BAB 3

ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Sistem

Pada bagian analisis sistem ini akan dilakukan berbagai macam analisis yang berhubungan dengan game yang akan dibangun, mulai dari analisis terhadap game sejenis analisis berbagai kebutuhan baik fungsional maupun non fungsional.

3.1.1 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 yaitu mendapatkan konsep permainan dan perbandingan dari *game* yang telah ada.

3.1.1.1 Pengenalan

Permainan *game Jump* adalah game yang *bergenre platformers*. *Game* ini adalah *game* 2D (dua dimensi) yang dibangun menggunakan java. Jenis dari *game* ini adalah *side scrolling* dimana pemain akan berjalan dari kiri ke kanan untuk mendapatkan koin-koin yang dijadikan *score*.

Berikut adalah tampilan dari *game jump* terdapat pada gambar 3.1:



Gambar 3. 1 *Screenshot* Tampilan Permainan *Game Jump*

3.1.1.2 Gameplay

Pada *game Jump* ini pemain harus mengumpulkan semua koin yang tersebar di arena permainan. Apabila berhasil mengumpulkan semua koin maka akan mendapatkan *score*. Diarena permainan terdapat musuh-musuh yang menghadang si pemain. Musuh bisa dibunuh atau hanya dilewati oleh pemain.

3.1.1.3 *Elemen* Game

Elemen permainan yang terdapat dalam permainan game Jump diantaranya adalah sebagai berikut:

1. Arena Permainan

   Pada permainan *game Jump* arena permainanya adalah di sebuah ruangan yang dipenuhi banyak rintangan. Terdapat rintangan berbentuk musuh-musuh yang menghadang dan terdapat ranjau. Diarena permainan inilah

pemain harus meloloskan diri dari kejaran musuh dan dari ranjau-ranjau yang terdapat dalam permainan.

2. Karakter pemain

   Karakter pemain merupakan entitas utama yang akan dikendalikan oleh pengguna dalam arena permainan ini. Pemain harus mengumpulkan semua koin yang terdapat dalam arena permainan, koin terkumpul akan menjadikan skor dalam permainan. Selain itu pemain juga harus menghindari mush dan ranjau-ranjau yang terdapat dalam permainan.

3. Karakter musuh

   Karakter musuh dalam permainan *game Jump* adalah entitas yang dapat bergerak dan tidak mengejar pemain. Apabila musuh menyentuh pemain maka darah pemain akan berkurang. Musuh terdiri dari beberapa jenis diantaranya sejenis kuman, dan bubble dimana kemampuan kedua musuh ini berbeda.

4. *Life point*

   Life Point adalah entitas yang berisi *health point* (batas kehidupan) yang dimiliki oleh tiap karakter pemain dan karakter musuh. *Life point* pada tiap karakter berbeda-beda. *Life point* juga menjadi entitas terpenting dalam permainan, karena menentukan tiap karakter dapat bertahan dalam arena permainan.

3.1.1.4 *Skenario Game*

Skenario dalam permainan *game Jump* disusun beberapa hal sebagai berikut:

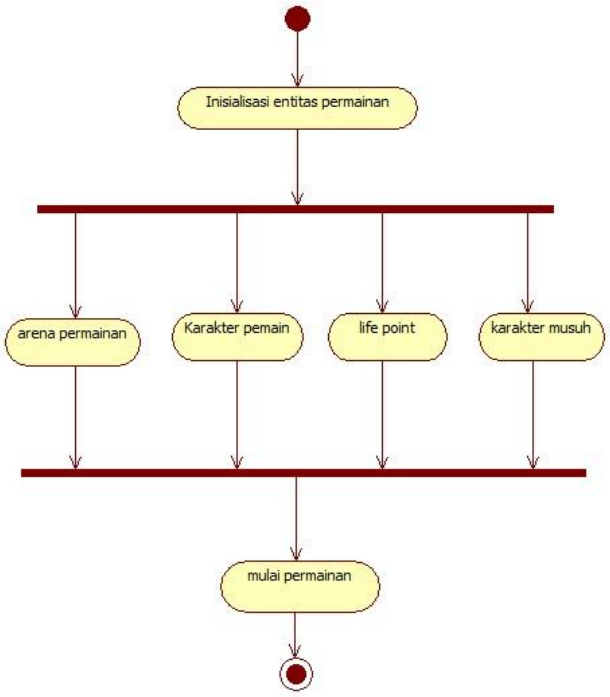
A. Alur sistem

   Alur Sistem pada *game Jump* adalah sebagai berikut:

   1. Sistem akan menginialisasi seluruh entitas permainan yaitu permainan, pemain, musuh, koin, ranjau, darah.

2. Musuh berada di posisi tertentu yang ditentukan secara acak oleh sistem.
3. Posisi pemain berada di sebalah kiri bawah.
4. Koin berada di sekitar arena permainan.
5. Permainan dimulai.
6. Pengguna mengontrol pergerakan pemain melalui melalui masukkan perintah keyboard.
7. Sistem akan melakukan update terhadap seluruh entitas.

   Adapun alur sistem dari game Jump dapat dilihat dalam gambar 3.2 :



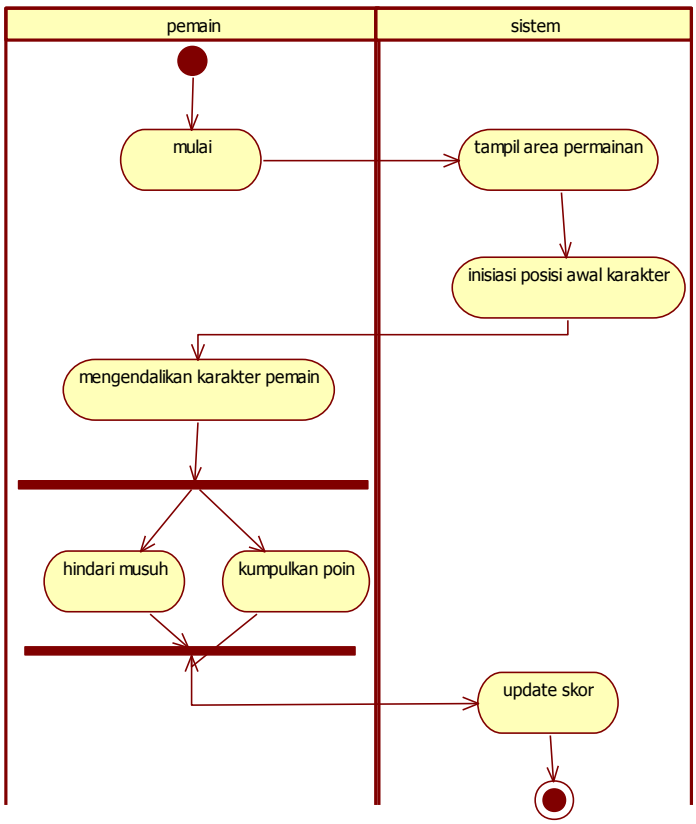
Gambar 3. 2 Alur Sistem Pada *Game Jump*

B. Alur Permainan

Alur permainan pada game Jump adalah sebagai berikut :

a. Pemain berusaha untuk keluar dari sebuah gedung yang dipenuhi dengan musuh yang menjaga gedung tersebut.
b. Pemain harus mengumpulkan koin untuk bisa keluar dari gedung tersebut.
c. Pemain harus menghindar dari musuh yang berada di arena permainan.
d. Apabila pemain mengenai musuh, maka darah akan berkurang dan apabila habis maka permainan akan berakhir.
e. Apabila pemain terperosok ke dalam ranjau, maka permainan akan berakhir.

Adapun alur permainan *Jump* dapat dilihat pada gambar 3.3



.:

Gambar 3. 3 Alur Permainan Pada *Game Jump*

C. Misi yang harus dicapai pemain

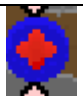
Pemain harus keluar dari gedung, dengan mengumpulkan koin di dalam arena permainan. Pemain menyusuri semua jalan yang ada di dalam permainan sebagai jalan keluar dari gedung tersebut.

3.1.1.5 Entitas Dalam *Game Jump*

Entitas-entitas yang terlibat dalam permainan *game Jump* dapat dilihat pada tabel 3.1 :

Tabel 3. 1 Entitas Dalam *Game Jump*

NO	NAMA	GAMBAR	KETERANGAN
1.	Pemain / *player*		Ini adalah karakter utama dalam game yang dikendalikan oleh pemain melalui inputan dari *keyboard*.
2.	*Girm Brown*		Ini adalah salah karakter musuh yang berada dalam game *jump*. musuh ini akan bergerak secara acak diarena permainan.
3.	*Beetle Green*		Ini adalah karakter musuh yang bernama *Beetle green* . musuh ini akan bergerak secara acak diarena permainan.
4.	*Bubble ghost*		Ini adalah karakter musuh yang bernama *Bubble ghost* . musuh ini akan bergerak secara acak diarena permainan.

4.	Koin		Ini adalah koin yang harus dikumpulkan oleh pemain .
5.	*Trap Jump*		Ini adalah item yang digunakan pemain untuk menjebak musuh. Apabila musuh terkena jebakan maka *life point* dari musuh akan berkurang
6.	*Life point*		Ini adalah item yang digunakan untuk menambah darah pemain.

3.1.2 Analisis *Game* yang Akan Dikembangkan

3.1.2.1 Pengenalan

Detective Code Out merupakan game berjenis *Platformers* yang akan dibangun. Pengambilan nama *detective* pada permainan ini memiliki arti detektif sedangkan *code out* memiliki arti memecahkan suatu masalah. Pada permainan *Detective Code Out*, detektif harus mengambil huruf-huruf yang tersebar di area permainan. *Detective Code out* merupakan game dengan sistem single player dan dibangun dengan grafis 2D (dua dimensi).

3.1.2.2 *Story Line*

Detective Code Out menceritakan tentang seorang detektif laki-laki bernama Brian. Brian adalah seorang detektif swasta terkenal yang ada di kota New York. Suatu hari di sebuah hotel di kota New York dihebohkan dengan sebuah kematian seorang direktur Bank terkemuka. Polisi yang datang ke TKP langsung melakukan olah TKP. Namun polisi yang bertugas menyelidiki kasus ini mengalami kesulitan untuk memecahkan kasus ini. Akhirnya atas permintaan polisi Brian diminta untuk ikut menyelidiki kasus ini, Brian pun menerima permintaan polisi untuk membantu mereka asalkan dalam melakukan tugasnya

Brian ingin melakukannya seorang diri. Brian menemukan beberapa petunjuk berupa huruf-huruf yang bertebaran di sekitar TKP sebagai bukti. Brian harus mengumpulkan huruf-huruf tersebut sebagai bukti yang sampai akhirnya ia dapat mengungkap siapa pelaku pembunuhan tersebut. Awalnya dalam mengumpulkan bukti-bukti Brian tidak mengalami masalah berarti, namun pada perjalanan mencari petunjuk Brian mendapatkan beberpa hadangan dari segerombolan preman. Segerombolan preman tersebut adalah gerombolan jack, gerombolan Billy.

Setelah mengalahkan segerombolan preman-preman dan mengumpulkan bukti, bukti tersebut mengarah kepada seorang mafia besar yang bernama Moggi Baressi. Akhirnya setelah lama bertarung Brian berhasil mengalahkan Moggi dan menyerahkannya ke polisi.

3.1.2.3 Tingkat Kesulitan

Dalam *game* ini, terdapat 3 level yang masing-masing level dibedakan berdasarkan alur cerita, tingkat kesulitan rintangan yang harus dihadapi, jumlah dan jenis musuh yang bertambah. Di setiap peningkatan level, pemain akan dihadapkan pada musuh dengan tingkat kesulitan yang akan bertambah.

Di *level* pertama, pemain dalam perjalanan menuju TKP pembunuhan yang berada di sebuah hotel. Di perjalanan, ia harus menghadapi segerombolan Jack, pemain harus mengalahkan semua musuh dan mengumpulkan huruf untuk dapat melanjutkan permainan ke *level* selanjutnya.

Di *level* kedua, pemain harus menghadapi segerombolan Jack dan segerombolan Billy, pemain harus mengalahkan semua musuh dan mengumpulkan huruf untuk dapat melanjutkan permainan ke *level* selanjutnya.

Di *level* ketiga, Akhirnya pemain telah mengetahui siapa pelaku utamanya yaitu Moggi, dengan jumlah musuh yang semakin banyak, ia pun harus mengalahkan segerombolan Jack,dan segerombolan,Billy. Setelah semua musuh dikalahkan, pemain harus mengalahkan Bos Utama yaitu Moggi dari kasus pembunuhan ini, Brian untuk memenangkan *game* ini.

3.1.2.4 *Gameplay*

Tugas utama pemain dalam memainkan permainan ini yaitu harus melalui 3 level, dimana tiap level memiliki tingkatan kesulitan yang berbeda, semakin tinggi level maka tingkat kesulitan akan semakin sulit. Tiap level permaian harus menyelesaikan misi dengan cara mengumpulkan huruf yang tersebar di arena permainan. Selain itu, pemain harus bisa mengalahkan musuh-musuh yang menghalangi pemain serta pemain dapat mengambil item-item yang tersedia di arena permainan seperti koin dan huruf.

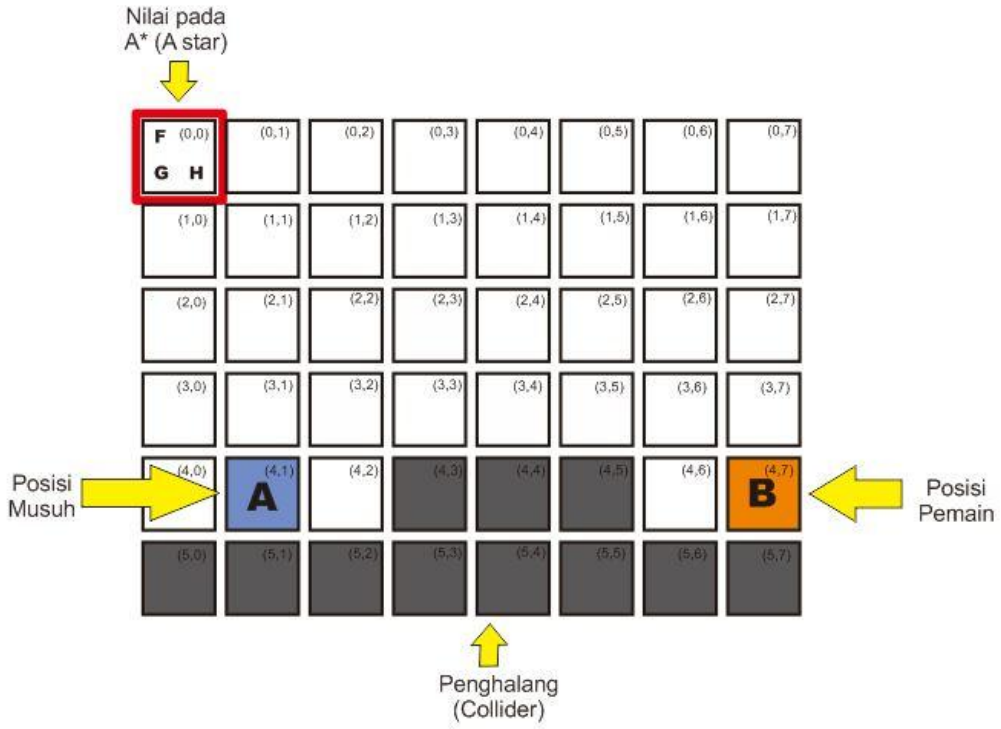
3.1.2.5 *Scoring*

Pada *game Detective Code Out* terdapat poin terhadap setiap musuh yang dikalahkan, poin yang diberikan terhadap musuh adalah 50 poin. Biasanya karakter memiliki kekuatan loncat hanya 2 kekuatan saja, tetapi karakter pemain dapat menggunakan berhasil mengumpulkan poin 200 maka akan memiliki kekuatan loncat sebanyak 3 kekuatan untuk satu kali loncat.

