BAB III
ANALISIS DAN PERANCANGAN SISTEM

III.1 Analisis Sistem

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

III.1.1 Analisis *Game* Sejenis

III.1.1.1 *Super Mario Bros*

Super Mario Bros adalah sebuah *game* bertipe *platform* yang sangat terkenal dan bahkan menjadi ikon dari perusahan Nintendo. Dengan *setting* di *Mushroom Kingdom*, pemain akan berperan sebagai Mario dalam sebuah petualangan menyelamatkan putri dari kura – kura raksasa bernama Bowser. Pada *game* ini terdapat banyak musuh serta beberapa item yang bervariatif[14].

Game "*Super Mario Bros"* memiliki beberapa kelebihan diantaranya :

1. Layar dan *background* bergerak sesuai posisi Mario. *Game* pada saat itu hanya mempunyai *single screen*, sehingga *gameplay* pada umumnya berpindah *screen.*
2. Item yang dapat diambil oleh pemain cukup bervariatif, seperti koin dan berbagai macam jamur yang memiliki kegunaannya masing-masing.
3. Adanya *timer* membuat pemain harus secepat mungkin menyelesaikan permainan karena akan dihitung sebagai poin.

Sedangkan kekurangan yang terdapat dalam *game* ini diantaranya :

1. Tampilan secara keseluruhan masih berbentuk blok-blok sehingga membuat *game* tersebut terlihat relatif kaku.
2. Suara-suara yang terdapat dalam *game* berformat midi sehingga terdengar kurang realistis.

3. Gerakan Mario kurang bervariasi, hanya berjalan serta melompat yang dapat digunakan untuk mencapai *platform* yang lebih tinggi dan juga digunakan untuk mengalahkan musuh.
4. Tidak adanya *reward* setelah permainan berakhir, seperti misalnya pengaturan *high score*.
5. Tidak terdapat pesan ataupun pelajaran yang dapat diambil.[17]

III.1.1.2 *Sonic The Hedgehog*

Sonic The Hedgehog merupakan video *game* dengan penjualan terbaik yang dirilis oleh Sega. *Game* ini bertipe *platform* dengan karakter utama bernama Sonic, seekor landak berwarna biru yang berusaha untuk menyelamatkan dunia dari berbagai ancaman, terutama musuh besarnya Dr. Ivo "*Eggman*" Robotnik. Tujuan utama musuh besarnya itu adalah untuk menguasai dunia, sehingga selalu berusaha untuk mengalahkan Sonic dan mendapatkan *Chaos Emeralds.*[15]

Game "*Sonic The Hedgehog"* memiliki beberapa kelebihan diantaranya :

1. Tampilan secara keseluruhan lebih baik dibandingkan *Super Mario Bros.*
2. Mempunyai gerakan yang lebih bervariasi seperti berlari dan melompat.
3. Adanya perlakuan khusus terhadap item *golden ring* yang telah diambil, Sonic dapat kehilangan *ring*-nya jika terkena musuh maupun objek yang dapat membuat Sonic terluka. Namun, *golden ring* tersebut dapat kembali diambil dalam waktu tertentu sebelum kembali menghilang.
4. Adanya *reward* bagi pemain, contohnya jika berhasil mengumpulkan 100 *ring* pemain akan diberikan *extra life.*

Sedangkan kekurangan yang terdapat dalam *game* ini adalah :

1. Suara – suara yang terdapat dalam *game* berformat midi sehingga terdengar kurang realistis.
2. Tidak adanya *reward* setelah permainan berakhir, seperti misalnya pengaturan *high score*.

3. Tidak terdapat pesan ataupun pelajaran yang dapat diambil.[16]

III.1.1.3 *Dasar Jana: Mengejar Dewi Suda*

Game yang akan dibuat berjudul "*Dasar Jana: Mengejar Dewi Suda"* memiliki beberapa kesamaan dengan game *"Super Mario Bros"* dan "*Sonic The Hedgehog"* sebagai berikut :

1. Menggunakan konsep *side scrolling* 2D.
2. *Game* bertipe *platform* dengan *genre action, adventure*.

Sedangkan penambahan pada *game* yang akan dibuat adalah sebagai berikut :

1. Perbaikan pada sisi *sound*, menggunakan efek suara yang lebih realistis dengan format .wav dan musik latar dengan format .mp3.
2. Memberikan kecerdasan tertentu pada semua NPC (*Non-Playable Character)*.
3. Adanya *reward* kepada pemain dalam bentuk *high score* dengan perhitungan poin yang didapat dari mengumpulkan item dan mengalahkan musuh.
4. Menyisipkan komik dan pesan – pesan motivasi pada *game* dalam bentuk *cut scenes*.

Hasil analisa perbandingan ketiga *game* yang telah dibahas sebelumnya dapat dilihat pada Tabel III.1.

Tabel III.1 Analisa Perbandingan Game Sejenis

Poin Pembanding	*Super Mario Bros*	*Sonic The Hedgehog*	*Dasar Jana*
Graphics	16 bit	16 bit	16 bit
Sound	.midi	.midi	.wav, .mp3
Cut Scenes	Tidak Ada	Tidak Ada	Ada
Pengaturan *High Score*	Tidak Ada	Tidak Ada	Ada
Batasan usia	Tidak Ada	Tidak Ada	Di atas 10 tahun
Pesan positif	Implisit	Implisit	Eksplisit

III.1.2 Analisis Game yang Akan Dikembangkan

III.1.2.1 *Storyline*

Suatu hari, diceritakan seorang mahasiswa tingkat akhir bernama Jana berkuliah di *Miracle Institute Technology* yang berada di kota Bandung. Ia

bersama mahasiswa lainnya merasa tegang karena pada saat itu akan dilakukan *review* terhadap proposal tugas akhir yang diberikan. Meskipun begitu, Jana tetap merasa optimis bisa melewatinya karena selama berkuliah hingga tahap ini semua masalah perkuliahan dapat dihadapi dengan lancar.

Namun takdir berkata lain, disaat seharusnya ia lulus kuliah dengan lancar justru mengalami hal yang tak terduga, proposal yang diajukannya untuk tugas akhir ternyata ditolak. Ia pun kebingungan karena tidak mempunyai proposal lain untuk diajukan pada kesempatan kedua. Lalu, Jana pulang ke rumah dengan perasaan yang tidak menentu dan merasa ini merupakan saat-saat terburuk yang pernah dialaminya.

Jana yang lelah pun tertidur dengan keadaan yang sangat buruk. Tiba-tiba dalam tidurnya ia bermimpi bertemu dengan sesosok wanita yang sangat cantik dan langsung membuatnya jatuh hati. Wanita tersebut belakangan diketahui bernama Dewi Suda. Mereka berjalan dan bercanda bersama diantara bunga-bunga yang sedang mekar. Tidak lama kemudian, muncul mahluk misterius berusaha menculik wanita tersebut. Sang dewi pun akhirnya berhasil diculik, namun ia sempat berteriak kepada Jana untuk minta diselamatkan. Jana yang merasa kesal lalu bertekad untuk mengejar cintanya.

III.1.2.2 Analisis Level

Game Dasar Jana memiliki tiga buah level dengan aturan level kedua dapat dimainkan jika level pertama telah berhasil diselesaikan, begitu juga dengan level ketiga yang dapat dimainkan jika level kedua telah berhasil diselesaikan oleh *user*. Sedangkan jika suatu level gagal diselesaikan dan *user* memilih untuk melanjutkan permainan, permainan akan diulang kembali pada level dimana *user* gagal menyelesaikannya. Perbedaan yang terdapat pada setiap level diantaranya variasi *enemy* yang dihadapi, *background*, *sound* serta untuk memperkuat cerita dibalik *game* akan disisipkan komik pada saat sebelum dan setelah suatu level dimainkan oleh *user*.

III.1.2.3 *Gameplay*

Misi atau tugas utama *user* dalam *game* Dasar Jana adalah berusaha mengejar serta menyelamatkan Dewi Suda dengan cara mencapai akhir dari tiap level. Akhir dari setiap level dapat dicapai dengan cara mengalahkan bos musuh lalu menemukan portal untuk berpindah level. Khusus untuk level ketiga akan dimunculkan karakter Dewi Suda di dekat portal.

