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.

3.2. Analisis Masalah

Berdasarkan analisis yang dilakukan terhadap *game* sejenis yang berjudul "*TILEGAMEPlatformer*" dapat disimpulkan bahwa dalam *game* tersebut, dalam hal cara bermain masih terlalu monoton, hal ini dibuktikan pada alur permainannya yang masih sederhana, mulai dari tingkatan kesulitan yang hanya dibedakan dalam bentuk gelombang dan misi dari permainan ini yang masih belum jelas akhirnya.

3.2.1. Analisis Aplikasi Permainan Sejenis

Dalam hal ini diperlukan sebuah analisis yang dilakukan terhadap aplikasi permainan lain yang sejenis untuk menentukan dimana letak kekurangan aplikasi permainan tersebut, sehingga dapat penulis kembangkan sebagai acuan dalam membangun aplikasi permainan baru agar semakin menarik dibandingkan dengan aplikasi permainan yang dianalisis. Aplikasi permainan sejenis yang akan dianalisis adalah aplikasi permainan yang berjudul *TILEGAMEPlatformer*, seperti pada gambar 3.1.



Gambar 3.1. **ScreenshotTILEGAMEPlatformer [18]**

TILEGAMEPlatformer menawarkan sebuah permainan *game* ber-genre *Platformer*.Pada *game* tersebut, musuh terdiri dari satu jenis karakter, musuh sendiri dibedakan dari jenis dan warnanya. Musuh diharuskan untuk terus maju menghalau karakter yang berbentuk bola biru, ketika musuh tersebut bersentuhan pada karakter utama maka secara langsung karakter akan mati dan kembali memulai permainan dari awal.

Pada *game* ini disediakan satu jenis item bintang yang berwarna kuning, karakter utama disarankan untuk mengambil bintang tersebut. Setiap bintang terletak di beberapa platform, untuk memperoleh bintang yang terletak di atas ataupun mengapung, maka karakter harus menginjak setiap musuh yang ada, dan melompat ke atas untuk mengambil bintang tersebut. Kekurangan yang terdapat dalam *game* tesebut yaitu karakteristik terhadap NPC masih bersifat pasif dalam bergerak maupun menyerang dan tidak memiliki info spesifik terhadap *Gameplay*-control.

a. ElemenPermainan

Elemen permainan yang terdapat pada *TILEGAMEPlatformer* diantaranya adalah sebagai berikut:

1. Arena

   Arena permainan yang terdapat pada *TILEGAME* adalah sebuah tempat dengan background langit yang luas dan terdapat beberapa rintangan. Rintangan yang ditemukan adalah beberapa musuh yang menghadang pemain di beberapa tempat tertentu. Pemain harus melewati rintangan tersebut untuk menyelesaikan pertarungan.

2. Karakter Utama

   Karakter utama adalah entitas utama yang akan dimainkan oleh pemain di dalam arena permainan ini. Pemain berupa karakter berbentuk bola biru, pemain harus mengambil item-item yang berbentuk bintang dan mengalahkan musuh yang menghadang dengan cara menginjaknya.

3. Karakter Musuh

   Karakter musuh yang terdapat pada *TILEGAME* adalah sejenis serangga berupa ulat dan lebah karakter tersebut bergerak dan tidak mengejar pemain. Apabila pemain menyentuh musuh maka pemain akan mati dan mulai permainan dari awal.

4. Item

   Item yang terdapat pada *TILEGAME* adalah berupa bintang berwarna kuning dan satu buah note lagu. Item-item tersebut bisa didapatkan pemain di setiap *level*.

b. Skenario Permainan

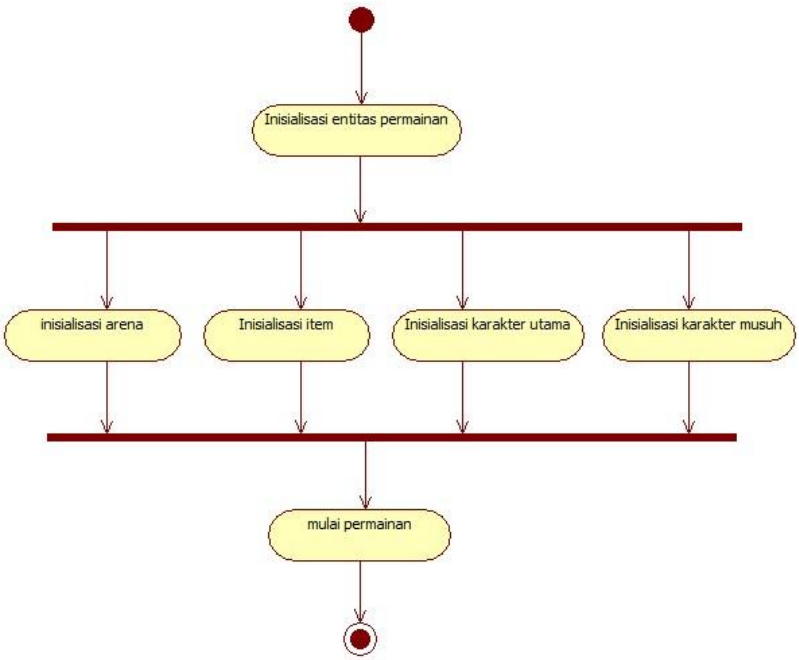
Skenario permainan yang terdapat pada *TILEGAMEPlatformer* diantaranya adalah sebagai berikut:

1. Alur sistem

   Alur sistem pada *TILEGAMEPlatformer* adalah sebagai berikut:

   a. Sistem akan menginisialisasi seluruh entitas permainan yaitu Arena, Karakter utama, Karakter musuh, item.
   b. Posisi musuh ditentukan oleh sistem pada sebuah arena.
   c. Posisi pemain diletakan disebelah kiri arena permainan.
   d. Item bintang bisa di temukan di setiap paltform di arena permainan.
   e. Kontrol yang digunakan oleh pemain mengunakan perintah melalui keyboard.
   f. Update terhadap entitas akan dilakukan oleh sistem.

Adapun alur sistem dari *gameTILEGAMEPlatformer* dapat dilihat dalam gambar 3.2.



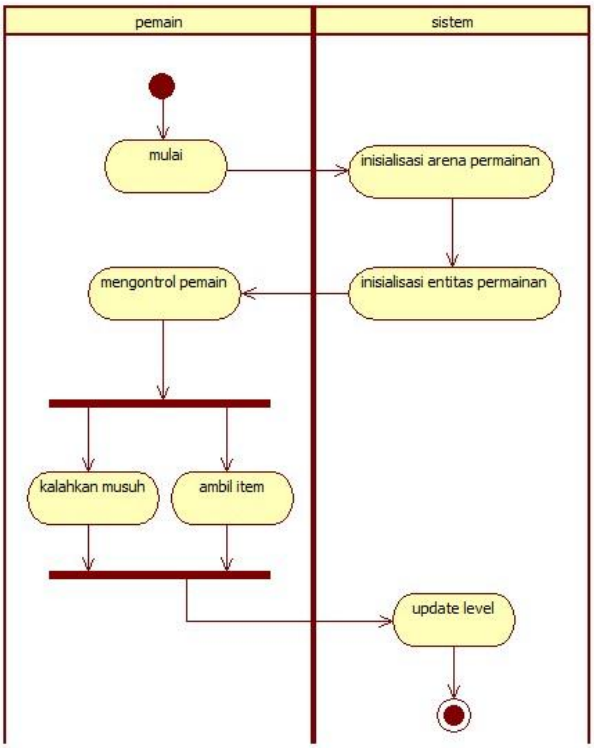
Gambar 3.2. **sistem pada TILEGAME Platformer**

2. Alur Permainan

Alur permainan pada *TILEGAMEPlatformer* adalah sebagai berikut:

a. Pemain harus mengumpulkan semua item dan mengalahkan musuh untuk menyelesaikan permainan.
b. Pemain harus melompat ke platform untuk mengambil item yang berada di luar jangkauan lompat.
c. Pemain mengalahkan musuh dengan cara menginjaknya dari atas.
d. Pemain akan mati apabila terkena musuh.
e. *Level* akan berubah apabila semua musuh telah mati.

Adapun alur permainan dari *gameTILEGAMEPlatformer* dapat dilihat dalam gambar 3.3.



Gambar 3.3. **alur permainan pada *TILEGAMEPlatformer***

3.2.1.2. Pengenalan

The Legend Of Cursed Amulet merupakan nama aplikasi permainan bergenre *Platformer* yang akan dibuat. Kata Legend dalam aplikasi permainan ini berarti legenda atau sebuah kisah, dan kata Cursed Amulet yang berarti sebuah jimat di dalamnya terdapat kutukan yang jahat. Dalam pemberian nama pada aplikasi permainan ini bertujuan selain memberikan hiburan kepada para pemain, juga memberikan sebuah pengajaran bahwa para pemain dapat menerapkan apa yang ada di dalam aplikasi permainan ini ke dalam dunia nyata, yaitu dengan melawan orang-orang yang berusaha menguasai seluruh duniadengan menggunakan jimat yang terkutuk. Aplikasi permainan ini dibangun dalam bentuk dua dimensi (2D), adapun kelebihan-kelebihan yang terdapat di dalam aplikasi permainan *The Legend Of Cursed Amulet* terdapat pada tabel 3.1.

Tabel 3.1. Kelebihan Aplikasi Permainan

Kelebihan Aplikasi Permainan *The Legend Of Cursed Amulet*
1. Terdapat bermacam-macam musuh dan item 2. Penerapan konsep AI pada musuh di aplikasi permainan tersebut pada tingkatan tertentu 3. Terdapat tambahan health jika hero yang digunakan oleh pemain terkena serangan oleh musuh, maka hero dapat mencarisebuah itemhealth yang berada pada tempat-tempat tertentu

3.2.1.3. Story LineGame

GameThe Legend Of Cursed Amulet bercerita tentang seorang ninja terpilih yang diutus untuk menjalankan misi merebut kembali sebuah kristal jimat terkutuk yang telah dicuri dari suatu desa yang bernama Hidden Forest. Jimat tersebut dicuri oleh seorang assassin jahat yang berambisi untuk menguasai seluruh dunia.Konon jimat tersebut memiliki kekuatan yang sangat mengerikan apabila jatuh ke tangan yang salah karena jimat tersebut bisa membangkitkan monster-monster yang bisa menyerang siapa saja yang ada dihadapannya.
Alur dan misi dari aplikasi permainan *The Legend Of Cursed Amulet*, yaitu :

A. Alur Aplikasi Permainan *The Legend Of Cursed Amulet*

Alur permainan merupakan ketentuan-ketentuan yang terdapat pada sebuah aplikasi permainan agar setiap pemain dapat mengetahui bagaimana cara untuk memainkan permainan ini dan untuk menyelesaikan misi pada aplikasi permaianan tersebut.

Alur permainan dari aplikasi permainan *The Legend Of Cursed Amulet* terdapat pada tabel 3.2.

Tabel 3.2. Alur Aplikasi Permainan

Alur Permainan
1. Hero bergerak dari awal pltaform bergerak dan melompat menuju platformtertentu sesuai dengan jalur yang telah disediakan mengalahkan musuh yang akan menghadang dan mengambil semua item yang tersedia pada *level* tersebut 2. Pada *level* tertentu musuh bergerak dengan menentukan rute terpendek untuk mencapai tujuannya 3. Musuh yang berada pada jarak jangkauan otomatis akan mengejarhero yang berada di dekatnya 4. Jika hero berhasil membunuh musuh, maka poin permainan akan bertambah dan musuh akan hilang 5. Jika musuh menyerang hero secara langsung, maka health pemain akan berkurang sesuai dengan serangan yang diterima 6. Jika health pemain menjadi 0 (nol) maka permainan dinyatakan berakhir. 7. Jika musuh berhasil dikalahkan sesuai dengan banyaknya musuh yang terdapat pada *level* tersebut sebelum health pemain menjadi 0 (nol) dan semua item amulettelah didapatkan maka permainan dapat dilanjutkan ke *level* berikutnya

B. Misi Aplikasi Permainan *The Legend Of Cursed Amulet*

Misi yang harus diselesaikan pada aplikasi permainan *The Legend Of Cursed Amulet* ini yaitu dengan melakukan petualangan dan mengalahkan monster-monster yang bangkit akibat dari kutukan yang terdapat pada jimat tersebut, dan di *level* terakhir yaitu mengalahkan boss yang telah mengambil jimat tersebut serta merebut kembali jimat tersebut.

3.2.1.4. Analisis Tingkat Kesulitan

Analisis tingkat kesulitan pada *gameThe Legend Of Cursed Amulet* terdapat pada tabel 3.3.

Tabel 3.3. Analisis Tingkat Kesulitan Permainan

***Level* 1**	Misi di *level* ini adalah mengalahkan semua monster dan mencari item-item yang terdapat di suatu tempat. Apabila item-item telah di ambil dan semua musuh sudah di kalahkan maka pemain bisa menuju ke *level* selanjutnya. Musuh akan terus mengejar pemain apabila pemain berada dalam jangkauan musuh dan akan mengurangi life point jika pemain terkena musuh, dan jika pemain mati maka akan mengulang di *level* tersebut.
***Level* 2**	Pada *level* ini misi yang di lakukan sama dengan misi sebelumnya yaitu mengalahkan musuh dan mengambil semua item-item dalam permainan, namun pemain harus melalui rintangan yang berbeda dan mengalahkan musuh yang lebih kuat dan lebih banyak dibandingkan *level* sebelumnya, maka tantangan yang diberikan lebih sulit dari *level* sebelumnya. Musuh yang terdapat di *level* ke-dua adalah monster dengan kemampuan bergerak lebih cepat dibandingkan dengan *level* sebelumnya.
***Level* 3**	*Level* ini adalah *level* terakhir, di misi ini pemain harus mengalahkan bos assassin untuk bisa memenangkan *game* ini dan merebut kembali kristal jimat.

3.2.1.5. Gameplay

Misi yang harus dilakukan oleh pemain di *game*The Legend of Cursed Amule terdapat pada tabel 3.4.

Tabel 3.4. Analisis *Gameplay* Permainan

A. *Level* 1	1. Pemain harus mengalahkan musuh dengan cara menembak dengan menekan tombol ”B” 2. Karakter pemain mempunyai 3 buah health poin 3. Jika health poin habis maka permainan selesai dan akan mengulang permainan pada *level* tersebut 4. Health musuh pada *level* ini adalah 2 5. Terdapat beberapa item pada *level* ini yang membantu pemain untuk melewati rintangan 6. Jumlah musuh yang tersedia adalah 15 buah 7. Kalahkan semua musuh dan ambil 3 kristal untuk menuju ke-*level* berikutnya.
B. *Level* 2	1. Pemain harus melompat ke platform atas untuk melewati tembok yang tinggi dan mengalahkan musuh yang akan menghadang di beberapa platform. 2. Karakter pemain mempunyai 3 buah health poin 3. Jika health poin habis maka permainan selesai dan akan mengulang permainan pada *level* tersebut 4. Health musuh pada *level* ini adalah 3 5. Musuh pada *level* ini bergerak lebih cepat dibanding *level* sebelumnya 6. Terdapat beberapa item pada *level* ini yang membantu pemain untuk melewati rintangan 7. Jumlah musuh yang tersedia adalah 20 buah Kalahkan semua musuh dan ambil 3 kristal untuk menuju ke-*level* berikutnya.
C. *Level* 3	1. Pemain harus mengalahkan musuh terakhir yaitu Boss 2. Health pada boss adalah 20 3. Terdapat beberapa item pada *level* ini yang membantu pemain untuk melewati rintangan 4. Ambil 3 kristal untuk menyelesaikan permainan

3.2.1.6. Scoring

Pada *gameThe Legend Of Cursed Amulet* terdapat poin terhadap setiap musuh yang dikalahkan, poin yang diberikan terhadap musuh adalah 50 poin, poin akan di akumulasikan diakhir permainan setelah menyelesaikan semua misi.

3.2.2. Analisis Algoritma

Pada permainan *The Legend Of Cursed Amulet* ini, diterapkan metode dan algoritma yang akan berguna untuk membuat permainan ini menjadi lebih menarik dan menantang

3.2.2.1. Penerapan Metode Collision detectionThe Legend Of Cursed Amulet

Agar permainan *The Legend Of Cursed Amulet* lebih menarik dan menantang, maka akan diterapkan metode *Collision detection*. Metode ini berfungsi untuk penerapan terhadap gambar atau sprite yang bersinggungan dengan objek secara langsung, baik itu sprite dengan sprite maupun dengan objek yang lain.

Pada permainan The Legend Of Cursed metode *Collision detection* akan diterapkan pada peluru yang ditembakan oleh hero untuk membunuh setiap monster dengan metode *Collision* antara objek dengan objek lain yang akan terjadi apabila objek tersebut saling bersentuhan pada titik tertentu.

Tabel source code metode *collision* peluru dengan musuh ditunjukkan pada Tabel3.5.

```
1   Bullet bul = (Bullet) bullets.get(i);
2       for (int j = 0; j < gameWorld.getSprites().size(); j++)
        {
3           NPC level_1 = (NPC) gameWorld.getSprites().get(j);
4           if (gameWorld.spriteCollision(bul, level_1, true)
    && (level_1.npcType == NPC.ANPC)
    && (level_1.state != Sprite.ISHITTED) && bul.isActive())
            {
              Greenfoot.playSound("collision.wav");
5             NPC.LIVE--;
6             if(NPC.LIVE == 0)
7             {
                level_1.changeState(Sprite.ISHITTED);
8             }
9             //          gameWorld.anpcNum--;
              bul.changeState(Sprite.DISMISS);
10            score += 50;
11          }
12  }
13
```

Tabel 3.5. Source Code metode collision Peluru dengan Musuh The Legend Of Cursed Amulet

Tabel source code metode *collision* musuh dengan hero ditunjukkan pada Tabel3.6.

```
1       public Point getTileCollision(Sprite sprite, int oldX, int oldY, int
        newX, int newY)
          {
2           Point pointCache = new Point(0, 0);
3           int fromX = Math.min(oldX, newX);
4           int fromY = Math.min(oldY, newY);
5           int toX = Math.max(oldX, newX);
6           int toY = Math.max(oldY, newY);
            // get the tile locations
7           int fromTileX = tMap.pixelsToTiles(fromX);
8           int fromTileY = tMap.pixelsToTiles(fromY);
9           int toTileX = tMap.pixelsToTiles(toX + sprite.getWidth() - 1);
10          int toTileY = tMap.pixelsToTiles(toY + sprite.getHeight() - 1);
            // check each tile for a collision
11          for (int x = fromTileX; x <= toTileX; x++) {
12             for (int y = fromTileY; y <=toTileY; y++) {
13                if (x < 0 || x >= tMap.getWidth() || tMap.getTile(x, y) !=
        null) {
        // collision found, return the tile
14                   pointCache.setLocation(x, y);
15                   return pointCache;
16                }
17             }
18          }
19  }
```

Tabel 3.6. Source Code metode collision Musuh dengan Hero The Legend Of Cursed Amulet

3.2.2.2. Penerapan Algoritma A* (A star) The Legend Of Cursed Amulet

Dalam penerapan metode *Collision* pada aplikasi permainan *The Legend Of Cursed Amulet*, pada *level* tertentu juga diterapkan algoritma A* (*A star*) pada musuh sebagai pathfinding, yaitu untuk mencari rute terpendek pada musuh dimulai dari jarak jangkauan serang musuh menuju karakter hero utama.