3.1.3 Analisis Algoritma

3.1.3.1 *Analisis A* (*A Star*)*

Algoritma A* di dalam *game* akan diterapkan pada musuh sebagai penentuan jalur terpendek atau *pathfinding* menuju *target.* Cara kerja algoritma A* di dalam *game detective code out* terdapat pada gambar 3.4.



Gambar 3. 4 Cara Kerja Algoritma A*

Pada gambar 3.4 terdapat beberapa kotak yang memiliki koordinat dari (0,1) sampai (5,7). Kotak atau *node* yang masing-masing telah memiliki koordinat merupakan *node* yang digunakan untuk melakukan pencarian jalur terpendek / *pathfinding*. Terdapat beberapa *node* yang memiliki warna berbeda, berikut penjelasan dari masing -masing *node* tersebut :

- *Node* warna biru : Node awal dari posisi musuh yang berada pada node di koordinat (4,1).
- *Node* warna abu : Node penghalang yang berada pada *node* di koordinat (4,3), (4,4), (4,5),(5,0).(5,1), (5,2), (5,3), (5,4), (5,5), (5,6), dan (5,7).
- *Node* warna merah : *Node* dari posisi pemain atau pemain yang berada pada *node* di koordinat (4,7).

Untuk nilai-nilai yang digunakan untuk perhitungan A star, terdapat 3 nilai yaitu :

- Nilai F, nilai yang diperoleh dari penjumlahan nilai G, jumlah nilai tiap simpul dalam jalur terpendek dari *starting point* ke *current node*, dan H, jumlah nilai perkiraan dari sebuah simpul ke *target point*.
- Nilai G, nilai atau biaya yang dikeluarkan dari *node* awal / *node* A ke *node* yang ada di sekitarnya.
- Nilai H, nilai *heuristic* yang digunakan untuk menghitung perkiraan biaya yang dikeluarkan dari *node* awal / *node* A menuju *node* tujuan / *node* B.

Perhitungan nilai F didapat dari persamaan F= G + H. Sedangkan untuk nilai G digunakan nilai 1 untuk gerakan *vertical* atau *horizontal*, Sedangkan untuk gerakan *diagonal* maka akan dikalikan dengan √2 yang berarti 1 x√2 = 1,4.

Untuk 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 , yang kemudian nilainya dikalikan dengan 10. Atau dirumuskan dengan:

H = 1*(abs(*current*X-*target*X) + abs(*current*Y-*target*Y))

Pada pencarian jalur terpendek oleh A star ada beberapa istilah yang digunakan seperti open list, closed list, parrent, current node dan pointer.

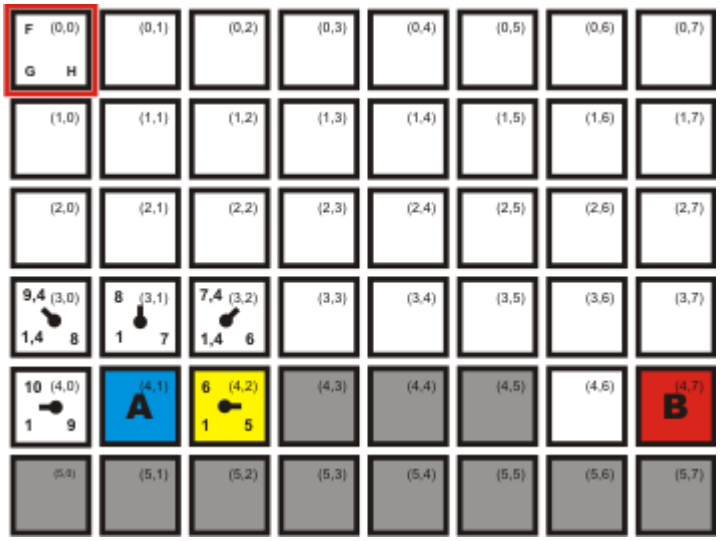
- *Open list* adalah *list* untuk memeriksa *node-node*.
- *Closed list* adalah *node-node* yang diabaikan atau *node-node* yang tidak perlu diperiksa.
- *Parrent* adalah digunakan untuk menelusuri jalan.
- *Current Node* adalah simpul yang sedang dijalankan dalam algoritma pencarian jalan terpendek.
- *Pointer* adalah penunjuk jalan terhadap *Node* A *(back* pointer) ketika *node* B (target) telah ditemukan.

Langkah berikutnya ketika telah mendapatkan node awal adalah menentukan rute terpendek dengan cara memeriksa setiap *node* yang berdekatan

dengan *node* A sampai *node* B. Di bawah ini merupakan penjelasan langkah-langkahnya.

1. Langkah pertama adalah memasukan *node* A kedalam *open list*.
2. Langkah kedua adalah memeriksa *node-node* yang berdekatan dengan *node* A, abaikan *node* dengan penghalang lalu tambahkan semua *node* tersebut kedalam *open list*. *Node* A sendiri nantinya akan menjadi "*parent*" dan digunakan untuk menelusuri jalan.
3. Langkah ketiga adalah menghapus *node* A dari *open list*, dan dimasukan ke dalam *closed list*.

Berikut ini perhitungan pada langkah pertama yang berada pada node dengan koordinat (4,2) terdapat pada gambar 3.5.



Gambar 3. 5 Langkah Pertama Perhitungan Algoritma A*

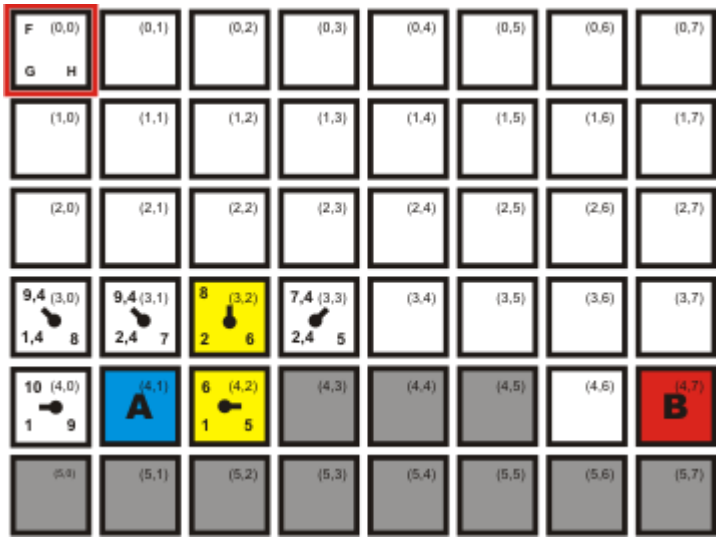
Pada langkah pertama, *node* (4,1) dimasukan ke dalam open list dan selanjutnya memeriksa *neighbor node* atau *node* yang berdekatan dengan *node* (4,1) yaitu, *node* (4,2), (4,0), (3,0), (3,1), (3,2). Setelah diperiksa *node-node* tersebut dimasukan ke dalam *open list*. Hapus *node* (4.1) dari *open list* dan masukan *node* tersebut kedalam *closed list.* Lalu dari *node-node* yang berada di dalam *open list* pilih *node* dengan nilai F yang paling rendah, yaitu *node* (4,2) dengan nilai F = 6.

Langkah pertama menghasilkan *node-node* di dalam *open list* dan *closed list* dijelaskan dengan tabel 3.2.

Tabel 3. 2 Perhitungan Langkah Pertama

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

Berikut ini perhitungan pada langkah kedua yang berada pada node dengan koordinat (4,2) terdapat pada gambar 3.6.



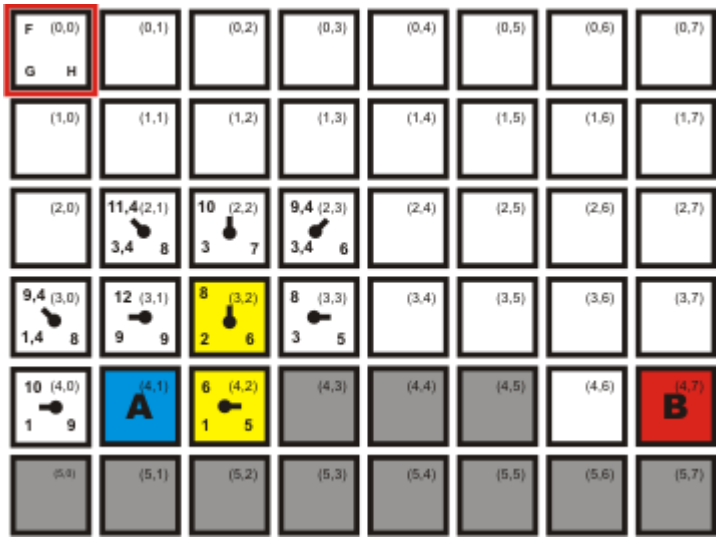
Gambar 3. 6 Langkah Kedua Perhitungan Algoritma A*

Berikut ini penjelasan melalui tabel untuk pencarian jalan terpendek pada langkah kedua terdapat pada tabel 3.3.

Tabel 3. 3 Tabel Perhitungan langkah kedua

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

Berikut ini perhitungan pada langkah ketiga yang berada pada *node* dengan koordinat (3,2) terdapat pada gambar 3.7.



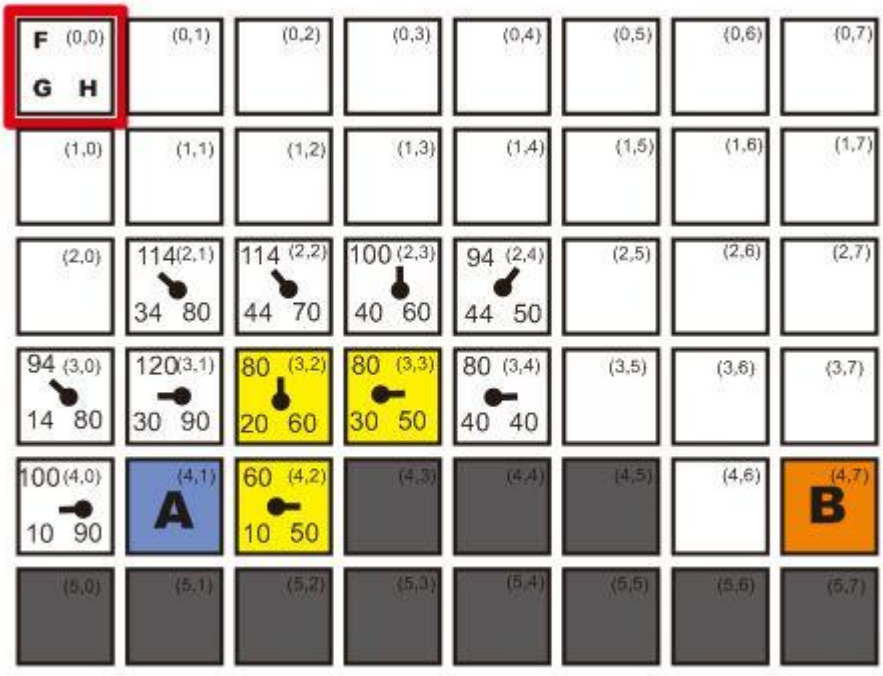
Gambar 3. 7 Penjelasan Astar Pada Langkah Ketiga

Berikut ini penjelasan melalui tabel untuk pencarian jalan terpendek pada langkah ketiga terdapat pada tabel 3.4.

Tabel 3. 4 Perhitungan langkah ketiga

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

Berikut ini perhitungan pada langkah keempat yang berada pada *node* dengan koordinat (0,3) terdapat pada gambar 3.8.



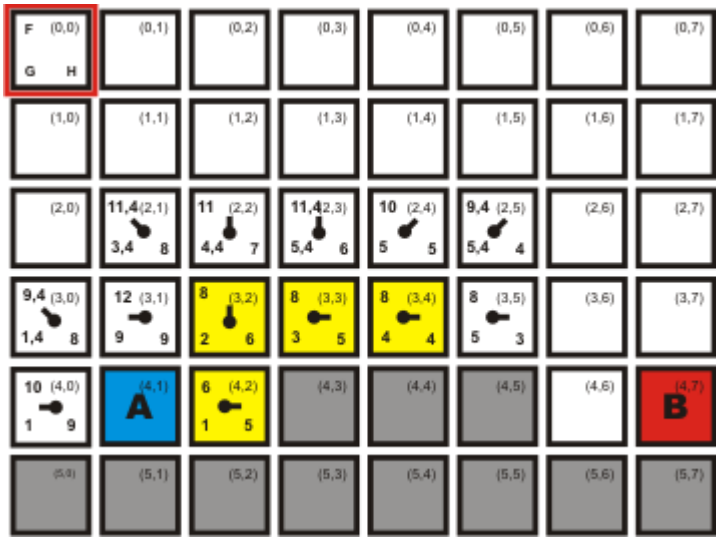
Gambar 3. 8 Penjelasan Astar pada langkah keempat

Berikut ini penjelasan melalui tabel untuk pencarian jalan terpendek pada langkah keempat terdapat pada tabel 3.5..

Tabel 3. 5 Tabel Perhitungan langkah keempat

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

Berikut ini perhitungan pada langkah keelima yang berada pada *node* dengan koordinat (0,4).



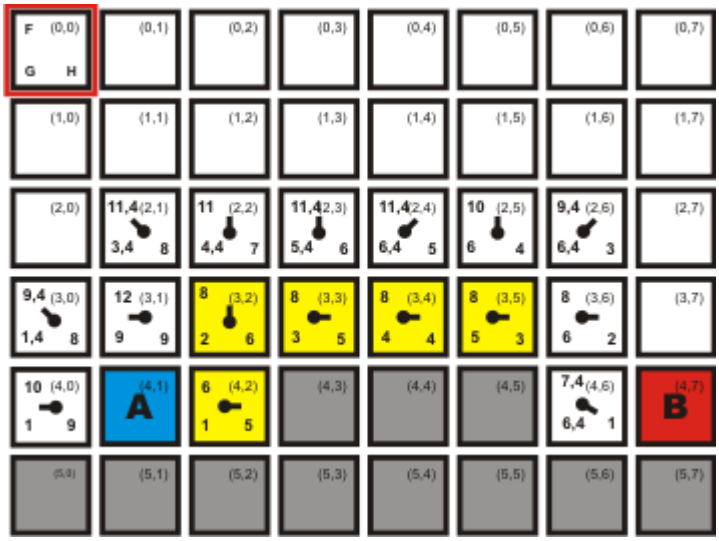
Gambar 3. 9 Penjelasan Astar Pada Langkah Kelima

Berikut ini penjelasan melalui tabel untuk pencarian jalan terpendek pada langkah kelima terdapat pada tabel 3.6.