Dalam perjalanannya, *user* akan dihadapkan pada beberapa *obstacle* atau rintangan, dan *enemy* yang akan semakin sulit dikalahkan. Selain dengan mengalahkan *enemy, user* juga dapat memperbesar poin dengan cara mengumpulkan item dokumen yang tersebar di setiap level. Poin-poin yang didapatkan akan dikalkulasikan dengan *health point* kemudian dilakukan pengecekan pada akhir permainan apakah layak untuk dijadikan skor tertinggi (*high score*).

Terdapat dua buah interaksi yang dapat dilakukan terhadap *game,* yakni dengan menggunakan *mouse* untuk memilih menu dan tombol-tombol yang terdapat dalam *game* serta *keyboard* digunakan untuk menggerakkan karakter Jana. Kontrol yang dapat *user* gunakan untuk menggerakkan Jana dapat dilihat pada Tabel III.2.

Tabel III.2 Kontrol Gerakan Jana

Kontrol	Deskripsi
Arrow Right	Menggerakkan Jana ke sebelah kanan
Arrow Left	Menggerakkan Jana ke sebelah kiri
Arrow Up	Menggerakkan Jana untuk melompat
A	Menggerakkan Jana untuk menyerang dengan menggunakan tas
S	Menggerakkan Jana untuk menyerang dengan tendangan

III.1.2.4 *Scoring*

Scoring dibuat untuk memudahkan kalkulasi skor pada akhir permainan, terlebih lagi dengan adanya *reward* bagi *user* yang memperoleh skor tertinggi sehingga memudahkan sistem untuk melakukan pengecekan skor tertinggi. Pengaturan skor di dalam game dibagi ke dalam tiga kelompok sebagai berikut :

1. Perolehan Skor dari Item

User dapat memperoleh poin dari hasil mendapatkan item dokumen. Penambahan skor yang di dapatkan dari item dokumen adalah sebesar 10 poin. Penjelasan lebih detail mengenai item dapat dilihat pada Tabel III.4.

2. Perolehan Skor dari *Enemy*

User dapat memperoleh poin dari hasil mengalahkan *enemy* (untuk selanjutnya disebut *defeat point*). Penambahan skor yang di dapatkan dari *enemy* memiliki besar poin yang bervariasi bergantung pada masing-masing *enemy.* Penjelasan lebih detail mengenai *enemy* dapat dilihat pada Tabel III.3.

3. Perolehan Skor dari *Health Point*

Health point disini merupakan nyawa yang dimiliki karakter yang dimainkan oleh *user.* Pada awal permainan, *user* akan diberikan *health point* maksimal, yakni sebesar 100 poin. Pada akhir permainan, *health point* yang tersisa kemudian dikalikan dengan 40% lalu dibulatkan ke atas dengan fungsi *ceil* untuk dijadikan skor.

Dari perolehan ketiga skor di atas, seluruhnya kemudian dijumlahkan untuk mendapatkan skor akhir. Selanjutnya akan dilakukan pengecekan terhadap skor akhir (membandingkan skor yang diperoleh saat ini dengan skor yang diperoleh sebelumnya baik dengan *user* itu sendiri maupun dengan *user* lainnya) dan menetukan apakah skor tersebut layak berada pada *list* skor tertinggi (terbatas hanya lima *username* dengan skor tertinggi) yang terdapat pada menu skor tertinggi.

III.1.2.5 *Game Object*

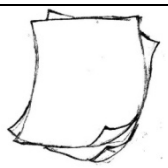
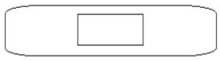
Objek-objek yang terdapat pada aplikasi *game* akan dibangun ini terdiri dari karakter, item, dan rintangan (*obstacle).* Penjelasan masing – masing objek dapat dilihat pada Tabel III.3, III.4, dan III.5.

Tabel III.3 Penjelasan Kemampuan Setiap Karakter pada *Game*

Karakter	Kemampuan	Keterangan
 Jana	*Health Point* : 100 *Attack Point* • *Hit/menyerang dengan menggunakan tas* : 15 • *Kick*/Tendang : 10	Jana merupakan *karakter* yang dapat dimainkan di dalam *game*, memiliki kemampuan untuk berjalan, melompat serta menyerang dengan menggunakan tas dan juga tendangan.
 Tanma	*Health Point* : 20 *Attack Point* : 5 *Defeat Point* : 5	Tanma merupakan NPC dengan bentuk tanaman liar yang memiliki kemampuan untuk menyerang, mengejar, kabur serta patroli.
 Pusa	*Health Point* : 30 *Attack Point* : 10 *Defeat Point* : 10	Pusa merupakan NPC dengan perwujudan roh ataupun aura negatif yang yang memiliki kemampuan untuk menyerang, mengejar, kabur serta patroli.
 Anaj	*Health Point* : 50 *Attack Point* • *Hit* /Pukul : 15 • *Kick*/Tendang : 10 *Defeat Point* : 100	Anaj merupakan bos musuh pada level pertama. Anaj merupakan NPC dengan sosok sisi jahat dari Jana yang menyerang dengansenjata dan tendangan, mengejar, kabur serta patroli.
 Jingli	*Health Point* : 30 *Attack Point* : 15 *Defeat Point* : 15	Jingli merupakan NPC dengan bentuk anjing liar yang yang memiliki kemampuan untuk menyerang, mengejar, kabur serta patrol.
 Buli	*Health Point* : 30 *Attack Point* : 10 *Defeat Point* : 10	Buli merupakan NPC dengan bentuk burung liar yang yang memiliki kemampuan untuk menyerang, mengejar, kabur serta patrol.

Karakter	Kemampuan	Keterangan
 Preman	*Health Point* : 60 *Attack Point* • Pisau : 15 • *Kick*/Tendang : 10 *Defeat Point* : 150	Preman merupakan bos musuh pada level kedua. Preman merupakan NPC dengan sosok preman yang yang memiliki kemampuan untuk menyerang dengan senjata dan tendangan, mengejar, kabur serta patroli.
 Deadline	*Health Point* : 70 *Attack Point* • Laser : 20 • *Kick*/Tendang : 15 *Defeat Point* : 200	Dead Line merupakan bos musuh pada level ketiga. Deadline merupakan NPC dengan bentuk imajinatif yang memiliki kemampuan untuk menyerang dengan senjata dan tendangan, bertahan, mengejar, serta patroli.

Tabel III.4 Penjelasan Item yang Terdapat pada *Game*

Item	Deskripsi
	Sebuah objek berupa dokumen yang dapat dikumpulkan untuk menambah *score* sebesar 10 poin.
	Sebuah objek berupa obat merah yang dapat digunakan untuk menambah *Health Point* sebesar 15 poin.
	Sebuah objek berupa plester yang dapat digunakan untuk menambah *Health Point* sebesar 10 poin.

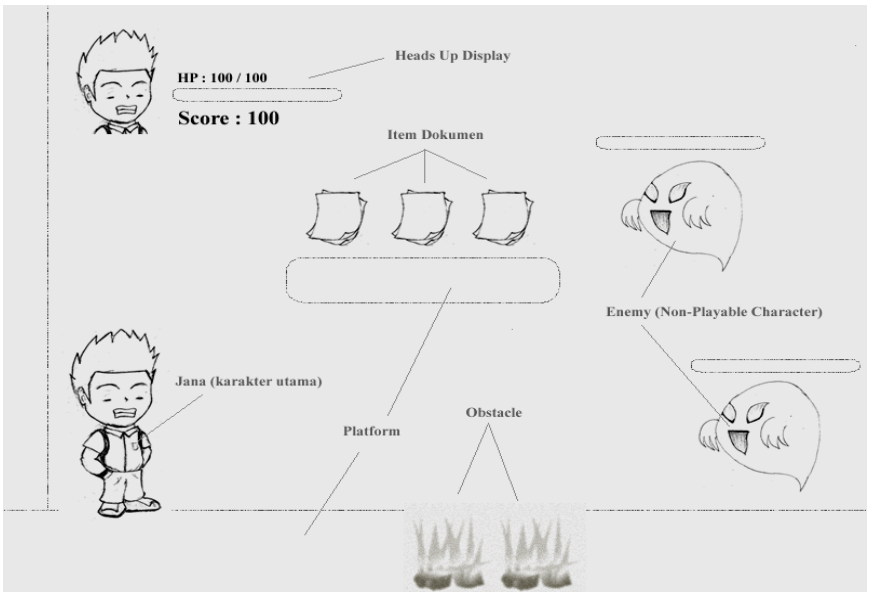
Tabel III.5 Penjelasan *Obstacle* yang Terdapat pada *Game*

Item	Deskripsi
	Sebuah objek berupa asap misterius yang jika terkena objek tersebut dapat mengurangi *health point* sebesar 5 poin.

