*Level* 3
Aksi Aktor	**Reaksi Sistem**
	1. Menampilkan arena permainan
	2. Menempatkan *player* di arena permainan
	3. Menempatkan musuh di arena permainan
	4. Inisialisasi informasi permainan
5. Menyelesaikan permainan	
	6. Melawan *Boss* terakhir
7. Mengalahkan *boss* terakhir	
	8. Menampilkan narasi penutup
Skenario Alternatif	
	1. Menampilkan arena permainan
	2. Menempatkan *player* di arena permainan
	3. Menempatkan musuh di arena permainan
	4. Inisialisasi informasi permainan
5. Permainan tidak selesai	
	6. Kembali ke level 3

5. Skenario *Use Case* Perhitungan Skor

Skenario *use case* ini menjelaskan interaksi antara aktor yaitu pemain dengan *use case* memulai permainan yang akan dijelaskan pada tabel III.9

Tabel III.9 *Use Case Scenario* Perhitungan Skor

Identifikasi Masalah	
Nama	Perhitungan Skor
Tujuan	Proses untuk menghitung skor
Deskripsi	Proses yang akan menginformasikan skor yang didapatkan oleh pemain.
Aktor	Pemain
Skenario Utama	
Kondisi Awal	Pemain berada di area permainan
Aksi Aktor	**Reaksi Sistem**
1.Pemain mengenai kantung bibit	
	2. Objek kantung bibit tersentuh pemain
Kondisi akhir	3. Skor pemain bertambah
Skenario Alternatif	
Kondisi Awal	Pemain berada di area permainan
Aksi Aktor	**Reaksi Sistem**
1.Pemain tidak mengenai kantung bibit	
Kondisi Akhir	2. Skor pemain tetap

6. Skenario *Use Case* Pengendalian Karakter

Skenario *use case* ini menjelaskan interaksi antara aktor yaitu pemain dengan *use case* memulai permainan yang akan dijelaskan pada tabel III.10

Tabel III.10 *Use Case Scenario* Pengendalian Karakter

Identifikasi Masalah	
Nama	Pengendalian karakter
Tujuan	Mengendalikan karakter pemain
Deskripsi	Proses yang dilakukan oleh pemain untuk menggerakkan karakter
Aktor	Pemain

Skenario Utama	
Kondisi Awal	Tampilan menu utama *game*
Aksi Aktor	**Reaksi Sistem**
1. Menekan tombol A di *keyboard*	
	2. Menggerakkan karakter pemain ke kiri
3. Menekan tombol D di *keyboard*	
	4. Menggerakkan karakter pemain ke kanan
5. Menekan tombol W di *keyboard*	
	6. Karakter melompat
Kondisi akhir	Karakter bergerak sesuai *input* pemain.

7. Skenario *Use Case* Menembak musuh

Skenario *use case* ini menjelaskan interaksi antara aktor yaitu pemain dengan *use case* memulai permainan yang akan dijelaskan pada tabel III.11

Tabel III.11 *Use Case Scenario* Menembak musuh

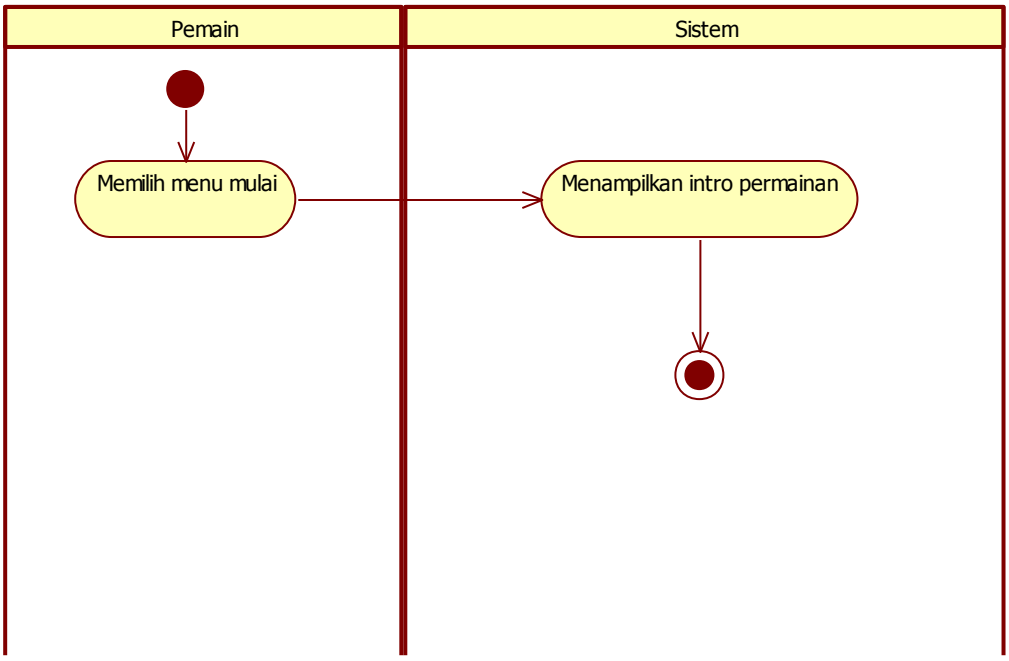
Identifikasi Masalah	
Nama	Menembak musuh
Tujuan	Menembak musuh
Deskripsi	Menembakkan peluru untuk menyerang musuh.
Aktor	Pemain
Skenario Utama	
Kondisi Awal	Permainan telah dimulai
Aksi Aktor	**Reaksi Sistem**
1. Menekan tombol f di *keyboard*	
	2. Mengeluarkan peluru
Kondisi akhir	Karakter menembakkan peluru

III.1.6.5 *Acitivity* Diagram

Activity Diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis.

A. *Activity* Diagram Memulai Permainan

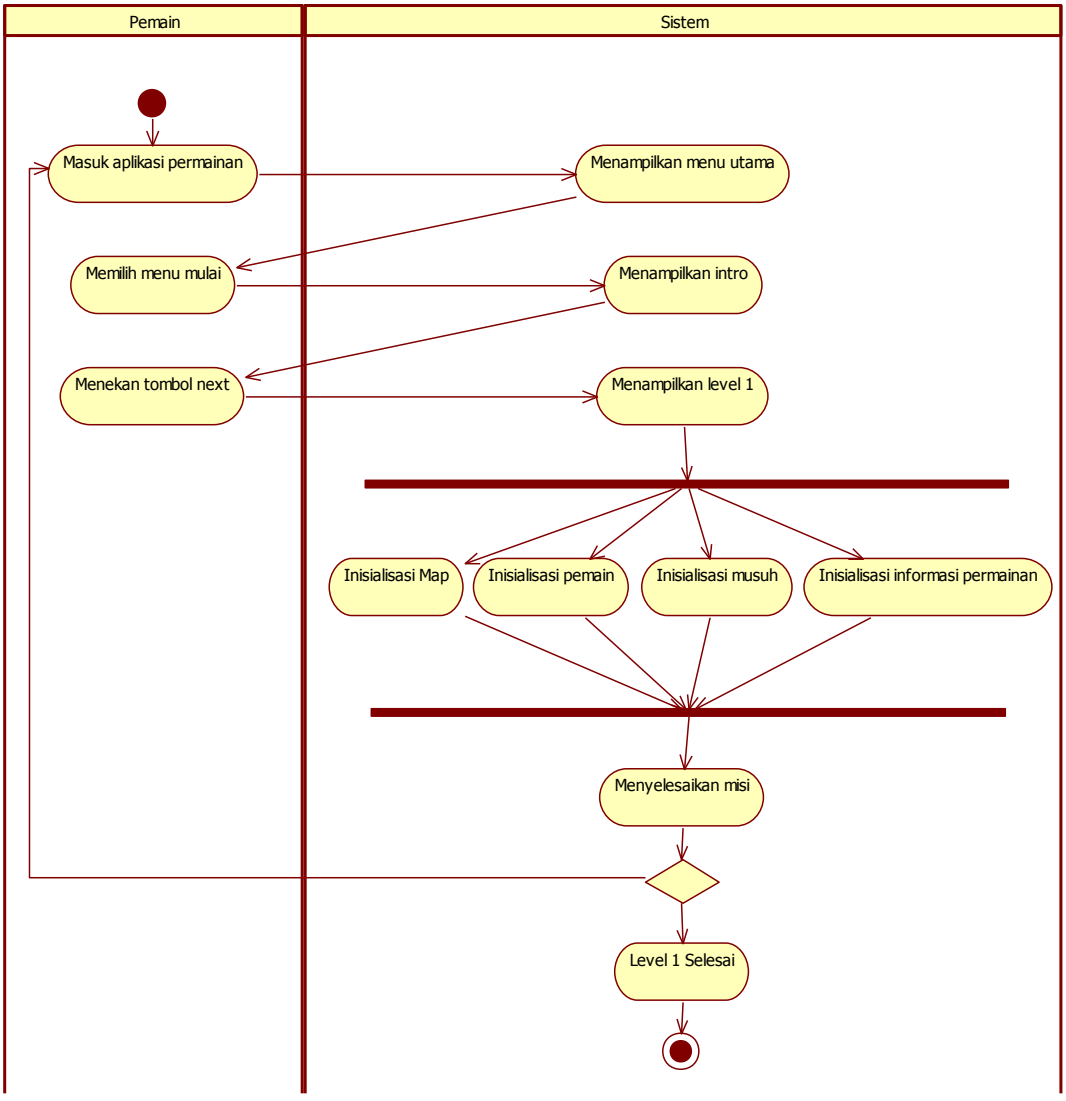
Aktivitas ini terjadi saat permainan dijalankan, sistem akan menampilkan *intro* permainan. Diagram *Activity* memulai permainan dapat dilihat pada gambar III.14