Tabel 3. 6 Tabel Perhitungan Langkah Kelima

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

Berikut ini perhitungan pada langkah keenam yang berada pada *node* dengan koordinat (3,5).



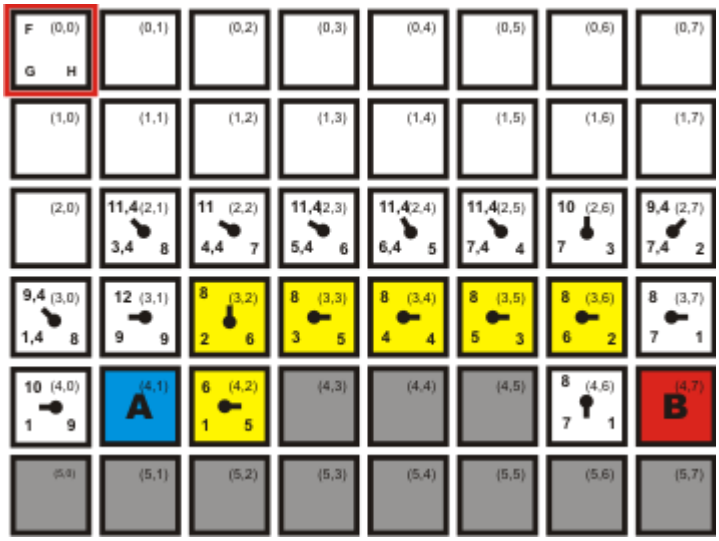
Gambar 3. 10 Penjelasan Astar Pada Langkah Keenam

Berikut ini penjelasan melalui tabel untuk pencarian jalan terpendek pada langkah keenam.

Tabel 3. 7 Tabel Perhitungan Langkah Keenam

Open List	(2,5), (2,6), (3,6),(4,6)
Closed list	(4,1), (4,2), (3,2), (3,3), (3,4), (3,5)
Parent	(3,5)
Current node	(3,6)

Berikut ini perhitungan pada langkah ketujuh yang berada pada *node* dengan koordinat (3,6).



Gambar 3. 11 Penjelasan Astar Pada Langkah Ketujuh

Berikut ini penjelasan melalui tabel untuk pencarian jalan terpendek pada langkah keenam.

Tabel 3. 8 Tabel Perhitungan Langkah Keenam

Open List	(2,5), (2,6), (2,7),(3,7)
Closed list	(4,1), (4,2), (3,2), (3,3), (3,4), (3,5),(3,6)
Parent	(3,6)
Current node	(4,6)

Penentuan jalan terbaik dengan melakukan pemeriksaan dimulai dari *target* atau *node* B, dalam ilustrasi adalah kotak berwarna kuning. Dengan mengikuti arah *pointer* yang berarti akan membawanya kembali ke *node* awal, maka akan didapatkan jalan terbaiknya, Maka jika diimplementasikan pada *game*, nantinya musuh akan melewati halangan dan akhirnya akan menemukan rute terpendek ke tujuan.

Berikut adalah *pseudo code* algoritma A star terdapat pada tabel 3.9.

Tabel 3. 9 *Pseudo Code* Algoritma A*

Algoritma A*	
1	create the open list of nodes, initially containing only our starting node
2	create the closed list of nodes, initially empty
3	while (we have not reached our goal) {
4	consider the best node in the open list (the node with the lowest f value)
5	if (this node is the goal) {
6	then we're done
7	}
8	else {
9	move the current node to the closed list

```
                and consider all of its neighbors
10              for (each neighbor) {
11                      if (this neighbor is in the closed
                        list and our current g value is
                        lower) {
12                              update the neighbor with
                                the new, lower, g value

                                change the neighbor's
                                parent to our current node
13                      }
14                      else if (this neighbor is in the
                        open list and our current g value
                        is lower) {
15                              update the neighbor with
                                the new, lower, g value

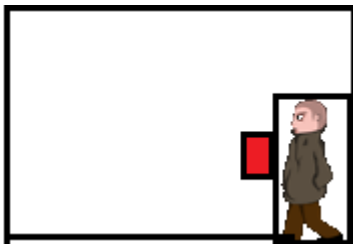
                                change the neighbor's
                                parent to our current node
16                      }
17                      else this neighbor is not in either
                        the open or closed list {
18                              add the neighbor to the
                                open list and set its g
                                value
19                      }
20              }
21      }
22  }
```

3.1.3.2 Analisis *Collision Detection*

Agar aplikasi permainan *detective code out* ini menjadi lebih menarik, maka akan diterapkan metode *collision detection* yang berupa metode dimana yang bersinggungan merupakan objek secara langsung, yaitu *sprite* dengan *sprite* maupun *sprite*.

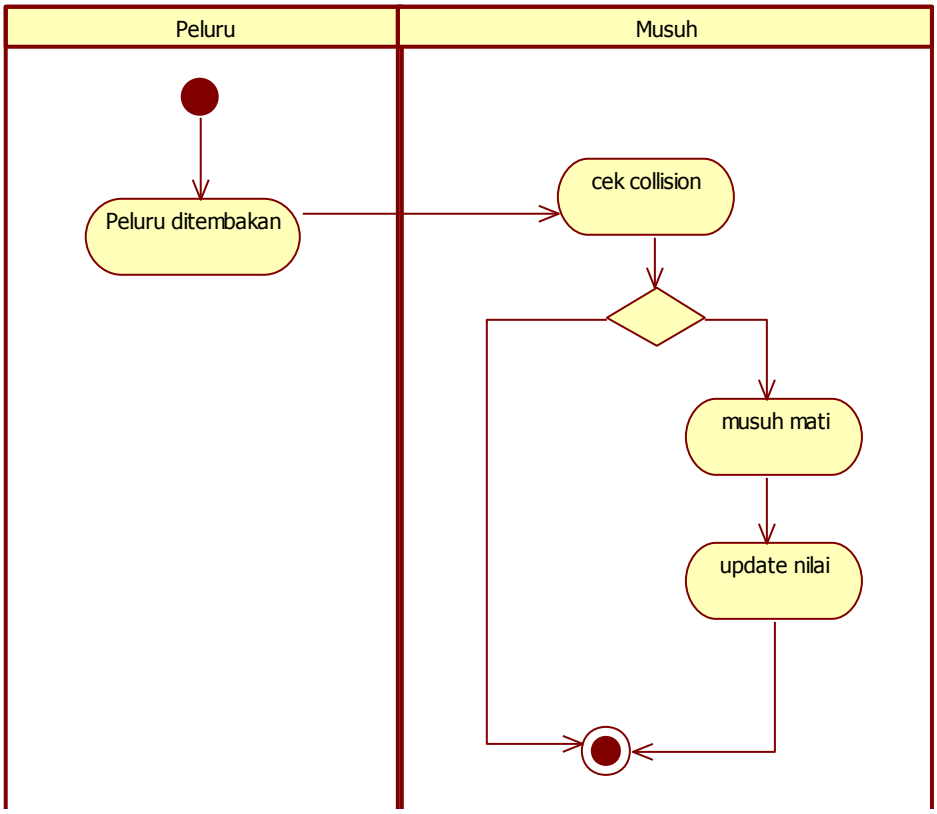
Pada aplikasi permainan *Detective code out* , program akan mendeteksi terjadinya tabrakan ketika bound kedua sprite bersinggunan meskipun secara kasat mata sprite tersebut sama sekali tidak bersinggungan.

Gambar metode collision peluru dengan musuh ditunjukan pada gambar 3.12.



Gambar 3. 12 *Collision Detection* Peluru dengan Musuh

Diagram *Activity metode collision* peluru dengan musuh ditunjukan pada gambar.3.13.



Gambar 3. 13 Anilisis *Collision* Tembakan Pemain Menyerang Musuh

Pseudo code collision peluru dengan musuh ditunjukan pada tabel 3.10.

Tabel 3. 10 *Pseudo Code Collision* Peluru dengan Musuh

1	If sprite1_X < (sprite2_x + sprite2_lebar) AND Sprite2_X < (sprite1_X + sprite1_lebar) AND Sprite1_Y < (sprite2_Y + sprite2_lebar) AND Sprite2_Y < (sprite1_Y +sprite1_lebar)
2	Collision terjadi
3	Else Collision tidak terjadi
4	Else if

3.1.4 Analisis Kebutuhan Non Fungsional

3.1.4.1 *Analisis Kebutuhan Perangkat Keras*

Analisis perangkat keras digunakan untuk mengetahui spesifikasi perangkat keras untuk membangun aplikasi permainan *Detective code out***.** Berikut adalah spesifikasi perangkat keras terdapat pada tabel 3.11.

Tabel 3. 11 Spesifikasi Perangkat Keras

Nama Perangkat	Spesifikasi
Prosesor	2.20 GH
RAM	4 GB
Harddisk	250 GB
Display	Resolusi 1366 x 768
Keyboard	Standar
Mouse	Standar

3.1.4.2 Analisis Kebutuhan Perangkat Lunak

Analisis perangkat lunak digunakan untuk mengetahui spesifikasi perangkat lunak untuk membangun aplikasi permainan *Detective code out* **,** Berikut adalah spesifikasi perangkat lunak terdapat pada tabel 3.12.

Tabel 3. 12 Analisisi Kebutuhan Perangkat Lunak

Nama Perangkat Lunak	Spesifikasi
Sistem Operasi	Microsoft Windows 7 Ultimate
Tools	JDK 6

3.1.4.3 Analisis *User*

Ada 2 hal yang akan memudahkan pemain ketika memainkan *game* ini yaitu pengetahuan dan pengalaman, terutama pengetahuan dan pengalaman dalam memainkan *game Detective Code Out*.

1. *User Knowledge and Experience*

Berikut ini klasifikasi *knowledge and experience* dari pengguna aplikasi terdapat pada tabel 3.13.

Tabel 3. 13 *Knowledge and Experience*

Education Level	Bisa digunakan oleh berbagai kalangan, seperti pelajar, mahasiswa hingga masyarakat awam.
Reading Level	Bisa digunakan oleh berbagai *level* pendidikan dengan reading *level* yang sedang.
Typing Skills	Tidak memerlukan *typing skills* yang tinggi.
Computer Literacy	Moderate (menengah)
Task Experience	Bisa digunakan oleh dengan pengalaman penggunaan komputer dan *game* yang sedang.
System Experience	Bisa digunakan oleh dengan pengalaman penggunaan komputer dan *game* yang sedang.
Application Experience	Bisa digunakan dalam semua sistem operasi Windows xp / vista/ 7.
Native Language	Menggunakan satu bahasa, yakni Indonesia.
Use Of Other System	Bisa di jalankan tanpa perlu install applikasi lain.

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.*

Berikut ini klasifikasi *Users Physical Characteristic* dari pengguna aplikasi terdapat pada tabel 3.14.

Tabel 3. 14 *Users Physical Characteristic*

Age	6 tahun ke atas [15]
Gender	Pria dan Wanita
Handedness	Kanan dan Kiri
Color Blind	*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.

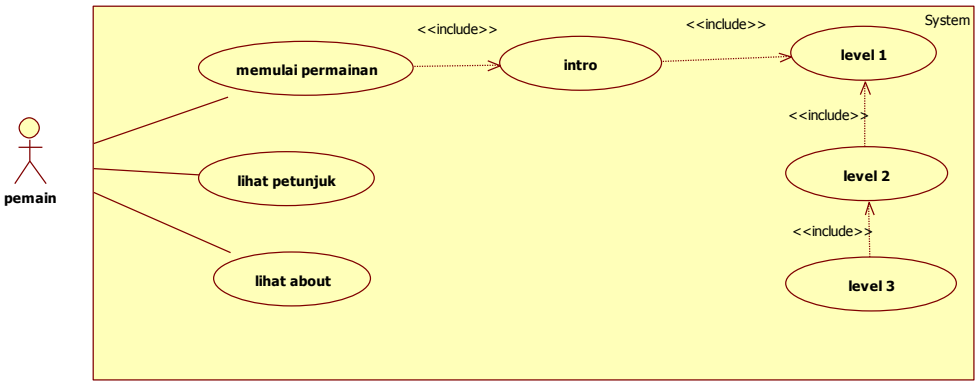
3.1.5 Analisis Kebutuhan Fungsional

Identifikasi aktor dapat dilakukan dalam analisis berorientasi objek dengan menggunakan UML yaitu menentukan aktor atau pengguna sistem. Aktor dalam konteks UML menampilkan peran pengguna atau sesuatu diluar sistem yang dikembangkan dapat berupa perangkat keras, end user, sistem yang lain dan sebagainya.

3.1.5.1 Use Case Diagram

Use case adalah abstraksi dari interaksi antara sistem dan dan aktor. Use case mendeskripsikan bagaimana sistem akan terlihat di mata user.

Secara umum, interaksi antara pengguna dengan aplikasi permainan *Detective code out* digambarkan pada diagram *use case* pada gambar 3.14



Gambar 3. 14 *Use Case Diagram Dtective Code Out*

3.1.5.2 Skenario *Use Case*

Skenario pada diagram *use case* berisi tentang kejadian *'flow if event' use case* utama. Di dalamnya menjelaskan urutan interaksi antara aktor dengan *use case* tersebut dari awal sampai akhir. Adapun scenario *use case* yang terdapat pada aplikasi permainan *Detective Code Out* ini adalah sebagai berikut :

1. **Skenario *Use Case* Memulai Permainan**

Skenario *use case* ini menjelaskan interaksi antara aktor dan sistem , yaitu pemain, dengan *use case* mulai permainan yang dijelaskan pada tabel 3.15.

Tabel 3. 15 Skenario *Use Case* Memulai Permainan

Identifikasi		
Nama	*Intro*	
Tujuan	Memulai permainan	
Aktor	Pemain	
Skenario		
Kondisi Awal	Menu Utama ditampilkan	
Aksi Aktor		**Reaksi Sistem**
a. Pilih Mulai Permainan		
		b. Masuk intro
Kondisi Akhir	Memasuki layar intro	

2. **Skenario *Use Case Intro***

Skenario *use case* ini menjelaskan interaksi antara aktor dan sistem , yaitu pemain, dengan *use case* mulai permainan yang dijelaskan pada tabel 3.16.

Tabel 3. 16 Skenario Use Case Intro

Identifikasi		
Nama	Intro	
Tujuan	Masuk ke halaman intro	
Aktor	Pemain	
Skenario		
Kondisi Awal	Menu Utama ditampilkan	
Aksi Aktor		**Reaksi Sistem**
a. Pilih Mulai Permainan		
		b. Masuk intro

Kondisi Akhir	Pemain masuk ke layar permainan level 1

3. **Skenario Use Case Level 1**

Skenario *use case* ini menjelaskan interaksi antara aktor, yaitu pemain, dengan *use case* Level 1 yang dijelaskan pada tabel 3.17.