Item	Deskripsi
	Sebuah objek berupa paku yang jika terkena objek tersebut dapat mengurangi *health point* sebesar 7 poin.
	Sebuah objek berupa duri yang jika terkena objek tersebut dapat mengurangi *health point* sebesar 10 poin.

III.1.2.6 *Storyboard*

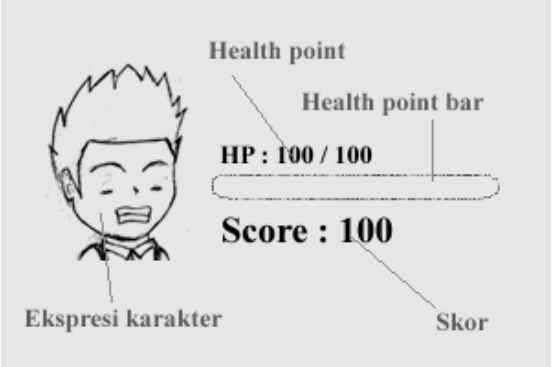
Storyboard adalah sketsa gambar yang disusun seperti halnya sebuah naskah cerita. Dengan *storyboard* ide – ide cerita dapat disampaikan kepada orang lain dengan cara yang lebih mudah, karena dapat membawa imajinasi seseorang menuju persepsi yang diinginkan melalui gambar-gambar yang telah dibuat. *Storyboard game "Dasar Jana: Mengejar Dewi Suda"* dapat dilihat pada Gambar III.1 sampai dengan Gambar III.4.



Gambar III.1 *Storyboard Game*

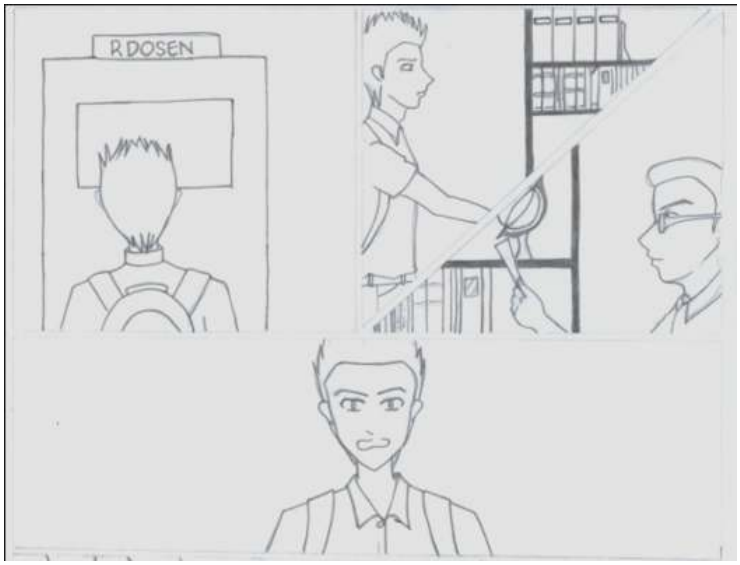
Heads Up Display (HUD) dibuat sebagai indikator untuk menggambarkan kondisi karakter, sesuai dengan namanya HUD berposisi di atas pada layar permainan. Pada *game "Dasar Jana: Mengejar Dewi Suda"* terdapat satu buah HUD untuk memperlihatkan tiga buah kondisi yakni, nyawa karakter yang

dimainkan pada properti HP (*Health Point)* baik dalam bentuk angka maupun dalam bentuk *bar*, skor yang diperoleh pada properti *score*, dan ekspresi wajah karakter yang berubah sesuai dengan kondisi nyawa yang dimilikinya.

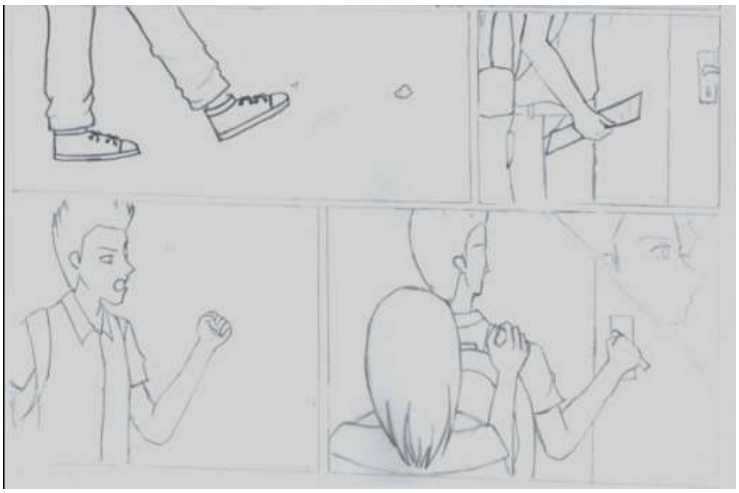


Gambar III.2 *Heads Up Display*

Cut scene dibuat untuk memperkuat cerita di dalam *game*, dapat berupa teks, gambar, maupun video. Pada *game "Dasar Jana: Mengejar Dewi Suda"* terdapat enam buah *cut scene* dalam bentuk komik yang masing – masing diposisikan baik sebelum maupun sesudah *user* memainkan suatu level.



Gambar III.3 Contoh Sketsa *Cut Scene* 1



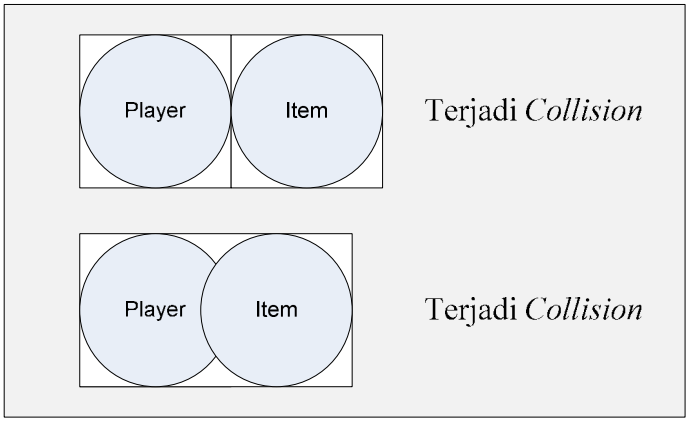
Gambar III.4 Contoh Sketsa *Cut Scene* 2

III.1.3 Analisis *Collision Detection*

Collision detection merupakan suatu metode yang digunakan untuk mendeteksi *collision* atau tabrakan antara suatu objek dengan objek yang lain. Pada pembangunan *game Dasar Jana: Mengejar Dewi Suda* terdapat beberapa metode untuk menangani masalah *collision detection,* yakni dengan memanfaatkan method yang terdapat di dalam *library* Flash dan juga dengan metode lain yang akan dijelaskan selanjutnya. Beberapa objek di dalam *game* yang akan ditangani *collision detection*-nya adalah sebagai berikut :

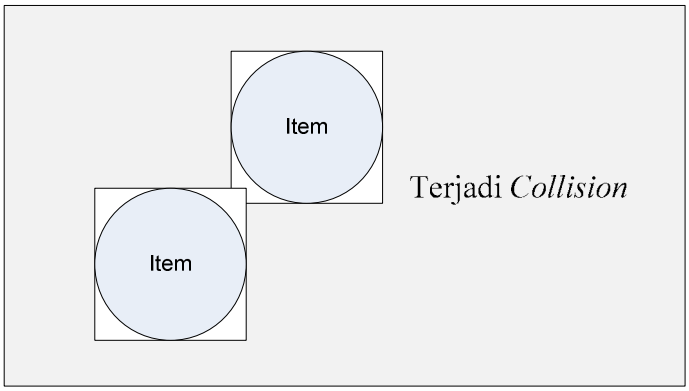
1. *Collision Detection Player* dengan *Item, Obstacle,* dan *Enemy*

Collision detection yang akan diterapkan untuk menangani *player* dengan *item, obstacle,* dan *enemy* adalah dengan menggunakan method yang terdapat di dalam *library* Flash, yakni hitTestObject. Pada dasarnya, method tersebut menggunakan metode *bounding boxes* dimana seluruh objek yang terdapat di dalam *stage* Flash dikelilingi oleh sebuah kotak yang tidak terlihat, sehingga ketika kotak-kotak tersebut bersentuhan atau beririsan akan terdeteksi sebagai *collision.* Ilustrasi *collision detection* dengan menggunakan method hitTestObject dapat dilihat pada Gambar III.5.