A. Analisis Algoritma A* (*A star*) padaaplikasi permainan *The Legend Of Cursed Amulet*

Analisis algoritma A* (*A star*) yang terdapat di aplikasi permainan *The Legend Of Cursed Amulet* ini adalah pencarian jalan terpendek yang digambarkan melalui sebuah tahapan proses dimulai ketika musuh yang terletak pada suatu tempat (point A) menuju tempat dimana hero utama berdiri (point B), Hasil pencarian algoritma tersebut akan menghasilkan jalan terpendek dari suatu titik awal ke titik tujuan.

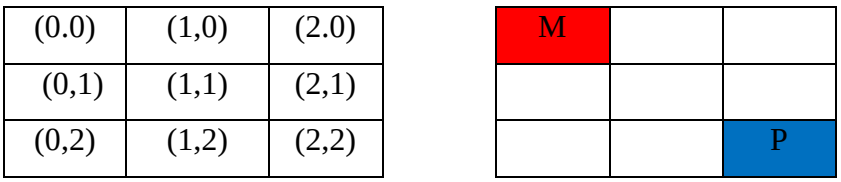
Algoritma A* (*A star*) akan diterapkan pada analisis kasus pencarian jalan terpendek menuju tujuan pada aplikasi permainan Legend Of Cursed Amulet untuk mencari kecepatan, jarak langkah menuju tujuan dan simpul yang diperiksa. Dalam kondisi jarak yang ditempuh sangat jauh dan memiliki banyak rintangan.Setiap pergerakan yang dilakukan pada kotak statusnya akan disimpan pada suatu list. List ini akan digunakan untuk melakukan pengecekan apakah kita sudah pernah membangun status tersebut atau belum agar kita tidak menggerakkan kotak yang sama berkali-kali ke status yang sama. Dengan menerapkan strategi ini, selain menemukan solusi, algoritma ini juga bisa menemukan langkah terpendek untuk mencapai solusi tersebut.

Algoritma A* (*A star*) adalah 49 pengembangan dari BFS yang digunakan untuk mencari jalan terpendek (shortest path) yang sering dipakai dalam *game* programming.Berikut merupakan tahapan proses yang terjadi padaaplikasi permainan *The Legend Of Cursed Amulet* yang menerapkan algoritma A* (*A star*) sebagai pathfinding dari musuh menuju tempat hero berada:

3.2.3. Analisis Metode Pada Aplikasi Permainan *The Legend Of Cursed Amulet*

Analisis metode pada aplikasi permainan *The Legend Of Cursed Amulet* adalah Algoritma A*. Di dalam *game* akan diterapkan pada musuh sebagai penentuan jalur terpendek atau pathfinding menuju target berupa kotak atau ordo X dan Y.

Berikut adalah arena permainan pada aplikasi permainan*The Legend Of Cursed Amulet* ditunjukkan pada gambar 3.4.



Gambar 3.4. **Arena Pada Aplikasi Permainan *The Legend Of Cursed Amulet***

Keterangan : P : pemain
M : musuh

Terdapat beberapa hal yang perlu didefinisikan terlebih dahulu istilah-istilah dalam penerapan algoritma A* (*A star*). Adapun istilah-istilah yang akan dibahas yaitu *OPEN LIST*, *CLOSED LIST*, nilai f, g dan n. Algoritma A* menggunakan dua senarai yaitu OPEN dan CLOSED. OPEN adalah senarai (list) yang digunakan untuk menyimpan simpul-simpul yang pernah dibangkitkan dan nilai heuristiknya telah dihitung tetapi belum terpilih sebagai simpul terbaik (best node) dengan kata lain, OPEN berisi simpul-simpul masih memiliki peluang untuk terpilih sebagai simpul terbaik, sedangkan CLOSED adalah senarai untuk menyimpan simpul-simpul yang sudah pernah dibangkitkan dan sudah pernah terpilih sebagai simpul terbaik. Artinya, CLOSED berisi simpul-simpul yang tidak mungkin terpilih sebagai simpul terbaik (peluang untuk terpilih sudah tertutup).

1. *OPEN LIST*adalah list yang menyimpan kemungkinan path yang akan diperiksa. *OPEN LIST*dibuat terurut berdasarkan nilai f. *OPEN LIST*digunakan untuk menentukan secara selektif (berdasarkan nilai f) jalan yang dikira lebih dekat menuju pada path tujuan. OPEN berisi simpul-simpul yang masih memiliki peluang untuk terpilih sebagai simpul terbaik (best node).
2. CLOSED adalah senarai (list) untuk menyimpan simpul-simpul yang sudah pernah dibangkitkan dan sudah pernah terpilih sebagai simpul terbaik (best node) atau senarai yang menyimpan jalan yang sudah diperiksa dari *OPEN LIST*. Artinya, CLOSED berisi simpul-simpul yang tidak mungkin terpilih sebagai simpul terbaik (peluang untuk terpilih sudah tertutup). Kedua list (*OPEN LIST*dan *CLOSED LIST*) ini bertujuan juga untuk menghindari penelusuran berkali-kali jalan (rute) yang memang sudah diidentifikasi agar tidak masuk kembali ke dalam *OPEN LIST*.
3. Nilai F adalah cost perkiraan suatu path yang teridentifikasi. Nilai F merupakan hasil dari f(n).
4. Nilai G hasil dari fungsi g(n), adalah banyaknya langkah yang diperlukan untuk menuju ke path sekarang.
5. Setiap simpul (node) harus memiliki informasi nilai h(n), yaitu estimasi harga simpul tersebut dihitung dari simpul tujuan yang hasilnya menjadi nilai H

Fungsi f sebagai estimasi fungsi evaluasi terhadap node n, dapat dituliskan :

$$f(n) = g(n) + h(n)$$

dimana,

f(n) : fungsi evaluasi (jumlah g(n) dengan h(n))

g(n) : nilai yang dibutuhkan untuk menempuh jarak dari point awal menuju kotak selanjutnya

h(n) : estimasi nilai yang dibutuhkan untuk menempuh jarak dari kotak selanjutnya untuk menempuh point tujuan.

Pergerakan diagonal pada map diperbolehkan, maka digunakan fungsi heuristic Non-Manhattan Distance. Maka fungsi heuristic yang digunakan adalah sebagai berikut :

h_diagonal(n) = min(abs(n.x-goal.x) + abs(n.y-goal.y))

h_orthogonal(n) = (abs(n.x-goal.x) + abs(n.y-goal.y))

h(n) = h_diagonal(n) + (h_orthogonal (n) – (2 * h_diagonal(n)))

Dengan perhitungan biaya seperti diatas, algoritma A* adalah complete dan optimal. Untuk lebih jelasnya berikut merupakan gambaran dari penggunaan algoritma A* dalam pencarian rute terpendek dari point A menuju point B :

berikut adalah perhitungan yang akan dilakukan dengan algoritma A* dengan kondisi tanpa penghalang yang terlihat pada gambar 3.5.

(0.0)	(1,0)	(2.0)
(0,1)	(1,1)	(2,1)
(0,2)	(1,2)	(2,2)

M		
		P

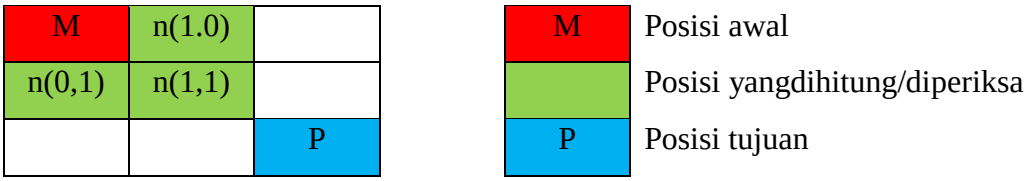
Gambar 3.5. **Kondisi Perhitugan yang Akan Diperiksa Algoritma A***

Keterangan :

M	Simpul awal atau musuh
P	Simpul tujuan atau pemain
	Simpul yang mungkin diperiksa

Langkah Awal

Berikut adalah Langkah awal akan dihitung pada koordinat (1,1), yaitu koordinat n(1,0) dan koordinat n(0,1)terlihat pada gambar 3.6.



Gambar 3.6. **Langkah Awal perhitungan pada koordinat (1,1), koordinat (1,0), dan koordinat (0,1)**

1. Koordinat n(1,0)

   Nilai n.x = 1

   Nilai n.y = 0

   Simpul tujuan(2,2) sehingga goal.x = 2, goal.y = 2

   g (1,0) = 1

   h_orthogonal(n) = (abs(n.x-goal.x) + abs(n.y-goal.y))

   h_orthogonal(1,0) = (abs(1 - 2) + abs(0 - 2))

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

   = 3

   h_diagonal(n) = min(abs(n.x-goal.x), abs(n.y-goal.y))

   h_diagonal(1,0) = min(abs(1 - 2)+abs(0 - 2))

   = min(abs(-1)+abs(-2))

   = min 3

   h(n) = h_diagonal(n) + (h_orthogonal (n) – (2 * h_diagonal(n)))

   h(1,0) = (- 3) + (3-(2*(-3)))

   = -3 + 9

   = 6

   f (1,0) = g (1,0) + h (1,0)

   = 1 + 6

   = 7

2. Koordinat n(0,1)

Nilai n.x = 0

Nilai n.y = 1

Simpul tujuan(2,2) sehingga goal.x = 2, goal.y = 2

g(n) = g (0,1) = 1

h_orthogonal(n) = (abs(n.x-goal.x) + abs(n.y-goal.y))

h_orthogonal(0,1) = (abs(0 - 2) + abs(1 - 2))

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

= 3

h_diagonal(n) = min(abs(n.x-goal.x) + abs(n.y-goal.y))

h_diagonal(0,1) = min(abs(0 - 2)+abs(1 - 2))

= min(abs(-2)+abs(-1))

= min 3

h(n) = h_diagonal(n) + (h_orthogonal (n) – (2 * h_diagonal(n)))

h(0,1) = (-3) + (3-(2*(-3)))

= -3 + 9

= 6

f (0,1) = g (0,1) + h (0,1)

= 1 + 6

= 7

3. Koordinat (1,1)

Nilai n.x = 1

Nilai n.y = 1

Simpul tujuan(2,2) sehingga goal.x = 2, goal.y = 2

g (1,1) = 1

h_orthogonal(n) = (abs(n.x-goal.x) + abs(n.y-goal.y))

h_orthogonal(1,1) = (abs(1 - 2) + abs(1 - 2))

= (abs(-1) + abs(-1))

= 2

h_diagonal(n) = min(abs(n.x-goal.x) + abs(n.y-goal.y))

h_diagonal(1,1) = min(abs(1 - 2)+abs(1 - 2))

= min(abs(-1)+abs(-1))

= min 2

h(n) = h_diagonal(n) + (h_orthogonal (n) – (2 * h_diagonal(n)))

h(1,1) = (-2) + (2-(2*(-2)))

= -2 + 6

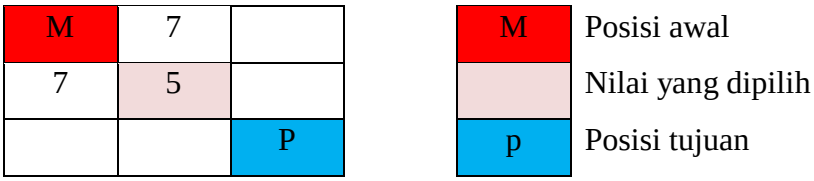
= 4

f (1,1) = g (1,1) + h (1,1)

= 1 + 4

= 5

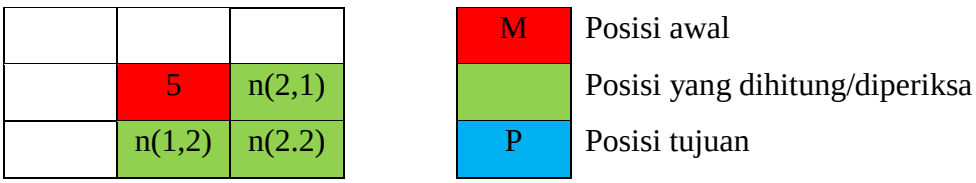
Berdasarkan perhitungan langkah pertama maka hasil perhitungan memiliki nilai f tekecil yaitu 5. Terlihat pada gambar 3.7.



Gambar 3.7. **Hasil Langkah Pertama Perhitungan pada koordinat (1,1), koordinat (1,0), dan koordinat (0,1)**

Langkah kedua

Dari hasil perhitungan pertama dihasilkan nilai f terkecil adalah 5, yaitu pada koordinat 1,1, maka pencarian nilai f terkecil selanjutnya dimulai dari koordinat 1,1, untuk mencari simpul selanjutnya yang mungkin dicari adalah koordinat n(2,1), koordinat n(2,2) dan koordinat n(1,2), terlihat pada gambar 3.8.



Gambar 3.8. **Langkah kedua perhitungan pada koordinat (2,1), koordinat (2,2), dan koordinat (1,2)**

1. Koordinat n(2,1)

Nilai n.x = 2

Nilai n.y = 1

Simpul tujuan(2,2) sehingga goal.x = 2, goal.y = 2

h_orthogonal(n) = (abs(n.x-goal.x) + abs(n.y-goal.y))

h_orthogonal(2,2) = (abs(2 - 2) + abs(1 - 2))

= (abs(0) + abs(-1))

= 1

h_diagonal(n) = min(abs(n.x-goal.x) + abs(n.y-goal.y))

h_diagonal(2,2) = min(abs(2 - 2)+abs(1 - 2))

= min(abs(0)+abs(-1))

= min 1

h(n) = h_diagonal(n) + (h_orthogonal (n) – (2 * h_diagonal(n)))

h(2,2) = (-1) + (1-(2*(-1)))

= -1 + 3

= 2

f (2,2) = g (2,2) + h (2,2)

= 2 + 2

= 4

2. Koordinat n(1,2)

Nilai n.x = 1

Nilai n.y = 2

Simpul tujuan(2,2) sehingga goal.x = 2, goal.y = 2

h_orthogonal(n) = (abs(n.x-goal.x) + abs(n.y-goal.y))

h_orthogonal(2,2) = (abs(1 - 2) + abs(2 - 2))

= (abs(-1) + abs(0))

= 1

h_diagonal(n) = min(abs(n.x-goal.x) + abs(n.y-goal.y))

h_diagonal(2,2) = min(abs(1 - 2)+abs(2 - 2))

= min(abs(-1)+abs(0))

= min 1

h(n) = h_diagonal(n) + (h_orthogonal (n) – (2 * h_diagonal(n)))

h(2,2) = (-1) + (1-(2*(-1)))

= -1 + 3

= 2

f (2,2) = g (2,2) + h (2,2)

= 2 + 2

= 4

3. Koordinat n(2,2)

Nilai n.x = 2

Nilai n.y = 2

Simpul tujuan(2,2) sehingga goal.x = 2, goal.y = 2

g(n) = g (2,2) = 2

h_orthogonal(n) = (abs(n.x-goal.x) + abs(n.y-goal.y))

h_orthogonal(2,2) = (abs(2 - 2) + abs(2 - 2))

= (abs(0) + abs(0))

= 0

h_diagonal(n) = min(abs(n.x-goal.x) + abs(n.y-goal.y))

h_diagonal(2,2) = min(abs(2 - 2)+abs(2 - 2))

= min(abs(0)+abs(0))

= min 0

h(n) = h_diagonal(n) + (h_orthogonal (n) – (2 * h_diagonal(n)))

h(2,2) = (-0) + (0-(2*(-0)))

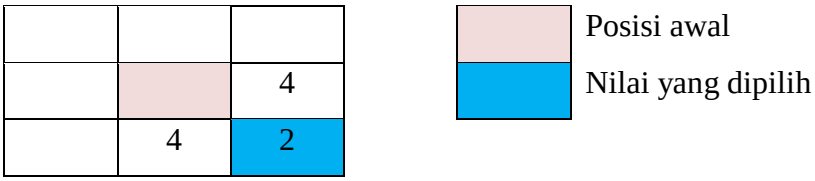
= 0 + 0

= 0

f (2,2) = g (2,2) + h (2,2)

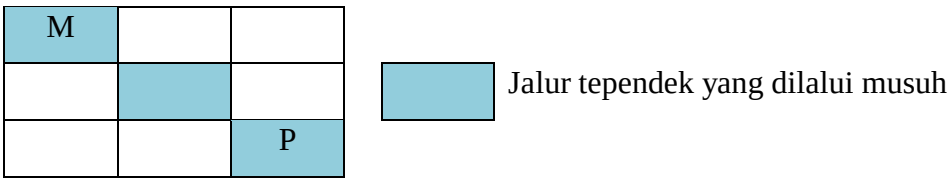
= 2 + 0

= 2



Gambar 3.9. **Hasil Perhitungan Langkah kedua perhitungan pada koordinat (2,1), koordinat (2,2), dan koordinat (1,2)**

Dari perhitungan pertama dan kedua yang telah dilakukan maka di pilih lah nilai terkecil pada setiap langkahnya sehingga akan menghasilkan jalur terpendek yang akan dilalui musuh untuk melakukan pencarian posisi pemain seperti pada gambar 3.10.



Gambar 3.10. **Jalur Terpendek Yang dilalui musuh untuk Mencari posisi Pemain**

Setelah melakukan pengecekan kotak sekitar, kotak sekitar yang terisi oleh hero maupun oleh penghalang maka tidak dilakukan perhitungan, dan untuk kotak sekitar yang kosong maka dilakukan perhitungan pada setiap musuh berpindah dari satu kotak ke kotak lainnya.

Flowchart dari algoritma A* (*A star*) pada aplikasi permainan *The Legend Of Cursed Amulet* ditunjukkan pada gambar3.11.



Gambar 3.11. **Flowchart algoritma A* The Legend Of Cursed Amulet**

3.2.4. Analisis dan Kebutuhan Non Fungsional

3.2.4.1. Analisis Kebutuhan Perangkat Keras

Analisis perangkat keras yang digunakan untuk mengetahui spesifikasi perangkat keras dalam membangun aplikasi permainan *The Legend Of Cursed Amulet*, seperti pada tabel 3.7.

Tabel 3.7. Spesifikasi Perangkat Keras Developer

Nama Perangkat	Spesifikasi
Prosesor	2.2GHz
RAM	512MB
Harddisk	50GB
Display	Resolusi 800 x 600
Keyboard	Standar
Mouse	Standar

Analisis perangkat keras terhadap pengguna dalam menjalankan aplikasi permainan *The Legend Of Cursed Amulet*, seperti pada tabel 3.8.

