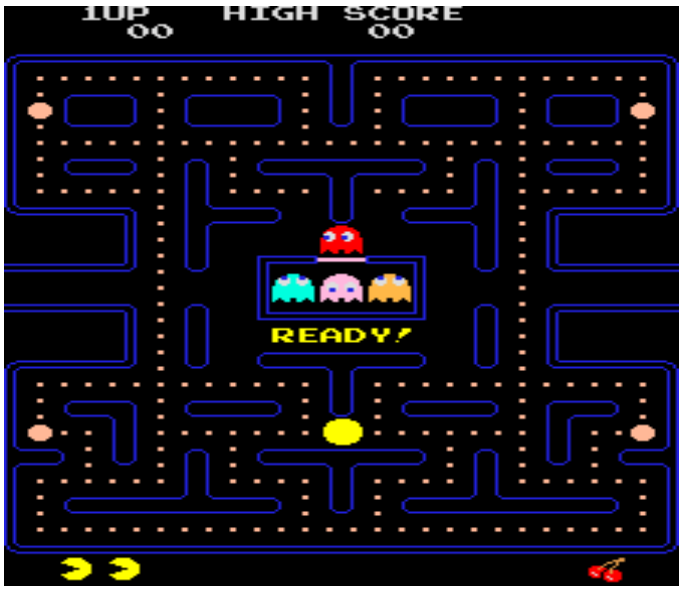
BAB 3

ANALISIS DAN PERANCANGAN

Analisis merupakan proses penguraian konsep ke dalam bagian-bagian yang lebih sederhana, sehingga struktur logisnya menjadi jelas lalu mengidentifikasi permasalahan-permasalahan yang terdapat pada sistem dan menentukan kebutuhan-kebutuhan dari sistem yang akan dibangun.

3.1 Analisis Sistem

Game dengan tema labirin memang sangat menarik untuk dimainkan dan lebih menantang, contohnya *game* pac-man [2].



Gambar 3.1 *Game* pac-man

mempunyai *gameplay* yang menarik dimana player harus menghabiskan titik-titik yang tersedia untuk menyelesaikan *game* ini dan kecerdasan buatan pada tiap musuhnya sudah cukup bagus, namun *game* ini masih memiliki kekurangan diantaranya : gambar yang masih sederhana, peta labirin di setiap *level* masih sama sehingga akan membuat pemainnya merasa bosan.

3.2 Analisis Kebutuhan Non Fungsional

3.2.1 Analisis Kebutuhan Perangkat Keras

Perangkat keras yang dibutuhkan untuk memainkan *game* ini antara lain :

Tabel 3.1 Perangkat keras yang digunakan

Perangkat Keras	Spesifikasi
Processor	Kecepatan 1,8 GHz
Memory	512 MB
VGA	512 MB
Harddisk	*Free space* 100 MB
Monitor	Resolusi 1024 x 768
Keyboard	
Mouse	

3.2.2 Analisis Kebutuhan Perangkat Lunak

Selain perangkat keras, perangkat lunak (*software*) pendukung juga dibutuhkan dalam menjalankan sebuah aplikasi. Perangkat lunak pendukung yang digunakan untuk membangun aplikasi ini antara lain :

1. Sistem operasi komputer Windows XP.
2. *Adobe Flash Player 10.*
3. *Adobe Flash CS5.*

3.2.3 Analisis AI (*Artificial Intelligence*)

Pada *game* yang akan dibangun algoritma A* (A *Star*) akan diterapkan pada musuh untuk mencari jalan menuju karakter *player*. Pencarian jalan terpendek pada *game* ini akan diasumsikan dengan titik awal (musuh) dan titik tujuan (*player*) dimana titik tujuan bersifat dinamis sehingga akan membuat

musuh terus mengejar *player* dan akan diimplementasikan pada pencarian dalam ruangan yang mempunyai penghalang.

m	m	m	m	m	m	m	m	m	m	m	m	m
m	P			m							m	m
m		m										m
m		m		m	m	m	m	m	m	m		m
m										E		m
m		m	m	m		m	m	m		m		m
m												m
m	m		m	m	m		m		m	m		m
m							m					m
m		m		m	m		m				m	m
m									m			m
m	m	m	m	m	m	m	m	m	m	m	m	m

Gambar 3.2 Peta *level* 1

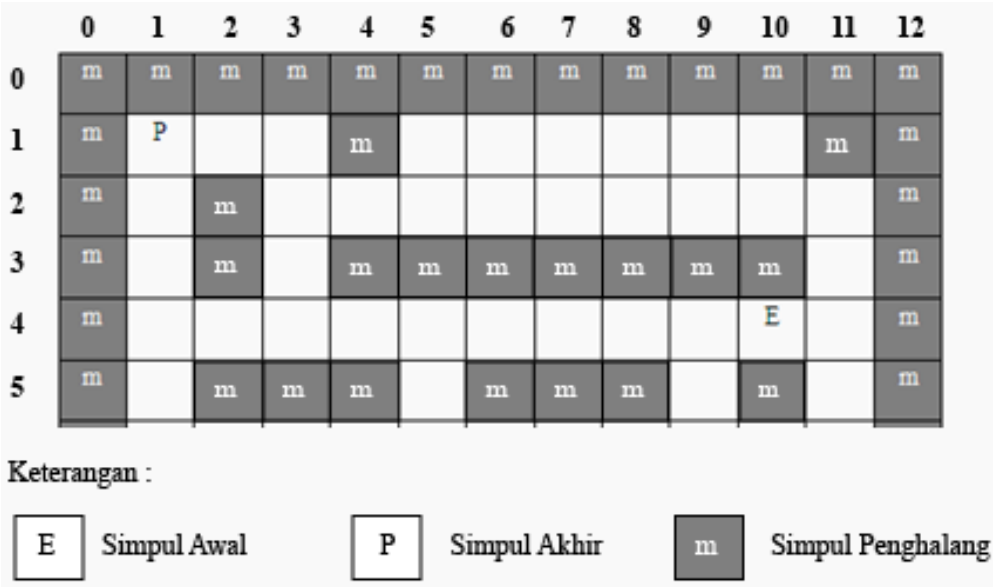
Pada Gambar 3.2 terlihat bahwa terdapat 3 objek, yaitu : *player*, musuh dan tembok penghalang. Dimana P adalah karakter *player*, E adalah karakter *enemy* dan m adalah sebagai tembok penghalang. Karakter P dan E dipergunakan untuk menunjukkan lokasi titik awal (P) dan titik akhir (E). Dalam proses pencarian algoritma A* terdapat 3 nilai yaitu *g*, *h* dan *f*. Nilai *g* adalah biaya sebenarnya, nilai *g* mempunyai nilai yang berbeda untuk pergerakan secara horizontal *cost*nya adalah 10 sedangkan untuk yang diagonal adalah 14, nilai *h* adalah biaya perkiraannya dan rumus untuk mendapatkan nilai *heuristic* adalah *h*(n) = abs(n.x-goal.x) + abs(n.y-goal.y), sedangkan *f* adalah total biaya yang

diperhitungkan. Dalam notasi matematika algoritma A* dapat dituliskan dengan $f(n) = g(n) + h(n)$.

Proses pertama yang dilakukan oleh musuh adalah mencari tahu jalan menuju *player*, proses yang kedua adalah mencari langkah yang akan diambil kemudian proses selanjutnya adalah mengambil langkah dan apabila langkah menuju *player* terhalang oleh tembok maka ulangi proses pencarian sampai menemukan jalan terpendek menuju *player*.

Jika musuh sudah mencapai *player* maka proses pencarian akan selesai lalu nyawa *player* akan berkurang satu dan waktu akan berhenti kemudian permainan akan berulang sampai nyawa *player* habis.

Untuk lebih jelas, berikut adalah langkah perhitungan dalam pencarian jalur menuju player :



Gambar 3.3 Kondisi awal pencarian A*

Posisi simpul awal = currentX :10, currentY : 4

Posisi simpul tujuan = targetX : 1, targetY : 1

Langkah pertama

G (10,4) = 10

H (10,4) = 10 * (abs(currentX-targetX) + abs(currentY-targetY))
= 10 * (abs(9-1) + abs(4-1))
= 10 * (8+3)
= 10 * 11
= 110

F (10,4) = G(10,4) + H(10,4)
= 10 + 110
= 120

G (9,5) = 14

H (9,5) = 10 * (abs(currentX-targetX) + abs(currentY-targetY))
= 10 * (abs(9-1) + abs(5-1))
= 10 * (8+4)
= 10 * 12
= 120

F (9,5) = G(9,5) + H(9,5)
= 14 + 120
= 134

G (11,3) = 14

H (11,3) = 10 * (abs(currentX-targetX) + abs(currentY-targetY))
= 10 * (abs(11-1) + abs(3-1))
= 10 * (10+2)
= 10 * 12
= 120

F (11,3) = G(11,3) + H(11,3)

= 14 + 120

= 134

G (11,4) = 10

H (11,4) = 10 * (abs(currentX-targetX) + abs(currentY-targetY))

= 10 * (abs(11-1) + abs(4-1))

= 10 * (10+3)

= 10 * 13

= 130

F (11,4) = G (11,4) + H (11,4)

= 10 + 130

= 140

G (11,5) = 14

H (11,5) = 10 * (abs(currentX-targetX) + abs(currentY-targetY))

= 10 * (abs(11-1) + abs(5-1))

= 10 * (10+4)

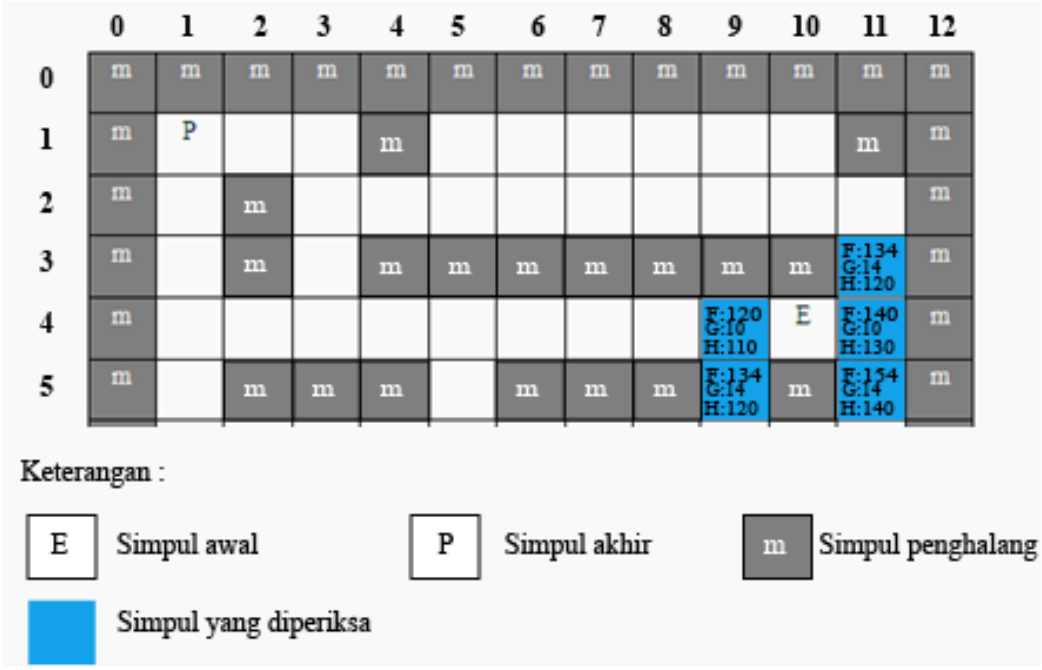
= 10 * 14

= 140

F (11,5) = G (11,5) + H (11,5)

= 14 + 140

= 154



Gambar 3.4 Langkah pertama pencarian *best node*

Pada Gambar 3.4 terdapat lima simpul yang mungkin menjadi *best node* yaitu (9,4) dengan F(n) = 120, (9,5) dengan F(n) = 134, (11,3) dengan F(n) = 134, (11,4) dengan F(n) = 140 dan (11,5) dengan F(n) = 154, maka yang dipilih adalah simpul (9,4) dengan biaya 120.