Gambar III.5 Ilustrasi *Collision Detection* dengan Menggunakan hitTestObject

Method hitTestObject memiliki beberapa kekurangan, salah satunya adalah apabila objek yang tidak berbentuk kotak yang sebenarnya tidak bersentuhan terdeteksi sebagai *collision* karena *bounding box* tersebut bersentuhan, contohnya dapat dilihat pada Gambar III.6.



Gambar III.6 Ilustrasi Kekurangan Method hitTestObject

Meskipun begitu, method hitTestObject tetap akan digunakan untuk menangani *collision detection* diantara *player* dengan *item,obstacle*, dan *enemy* (ketika dalam posisi menyerang) dikarenakan pergerakan objek di dalam *game* dengan kecepatan 30 *fps* (*frame per second*) cukup cepat dan celah diantara kedua *bounding box* menjadi kecil sehingga pemain tidak akan menyadari bahwa *collision detection* tidak begitu akurat.

2. *Collision Detection Player* dengan *Enemy*

Method hitTestObject mempunyai kekurangan yakni tidak bisa menentukan dimana letak terjadi *collision* misalnya tidak bisa menetukan apakah *player* berada di atas *enemy* sehingga tidak dapat menangani *blocking* antara *player* dan *enemy,* dibutuhkan suatu metode *collision detection* yang lain. Metode yang akan diterapkan pada *player* dengan *enemy* adalah dengan metode *axis-based collision detection.* Terdapat dua hal penting yang diperlukan metode ini untuk dapat bekerja, yakni jarak antara kedua objek dan juga berapa besar kedua objek beririsan[11].

Besar jarak dalam sumbu x yang dimisalkan dengan *dx,* didapatkan dari selisih nilai x objek A dari objek B, sehingga didapatkan persamaan :

$$dx = A.x - B.x \qquad (1)$$

Besar jarak dalam sumbu y yang dimisalkan dengan *dy,* didapatkan dari selisih nilai y objek B dari objek A, sehingga didapatkan persamaan :

$$dy = B.y - A.y \qquad (2)$$

Besar irisan dalam sumbu x yang dimisalkan dengan *ox,* didapatkan dari jumlah setengah lebar objek A ditambah setengah lebar objek B dikurangi nilai absolut dari selisih nilai y objek A dari objek B, sehingga didapatkan persamaan :

$$ox = A.lebar/2 + B.lebar/2 - |dx| \qquad (3)$$

Besar irisan dalam sumbu y yang dimisalkan dengan *ox,* didapatkan dari jumlah setengah tinggi objek A ditambah setengah tinggi objek B dikurangi nilai absolut dari selisih nilai y objek B dari objek A, sehingga didapatkan persamaan :

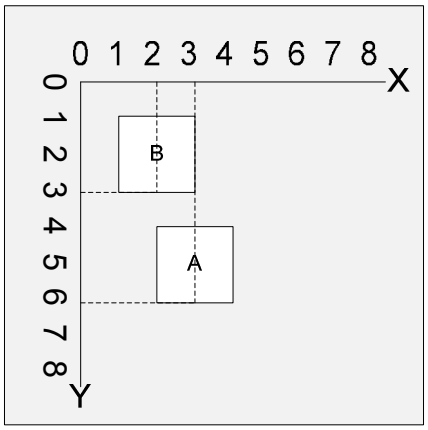
$$oy = A.tinggi/2 + B.tinggi/2 - |dy| \qquad (4)$$

Kemudian untuk menentukan apakah *collision* benar-benar terjadi dengan menggunakan metode *axis-based* adalah sebagai berikut :

1. Lakukan pengecekan apakah objek beririsan dalam sumbu x. Irisan dalam sumbu x terjadi apabila ox > 0.
2. Jika pengecekan pada langkah satu bernilai benar, lakukan pengecekan apakah objek beririsan dalam sumbu y. Irisan dalam sumbu y terjadi apabila oy > 0.

3. Jika pengecekan pada langkah satu bernilai benar, berarti kedua objek beririsan dalam kedua sumbu, maka *collision* benar terjadi. Kemudian untuk menentukan letak terjadinya *collision* dilakukan pengecekan :
   a. Jika $ox < oy$ bernilai benar, lakukan pengecekan jika $dx < 0$ bernilai benar maka *collision* terjadi pada sisi kanan, jika $dx < 0$ bernilai salah maka *collision* terjadi pada sisi kiri.
   b. Jika $ox < oy$ bernilai salah, lakukan pengecekan jika $dy < 0$ bernilai benar maka *collision* terjadi pada sisi atas, jika $dy < 0$ bernilai salah maka *collision* terjadi pada sisi bawah.

Contoh *collision detection* dengan menggunakan metode *axis-based* dapat dilihat dalam Gambar III.7 dan Gambar III.8.



Gambar III.7 Ilustrasi Pengecekan Dengan Metode *Axis-Based* 1

Pada Gambar III.7 diketahui dua buah objek, objek *player* dimisalkan dengan B memiliki *registration point* (2,3) dan objek *enemy* dimisalkan dengan A memiliki *registration point* (3,6) dimana sisi-sisi kedua objek memiliki lebar dan tinggi yang sama yakni 2. Lakukan pengecekan apakah terjadi *collision.*

Berdasarkan data diatas, maka selanjutnya adalah menentukan besar jarak dan besar irisan dalam kedua sumbu sebagai berikut :

$$dx = 3 - 2 = 1;$$

$$dy = 3 - 6 = -3;$$

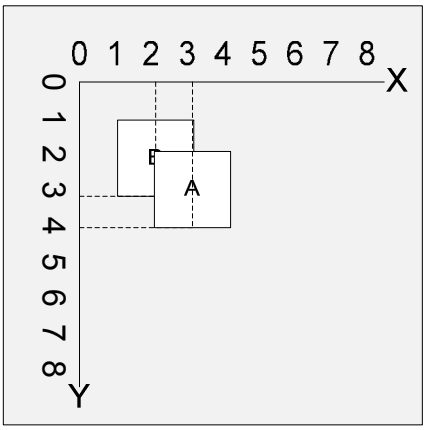
$$ox = 2/2 + 2/2 - |1| = 1;$$

$$oy = 2/2 + 2/2 - |-3| = -1$$

Kemudian dilakukan pengecekan berdasarkan langkah-langkah yang telah dijelaskan sebelumnya.

1. *ox* = 1, Apakah *ox* > 0? Benar, lanjutkan ke langkah kedua.
2. *oy* = -1, Apakah *oy* > 0? Salah, pengecekan berhenti.

Kesimpulan yang dapat diambil dari Gambar III.7 adalah tidak terdeteksi *collision* pada kedua objek.



Gambar III.8 Ilustrasi Pengecekan Dengan Metode *Axis-Based* 2

Pada Gambar III.8 diketahui dua buah objek, objek *player* dimisalkan dengan B memiliki *registration point* (2,3) dan objek *enemy* dimisalkan dengan A memiliki *registration point* (3,4) dimana sisi-sisi kedua objek memiliki lebar dan tinggi yang sama yakni 2. Lakukan pengecekan apakah terjadi *collision.*

Berdasarkan data diatas, maka selanjutnya adalah menentukan besar jarak dan besar irisan dalam kedua sumbu sebagai berikut :

$$dx = 3 - 2 = 1;$$
$$dy = 3 - 4 = -1;$$
$$ox = 2/2 + 2/2 - |1| = 1;$$
$$oy = 2/2 + 2/2 - |-1| = 1$$

Kemudian dilakukan pengecekan berdasarkan langkah-langkah yang telah dijelaskan sebelumnya.