Tabel 3.8. Spesifikasi Perangkat Keras Pengguna

Nama Perangkat	Spesifikasi
Prosesor	1.8 GHZ
RAM	512MB
Harddisk	40GB
Display	Resolusi 800 x 600
Keyboard	Standar
Mouse	Standar

3.2.4.2. Analisis Kebutuhan Perangkat Lunak

Analisis perangkat lunak digunakan untuk mengetahui spesifikasi perangkat keras dalam menjalankan aplikasi permainan *The Legend Of Cursed Amulet*, seperti pada tabel 3.9.

Tabel 3.9. Spesifikasi Perangkat Lunak Pengguna

Nama Perangkat Lunak	Spesifikasi
Sistem Operasi	Microsoft Windows XP, Linux
Tools	JDK 6 Greenfoot

3.2.4.3. Analisis Kebutuhan Perangkat Pikir

Spesifikasi kebutuhan perangkat pikir padapermainan *The Legend Of Cursed Amulet*, seperti pada tabel 3.10.

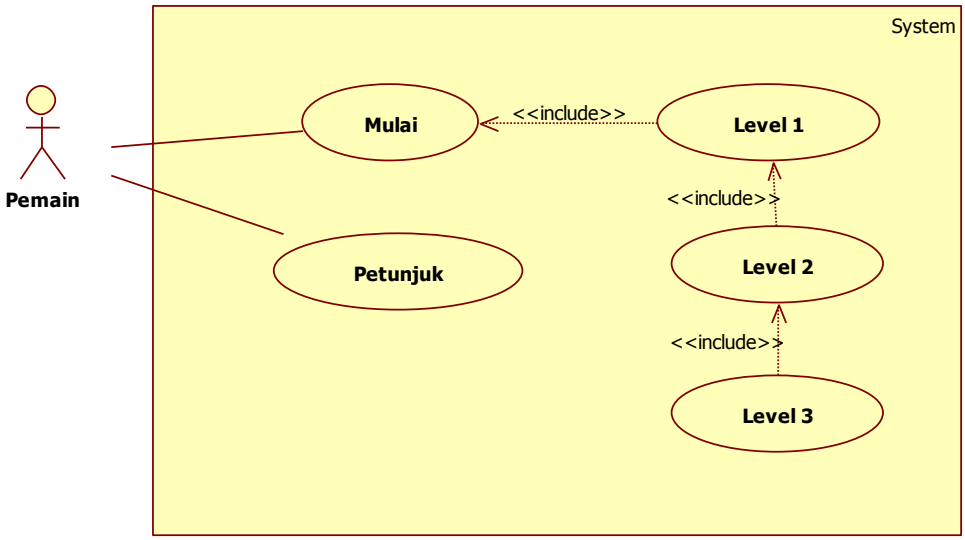
Tabel 3.10. Analisis Kebutuhan Perangkat Pikir

Ketentuan	Spesifikasi
Peran	Pemain
Usia	15 tahun keatas
Kondisi fisik	Tidak buta, memiliki gangguan fisik yang tidak memungkinkan pengguna untuk mengoperasikan komputer
Pengalaman	Pernah memainkan permainan yang sejenis maupun *game* dengan jenis permainan yang berbeda pada komputer

3.2.5. Analisis Kebutuhan Fungsional

3.2.5.1. Diagram *Use Case*

Secara umum, interaksi antara pengguna yaitu pemain dengan aplikasi permainan *The Legend Of Cursed Amulet* digambarkan pada diagram *Use Case*, seperti pada gambar 3.12.



Gambar 3.12. **Diagram *Use Case*The Legend Of Cursed Amulet**

Definisi dari *Use Case* yang terdapat pada Gambar Diagram *Use Case* adalah terdapat pada tabel 3.11.

Tabel 3.11. Definisi *Use Case*

No	Use Case	Deskripsi
1	Mulai	Proses ketika pemain membuka menu utama dan memilih tombol mulai pada permainan
2	Petunjuk	Proses ketika pemain memilih tombol petunjuk pada menu utama
3	Level 1	Proses ketika memasiki arena *level* 1
4	Level 2	Proses ketika memasiki arena *level* 2
5	Level 3	Proses ketika memasiki arena *level* 3

3.2.5.2. Skenario *Use Case*

Skenario pada diagram *Use Case* berisi tentang alur kejadian ‘flow of 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 *The Legend Of Cursed Amulet* ini adalah sebagai berikut:

1. Skenario *Use Case* Mulai

Skenario *Use Case* ini menjelaskan interaksi antara aktor, yaitu pemain, dengan *Use Case*memulai permainan pada menu utama aplikasi permainan*The Legend Of Cursed Amulet* yang dijelaskan pada tabel 3.12.

Nama *Use Case* : Mulai

Skenario :

Tabel 3.12. Skenario *Use Case*Mulai

Identifikasi		
Nama	Mulai	
Tujuan	Memulai Aplikasi	
Aktor	Pemain	
Skenario Utama		
Kondisi Awal	Membuka aplikasi permainan	
Aksi Aktor		**Reaksi Sistem**
a. Memilih tombol mulai		
		b. Menampilkan Intro
c. Menekan Tombol SPACE		
		d. Menampilkan arena*level*1
Kondisi Akhir	Arena *level* 1 ditampilkan	

2. Skenario *Use Case* Petunjuk

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

Nama *Use Case* : Petunjuk

Skenario :

Tabel 3.13. Skenario *Use Case* Petunjuk

Identifikasi		
Nama	Petunjuk	
Tujuan	Menampilkan Petunjuk	
Aktor	Pemain	
Skenario Normal		
Kondisi Awal	Menu utama ditampilkan	
Aksi Aktor		**Reaksi Sistem**
a. Memilih Tombol Petunjuk		
		b. Menampilkan layar petunjuk permainan
Kondisi Akhir	Menampilkan layar petunjuk permainan	

3. Skenario *Use CaseLevel* 1

Skenario *Use Case* ini menjelaskan interaksi antara aktor, yaitu pemain, dengan *Use Case* mulai permainan *level* 1 yang dijelaskan pada tabel 3.14.

Nama *Use Case* :*Level* 1

Skenario :

Tabel 3.14. Skenario Use CaseLevel 1

Identifikasi		
Nama	*Level* 1	
Tujuan	Menampilkan arena *level*1	
Aktor	Pemain	
Skenario Normal		
Kondisi Awal	Arena *level*1 ditampilkan	
Aksi Aktor		**Reaksi Sistem**
a. Menggerakan hero		
		b.Menggerakan hero sesuai input pemain
		c. Cek tabrakan hero
		d. Jika hero menabrak musuhlife berkurang

	e. cek life poin f. Jika life=0, maka menampilkan permainan berakhir g. jika Jika life>0, hero masih bisa bergerak h. Jika tidak menabrak, hero masih bisa bergerak
	i. Menampilkan gerakan hero
jMenekan tombol B	
	k. Hero menembakan peluru
	l. Cek tabrakan peluru
	m. Jika peluru menabrak musuh, life musuh berkurang n Jika life musuh <0, maka musuh mati. o. Jika tidak, life musuh tidak berkurang p. Cek kondisi jumlah musuh q. Jika musuh habis melanjutkan ke level 2 r. Jika musuh belum habis, permainan masih dilanjutkan
Kondisi Akhir	Menampilkan arena level 2

4. Skenario *Use CaseLevel* 2

Skenario *Use Case* ini menjelaskan interaksi antara aktor, yaitu pemain, dengan *Use Caselevel*2 yang dijelaskan pada tabel 3.15.

Nama *Use Case* :*Level* 2

Skenaroio :

Tabel 3.15. Skenario Use CaseLevel 2

Identifikasi	
Nama	*Level* 2
Tujuan	Menampilkan arena *level*2
Aktor	Pemain
Skenario Normal	
Kondisi Awal	Arena *level*2 ditampilkan

Aksi Aktor	**Reaksi Sistem**
a. Menggerakan hero	
	b. Menggerakan hero sesuai input pemain
	c. Cek tabrakan hero
	d. Jika hero menabrak musuh life berkurang e. cek life poin f. Jika life=0, maka menampilkan permainan berakhir g. jika Jika life>0, hero masih bisa bergerak h. Jika tidak menabrak, hero masih bisa bergerak
	i. Menampilkan gerakan hero
j Menekan tombol B	
	k. Hero menembakan peluru
	l. Cek tabrakan peluru
	m. Jika peluru menabrak musuh, life musuh berkurang n Jika life musuh <0, maka musuh mati. o. Jika tidak, life musuh tidak berkurang p. Cek kondisi jumlah musuh q. Jika musuh habis melanjutkan ke level 3

	r. Jika musuh belum habis, permainan masih dilanjutkan
Kondisi Akhir	Menampilkan arena level 3

5. Skenario *Use Case* Mulai Permainan *Level* 3

Skenario *Use Case* ini menjelaskan interaksi antara aktor, yaitu pemain, dengan *Use Caselevel*3 yang dijelaskan pada tabel 3.16.

Nama *Use Case* :*Level*3

Skenario :

Tabel 3.16. Skenario Use CaseLevel 3

Identifikasi	
Nama	*Level* 3
Tujuan	Menampilkan arena *level*3
Aktor	Pemain
Skenario Normal	
Kondisi Awal	Arena *level*2 ditampilkan
Aksi Aktor	**Reaksi Sistem**
a. Menggerakan hero	
	b. Menggerakan hero sesuai input pemain
	c. Cek tabrakan hero
	d. Jika hero menabrak musuh life berkurang e. cek life poin f. Jika life=0, maka menampilkan permainan berakhir g. jika Jika life>0, hero masih bisa bergerak h. Jika tidak menabrak, hero masih bisa bergerak
	i. Menampilkan gerakan hero

j Menekan tombol B	
	k. Hero menembakan peluru
	l. Cek tabrakan peluru
	m. Jika peluru menabrak musuh, life musuh berkurang n Jika life musuh =0, maka musuh mati. o. Jika tidak, life musuh tidak berkurang p. Cek kondisi jumlah musuh q. Jika musuh habis permainnan berakhir r. Jika musuh belum habis, permainan masih dilanjutkan
Kondisi Akhir	Menampilkan informasi permainnan selesai

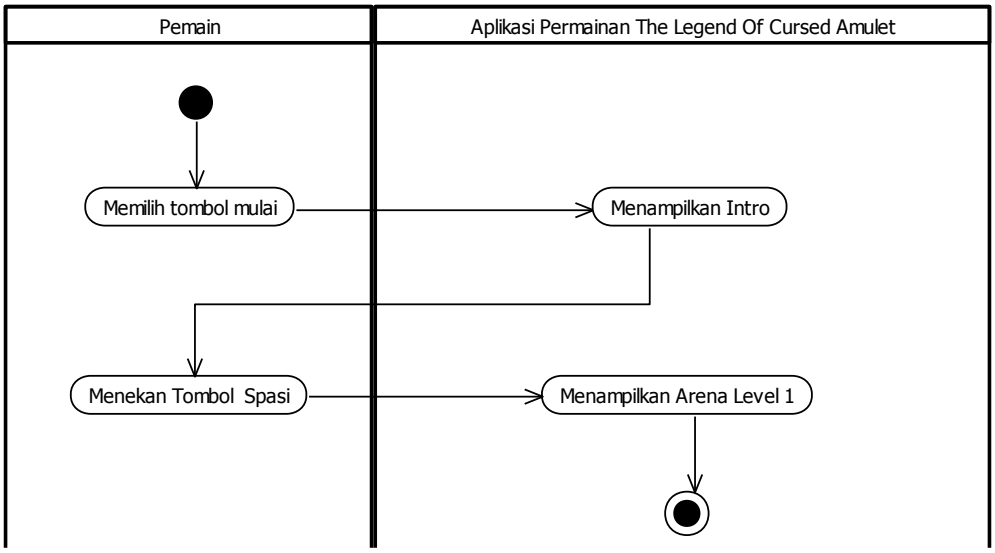
3.2.5.3. Diagram *Activity*

Diagram ini berisi penjelasan secara lengkap mengenai urutan aktivitas yang terjadi antara pemain dengan aplikasi permainan *The Legend Of Cursed Amulet*.

1. Diagram *Activity*Mulai Permainan

Proses ini terjadi saat pemain memulai aplikasi permainan *The Legend Of Cursed Amulet*,Sistem akan menampilkan halaman utama dari aplikasi permainan*The Legend Of Cursed Amulet*.

Diagram *Activity*Mulai Permainan terdapat pada gambar3.13.

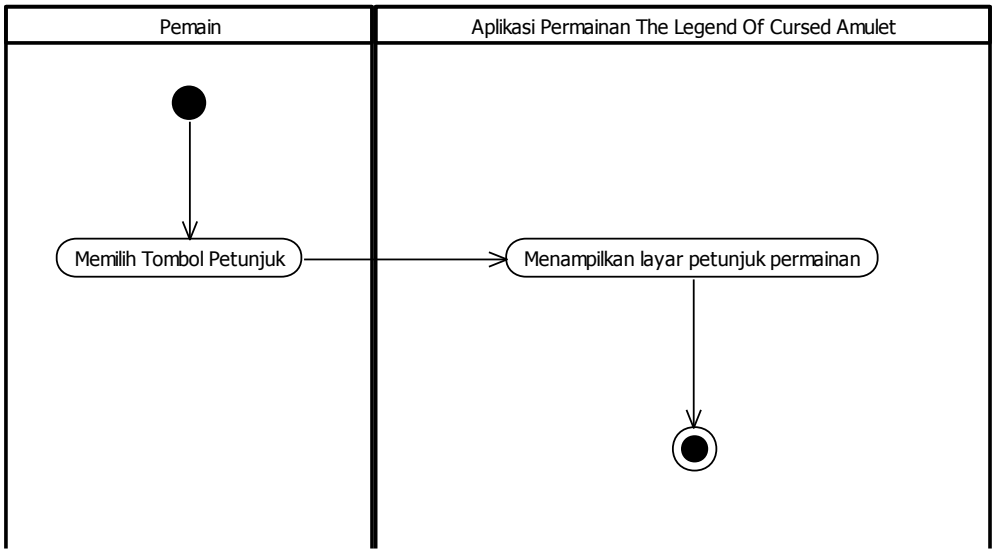


Gambar 3.13. **Diagram *Activity* Mulai Permainan pada Permainan *The Legend Of Cursed Amulet***

2. Diagram *Activity* Tampilan Petunjuk

Proses ini terjadi saat pemain memulai permainan dan memilih petunjuk pada menu utama aplikasi *The Legend Of Cursed Amulet*,Sistem akan menampilkan petunjuk permainan dari aplikasi permainan*The Legend Of Cursed Amulet*.

Diagram *Activity* tampil petunjuk terdapat pada gambar3.14.

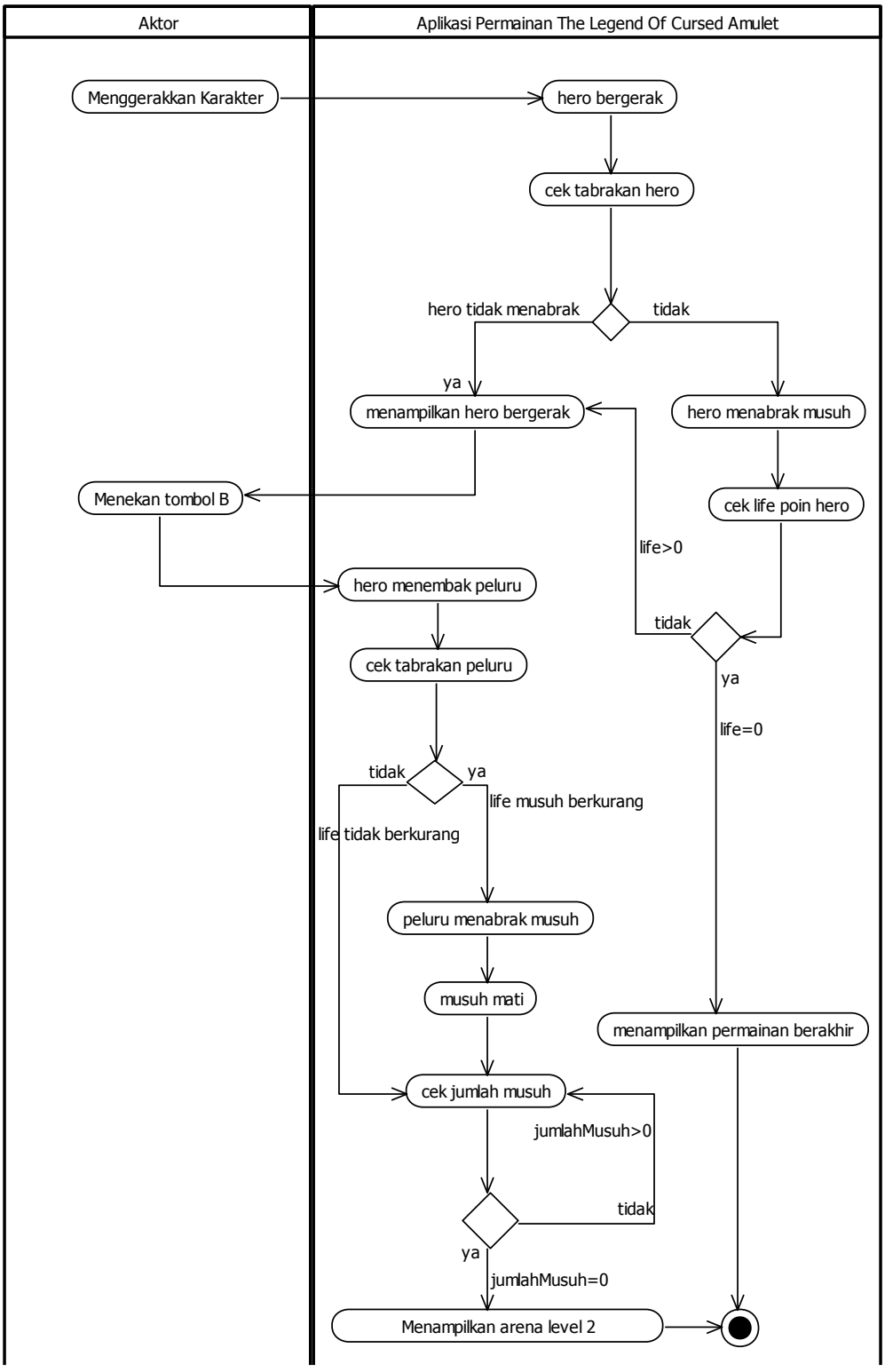


Gambar 3.14. **Diagram *Activity* Tampil Petunjuk pada Permainan *The Legend Of Cursed Amulet***

3. Diagram *Activity* Permainan Pada *Level* 1

Proses ini terjadi saat pemain memilih menu mulai permainan pada menu utama aplikasi permainan *The Legend Of Cursed Amulet*. Sistem akan menampilkan arena permainan dimana pemain akan memulai dan menyelesaikan misi permainan pada *level* 1.

Diagram *Activity* mulai permainan *level* 1 terdapat pada gambar 3.15.