Langkah kedua

G (9,5) = 20

H (9,5) = 10 * (abs(currentX-targetX) + abs(currentY-targetY))
= 10 * (abs(9-1) + abs(5-1))
= 10 * (8+4)
= 10 * 12
= 120

F (9,5) = G(9,5) + H(9,5)
= 20 + 120
= 140

G (8,4) = 20

H (8,4) = 10 * (abs(currentX-targetX) + abs(currentY-targetY))

= 10 * (abs(8-1) + abs(4-1))

= 10 * (7+3)

= 10 * 10

= 100

F (8,4) = G (8,4) + H (8,4)

= 20 + 100

= 120



Gambar 3.5 Langkah kedua pencarian *best node*

Pada Gambar 3.5 terdapat dua simpul yang mungkin menjadi *best node* yaitu (8,4) dengan F(n) = 120 dan (9,5) dengan F(n) = 140, maka yang dipilih adalah simpul (8,4) dengan biaya 120.

Langkah ketiga

G (7,4) = 30

H (7,4) = 10 * (abs(currentX-targetX) + abs(currentY-targetY))

= 10 * (abs(7-1) + abs(4-1))

= 10 * (6+3)

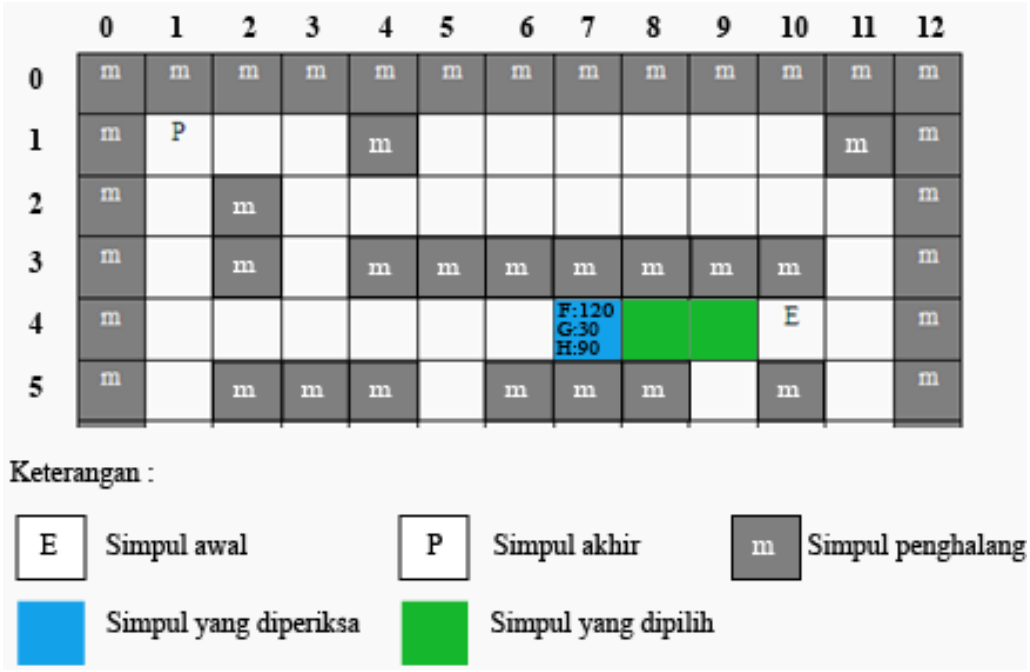
= 10 * 9

= 90

F (7,4) = G(7,4) + H(7,4)

= 30 + 90

= 120



Gambar 3.6 Langkah ketiga pencarian *best node*

Pada Gambar 3.6 terdapat satu simpul yang menjadi *best node* yaitu (7,4) dengan F(n) = 120, maka yang dipilih adalah simpul (7,4) dengan biaya 120.

Langkah keempat

G (6,4) = 40

H (6,4) = 10 * (abs(currentX-targetX) + abs(currentY-targetY))

= 10 * (abs(6-1) + abs(4-1))

= 10 * (5+3)

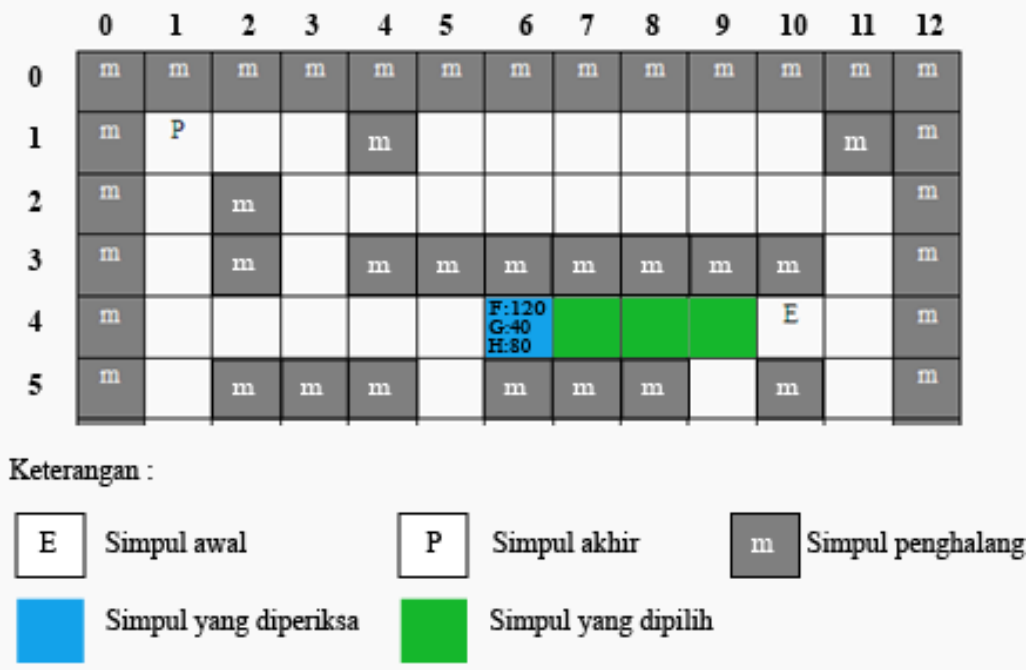
= 10 * 8

= 80

F (6,4) = G(6,4) + H(6,4)

= 40 + 80

= 120



Gambar 3.7 Langkah keempat pencarian *best node*

Pada Gambar 3.7 terdapat satu simpul yang menjadi *best node* yaitu (6,4) dengan F(n) = 120, maka yang dipilih adalah simpul (6,4) dengan biaya 120.

Langkah kelima

G (5,4) = 50

H (5,4) = 10 * (abs(currentX-targetX) + abs(currentY-targetY))

= 10 * (abs(5-1) + abs(4-1))

= 10 * (4+3)

= 10 * 7

= 70

F (5,4) = G(5,4) + H(5,4)

= 50 + 70

= 120

G (5,5) = 54

H (5,5) = 10 * (abs(currentX-targetX) + abs(currentY-targetY))

= 10 * (abs(5-1) + abs(5-1))

= 10 * (4+4)

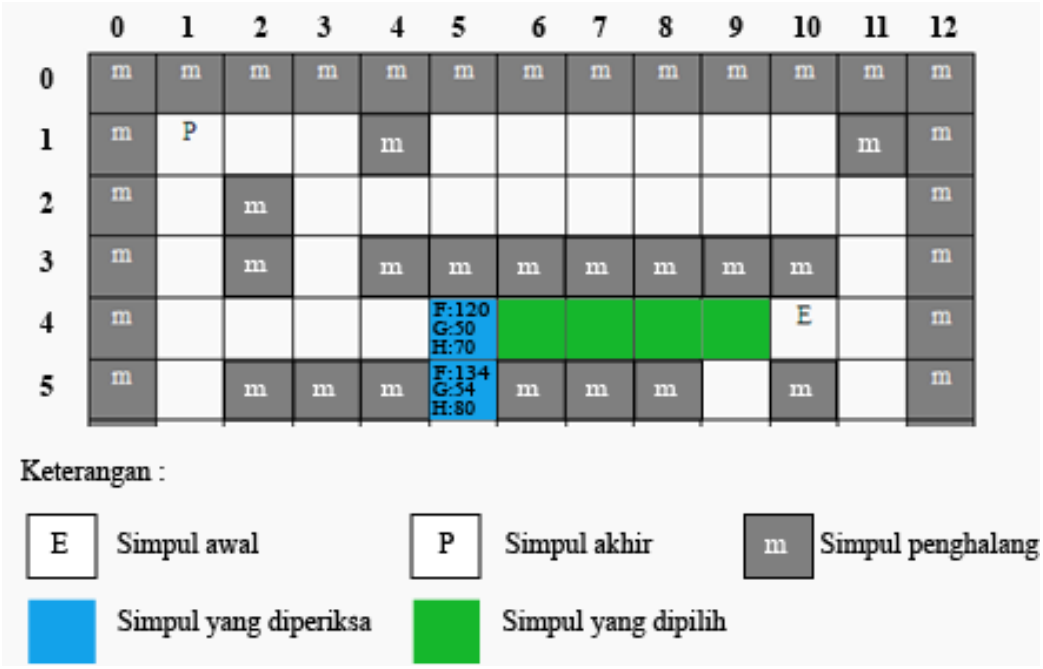
= 10 * 8

= 80

F (5,5) = G(5,5) + H(5,5)

= 54 + 80

= 134



Gambar 3.8 Langkah kelima pencarian *best node*

Pada Gambar 3.8 terdapat dua simpul yang mungkin menjadi *best node* yaitu (5,4) dengan F(n) = 120 dan (5,5) dengan F(n) = 134, maka yang dipilih adalah simpul (5,4) dengan biaya 120.