1. *ox* = 1, Apakah *ox* > 0? Benar, lanjut ke langkah kedua.
2. *oy* = 1, Apakah *oy* > 0? Benar, lanjut ke langkah ketiga.
3. Apakah *ox* < *oy*? Salah, lakukan pengecekan ke langkah 3.b.

b. Apakah $dy < 0$? Benar. Pengecekan berhenti.

Kesimpulan yang dapat diambil dari Gambar III.8 adalah terjadi *collision* pada kedua objek dengan objek B berada pada sisi atas dari objek A.

III.1.4 Analisis Kebutuhan Non Fungsional

Analisis kebutuhan non-fungsional meliputi analisis kebutuhan perangkat keras, analisis kebutuhan perangkat lunak, dan analisis pengguna.

III.1.4.1 Analisis Kebutuhan Perangkat Keras

Perangkat keras yang dibutuhkan untuk membangun aplikasi *game* 2D *"Dasar Jana: Mengejar Dewi Suda"* ini adalah sebagai berikut:

1. Processor : 2.2 GHz
2. VGA : 128 MB
3. RAM : 1 GB
4. Harddisk : 320 GB
5. Monitor : 14 inch.
6. Mouse dan Keyboard
7. Speaker

Perangkat keras yang dibutuhkan untuk menjalankan aplikasi *game* 2D *"Dasar Jana: Mengejar Dewi Suda"* ini adalah sebagai berikut:

1. Processor : 2.2 GHz
2. VGA : 128 MB
3. RAM : 1 GB
4. Harddisk : 40 GB
5. Monitor : 14 inch.
6. Mouse dan Keyboard
7. Speaker (opsional)

Perangkat keras yang telah tersedia saat ini adalah sebagai berikut:

1. Processor : Dual-Core 2.2 GHz
2. VGA : 256 MB
3. RAM : 2 GB
4. Harddisk : 320 GB

5. Monitor : 14 inch.
6. Mouse dan Keyboard
7. Speaker

Hasil evaluasi dengan melihat spesifikasi perangkat keras yang dibutuhkan untuk membangun aplikasi serta yang telah tersedia saat ini, maka tidak ada hambatan dan tidak perlu ada proses *upgrade* pada perangkat keras yang telah tersedia.

III.1.4.2 Analisis Kebutuhan Perangkat Lunak

Perangkat lunak yang dibutuhkan untuk membangun aplikasi *game* 2D *"Dasar Jana: Mengejar Dewi Suda"* ini adalah sebagai berikut:

1. Sistem Operasi Windows 7.
2. Adobe Flash CS 5.5.
3. Adobe Photosop CS 4.
4. Flash Player 10.2 ke atas.

Perangkat lunak yang dibutuhkan untuk menjalankan aplikasi *game* 2D *"Dasar Jana: Mengejar Dewi Suda"* ini adalah sebagai berikut:

1. Sistem Operasi Windows XP ke atas.
2. Flash Player 10.2 ke atas.

Perangkat lunak telah tersedia saat ini adalah sebagai berikut:

1. Sistem Operasi Windows 7 (32 bit).
2. Adobe Flash CS 5.5.
3. Adobe Photosop CS 4.
4. Flash Player 11.6.

Hasil evaluasi dengan melihat spesifikasi perangkat lunak yang dibutuhkan untuk membangun aplikasi serta yang telah tersedia saat ini, maka tidak ada hambatan dan tidak perlu ada proses instalasi perangkat lunak.

III.1.4.3 Analisis Pengguna

Game yang akan dibuat ditujukan khususnya untuk mahasiswa tingkat akhir, tetapi juga dapat digunakan oleh publik dengan spesifikasi minimal sebagai berikut:

1. Peran : Pemain
2. Usia : Lebih dari 10 tahun
3. Tingkat Keterampilan : Dapat menggunakan komputer
4. Pengalaman : Pernah memainkan aplikasi *game*
5. Bahasa : Dapat memahami bahasa Indonesia

III.1.5 Analisis Kebutuhan Fungsional

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

III.1.5.1 *Use Case Diagram*

Use case adalah teknik untuk merekam persyaratan fungsional sebuah sistem. Sistem yang akan dibuat terdiri dari beberapa fungsional yaitu memulai permainan, mainkan *game* level 1, mainkan *game* level 2, mainkan *game* level 3, menggerakkan karakter, menyimpan skor tertinggi, menampilkan skor tertinggi, menampilkan petunjuk, menampilkan tentang kreator, mengatur musik latar, dan mengatur efek suara. *Use case diagram* dapat dilihat pada Gambar III.9.



Gambar III.9 *Use Case Diagram* Aplikasi *Game*

III.1.5.1.1 Identifikasi Aktor

Kata aktor dalam konteks UML, menampilkan peran (*roles)* yang pengguna atau sesuatu diluar sistem yang dikembangkan yang dapat berupa perangkat keras, *end user,* sistem yang lain, dan sebagainya. Pada rancangan ini hanya terdapat satu aktor yakni *user*.

III.1.5.1.2 Definisi *Use Case*

Definisi *use case* berfungsi untuk menjelaskan proses yang terdapat pada setiap *use case*. Penjelasan tiap *use case* dapat dilihat pada Tabel III.6.

Tabel III.6 Definisi *Use Case*

No.	*Use Case*	Deskripsi
1.	Memulai Permainan	Proses untuk memulai permainan

No.	*Use Case*	Deskripsi
2.	Mainkan *Game* Level 1	Proses untuk memainkan *game* level pertama
3.	Mainkan *Game* Level 2	Proses untuk memainkan *game* level kedua
4.	Mainkan *Game* Level 3	Proses untuk memainkan *game* level ketiga
5.	Menggerakkan Karakter	Proses untuk menggerakkan karakter dalam *game*
6.	Menyimpan Skor Tertinggi	Proses untuk menyimpan skor tertinggi *game*
7.	Menampilkan Skor Tertinggi	Proses untuk menampilkan skor tertinggi *game*
8.	Menampilkan Petunjuk	Proses untuk melihat petunjuk *game*
9.	Menampilkan Tentang Kreator	Proses untuk menampilkan informasi kreator aplikasi *game*
10.	Mengatur Suara	Proses untuk mengatur suara dalam *game*

III.1.5.1.3 Skenario *Use Case*

Skenario setiap bagian pada *use case* menunjukkan proses apa yang terjadi pada setiap bagian di dalam *use case* tersebut, dimana *user* memberikan perintah pada setiap bagian dan respon apa yang diberikan oleh sistem kepada *user* setelah *user* memberikan perintah pada setiap bagian-bagian *use case*. Skenario setiap *use case* yang ada pada *use case diagram* dapat dilihat pada Tabel III.7 sampai dengan Tabel III.16 sebagai berikut:

Tabel III.7 *Use Case Scenario* Memulai Permainan

Identifikasi		
Nomor	1	
Nama	Memulai Permainan	
Tujuan	Memainkan *game Dasar Jana: Mengejar Dewi Suda*	
Deskripsi	Memulai permainan	
Aktor	*User*	
Kondisi Awal	*User* berada pada tampilan menu utama *game*	
Skenario Normal		
Aksi Aktor		Reaksi Sistem
1. Memilih menu mulai *game*		-
-		2. Menampilkan komik 1
-		3. Menampilkan *game* level pertama
Skenario Alternatif		
Aksi Aktor		Reaksi Sistem
1. Memilih menu mulai *game*		-
-		2. Menampilkan komik 1
3. Menekan tombol lewati pada saat tampilan komik belum selesai		-
-		4. Melewati komik 1
-		5. Menampilkan *game* level pertama
Kondisi Akhir	*User* dapat melihat layar permainan level pertama	