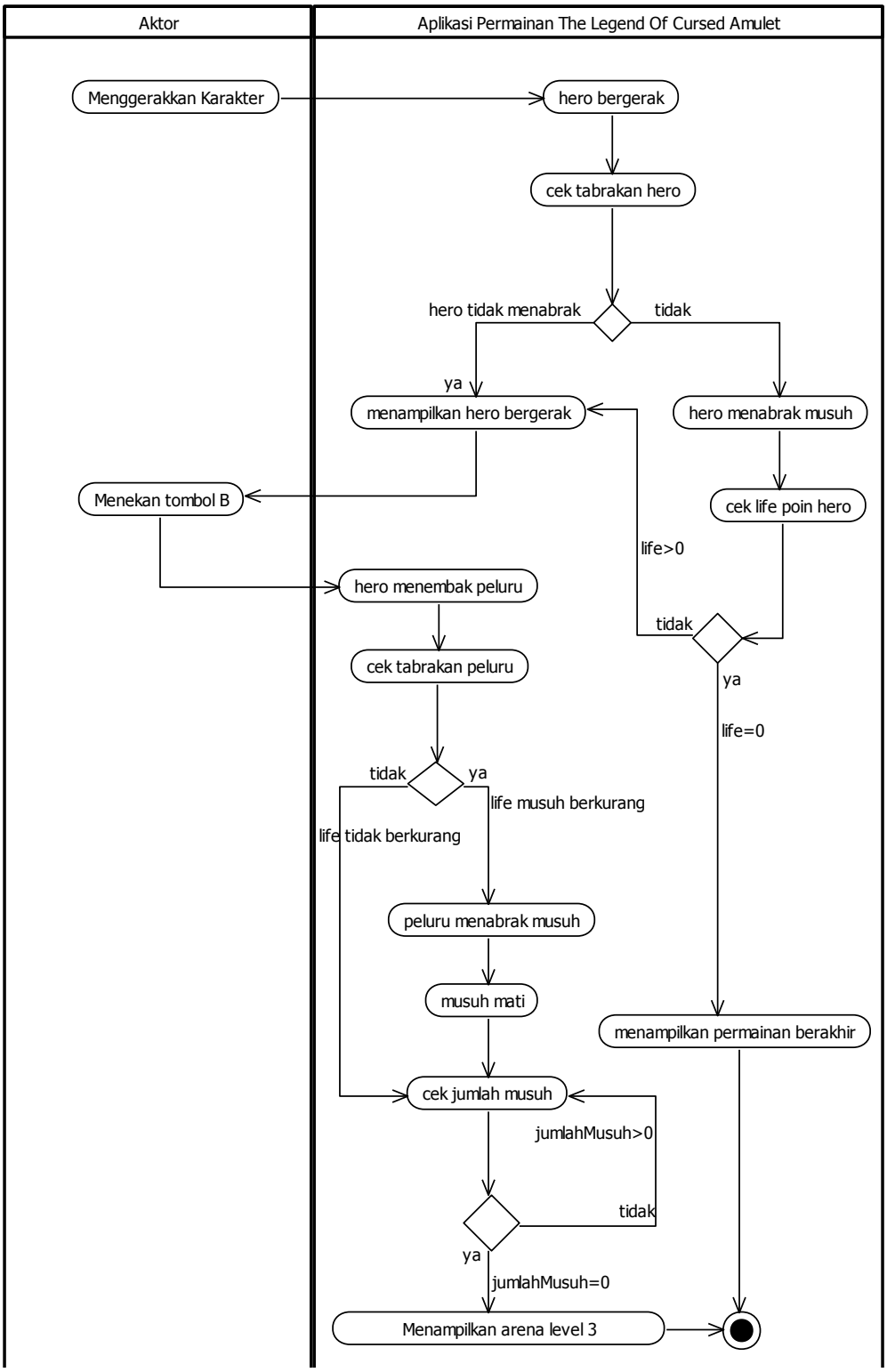


Gambar 3.15. **Diagram Activity Mulai Permainan Level 1 The Legend Of Cursed Amulet**

4. Diagram *Activity* Permainan Pada *Level* 2

Proses ini terjadi apabila pemain telah menyelesaikan permainan pada *The Legend Of Cursed Amulet level* 1.Maka sistem akan menampilkan arena pada *level* 2 dan pemain harus menyelesaikan permainan pada *level* 2, apabila misi permainan akan melanjutkan permainan ke *level* selanjutnua.

Diagram *Activity* mulai permainan *level*2 terdapat pada gambar 3.16.

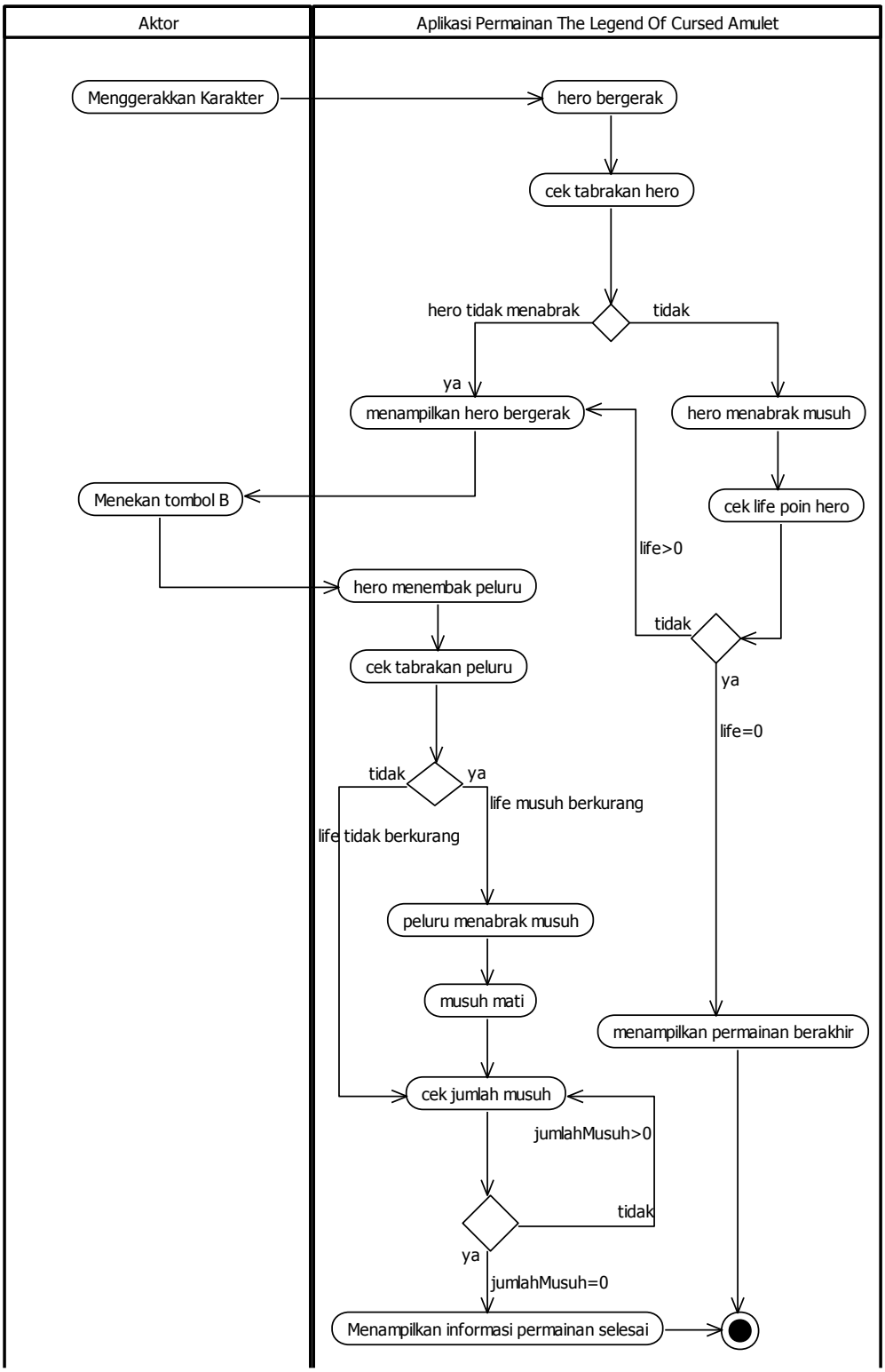


Gambar 3.16. **Diagram Activity Mulai Permainan Level 2 The Legend Of Cursed Amulet**

5. Diagram *Activity* Permainan Pada *Level* 3

Proses ini terjadi apabila pemain telah menyelesaikan permainan pada *The Legend Of Cursed Amulet level* 2.Maka sistem akan menampilkan arena pada *level* 3 dan pemain harus menyelesaikan permainan pada *level* 3, apabila misi permainan gagal maka pemain akan kembali bermain ke titik awal permainan tersebut.

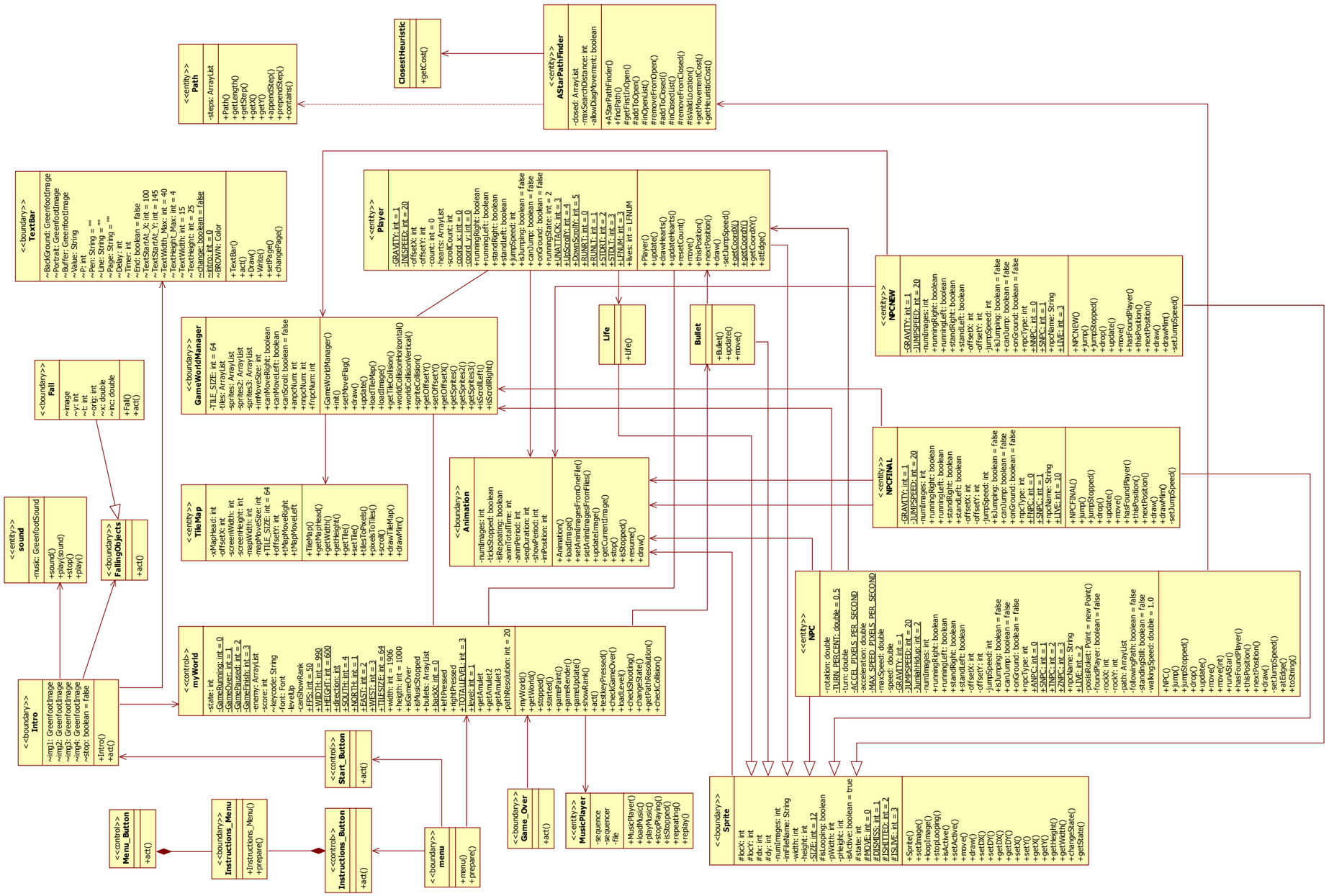
Diagram *Activity* mulai permainan *level*3 terdapat pada gambar 3.17.



Gambar 3.17. **Diagram Activity Mulai Permainan Level 3 The Legend Of Cursed Amulet**

3.2.5.4. Diagram *Class*

Diagram *Class* aplikasi permainan *The Legend Of Cursed Amulet*terdapat pada gambar 3.18.



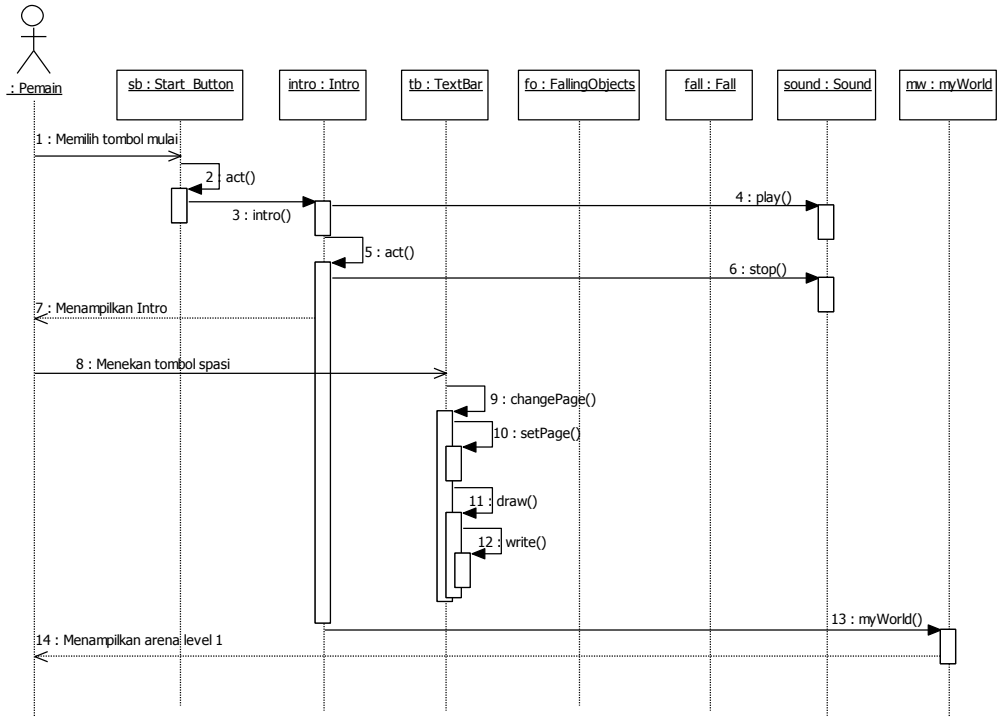
Gambar 3.18. **Diagram Class Permainan The Legend Of Cursed Amulet**

3.2.5.5. Diagram *Sequence*

Berikut ini merupakan diagram *Sequence* yang terdapat pada aplikasi permainan *The Legend Of Cursed Amulet* berdasarkan *Use Case* yang ada.

A. Diagram *Sequence* dari *Use Case* : Mulai

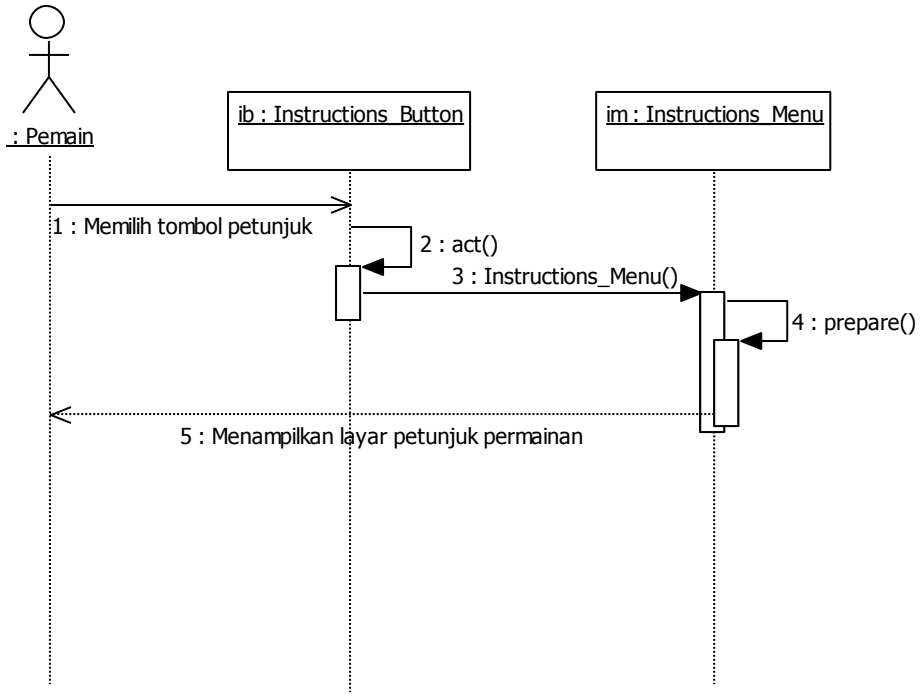
Diagram *Sequence* untuk memulai permainan terdapat pada gambar 3.19.