Tabel 3. 17 Skenario *Use Case* Level 1

Identifikasi	
Nama	Level 1
Tujuan	Masuk ke arena permainan kemudian mulai bermain dan menyelesaikan level 1
Aktor	Pemain
Skenario	
Kondisi Awal	Permainan telah dimulai
Aksi Aktor	**Reaksi Sistem**
a. Pilih Mulai Permainan	
	b. Tampilkan layar permainan Level 1
	c. Inisialisasi posisi pemain
	d. Inisialisasi huruf
	e. Inisiasi posisi penghalang
	f. Inisialisasi posisi musuh
g. Mengendalikan karakter Pemain	
	h. Musuh jack keluar
	i. Jalankan *AI*
	j. Musuh mengejar pemain
k. Mengalahkan musuh	

	l. *Update* status musuh
Kondisi Akhir	Menyelesaikan level 1

4. **Skenario Use Case Level 2**

Skenario *use case* ini menjelaskan interaksi antara aktor, yaitu pemain, dengan *use case* level 2 yang dijelaskan pada tabel 3.18.

Tabel 3. 18 Skenario Use Case Level 2

Identifikasi	
Nama	Level 2
Tujuan	Masuk ke arena permainan level 2 kemudian mulai bermain dan menyelesaikan level 2
Aktor	Pemain
Skenario Normal	
Kondisi Awal	Permainan Telah dimulai dan telah menyelesaikan level 1
Aksi Aktor	**Reaksi Sistem**
a. Pilih Mulai Permainan	
	b. Tampilkan layar permainan Level 2
	c. *Inisialisasi* semua konten yang dibutuhkan permainan di level 2
	d. *Inisialisasi* huruf
	e. Inisiasi posisi penghalang
	f. Menempatkan posisi karakter pemain di sebelah kiri atas arena permainan

	g. inisiasi posisi musuh
h. Mengendalikan karakter Pemain	
	i. Musuh jack keluar
	j. Musuh billy keluar
	k. Jalankan *AI*
	l. Semua musuh mengejar pemain
m. Mengalahkan musuh	
	n. *Update* status musuh
Kondisi Akhir	Menyelesaikan level 2

5. **Skenario Use Case Level 3**

Skenario *use case* ini menjelaskan interaksi antara aktor, yaitu pemain, dengan *use case* Level 3 yang dijelaskan pada tabel 3.19.

Tabel 3. 19 Skenario Use Case Level 3

Identifikasi	
Nama	Level 3
Tujuan	Masuk ke arena permainan kemudian mulai bermain dan menyelesaikan level 3
Aktor	Pemain
Skenario Normal	
Kondisi Awal	Permainan Telah dimulai dan telah menyelesaikan level 2
Aksi Aktor	**Reaksi Sistem**
a. PilihMulai Permainan	
	b. Tampilkan layar permainan Level 3
	c. *Inisialisasi* semua konten yang dibutuhkan permainan di level 3

	d. *Inisialisasi* huruf
	e. Inisiasi posisi penghalang
	f. Menempatkan posisi karakter pemain di sebelah kiri atas arena permainan
	g. inisiasi posisi musuh
g. Mengendalikan karakter Pemain	
	h. Musuh jack keluar
	i. Musuh billy keluar
	j. Jalankan *AI*
	k. Semua musuh mengejar pemain
	l. Musuh Moggi keluar
m. Mengalahkan musuh	
	n. *Update* status musuh
Kondisi Akhir	Menyelesaikan level 3 dan permainan selesai

6. **Skenario Use Case Lihat Petunjuk**

Skenario *use case* ini menjelaskan interaksi antara aktor, yaitu pemain, dengan seluruh *use case* Petunjuk yang dijelaskan pada tabel 3.20 :

Tabel 3. 20 Skenario Use Case Lihat Petunjuk

Identifikasi	
Nama	Petunjuk
Tujuan	Masuk ke submenu yang berisi petunjuk *game*
Aktor	Pemain
Skenario Utama	

Kondisi Awal	Menu Utama ditampilkan	
Aksi Aktor		**Reaksi Sistem**
a. Pilih Petunjuk		
		b. Masuk ke submenu Petunjuk
		c. Menampilkan Layar info cara memainkan *game*
Kondisi Akhir	Menampilkan informasi bagaimana memainkan aplikasi *detective code out*	

7. **Skenario *Use Case* Lihat About**

Skenario *use case* ini menjelaskan interaksi antara aktor, yaitu pemain, dengan seluruh *use case* About yang dijelaskan pada tabel 3.21.

Tabel 3. 21 Skenario *Use Case* Lihat About

Identifikasi		
Nama	About	
Tujuan	Masuk ke submenu yang berisi informasi about *game*	
Aktor	Pemain	
Skenario Utama		
Kondisi Awal	Menu Utama ditampilkan	
Aksi Aktor		**Reaksi Sistem**
a. Pilih about		
		b. Masuk ke submenu about
		c. Menampilkan Layar informasi pembuat game
Kondisi Akhir	Menampilkan informasi tentang pembuatan aplikasi game detective code out	

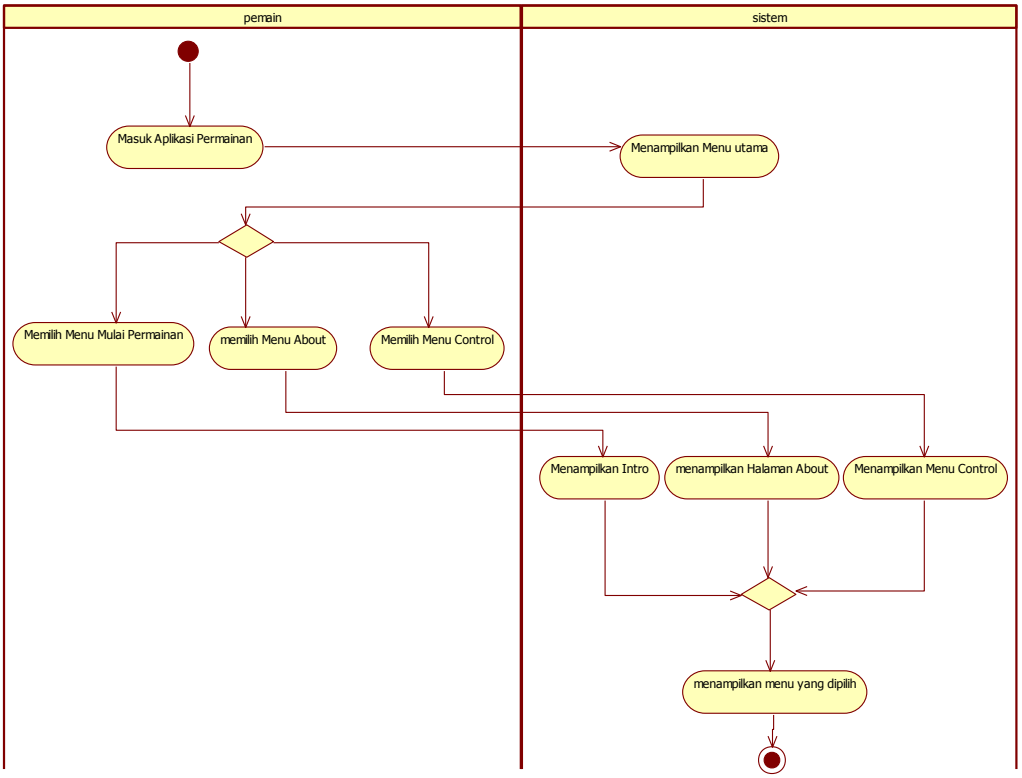
3.1.5.3 *Activity Diagram*

Activity Diagram berisi penjelasan lebih lengkap mengenai urutan mengenai urutan suatu proses yang terjadi antara pelaku dan sistem. Aktivitas yang pertama selalu dimulai dari pelaku, kemudian akan mendapat respon dari sistem yang nantinya akan menampilkan akhir dari proses tersebut.

1. *Activity Diagram* Mulai Permainan

Proses ini terjadi saat pemain memilih menu mulai permainan pada halaman utama permainan *Detective code out.* Sistem akan menampilkan arena permainan, kemudian sistem akan menginisialisasi entitas yang ada dalam permainan,

Activity diagram mulai permainan level 1 terdapat pada gambar 3.15.

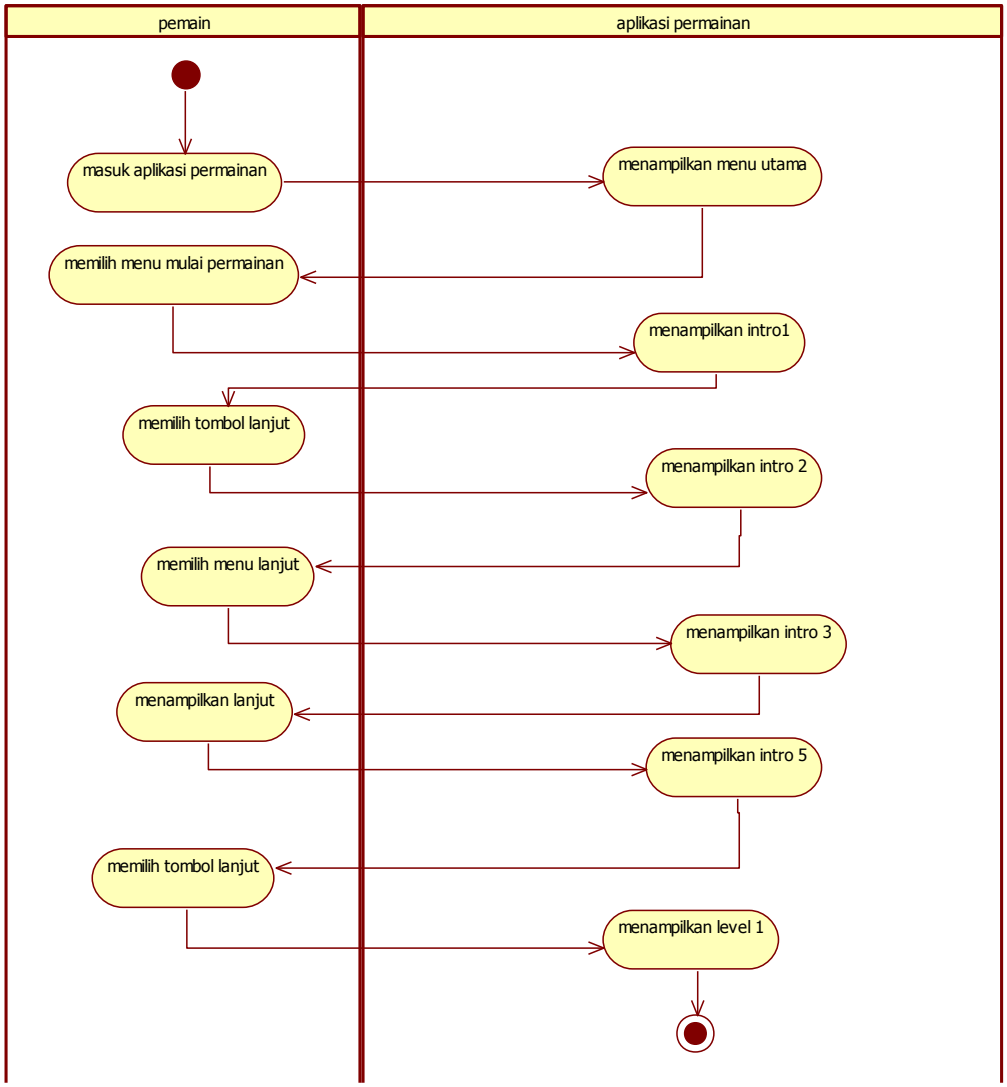


Gambar 3. 15 *Activity Diagram* Mulai Permainan

2. *Activity Diagram* Intro

Proses ini terjadi saat pemain memilih menu mulai permainan pada halaman utama permainan *game Detective code out*. Sistem akan menampilkan *intro.*

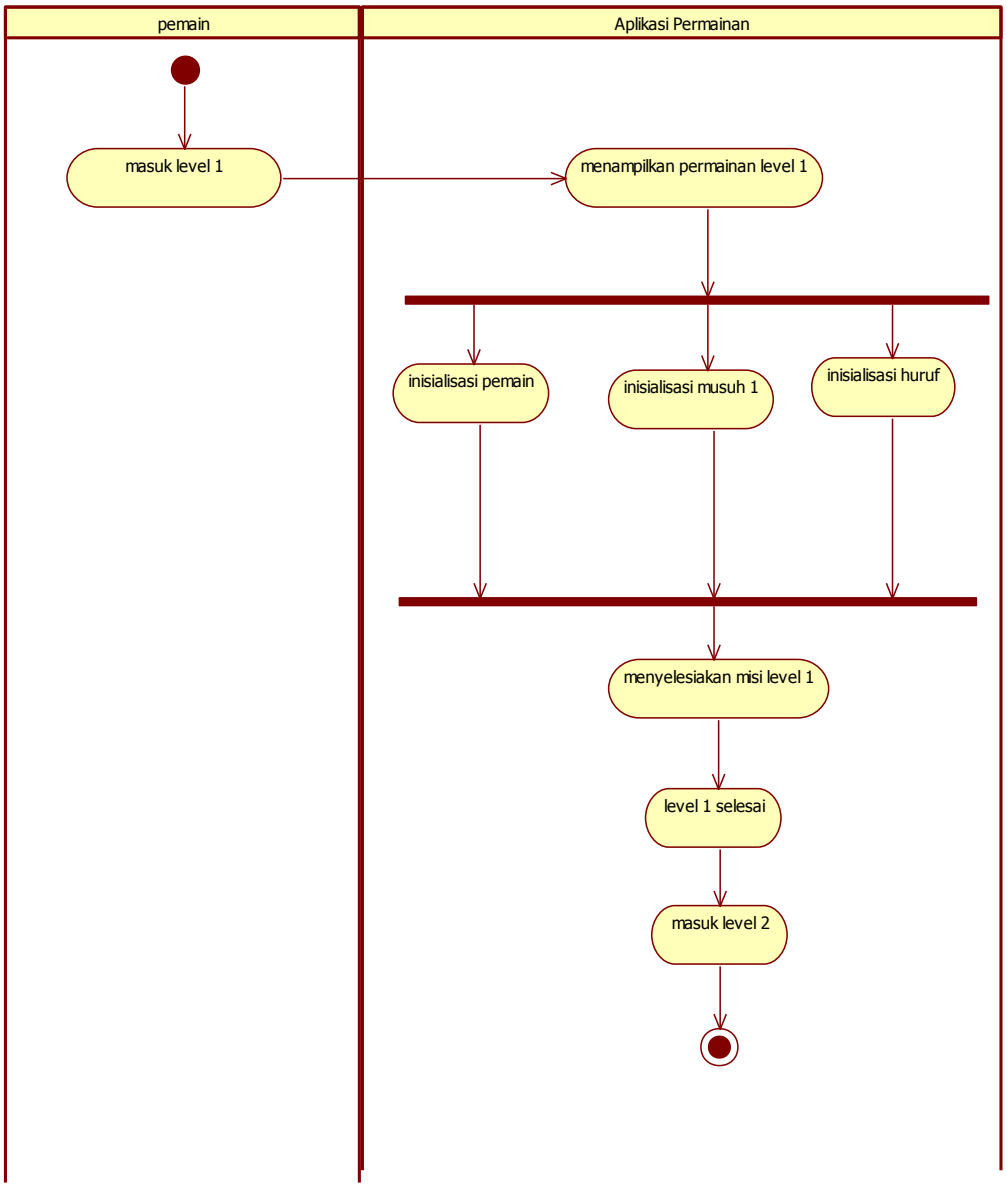
Activity diagram mulai permainan intro terdapat pada gambar 3.16.