Langkah keenam

G (4,4) = 60

H (4,4) = 10 * (abs(currentX-targetX) + abs(currentY-targetY))

= 10 * (abs(4-1) + abs(4-1))

= 10 * (3+3)

= 10 * 6

= 60

F (4,4) = G(4,4) + H(4,4)

= 60 + 60

= 120

G (5,5) = 60

H (5,5) = 10 * (abs(currentX-targetX) + abs(currentY-targetY))

= 10 * (abs(5-1) + abs(5-1))

= 10 * (4+4)

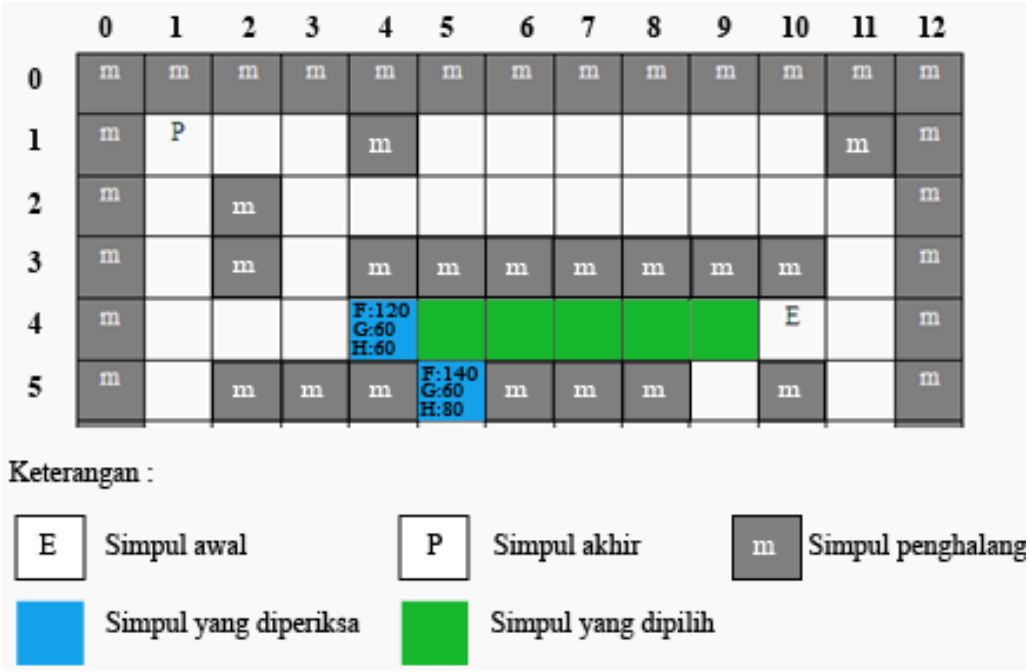
= 10 * 8

= 80

F (5,5) = G(5,5) + H(5,5)

= 60 + 80

= 140



Gambar 3.9 Langkah keenam pencarian *best node*

Pada Gambar 3.9 terdapat dua simpul yang mungkin menjadi *best node* yaitu (4,4) dengan F(n) = 120 dan (5,5) dengan F(n) = 140, maka yang dipilih adalah simpul (4,4) dengan biaya 120.

Langkah ketujuh

G (3,3) = 74

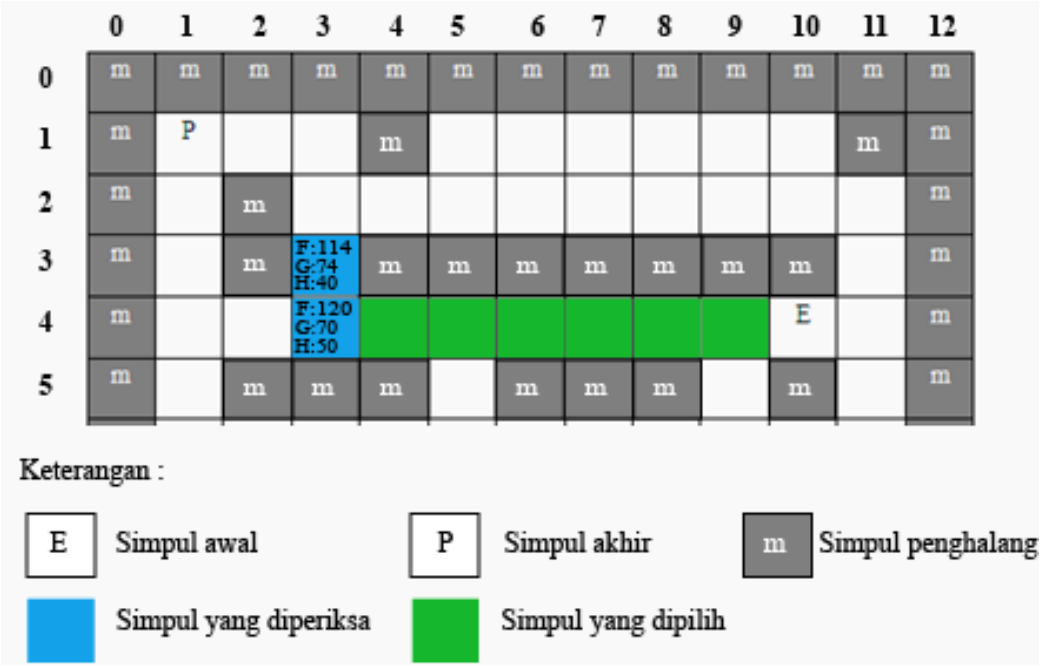
H (3,3) = 10 * (abs(currentX-targetX) + abs(currentY-targetY))
= 10 * (abs(3-1) + abs(3-1))
= 10 * (2+2)
= 10 * 4
= 40

F (3,3) = G(3,3) + H(3,3)
= 74 + 40
= 114

G (3,4) = 70

H (3,4) = 10 * (abs(currentX-targetX) + abs(currentY-targetY))
= 10 * (abs(3-1) + abs(4-1))
= 10 * (2+3)
= 10 * 5
= 50

F (3,4) = G(3,4) + H(3,4)
= 70 + 50
= 120



Gambar 3.10 Langkah ketujuh pencarian *best node*

Pada Gambar 3.10 terdapat dua simpul yang mungkin menjadi *best node* yaitu (3,3) dengan F(n) = 114 dan (3,4) dengan F(n) = 120, maka yang dipilih adalah simpul (3,3) dengan biaya 114.

Langkah kedelapan

G (3,2) = 84

H (3,2) = 10 * (abs(currentX-targetX) + abs(currentY-targetY))

= 10 * (abs(3-1) + abs(2-1))

= 10 * (2+1)

= 10 * 3

= 30

F (3,2) = G(3,2) + H(3,2)

= 84 + 30

= 114

G (4,2) = 88

H (4,2) = 10 * (abs(currentX-targetX) + abs(currentY-targetY))

= 10 * (abs(4-1) + abs(2-1))

= 10 * (3+1)

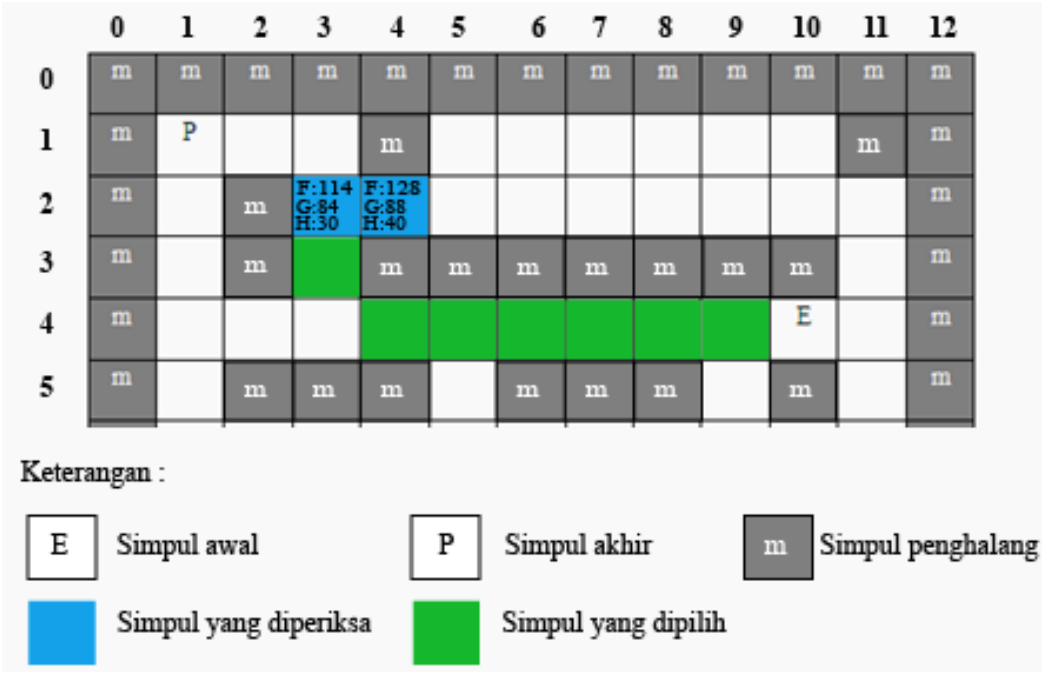
= 10 * 4

= 40

F (4,2) = G(4,2) + H(4,2)

= 88 + 40

= 128



Gambar 3.11 Langkah kedelapan pencarian *best node*

Pada Gambar 3.11 terdapat dua simpul yang mungkin menjadi *best node* yaitu (3,2) dengan F(n) = 114 dan (4,2) dengan F(n) = 128, maka yang dipilih adalah simpul (3,2) dengan biaya 114.