Gambar III.14 *Activity* Diagram Memulai Permainan

B. *Activity* Diagram Menampilkan *Level* 1

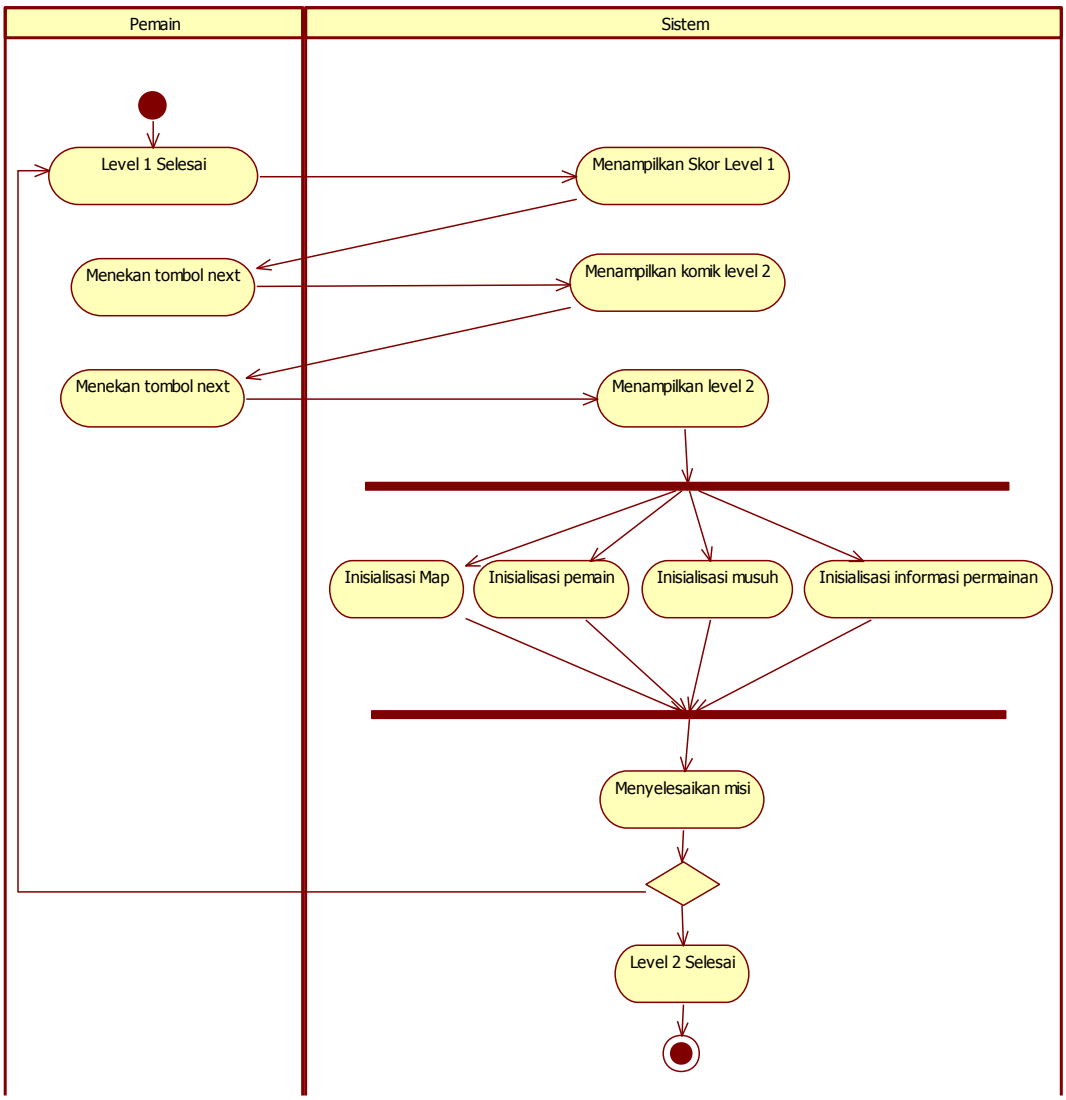
Aktivitas ini terjadi setelah menu mulai dijalankan, sistem akan menampilkan arena permainan *level* 1. Apabila misi permainan tidak terselesaikan maka pemain akan memulai permainan dari *level* 1. Diagram *Activity* menampilkan *level* 1 dapat dilihat pada gambar III.15



Gambar III.15 *Activity* Diagram Menampilkan *Level* 1

C. *Activity* Diagram Menampilkan *Level* 2

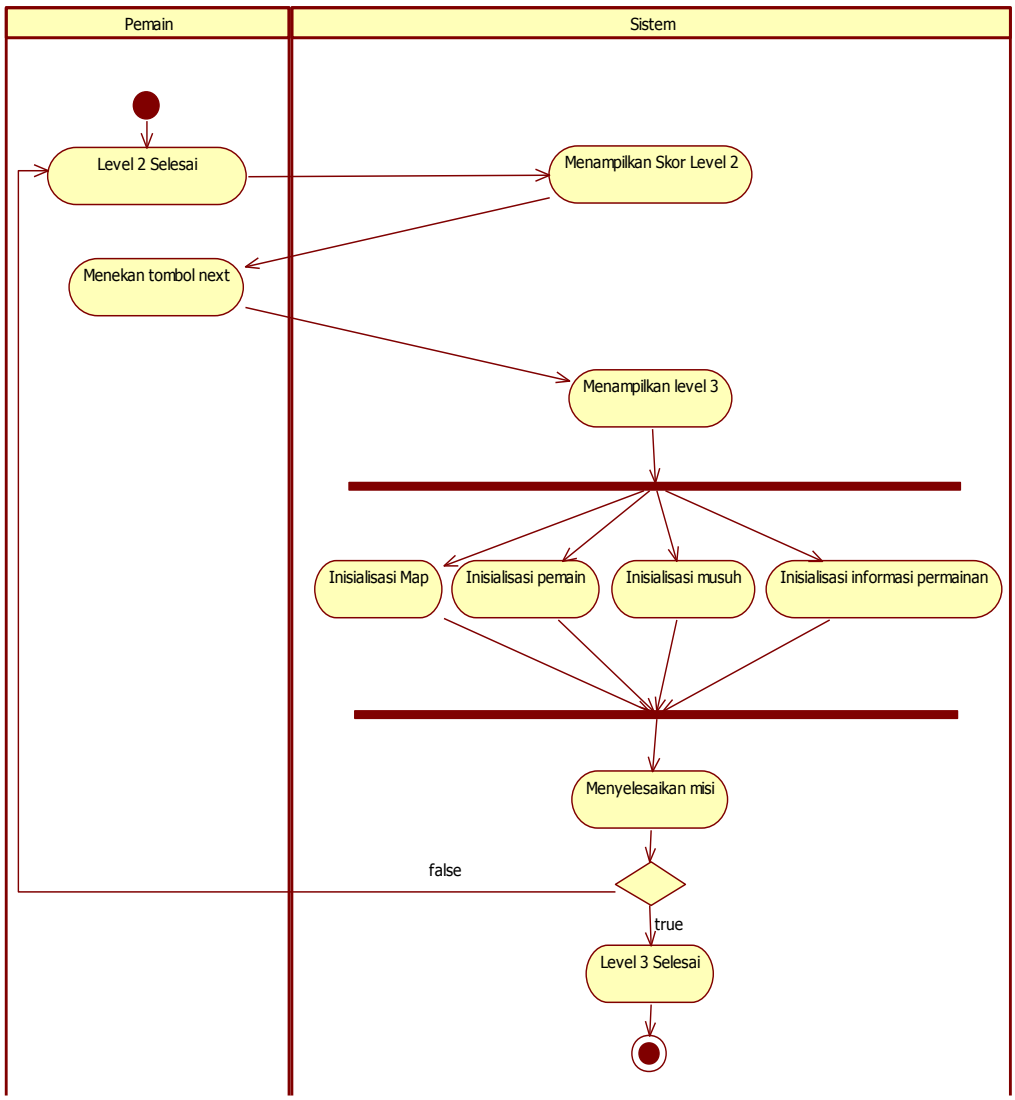
Aktivitas ini terjadi setelah *level* 1 selesai, sistem akan menampilkan arena permainan *level* 2. Apabila misi permainan tidak terselesaikan maka pemain akan memulai permainan dari *level* 2. Diagram *Activity* menampilkan *level* 2 dapat dilihat pada gambar III.16



Gambar III.16 *Activity* Diagram Menampilkan *Level* 2

D. *Activity* Diagram Menampilkan *Level* 3

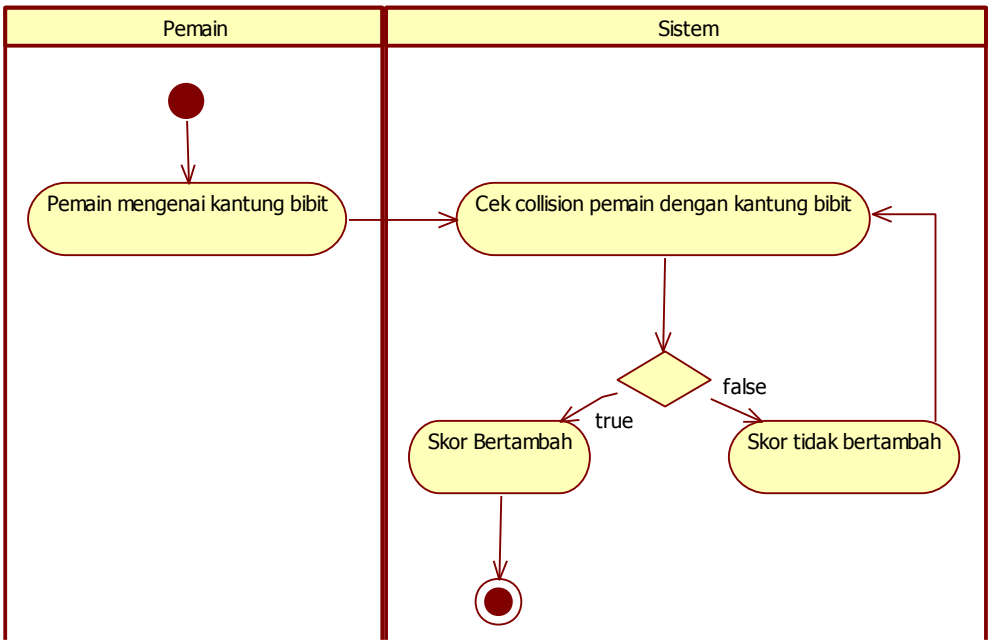
Aktivitas ini terjadi setelah *level* 2 selesai, sistem akan menampilkan arena permainan *level* 3. Apabila misi permainan tidak terselesaikan maka pemain akan memulai permainan dari *level* 3. Diagram *Activity* menampilkan *level* 3 dapat dilihat pada gambar III.17