Gambar 3.19. **Diagram *Sequence* untuk *Use Case*Mulai**

B. Diagram *Sequence* dari *Use Case* : Petunjuk

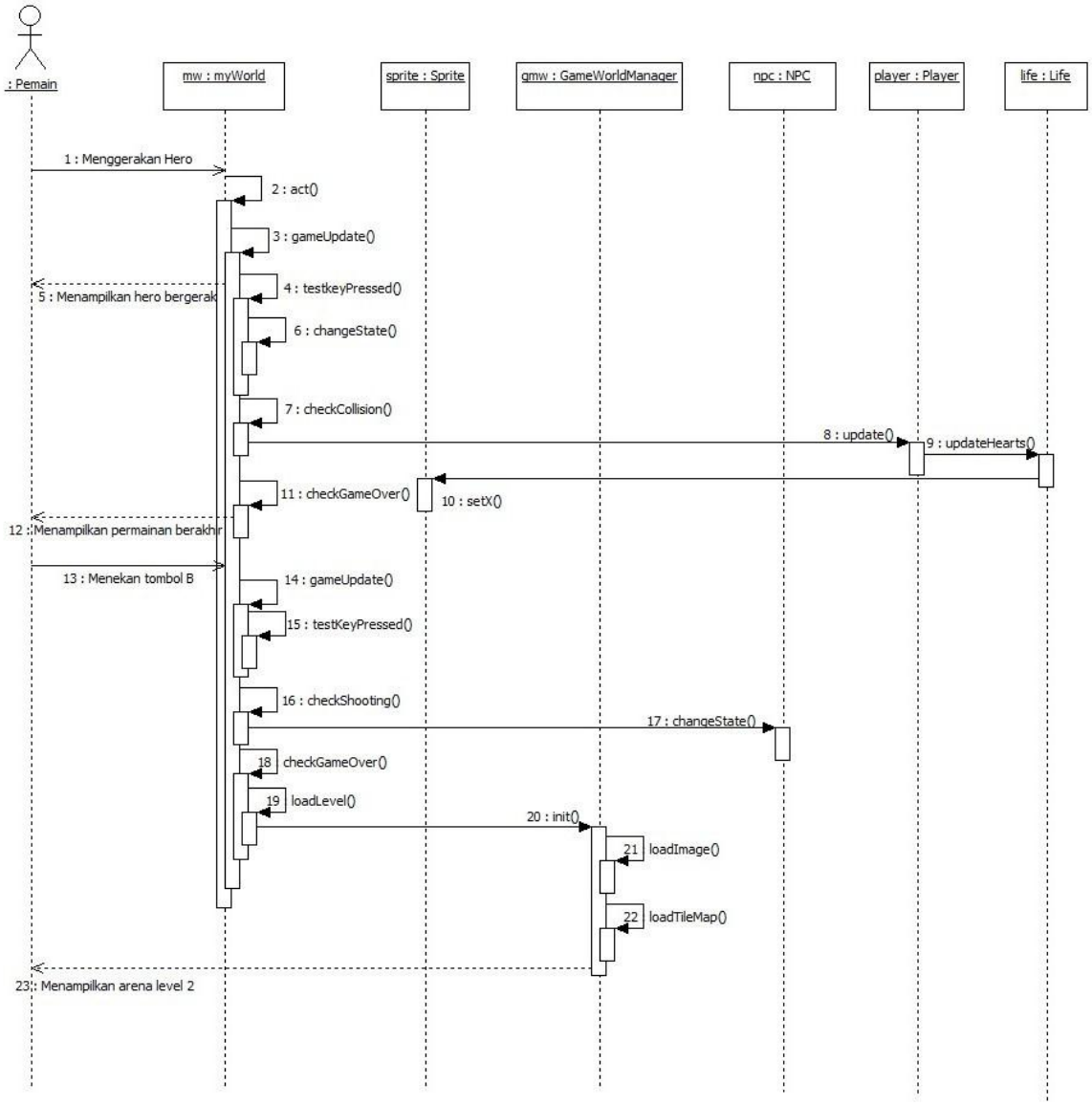
Diagram *Sequence* untuk melihat Petunjuk terdapat pada gambar 3.20.



Gambar 3.20. **Diagram *Sequence* untuk *Use Case* Petunjuk**

C. Diagram *Sequence* dari *Use Case* : *Level*1

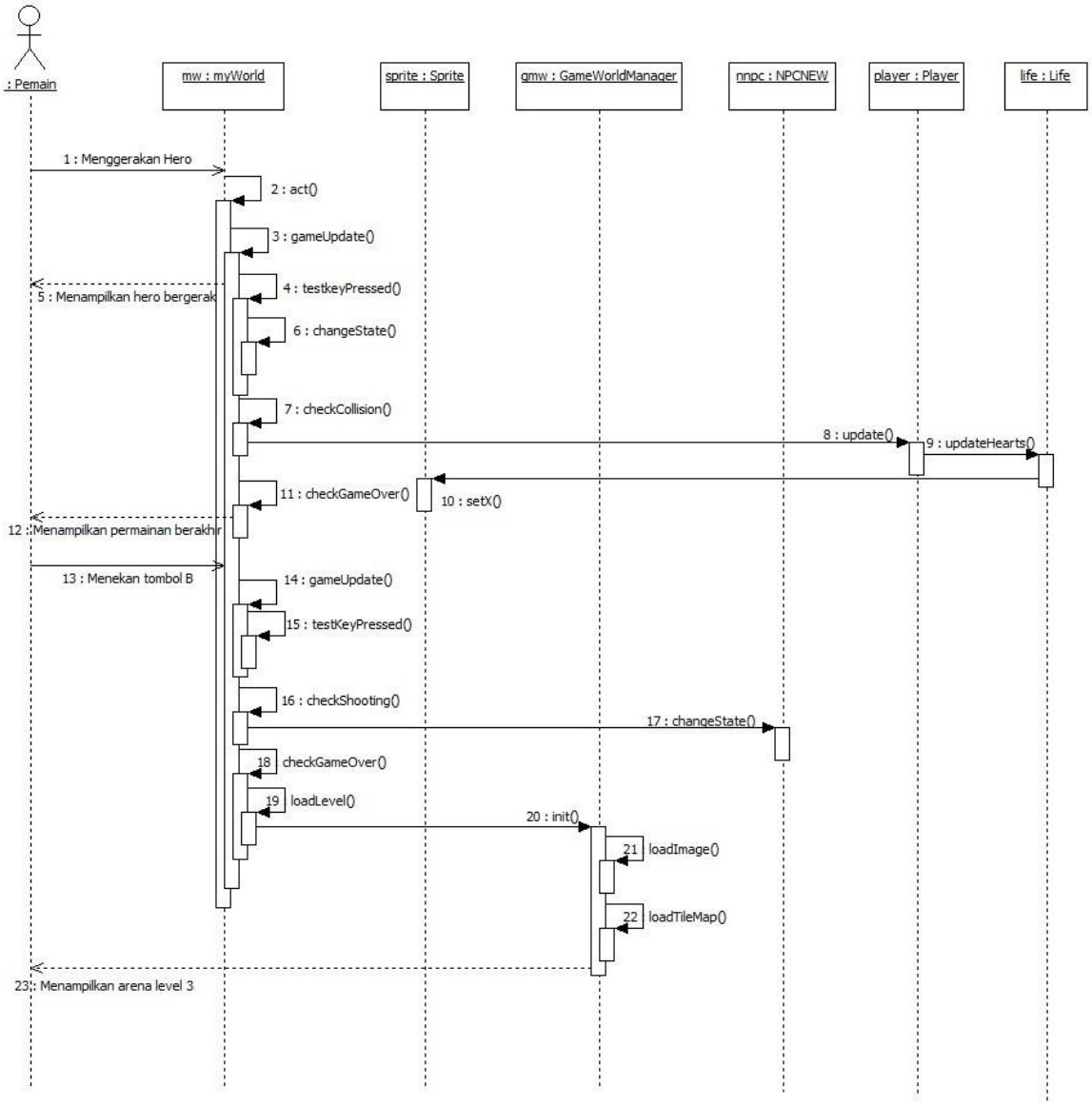
Diagram *Sequence* Mulai Permainan *Level* 1 terdapat pada gambar 3.21.



Gambar 3.21. **Diagram Sequence untuk Use CaseLevel 1**

D. Diagram *Sequence* dari *Use Case* : *Level* 2

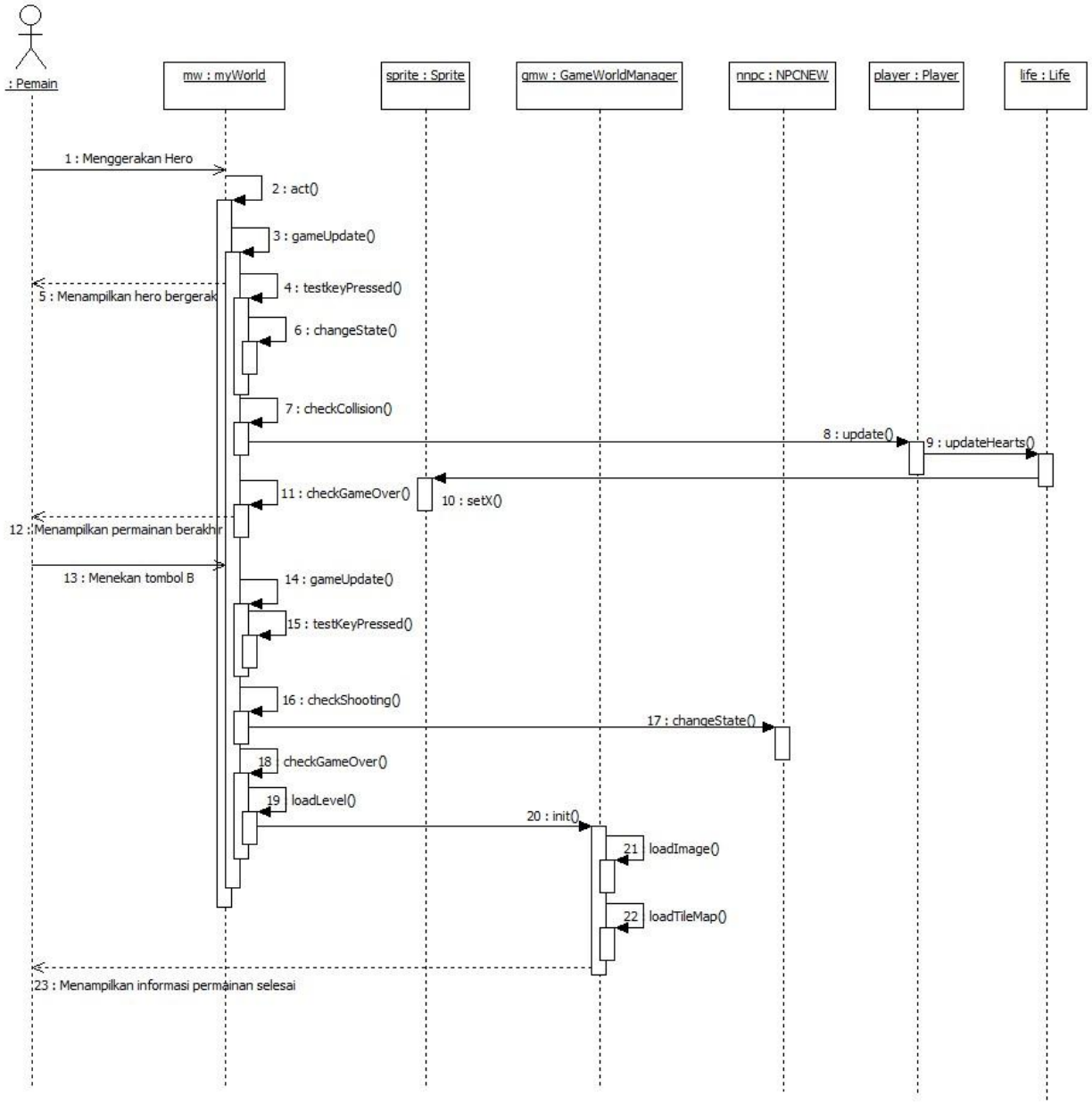
Diagram *Sequence*Mulai Permainan*Level* 2 terdapat pada gambar 3.22.



Gambar 3.22. **Diagram Sequence untuk Use CaseLevel 2**

E. Diagram *Sequence* dari *Use Case* : *Level* 3

Diagram *Sequence*Mulai Permainan *Level* 3 terdapat pada gambar 3.23.



Gambar 3.23. **Diagram Sequence untuk Use CaseLevel 3**

3.3. Perancangan Sistem

Perancangan sistem yang terdapat pada aplikasi permainan *The Legend Of Cursed Amulet* meliputi perancangan komponen permainan, perancangan struktur menu, perancangan antar muka, jaringan semantik dam perancangan *method*. Dalam perancangan sistem ini menunjukan rancangan sistem yang akan dibangun pada aplikasi permainan *The Legend Of Cursed Amulet.*

3.3.1. Perancangan Komponen Permainan

Perancangan komponen peramainan terdiri dari perancangan struktur kelas, entitas *game* dan *storyboard*. Perancangan komponen permainan menjelaskan tentang komponen-komponen yang terdapat pada aplikasi permainan *The Legend Of Cursed Amulet.*

3.3.1.1. Perancangan Struktur Kelas

Kelas yang terdapat pada sistem yang akan dibangun terdiri dari5 kelas *entity*. Struktur dari atribut kelas-kelas tersebut adalah sebagai berikut:

A. Struktur Atribut Kelas Player terdapat pada tabel 3.17.

Tabel 3.17. Struktur Atribut Kelas Player

Atribut	Tipe Data	Akses
GRAVITY	final static int	*Private*
INISPEED	final static int	*Private*
Offset	int	*Private*
Offset	int	*Private*
Count	Int	*Private*
Hearts	ArrayList	*Private*
Heart	Life	*Private*
scrollCount	Int	*Private*
coord_x	Static int	*Private*
coord_y	Static int	*Private*
runningRight	Boolean	*Public*

runningLeft	Boolean	*Public*
standRight	Boolean	*Public*
standLeft	Boolean	*Public*
jumpSpeed	Int	*Public*
isJumping	Boolean	*Public*
canJump	Boolean	*Public*
onGround	Boolean	*Public*
runningState	Int	*Public*
UNATTACK	static final int	*Public*
UpScrollY	static final int	*Public*
DownScrollY	static final int	*Public*
RUNRT	static final int	*Public*
RUNLT	static final int	*Public*
STDRT	static final int	*Public*
STDLT	static final int	*Public*
LFNUM	static final int	*Public*
Live	Int	*Public*

B. Struktur Atribut Kelas NPC terdapat pada tabel 3.18.

Tabel 3.18. Struktur Atribut Kelas NPC

Atribut	Tipe Data	Akses
Mx	MathX	*Private*
Rotation	Double	*Private*
TURN_PERCENT	final static double	*Private*
Turn	Double	*Private*
ACCEL_PIXELS_PER_SECOND	final static double	*Private*
Acceleration	Double	*Private*
AX_SPEED_PIXELS_PER_SECOND	final static double	*Public*
maxSpeed	Double	*Public*
Speed	Double	*Public*

GRAVITY	final static int	*Public*
JUMPSPEED	final static int	*Private*
JumlahHidup	final static int	*Private*
numImages	Int	*Public*
runningRight	Boolean	*Public*
runningLeft	Boolean	*Public*
standRight	Boolean	*Public*
standLeft	Boolean	*Public*
Offset	Int	*Public*
offsetY	Int	*Public*
jumpSpeed	Int	*Public*
isJumping	Boolean	*Public*
canJump	Boolean	*Public*
onGround	Boolean	*Public*
ANPC	static final int	*Public*
SNPC	static final int	*Public*
TNPC	static final int	*Public*
ZNPC	static final int	*Public*
npcName	String	*Public*
LIVE	static int	*Public*
posisiRoket	Point	*Private*
foundPlayer	Boolean	*Private*
rockX	Int	*Private*
rockY	Int	*Private*
Path	ArrayList	*Private*
followingPath	Boolean	*Private*
standingStill	Boolean	*Private*
walkingSpeed	Double	*Private*

C. Struktur Atribut Kelas NPCNEW terdapat pada tabel 3.19.

Tabel 3.19. Struktur Atribut Kelas NPCNEW

Atribut	Tipe Data	Akses
GRAVITY	final static int	*Public*
JUMPSPEED	final static int	*Private*
JumlahHidup	final static int	*Private*
numImages	int	*Public*
runningRight	Boolean	*Public*
runningLeft	Boolean	*Public*
standRight	Boolean	*Public*
standLeft	Boolean	*Public*
Offset	int	*Public*
offsetY	int	*Public*
jumpSpeed	int	*Public*
isJumping	Boolean	*Public*
canJump	Boolean	*Public*
onGround	Boolean	*Public*
NNPC	static final int	*Public*
SNPC	static final int	*Public*
npcName	String	*Public*
LIVE	static int	*Public*

D. Struktur Atribut Kelas NPCFINAL terdapat pada tabel 3.20.

Tabel 3.20. Struktur Atribut Kelas NPCFINAL

Atribut	Tipe Data	Akses
GRAVITY	final static int	*Public*
JUMPSPEED	final static int	*Private*
JumlahHidup	final static int	*Private*
numImages	Int	*Public*

runningRight	Boolean	*Public*
runningLeft	Boolean	*Public*
standRight	Boolean	*Public*
standLeft	Boolean	*Public*
Offset	Int	*Public*
offsetY	Int	*Public*
jumpSpeed	Int	*Public*
isJumping	Boolean	*Public*
canJump	Boolean	*Public*
onGround	Boolean	*Public*
NNPC	static final int	*Public*
SNPC	static final int	*Public*
npcName	String	*Public*
LIVE	static int	*Public*

E. Struktur Atribut Kelas TileMap terdapat pada tabel 3.21.

Tabel 3.21. Struktur Atribut Kelas TileMap

Atribut	Tipe Data	Akses
xMapHead	int	*Private*
offsetX	int	*Private*
screenWidth	int	*Private*
screenHeight	int	*Private*
mapWidth	int	*Private*
mapMoveSize	int	*Private*
TILE_SIZE	int	*Public*
offsetY	int	*Public*
tMapMoveRight	Boolean	*Public*
tMapMoveLeft	Boolean	*Public*

F. Struktur Atribut Kelas Sound terdapat pada tabel 3.22.

Tabel 3.22. Struktur Atribut Kelas Sound

Atribut	Tipe Data	Akses
music	GreenfootSound	*Private*

G. Struktur Atribut Kelas MusicPlayer terdapat pada tabel 3.23.

Tabel 3.23. Struktur Atribut Kelas MusicPlayer

Atribut	Tipe Data	Akses
sequence	Sequence	*Private*
sequencer	Sequencer	*Private*
file	File	*Private*

H. Struktur Atribut Kelas AstarPathFinderterdapat pada tabel 3.24.

Tabel 3.24. Struktur Atribut Kelas AStarPathFinder

Atribut	Tipe Data	Akses
closed	ArrayList	*Private*
maxSearchDistance	int	*Private*
allowDiagMovement	Boolean	*Private*

I. Struktur Atribut Kelas Path terdapat pada tabel 3.25.

Tabel 3.25. Struktur Atribut Kelas Path

Atribut	Tipe Data	Akses
steps	ArrayList	*Private*

3.3.1.2. Entitas *Game*

Entitas yang terdapat di dalam *game* terbagi menjadi dua, yaitu karakter hero dan NPC yang berupa musuh, selain itu terdapat juga item yang digunakan untuk membantu hero dalam pertahanan.

Entitas karakter dan item yang terdapat di dalam *gameThe Legend Of Cursed Amulet* ditunjukkan pada tabel 3.16.