Gambar 3. 16 *Activity Diagram* Intro

3. *Activity Diagram* Level 1

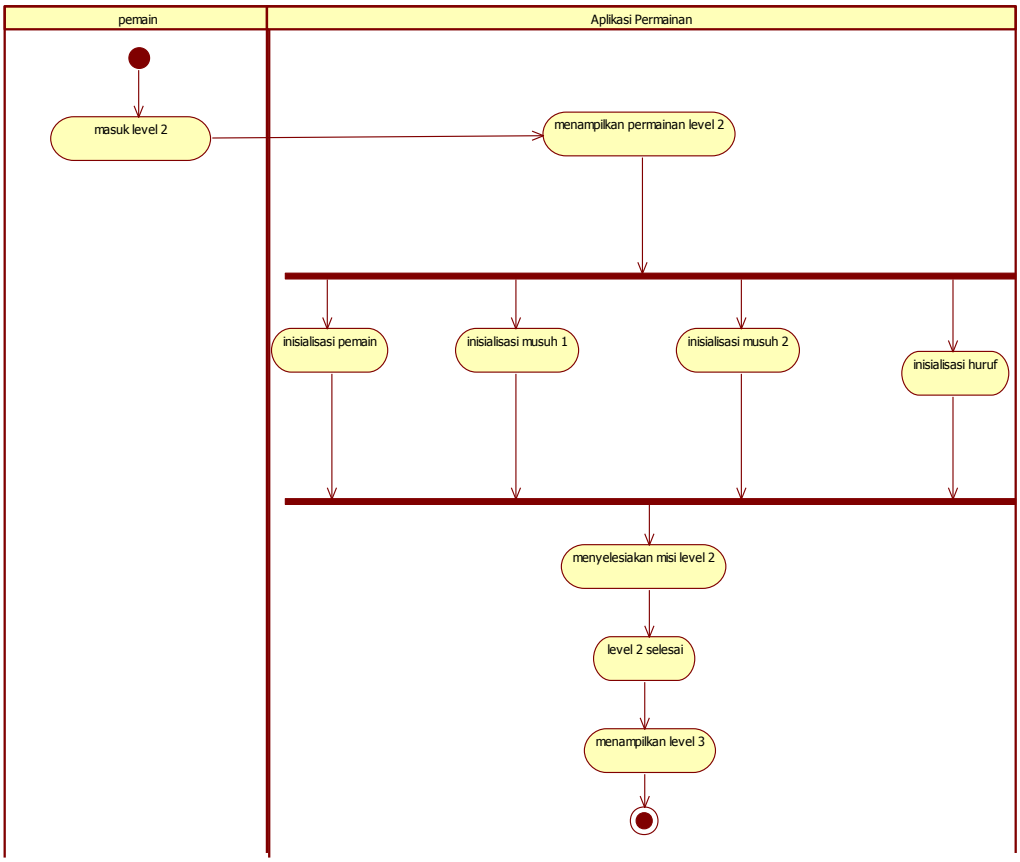
Proses ini terjadi saat permainan di jalankan. Sistem akan menampilkan arena permainan dimana pemain dapat langsung memainkan level 1 dan menyelesaikan misi permainan pada *level* 1, apabila misi tidak terselesaikan maka pemain kembali ke permainan *level* 1 lagi. Diagram *activity* mulai permainan *level* 1 terdapat pada gambar 3.17.



Gambar 3. 17 *Activity Diagram* Level 1

4. *Activity Diagram* Level 2

Proses ini terjadi saat pemain telah menyelesaikan permainan *detective code out* pada *level* 1. Sistem akan menampilkan arena permainan *level* 2 dan pemain harus menyelesaikan misi permainan pada *level* 2, apabila misi permainan tidak terselesaikan maka pemain akan tetap memulai permainan dari *level* 1. Diagram *activity level* 2 terdapat pada gambar 3.18.

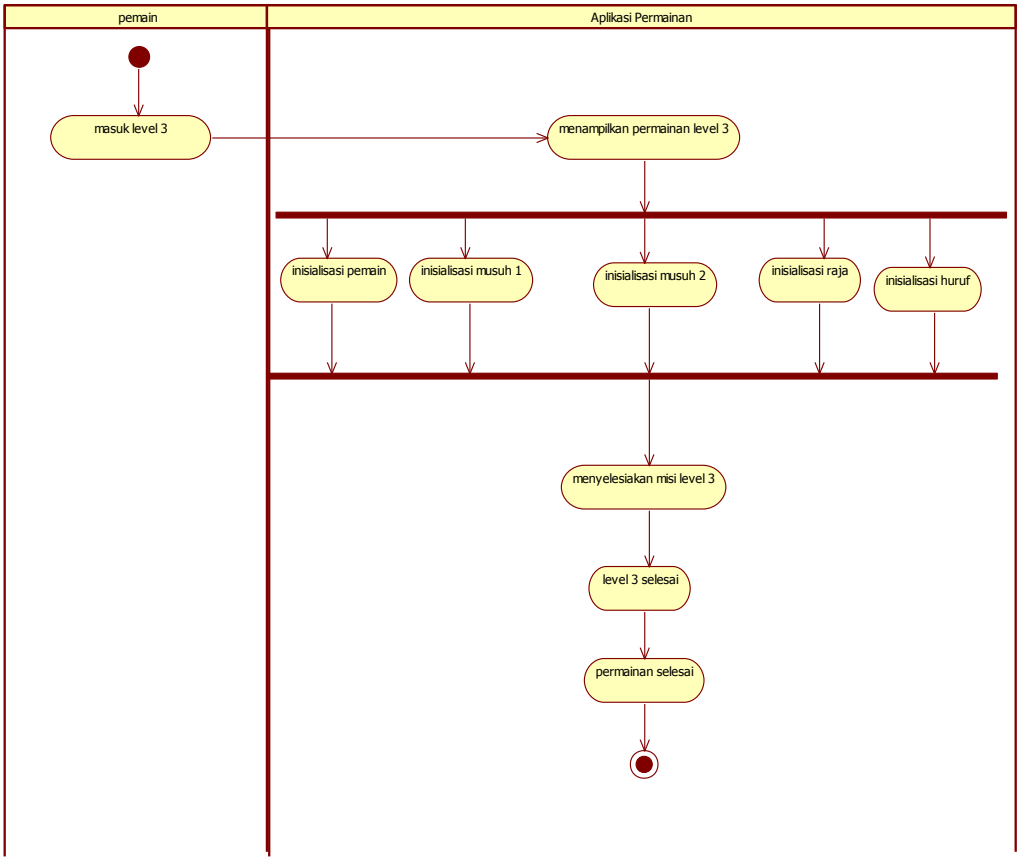


Gambar 3. 18 *Activity Diagram* Level 2

5. *Activity Diagram* Level 3

Proses ini terjadi saat pemain telah menyelesaikan permainan *detective code out* pada *level* 2. Sistem akan menampilkan arena permainan *level* 3 dan pemain harus menyelesaikan misi permainan pada *level* 3, apabila misi permainan

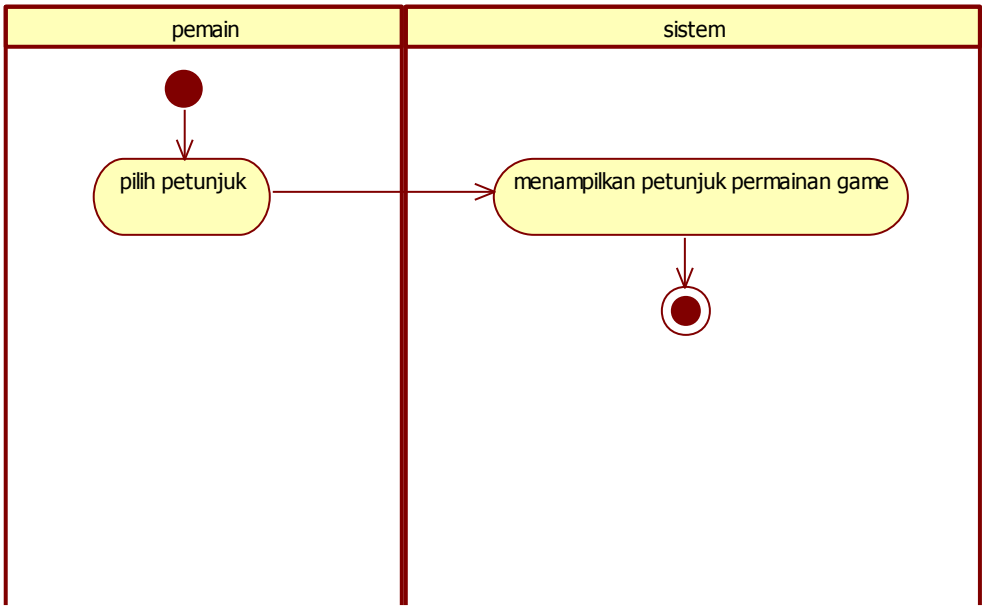
tidak terselesaikan maka pemain akan tetap memulai permainan dari *level* 1. Diagram *activity level* 3 terdapat pada gambar 3.19.



Gambar 3. 19 *Activity Diagram* Level 3

6. *Activity Diagram* Lihat Petunjuk

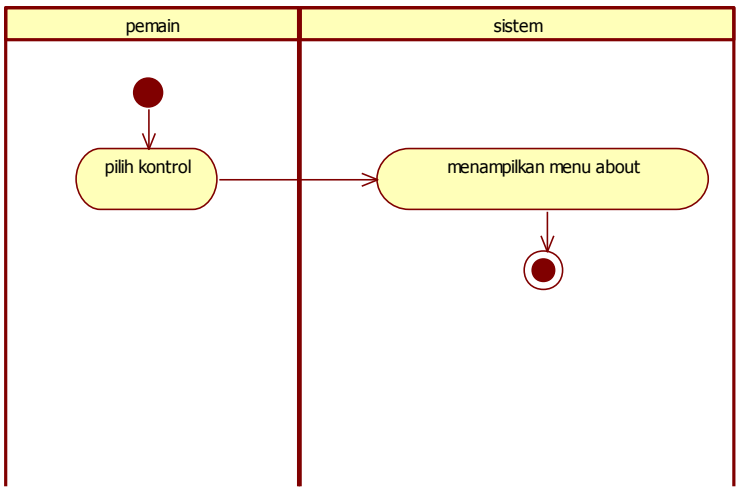
Proses ini terjadi saat pemain memilih menu lihat petunjuk pada halaman utama. Sistem akan menampilkan layar lihat petunjuk untuk memberikan informasi kepada pemain tentang petunjuk memainkan *game*. *Activity diagram* dapat dilihat pada gambar 3.20.



Gambar 3. 20 *Activity Diagram* Lihat Petunjuk

7. *Activity Diagram* Lihat *About*

Proses ini terjadi saat pemain memilih menu *about* pada halaman utama. Sistem akan menampilkan layar lihat *about* untuk memberikan informasi kepada pemain tentang pembuat dari aplikasi *Detective code out. Activity diagram about* dapat dilihat di gambar 3.21.

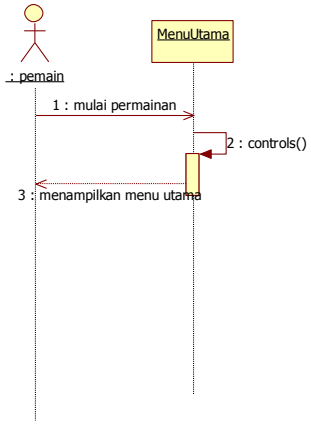


Gambar 3. 21 *Activity Diagram* Lihat About

3.1.5.4 Sequence Diagram

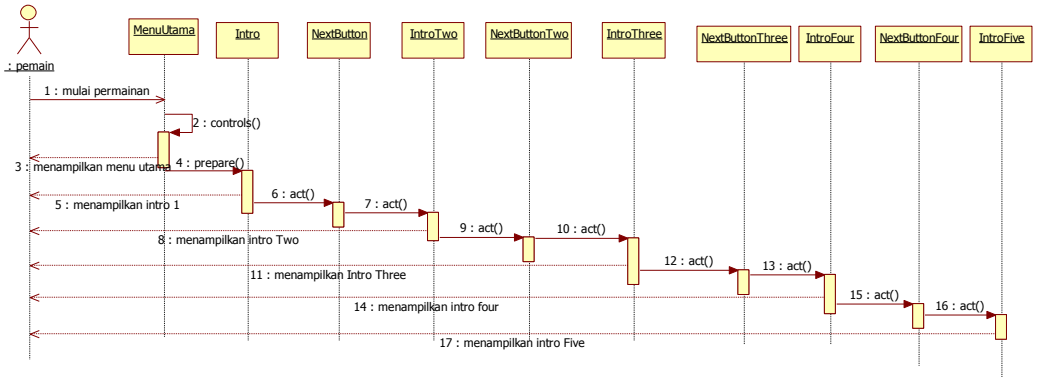
Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup dan *message* yang dikirimkan dan diterima antarobjek.

1. ***Sequence Diagram* Mulai Permainan Baru**



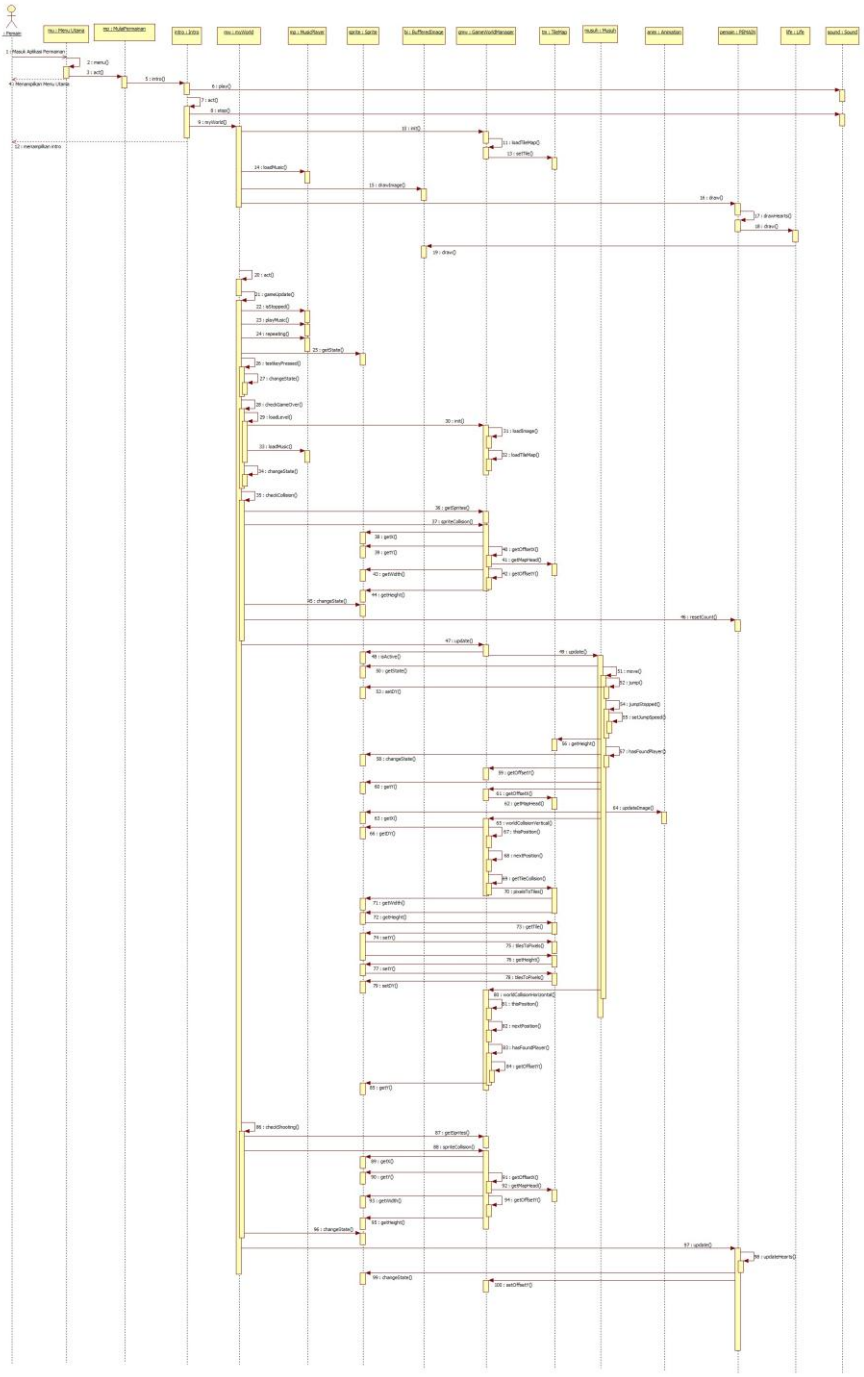
Gambar 3. 22 *Sequence Diagram* Mulai permainan Baru