Gambar III.17 *Activity* Diagram Menampilkan *Level* 3

E. *Activity* Diagram Perhitungan Skor

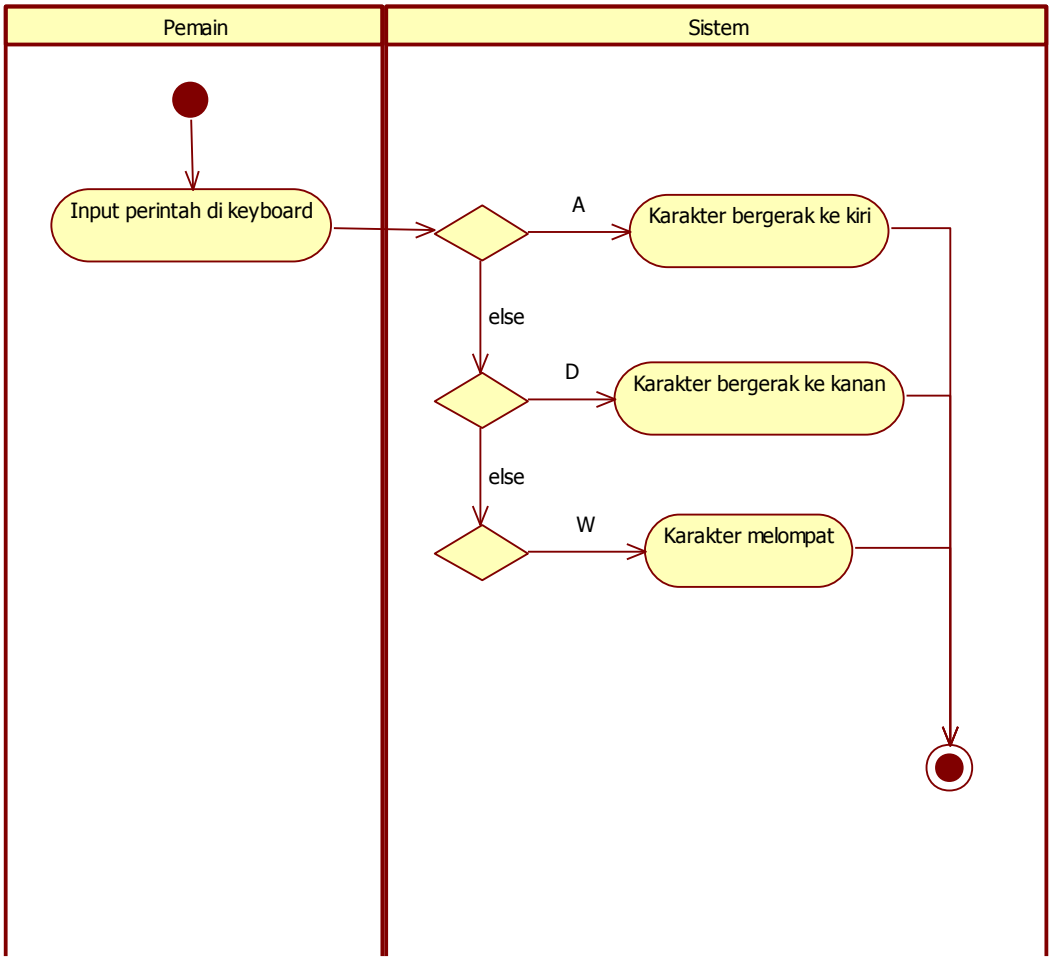
Aktivitas ini terjadi di *level* 1-3, proses ini terjadi ketika pemain mengenai kantung bibit. Diagram *Activity* perhitungan skor dapat dilihat pada gambar III.18



Gambar III.18 *Activity* Diagram Perhitungan Skor

F. *Activity* Diagram Pengendalian Karakter

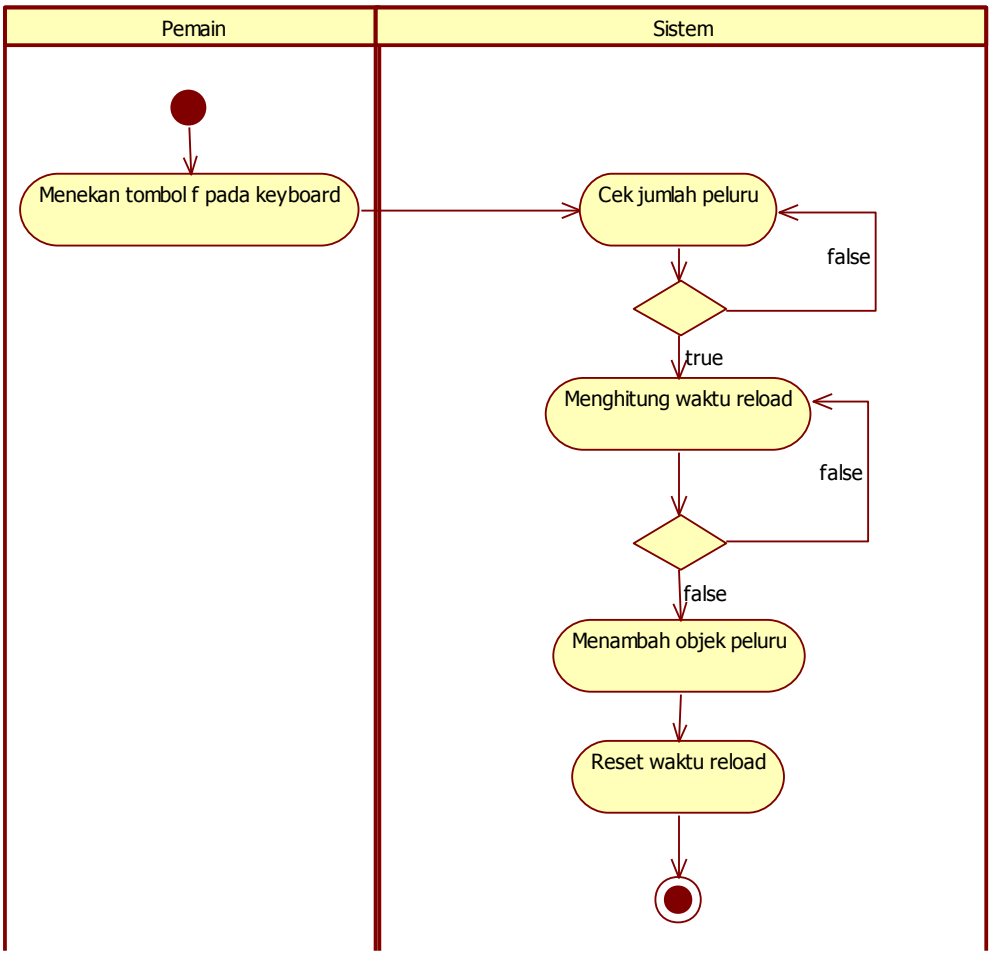
Aktivitas ini terjadi di *level* 1-3, sistem akan merespon *input* dari *keyboard* pemain. Diagram *Activity* pengendalian karakter dapat dilihat pada gambar III.19



Gambar III.19 *Activity* Diagram Pengendalian Karakter

G. *Activity* Diagram Menembak peluru

Aktivitas ini terjadi di *level* 1-3, proses ini terjadi ketika pemain menembakkan peluru. Diagram *Activity* menembak peluru dapat dilihat pada gambar III.20



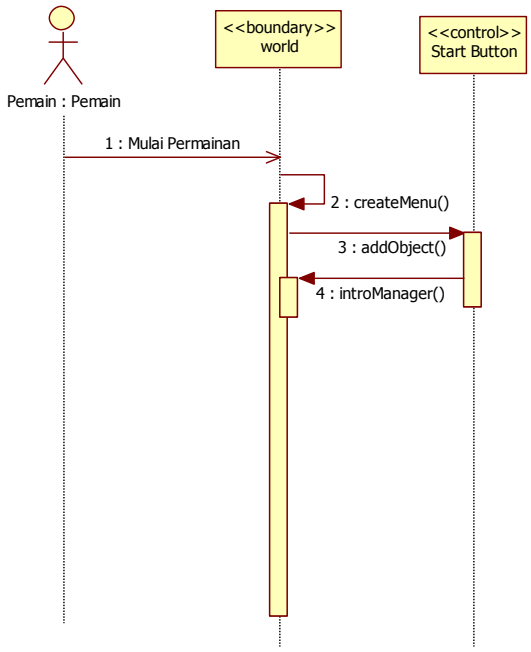
Gambar III.20 Activity Diagram Menembak Peluru

III.1.6.6 *Sequence* Diagram

Sequence Diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antarobjek.

A. *Sequence* Diagram Memulai Permainan

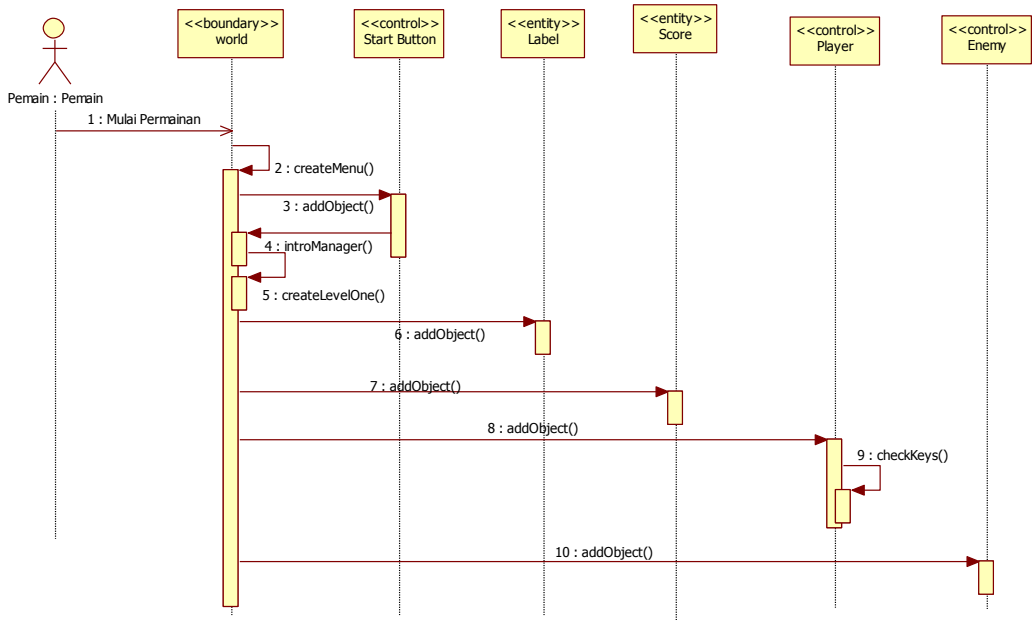
Sequence diagram memulai permainan dapat dilihat pada gambar III.21



