BAB 3

ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Sistem

Analisis sistem merupakan proses penguraian konsep ke dalam bagian-bagian yang lebih sederhana, sehingga struktur logisnya menjadi lebih jelas. Pada analisis sistem ini akan mencakup analisis masalah, analisis kebutuhan non-fungsional dan analisis kebutuhan fungsional. Metode digunakan untuk menguji, membandingkan, menilai, dan memahami sistem pemikiran yang kompleks dan memecahkan permasalahan dan berinteraksi untuk mencapai tujuan yang dicapai.

3.1.1 Analisis Masalah

Berdasarkan hasil wawancara dengan Bapak Ade Oma.S.Pd., guru di SMA Pasundan 1 Cimahi bahwa paradigma siswa-siswi mengenai pelajaran tersebut tidak baik. Siswa-siswi merasa pelajaran tersebut membuat jenuh karena kompleksitas materi yang cukup rumit, luas, dan cenderung banyak hapalan. Palagan Ambarawa sendiri merupakan salah satu cerita pertempuran yang terjadi di Indonesia yang saat ini mulai terlupakan oleh siswa padahal cerita tersebut merupakan salah satu pertempuran yang penuh dengan nilai historis.

Berdasarkan uraian latar belakang masalah tersebut, maka dapat dirumuskan beberapa masalahnya yaitu :

1. Bagaimana membangun suatu *game* yang memberikan informasi tentang cerita sejarah Palagan Ambarawa.
2. Bagaimana membangun suatu *game* yang tidak hanya sekedar memberikan materi yang rumit dan cenderung banyak hafalan.

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.

Game P10NER, Pertempuran 10 November merupakan *game* yang menceritakan pertempuran yang terjadi di Surabaya pada tanggal 10 November 1945. *Game* ini mengambil setting di Hotel Yamato saat perobekan bendera Belanda. Berikut tampilan menu pada *game* P10NER terlihat pada gambar 3.1.



Gambar. 3.1 Game P10NER, Pertempuran 10 November 1945 [3].

Dalam *game* ini *player* berperan sebagai Wawan yaitu pejuang yang berusaha untuk mengalahkan serangan tentara Belanda yang dipersenjatai peralatan yang canggih, dan *player* diharuskan melewati beberapa misi seperti merebut Hotel Yamato dari tangan sekutu dan terakhir melewati misi yaitu pertempuran 10 November 1945.

Game ini memiliki keterbatasan yaitu, grafik yang tidak terlalu bagus dan pengambilan tempat yang ada pada *game* ini tidak sesuai dengan fakta yang ada. Fakta tersebut dapat dilihat di gambar 3.2 dan analisis yang terdapat *game* P10NER dapat dilihat pada Tabel 3.1.



Gambar. 3.2 Tampilan game P10NER[3].

Tabel 3.1 Analisis *game* sejenis

Nama *Game*	*Genre Game*	Desain Grafis	*Storyline* dan *gameplay*	*Multiplayer*	Basis
P10NER, Pertempuran 10 November	*First Person Shooter*	3D tetapi lebih terkesan kartun dan penuh warna.	alur cerita sedikit berbeda dengan fakta sejarah pertempuran 10 November 1945	Tidak.	*Desktop*

a. Gameplay

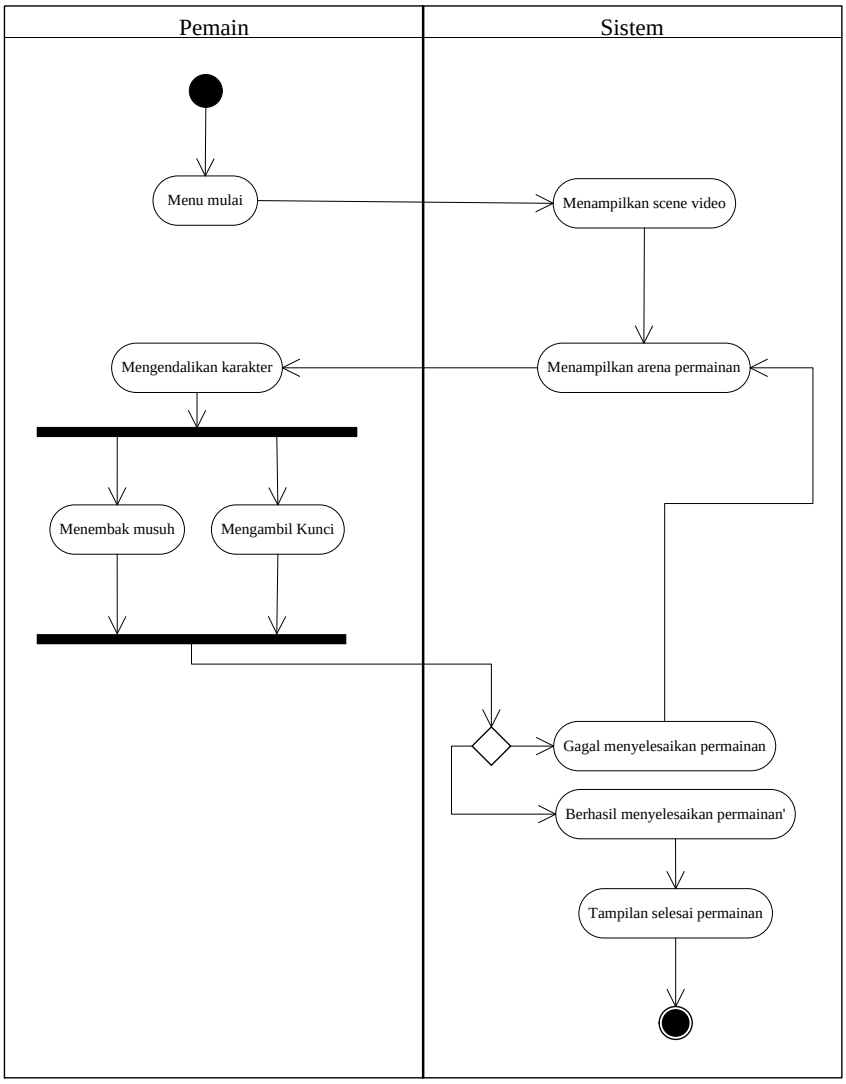
Pada *game* P10NER 10 November 1945, pemain harus mencari dan mengambil beberapa kunci yang disediakan oleh sistem disetiap *level* yang dimainkan, untuk digunakan pemain ketika akan membuka pintu yang membutuhkan kunci untuk membukanya.

b. Skenario *Game*

Skenario dalam permainan *game* P10NER dapat dilihat dari alur permaiannya sebagai berikut :

a. Pemain berada di dalam Hotel Yamato yang dipenuhi oleh musuh.
b. Pemain harus mencari kunci apabila hendak membuka sebuah pintu atau jalan utama ketika berada di dalam permainan.
c. Pemain harus menghindari dan mematikan musuh ketika berada di arena permainan.
d. Jika pemain terkena tembakan oleh musuh, maka darah / *health point* akan berkurang dan apabila habis maka permainan diulangi ke awal permainan.

Alur sistem dari permainan *game* P10NER dapat di gambarkan melalui *activity diagram* yang terlihat pada gambar 3.3.



Gambar 3.3 Alur permainan pada *game* P10NER

Game P10NER memiliki beberapa keterbatasan yaitu,

1. *Game* ini merupakan *game* edukasi akan tetapi konten edukasi yang diperlihatkan pada *game* ini tidak terlalu terlihat.
2. *Setting* tempat / *mapping* yang ada pada *game* ini kurang sesuai dengan fakta sejarah yang terjadi. Seperti ketika peristiwa perobekan bendera Belanda di hotel Yamato, fakta sejarah yang ada adalah dahulu rakyat Surabaya bergerak dari arah luar Hotel Yamato dan menurunkan bendera dengan memanjat. Tapi pada *game* ini pemain secara tiba-tiba telah masuk ke dalam hotel Yamato.

3. Materi tentang sejarah Pertempuran 10 November 1945 hanya diberikan melalui *scene video*.

Disamping ada beberapa keterbatasan, *game* P10NER sendiri memiliki banyak kelebihan seperti :

1. Terdapat fitur *save* dan *load game*
2. Senjata yang dapat digunakan *player* beragam.

3.1.3 Analisis *game* yang dibangun

Game Palagan Ambarawa yang akan dibangun yaitu *game* yang mempunyai grafis 3D yang ber-*genre first person shooter (FPS). Game* ini mempunyai alur cerita yang sama seperti sejarah Pertempuran Ambarawa yang sebenarnya.

Game dibangun dengan konsep sudut pandang orang pertama atau *first person shooter* dengan tujuan agar kesan pertempuran yang dihadirkan lewat *game* lebih terlihat nyata.

Beberapa fitur yang ditawarkan di dalam *game* ini :

1. *Single player*
2. *Game* diangkat dari cerita salah satu sejarah pertempuran di Indonesia
3. *Game* dibuat dengan grafis 3D
4. *Game* dibuat dengan *genre first person shooter* dengan konsep sudut pandang orang pertama.

3.1.4 *Storyline*

Game yang dibangun *single player* atau dengan satu pemain yang dijalankan seorang *actor* yang bernama Tarmidji. *Actor* tersebut harus menyelesaikan 3 *level* yang tiap-tiap *level*nya memiliki rintangan yang berbeda-beda.

Level 1, insiden ini bermula ketika tawanan Jepang yang awalnya akan dibebaskan tetapi malah dipersenjatai oleh sekutu. Hal ini menyebabkan kemarahan tentara Indonesia hingga terjadi pertempuran. Pada insiden ini *actor* diharuskan mengalahkan tentara jepang yang telah di persenjatai oleh sekutu.

Level 2, pada *level* ini juga *actor* akan dihadapkan dengan serangkaian-serangkaian pertempuran dan pengejaran antara tentara Indonesia terhadap sekutu yang melarikan diri menuju Ambarawa. Pada insiden ini *actor* diharuskan mengejar tentara sekutu yang menuju Ambarawa dengan navigasi kemudian mengalahkan tentara sekutu yang menghalangi pengejaran.

Level 3, kematian Letkol Isdiman menjadi kemarahan tentara Indonesia saat itu maka pertempuran Ambarawa pun terjadi dibawah pimpinan Kolonel Sudirman yang saat itu sedang sakit. Pada insiden ini *actor* akan menghadapi tentara sekutu di Ambarawa dengan tujuan menguasai wilayah ambarawa dan mengusir tentara sekutu. Pertempuran dimenangkan oleh tentara Indonesia dan kegigihan Kolonel Sudirman diabadikan dengan didirikan "Monumen Pertempuran Ambarawa".

3.1.5 Gameplay

Tugas utama pemain dalam memainkan *game* ini adalah menyelesaikan tiga misi yang diberikan. Misi dari *game* ini yaitu pemain harus menemukan bendera merah putih di markas para tawanan tentara sekutu di kota Magelang pada misi tersebut pemain harus mendapatkan informasi yang bisa di dapatkan dari tiap *level*nya. *Level* pertama bendera merah putih berada di markas para tawanan tentara sekutu di kota Magelang. Anda harus mengalahkan para tawanan yang dipersenjatai oleh sekutu. *Level* kedua, pemain harus mampu mengalahkan para tentara sekutu yang melarikan diri ke desa jambu. *Level* ketiga, pemain harus mampu mengalahkan para tentara sekutu dan memukul mundur pasukan sekutu yang ada di Ambarawa.

3.1.6 Analisis Tingkat Kesulitan

Tingkat kesulitan dari tiap *level* akan dibedakan dari jumlah musuh dan *delay* tembakan musuh. Perbandingan tingkat kesulitan dari tiap *level* dapat dilihat pada tabel 3.2.

Tabel 3.2 Tingkat Kesulitan tiap *Level*

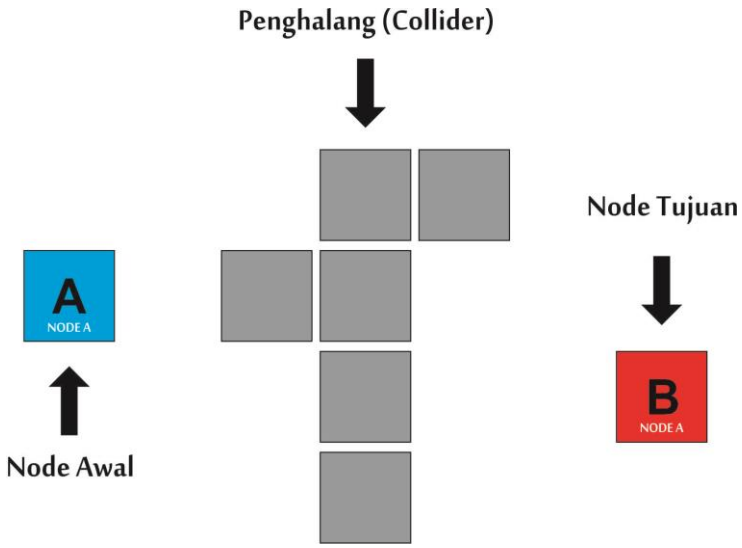
No	***Level* 1**	***Level* 2**	***Level* 3**
Jumlah Musuh	10	15	20
Delay Tembakan Musuh	Lambat	Lambat	Cepat

3.1.7 Analisis *Scoring*

Game Palagan Ambarawa ini tidak memiliki *score*, satu-satunya penilaian yang ada adalah sukses atau tidaknya misi dalam setiap *level*.

3.1.8 Analisis Algoritma A* (*A Star*)

Algoritma A* di dalam *game* akan diterapkan pada karakter teman sebagai penentuan jalur terpendek atau *pathfinding* menuju *target.* Cara kerja algoritma A* di dalam *game* lebih jelasnya dapat dilihat pada gambar 3.4.



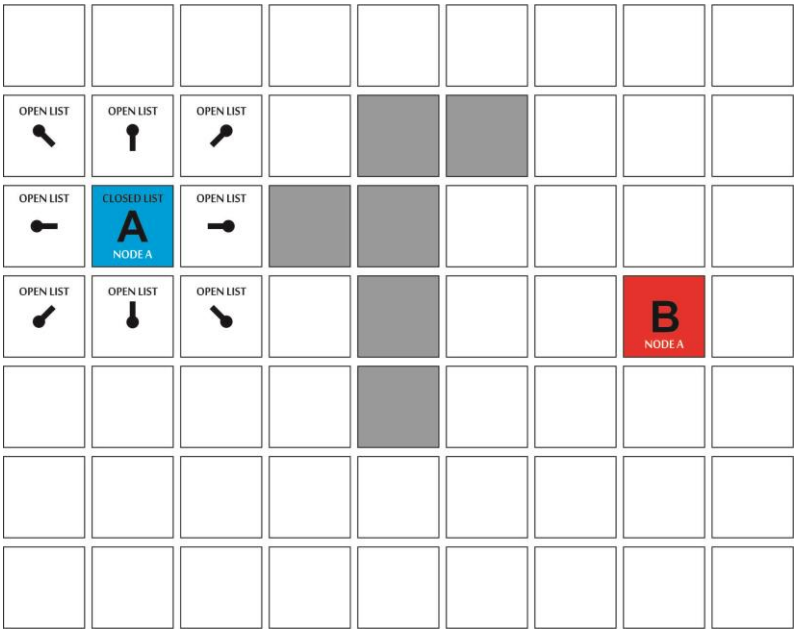
Gambar 3. 4 Ilustrasi A*

Gambar 3.4 diasumsikan bahwa wilayah pencarian dibagi-bagi menjadi kotak-kotak atau disebut *node*, *node* yang berwarna biru merupakan *node* awal (*node* A) dan *node* yang berwarna merah

merupakan *node* tujuan (*node* B), sedangkan *node* yang berwarna abu-abu merupakan dinding penghalang atau disebut *collision*. Diasumsikan pencarian jalan terbaik dimulai dari *node* A menuju *node* B dengan *node* abu-abu sebagai *collision*nya. Selanjutnya adalah, melakukan pencarian untuk menemukan jalan terpendek dengan cara memeriksa *node-node* yang berdekatan dari *node* A sampai *node* B. Langkah-langkahnya adalah sebagai berikut:

1. Pertama, dimulai dari *node* A dan tambahkan ke dalam *open list. Open list* merupakan *list* untuk *node-node* yang harus diperiksa.
2. Periksa semua *node* yang berdekatan dengan *node* A apakah *walkable* atau tidak, abaikan *node* dengan *collision* atau penghalang lalu tambahkan semua *node* tersebut ke dalam *open list.* Dan untuk setiap *node* yang baru ditambahkan, simpan *node* A sebagai "*parent*" yang nantinya akan digunakan untuk menelusuri jalan.
3. Hapus *node* A 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.

Open list dan *closed list* diilustrasikan pada gambar 3.5 sebagai berikut.

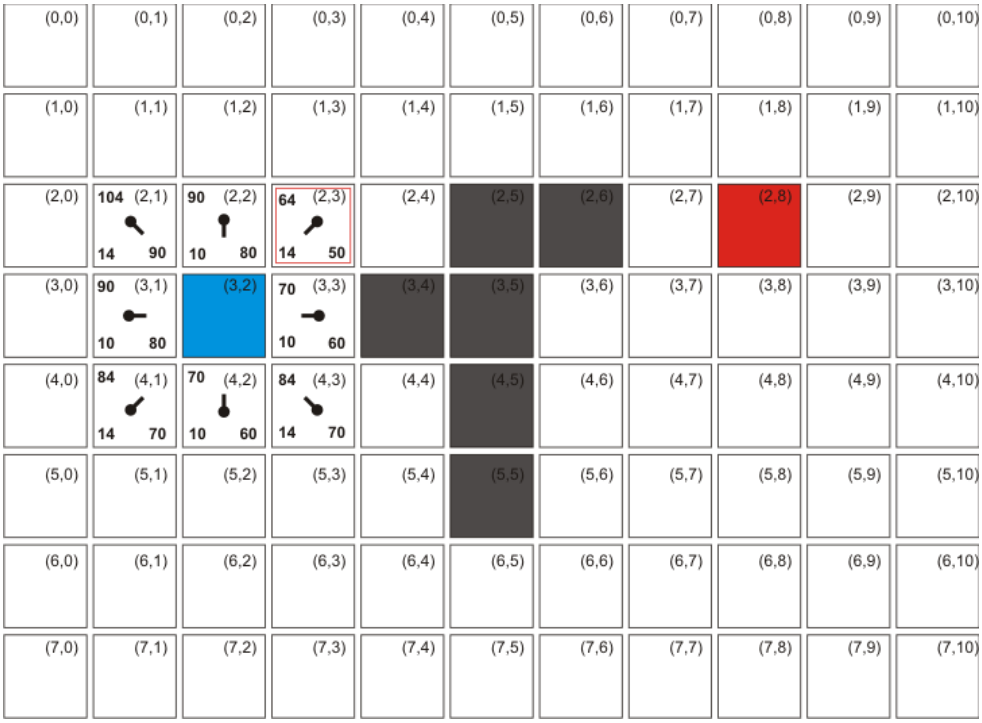


Gambar 3. 5 Ilustrasi *open list* dan *closed list*

Kotak yang berwarna biru adalah *node* A, dengan ditandai dengan garis berwarna biru muda yang berarti *node* tersebut sudah ditambahkan ke dalam *closed list.* Sedangkan kotak yang berada di sekelilingnya adalah *open list* yang berarti *node-node* yang harus diperiksa. Masing-masing *node* yang berada di dalam *open list* memiliki *pointer* yang mengarah ke *node* A sebagai induknya atau *parent* yang merupakan *node* awalnya.

Node yang memiliki nilai F paling rendah di dalam *open list* dengan menggunakan rumus F = G + H. G adalah biaya yang dikeluarkan dari *node* awal atau disini *node* A ke *node* yang ada di sekitarnya. Sedangkan H adalah fungsi *heuristic* yang digunakan untuk menghitung perkiraan biaya yang dikeluarkan dari *node* awal menuju *node* tujuan.

Proses ini dilakukan berulang kali atau *looping*, dimulai dari *open list* dan menemukan *node* dengan nilai F paling rendah. Perhitungan persamaan untuk mendapatkan nilai F dijelaskan dengan ilustrasi pada gambar 3.6 sebagai berikut.