Tabel III.8 *Use Case Scenario* Mainkan *Game* Level 1

Identifikasi	
Nomor	2
Nama	Mainkan *Game* Level 1
Tujuan	Memainkan *game Dasar Jana: Mengejar Dewi Suda* pada level pertama
Deskripsi	Memainkan *game* level pertama
Aktor	*User*
Kondisi Awal	*User* berada pada layar permainan level pertama
Skenario Normal	
Aksi Aktor	Reaksi Sistem
1. Menggerakkan karakter	2. Menggerakkan NPC
3. Mengalahkan bos musuh	-
-	4. Menampilkan portal
5. Menekan tombol *enter* pada *keyboard* saat berada di portal	-
-	6. Menampilkan komik 2
-	7. Menampilkan komik 3
-	8. Menampilkan *game* level kedua
Skenario Alternatif	
Aksi Aktor	Reaksi Sistem
1. Menggerakkan karakter	2. Menggerakkan NPC
3. Mengalahkan bos musuh	-
-	4. Menampilkan portal
5. Menekan tombol *enter* pada *keyboard* saat berada di portal	-
-	6. Menampilkan komik 2
7. Menekan tombol lewati pada saat tampilan komik belum selesai	
	8. Melewati komik 2
-	9. Menampilkan komik 3
10. Menekan tombol lewati pada saat tampilan komik belum selesai	-
	11. Melewati komik 3
-	12. Menampilkan *game* level kedua
Kondisi Akhir	*User* dapat melihat layar permainan level kedua

Tabel III.9 *Use Case Scenario* Mainkan *Game* Level 2

Identifikasi	
Nomor	3
Nama	Mainkan *Game* Level 2
Tujuan	Memainkan *game Dasar Jana: Mengejar Dewi Suda* pada level kedua
Deskripsi	Memainkan *game* level kedua

Aktor	*User*	
Kondisi Awal	*User* berada pada layar permainan level kedua	
Skenario Normal		
Aksi Aktor		Reaksi Sistem
1. Menggerakkan karakter		2. Menggerakkan NPC
3. Mengalahkan bos musuh		-
-		4. Menampilkan portal
5. Menekan tombol *enter* pada *keyboard* saat berada di portal		-
-		6. Menampilkan komik 4
-		7. Menampilkan komik 5
-		8. Menampilkan *game* level ketiga
Skenario Alternatif		
Aksi Aktor		Reaksi Sistem
1. Menggerakkan karakter		2. Menggerakkan NPC
3. Mengalahkan bos musuh		-
-		4. Menampilkan portal
5. Menekan tombol *enter* pada *keyboard* saat berada di portal		-
-		6. Menampilkan komik 4
7. Menekan tombol lewati pada saat tampilan komik belum selesai		
		8. Melewati komik 4
-		9. Menampilkan komik 5
10. Menekan tombol lewati pada saat tampilan komik belum selesai		-
		11. Melewati komik 5
-		12. Menampilkan *game* level ketiga
Kondisi Akhir	*User* dapat melihat layar permainan level ketiga	

Tabel III.10 *Use Case Scenario* Mainkan *Game* Level 3

Identifikasi		
Nomor	4	
Nama	Mainkan *Game* Level 3	
Tujuan	Memainkan *game Dasar Jana: Mengejar Dewi Suda* pada level ketiga	
Deskripsi	Memainkan *game* level ketiga	
Aktor	*User*	
Kondisi Awal	*User* berada pada layar permainan level ketiga	
Skenario Normal		
Aksi Aktor		Reaksi Sistem
1. Menggerakkan karakter		2. Menggerakkan NPC
3. Mengalahkan bos musuh		-
-		4. Menampilkan portal

5. Menekan tombol *enter* pada *keyboard* saat saat berada di portal	-
-	6. Menampilkan komik 6
Kondisi Akhir	*User* dapat melihat komik setelah level ketiga

Tabel III.11 *Use Case Scenario* Menggerakkan Karakter

Identifikasi	
Nomor	5
Nama	Menggerakkan Karakter
Tujuan	Menggerakkan karakter dalam *game Dasar Jana: Mengejar Dewi Suda*
Deskripsi	Menggerakkan karakter dalam *game*
Aktor	*User*
Kondisi Awal	*User* berada di dalam permainan (level pertama, level kedua, level ketiga)

Skenario Normal	
Aksi Aktor	Reaksi Sistem
1. Menekan tombol (*right arrow*) pada *keyboard*	-
-	2. Memeriksa tombol yang ditekan
-	3. Menggerakkan karakter untuk berjalan ke sebelah kanan

Skenario Alternatif 1	
Aksi Aktor	Reaksi Sistem
1. Menekan tombol (*left arrow*) pada *keyboard*	-
-	2. Memeriksa tombol yang ditekan
-	3. Menggerakkan karakter untuk berjalan ke sebelah kiri

Skenario Alternatif 2	
Aksi Aktor	Reaksi Sistem
1. Menekan tombol (*up arrow*) pada *keyboard*	-
-	2. Memeriksa tombol yang ditekan
-	3. Menggerakkan karakter untuk melompat

Skenario Alternatif 3	
Aksi Aktor	Reaksi Sistem
1. Menekan tombol (A) pada *keyboard*	-
-	2. Memeriksa tombol yang ditekan
-	3. Menggerakkan karakter untuk menyerang dengan menggunakan tas

Skenario Alternatif 4	
Aksi Aktor	Reaksi Sistem
1. Menekan tombol (S) pada *keyboard*	-
-	2. Memeriksa tombol yang ditekan
-	3. Menggerakkan karakter untuk menyerang dengan tendangan
Kondisi Akhir	*User* dapat melihat karakter bergerak di dalam *game*

Tabel III.12 *Use Case Scenario* Menyimpan Skor Tertinggi

Identifikasi	
Nomor	6
Nama	Menyimpan Skor Tertinggi
Tujuan	Menyimpan skor beserta *username* pemain ke dalam skor tertinggi *board*
Deskripsi	Menyimpan skor tertinggi *game*
Aktor	*User*
Kondisi Awal	*User* berada pada tampilan form dimana *user* diizinkan untuk mengisi *username* dan menekan tombol *ok*
Skenario Normal	
Aksi Aktor	Reaksi Sistem
1. Mengisi data *username*	-
2. Menekan tombok *ok*	-
-	3. Menyimpan data *username* beserta skornya
-	4. Menampilkan pesan data berhasil disimpan
Skenario Alternatif	
Aksi Aktor	Reaksi Sistem
1. Mengisi data *username*	-
2. Menekan tombok *ok*	-
-	3. Menyimpan data *username* beserta skornya
-	4. Menampilkan pesan data gagal disimpan
Kondisi Akhir	*User* dapat melihat pesan penyimpanan skor tertinggi.

Tabel III.13 *Use Case Scenario* Menampilkan Skor Tertinggi

Identifikasi	
Nomor	7
Nama	Menampilkan Skor Tertinggi
Tujuan	Menampilkan skor tertinggi yang pernah memainkan *game Dasar Jana: Mengejar Dewi Suda*
Deskripsi	Menampilkan petunjuk *game*
Aktor	*User*
Kondisi Awal	*User* berada pada tampilan menu utama *game*
Skenario Normal	
Aksi Aktor	Reaksi Sistem
1. Memilih menu skor tertinggi	-
-	2. Menampilkan skor tertinggi *game*
Skenario Alternatif	
Aksi Aktor	Reaksi Sistem
-	-
Kondisi Akhir	*User* dapat melihat skor tertinggi yang pernah didapatkan

Tabel III.14 *Use Case Scenario* Menampilkan Petunjuk

Identifikasi	
Nomor	8
Nama	Menampilkan Petunjuk
Tujuan	Menampilkan petunjuk untuk memainkan *game Dasar Jana:*

	Mengejar Dewi Suda	
Deskripsi	Menampilkan petunjuk *game*	
Aktor	*User*	
Kondisi Awal	*User* berada pada tampilan menu utama *game*	
Skenario Normal		
Aksi Aktor		Reaksi Sistem
1. Memilih menu petunjuk		-
-		2. Menampilkan petunjuk *game*
Skenario Alternatif		
Aksi Aktor		Reaksi Sistem
-		-
Kondisi Akhir	*User* dapat melihat petunjuk untuk memainkan *game*	