Gambar III.21 *Sequence* Diagram Memulai Permainan

B. *Sequence* Diagram Menampilkan *Level* 1

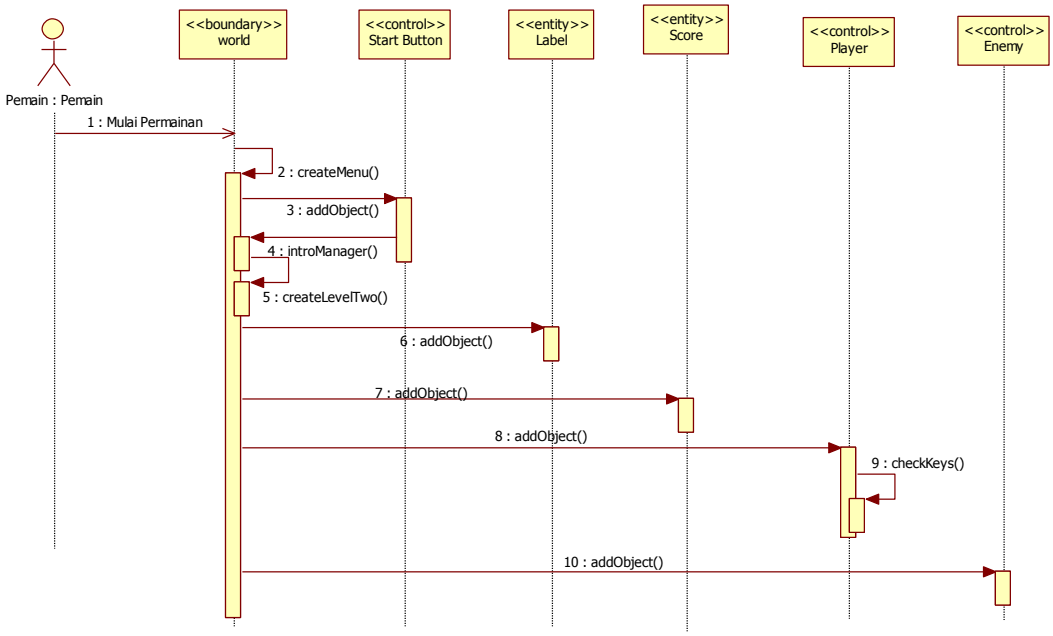
Sequence diagram menampilkan *level* 1 dapat dilihat pada gambar III.22



Gambar III.22 *Sequence* Diagram Menampilkan *Level* 1

C. *Sequence* Diagram Menampilkan *Level* 2

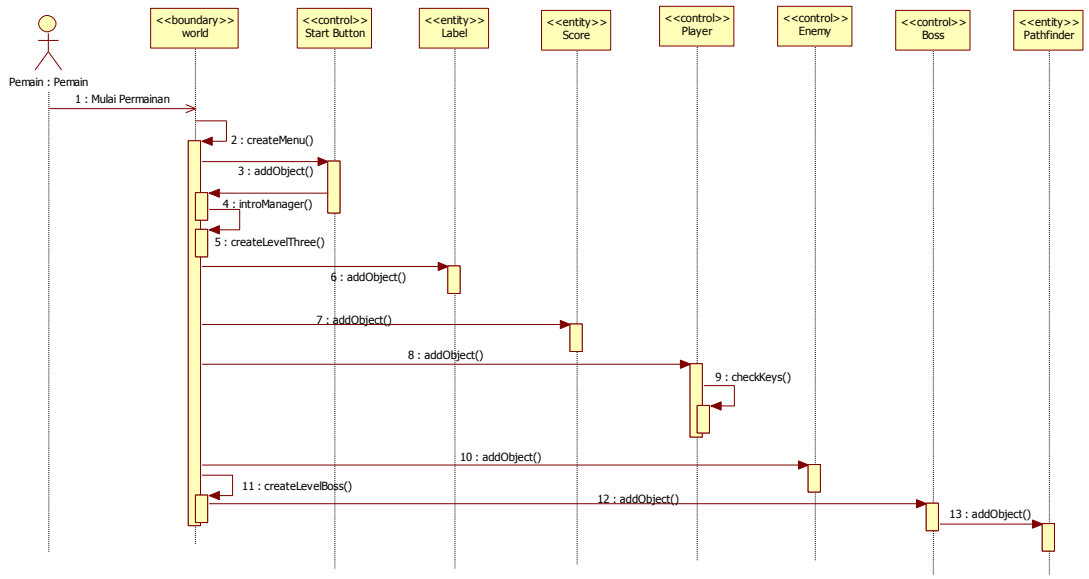
Sequence diagram menampilkan *level* 2 dapat dilihat pada gambar III.23



Gambar III.23 *Sequence* Diagram Menampilkan *Level* 2

D. *Sequence* Diagram Menampilkan *Level* 3

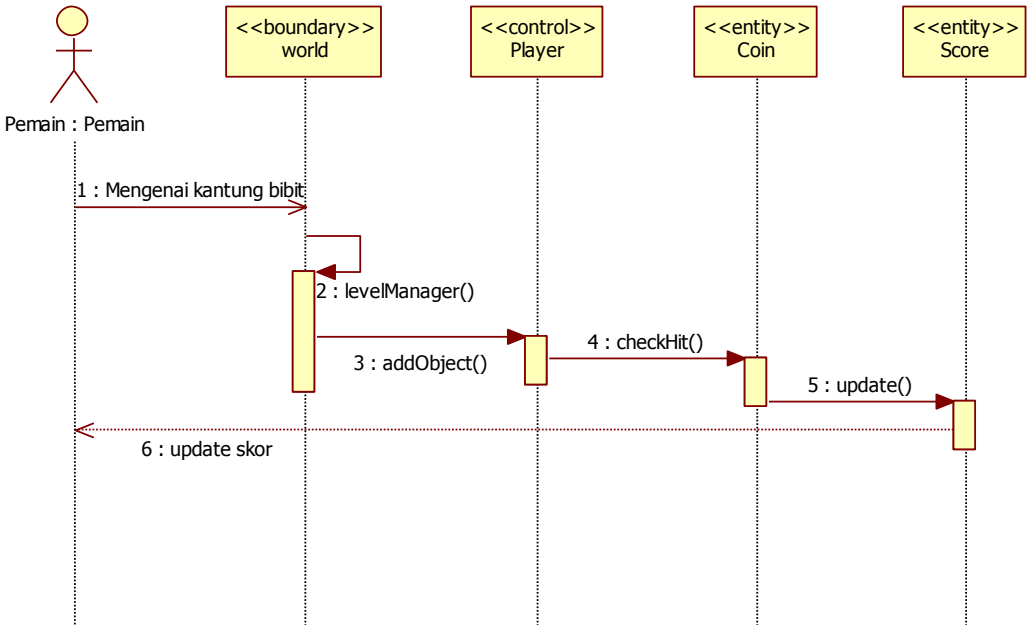
Sequence diagram menampilkan *level* 3 dapat dilihat pada gambar III.24



Gambar III.24 *Sequence* Diagram Menampilkan *Level* 3

E. *Sequence* Diagram Perhitungan Skor

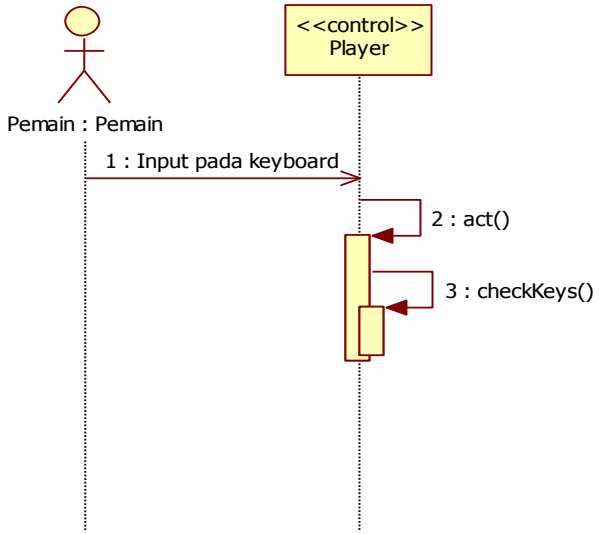
Sequence diagram memulai permainan dapat dilihat pada gambar III.25



Gambar III.25 *Sequence* Diagram Perhitungan Skor

F. *Sequence* Diagram Pengendalian Karakter

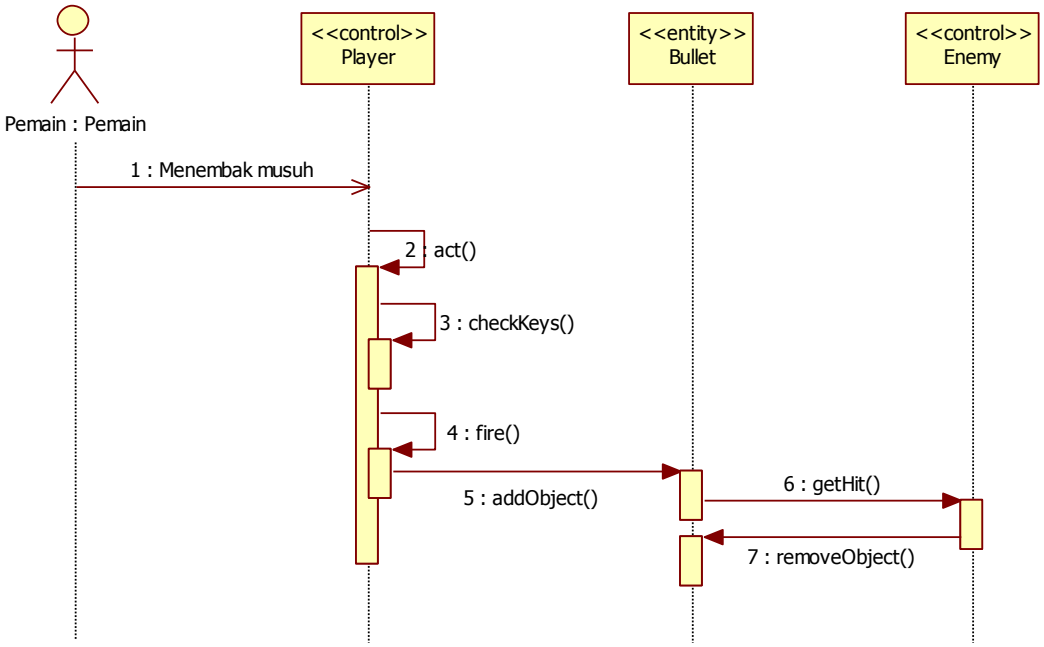
Sequence diagram pengendalian karakter dapat dilihat pada gambar III.26