Gambar 3. 6 Ilustrasi perhitungan menentukan nilai F

Open list dari *node* A nilai F yang dihasilkan adalah 64 yang didapatkan dari menambahkan nilai G = 14 dan nilai H = 50. 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 digunakan fungsi *heuristic*, metode yang digunakan di dalam contoh ini adalah metode *Manhattan* dimana perhitungan jumlah *node* hanya yang bergerak secara *vertical* dan *horizontal* menuju tujuannya serta mengabaikan penghalang atau *collision*, yang kemudian nilainya dikalikan dengan 10. Atau dirumuskan dengan:

H = 10*(abs(*current*X-*target*X) + abs(*current*Y-*target*Y))

Node yang terletak disebelah kanan dari *node* A diberikan nilai H = 60 yang didapatkan dari perhitungan 6 langkah atau 6 *node* menuju *node* B dan dikalikan 10, 6*10 = 60.

Nilai F yang telah didapatkan dari setiap *node* yang ada di *open list*, maka akan dipilih *node* yang memiliki nilai F paling rendah yang

kemudian disebut dengan *current node*, selanjutnya langkahnya adalah sebagai berikut.

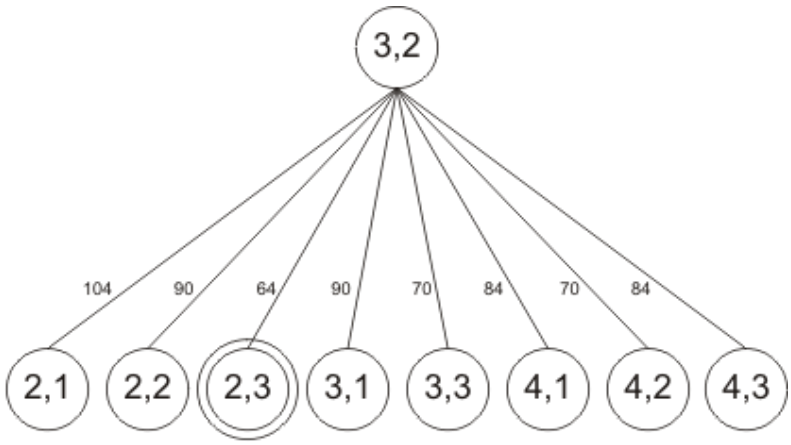
1. Hapus *current node* tersebut dari *open list* dan masukkan ke dalam *closed list.*
2. Periksa semua *node* yang berdekatan dengan *current node*, abaikan *node* yang *unwalkable* atau dalam contoh ini adalah *node* penghalang. Jika tidak ada pada *open list*, tambahkan ke dalam *open list.* Dan tambahkan "*parent*" untuk *node* tersebut.
3. Jika *node* sudah ada pada *open list*, periksa apakah ini jalan yang lebih baik dari *node* awal, dengan kata lain periksa apakah nilai G-nya lebih rendah apabila kita 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 3.2.

Tabel 3.2 Tabel Perhitungan langkah pertama

Open List	(2,1), (2,2), (2,3), (3,1), (3,3), (4,1), (4,2), (4,3)
Closed list	(3,2)
Parent	(3,2)
Current node	(2,3)

Gambat 3.6 merupakan penjelasan langkah pertama pencarian jalur terpendek / *pathfinding* menggunakan *tree*. *Node* (2,3) merupakan *current node* yang dihasilkan pada perhitungan langkah pertama.

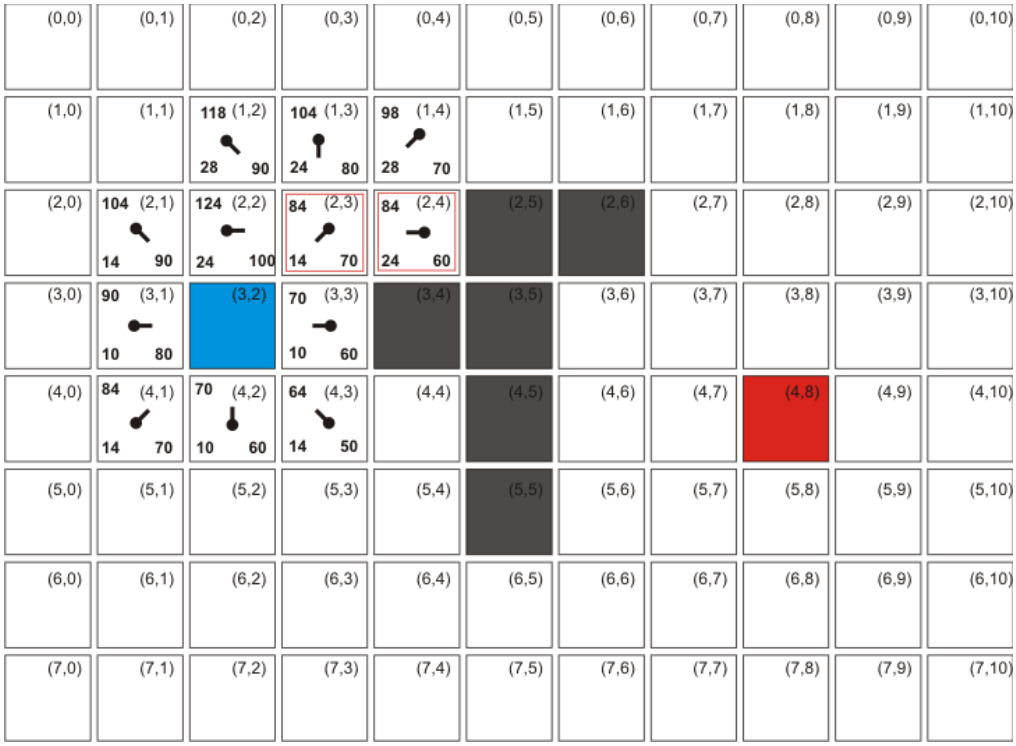


Gambar 3.7 Penjelasan A* dengan *tree* pada langkah pertama

Proses yang dijelaskan di atas, dilakukan berulang kali atau *looping*, dimulai dari *open list* dan menemukan *node* dengan nilai F paling rendah. Langkahnya adalah sebagai berikut :

1. Hapus *current node* tersebut dari *open list* dan masukkan ke dalam *closed list.*
2. Periksa semua *node* yang berdekatan dengan *current node*, abaikan *node* yang *unwalkable* atau dalam contoh ini adalah *node* penghalang. Jika tidak ada pada *open list*, tambahkan ke dalam *open list.* Dan tambahkan "*parent*" untuk *node* tersebut.
3. Jika *node* sudah ada pada *open list*, periksa apakah ini jalan yang lebih baik dari *node* awal, dengan kata lain periksa apakah nilai G-nya lebih rendah apabila menggunakan *node* ini, jika lebih rendah maka hitung ulang nilai F dan G serta merubah arah *pointer*, jika tidak maka jangan lakukan apa-apa.

Perhitungan pada langkah kedua yang berada pada *node* dengan koordinat (2,3) dapat dilihat pada gambar 3.8.



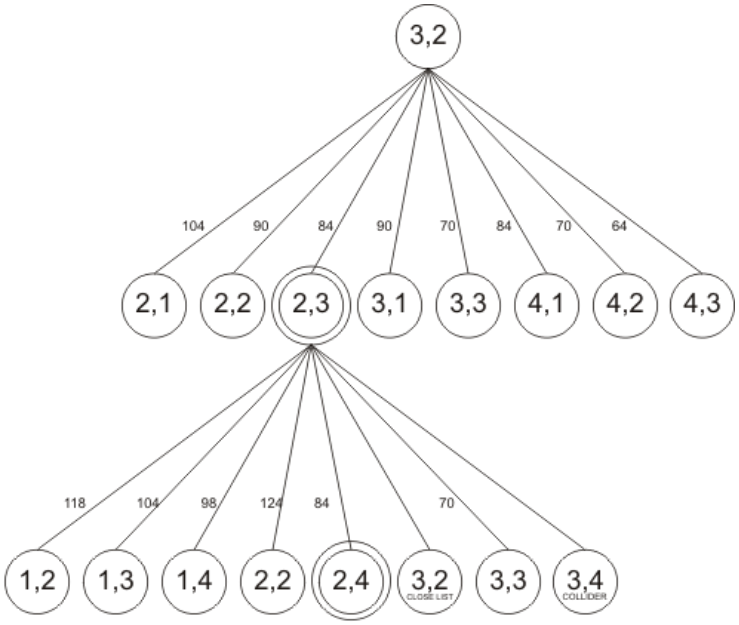
Gambar 3.8 Penjelasan Astar pada langkah kedua

Berikut ini penjelasan melalui tabel untuk pencarian jalan terpendek di langkah kedua dapat dilihat pada tabel 3.3.

Tabel 3.3 Tabel Perhitungan langkah kedua

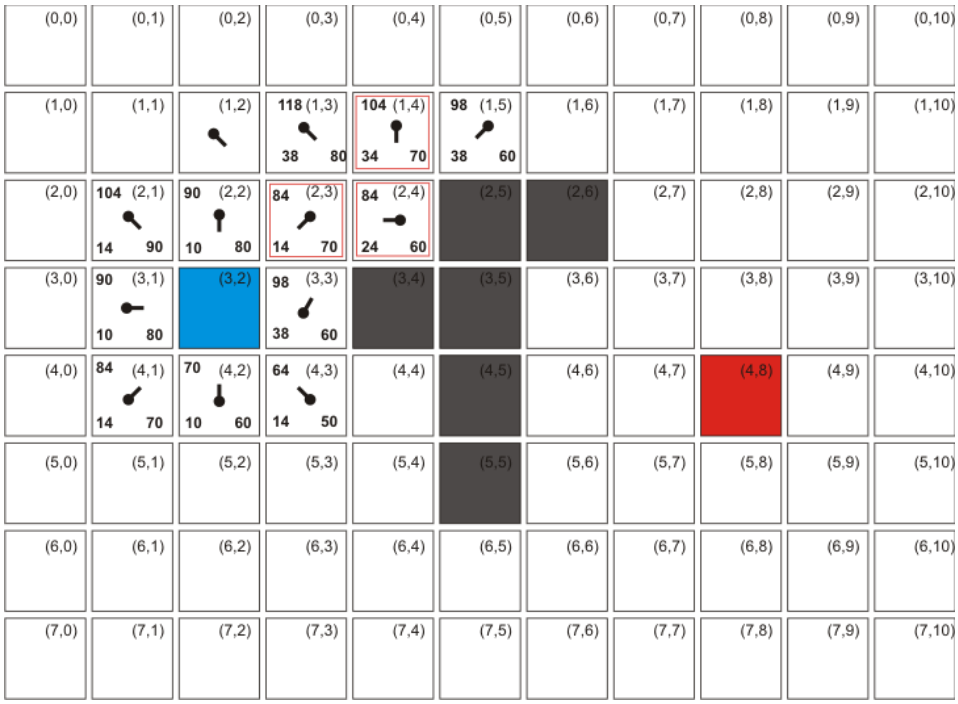
Open List	(1,2), (1,3), (1,4), (2,2), (3,3)
Closed list	(3,2),(2,3), (3,4)
Parent	(2,3)
Current node	(2,4)

Berikut ini penjelasan melalui *tree* untuk pencarian jalan terpendek pada langkah kedua dapat dilihat pada gambar 3.9.



Gambar 3.9 Penjelasan A* dengan *tree* pada langkah kedua

Perhitungan pada langkah ketiga yang berada pada *node* dengan koordinat (2,4) dapat dilihat pada gambar 3.10.



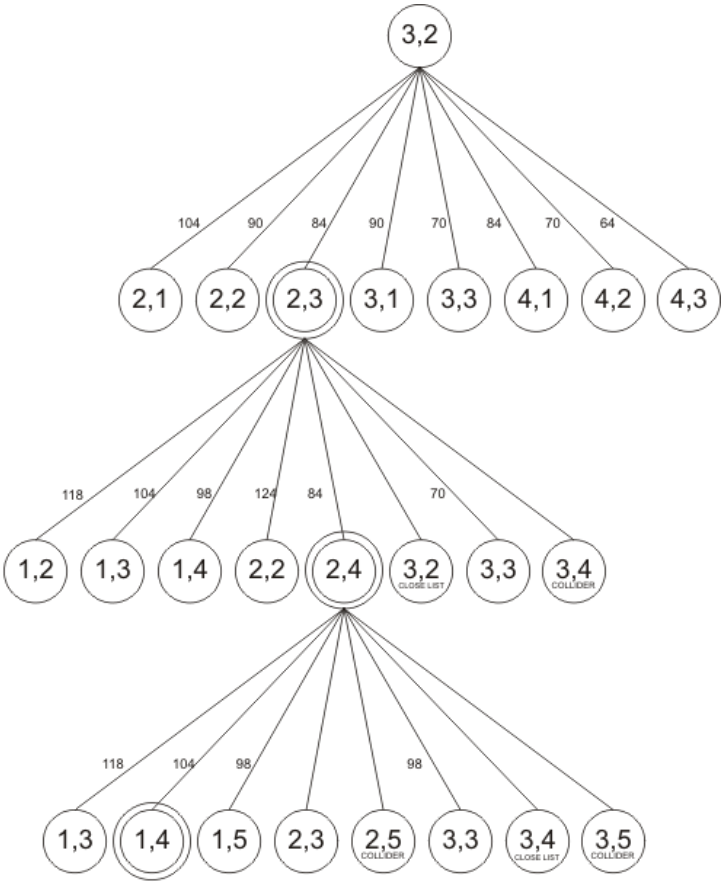
Gambar 3.10 Penjelasan Astar pada langkah ketiga

Berikut ini penjelasan melalui tabel untuk pencarian jalan terpendek di langkah ketiga dapat dilihat pada tabel 3.4.

Tabel 3.4 Tabel Perhitungan langkah ketiga

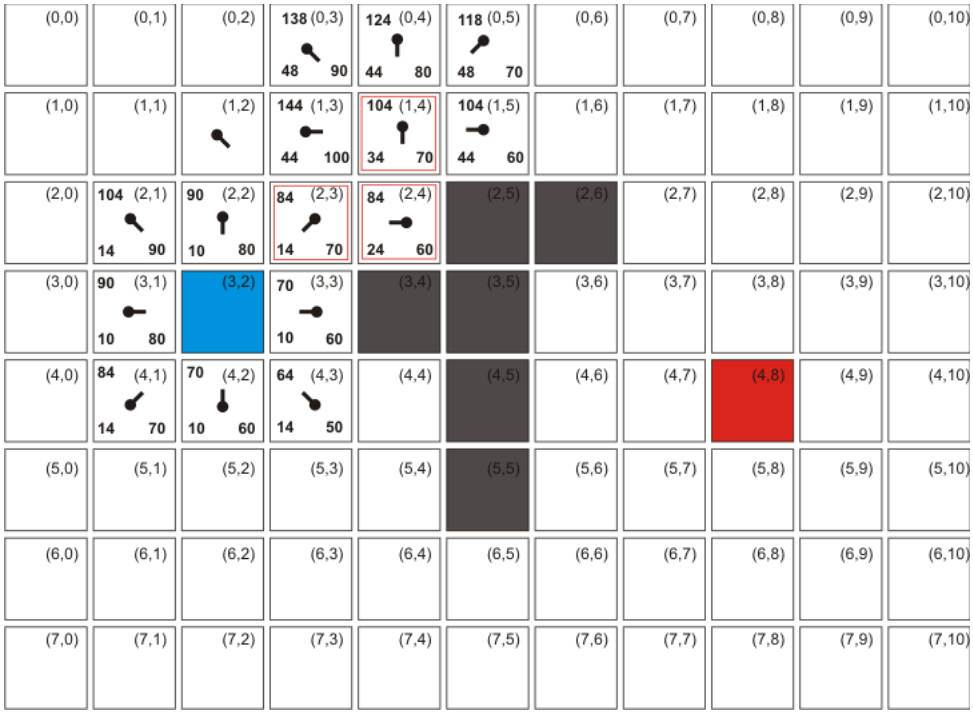
Open List	(1,3), (1,5), (3,3)
Closed list	(2,3),(2,5),(3,4),(3,5), (3,4)
Parent	(2,4)
Current node	(1,4)

Berikut ini penjelasan melalui *tree* untuk pencarian jalan terpendek pada langkah ketiga dapat dilihat pada gambar .



Gambar 3.11 Penjelasan A* dengan *tree* pada langkah ketiga

Perhitungan pada langkah keempat yang berada pada *node* dengan koordinat (1,4) dapat dilihat pada gambar 3.12.



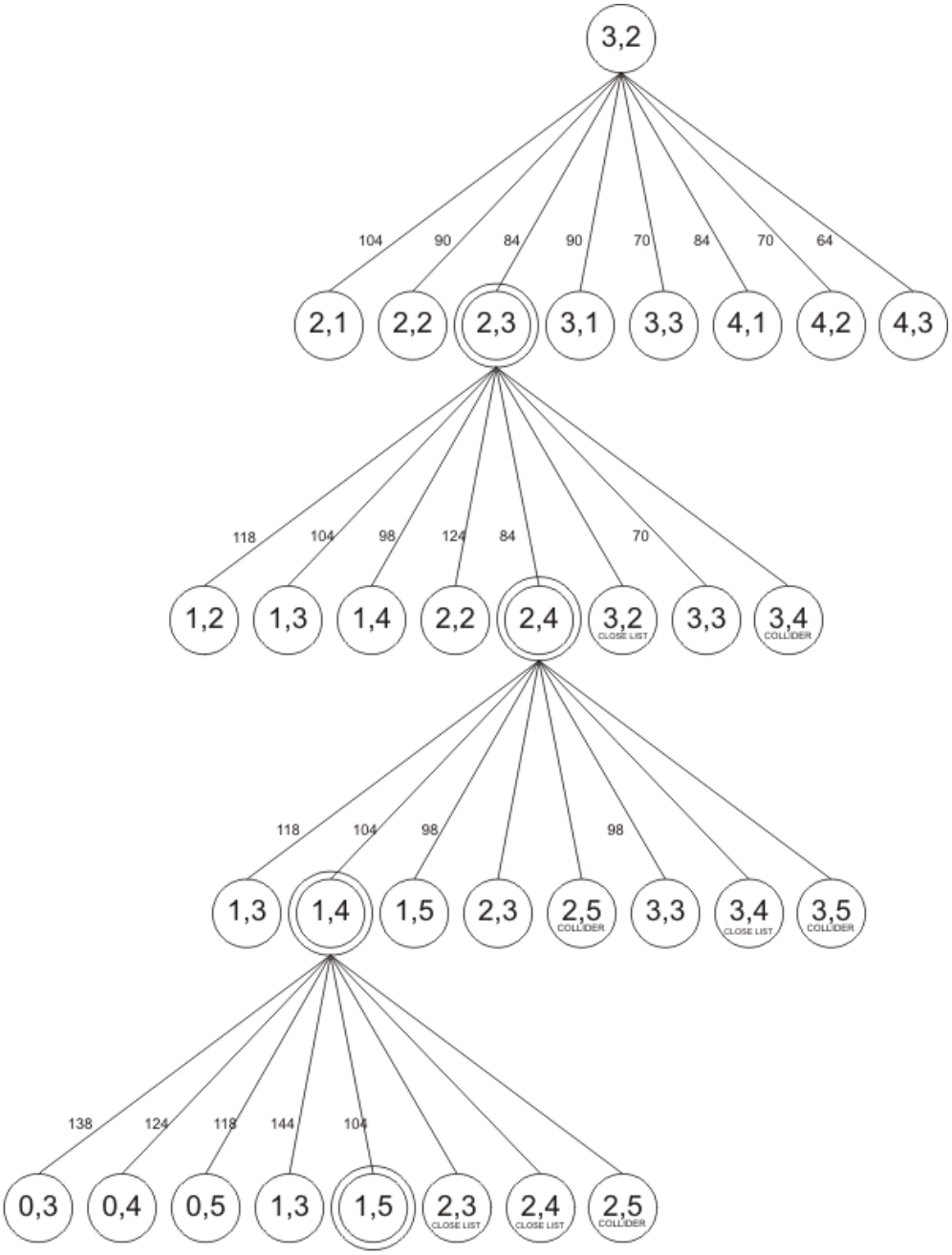
Gambar 3.12 Penjelasan Astar pada langkah keempat

Berikut ini penjelasan melalui tabel untuk pencarian jalan terpendek di langkah keempat dapat dilihat pada tabel 3.5.