Tabel III.15 *Use Case Scenario* Menampilkan Tentang Kreator

Identifikasi		
Nomor	9	
Nama	Menampilkan Tentang Kreator	
Tujuan	Menampilkan informasi tentang kreator aplikasi *game*	
Deskripsi	Menampilkan informasi kreator aplikasi *game*	
Aktor	*User*	
Kondisi Awal	*User* berada pada tampilan menu utama *game*	
Skenario Normal		
Aksi Aktor		Reaksi Sistem
1. Memilih menu info		-
-		2. Menampilkan informasi kreator *game*
Skenario Alternatif		
Aksi Aktor		Reaksi Sistem
-		-
Kondisi Akhir	*User* dapat melihat tampilan informasi tentang kreator *game*	

Tabel III.16 *Use Case Scenario* Mengatur Suara

Identifikasi		
Nomor	10	
Nama	Mengatur Suara	
Tujuan	Mengatur suara dalam *game Dasar Jana: Mengejar Dewi Suda*	
Deskripsi	Mengatur suara dalam *game*	
Aktor	*User*	
Kondisi Awal	*User* berada pada aplikasi *game*	
Skenario Normal		
Aksi Aktor		Reaksi Sistem
1. Memilih tombol *sound on*		-
-		2. Memeriksa tombol yang ditekan
-		3. Memunculkan suara dalam *game*
Kondisi Akhir	*User* dapat mendengar suara dalam *game*	
Skenario Alternatif		
Aksi Aktor		Reaksi Sistem

1. Memilih tombol *sound off*	-
-	2. Memeriksa tombol yang ditekan
-	3. Menghilangkan suara dalam *game*
Kondisi Akhir	*User* tidak dapat mendengar suara dalam *game*

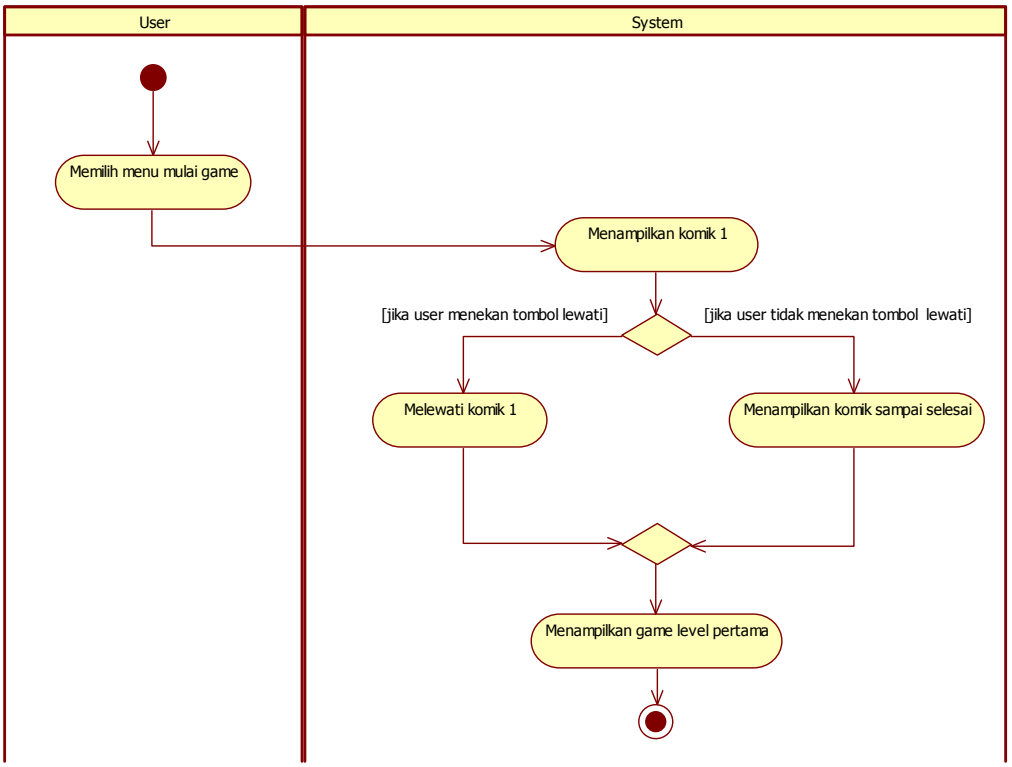
III.1.5.2 *Activity Diagram*

Activity diagram adalah teknik untuk menggambarkan logika prosedural, proses bisnis, dan jalur kerja. Dalam beberapa hal, diagram ini memainkan peran mirip sebuah diagram alir, tetapi perbedaan prinsip antara diagram ini dan notasi diagram alir adalah diagram ini mendukung *behavior* paralel.

Perancangan *activity diagram* aplikasi *game* dapat dilihat pada Gambar III.10 sampai dengan Gambar III.19.

1. *Activity Diagram* Memulai Permainan

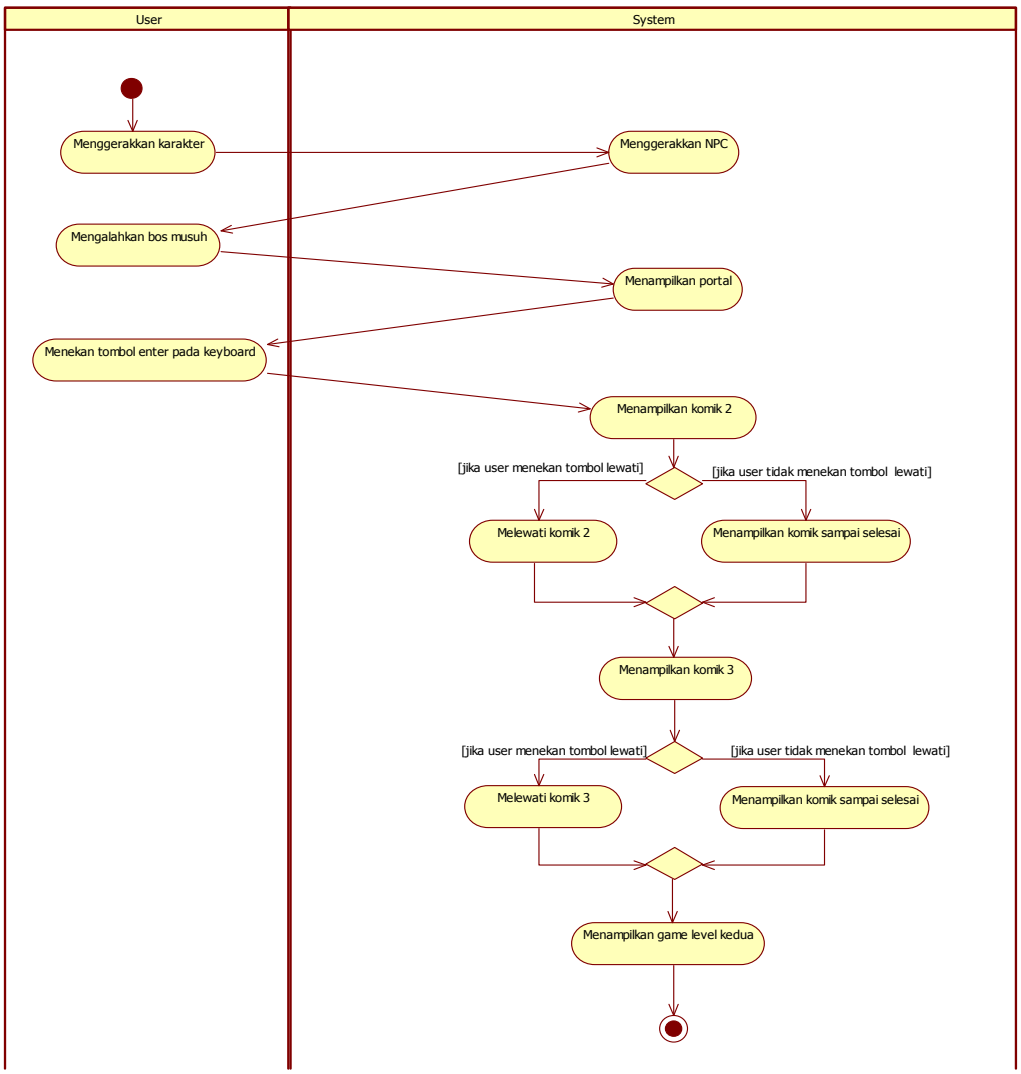
Proses ini terjadi ketika *user* memilih menu mulai *game* pada halaman utama *game Dasar Jana.* Sistem akan menampilkan komik 1. Jika *user* menekan tombol *lewati* maka komik yang sedang ditampilkan segera dilewati, namun jika *user* tidak menekan tombol *lewati* maka komik akan dilanjutkan sampai selesai. Kemudian sistem menampilkan *game* level pertama.