Gambar III.26 *Sequence Diagram* Pengendalian Karakter

G. *Sequence* Diagram Menembak Musuh

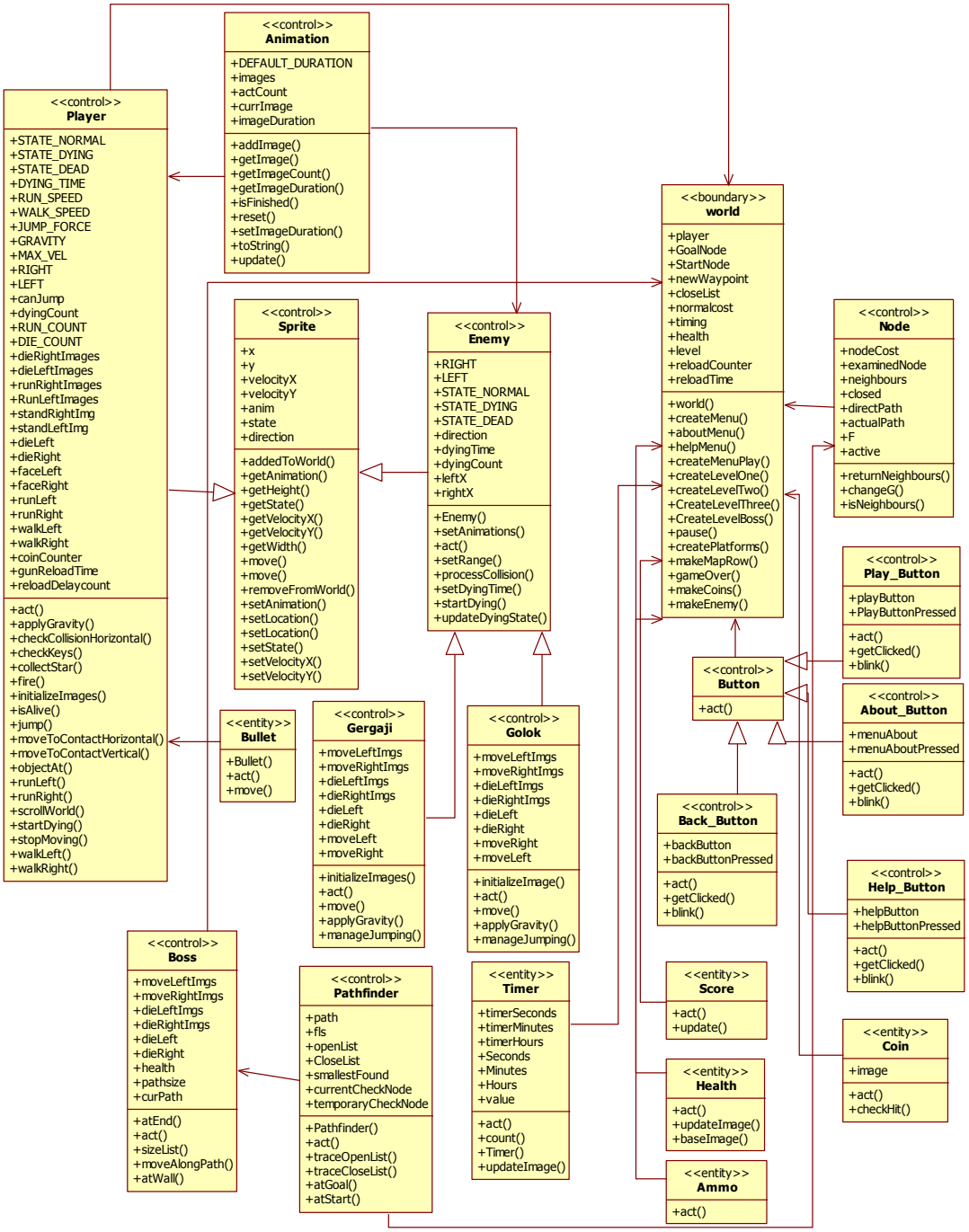
Sequence diagram menembak musuh dapat dilihat pada gambar III.27



Gambar III.27 *Sequence* Diagram Menembak Musuh

III.1.6.7 *Class* Diagram

Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. *Class* diagram *game Save Our Forest* dapat dilihat pada gambar III.28



Gambar III.28 *Class* Diagram

III.2 Perancangan Sistem

Perancangan sistem adalah suatu bagian dari metodologi pengembangan suatu perangkat lunak yang dilakukan setelah tahapan untuk memberikan gambaran secara terperinci.

III.2.1 Perancangan Komponen Permainan

Perancangan komponen permainan menggambarkan komponen – komponen yang akan dirancang dalam permainan berupa perancangan karakter dan perancangan *story board*

III.2.1.1 Karakter

Berikut ini adalah perancangan karakter yangakan diterapkan di aplikasi game yang akan dibangun dan akan dijelaskan di tabel 3.8.

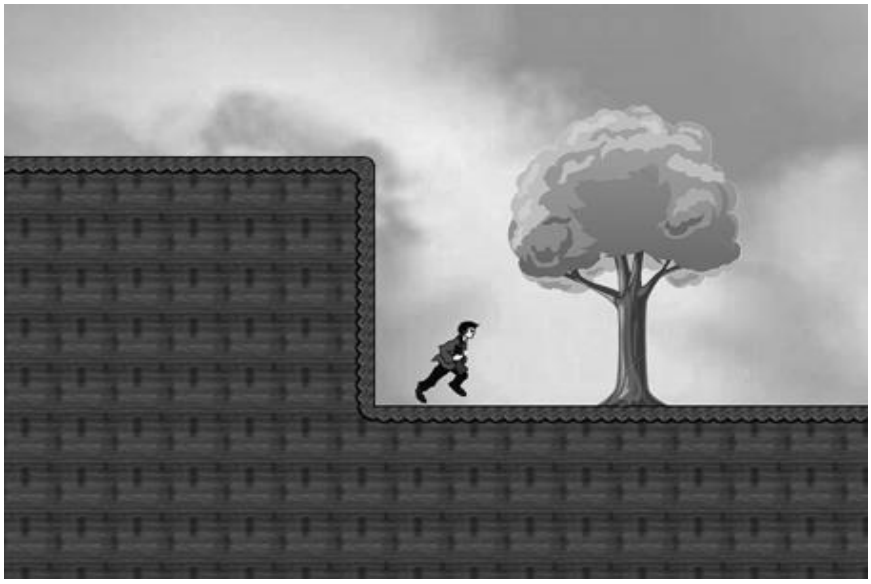
Tabel III.12 Karakter pada *Game Save Our Forest*

Nama Karakter	Kemampuan
 **Karakter Utama**	Merupakan karakter utama, memiliki kemampuan untuk melompat, menghindar, dan menembak Memiliki 5 *health point*
	Merupakan musuh pada level 1,2, dan 3 memiliki kemampuan untuk mengejar player, memiliki 1 *health point*
	Merupakan musuh pada level 2, dan 3 memiliki kemampuan untuk mengejar player, memiliki 3 *health point*

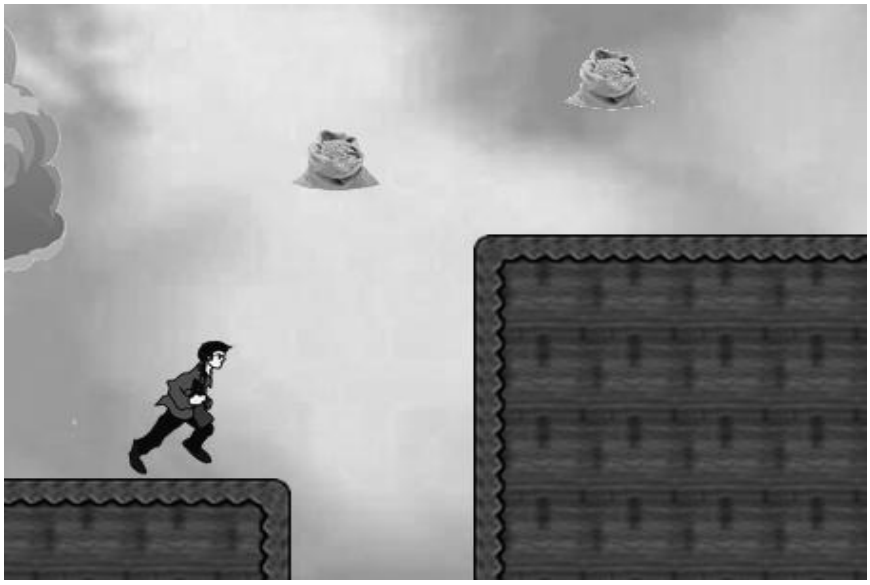
 **Bang Jack**	Bang Jack merupakan boss pada *game* ini, memiliki kemampuan untuk mengejar player AI yang digunakan untuk musuh ini adalah algoritma A* (A Star) Memiliki 7 *health point*

III.2.1.2 *Story Board*

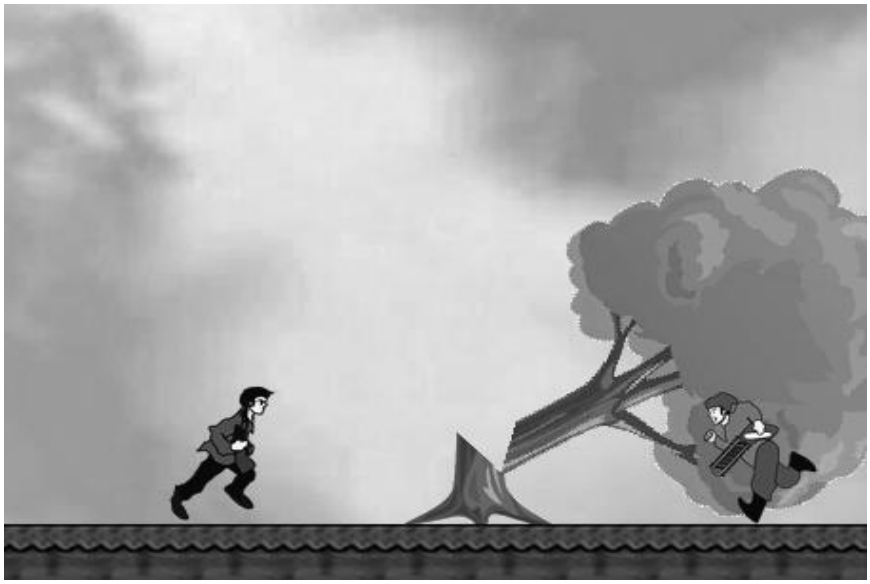
Story board adalah sketsa gambar yang disusun seperti halnya sebuah naskah cerita. *Storyboard* dapat menyampaikan ide cerita kepada orang lain dengan cara yang lebih mudah, karena dengan *story board* dapat menggiring imajinasi seseorang menuju persepsi yang kita inginkan melalui gambar-gambar yang dibuat.