Langkah kesembilan

G (2,1) = 98

H (2,1) = 10 * (abs(currentX-targetX) + abs(currentY-targetY))

= 10 * (abs(2-1) + abs(1-1))

= 10 * (1+0)

= 10 * 1

= 10

F (2,1) = G(2,1) + H(2,1)

= 98 + 10

= 108

G (3,1) = 94

H (3,1) = 10 * (abs(currentX-targetX) + abs(currentY-targetY))

= 10 * (abs(3-1) + abs(1-1))

= 10 * (2+0)

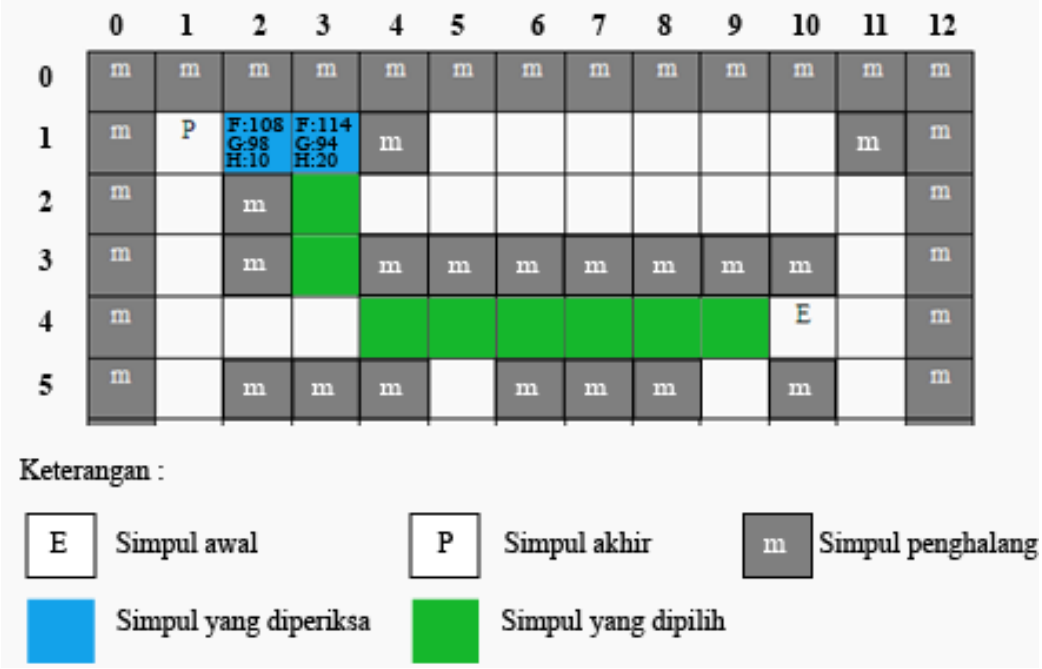
= 10 * 2

= 20

F (3,1) = G(3,1) + H(3,1)

= 94 + 20

= 114



Gambar 3.12 Langkah kesembilan pencarian *best node*

Pada Gambar 3.12 terdapat dua simpul yang mungkin menjadi *best node* yaitu (2,1) dengan F(n) = 108 dan (3,1) dengan F(n) = 114, maka yang dipilih adalah simpul (3,1) dengan biaya 108.

Langkah kesepuluh

G (1,1) = 108

H (1,1) = 10 * (abs(currentX-targetX) + abs(currentY-targetY))

= 10 * (abs(1-1) + abs(1-1))

= 10 * (0+0)

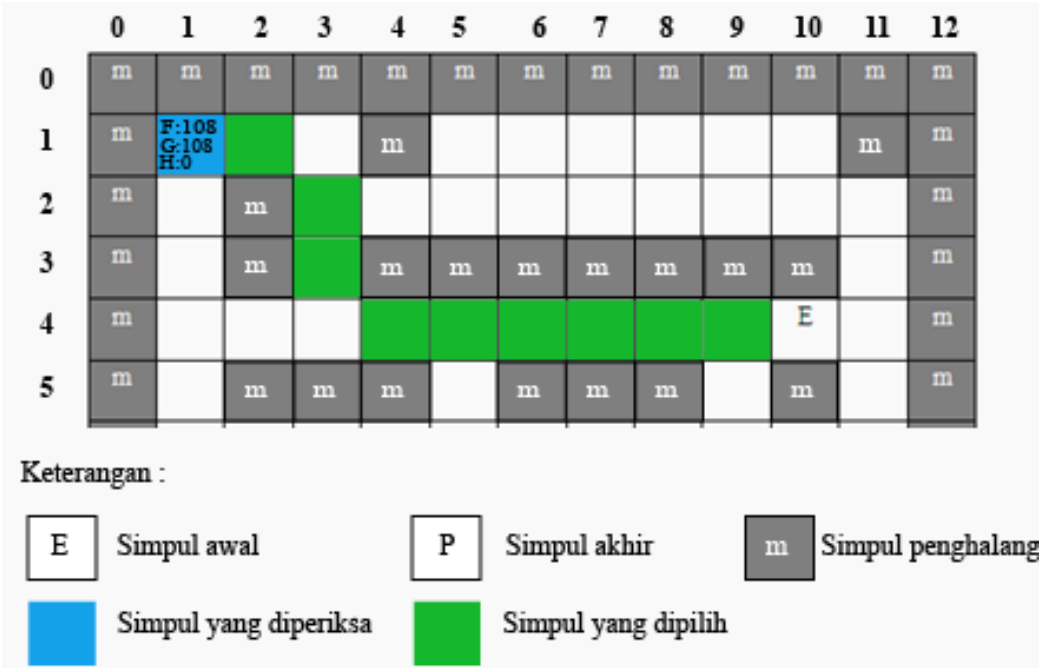
= 10 * 0

= 0

F (1,1) = G(2,1) + H(2,1)

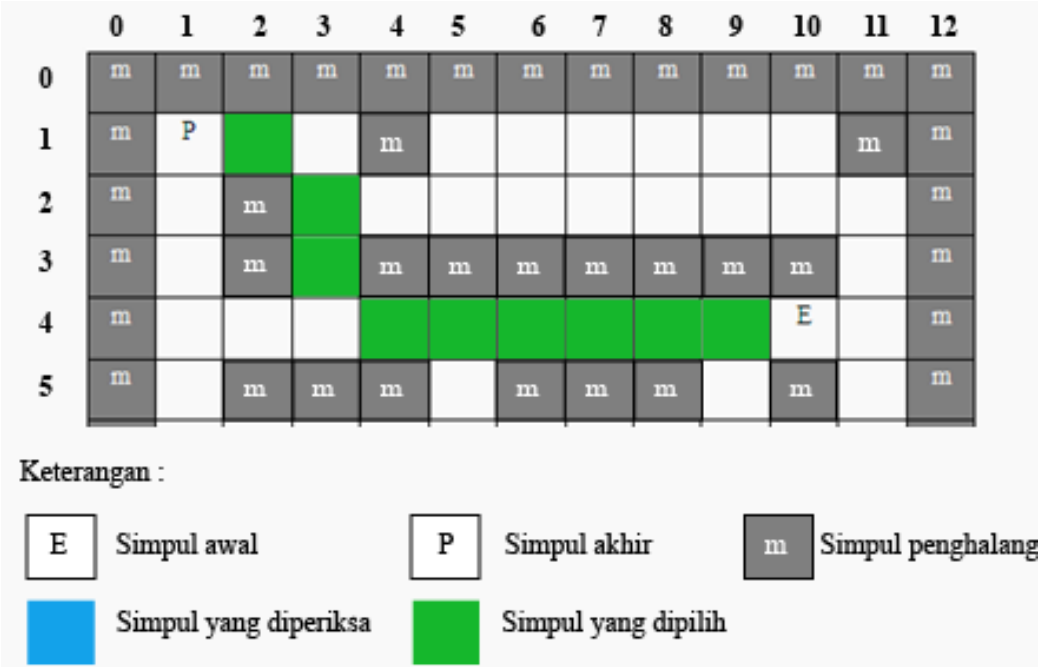
= 108 + 0

= 108



Gambar 3.13 Langkah kesepuluh pencarian *best node*

Pada Gambar 3.13 terdapat satu simpul (1,1) dengan F(n) = 108 dan dikenali sebagai simpul tujuan.



Gambar 3.14 Hasil pencarian jalan dengan algortima A*

Gambar 3.14 merupakan hasil pencarian jalan dari titik awal (E) menuju titik akhir (P).

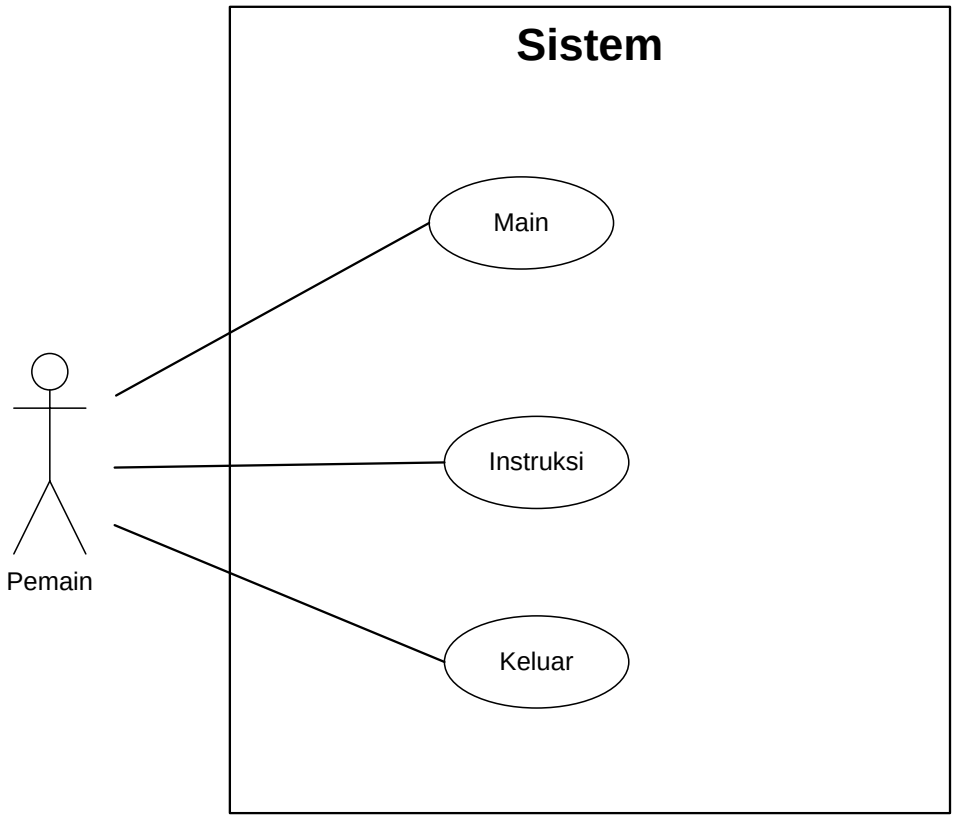
3.3 Analisis Kebutuhan Fungsional

Menggambarkan proses kegiatan yang akan diterapkan dalam sistem dan menjelaskan kebutuhan yang diperlukan agar sistem dapat berjalan dengan baik serta sesuai dengan kebutuhan proses bisnis sistem yang bersangkutan.

Analisis yang dilakukan dimodelkan dengan menggunakan UML (*Unified Modeling Language*). Tahapan pemodelan dalam analisis tersebut antara lain mengidentifikasi aktor, pembuatan *use case diagram*, *use case scenario*, *activity diagram*, *sequence diagram*, *class diagram*, dan *state diagram*.

3.3.1 *Use Case Diagram*

Use case adalah konstruksi untuk mendeskripsikan bagaimana sistem terlihat dimata pengguna. Sasaran pemodelan *use case* diantaranya adalah mendefinisikan kebutuhan fungsional dan operasional sistem dengan mendefinisikan skenario penggunaan yang disepakati antara pemakai dan pengembang (*developer*). Dari identifikasi aktor yang terlibat di atas maka *use case* diagram untuk aplikasi *game* jalan pintas dapat dilihat dalam Gambar 3.15.



Gambar 3.15 *Use case diagram*

3.3.2 *Use Case Scenario*

3.3.2.1 *Use Case Scenario* Main

Use case scenario main digunakan untuk memulai permainan yang dilakukan oleh pengguna. Skenario main dapat dilihat pada Tabel 3.2.