Tabel 3.26. Entitas karakter dan item

NO	NAMA	GAMBAR	PENJELASAN
1	Hero		Pada karakter hero ini, merupakan karakter utama yang dapat dimainkan dengan kemampuan melompat untuk berpindah ke platform lain maupun menghindari musuh dan menembak untuk mengalahkan musuh[23]
2	Warg		Pada NPC (*Non-player character*s) ini, merupakan karakter musuh yang berada di daratan akan menyerang hero secara langsunguntuk mencegah pemain melewati wilayah monster tersebut[23] HP = 2
3	Bad Bird		Pada NPC (*Non-player character*s) ini, merupakan karakter musuhyang berbentuk burung[23] HP = 2
4	Slime		Pada NPC (*Non-player character*s) ini, merupakan karakter musuh yang berbentuk seperti cairan yang berbahaya[23] HP = 2

5	Lizardman		Pada NPC (*Non-player character*s) ini, merupakan karakter musuh yang lebih kuat dari warg, monster ini berada di daratan dan akan menyerang hero secara langsunguntuk mencegah pemain melewati wilayah monster tersebut[23] HP = 2
6	Fire Spirit		Pada NPC (*Non-player character*s) ini, merupakan karakter musuh yang lebih kuat dari Lizard, monster ini berada di daratan dan akan menyerang hero secara langsunguntuk mencegah pemain melewati wilayah monster tersebut[23] HP = 3
7	Ghost		Pada NPC (*Non-player character*s) ini, merupakan karakter musuh yang bergerak lebih cepat, akan menyerang hero secara langsunguntuk mencegah pemain melewati wilayah monster tersebut[23] HP = 3
8	Evil Ghost		Pada NPC (*Non-player character*s) ini, merupakan karakter musuh yang bergerak lebih cepat, akan menyerang hero secara langsunguntuk mencegah pemain melewati wilayah monster tersebut[23] HP = 3

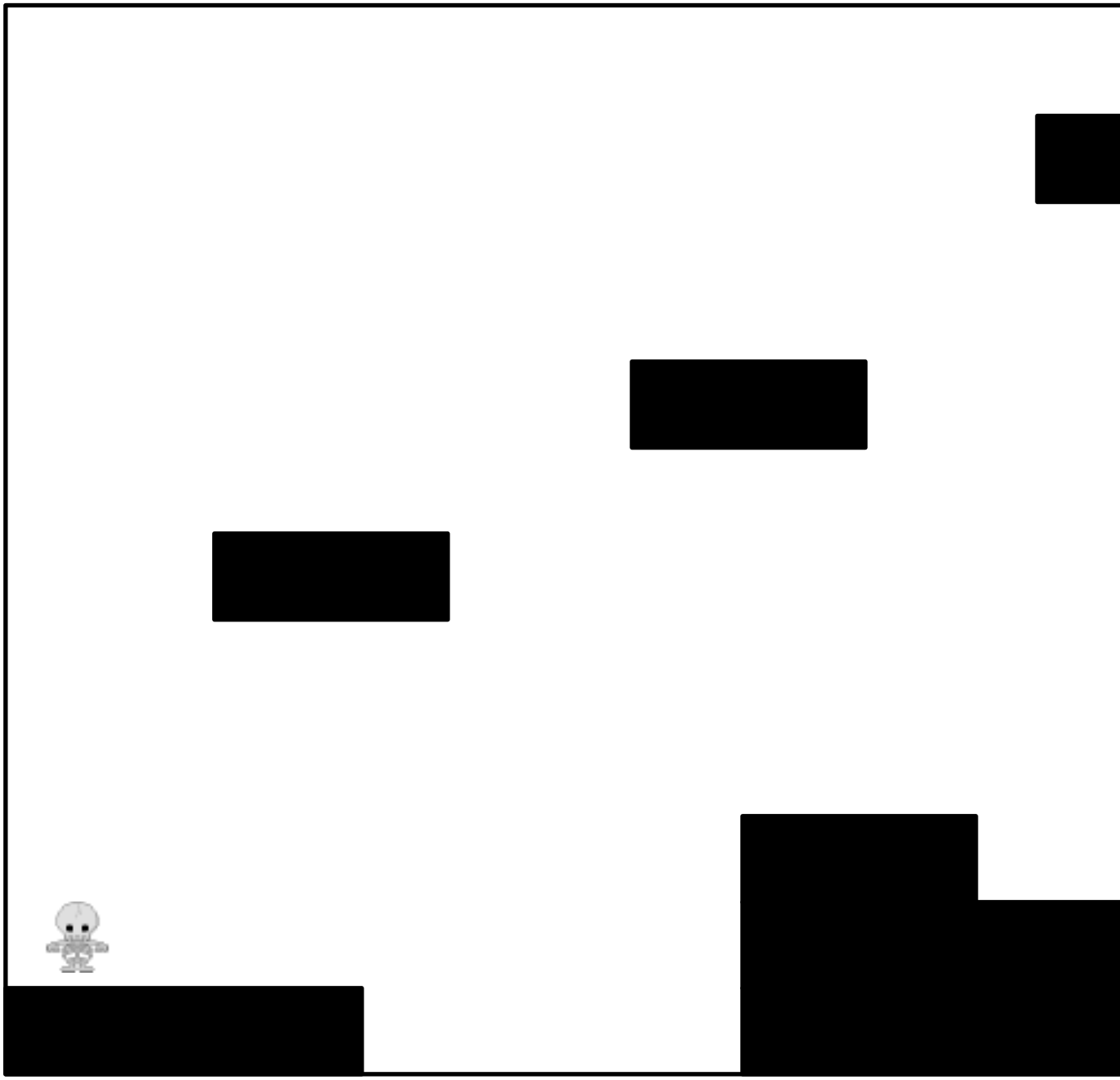
9	Skeleton		Pada NPC (*Non-player character*s) ini, merupakan karakter musuh yang berbentuk seperti tengkorak yang bangkit dari kematian[23] HP = 3
10	Boar		Pada NPC (*Non-player character*s) ini, merupakan karakter musuh yang berbentuk seperti Babi Hutan yang sangat buas[23] HP = 3
11	Assassin		Pada NPC (*Non-player character*s) ini, merupakan karakter boss dengan kecepatan dan serangan yang lebih besar karena lebih kuat dibanding musuh lainnya, harusdikalahkan untuk menyelesaikan permainnan tersebut[23] HP = 20
12	Whisper		Pada NPC (*Non-player character*s) ini, merupakan karakter musuh denganbentuk seperti hantu yang terbang[23]
13	Scroll		Sebuah item yang berfungsi untuk menambah poin[23]poin = 50
14	Health Potion		Sebuah item yang berfungsi untuk menambah health poinPemain[23]
15	Star Potion		Sebuah item yang berfungsi untuk membuat karakter kebal untuk beberapa detik[23]

16	Shuriken		Jenis peluru ini adalah peluru yang ditembakkan oleh hero yang pemain pasang[23]
17	Crystal Amulet		Jenis kristal yang harus diambil oleh pemain disetiap *level*agar bisa melanjutkan ke *level* berikutnya[23]

3.3.1.3. STORYBOARD

STORYBOARD merupakan sketsa gambar yang disusun berurutan sesuai naskah yang ada, dan dengan *STORYBOARD* kita dapat menyampaikan ide cerita kita kepada orang lain dengan mudah.

STORYBOARDlevel 1 dari aplikasi *The Legend Of Cursed Amulet*, terdapat pada gambar 3.24.

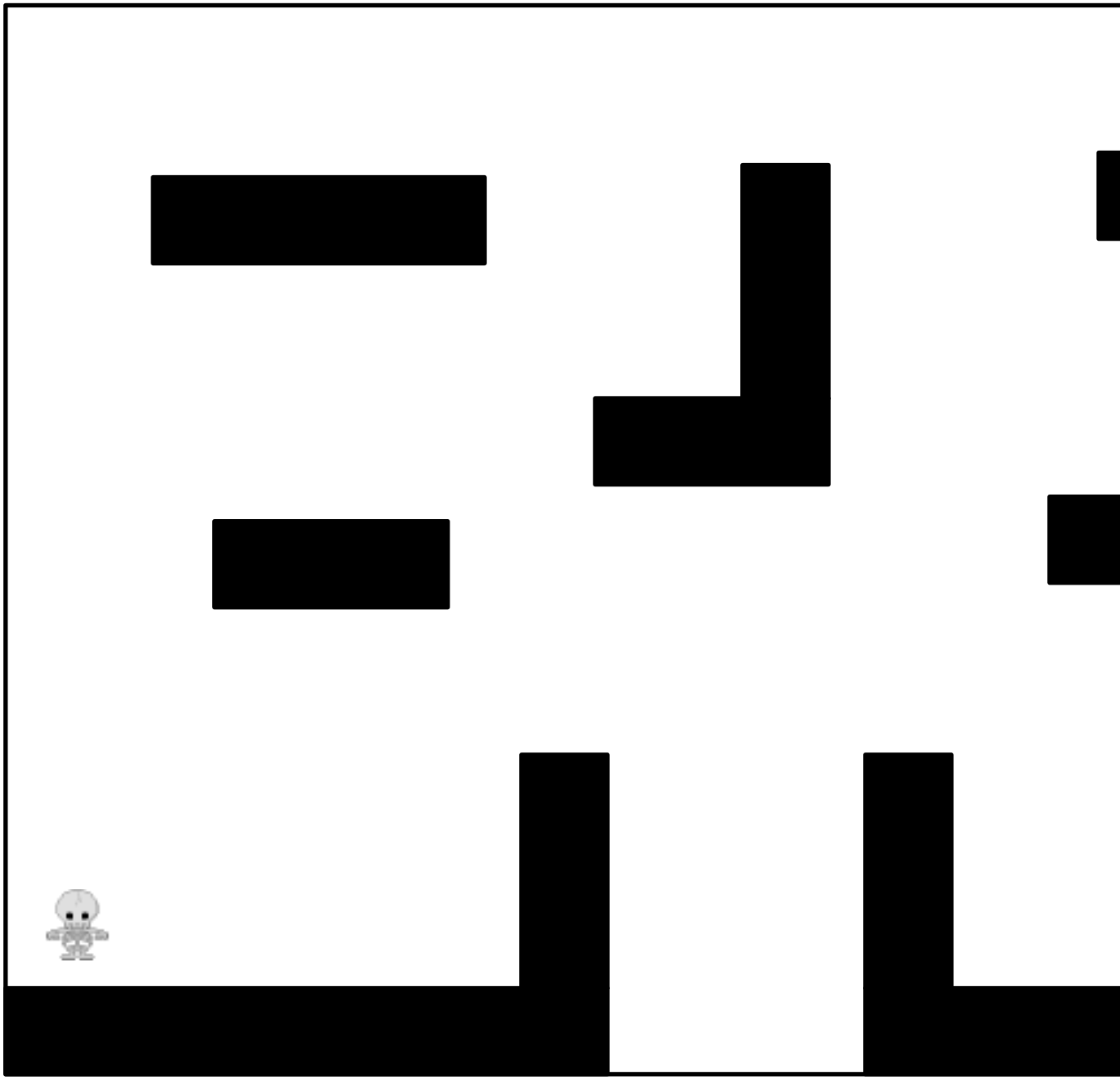


Gambar 3.24. **STORYBOARD level 1**

Pada *level* 1 ini, hero akan bergerak dan melompat dari point A(posisi awal) menuju menuju platform(tempat yang bisa dipakai sebagai pijakan) sesuai dengan jalur yang telah disediakan, musuh yang akan menghadang pada *level* ini yaitu musuh dengan tingkat kesulitan yang masih mudah. Setiap hero pada *level* ini dapat mengalahkan semua jenis monster untuk melakukan pertahanan dengan

caramenembak monster yang akan menghadangdan menyerang sesuai dengan kondisi apabila hero mendekati jarak wilayah serang musuh. Permainnan hanya bisa dilanjutkan ke *level* berikutnya apabila semua monster telah dikalahkan dan semua item telah didapatkan.

STORYBOARD level 2 dari aplikasi *gameThe Legend Of Cursed Amulet*, terdapat pada gambar 3.25.

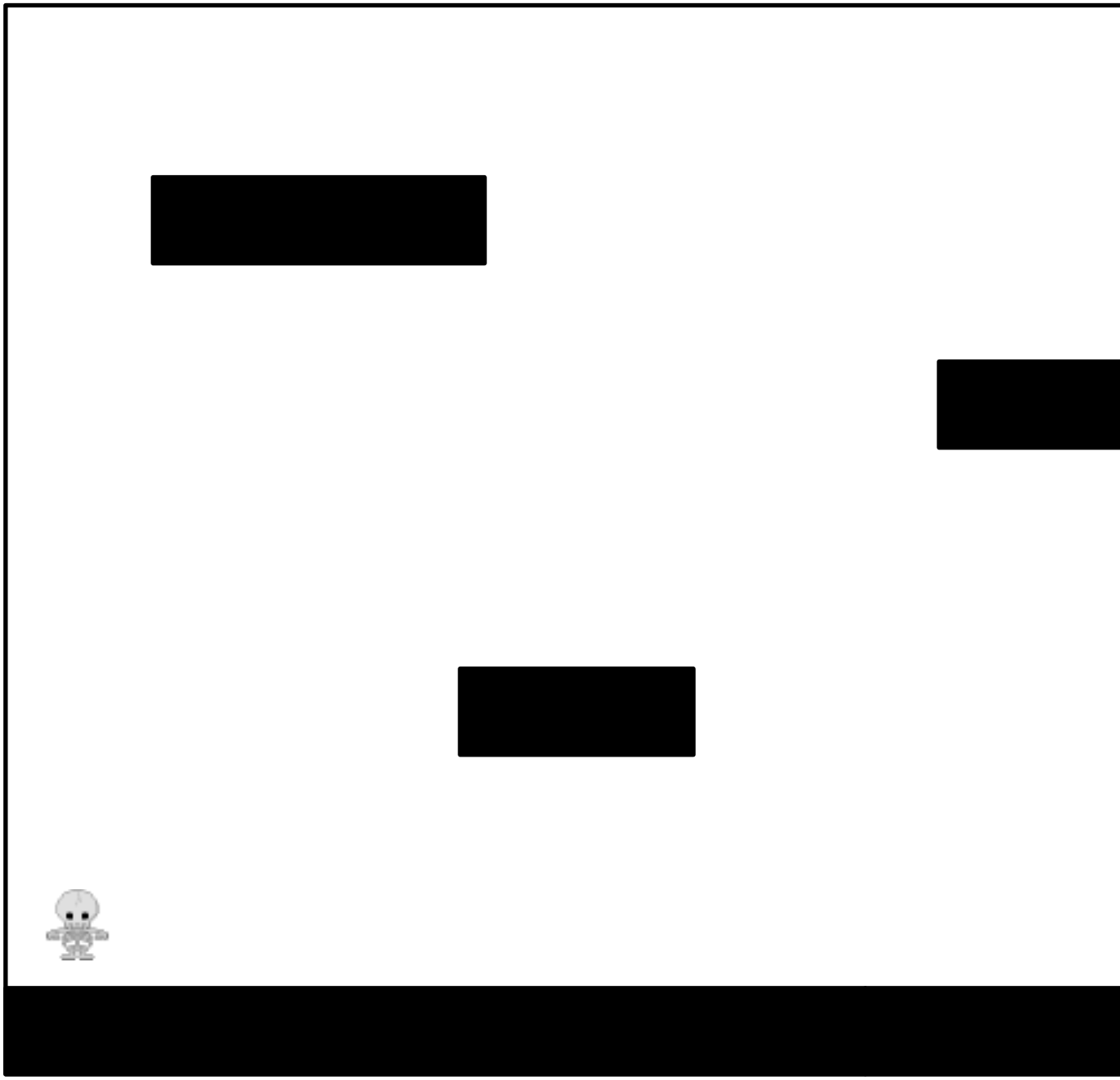


Gambar 3.25. **STORYBOARD level 2**

Pada *level* 2 ini, tingkat kesulitan terhadap musuh akan menjadi lebih tinggi, Musuh yang keluar pada *level* 2 ini adalah musuh dengan kemampuan memiliki kecepatan yang lebih tinggi dibanding dengan musuh di *level* 1 dan mempunyai health yang lebih banyak. Setiap pemaindapat menggunakan item yangberbentukpotion untuk menambahkan bar healthpada hero apabila pemain

yang terkena serangan atau tembakan dari monster tertentu. Item pada *level* 2 juga akan lebih sulit didapatkan dibandingkan dengan itemyang terdapat pada *level*1permainnan hanya bisa melanjutkan ke *level* berikutnya apabila semua monster telah dikalahkan dan semua item telah didapatkan.

STORYBOARD level 3 dari aplikasi *game The Legend Of Cursed Amulet*, terdapat pada gambar 3.26.



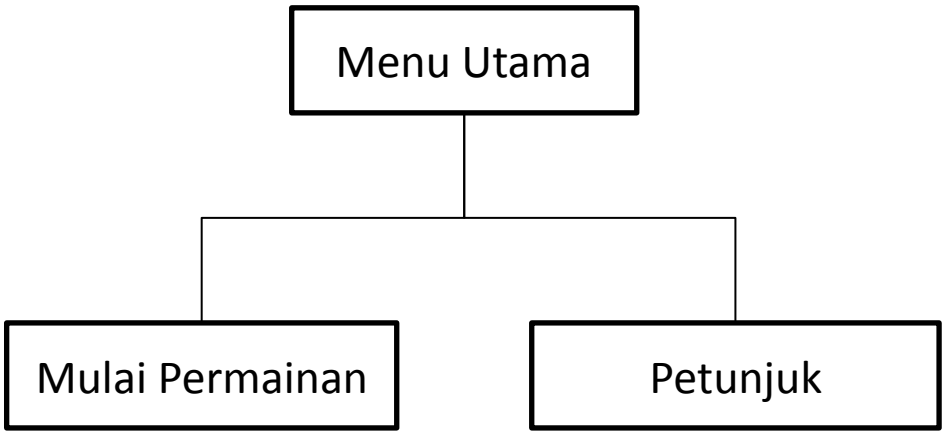
Gambar 3.26. **STORYBOARD level 3**

Pada *level* 3 ini, hero akan bertarung melawan musuh yang lebih sulit(boss), musuh diberi kemampuan untuk bergerak dan menyerang lebih cepat, untuk mengantisivasi serangan hero dapat mengambil item health yang terdapat pada platform tertentu sebagai item yang berguna untuk membantu mengalahkan musuh. Pada *level* ini musuh sudah diterapkan algoritma A* (*A star*) untuk

mencari rute terpendek untuk menuju ke tempat dimana hero akan mendekati musuh.

3.3.2. Perancangan Struktur Menu

Untuk mempermudah dalam pembuatan sistem, dibutuhkan perancangan struktur menu dari program uang akan di bangun. Perancangan struktur menu pada program ini membantu dalam merancang bagian-bagian dari sistem yang sebenarnya dan untuk mengetahui bahwa bagian mana yang akan di akses terlebih dahulu setelah program selesai dibuat. Perancangan struktur menu pada aplikasi permainan *The Legend Of Cursed Amulet* dapat dilihat pada gambar 3.27.