Gambar III.29 Storyboard Level 1



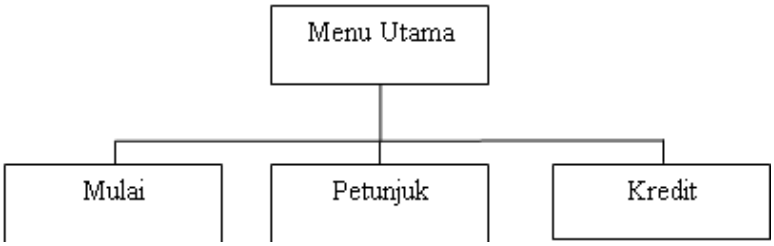
Gambar III.30 Storyboard Level 2



Gambar III.31 *Storyboard* Level 3

III.2.2 Perancangan Struktur Menu

Merupakan fitur-fitur yang disediakan pada sistem yang dirancang dan untuk mempermudah dalam penggunaan sistem nantinya. Perancangan struktur menu dapat dilihat pada gambar III.31

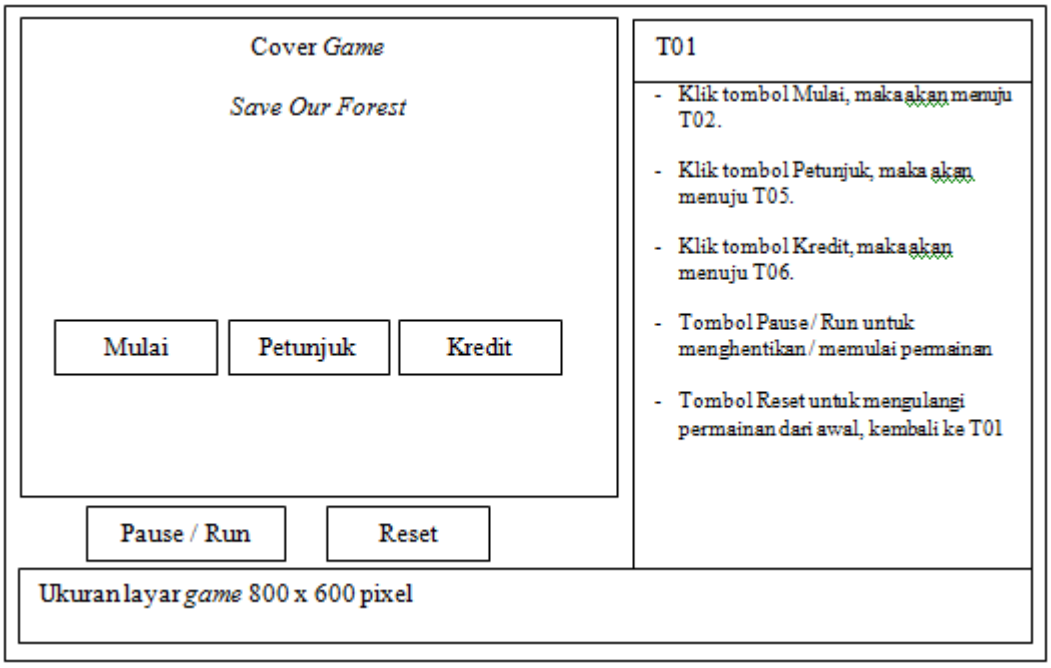


Gambar III.32 Struktur Menu

III.2.3 Perancangan Antar Muka

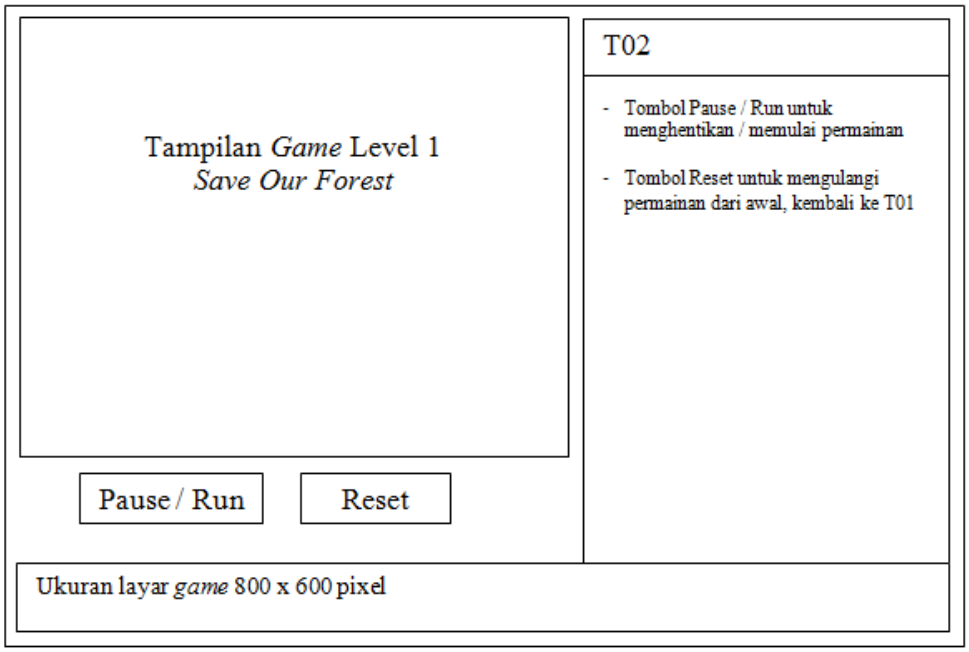
Perancangan antar muka bertujuan untuk memberikan gambaran tentang aplikasi yang akan dibangun, sehingga akan mempermudah dalam mengimplementasikan aplikasi serta akan memudahkan pembuatan aplikasi.

1. Menu Utama



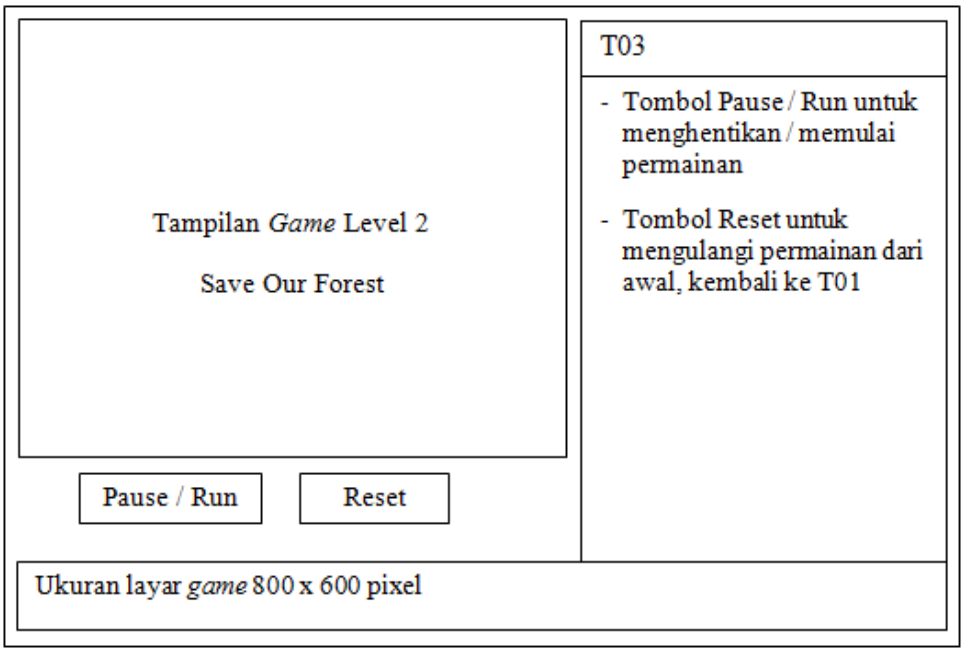
Gambar III.33 Perancangan Antarmuka Menu Utama *Game*

2. Level 1



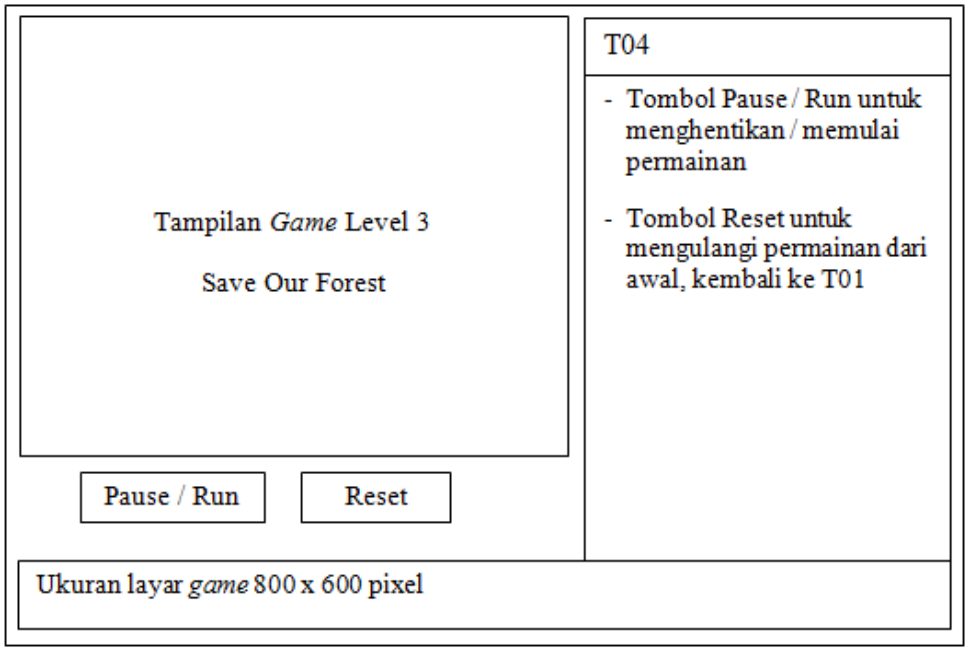
Gambar III.34 Perancangan Antarmuka *Game* Level 1

3. Level 2



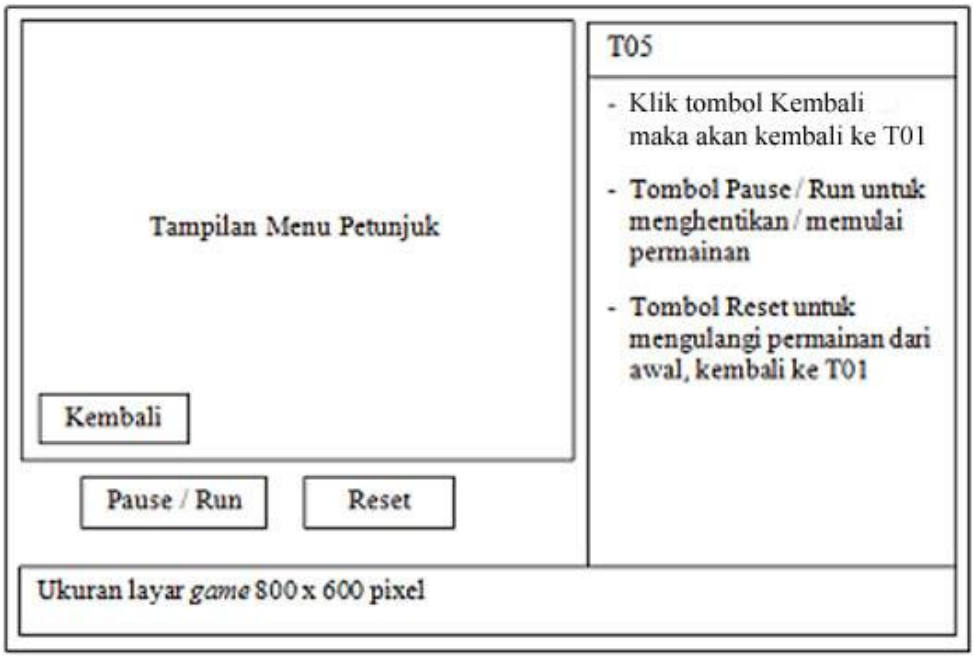
Gambar III.35 Perancangan Antarmuka *Game* Level 2