Tabel 3.2 *Use case scenario* main

Identifikasi	
Nomor	1
Nama	Main
Tujuan	Memainkan permainan
Deskripsi	Proses memulai untuk memainkan permainan
Aktor	*User*
***Use case* yang berkaitan**	-
Skenario Utama	
Kondisi Awal	Tampilan menu permainan.
Aksi Aktor	**Reaksi Sistem**
1. User menekan tombol main.	2. Memeriksa inputan yang dipilih oleh user.
	3. Menampilkan permainan.
4. User menekan tombol kiri, atas, kanan dan bawah untuk menggerakan karakter	5. Memeriksa inputan dari user lalu memproses inputan tersebut.
Kondisi Akhir	Memainkan permainan

3.3.2.2 Use Case Scenario Instruksi

Use case scenario instruksi digunakan untuk menampilkan instruksi dalam memainkan permainan yang dilakukan oleh pengguna. Skenario instruksi dapat dilihat pada Tabel 3.3.

Tabel 3.3 *Use case scenario* instruksi

Identifikasi	
Nomor	2
Nama	Instruksi
Tujuan	Memperoleh informasi permainan
Deskripsi	Proses memulai untuk memperoleh informasi permainan
Aktor	*User*
***Use case* yang berkaitan**	-
Skenario Utama	
Kondisi Awal	Tampilan menu permainan
Aksi Aktor	**Reaksi Sistem**
1. User menekan tombol instruksi.	2. Memproses data yang dipilih oleh user.
Kondisi Akhir	Menampilkan konten instruksi

3.3.2.3 *Use Case Scenario* Keluar

Use case scenario keluar digunakan untuk keluar dalam aplikasi permainan yang dilakukan oleh pengguna. Skenario keluar dapat dilihat pada Tabel 3.4.

Tabel 3.4 *Use case scenario* keluar

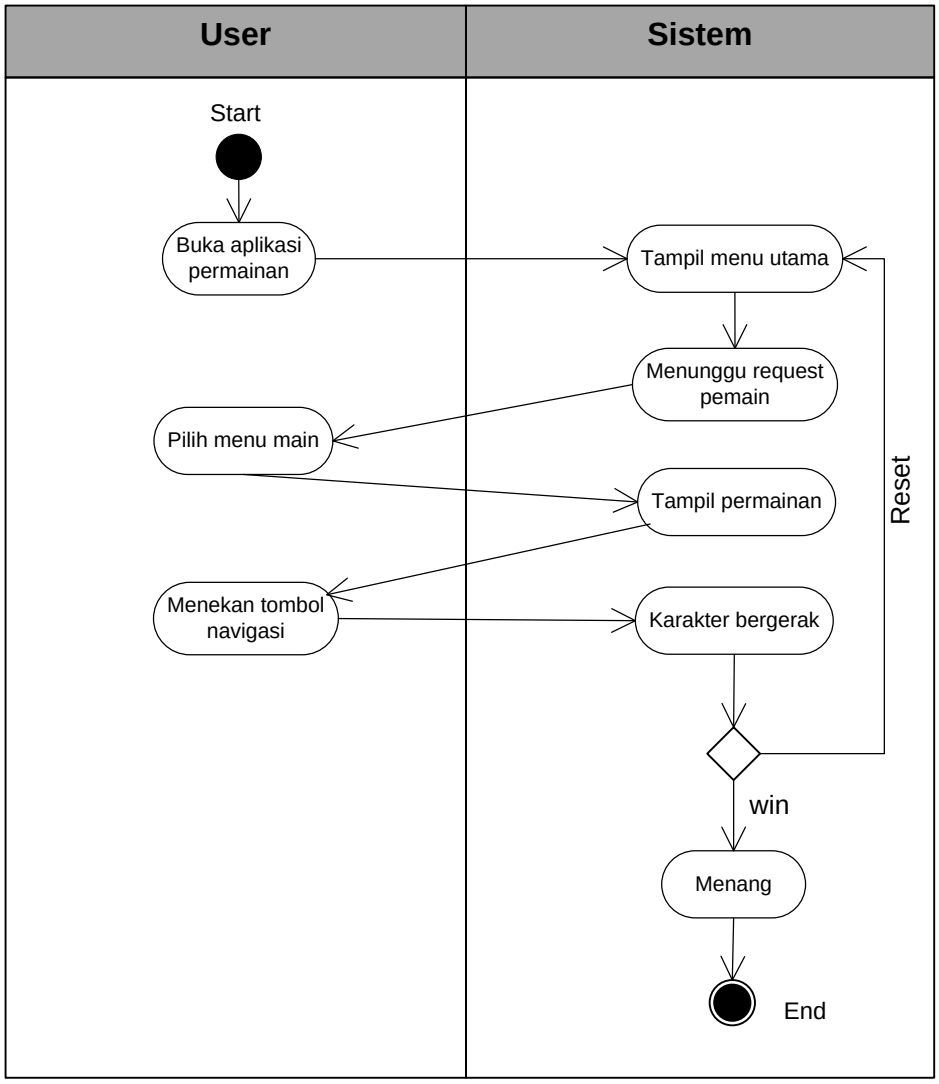
Identifikasi	
Nomor	3
Nama	Keluar
Tujuan	Keluar dari permainan
Deskripsi	Proses untuk keluar dari permainan
Aktor	*User*
***Use case* yang berkaitan**	-
Skenario Utama	
Kondisi Awal	Tampilan menu permainan
Aksi Aktor	**Reaksi Sistem**
1. User menekan tombol keluar.	2. Aplikasi berhenti.
Kondisi Akhir	User keluar dari permainan

3.3.3 Activity Diagram

Diagram aktifitas (*Activity diagram*) memodelkan aliran kerja atau *workflow* sebuah proses bisnis dan urutan aktifitas dalam suatu proses. Berikut gambaran diagram aplikasi *game* jalan pintas.

3.3.3.1 *Activity Diagram* Main

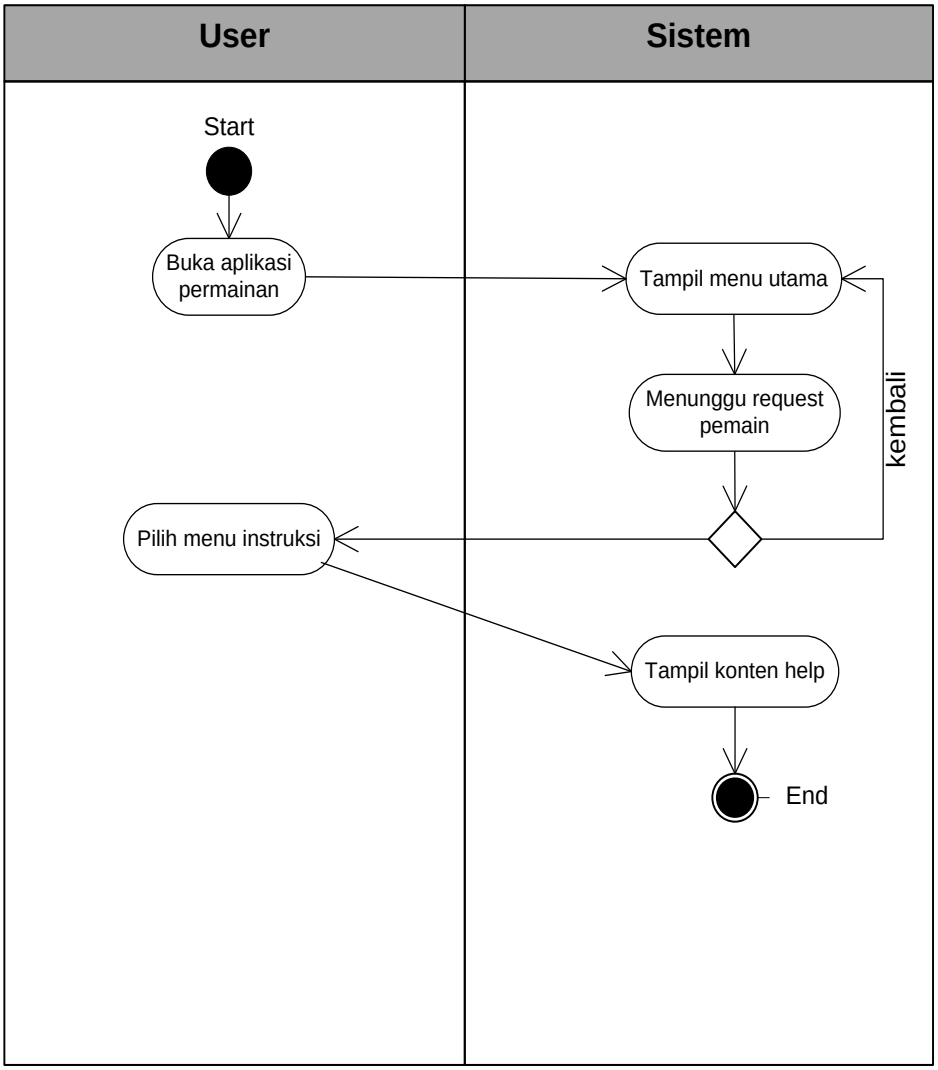
Aktifitas main merupakan aktifitas untuk memulai permainan yang dapat diakses oleh *user*. Seluruh tahapan aktifitas main dapat dilihat pada Gambar 3.16.



Gambar 3.16 *Activity diagram* main

3.3.3.2 *Activity* Diagram Instruksi

Aktifitas instruksi merupakan aktifitas untuk melihat petunjuk permainan yang dapat diakses oleh *user*. Seluruh tahapan aktifitas instruksi dapat dilihat pada Gambar 3.17.