2. *Sequence Diagram* Intro



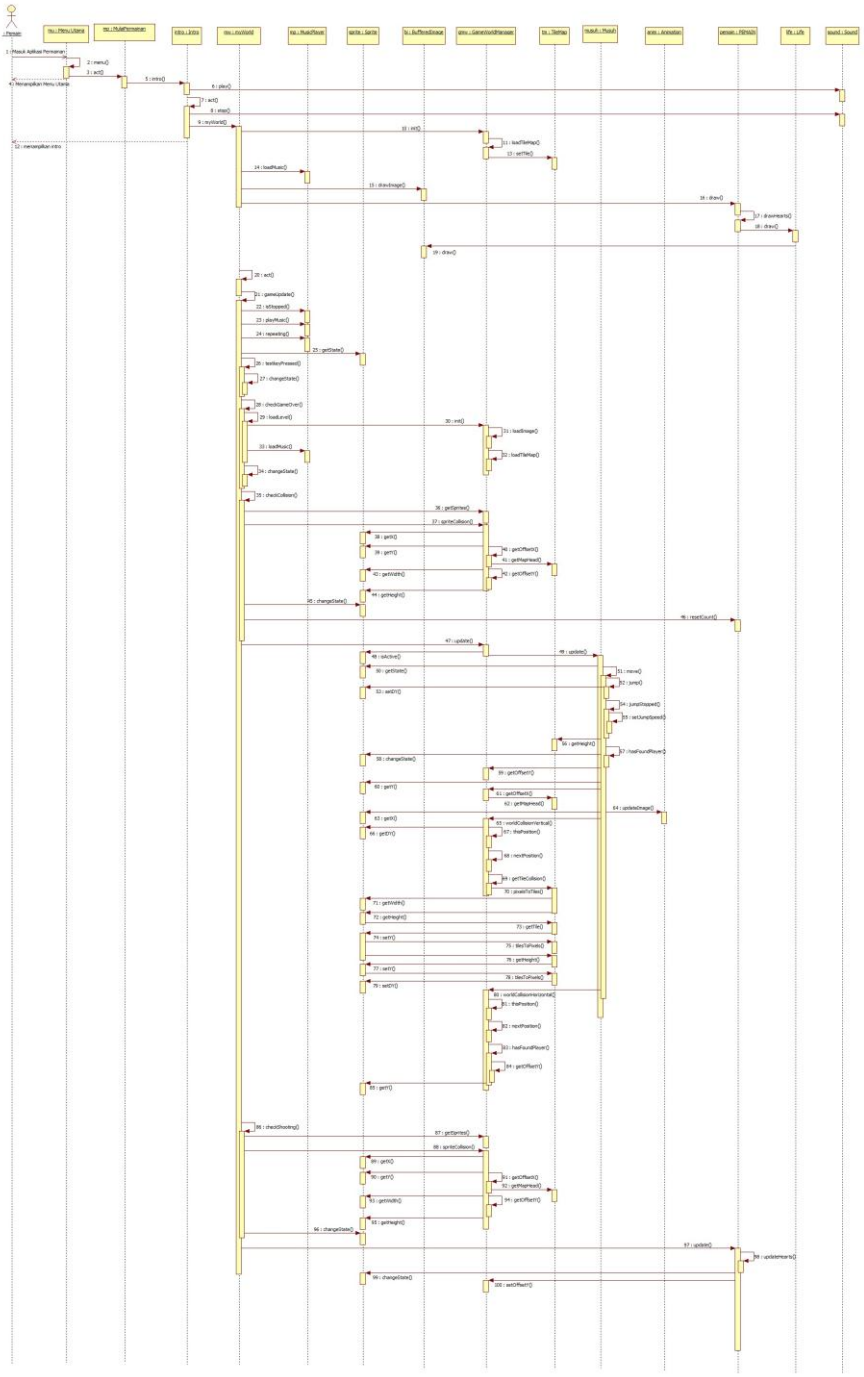
Gambar 3. 23 *Sequence Diagram* Intro

3. *Sequence Diagram* Level 1



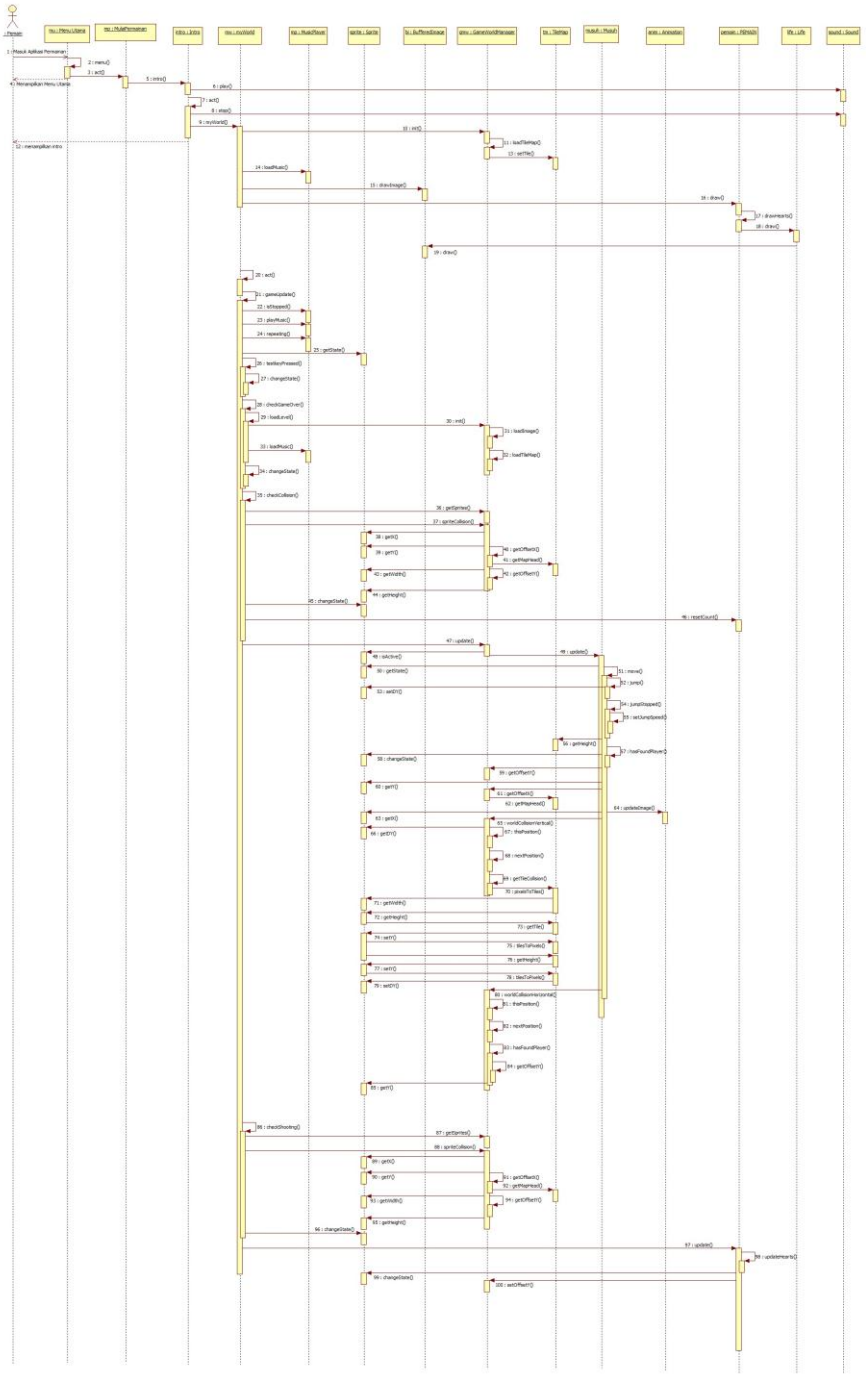
Gambar 3. 24 *Sequence Diagram* Level 1

4. *Sequence Diagram* Level 2



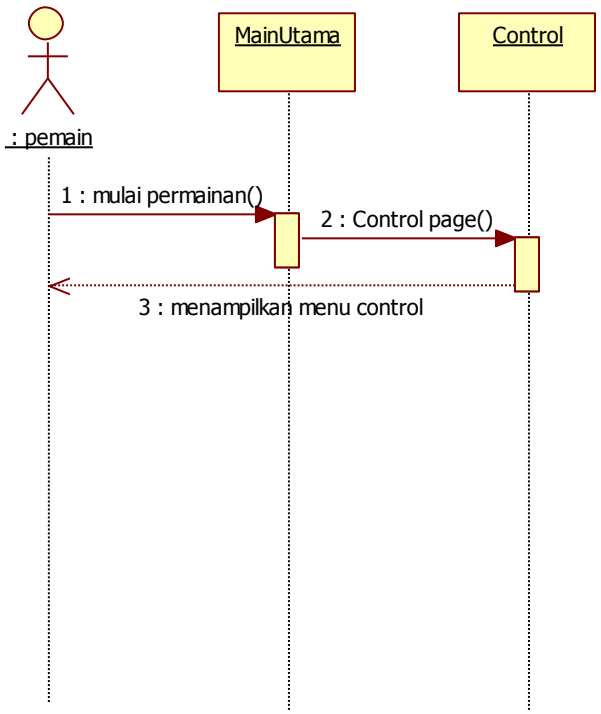
Gambar 3. 25 *Sequence Diagram* Level 2

5. *Sequence Diagram* Level 3



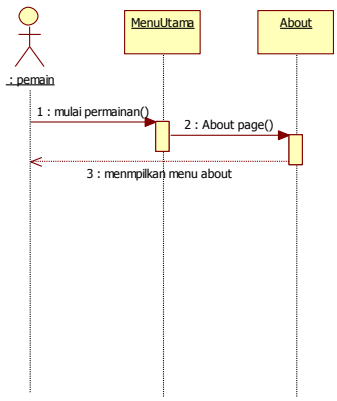
Gambar 3. 26 *Sequence Diagram* Level 3

6. *Sequence Diagram* Lihat Petunjuk



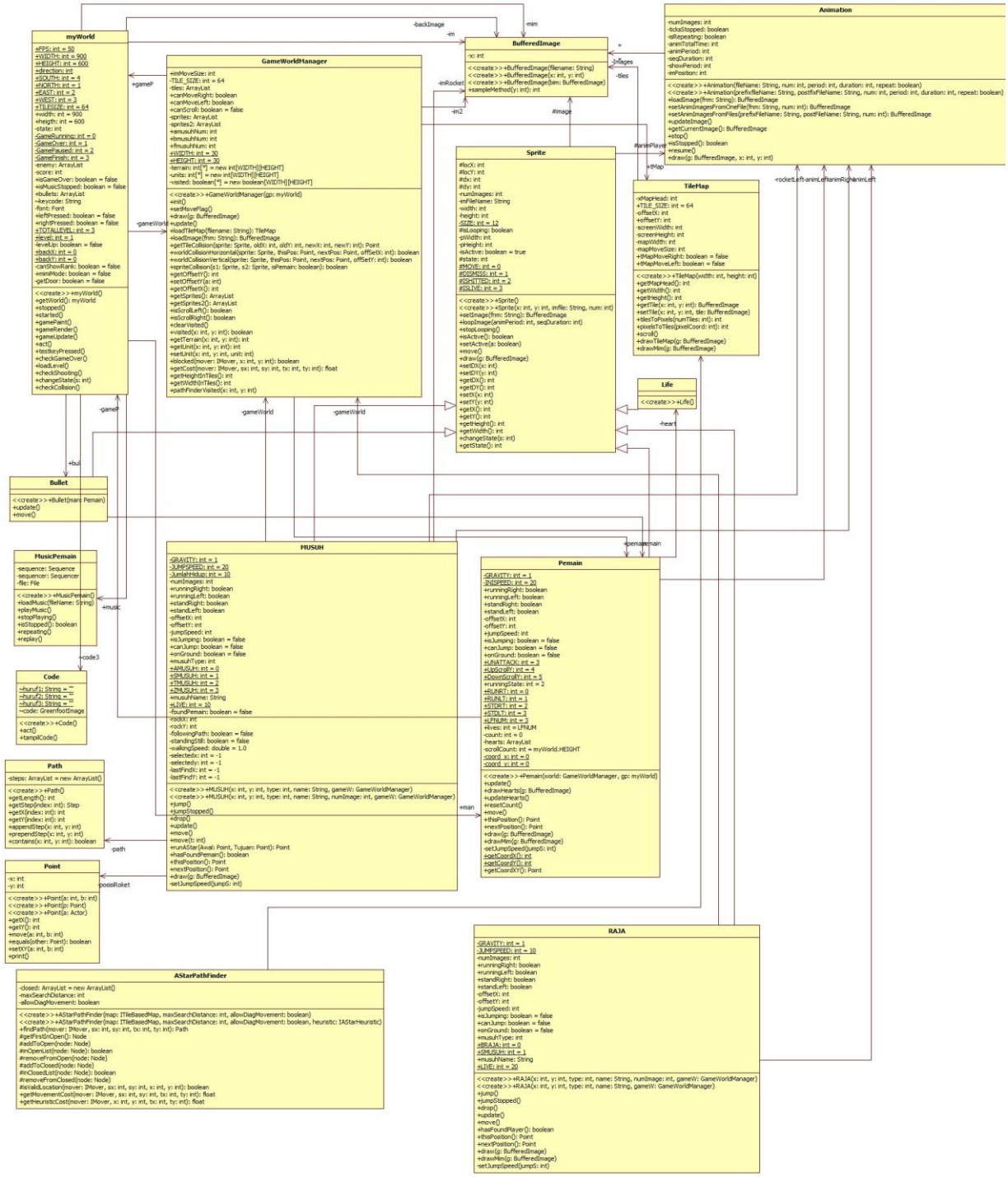
Gambar 3. 27 *Sequence Diagram* Lihat Petunjuk

7. *Sequence Diagram* Lihat *About*



Gambar 3. 28 *Sequence Diagram* Lihat *About*

3.1.5.5 *Class Diagram*



Gambar 3. 29 *Class Diagram*

3.2 Perancangan Sistem

3.2.1 Perancangan Komponen Permainan

3.2.1.1 Karakter

Karakter atau entitas dalam *game Detective Code Out* terbagi menjadi dua, yaitu karakter pemain dan musuh. Karakter dan item yang terdapat didalam game *Detective Code Out* ditunjukan pada tabel 3.22.