Tabel 3.5 Tabel Perhitungan langkah kee*pat*

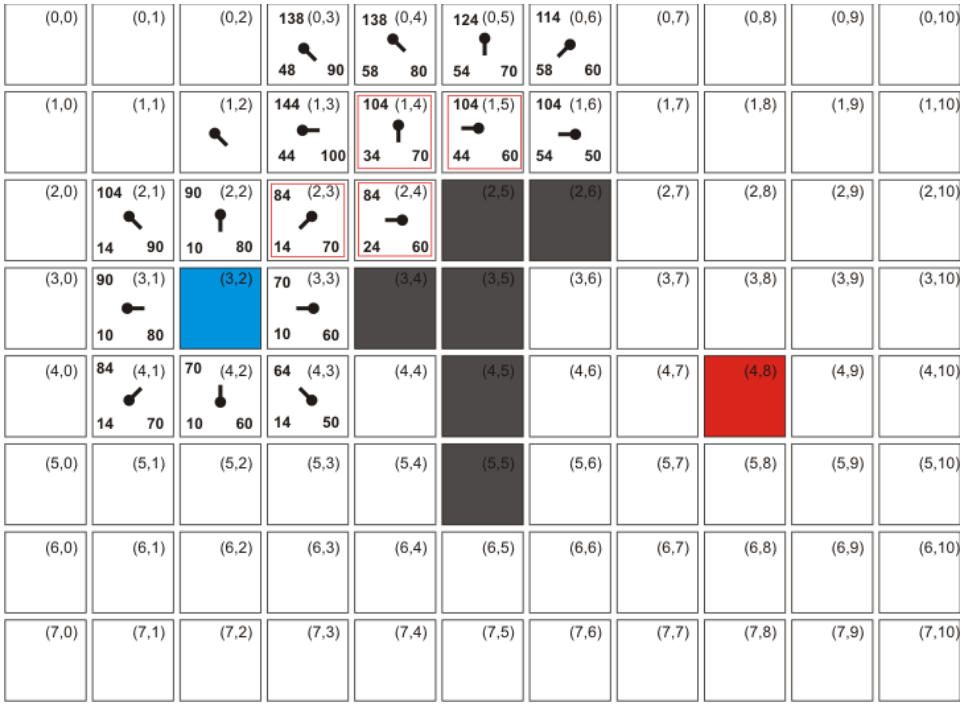
Open List	(0,3),(0,4),(0,5), (1,3)
Closed list	(2,3), (2,4),(2,5)
Parent	(2,4)
Current node	(1,5)

Berikut ini penjelasan melalui *tree* untuk pencarian jalan terpendek pada langkah keempat dapat dilihat pada gambar 3.13.



Gambar 3.13 Penjelasan A* dengan *tree* pada langkah keempat

Perhitungan pada langkah kelima yang berada pada *node* dengan koordinat (1,5) dapat dilihat pada gambar 3.14.



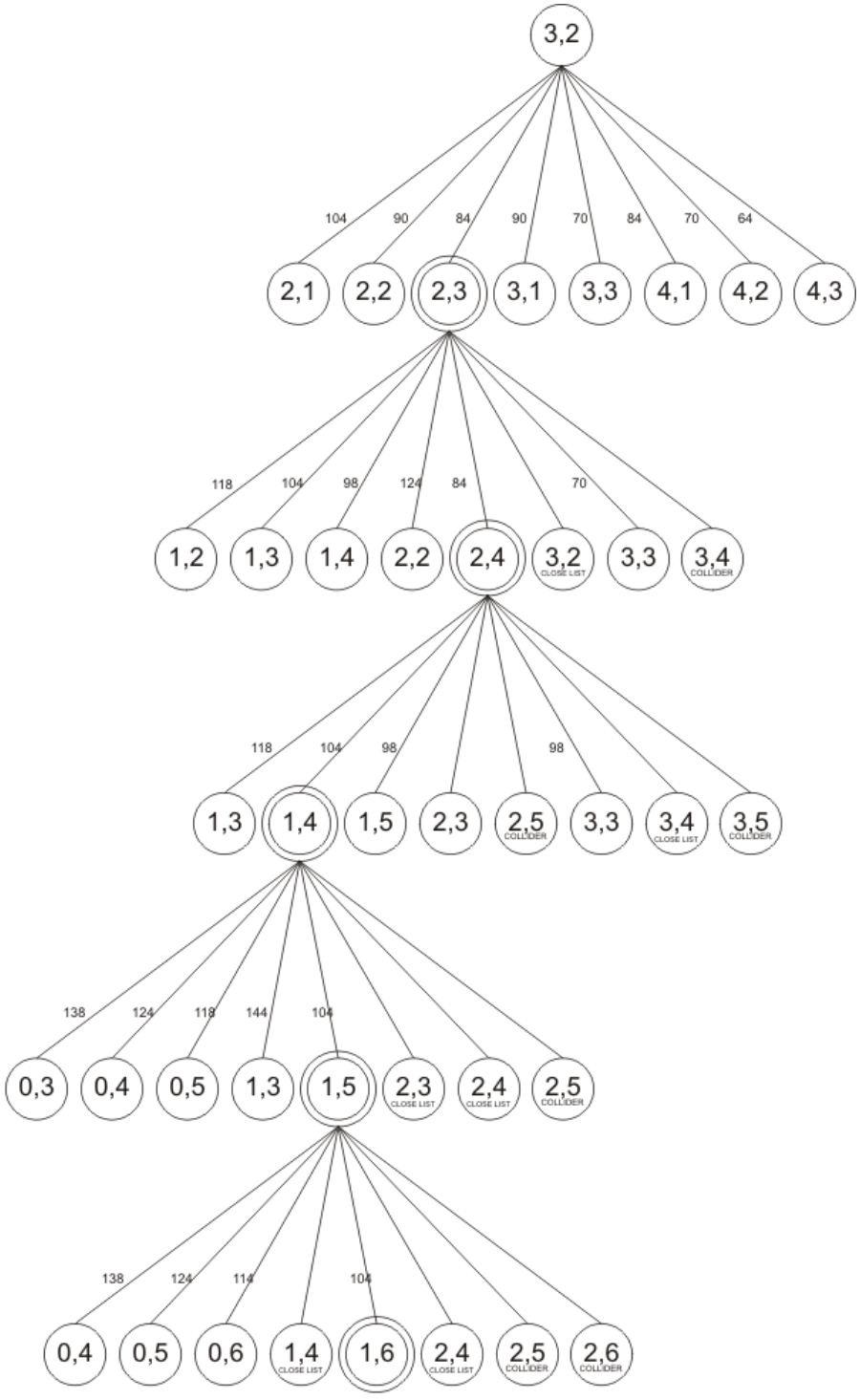
Gambar 3.14 Penjelasan Astar pada langkah kelima

Berikut ini penjelasan melalui tabel untuk pencarian jalan terpendek di langkah kelima dapat dilihat pada tabel 3.6.

Tabel 3.6 Tabel Perhitungan langkah kelima

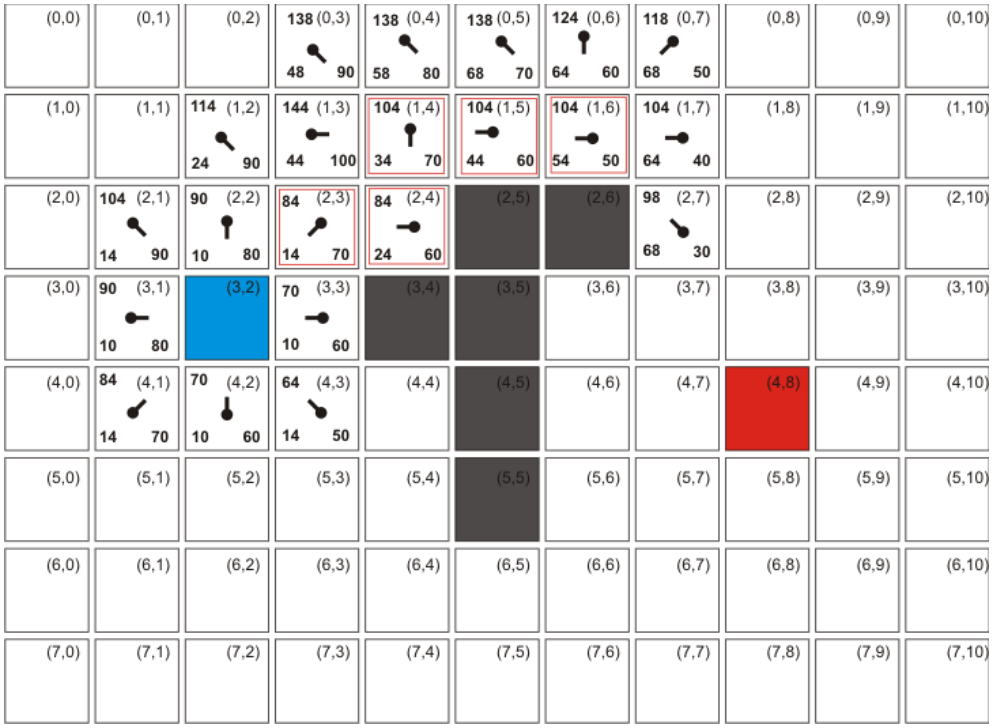
Open List	(0,4),(0,5),(0,6)
Closed list	(1,4),(2,4),(2,5),(2,6)
Parent	(1,5)
Current node	(1,6)

Berikut ini penjelasan melalui *tree* untuk pencarian jalan terpendek pada langkah kelima dapat dilihat pada gambar 3.15.



Gambar 3.15 Penjelasan A* dengan *tree* pada langkah kelima

Perhitungan pada langkah keenam yang berada pada *node* dengan koordinat (1,6) dapat dilihat pada gambar 3.16.



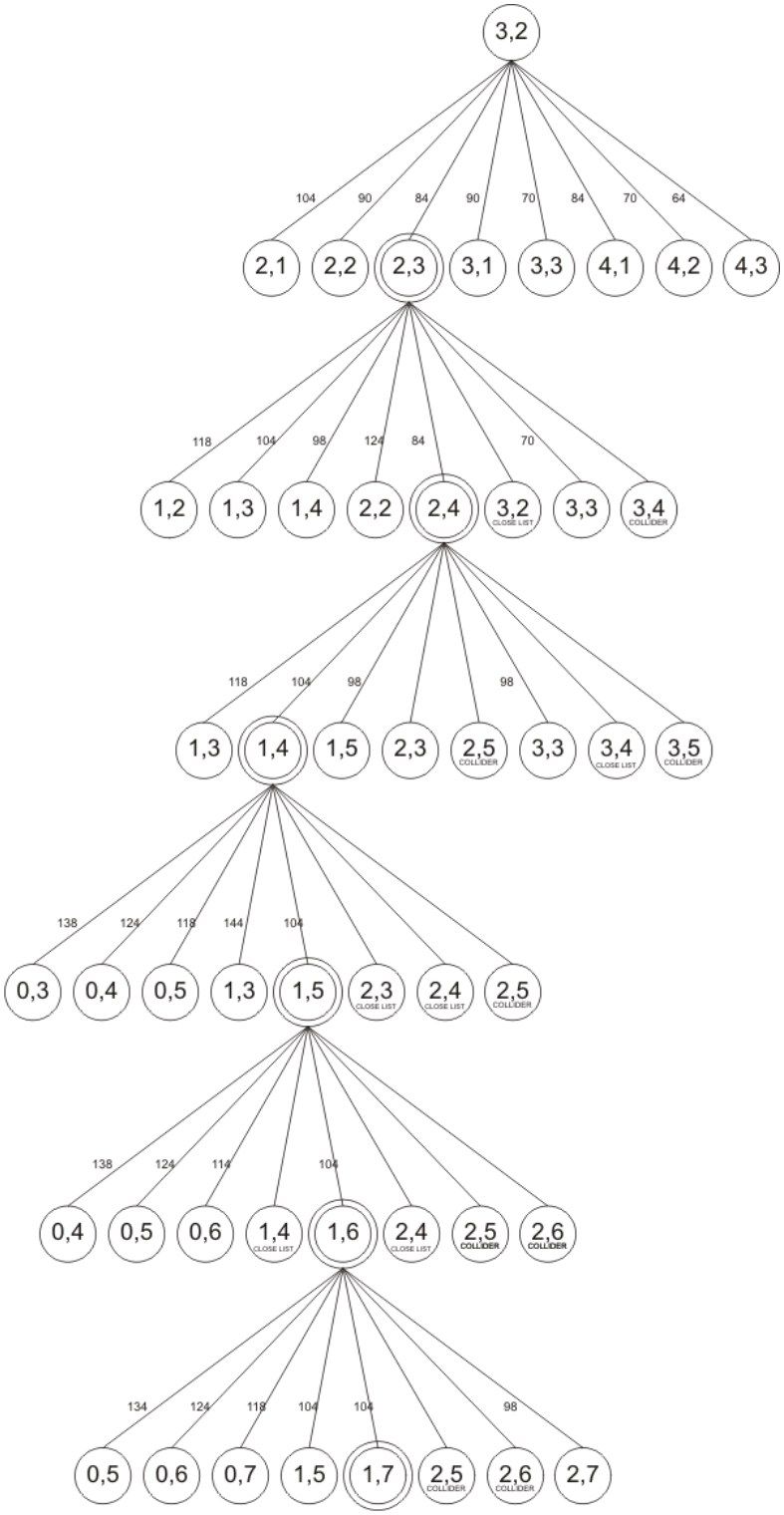
Gambar 3.16 Penjelasan Astar pada langkah keenam

Berikut ini penjelasan melalui tabel untuk pencarian jalan terpendek di langkah keenam dapat dilihat pada tabel 3.7.

Tabel 3.7 Tabel Perhitungan langkah keenam

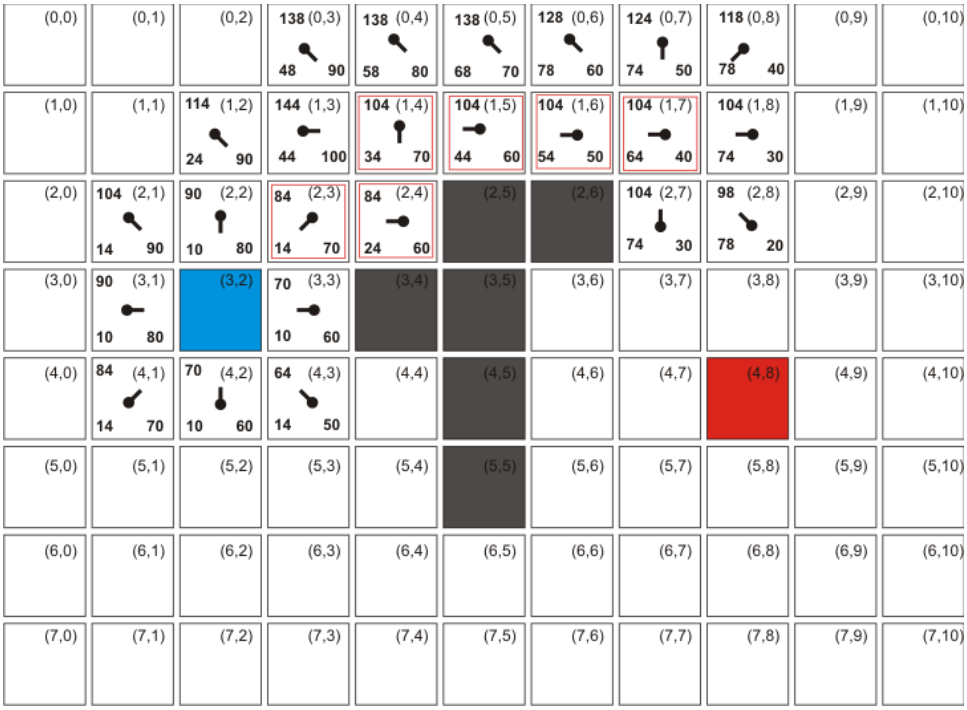
Open List	(0,5),(0,6),(0,7),(2,7)
Closed list	(1,5),(2,5),(2,6)
Parent	(1,6)
Current node	(1,7)

Berikut ini penjelasan melalui *tree* untuk pencarian jalan terpendek pada langkah keenam dapat dilihat pada gambar 3.17.



Gambar 3.17 Penjelasan A* dengan *tree* pada langkah keenam

Perhitungan pada langkah ketujuh yang berada pada *node* dengan koordinat (1,7) dapat dilihat pada gambar 3.18.



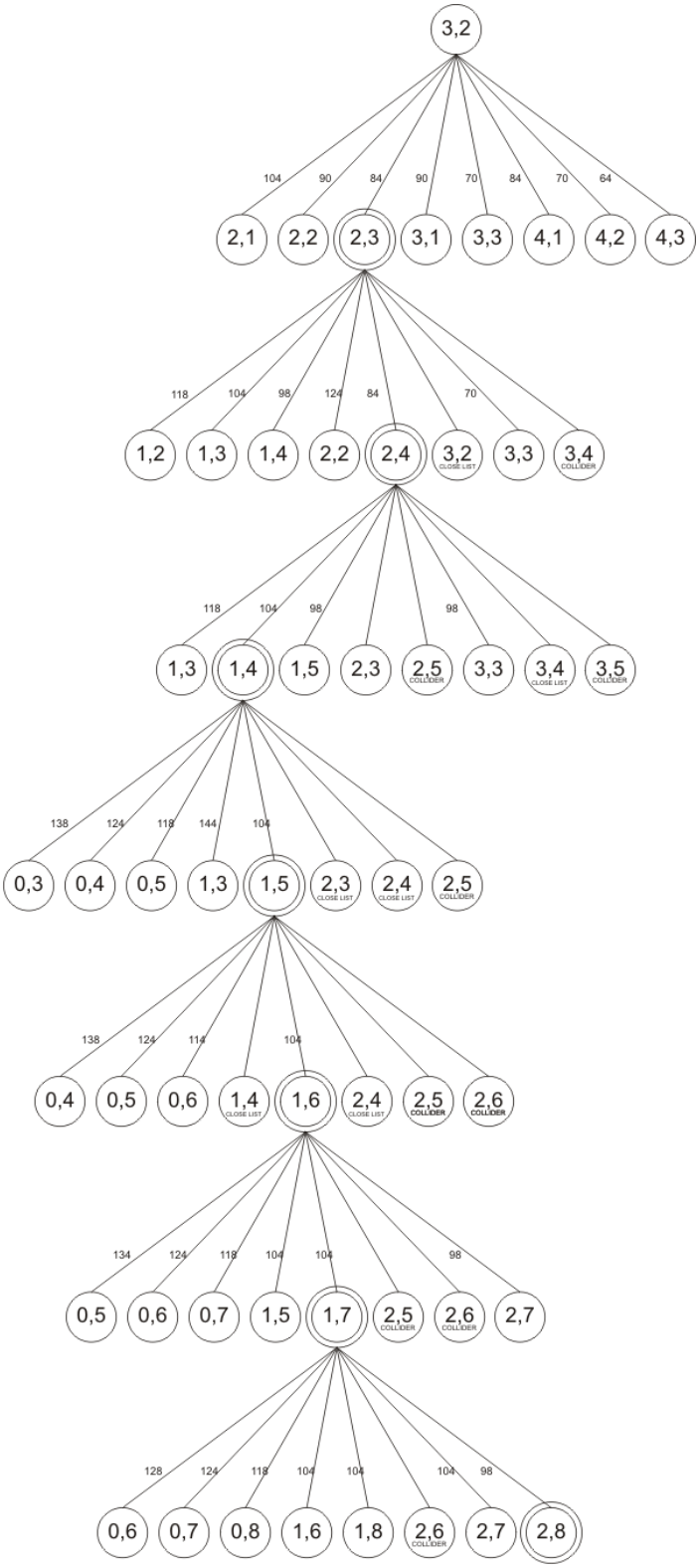
Gambar 3.18 Penjelasan Astar pada langkah ketujuh

Berikut ini penjelasan melalui tabel untuk pencarian jalan terpendek di langkah ketujuh dapat dilihat pada tabel 3.8.