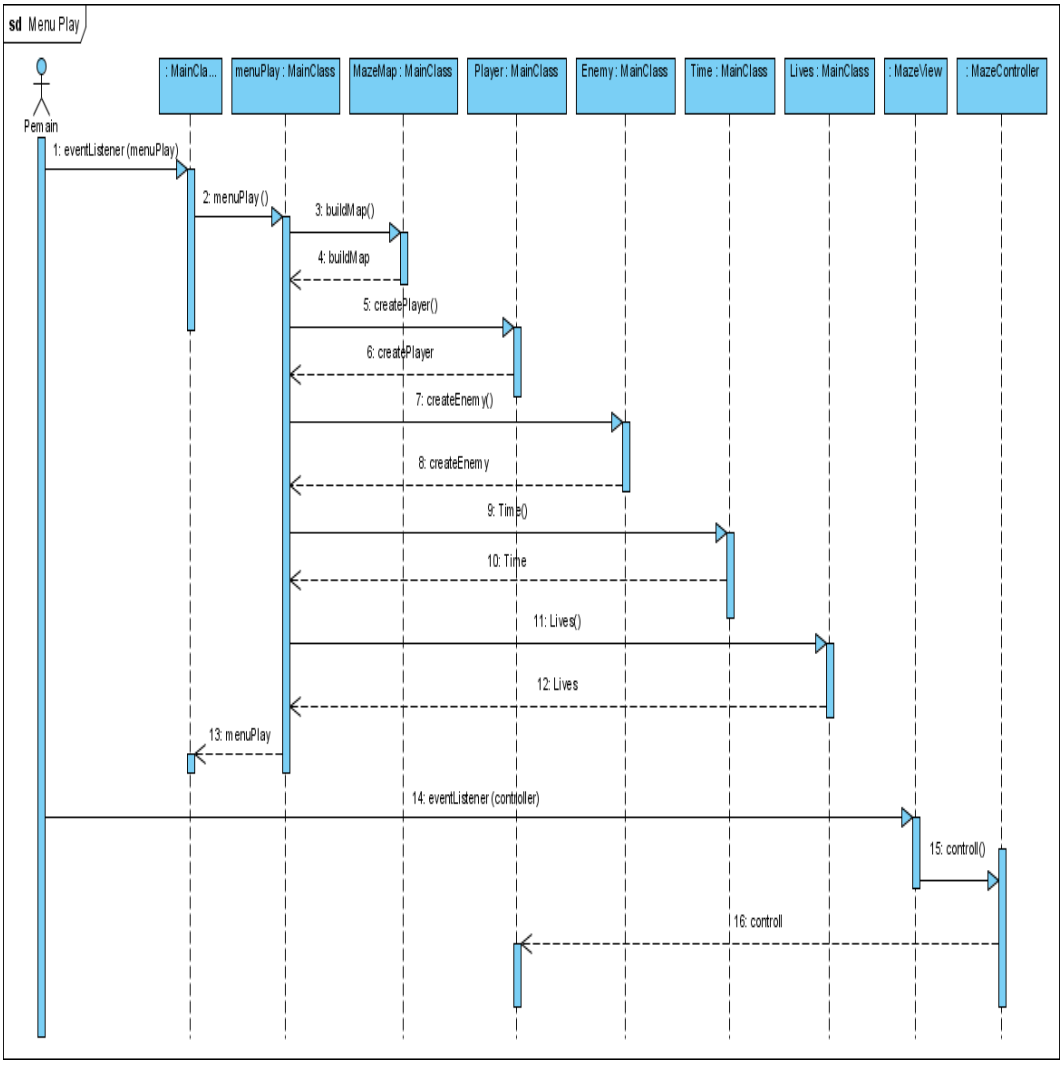
Gambar 3.17 *Activity diagram* instruksi

3.3.4 *Sequence Diagram*

Sequence diagram merupakan penerapan skenario aplikasi *game* jalan pintas dengan menunjukan interaksi dengan menampilkan setiap partisipan garis alir secara vertikal dan pengurutan pesan dari atas ke bawah.

3.3.4.1 *Sequence Diagram Use Case* Main

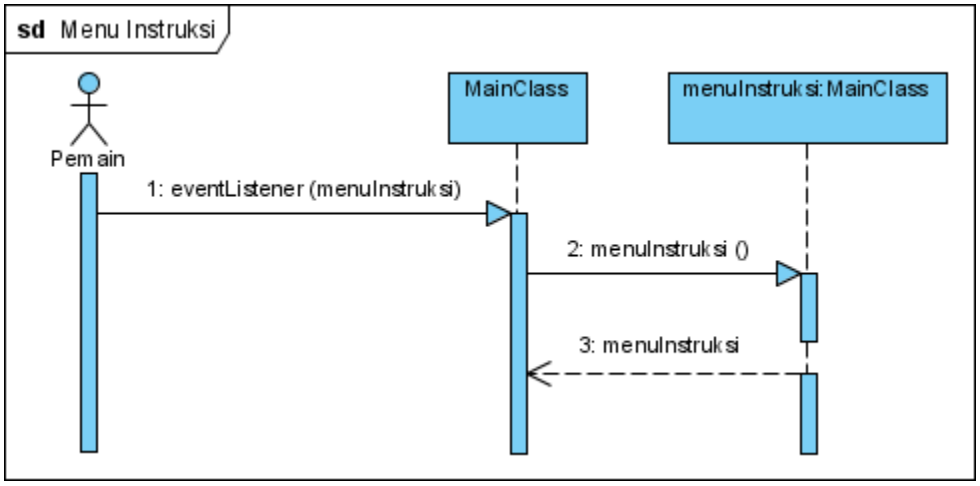
Sequence diagram untuk *use case* main menggambarkan interaksi antara objek dari *class* MainClass dan objek yang berkaitan dengan *class* karakter.



Gambar 3.18 *Sequence diagram* main

3.3.4.2 *Sequence Diagram* Instruksi

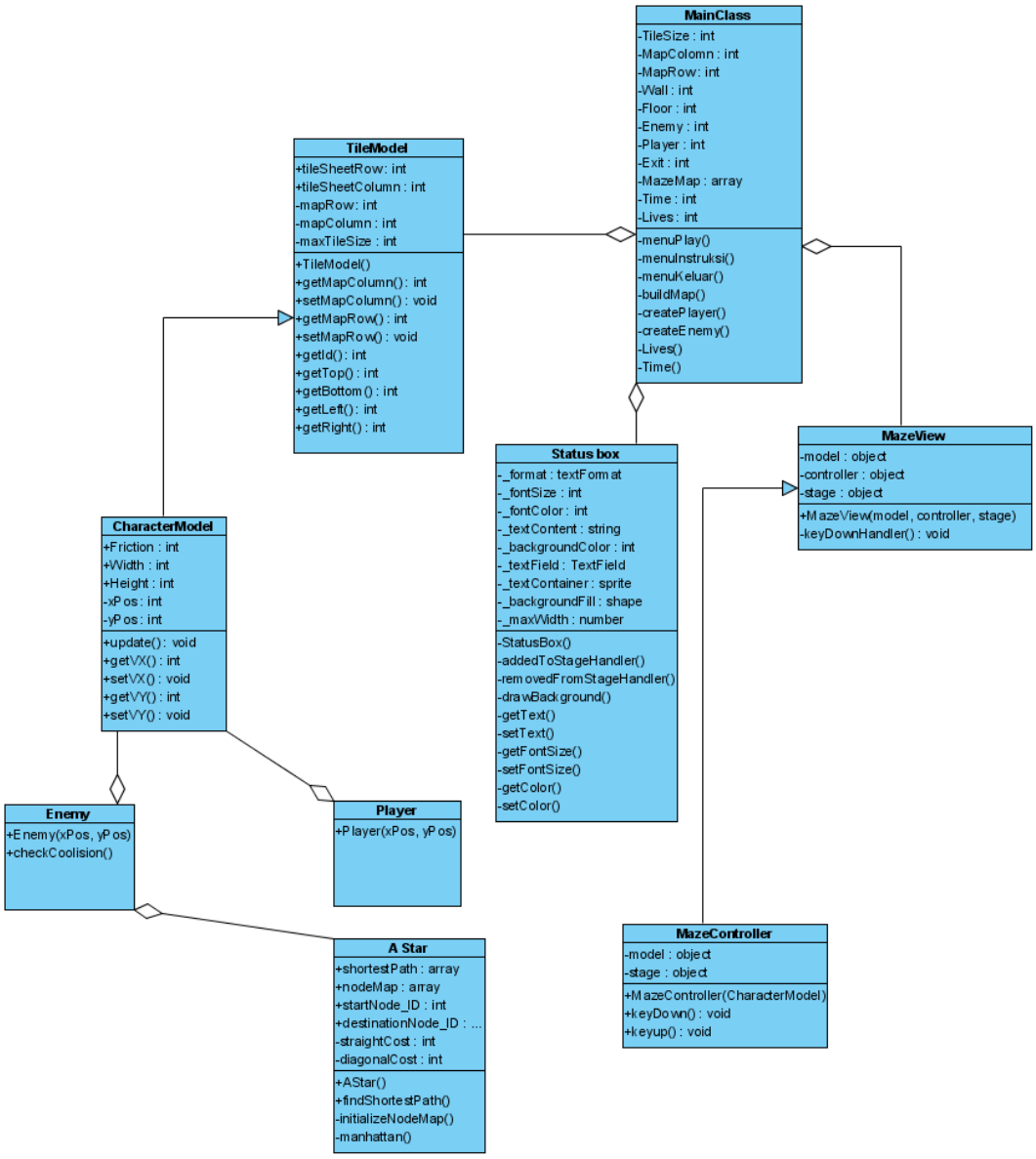
Sequence diagram untuk *use case* main menggambarkan interaksi antara objek dari *class* MainClass dengan *class* menu instruksi.



Gambar 3.19 *Sequence diagram* instruksi

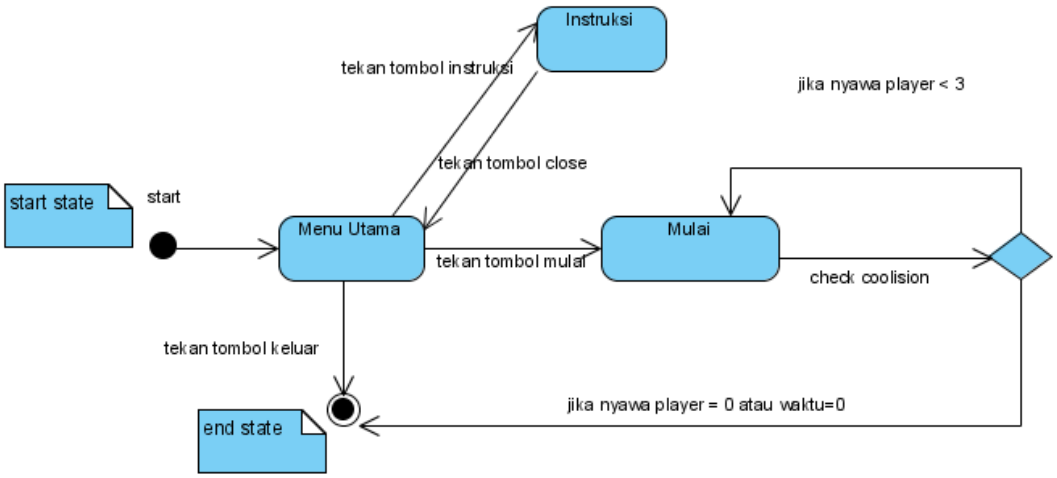
3.3.5 Class Diagram

Class Diagram menggambarkan struktur dan hubungan antar objek-objek yang ada pada sistem. Struktur itu meliputi atribut-atribut dan *method-method* yang ada pada masing-masing *class*, sedangkan hubungnnya meliputi pewarisan asosiasi, generalilasi dll.