Tabel 3. 22 Karakter Pada Pemainan *Detective Code Out*

NO.	NAMA	GAMBAR	PENJELASAN
1.	Brian		Brian adalah karakter player utama pada game, memiliki kemampuan untuk melompat, menghindar, dan juga bertarung dengan musuh
2.	Segerombolan Jack		Jack merupakan musuh pada level 1, jack memiliki kemampuan untuk mengejar Brian jika berada pada radius jarak penglihatannya, melompat dan bertarung dengan Brian
3.	Segerombolan Billy		Billy merupakan musuh pada level 2, Billy memiliki kemampuan untuk mengejar Brian jika berada pada radius penglihatannya, melompat dan bertarung dengan Brian
4.	Moggy		Moggi merupakan musuh di level 3. Moggi

			merupakan bos dari jack dan Billy.

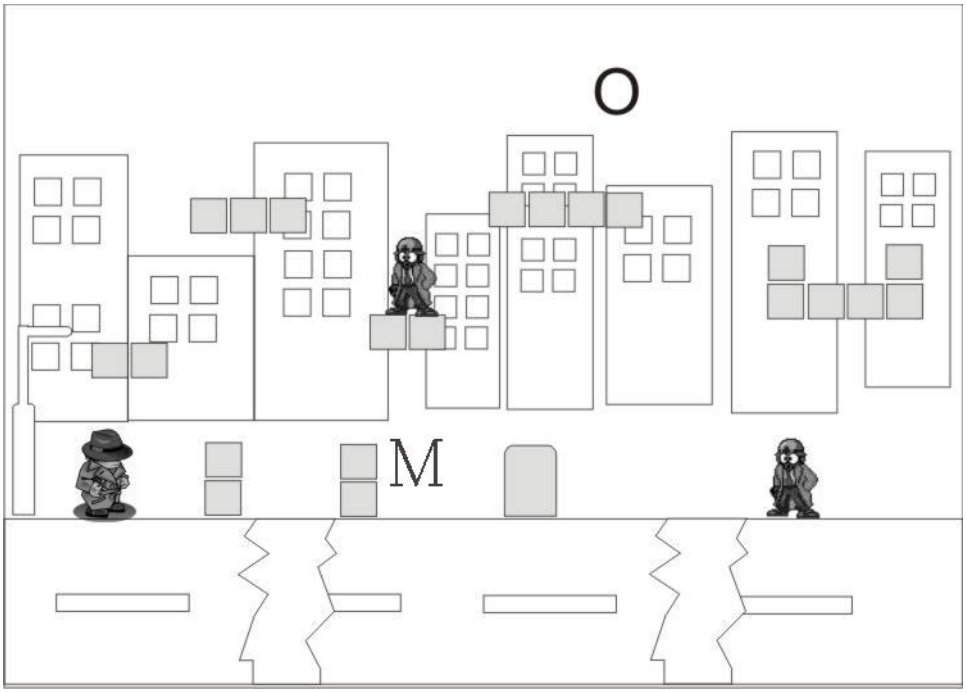
3.2.1.2 *Storyboard*

Storyboard merupakan sketsa gambar yang disusun berurutan sesuai dengan naskah yang ada, dengan *storyboard* kita dapat menyampaikan ide cerita kita kepada seorang lain dengan mudah.

1. ***Storyboard* level 1**

Pada level 1 ini, pemain harus mengalahkan segerombolan *Jack*. Misi utama selain mengalahkan segerombolan *jack*, pemain harus mengumpulkan huruf-huruf yang tersebar di sepanjang perjalanan. Huruf yang terkumpul adalah kode untuk mengungkap misteri di level 3.

Berikut adalah gambar *storyboa*rd level 1 permainan *detective code out* terdapat pada gambar 3.30.

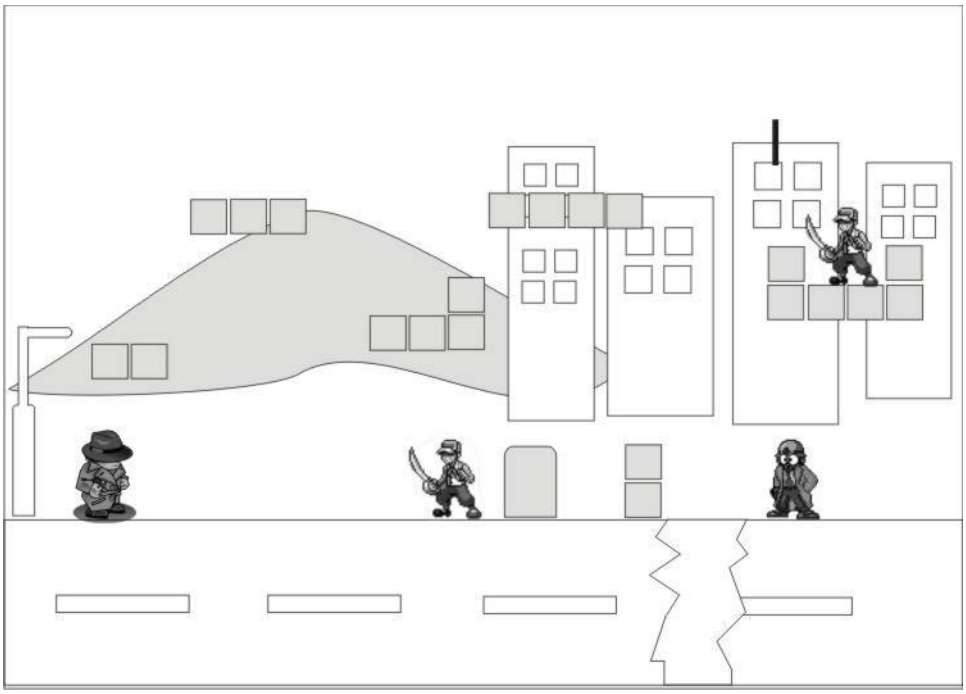


Gambar 3. 30 *Storyboard* level 1

***2. Storyboard* Level 2**

Pada level 1 ini, pemain harus mengalahkan segerombolan Jack, dan segerombolan Billy. Misi utama selain mengalahkan segerombolan jack dan segerombolan billy adalah pemain harus mengumpulkan huruf-huruf yang tersebar di sepanjang perjalanan. Huruf yang terkumpul adalah kode untuk mengungkap misteri di level 3.

Berikut adalah gambar *storyboard* level 2 permainan *detective code out* terdapat pada gamabar 3.31.

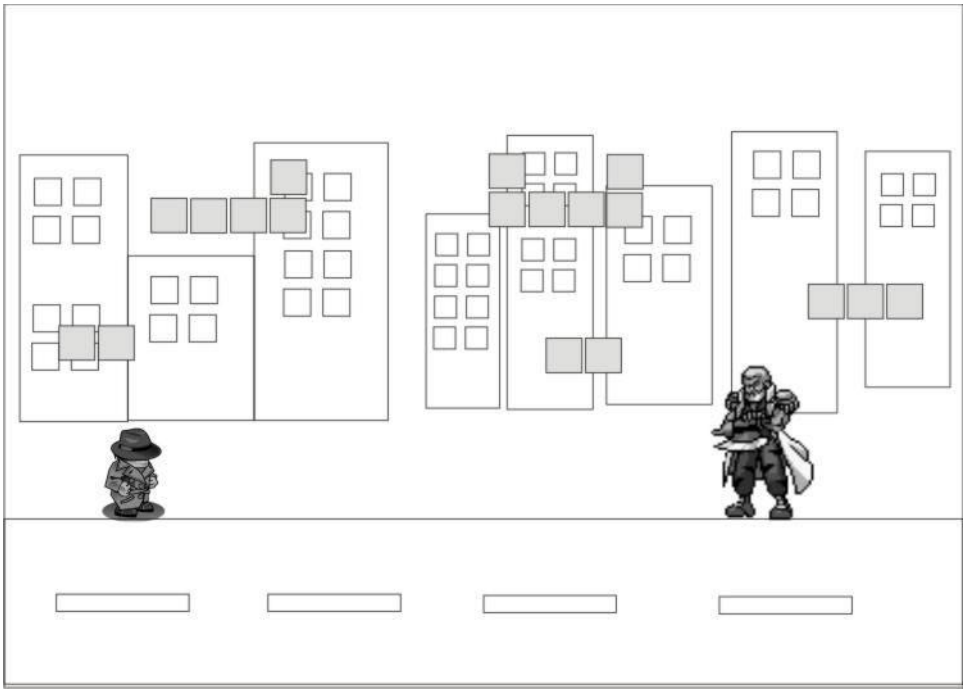


Gambar 3. 31 Storyboard level 2

3. Storyboard level 3

Setelah berhasil melewati level 1 dan level 2, maka pemain akan melawan Moggy yaitu seorang bos dan tersangka dari kasus pembunuhan ini. Di level ini pemain akan bertarung melawan boss dengan dibantu anak buahnya.apabila pemain dapat mengalahkan boss maka permainan berakhir dan pemain sebagai pemenang.

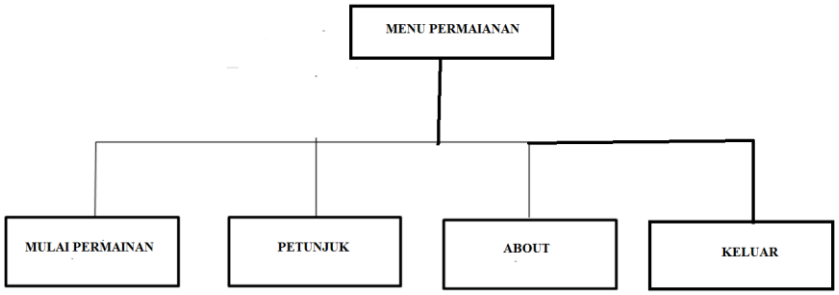
Berikut adalah gambar storyboard level 2 permainan *detective code out* terdapat pada gambar 3.32.



Gambar 3. 32 *Storyboard* level 3

3.2.2 Perancangan Struktur Menu

Untuk mempermudah dalam pembuatan sistem, diperlukan perancangan struktur menu program yang akan dibangun. Perancangan struktur menu program ini membantu dalam merancang bagian-bagian dari sistem yang sebenarnya dan untuk mengetahui bagian mana yang nantinya akan diakses terlebih dahulu setelah program selesai dibuat. Perancangan arsitektur menu pada aplikasi permainan *Detective Code Out* dapat dilihat pada gambar 3.33.



Gambar 3. 33 Perancangan Struktur Menu

A. Mulai Permainan : Menu untuk memulai *game detective code out.*

B. Petunjuk : Menu untuk menampilkan informasi tentang cara memainkan *game detective code out.*

C. *About* : Menu untuk menamplkan informasi about game *detective code out.*

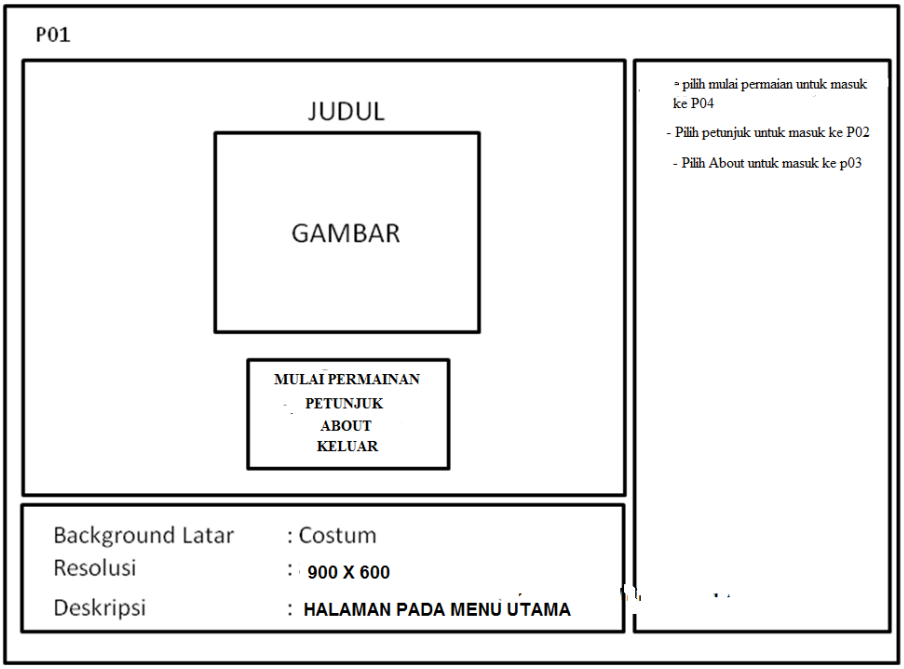
D. Keluar : Menu untuk keluar dari *game detective code out.*

3.2.3 Perancangan Antarmuka

Perancangan antarmuka yang telah dibuat meliputi beberapa perancangan diantaranya perancangan antar muka menu utama , perancangan antarmuka menu utama, perancangan antarmuka mulai permainan, perancangan antar muka petunjuk, perancangan antar muka about.

1. Perancangan Antarmuka menu Utama

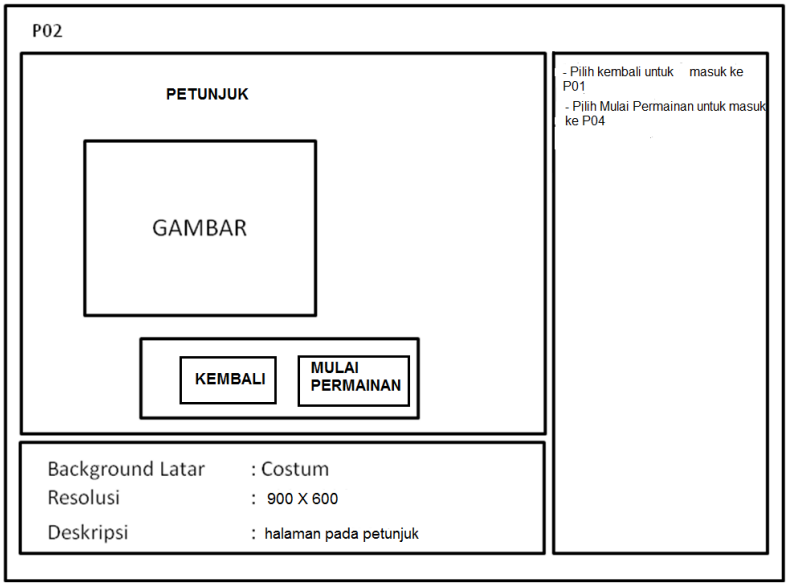
Perancangan antarmuka menu utama pada aplikasi permainan *detective code out* dapat dilihat pada gambar 3.34.