Tabel 3.8 Tabel Perhitungan langkah ketujuh

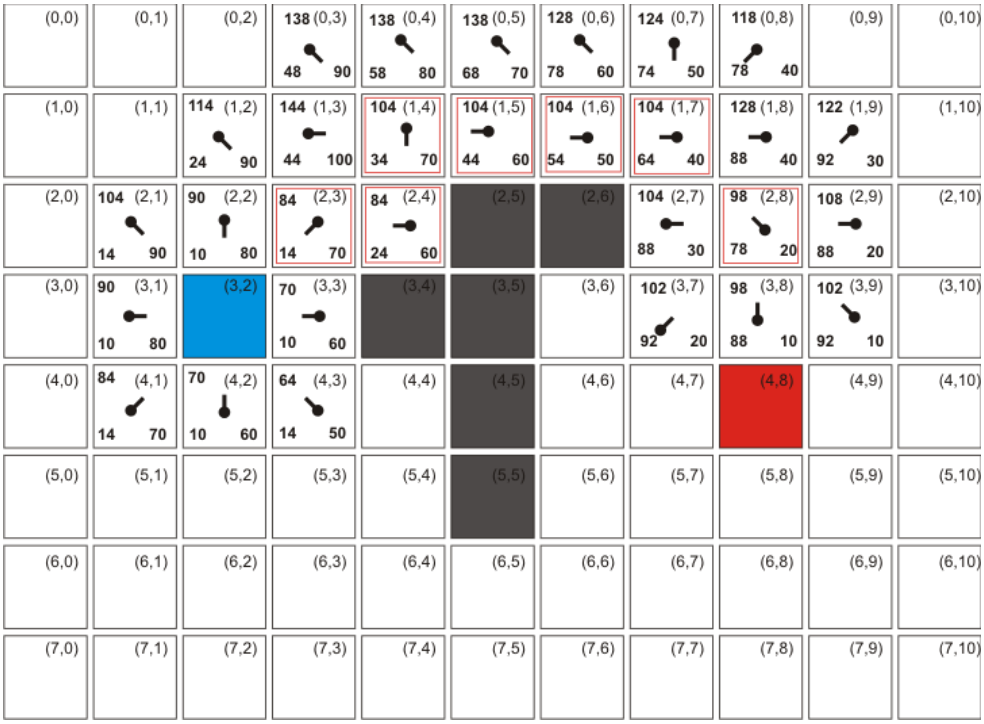
Open List	(0,6),(0,7),(0,8),(2,7),(2,8)
Closed list	(1.6),(2,6)
Parent	(1,7)
Current node	(2,8)

Berikut ini penjelasan melalui *tree* untuk pencarian jalan terpendek pada langkah ketujuh dapat dilihat pada gambar 3.19.



Gambar 3.19 Penjelasan A* dengan *tree* pada langkah ketujuh

Perhitungan pada langkah kedelapan yang berada pada *node* dengan koordinat (2,8) dapat dilihat pada gambar 3.20.



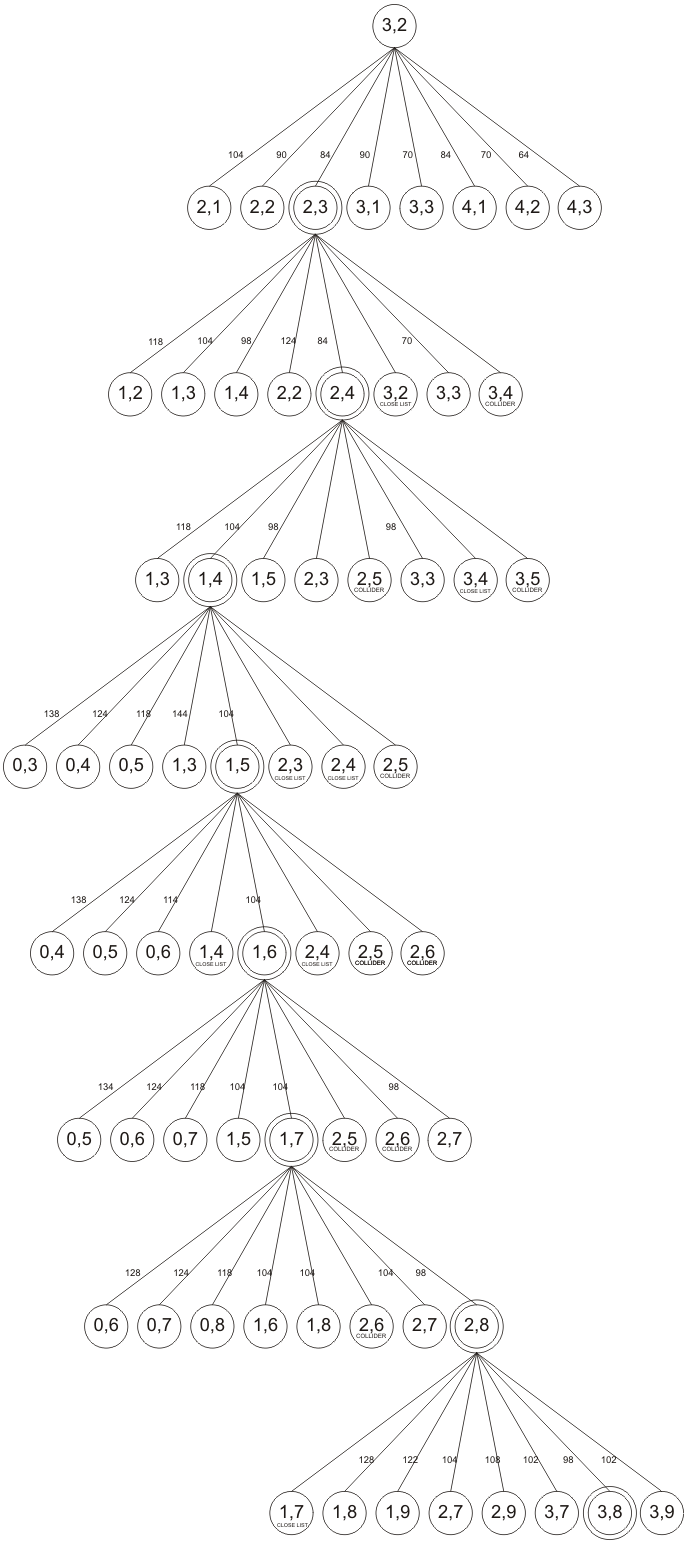
Gambar 3.20 Penjelasan Astar pada langkah kedelapan

Berikut ini penjelasan melalui tabel untuk pencarian jalan terpendek di langkah kedelapan dapat dilihat pada tabel 3.9.

Tabel 3.9 Tabel Perhitungan langkah kedelapan

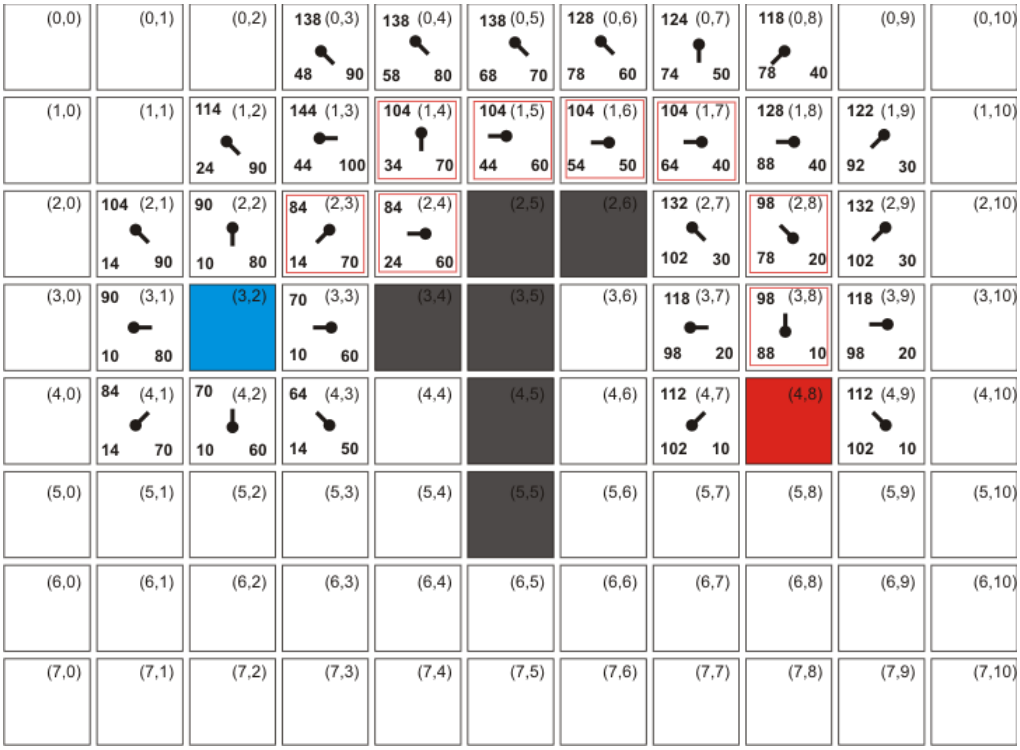
Open List	(1,8),(1,9),(2,7),(2,9),(3,7),(3,9)
Closed list	(1,7)
Parent	(2,8)
Current node	(3,8)

Berikut ini penjelasan melalui *tree* untuk pencarian jalan terpendek pada langkah kedelapan dapat dilihat pada gambar 3.21.



Gambar 3.21 Penjelasan A* dengan *tree* pada langkah kedelapan

Perhitungan pada langkah kesembilan yang berada pada *node* dengan koordinat (3,8) dapat dilihat pada gambar 3.22.



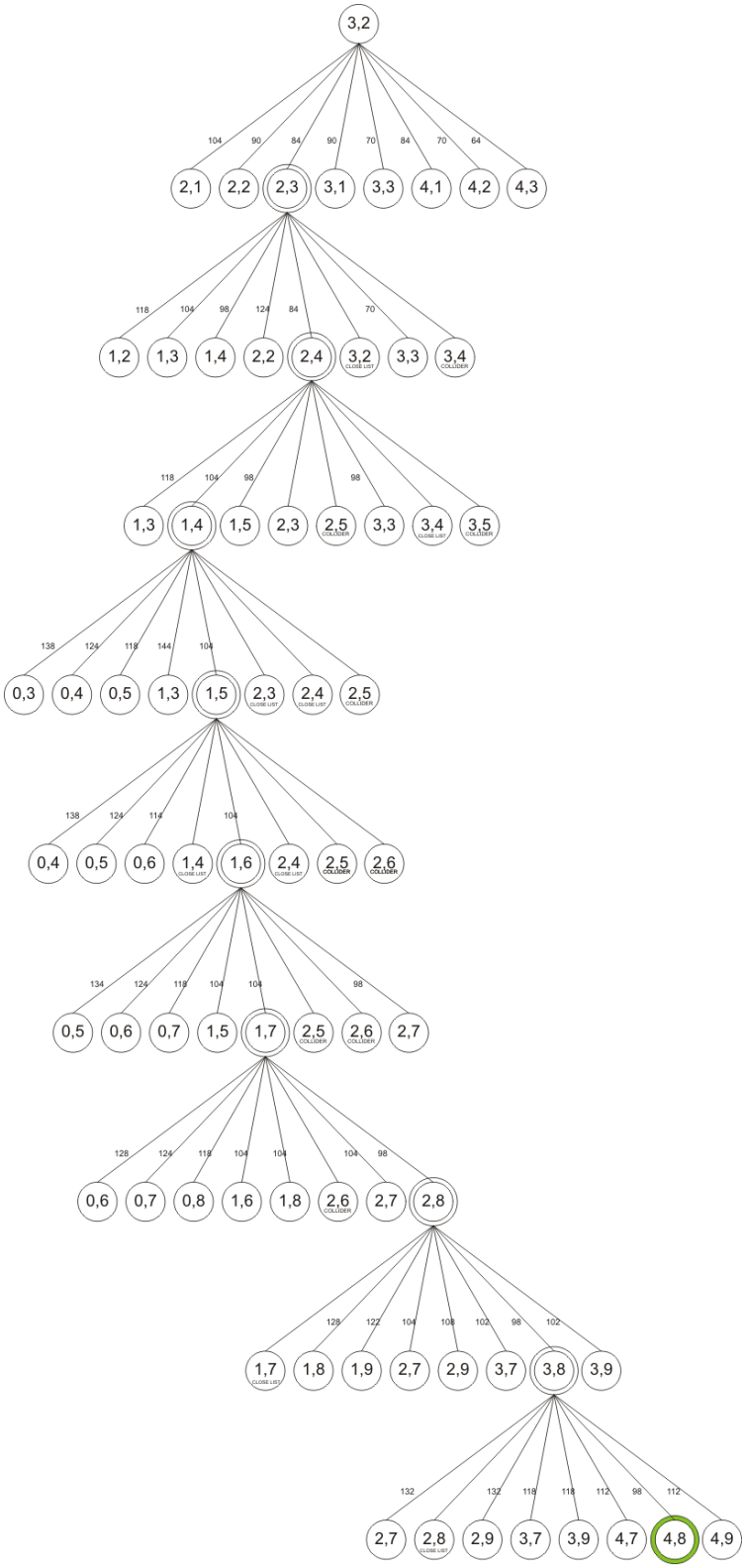
Gambar 3.22 Penjelasan Astar pada langkah kesembilan

Berikut ini penjelasan melalui tabel untuk pencarian jalan terpendek di langkah kesembilan dapat dilihat pada tabel 3.10.

Tabel 3.10Tabel Perhitungan langkah kesembilan

Open List	(2,7),(2,9),(3,7),(3,9),(4,7),(4,9)
Closed list	(2,8)
Parent	(3,8)
Current node	(4,8) goal

Berikut ini penjelasan melalui *tree* untuk pencarian jalan terpendek pada langkah kesembilan dapat dilihat pada gambar 3.23.



Gambar 3.23 Penjelasan A* dengan *tree* pada langkah kesembilan

3.2 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 lunak, analisis kebutuhan perangkat keras, dan analisis pengguna.

3.2.1 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 Pembangunan *Game* First Person Shooter 3D Palagan Ambarawa Berbasis Desktop ini adalah sebagai berikut :

1. Sistem operasi Windows 7.
2. Unity *version* 3.5.0 digunakan sebagai *engine* untuk membuat *game*.
3. 3ds Max *version* 2009 digunakan untuk pembuatan karakter .

3.2.2 Analisis Kebutuhan Perangkat Keras

Perangkat keras yang sesuai dengan kebutuhan aplikasi sangat diperlukan agar aplikasi dapat berjalan dengan baik. Spesifikasi minimum perangkat keras yang dibutuhkan agar dapat menjalankan *game* Palagan Ambarawa ini adalah sebagai berikut:

1. Processor 2,13 Ghz
2. RAM 512 MB
3. VGA 512 MB
4. Harddisk 20 GB
5. Mouse dan Keyboard

Perangkat keras minimum bagi pengguna untuk memainkan *game* ini adalah sebagai berikut :

1. Processor 1,80 Ghz
2. RAM 256 MB
3. VGA 512 MB
4. Harddisk 20 GB
5. Mouse dan Keyboard

3.2.3 Analisis User

Pada analisis *user* (pemakai) ini akan mencakup analisis beberapa parameter terhadap calon *user* dari applikasi.

3.2.3.1 User Knowledge and Experience

Game Palagan Ambarawa ini bisa digunakan oleh kalangan apapun, tetapi pengetahuan dan pengalaman akan memudahkan *user* dalam penggunaanya. Terutama pengetahuan dan pengalaman dalam memainkan *game action* atau *first person shooter*. Tabel 3.11 berikut menjelaskan analisis klasifikasi *knowledge and experience* dari pengguna aplikasi.

Tabel 3.11 Analisis klasifikasi *knowledge and experience*

Educational *level*	**Reading *Level***	**Typing Skills**
Bisa digunakan oleh berbagai kalangan, seperti pelajar, mahasiswa hingga masyarakat awam	Bisa digunakan oleh berbagai *level* pendidikan dengan reading *level* yang sedang	Average (40 wpm) Tidak memerlukan typing skills yang tinggi
Computer Literacy	**Task Experience**	**System Experience**
Moderate (menengah)	Bisa digunakan oleh dengan pengalaman penggunaan computer dan *game* yang sedang	Bisa digunakan oleh dengan pengalaman penggunaan komputer dan *game* yang sedang

Application Experience	**Native Language**	**Use Of Other System**
Bisa digunakan dalam semua sistem operasi Windows xp/vista/7	Menggunakan satu bahasa, yakni Indonesia	Bisa di jalankan tanpa perlu install applikasi lain

3.2.3.2 Users Physical Characteristic

Keadaan fisik seseorang mungkin akan berpengaruh pada penggunaan aplikasi *game* ini. Ada hal-hal yang harus diperhatikan juga terhadap user dari karakteristik fisiknya untuk dapat menggunakan aplikasi ini yaitu, Color Blind, Handednes, dan Gender seperti terlihat pada tabel 3.12.

Tabel 3.12 Analisis *Users Physical Characteristic*

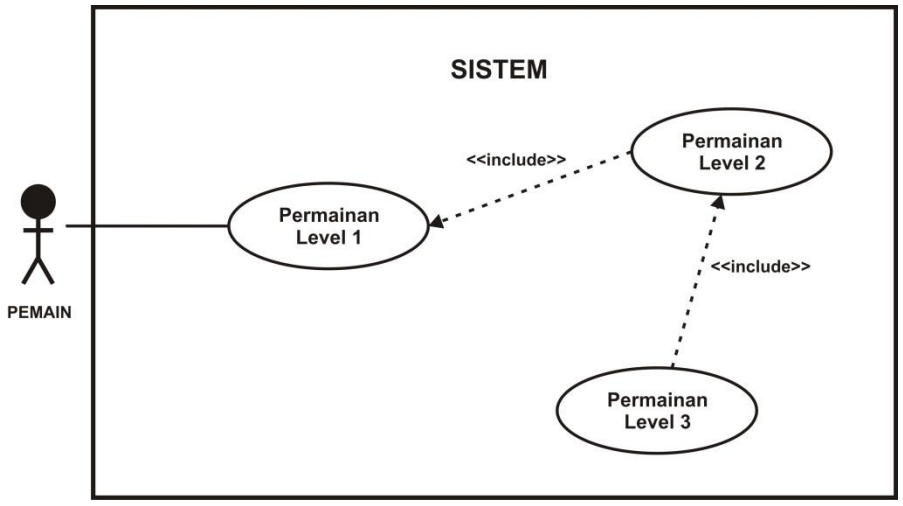
Usia	17 tahun keatas
Jenis Kelamin	Pria dan Wanita
Tangan	Kanan dan Kiri
Buta Warna	User yang tidak bisa membedakan warna yang satu dengan yang lainnya (buta warna) masih mampu menggunakan aplikasi ini, karena tidak ada indicator warna-warna khusus yang membedakan antara fungsional yang satu dengan fungsional yang lainnya. Akan tetapi penggunaannya tidak akan optimal karena dalam *game* ini terdapat banyak sekali perbedaan warna yang menunjang interaksi dan ketertarikan dalam permainan.

3.3 Analisis Kebutuhan Fungsional

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

3.3.1 *Use Case Diagram*

Usecase Diagram 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 *usecase diagram* untuk *game* 3D Palagan Ambarawa dapat dilihat pada gambar 3.24.



Gambar 3.24 *Use case Diagram Game* Palagan Ambarawa

3.3.2 Definisi *Actor*

Definisi *Actor* berfungsi untuk menjelaskan *Actor* yang terdapat pada *Use Case Diagram*. Definisi *Actor* diterangkan pada Tabel 3.13

Tabel 3.13 Definisi *Actor*

Actor	**Deskripsi**
Pemain	Orang yang akan memainkan *Game.*

3.3.3 Definisi *Use case*

Definisi *use case* berfungsi untuk menjelaskan proses yang terdapat pada setiap *use case*. Berikut definisi *use case* dapat dilihat pada tabel 3.14.