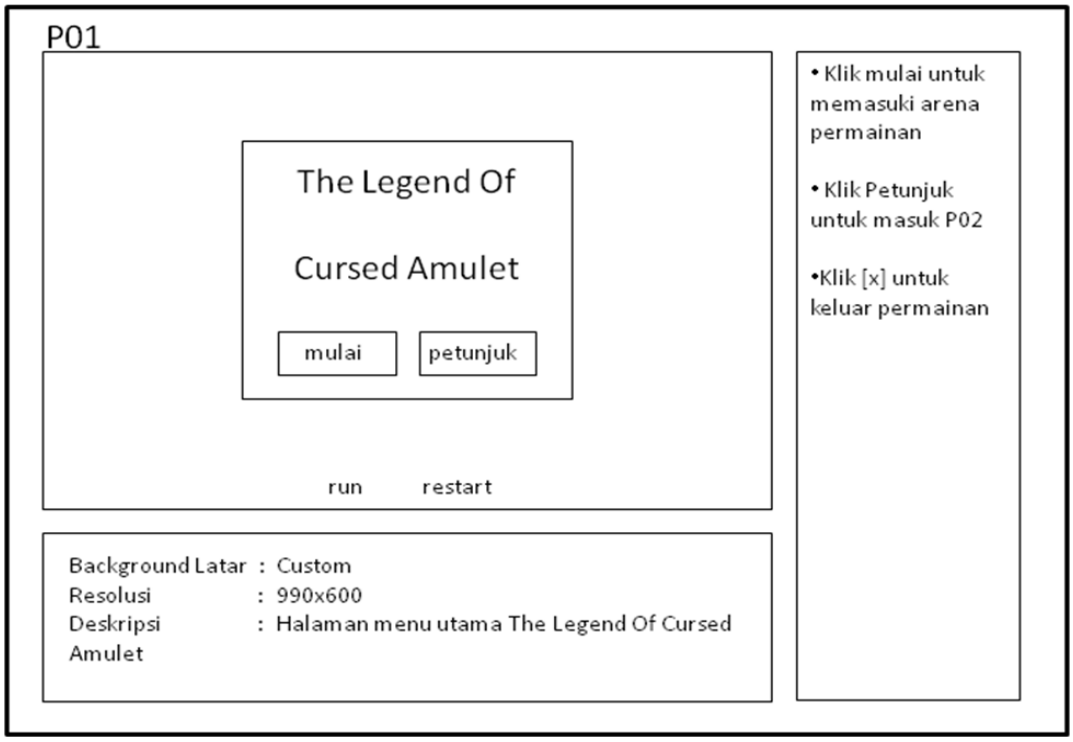


Gambar 3.27. **Struktur Menu The Legend Of Cursed Amulet**

A. Mulai Permainan : merupakan submenu untuk memulai permainan baru
B. Petunjuk : merupakan submenu untuk menampilkan informasi tentang cara memainkan aplikasi permainan *The Legend Of Cursed Amulet*

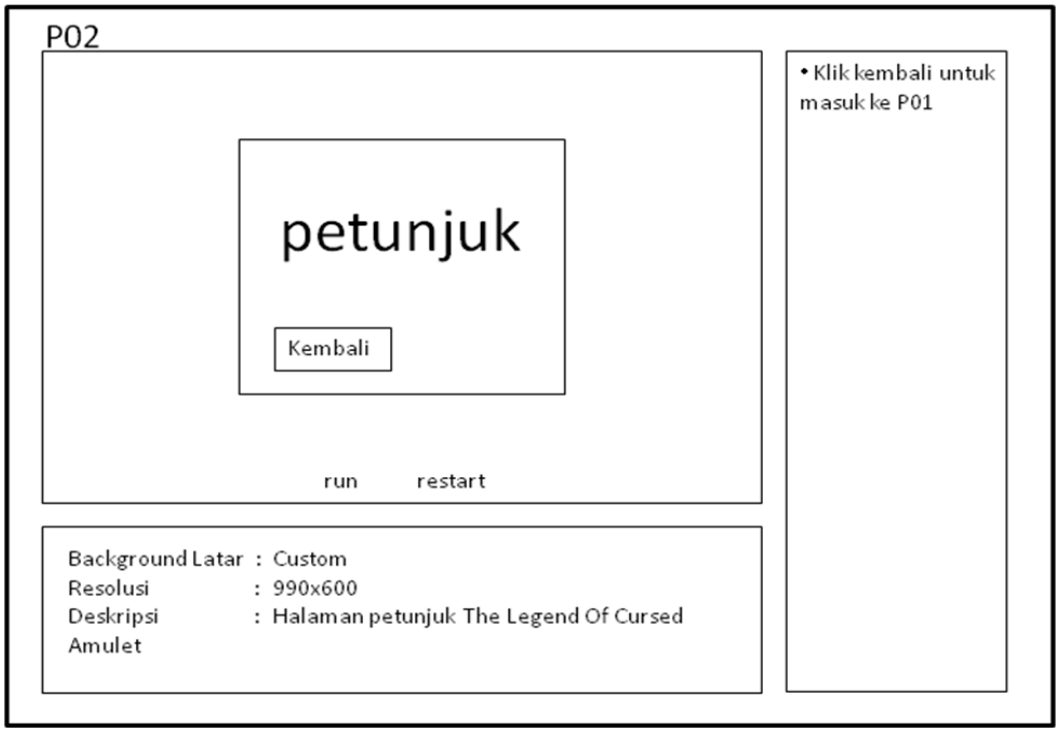
3.3.3. Perancangan Antar Muka

Perancangan antarmuka menu utama pada aplikasi permainan *The Legend Of Cursed Amulet* dapat dilihat pada gambar 3.28.



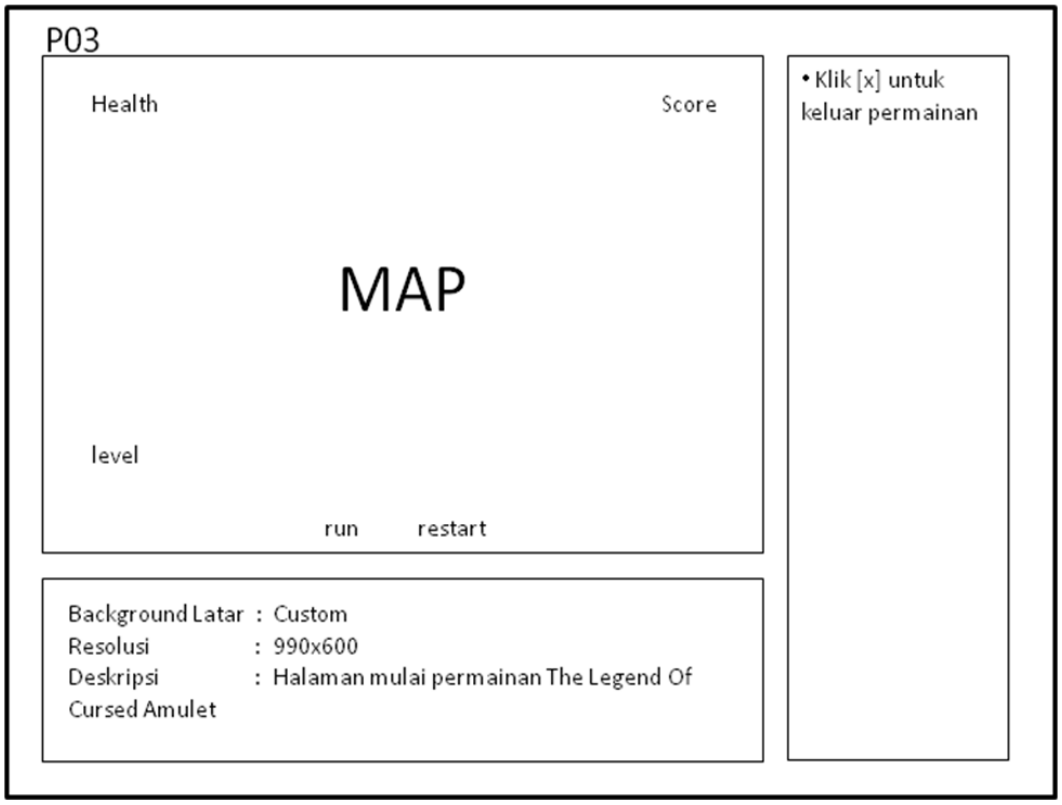
Gambar 3.28. **Perancangan Antarmuka Menu Utama *The Legend Of Cursed Amulet***

Perancangan antarmuka masuk petunjuk pada aplikasi permainan *The Legend Of Cursed Amulet* dapat dilihat pada gambar 3.29.



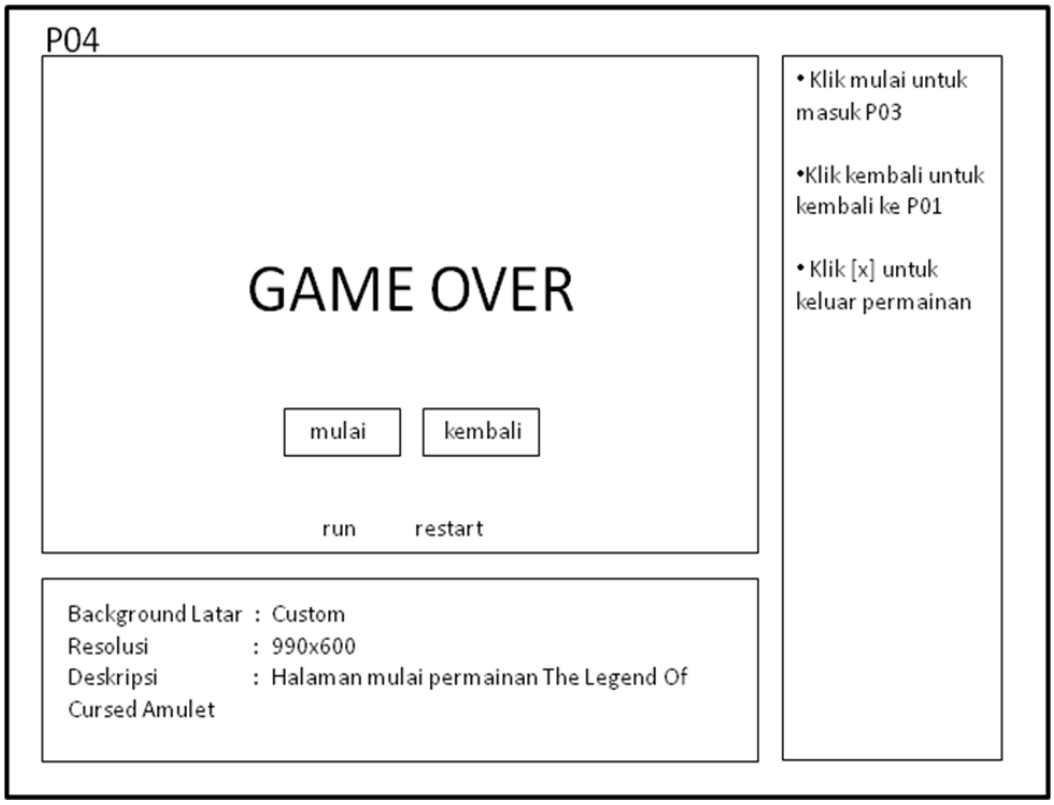
Gambar 3.29. **Perancangan Antarmuka Petunjuk *The Legend Of Cursed Amulet***

Perancangan antarmuka mulai permainan pada aplikasi permainan *The Legend Of Cursed Amulet* dapat dilihat pada gambar 3.30.



Gambar 3.30. **Perancangan Antarmuka lihat petunjuk *The Legend Of Cursed Amulet***

Perancangan antarmuka permainan berakhir pada aplikasi permainan *The Legend Of Cursed Amulet* dapat dilihat pada gambar 3.31.

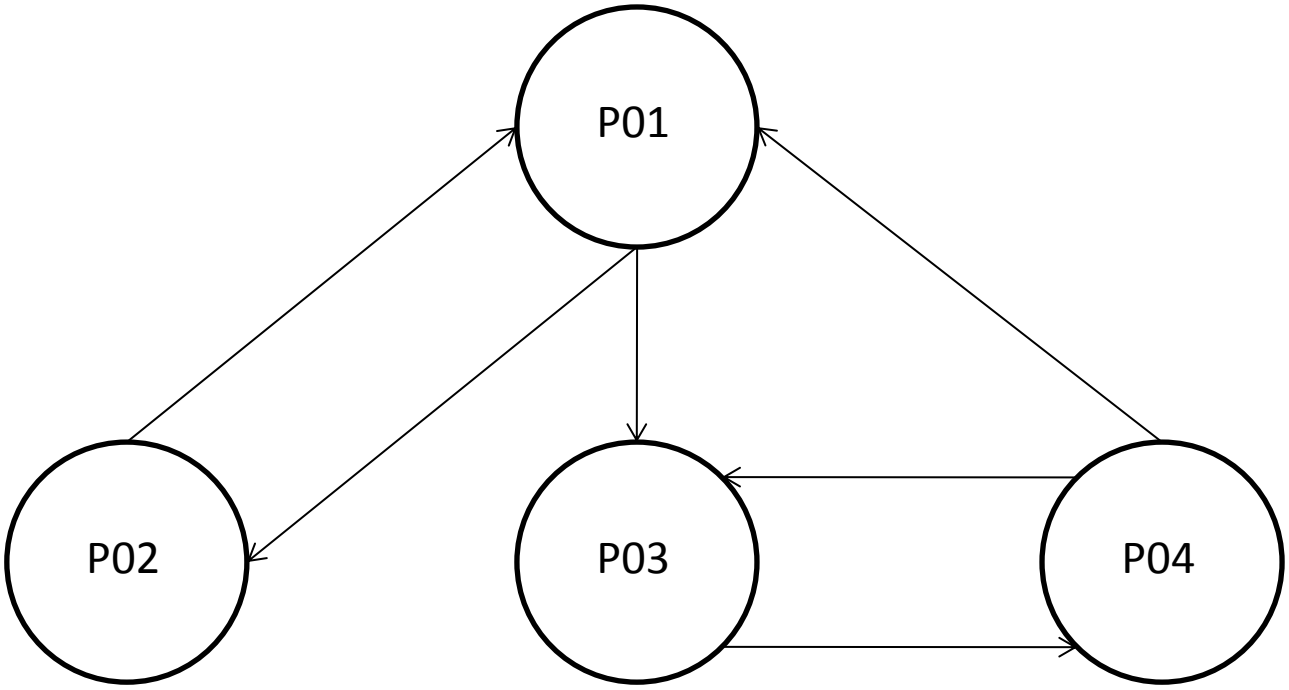


Gambar 3.31. **Perancangan Antarmuka Permainan Berakhir *The Legend Of Cursed Amulet***

3.3.4. Jaringan Semantik

Jaringan Semantik merupakan gambaran pengetahuan grafis yang menunjukkan hubungan antar objek (lingkaran) dan garis yang menggambarkan formasi antar objek tersebut.

Jaringan semantik pada aplikasi permainan *The Legend Of Cursed Amulet* dapat dilihat pada gambar 3.32.



Gambar 3.32. **Jaringan Semantik The Legend Of Cursed Amulet**

3.3.5. 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 *The Legend Of Cursed Amulet*. Adapun *method-method* yang terdapat dalam aplikasi permainan *The Legend Of Cursed Amulet* terdiri dari dua buah jenis *method,* yaitu *method* yang terdapat dalam *tools*pembangun (hanya digunakan sesuai dengan kebutuhan) dan *method* yang dirancang sesuai kebutuhan aplikasi.

3.3.5.1. *Method* yang tersedia di dalam *tools* pembangun

Method yang digunakan dari *tools* pembangun aplikasi dalam pembangunan aplikasi permainan *The Legend Of Cursed Amulet terdapat pada tabel* 3.27.

Tabel 3.27. Penggunaan *method* yang terdapat didalam *tools* pembangun

Music_Player		
No	**Nama Method**	**Keterangan**
1	MusicPlayer()	Sebagai media untuk menjalankan musik
2	loadMusic()	Mengambil musik yang terdapat pada directory
3	playMusic()	Memutar musik yang yang tersedia di directory
4	isStopped()	Stop pemutar musik
5	repeating()	Memutar kembali musik dari awal
6	Replay()	Memutar musik berkali-kali
myWorld		
7	myWorld()	Membuat world dengan ditentukan panjang dan lebar
8	getWorld()	Memanggil world yang sudah ditentukan
9	gamePaint()	Menempatkan background
10	gameRender()	menyesuaikan tampilan background
11	gameUpdate()	Update setting background
12	testKeyPressed()	Memeriksa tombol yang akan digunakan

13	checkGameOver()	Mengecek status permainan
14	changeState()	Mengubah state dalampermainan
15	getPathResolution()	Memanggil resolusi dari rute permainan
16	checkCollition()	Mengecek tabrakan terhadap objek
Sprite		
17	Sprite()	Menginisialisasi sprite
18	setImage()	Menempatkan gambar
19	loopImage()	Mengulang pergantian gambar
20	setDX(), setDY()	Inisialisasi kordinat DX, DY
21	getDX(), getDY()	Memanggil kordinat DX, DY
NPC		
22	runAStar()	Mengimplementasikan Astar pada objek
23	hasFoundPlayer()	Menempatkan kondisi dari objek terhadap player
24	atEdge()	Menggambarkan tepian dari objek
25	toString()	Mengubah ke bentuk string
Animation		
26	Animation()	Menginisialisasi animasi
27	setAnimImageFromFile()	Menempatkan animasi yang terdapat di dalam file
28	updateImage()	Memperbaharui gambar
29	getCurrentImage()	Mengambil gambar yang akan digunakan saat ini
30	resume()	Menjalankan kembali kondisi yang sedang berhenti
31	tileMap()	Inisialisasi Tilemap
32	getMapHead()	Menetapkan tepi atas dari map
33	getWidth()	Menetapkan lebar dari objek
34	getHeight()	Menetapkan ketinggian dari objek
35	getTile(), setTile()	Mengambil dan menempatkan tile
36	tileToPixel()	Mengubah tile menjadi pixel
37	pixelToTile()	Mengubah pixel menjadi tile
38	drawTileMap()	Menggambarkan objek tilemap
39	drawMim()	Menggambarkan mini map

GameWorldManager		
40	GameWorldManager()	Menampilkan game world
41	setMoveFlag()	Menempatkan point pergerakan objek
42	getTileCollision()	menetapkan tabrakan antara gambar dengan tile
43	worldCollitionHorizontal()	Tabrakan terhadap objek lebar
44	worldCollitionVertical()	Tabrakan terhadap objek tinggi
45	SpriteCollision()	Tabrakan antara objek sprite
46	getOffsetX(), getOffsetY	Mengambil lokasi pixel dari sprite
47	getSprite()	Mengambil objek sprite
48	isScrollRight(), isScrollLeft()	Scrolling ke kanan, ke kiri
Player		
49	drawHeart()	Menggambarkan objek heart
50	resetCount()	Mereset perhitungan
51	thisPosition()	Posisi saat ini
52	nextPosition()	Posisi berikutnya
AStarPathFinder		
53	AStarPathFinder()	Menginisialisasi rute terpendek dari method Astar
54	findPath()	Mencari rute
55	getfFirstInOpen()	Mengambil rute list open
56	addToOpen()	Memasukan ke node open
57	removeFromOpen()	Hilangkan dari list open
58	addToClosed()	Memasukan ke node close
59	removeFromClosed()	Hilangkan dari list close
60	isValidLocation()	Berada di lokasi yang benar
61	getMovmenCost()	Mengambil nilai jauh pergerakan
62	getHeuristicCost()	Mengambil nilai heuristik
ClosestHeuristic		
63	getCost()	Mendapatkan nilai heuristik
Path		
64	path()	Membuat path kosong

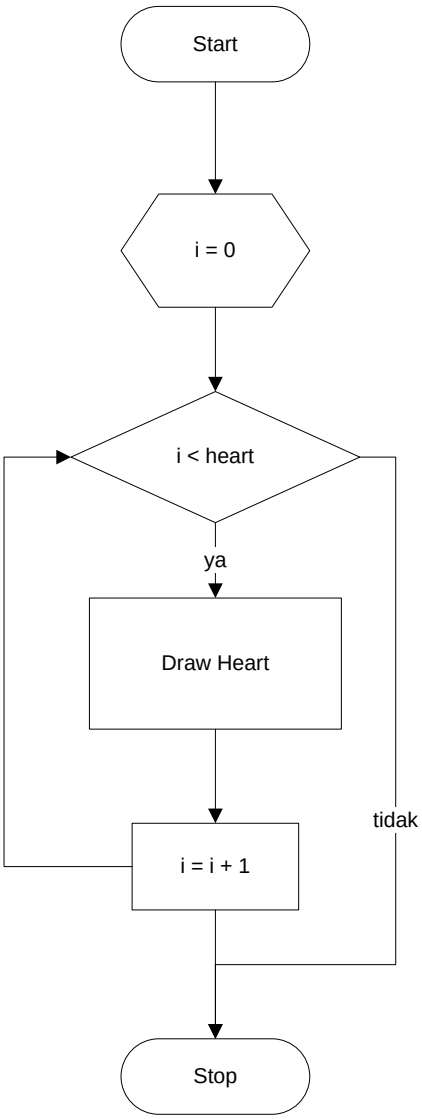
65	getLenght()	Mengambil panjang path
66	getStep()	Mengambil step yang di berikan oleh path
67	appendStep()	Menambahkan step untuk path
68	preppendStep()	Mempersiapkan step untuk path
69	Contains()	Cek jika di dalam path terdapat step

3.3.5.2. Perancangan *method* yang akan digunakan di dalam game

Method yang digunakan di dalam perancangan berdasarkan kebutuhan padaaplikasi permainan *The Legend Of Cursed Amulet*adalah sebagai berikut.