Gambar 3. 34 Perancangan Antar Muka menu Utama

2. Perancangan Antarmuka Petunjuk

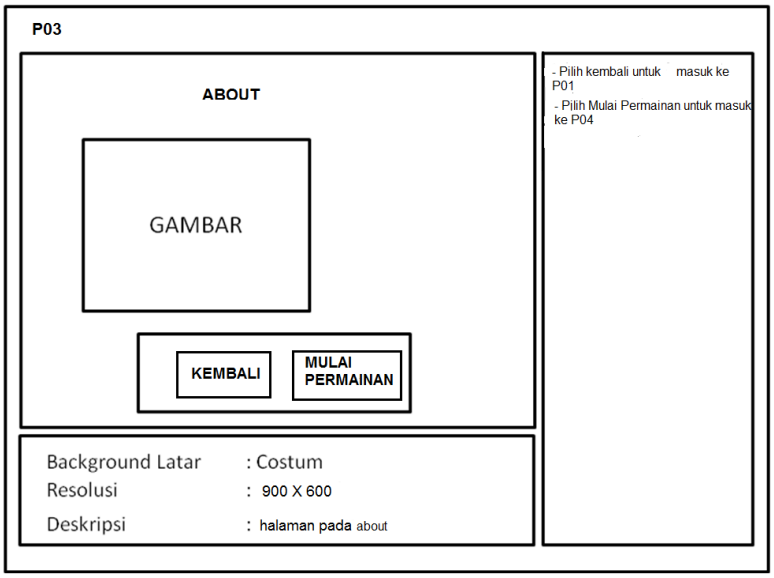
Perancangan antarmuka petunjuk pada aplikasi permainan *detective code out* dapat dilihat pada gambar 3.35.



Gambar 3. 35 Perancangan Antar Muka Petunjuk

3. Perancangan Antarmuka About

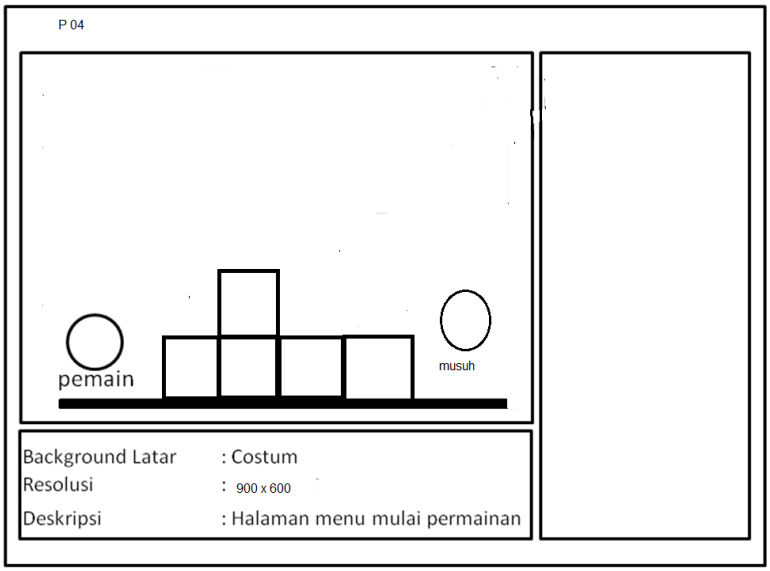
Perancangan antarmuka about pada aplikasi permainan *detective code out* dapat dilihat pada gambar 3.36.



Gambar 3. 36 Perancangan Antarmuka *About*

4. Perancangan Antar Muka Mulai Permainan

Perancangan antarmuka mulai permainan pada aplikasi permainan *detective code out* dapat dilihat pada gambar 3.34.

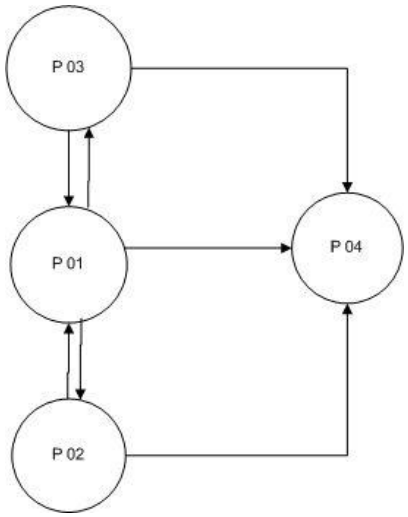


Gambar 3. 37 Perancangan Antarmuka Mulai Permainan

3.2.4 Jaringan Semantik

Jaringan semantik adalah menggambarkan hubungan antar menu pada aplikasi *game Detective code out.*

Berikut adalah jaringan semantik *game detective code out* terdapat pada gambar.3.38.



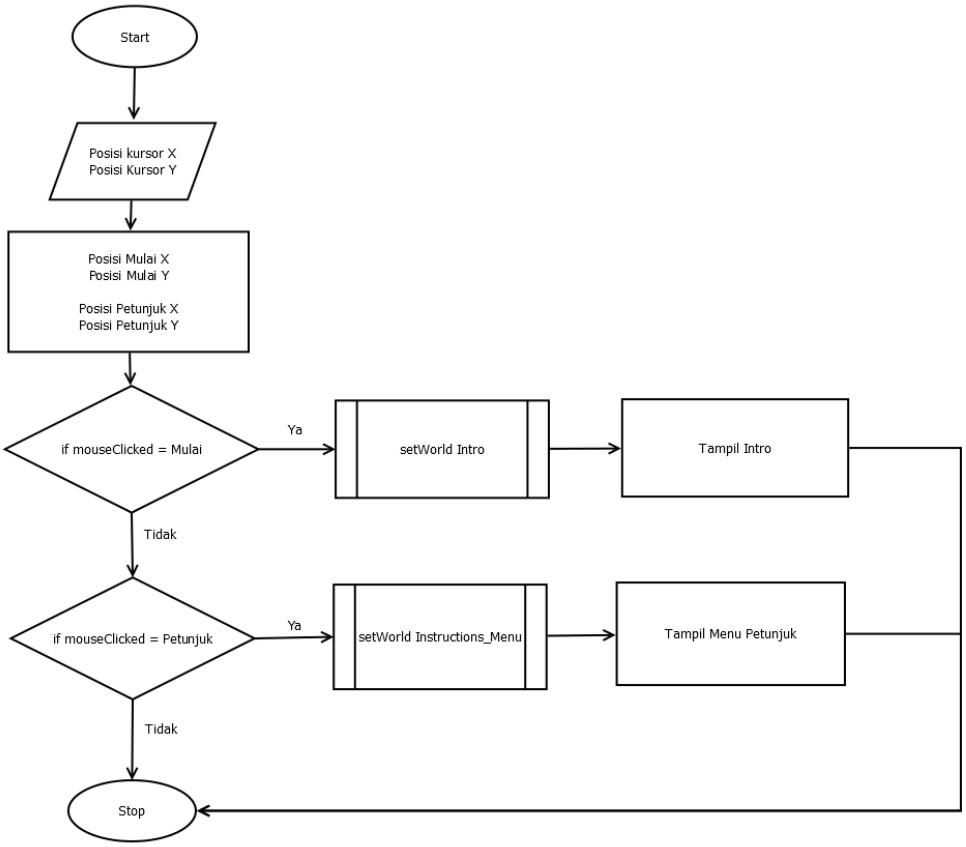
Gambar 3. 38 jaringan Semantik

3.2.5 Perancangan *Method*

Perancangan komponen *method* merupakan perancangan yang dibuat setelah perancangan struktur menu, antarmuka dan jaringan semantik. Perancangan ini berfungsi untuk mendeskripsikan method-method yang berada di dalam aplikasi permainan *Detective Code out*. Adapun method-method yang terdapat dalam aplikasi permainan *Detective code out* yang akan dibangun adalah sebagai berikut :

A. *Method* Menu

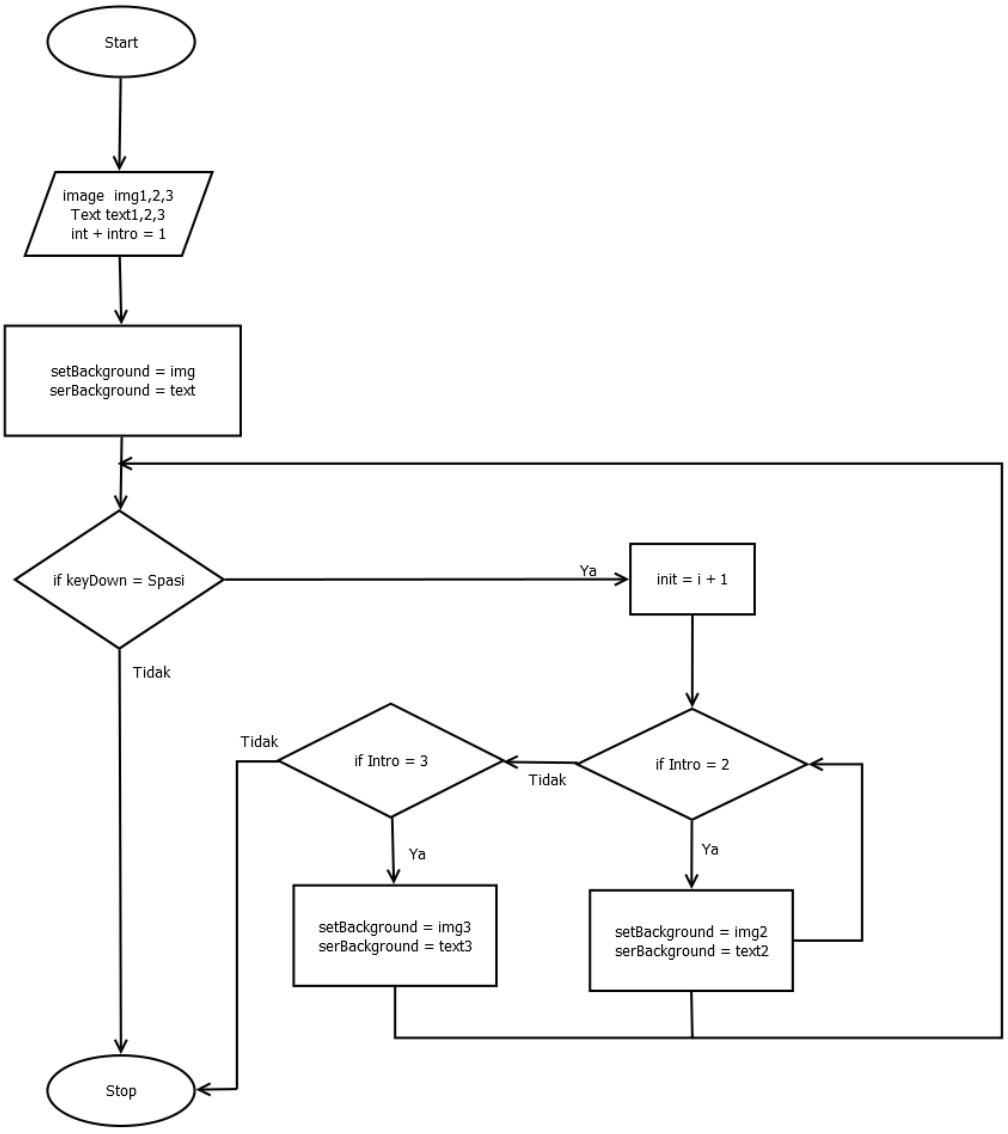
Method menu() akan dipanggil ketika pertamakali memulai permainan, dapat dilihat pada gambar 3.39.



Gambar 3. 39 *Method* Menu

B. *Method* intro()

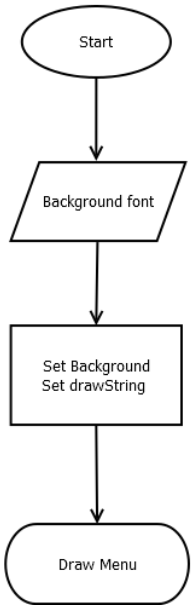
Method intro akan dipanggil ketika pemain memasuki menu utama dan memilih mulai permainan, dapat dilihat pada gambar 3.40.



Gambar 3. 40 *Method intro*

C. Method prepare()

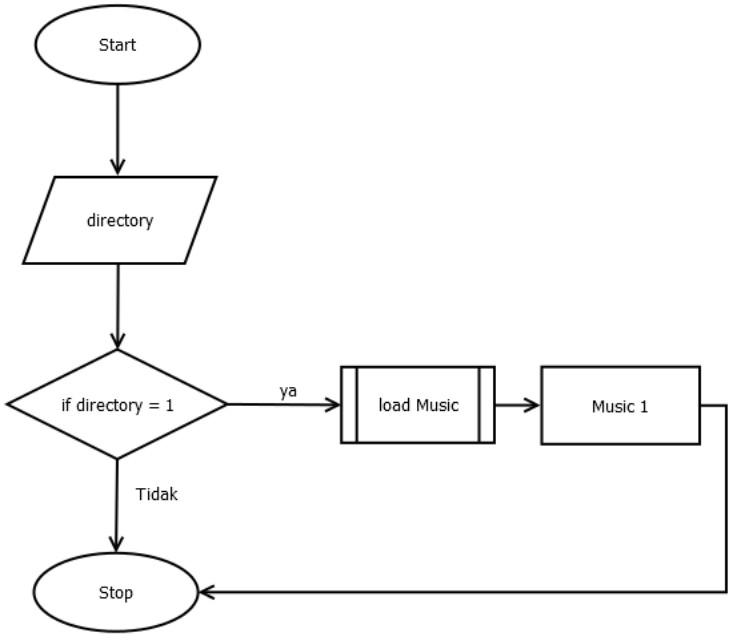
Method prepare akan dipanggil ketika pemain memasuki menu utama mulai permainan, dapat dilihat pada gambar 3.41.



Gambar 3. 41 *Method prepare*

D. Method loadMusic()

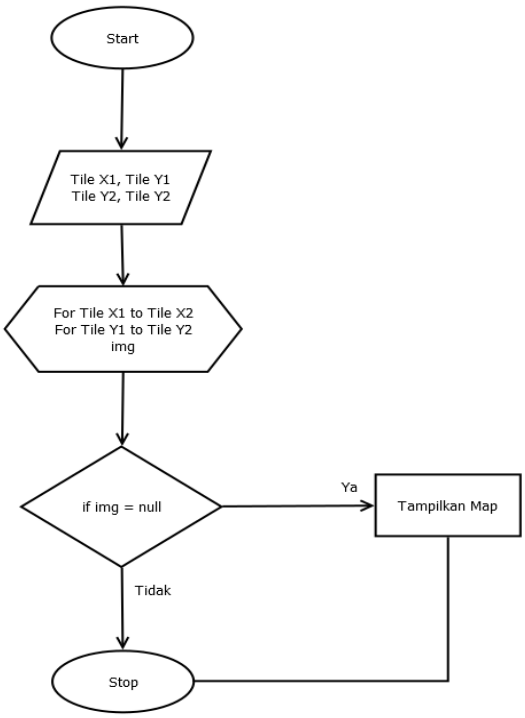
Method loadMusic akan dipanggil ketika pemain memulai permainan, dapat dilihat pada gambar 3.42.



Gambar 3. 42 *Method loadMusic*

E. Method draw Map()

Method drawMap akan dipanggil di ketika pemain memasuki arena permainan, dapat dilihat pada gambar 3.43.



Gambar 3. 43 Method draw Map