Tabel 3.14 *Definisi Use Case*

No.	*Use Case*	Deskripsi
1.	Permainan Level 1	Representasi Misi 1 (*level* 1).
2.	Permainan Level 2	Representasi Misi 2 (*level* 2).
3.	Permainan Level 3	Representasi Misi 3 (*level* 3).

3.3.4 *Skenario Use Case*

Use Case Skenario berisi mengenai skenario (*flow of event*) untuk beberapa *use case* utama, yang menggambarkan urutan interaksi *actor* dengan *use case* tersebut dari awal sampai akhir. *Use case* skenario yang terdapat pada pembangunan aplikasi ini antara lain sebagai berikut:

3.3.4.1 Skenario *Use Case* Permainan *Level* 1

Skenario *Use Case* Permainan *Level* 1 bertujuan untuk memperbolehkan pemain menuju permainan *Level 2*. Skenario *Use Case* diterangkan pada Tabel 3.15.

Tabel 3. 15 *Scenario Use Case Permainan Level* 1

Identifikasi	
Nama use case	Permainan *Level* 1
Aktor	Pemain
Tujuan	Memperbolehkan pemain menuju *level* 2
Kondisi Awal	Menampilkan permainan *level* 1
Kondisi Akhir	Pemain dinyatakan berhasil
Skenario	
Aksi Aktor	**Reaksi Sistem**
Skenario Normal	

a. Menekan tombol Main	
	b. Menampilkan informasi misi *level* 1
c. Menekan tombol mainkan sekarang	
	d. Menampilkan arena permainan *level* 1
e. Menggerakkan Tokoh Game	
	f. Tokoh game bergerak sesuai dengan arahan pemain
g. Menekan tombol tembak	
	h. Mengecek sasaran tembakan apakah terkena musuh atau tidak
	i. Melakukan pengecekan terhadap jumlah musuh yang dibunuh apakah sudah mencapai 12 atau lebih
	j. Memperbolehkan mengambil bendera dan masuk *level* 2

3.3.4.2 Skenario *Use Case* Permainan *Level*2

Skenario *Use Case* Permainan *Level* 2 bertujuan untuk menampilkan *Level2* permainan. Skenario *Use Case* diterangkan pada Tabel 3.16.

Tabel 3. 16 *Scenario Use Case Permainan Level* 2

Identifikasi	
Nama use case	Permainan *Level* 2
Aktor	Pemain
Tujuan	Memperbolehkan pemain menuju *level* 3
Kondisi Awal	Menyelesaikan misi di level 1
Kondisi Akhir	Pemain dinyatakan berhasil
Skenario	
Aksi Aktor	**Reaksi Sistem**
Skenario Normal	
a. Menyelesaikan misi di level 1	
	b. Menampilkan informasi misi *level* 2
c. Menekan tombol mainkan sekarang	
	d. Menampilkan arena permainan *level* 2
e. Menggerakkan Tokoh Game	
	f. Tokoh game bergerak sesuai dengan arahan pemain
g. Menekan tombol tembak	
	h. Mengecek sasaran tembakan apakah terkena musuh atau tidak

	i. Melakukan pengecekan terhadap jumlah musuh yang dibunuh apakah sudah mencapai 18 atau lebih
	j. Memperbolehkan mengambil bendera dan masuk *level* 3

3.3.4.3 Skenario *Use Case* Permainan *Level* 3

Skenario *Use Case* Permainan *Level* 3 bertujuan untuk menampilkan *Level 3* permainan. Skenario *Use Case* diterangkan pada Tabel 3.17.

Tabel 3. 17 *Scenario Use Case Permainan Level* 3

Identifikasi	
Nama use case	Permainan *Level* 2
Aktor	Pemain
Tujuan	Menampilkan *scene video* misi selesai
Kondisi Awal	Menyelesaikan misi di level 2
Kondisi Akhir	Pemain dinyatakan berhasil
Skenario	
Aksi Aktor	**Reaksi Sistem**
Skenario Normal	
a. Menyelesaikan misi di level 2	
	b. Menampilkan informasi misi *level* 3
c. Menekan tombol mainkan sekarang	
	d. Menampilkan arena permainan *level* 3
e. Menggerakkan Tokoh Game	
	f. Tokoh game bergerak sesuai dengan inputan musuh
g. Menekan tombol tembak	
	h. Mengecek sasaran tembakan apakah terkena musuh atau tidak
	i. Melakukan pengecekan terhadap jumlah musuh yang dibunuh apakah sudah mencapai 25 atau lebih
	j. Menampilkan scene video misi selesai

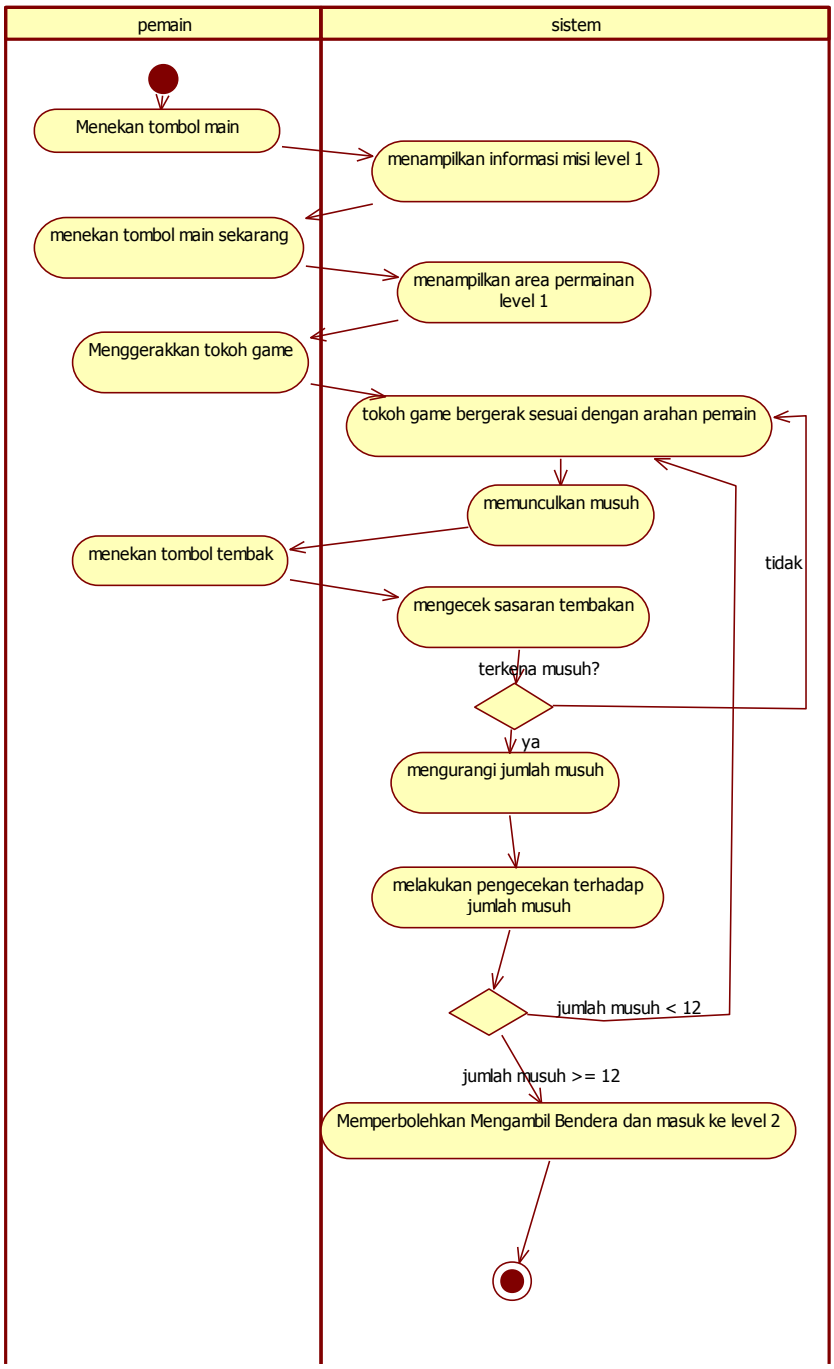
3.3.5 *Activity Diagram*

Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. *Activity diagram* merupakan *state diagram* khusus, dimana sebagian besar *state*

adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelum (*internal processing*).

3.3.5.1 *Activity Diagram* Permainan *Level* 1

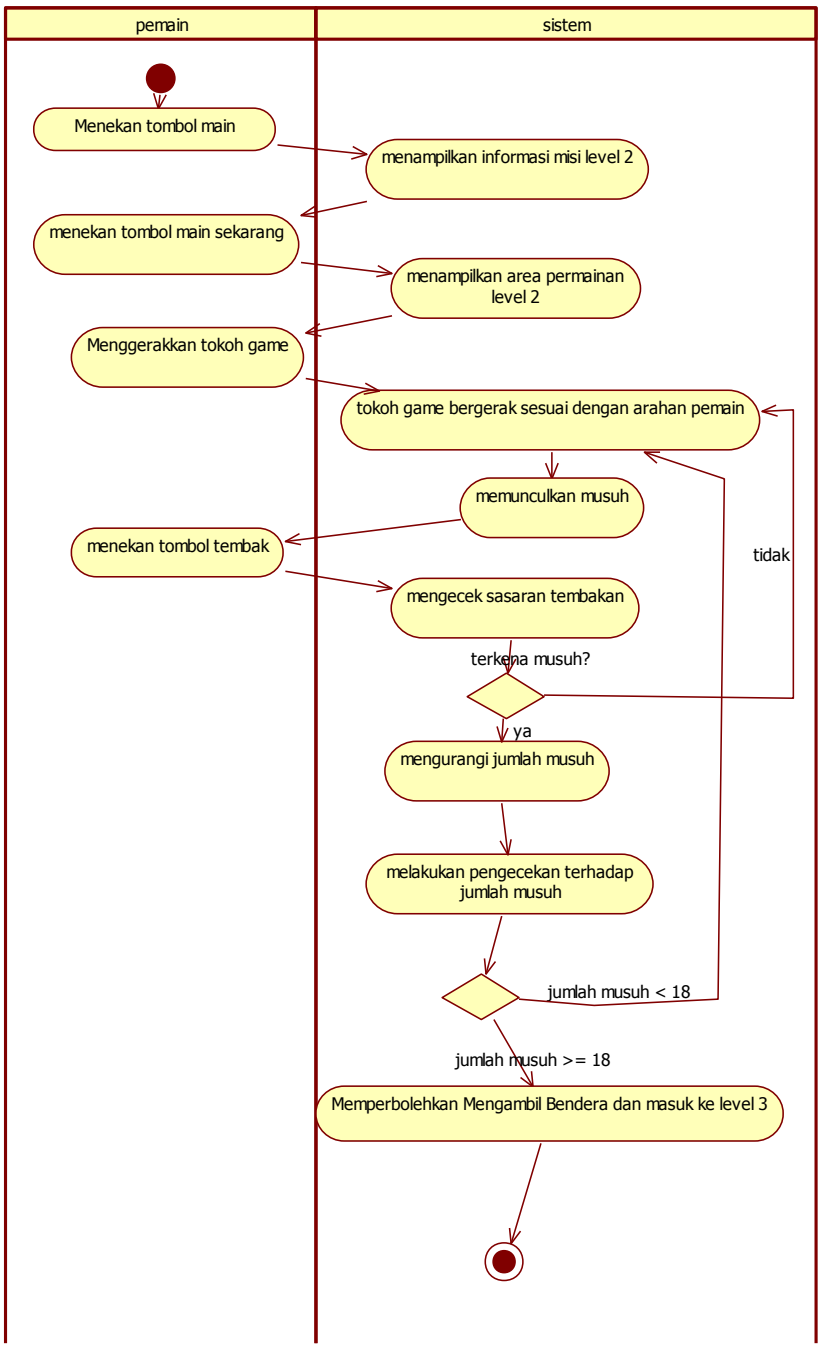
Aktivitas Diagram Informasi Permainan *Level* 1 adalah aktivitas permainan yang ada di *level* 1 yang dapat dilihat pada Gambar 3.25.



Gambar 3.25 *Activity diagram* Permainan *Level* 1

3.3.5.2 *Activity Diagram* Permainan *Level* 2

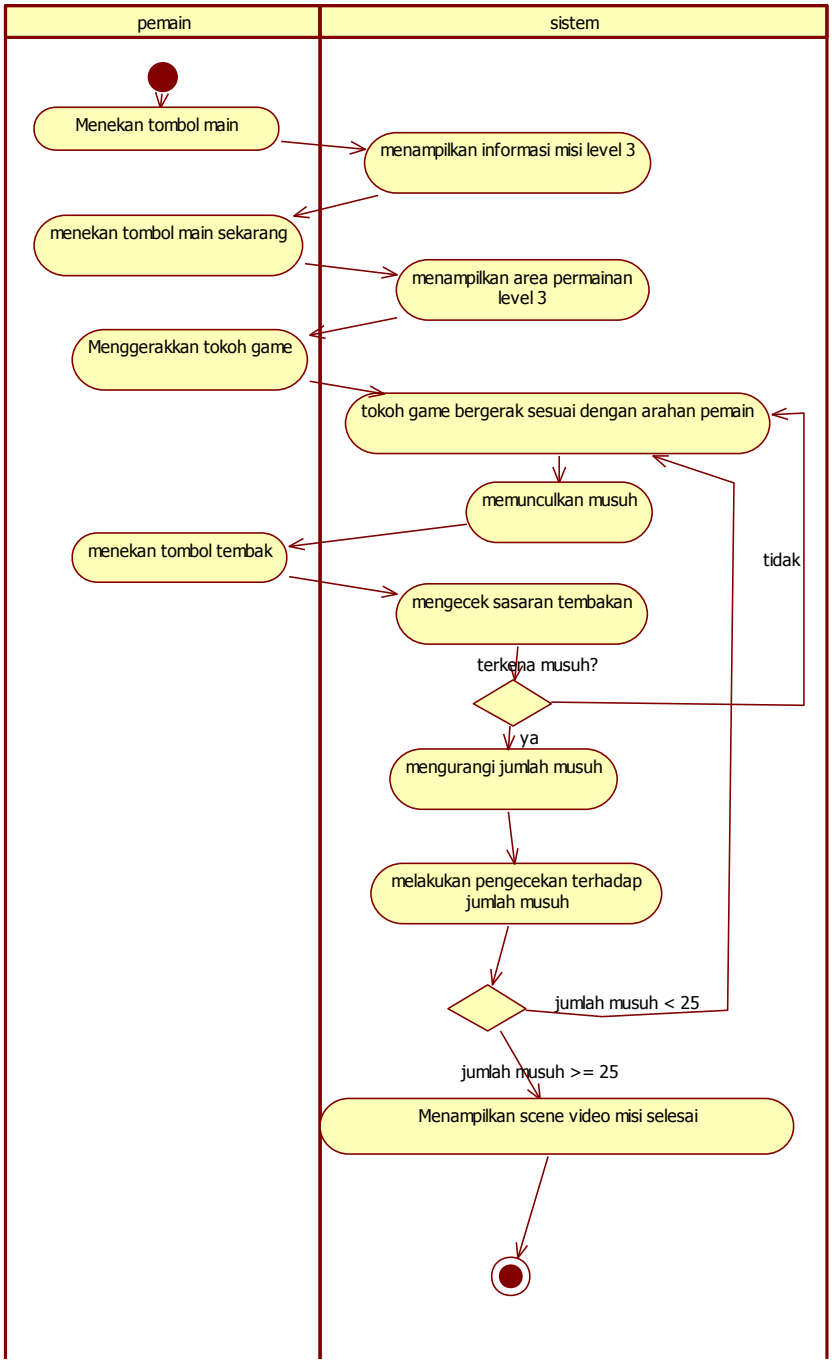
Aktivitas Diagram Informasi Permainan *Level* 2 adalah aktivitas permainan yang ada di *level* 2 yang dapat dilihat pada Gambar 3.26.



Gambar 3.26 *Activity diagram* Permainan *Level* 2

3.3.5.3 *Activity Diagram* Permainan *Level* 3

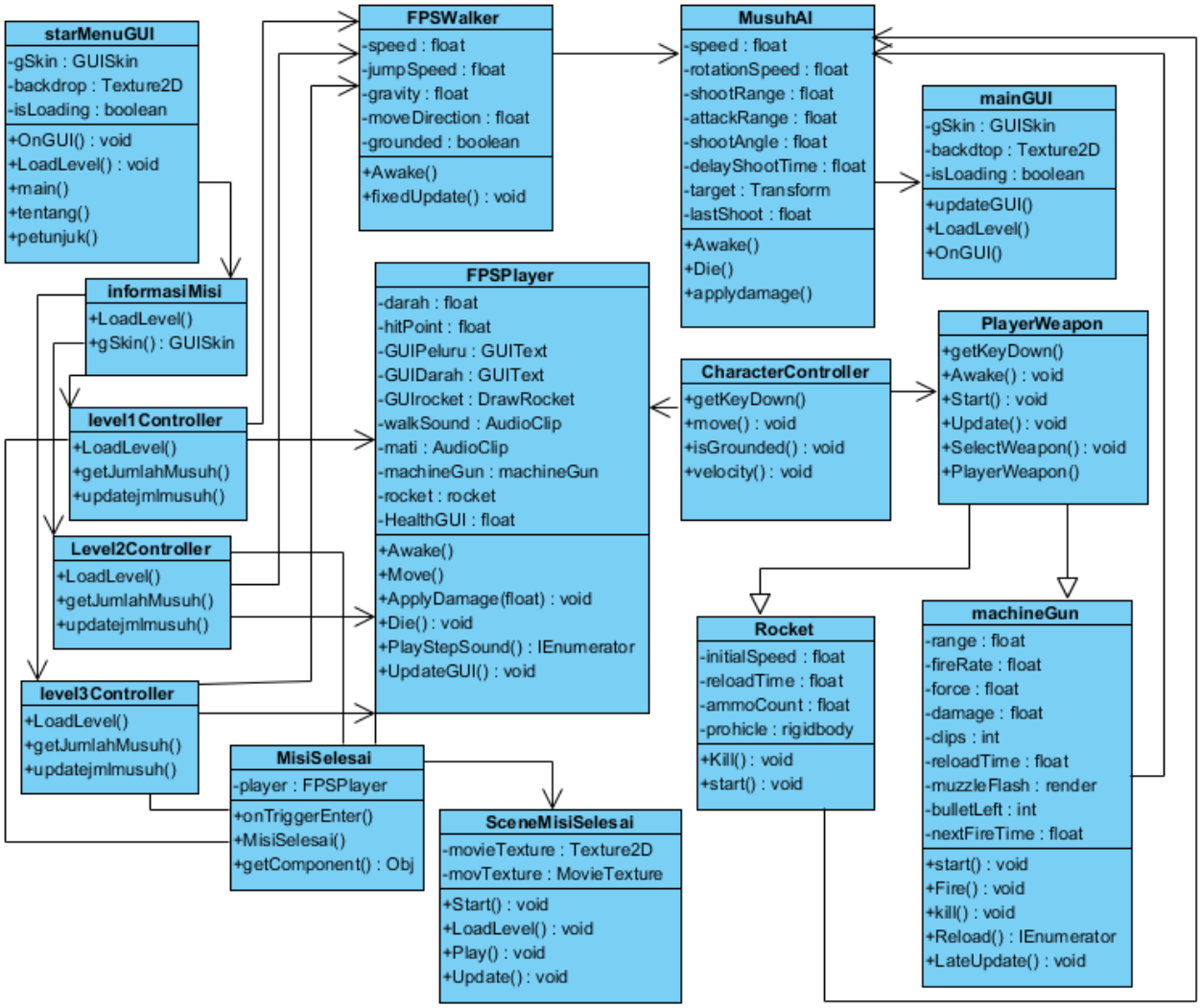
Aktivitas Diagram Informasi Permainan *Level* 3 adalah aktivitas permainan yang ada di *level* 3 yang dapat dilihat pada Gambar 3.27.