4. Level 3



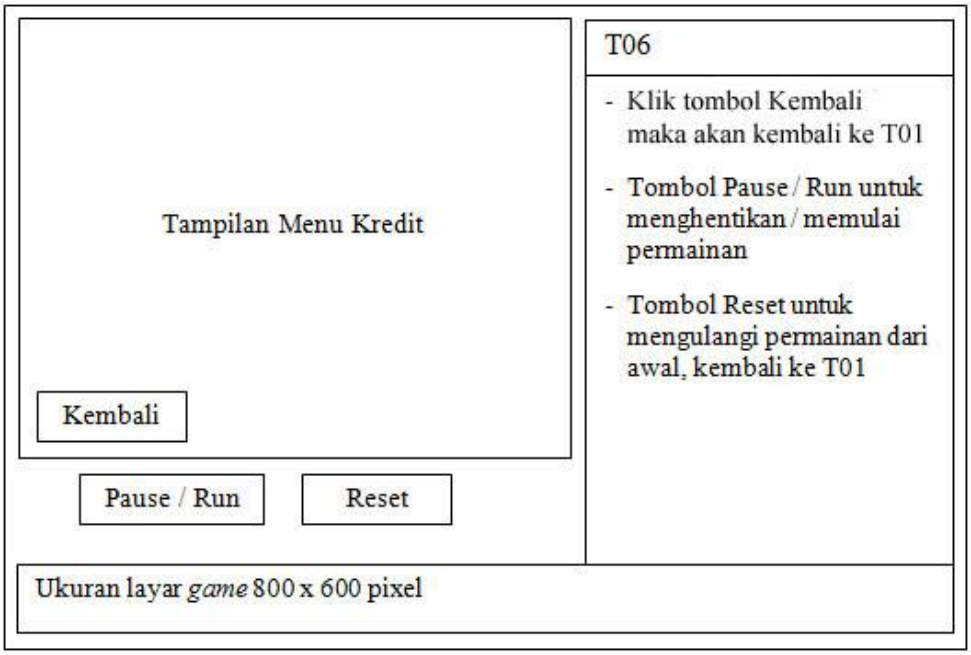
Gambar III.36 Perancangan Antarmuka *Game* Level 3

5. Petunjuk



Gambar III.37 Tampilan Antarmuka Menu Petunjuk

6. Kredit



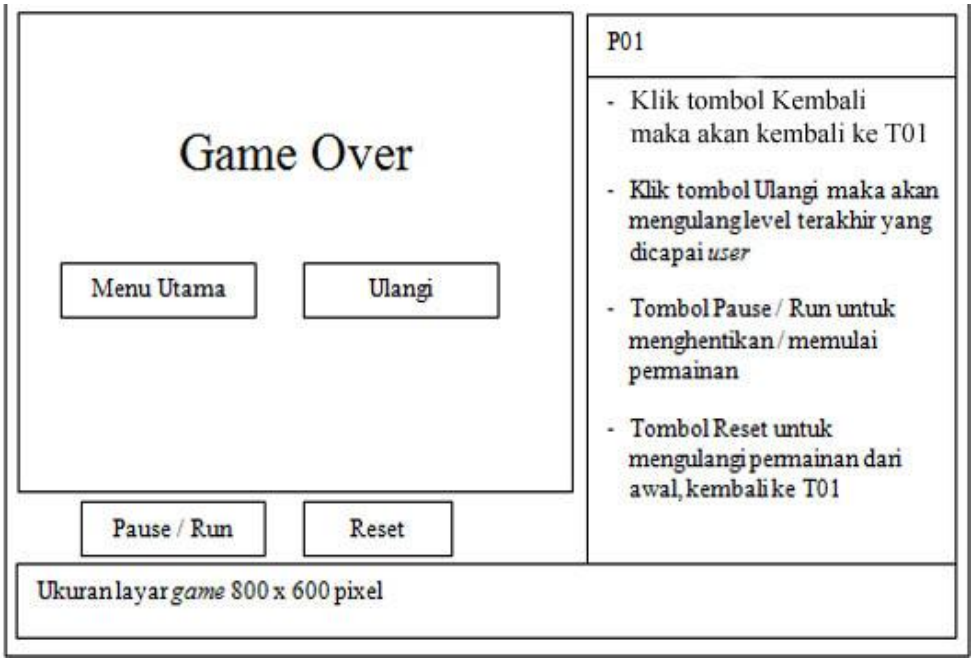
Gambar III.38 Tampilan Antarmuka Menu Kredit

III.2.4 Perancangan Pesan

Perancangan pesan bertujuan untuk memberikan peringatan jika user melakukan aktivitas tertentu yang akan memicu pesan.

1. Game Over

User akan mendapatkan pesan *Game Over* jika kalah dalam permainan.



Gambar III.39 Perancangan Pesan *Game Over*

2. Level Selesai

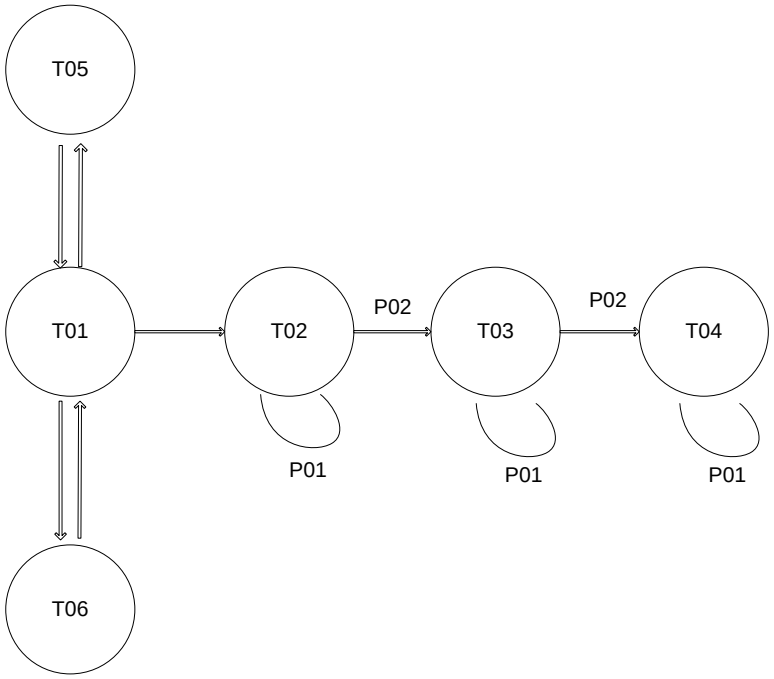
User akan mendapatkan pesan level selesai jika dapat menyelesaikan *level*.



Gambar III.40 Perancangan Pesan Level Selesai

III.2.5 Jaringan Semantik

Setelah melakukan perancangan antar muka dan perancangan pesan maka dilakukanlah perancangan terhadap hubungan antar *form* dan pesan. Untuk menggambarkan hubungannya digambarkan dalam sebuah jaringan semantik.



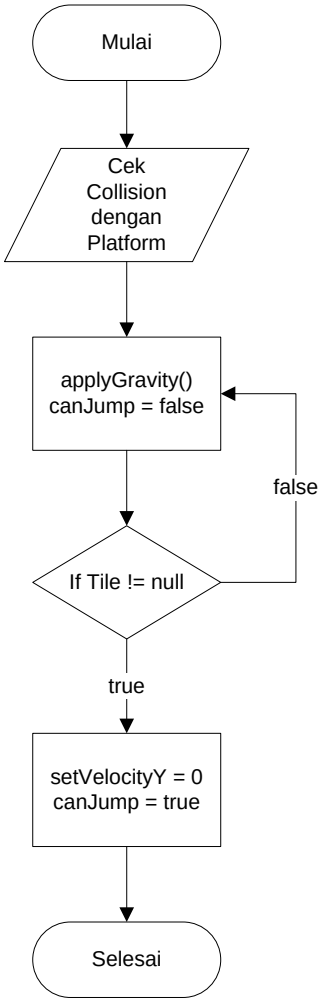
Gambar III.41 Jaringan Semantik

III.2.6 Perancangan Metode

Perancangan metode merupakan kumpulan metode yang ada pada sistem yang dirancang, dan berfungsi untuk menjalankan sistem. Dan berikut ini adalah perancangan metode yang ada dalam sistem.

1. Perancangan metode *manageJumping*

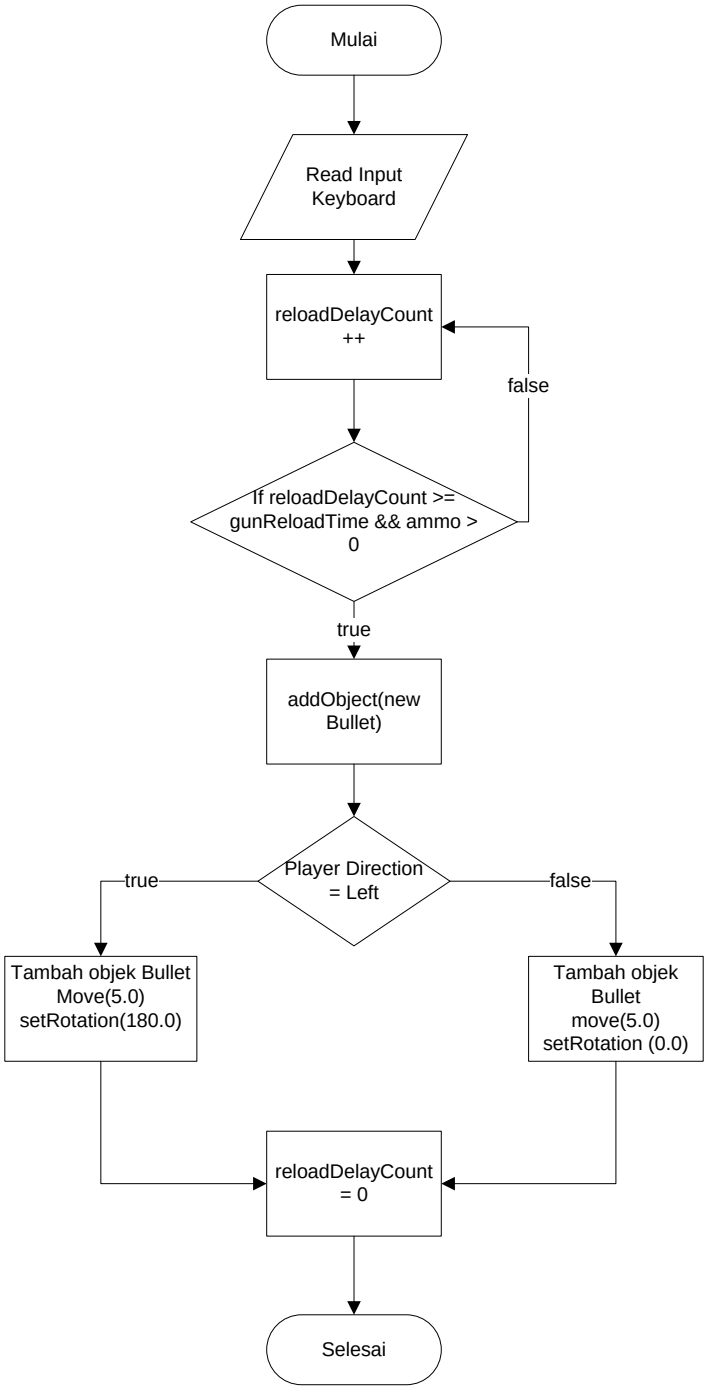
Metode ini berfungsi untuk penanganan gravitasi terhadap *player*. Untuk lebih jelas dapat dilihat pada gambar III.42