A. *Method* drawHeart()

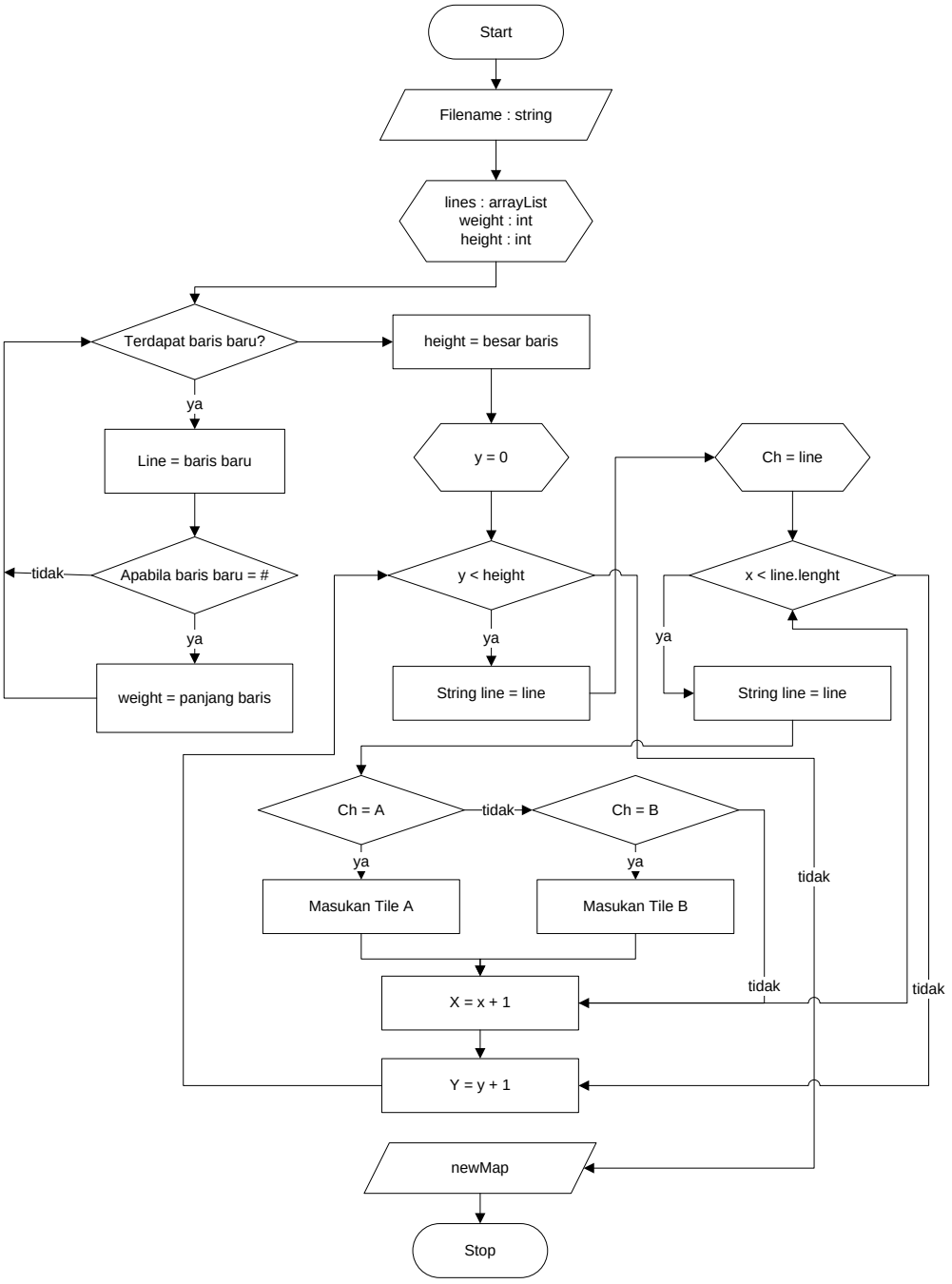
Method drawHeart() akan dipanggil ketika menampilkan *health* pemain, dapat dilihat pada gambar3.33.



Gambar 3.33. ***Flowchart Method*** **drawHeart()**

B. *Method* loadTileMap()

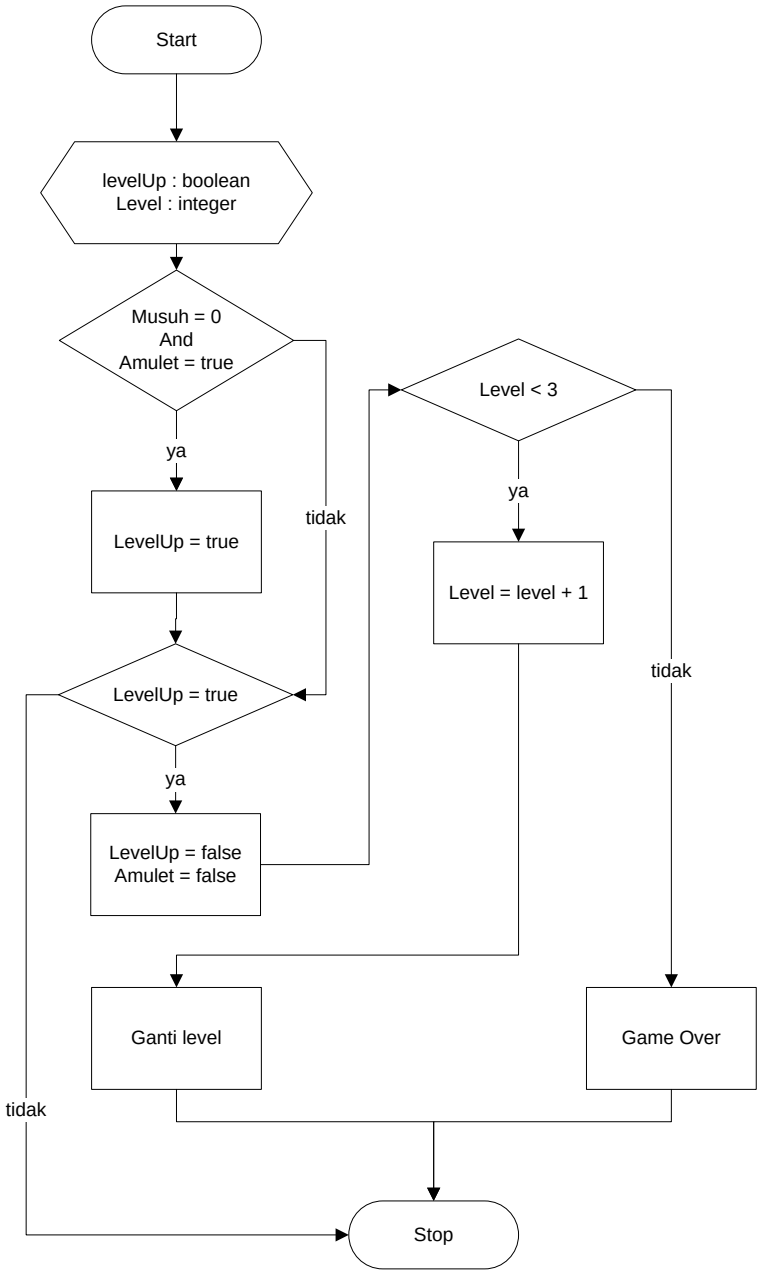
Method loadTileMa() akan dipanggil ketika pemain memasuki arena permainan, dapat dilihat pada gambar 3.34.



Gambar 3.34. ***FlowchartMethod*loadTileMa()**

C. *Method* loadLevel()

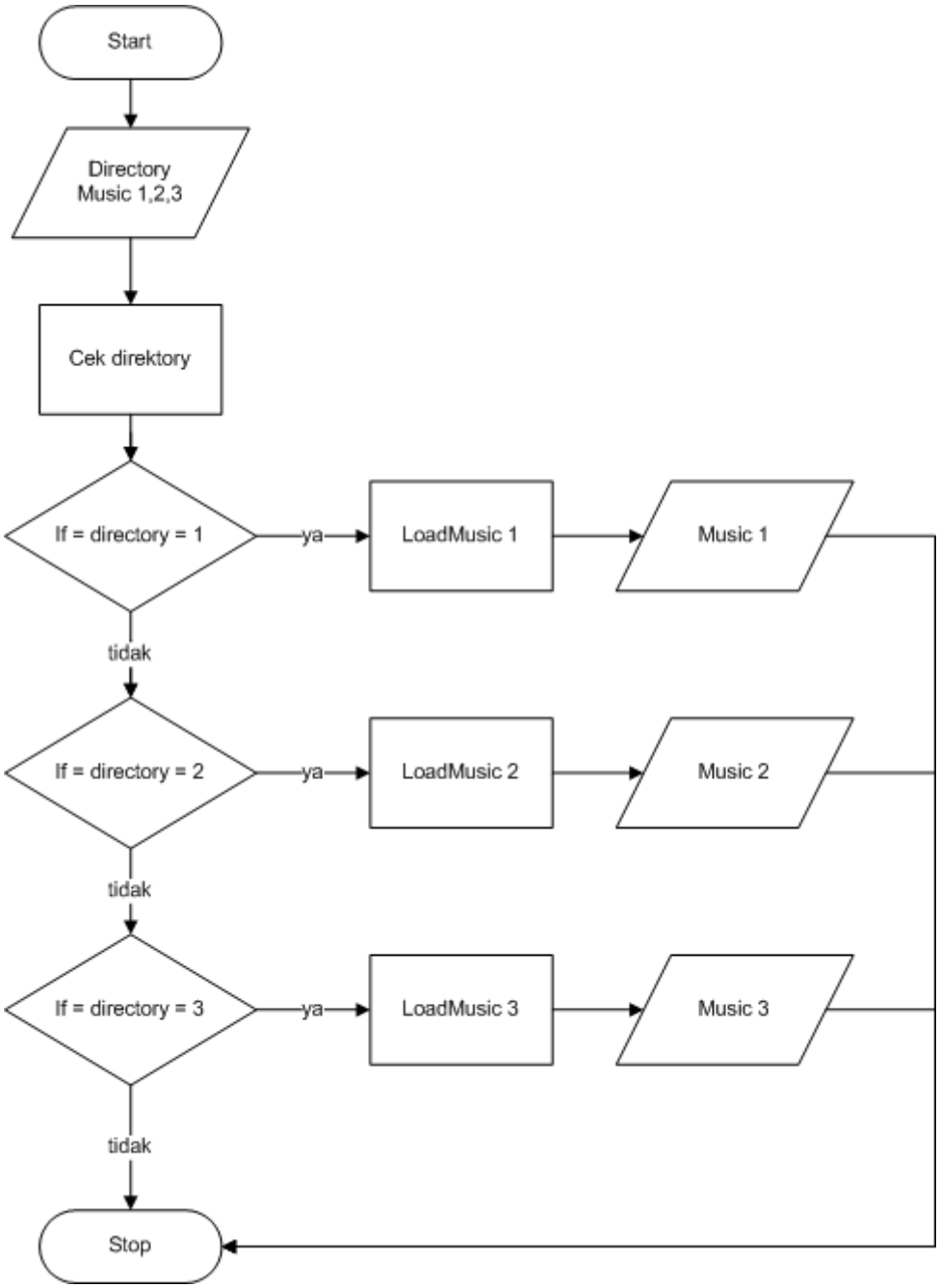
Method loadLevel() akan dipanggil ketika pemain memasuki arena permainan, dapat dilihat pada gambar 3.35.



Gambar 3.35. ***FlowchartMethod* loadLevel()**

D. *Method* loadMusic()

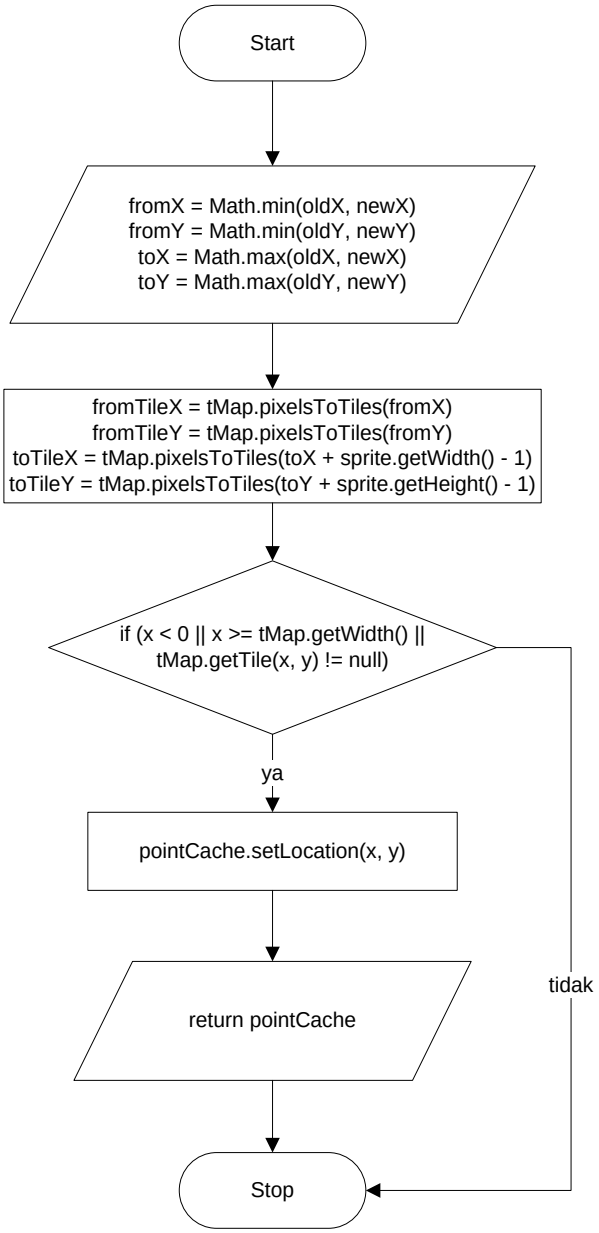
Method loadMusic() akan dipanggil ketika pemain memulai permainan, dapat dilihat pada gambar 3.36.



Gambar 3.36. ***Flowchart Method* loadMusic()**

E. *Method* getTile*Collision*()

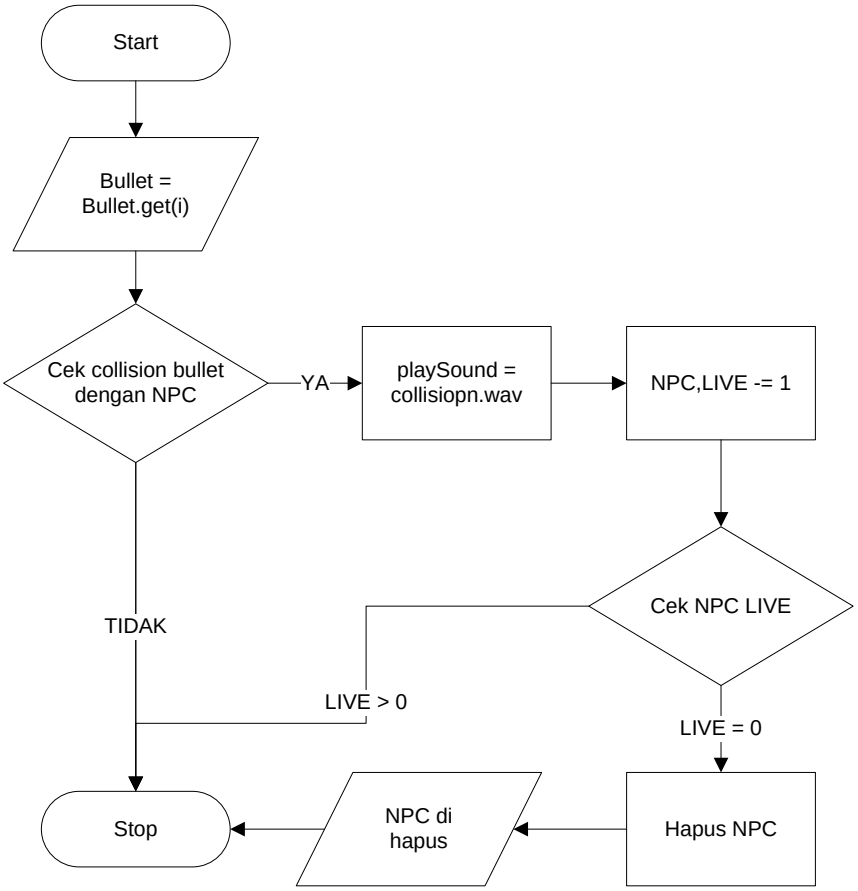
Method getTileColision() akan dipanggil di ketika pemain memasuki arena permainan, *method* ini terjadi apabila terjadi *collision* antara objek dengan objek, dapat dilihat pada gambar 3.37.



Gambar 3.37. ***Flowchart Method* getTileCollision()**

F. *Method* checkShooting()

Method checkShooting() akan dipanggil di ketika terjadi *collision* antara peluru dengan NPC, dapat dilihat pada gambar 3.38.



Gambar 3.38. ***Flowchar tMethod* checkShooting()**