Gambar 3.27 *Activity diagram* Permainan *Level* 3

3.3.6 *Class Diagram*

Class diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang dibuat untuk membangun sistem. *Class diagram* pada *game* ini dapat dilihat pada gambar 3.28.



Gambar 3.28 *Class Diagram Game* Palagan Ambarawa

3.3.7 *Sequence Diagram*

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, display, dan sebagainya) berupa *message* yang

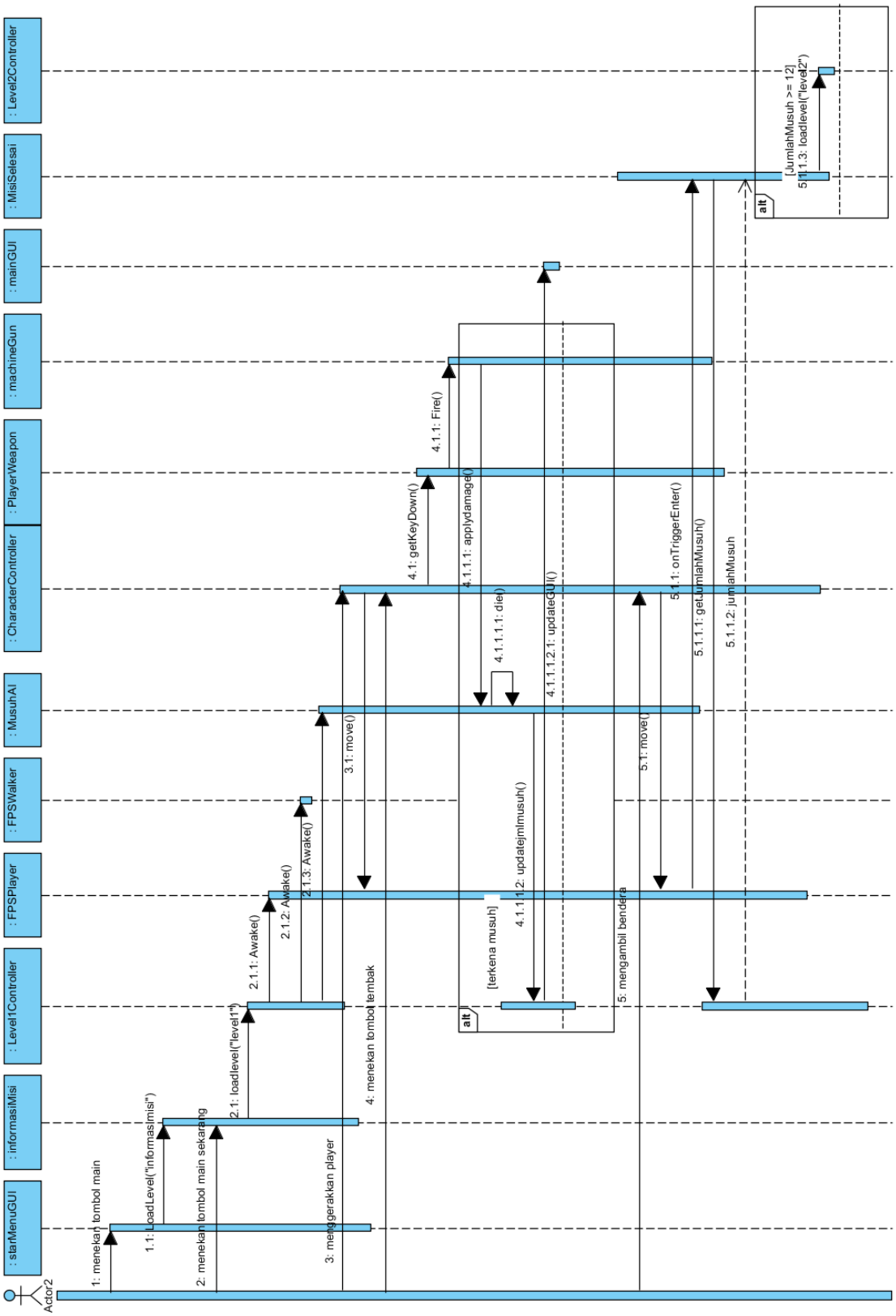
digambarkan terhadap waktu. *Sequence diagram* terdiri atas dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).

Sequence diagram biasa digunakan untuk menggambarakan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respon dari sebuah event untuk menghasilkan *output* tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan.

Berikut ini beberapa *sequencial diagram* yang terdapat pada *game* Palagan Ambarawa yaitu :

3.3.7.1 *Sequence Diagram* Permainan *Level* 1

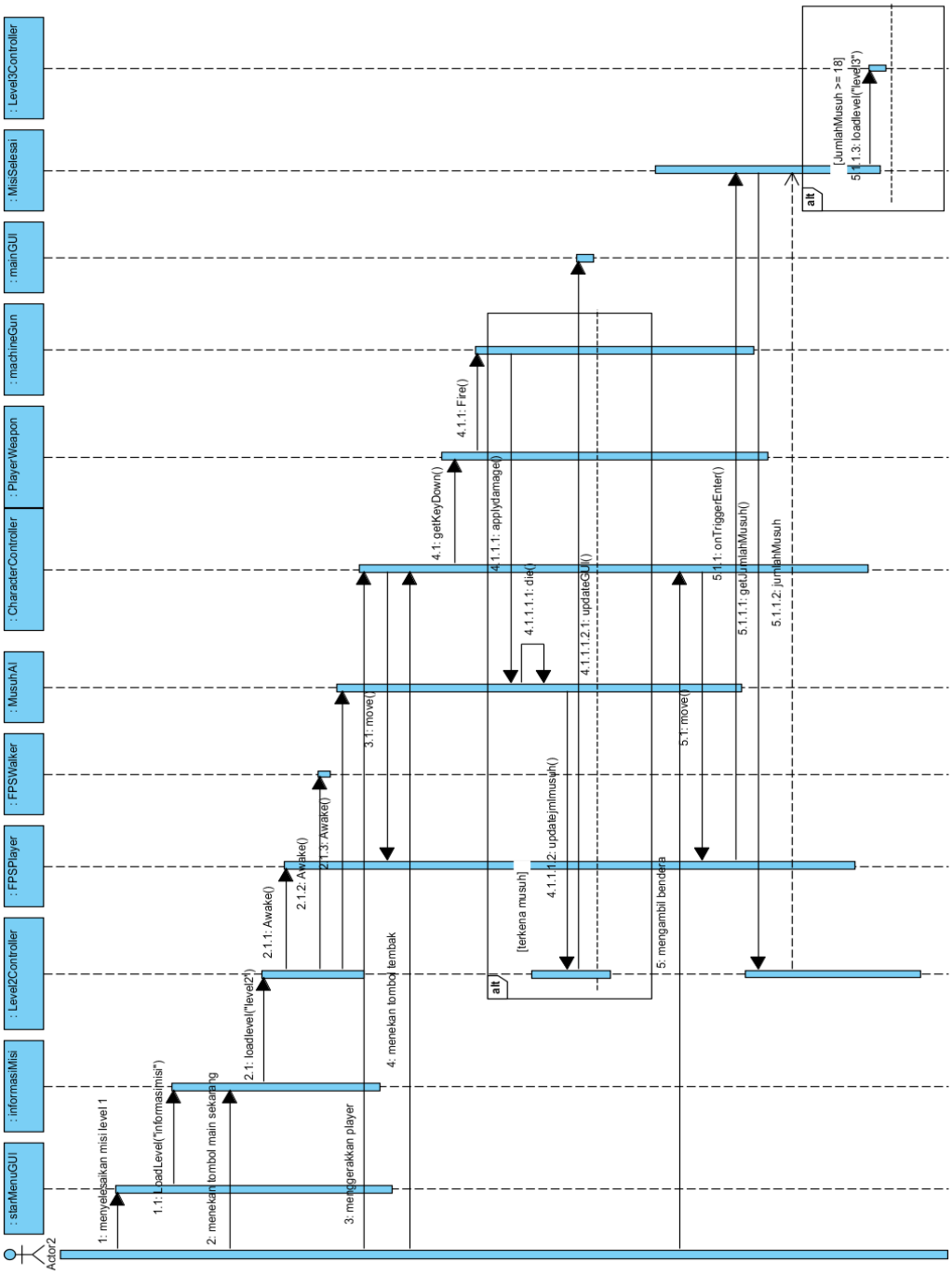
Sequence diagram Permainan *Level* 1menggambarkan interaksi antara objek dari *class* starMenuGUI dan objek yang berkaitan dengan *class* lainya. Keseluruhan *sequence* permainan dapat dilihat pada Gambar 3.29.



Gambar 3. 29 *Sequence Diagram* Permainan Level 1

3.3.7.2 *Sequence Diagram* Permainan *Level* 2

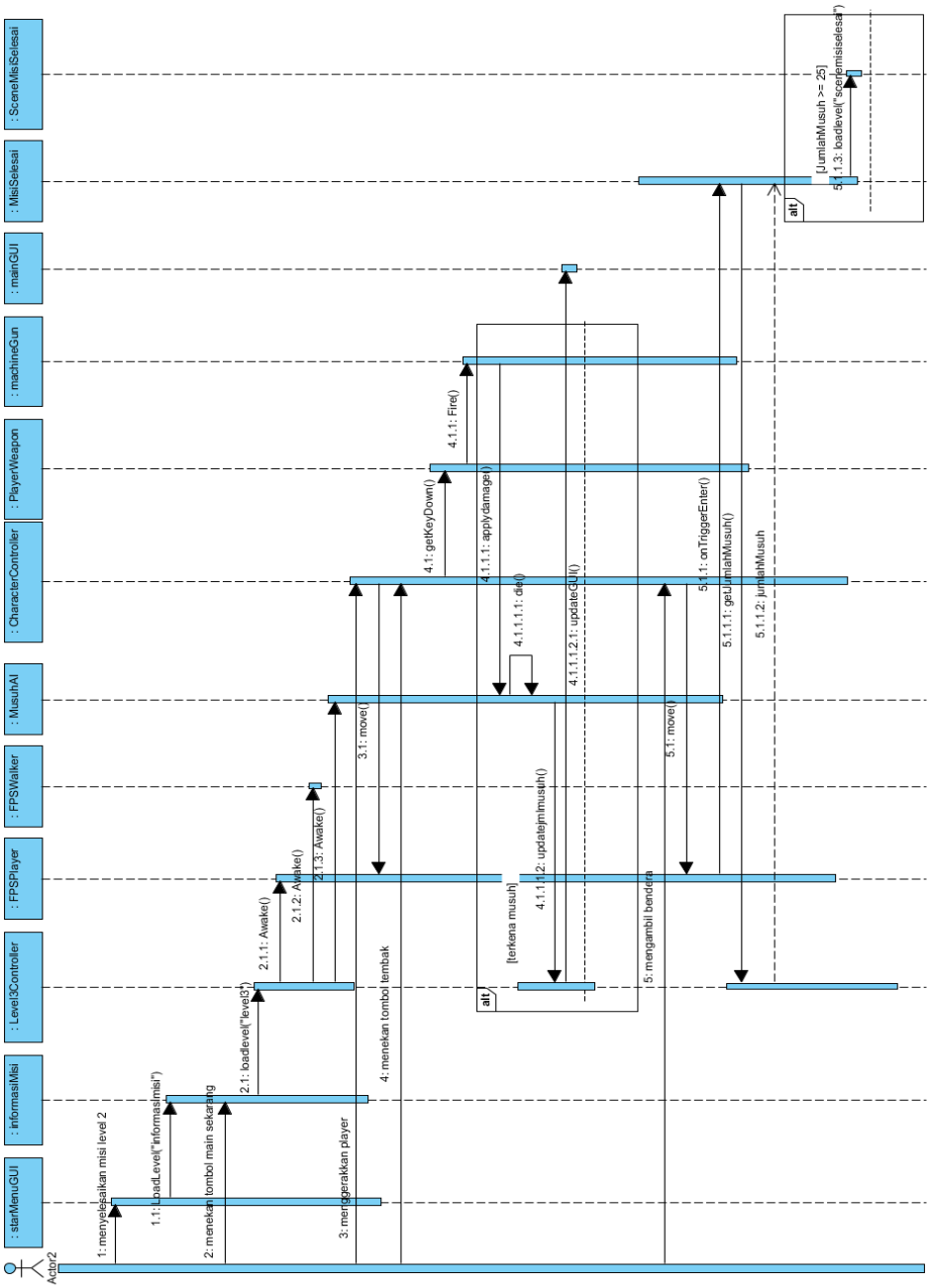
Sequence diagram Permainan *Level* 2 menggambarkan interaksi antara objek dari *class* starMenuGUI dan objek yang berkaitan dengan *class* lainya. Keseluruhan *sequence* permainan dapat dilihat pada Gambar 3.30.



Gambar 3. 30 *Sequence Diagram* Permainan Level 2

3.3.7.3 *Sequence Diagram* Permainan *Level* 3

Sequence diagram Permainan *Level* 3 menggambarkan interaksi antara objek dari *class* starMenuGUI dan objek yang berkaitan dengan *class* lainya. Keseluruhan *sequence* permainan dapat dilihat pada Gambar 3.31.



Gambar 3. 31 *Sequence Diagram* Permainan Level 3

3.4 Perancangan Sistem

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

3.4.1 Perancangan Komponen Permainan

Perancangan komponen permainan meliputi karakter dan *storyboard* yang didalamnya menjelaskan karakter beserta senjata apa saya yang dimiliki pemain maupun musuh (NPC) dan storyboard sebagai gambar perancangan pembuatan *game* Palagan Ambarawa.

3.4.1.1 Karakter

Karakter utama dalam *game* ini bernama Tarmidji seorang tentara yang dirancang untuk melawan para tentara sekutu. Tarmidji dilengkapi dengan persenjataan berupa satu senjata berupa laras panjang dan satu senjata berupa roket. Karena *game* ini merupakan *game* yang mempunyai *genre first person shooter* yang mempunyai sudut pandang orang pertama, maka karakter Tarmidji seperti tubuh, kepala, kaki dan tangan tidak diperlihatkan tetapi hanya memperlihatkan bagian moncong senjatanya saja. Properti yang dimiliki karakter utama dapat dilihat pada Tabel 3.37.

Tabel 3.37 properties karakter utama

Nama	Tarmidji
Darah	200
Kemampuan	Berlari, melompat, menembak
Senjata	Laras panjang dan Roket

Senjata yang digunakan oleh Tarmidji ada dua macam, pertama adalah senjata laras panjang yang memiliki peluru tidak terbatas dan yang kedua adalah roket yang mempunyai peluru terbatas sebanyak 40 yang dapat dilihat pada

Gambar 3.41 dan Gambar 3.42 dan *properties* senjata pada karakter dapat dilihat pada tabel 3.38.



Gambar 3.41 Laras Panjang



Gambar 3.42 Roket

Tabel 3.38

Properti	Laras Panjang	Roket
Damage	5	30
Jarak	50	100
Reload	0.5	1
Peluru	Unlimited	20

A. Karakter Musuh

Musuh yang terdapat pada *game* ini adalah tentara sekutu, tentara sekutu ini memiliki tingkat kesulitan untuk dikalahkan yang berbeda-beda disetiap *level* yang dilewati. Tentara-tentara Inggris ini memiliki satu senjata berupa laras panjang yang dapat dilihat pada Gambar 3.43.



Gambar 3.43 Karakter utama musuh

B. Karakter Teman

Karakter teman pada *game* ini terdapat satu yang bernama Piter, karakter teman ini akan secara otomatis mengikuti ketika *game* dimainkan pada *level* 2. Karakter teman dapat dilihat pada gambar 3.44 dan tabel properti musuh lebih jelasya dapat dilihat pada tabel 3.39.



Gambar 3.44 Karakter Teman

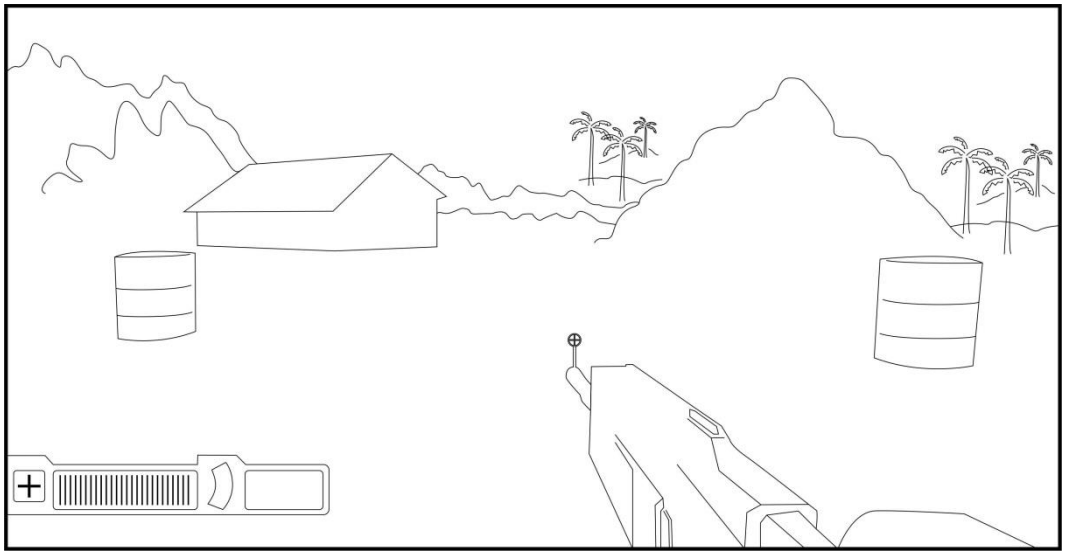
Tabel 3.39 Propertis Karakter Teman

Propertis	Karakter Teman
Kecepatan	5
Jarak Menembak	-
Jeda Menembak	-
Damage	-
Darah	100

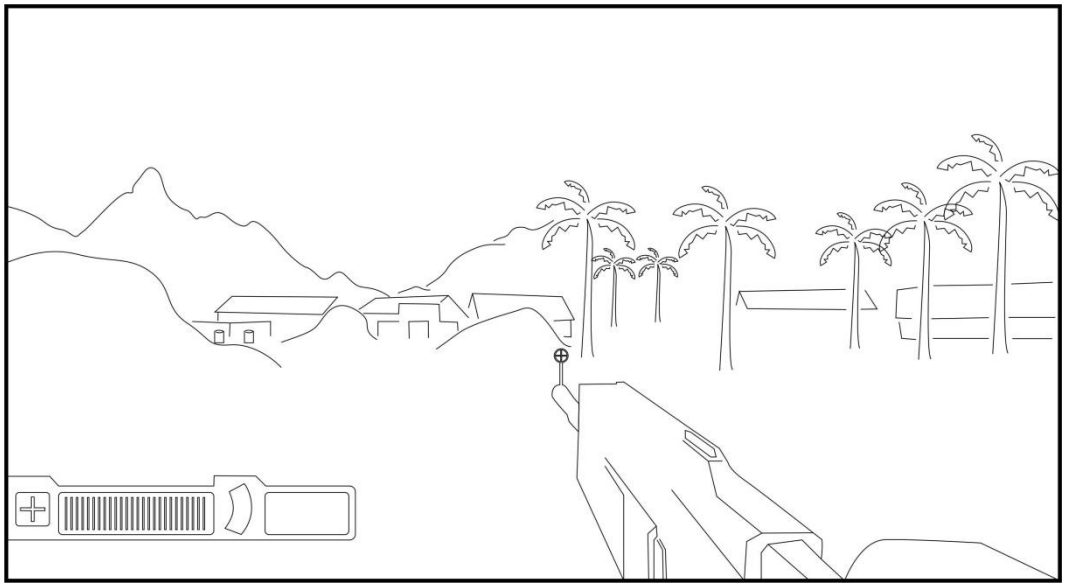
3.4.1.2 Story Board

Storyboard adalah sketsa gambar yang disusun berurutan sesuai dengan naskah, dengan menggunakan *storyboard* dapat menyampaikan ide cerita kepada orang lain dengan lebih mudah karena dapat dengan storyboard dapat mengimajinasikan khayalan yang sesuai dengan gambar-gambar yang ada sehingga dapat menghasilkan persepsi yang sama pada ide cerita yang dibuat.