Gambar III.42 Perancangan *Method manageJumping*

2. Perancangan metode *fire*

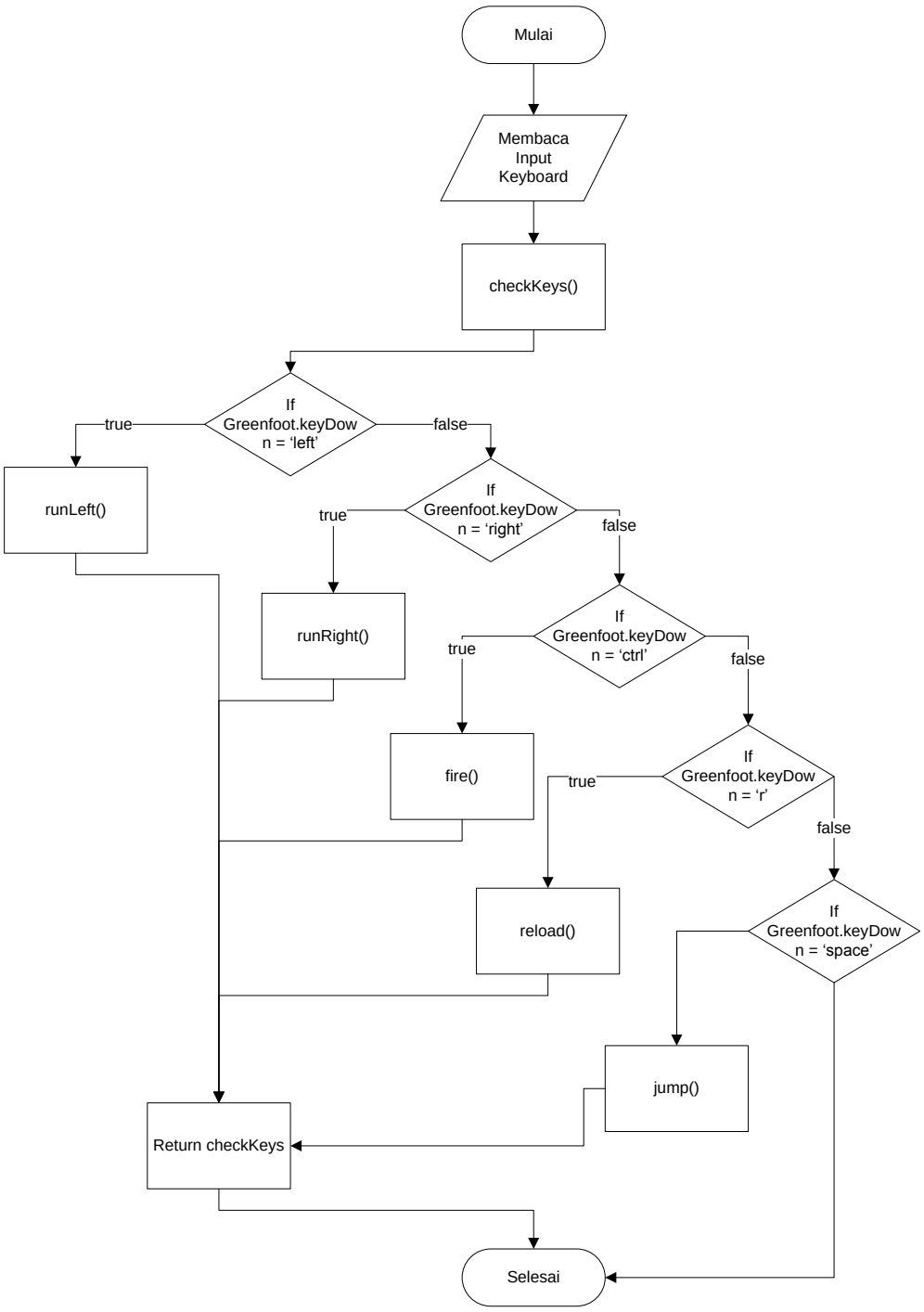
Metode ini berfungsi untuk penanganan senjata pada *player*. Untuk lebih jelas bisa dilihat pada gambar III.43



Gambar III.43 Perancangan *Method fire*

3. Perancangan metode *checkKeys*

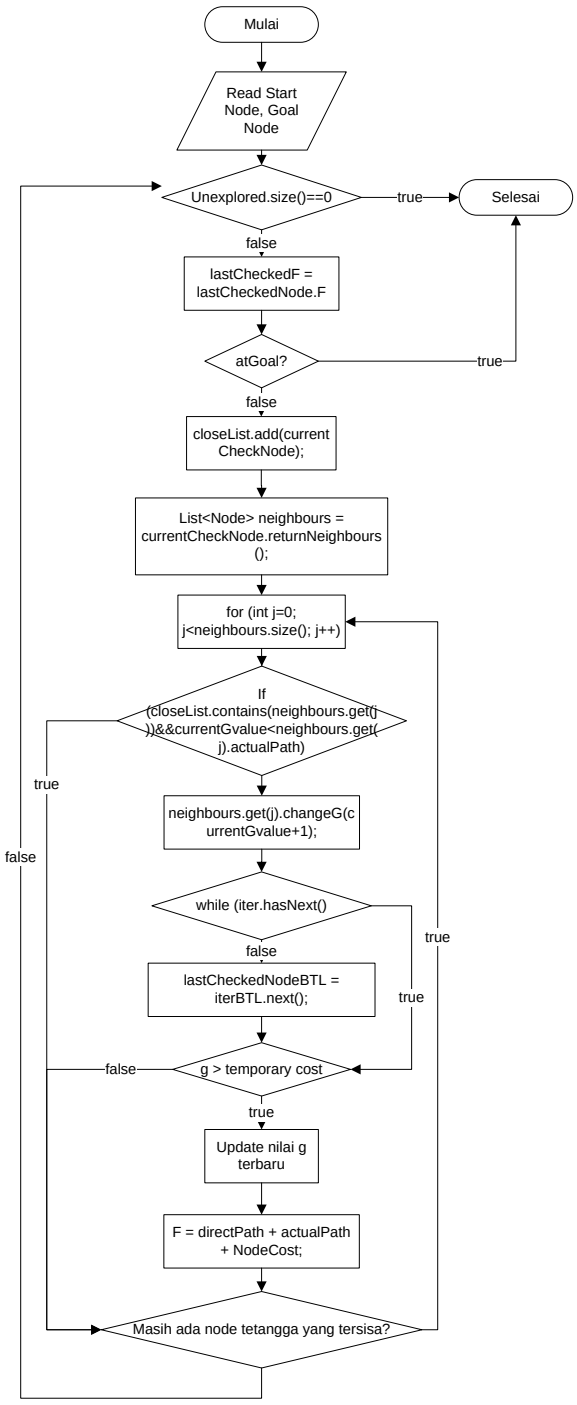
Metode ini berfungsi untuk penanganan pergerakan pada *player*. Untuk lebih jelas bisa dilihat pada gambar III.44



Gambar III.44 Perancangan *Method checkKeys*

4. Perancangan metode *Pathfinder*

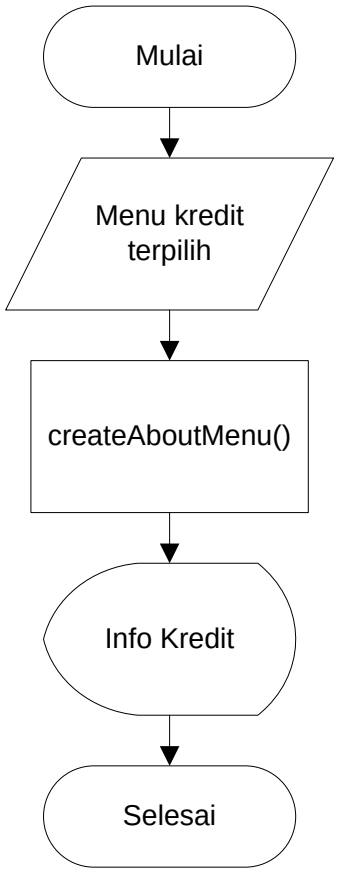
Metode ini berfungsi untuk *enemy* menemukan rute untuk mendekati *player*. Untuk lebih jelas bisa dilihat pada gambar III.45



Gambar III.45 Perancangan *Method Pathfinder*

5. Perancangan metode *aboutMenu*

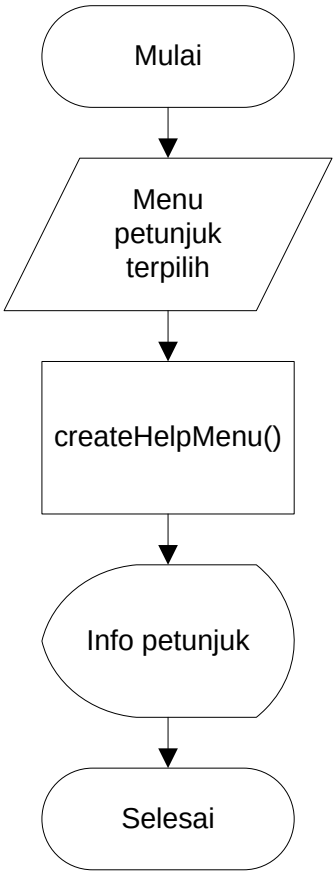
Metode ini akan dipanggil jika *user* memilih menu kredit. Untuk lebih jelas bisa dilihat pada gambar III.46



Gambar III.46 Perancangan *Method createAboutMenu*

6. Perancangan metode *helpMenu*

Metode ini akan dipanggil jika *user* memilih menu kredit. Untuk lebih jelas bisa dilihat pada gambar III.47



Gambar III.47 Perancangan *method createHelpMenu*