Gambar 3.20 *Class diagram*

3.3.6 *State Diagram*



Gambar 3.21 *State diagram*

3.4 Perancangan Sistem

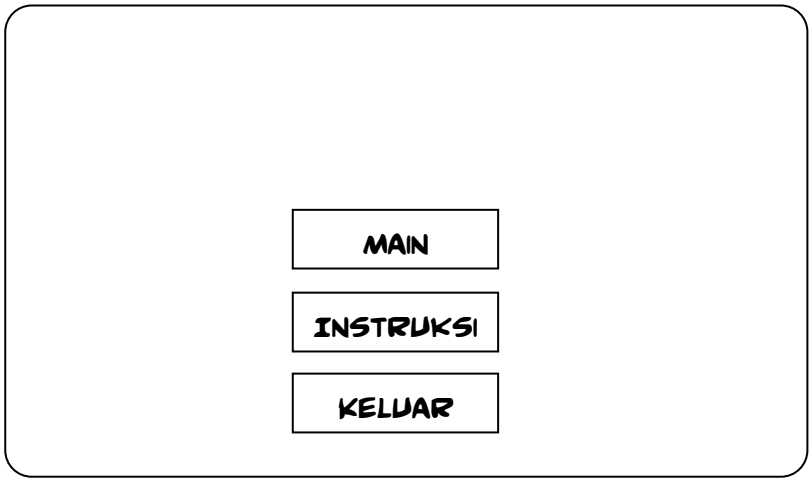
Perancangan sistem merupakan metodologi pengembangan suatu perangkat lunak yang dilakukan setelah melalui tahapan analisis. Dalam tahap ini digambarkan rancangan sistem yang akan dibangun sebelum dilakukan pengkodean ke dalam suatu bahasa pemrograman.

3.4.1 *Storyline*

Permainan jalan pintas menceritakan tentang seorang anak kucing yang tersesat ketika sedang bermain, saat akan kembali pulang ternyata dia lupa jalan pulang menuju rumahnya sedangkan dia harus segera pulang karena jika sampai tidak tepat waktu maka tidak akan diijinkan masuk ke rumah. Ternyata tempat yang dikunjunginya banyak penjahat yang suka mengganggu, dia harus mencari jalan pintas agar cepat sampai menuju rumahnya dan berusaha menghindari penjahat yang akan mengejarnya.

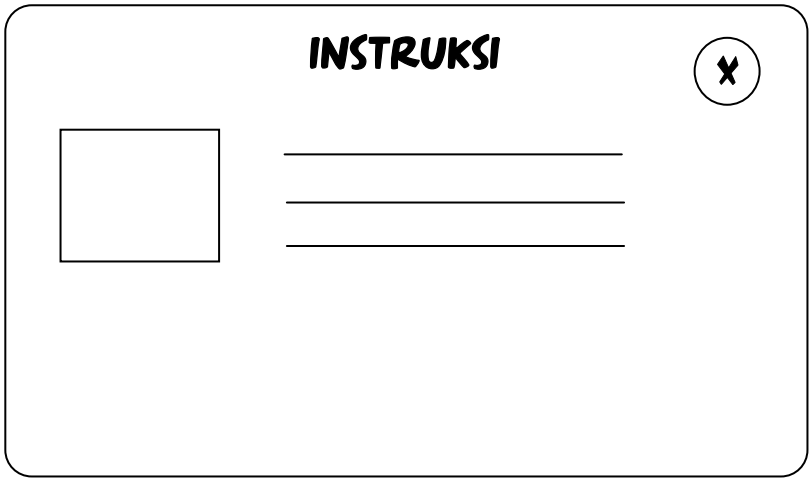
3.4.2 *Storyboard*

Pada saat membuka aplikasi *game* akan menampilkan menu utama dengan tiga buah tombol, pemain dapat langsung memainkan *game* atau memilih menu yang ada.



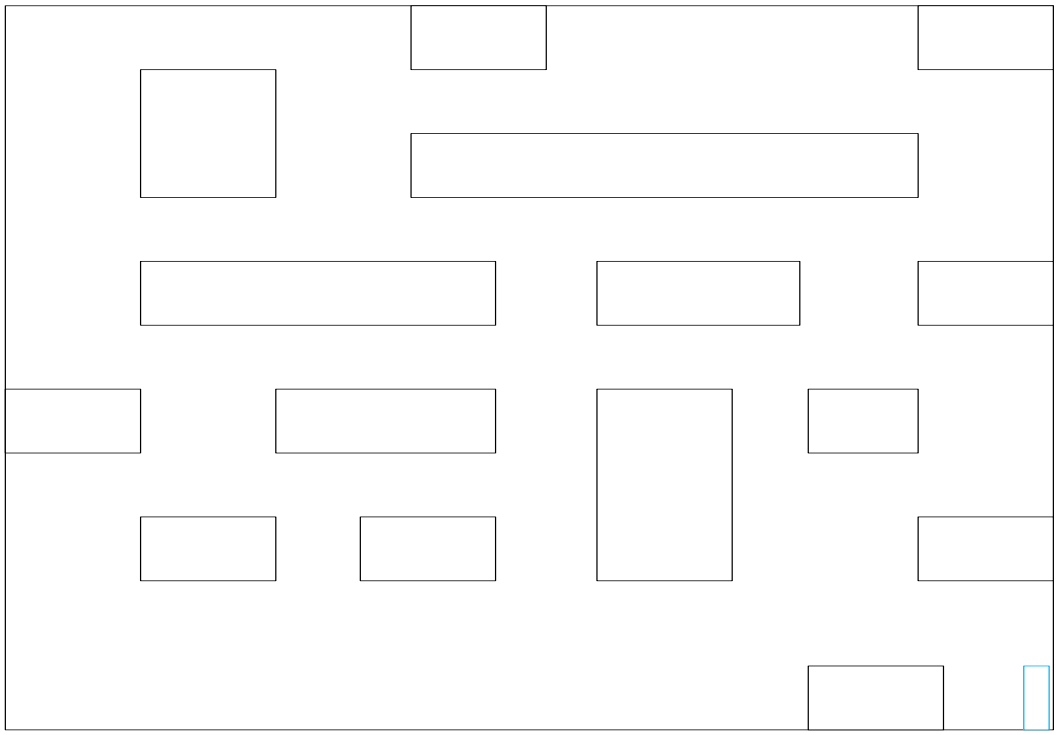
Gambar 3.22 Menu utama

Berikut ini adalah *story board* info instruksi setelah pemain memilih menu instruksi pemain akan dihadapkan pada tampilan info instruksi.



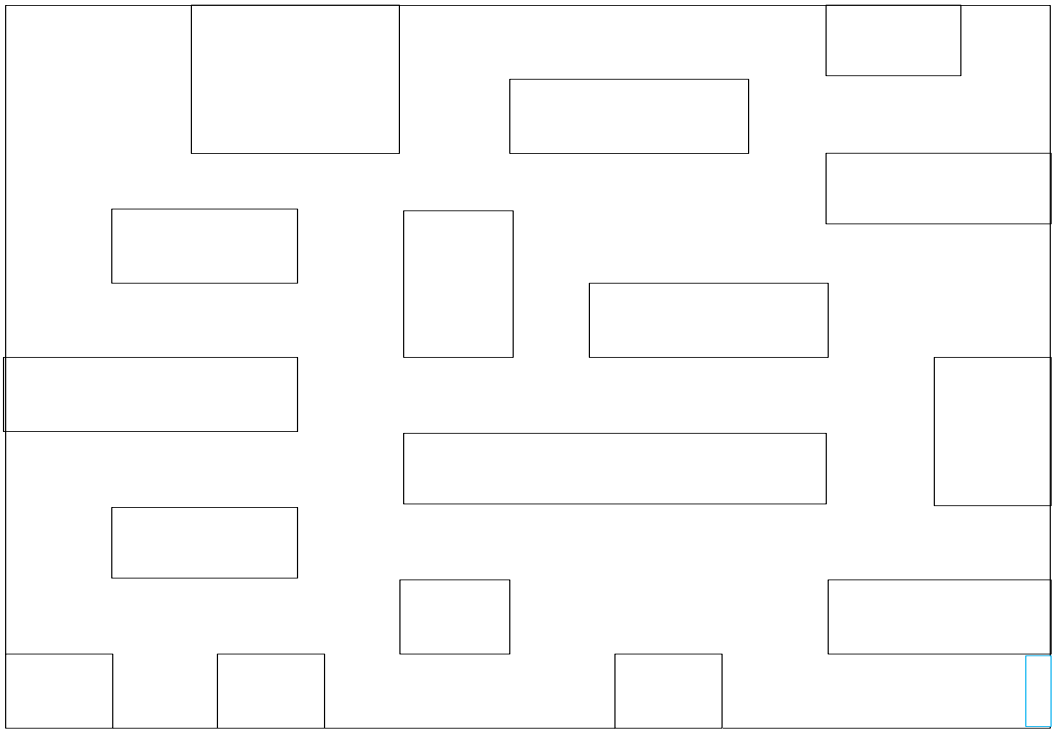
Gambar 3.23 Tampilan info instruksi

Setelah pemain memilih menu Main maka pemain dapat langsung memulai permainan mulai dari *level* 1, apabila sudah sampai ke tempat tujuan pemain akan memulai kembali permainan dan akan bertambah *level*.



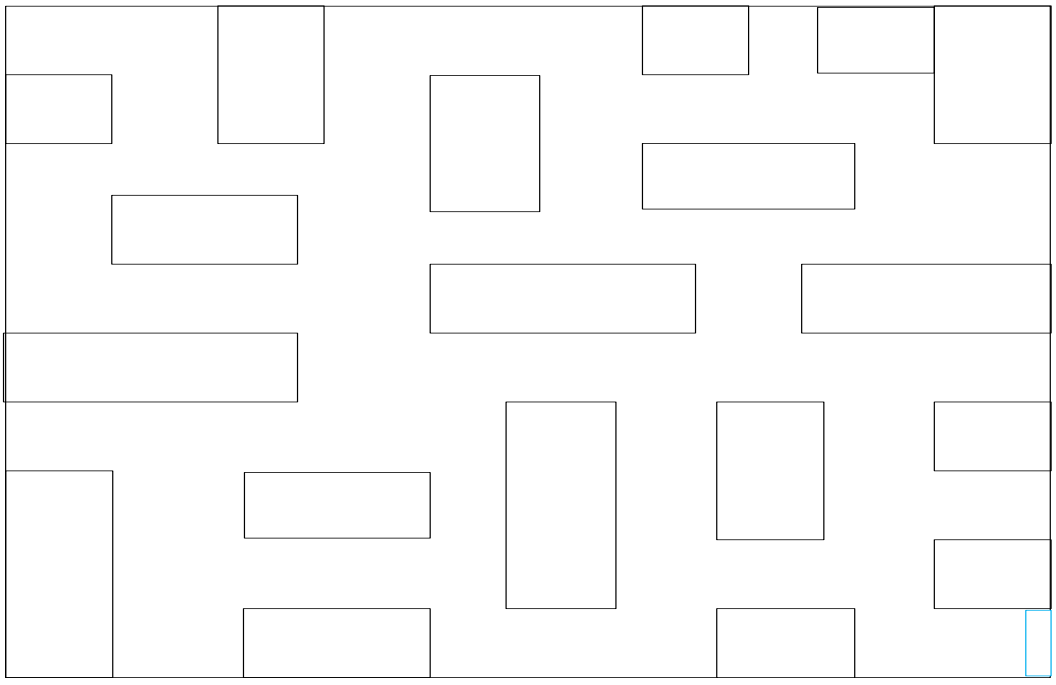
Gambar 3.24 Tampilan *level* 1

Setelah pemain sampai menuju pintu keluar maka pemain akan dihadapkan menuju *level 2.*



Gambar 3.25 Tampilan *level* 2

Setelah *level 2* terlewati maka selanjutnya pemain akan menuju level 3 yang merupakan *level* terakhir dari permainan.