Berikut beberapa *storyboard* dari *game first person shooter* 3D Palagan Ambarawa :

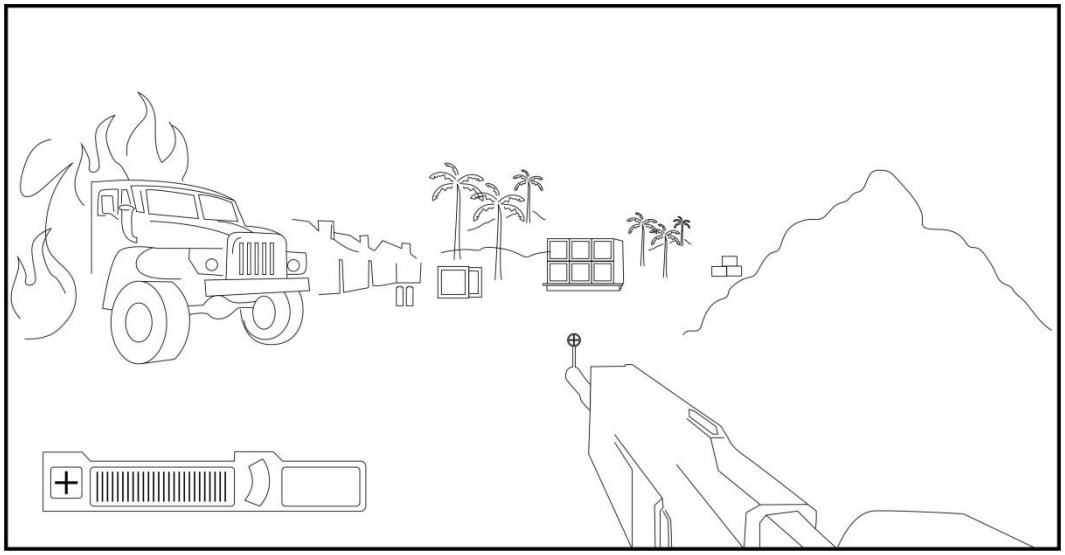


Gambar 3.45 *Storyboard level* 1-1

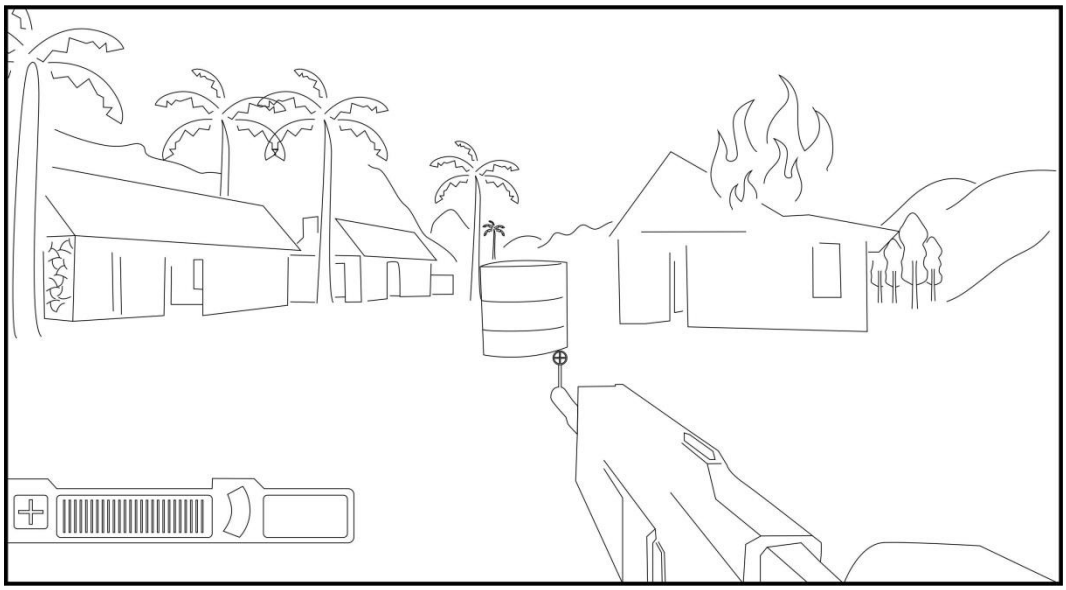


Gambar 3.46 *Storyboard level* 1-2

Gambar 3.45 menceritakan ketika pemain memulai permainan di *level* 1 dengan yang berada di perkampungan di Magelang Jawa Tengah dan disinilah awal pertempuran terjadi. Gambar 3.46 menceritakan ketika pemain memainkan permainan untuk menyelesaikan setiap misi yang ada pada *level* 1.



Gambar 3.47 *Storyboard level* 2-1



Gambar 3.48 *Storyboard level* 2-2

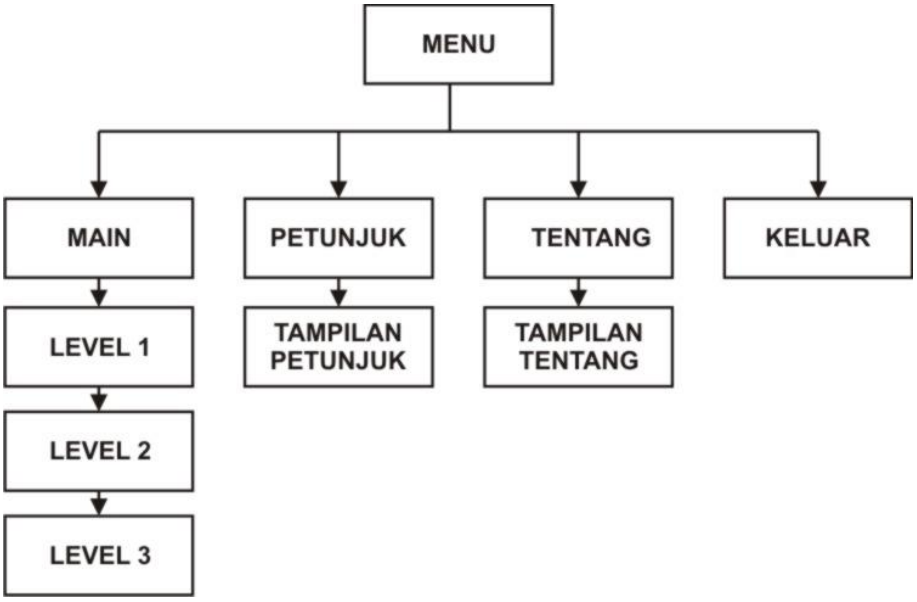
Gambar 3.47 menceritakan ketika pemain memulai permainan di *level* 2 dengan yang berada di perkampungan di daerah pasir jambu dan pertempuran di *level* 2 terjadi pada malam hari sehingga terlihat kobaran api yang membakar rumah-rumah sehingga dapat membuat situasi peperangan lebih menyerupai latar aslinya. Gambar 3.48 menceritakan ketika pemain memainkan permainan untuk menyelesaikan setiap misi yang ada pada *level* 2.

3.4.2 Perancangan Arsitektur

Setelah melakukan perancangan data pada sistem yang dibangun, maka dilakukanlah perancangan arsitektur. Perancangan arsitektur yang telah dibuat meliputi beberapa perancangan diantaranya struktur menu, perancangan antarmuka

3.4.2.1 Perancangan Struktur Menu

Perancangan menu dibuat dengan harapan agar pengguna dapat mengunakan *Game* dengan mudah tanpa kesulitan, sehingga memudahkan pemakai dalam memilih menu aplikasi yang sedang berjalan.Srtuktur menu dapat dilihat pada gambar 3.49.



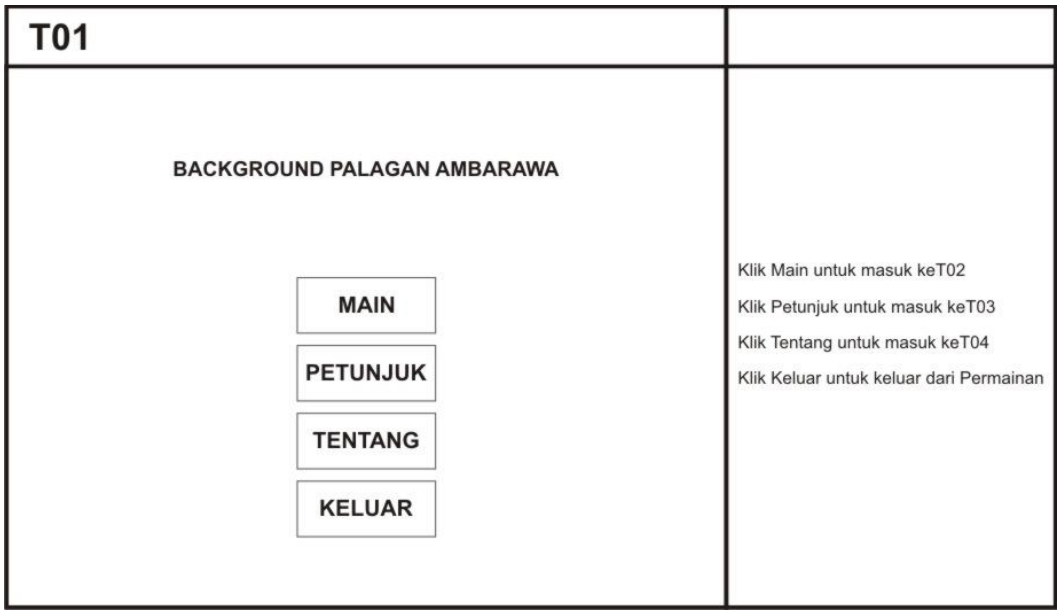
Gambar 3.49 Struktur menu utama

3.4.3 Perancangan Antarmuka

Setelah melakukan perancangan data pada sistem yang dibangun, maka dilakukanlah perancangan arsitektur. Perancangan arsitektur yang telah dibuat meliputi beberapa perancangan diantaranya struktur menu, perancangan antarmuka.

3.4.3.1 Antarmuka Menu Utama

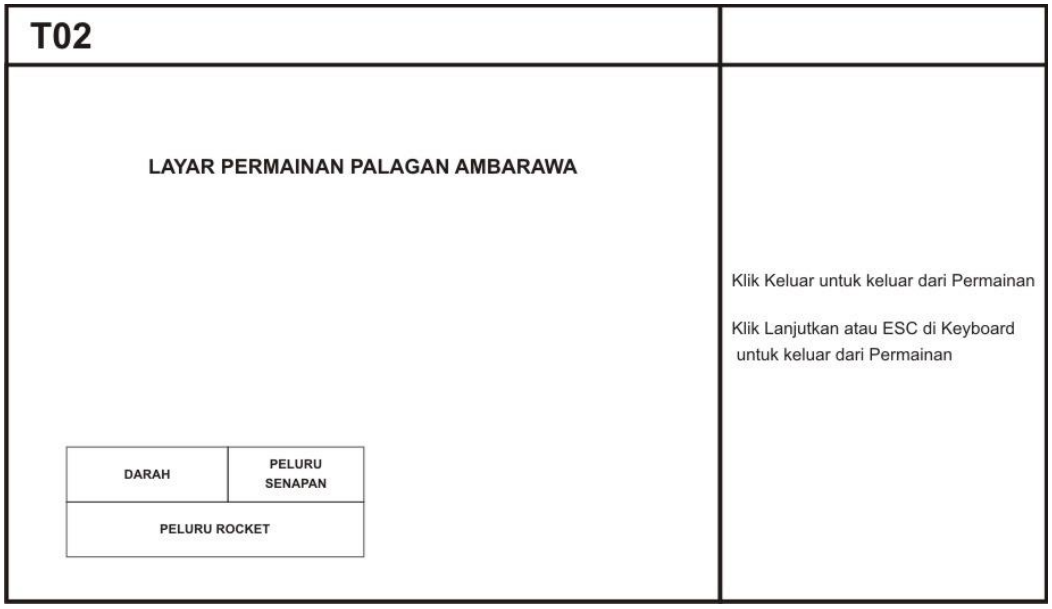
Perancangan antarmuka menu utama dari *game* Palagan Ambarawa dapat dilihat pada gambar 3.50.



Gambar 3.50 Antarmuka Menu

3.4.3.2 Antarmuka Permainan

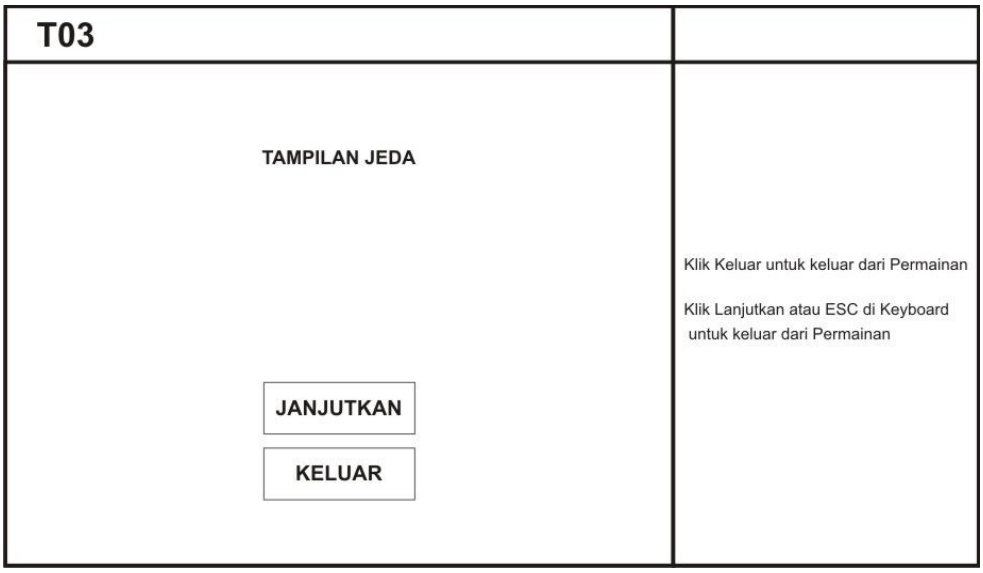
Perancangan antarmuka permainan dari *game* Palagan Ambarawa dapat dilihat pada gambar 3.51.



Gambar 3.51 Antarmuka Permainan

3.4.3.3 Antarmuka Jeda

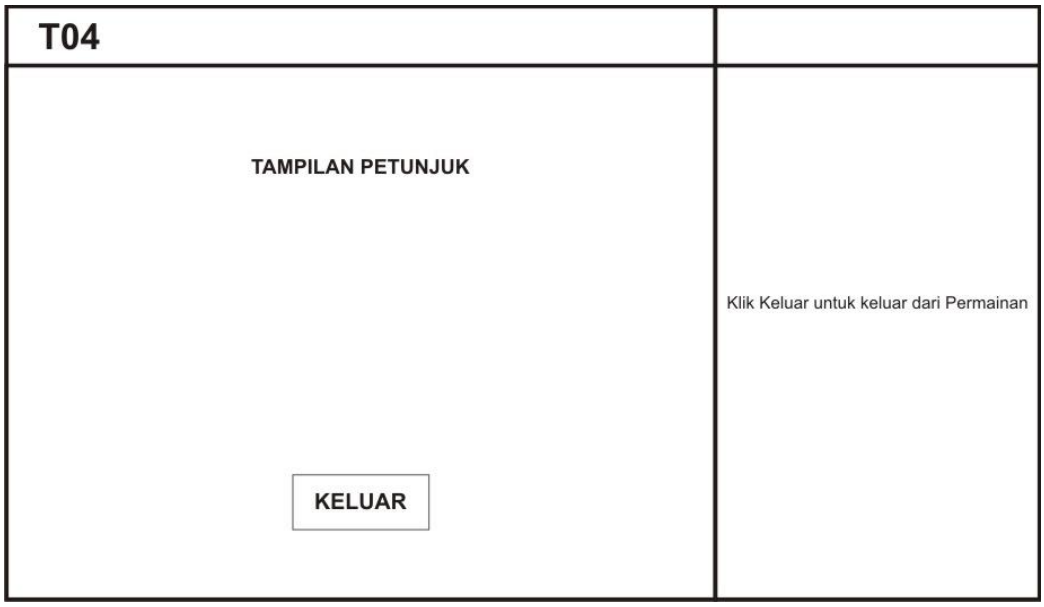
Perancangan antarmuka jeda dari *game* Palagan Ambarawa dapat dilihat pada gambar 3. 52.



Gambar 3.52 Antarmuka Jeda

3.4.3.4 Antarmuka Petunjuk

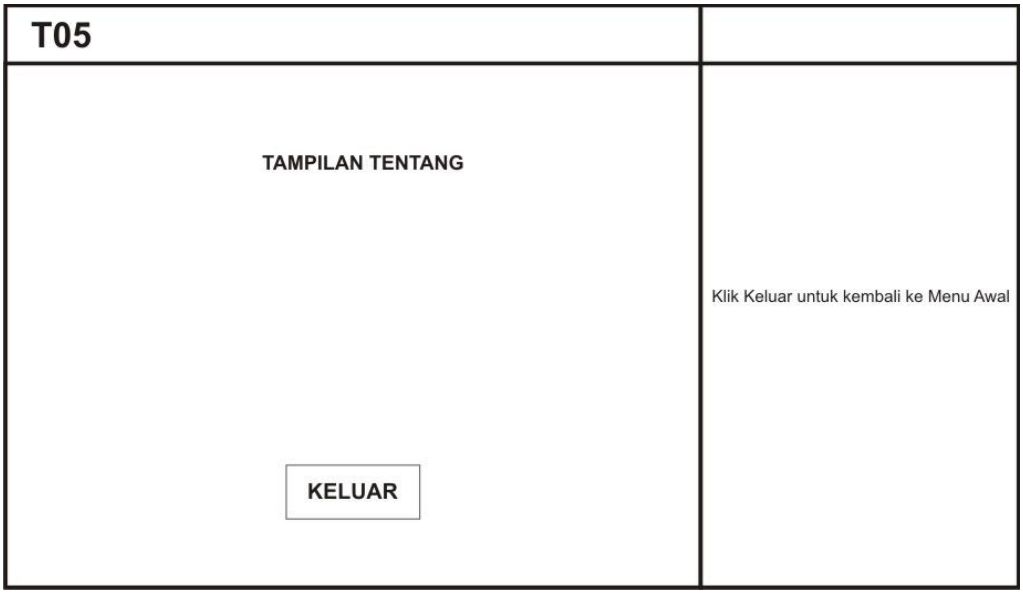
Perancangan antarmuka petunjuk dari *game* Palagan Ambarawa dapat dilihat pada gambar 3.53.



Gambar 3.53 Antarmuka Petunjuk

3.4.3.5 Antarmuka Tentang

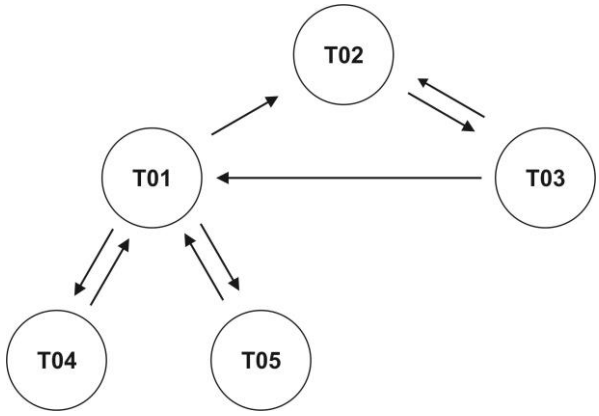
Perancangan antarmuka tentang dari *game* Palagan Ambarawa dapat dilihat pada gambar 3.54.



Gambar 3.54 Tampilan Tentang

3.4.4 Jaringan Semantik

Struktur aplikasi ini dibuat secara moduler, yaitu program dipecah menjadi modul-modul kecil yang mudah dibuat, mudah dites, dan mudah dimodifikasi. Dalam pembuatan *game* ini menggunakan alat bantu bagan struktural yang disebut juga jaringan semantik dapat dilihat pada gambar 3.55.



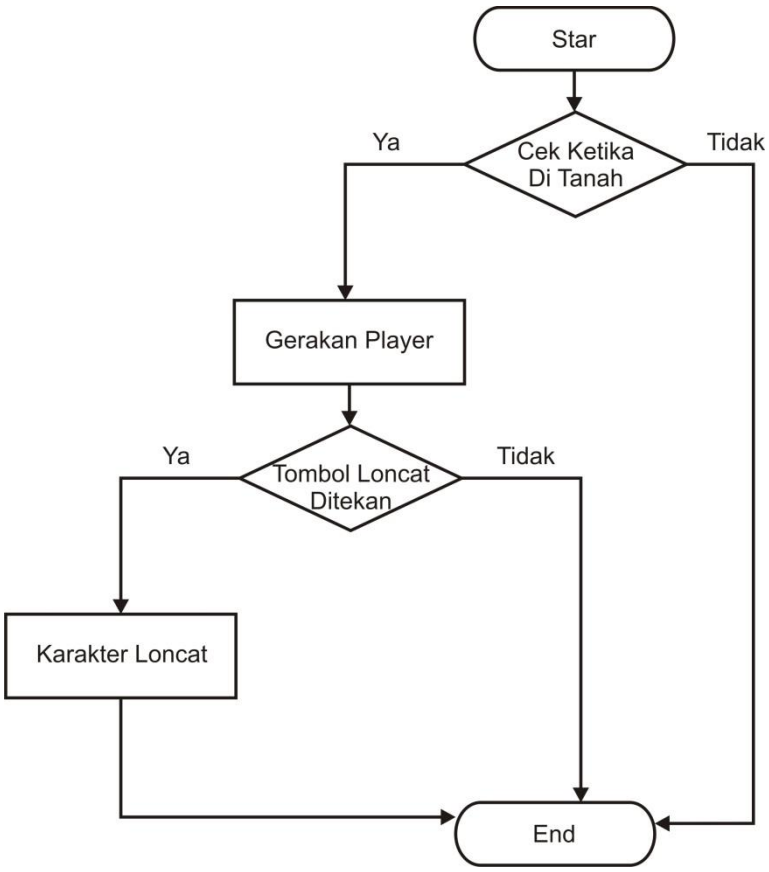
Gambar 3.55 Jaringan Semantik

3.4.5 Perancangan Method

Perancangan *method* merupakan perancangan yang berfungsi untuk mendeskripsikan *method-method* yang berada di dalam aplikasi. *Method* dapat dipanggil dengan menyertakan variabel, baik hanya satu variabel, banyak variabel atau bahkan tidak ada sama sekali .Adapun *method-method* yang terdapat dalam *game* 3D Palagan Ambarawa yang akan dibangun dijelaskan melalui *flowchart* sebagai berikut.

3.4.5.1 Method FPSwalker.FixedUpdate()

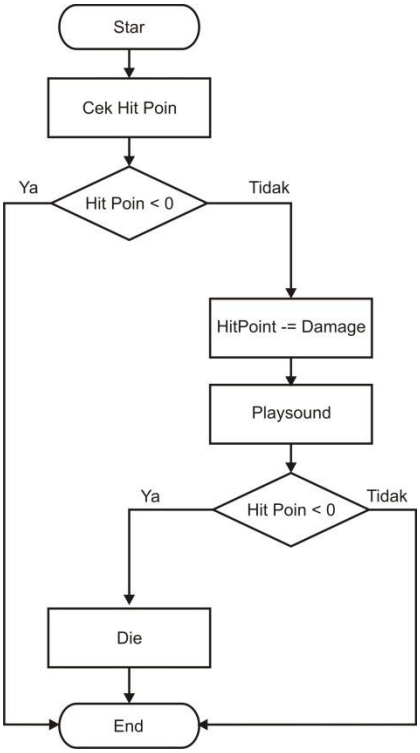
Method ini untuk menggerakkan karakter melakukan update berupa pergerakan meloncat terhadap karakter yang diawali dari inputan yang dilakukan pemain ketika menyentuh tanah. *Method* ini lebih jelasnya dapat dilihat pada gambar 3.56.



Gambar 3.56 *Flowchart Method FPSwalker.FixedUpdate()*

3.4.5.2 Method FPSPlayer.ApplyDamage()

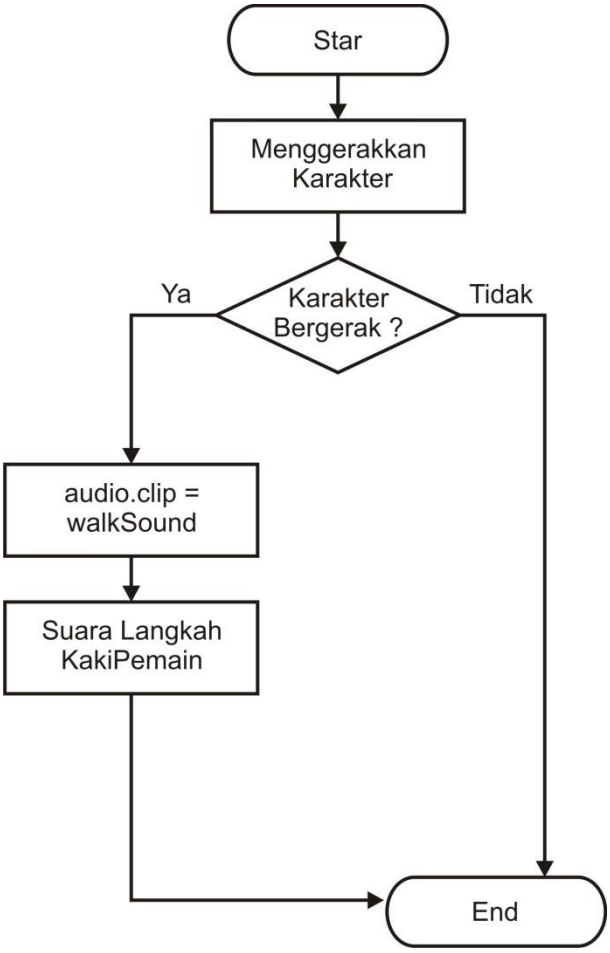
Method ini untuk pengecekan darah (*hitpoint)* yang dimiliki karakter ketika memainkan permainan di tiap *level*. *Method* ini lebih jelasnya dapat dilihat pada gambar 3.57.



Gambar 3.57 *Flowchart Method FPSPlayerApplyDamage*

3.4.5.3 Method FPSPlayer.PlayStepSounds()

Method ini akan memberikan efek suara langkah kaki terhadap karakter ketika pemain menggerakan karakter yang dapat dilihat pada gambar 3.58.

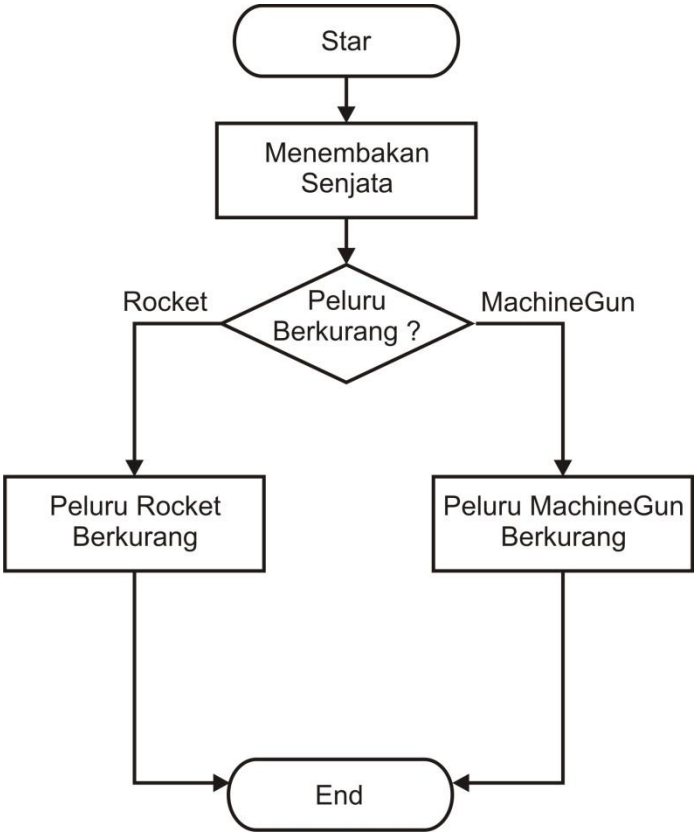


Gambar 3.58 *Flowchart Method FPSPlayer.PlayStepSounds()*

3.4.5.4 Method FPSPlayer.UpdateGUI

Method ini akan melakukan perubahan terhadap GUI berupa pengurangan peluru ketika pemain menembak, baik menembak menggunakan senjata laras panjang ataupun senjata roket yang dapat dilihat pada gambar 3.59.

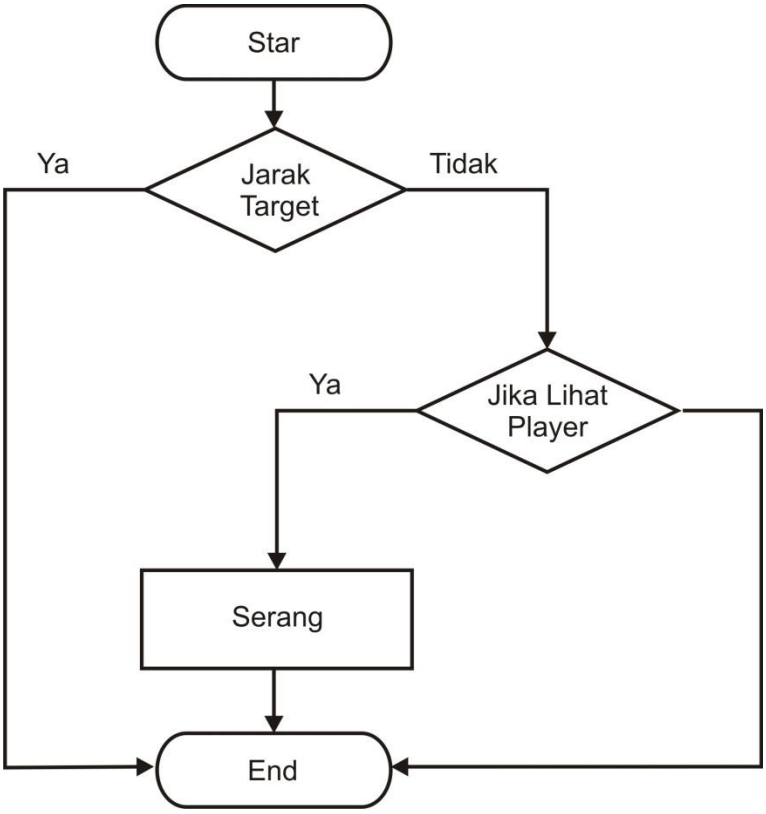
.



Gambar 3.59 *Flowchart Method FPSPlayer.UpdateGUI*

3.4.5.5 Method MusuhAI.CanSeeTarget()

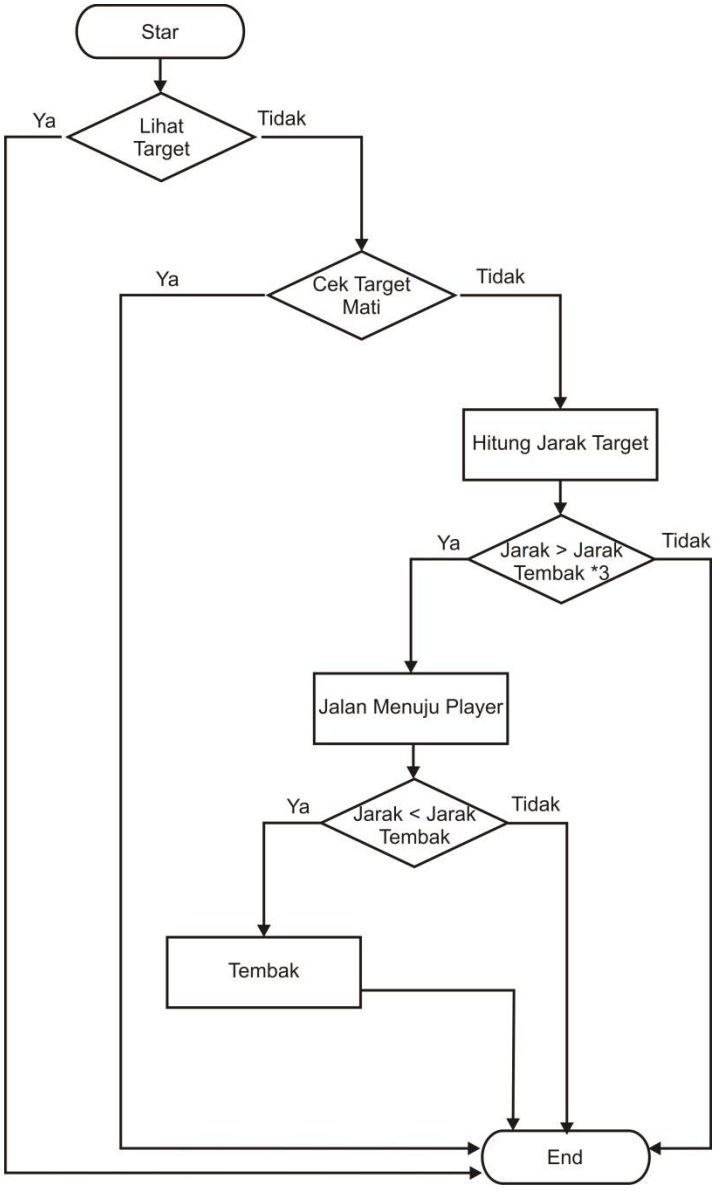
Method ini digunakan pada karakter musuh untuk menyerang ketika melihat pemain atau terget dalam permainan di tiap *level*. *Method* ini lebih jelasnya dapat dilihat pada gambar 3.60.



Gambar 3.60 *Flowchart Method MusuhAI.CanSeeTarget()*

3.4.5.6 Method MusuhAI.AttackPlayer()

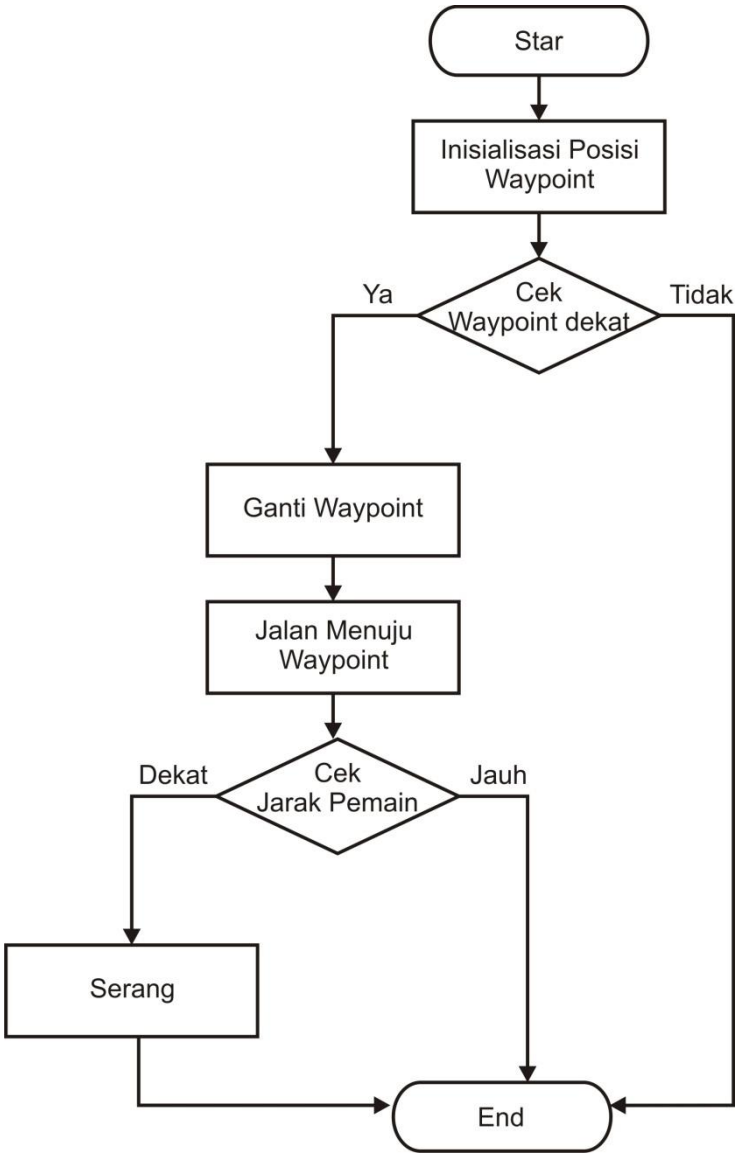
Method ini digunakan pada karakter musuh untuk mencari pemain atau terget berdasarkan jarak pemain di tiap *level* permainan. *Method* ini lebih jelasnya dapat dilihat pada gambar 3.61.



Gambar 3.61 *Flowchart Method MusuhAI. AttackPlayer()*

3.4.5.7 Method MusuhAI.Patrol()

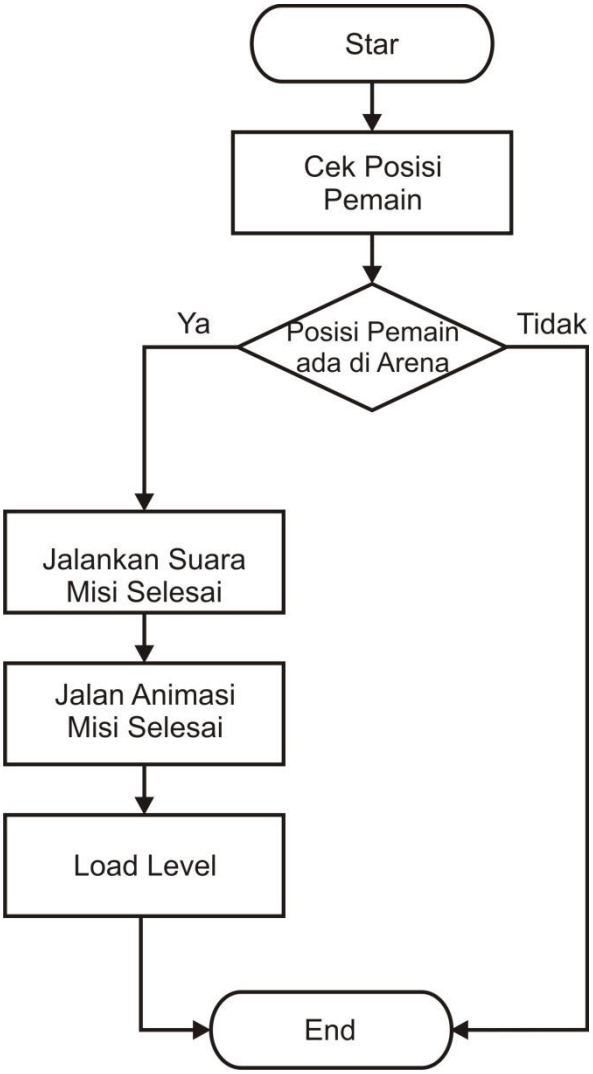
Method ini digunakan pada karakter ketika tidak melihat pemain atau terget maka musuh akan melakukan patroli/berkeliling sesuai dengan jalur *waypoint* yang berada di tiap *level* permainan. *Method* ini lebih jelasnya dapat dilihat pada gambar 3.62.



Gambar 3.62 *Flowchart Method MusuhAI.Patrol()*

3.4.5.8 Method MisiSelesai.MisiSelesai()

Method ini digunakan untuk pengecekan terhadap posisi pemain yang berada di arena misi selesai maka akan melakukan *load level* ke *level* selanjutnya. *Method* ini lebih jelasnya dapat dilihat pada gambar 3.63.



Gambar 3.63 *Flowchart Method MisiSelesai.MisiSelesai()*