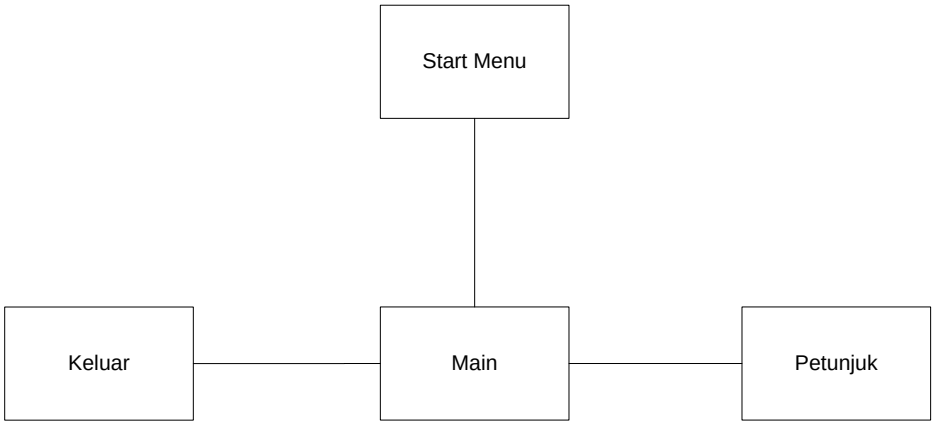
Gambar 3.26 Tampilan *level* 3

3.4.3 Karakter

Pada *game* jalan pintas ini terdapat tiga karakter, terdiri dari satu karakter *player* dan dua karakter musuh. Karakter *player* bisa bergerak ke kiri-kanan dan atas-bawah serta mempunyai tiga nyawa, sedangkan karakter musuh bergerak secara acak untuk mencari posisi *player* apabila musuh melihat *player* maka musuh akan mengejar dan setiap musuh diberi batasan jarak pandang.

3.4.4 Perancangan Struktur Menu

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



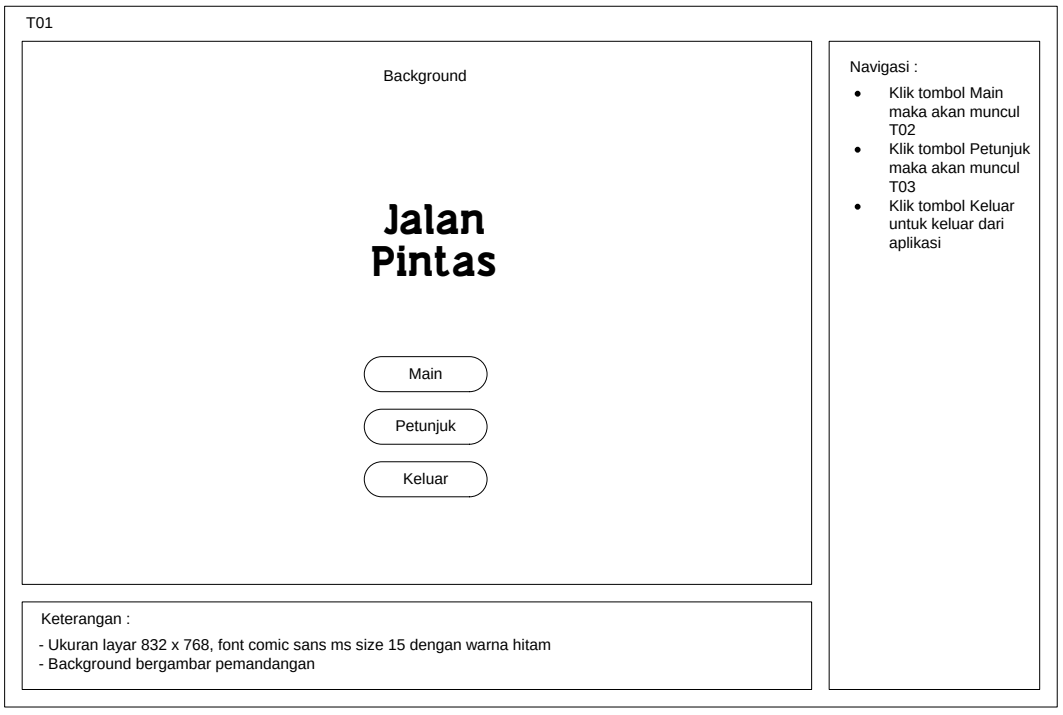
Gambar 3.27 Perancangan struktur menu

3.4.5 Perancangan Antarmuka

Perancangan antarmuka dilakukan untuk mempermudah damalam mengimplementasikan sistem yang akan dibangun.

3.4.5.1 Perancangan Antarmuka Menu Utama

Masuk ke menu utama *game* menampilkan tiga buah tombol pada layar yang akan dipilih oleh pemain untuk memainkan *game* langsung atau memilih menu-menu lainya.

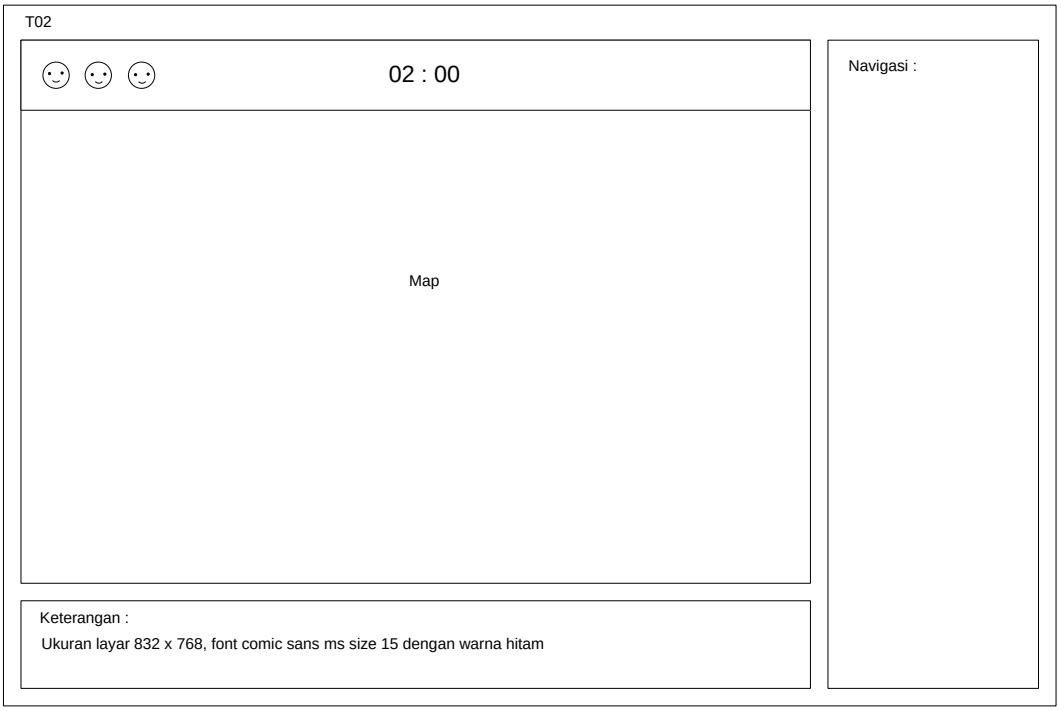


Gambar 3.28 Antarmuka menu utama

3.4.5.2 Perancangan Antarmuka *Game*

Masuk ke antarmuka *game*, pemain akan menghadapi jalan yang membingungkan untuk sampai ke tempat tujuan lalu pemain akan menghadapi

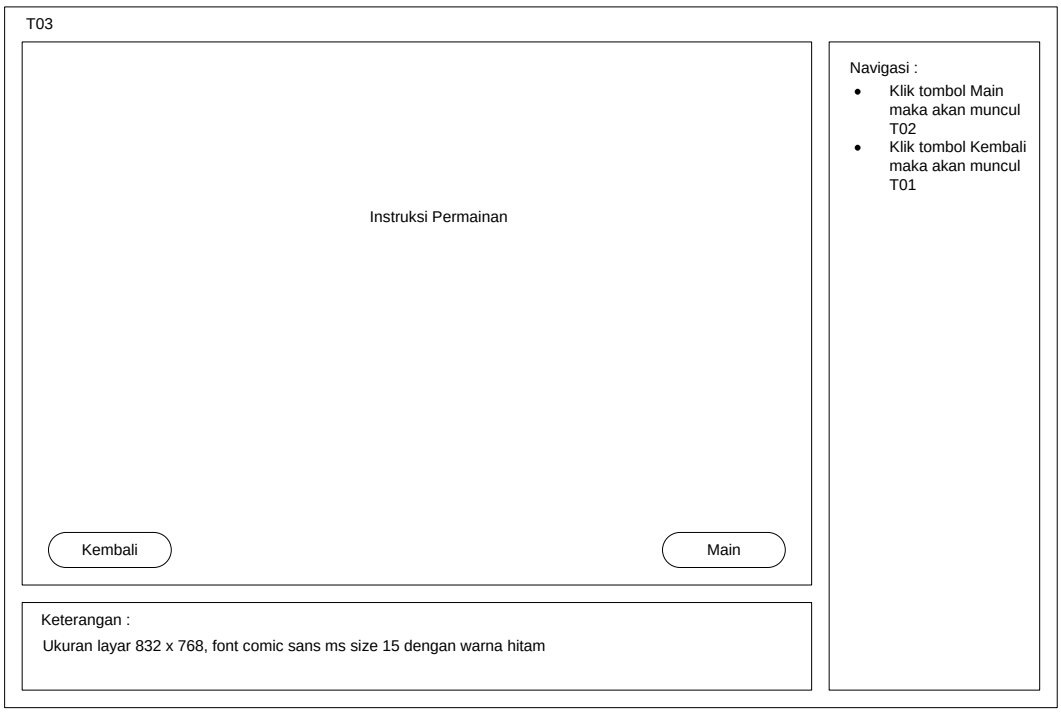
musuh yang akan menghadang selain itu juga pemain akan diberi tiga nyawa dalam permainan ini dan akan diberi waktu.



Gambar 3.29 Antarmuka *game*

3.4.5.3 Perancangan Antarmuka Instruksi

Masuk ke antarmuka instruksi, pemain akan diberikan instruksi tentang tata cara dalam memainkan *game*.



Gambar 3.30 Antarmuka instruksi

3.4.5.4 Perancangan Antarmuka Pesan

Antarmuka pesan akan muncul jika pemain terkena musuh sebanyak tiga kali atau waktu yang diberikan tidak cukup.



Gambar 3.31 Antarmuka pesan