Gambar III.10 *Activity Diagram* Memulai Permainan

2. *Activity Diagram* Mainkan Game Level 1

Proses ini terjadi ketika *user* sudah berada di dalam layar permainan (*playing screen*) pada level pertama. *User* dapat menggerakkan karakter dengan menekan tombol pada *keyboard*, sementara sistem menggerakkan *Non-Playable Character* (NPC). Jika *user* mengalahkan bos musuh, sistem akan menampilkan portal. Jika *user* menekan tombol *enter* pada *keyboard* saat berada di portal, sistem kemudian menampilkan komik 2. Jika *user* menekan tombol *lewati*, sistem akan melewati komik 2 kemudian langsung menampilkan komik 3. Jika *user* menekan tombol *lewati*, sistem akan melewati komik 3 kemudian langsung menampilkan *game* level kedua.

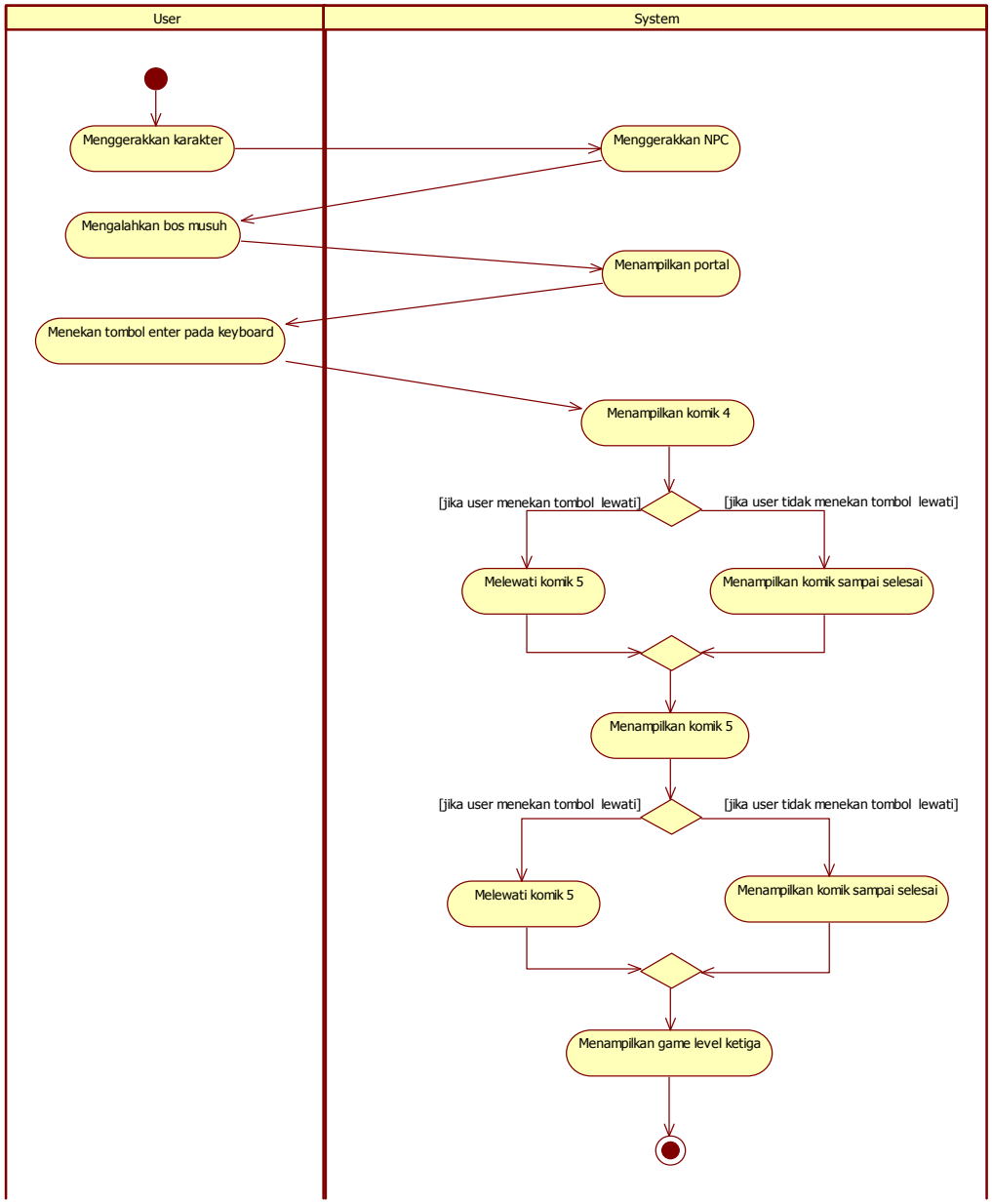


Gambar III.11 *Activity Diagram* Mainkan Game Level 1

3. *Activity Diagram* Mainkan Game Level 2

Proses ini terjadi ketika *user* sudah berada di dalam layar permainan (*playing screen*) pada level kedua. *User* dapat menggerakkan karakter dengan menekan tombol pada *keyboard*, sementara sistem menggerakkan *Non-Playable Character* (NPC). Jika *user* mengalahkan bos musuh, sistem akan menampilkan portal. Jika *user* menekan tombol *enter* pada *keyboard* saat berada di portal, sistem kemudian menampilkan komik 4. Jika *user* menekan tombol *lewati*, sistem akan melewati komik 4 kemudian langsung menampilkan komik 5. Jika *user*

menekan tombol *lewati*, sistem akan melewati komik 5 kemudian langsung menampilkan *game* level.

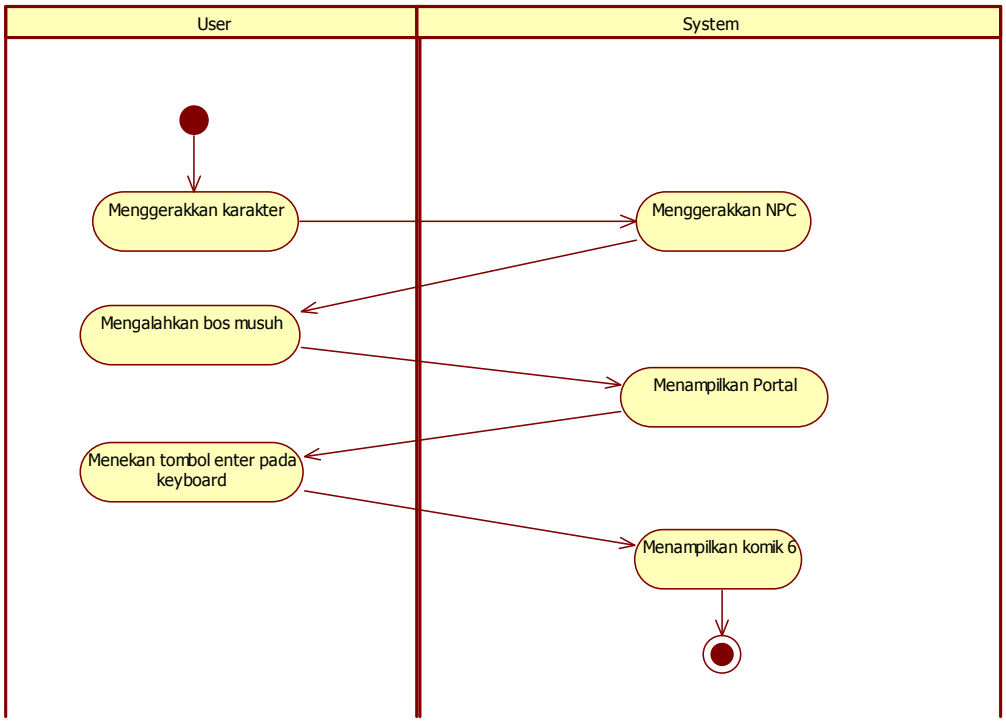


Gambar III.12 *Activity Diagram* Mainkan Game Level 2

4. *Activity Diagram* Mainkan Game Level 3

Proses ini terjadi ketika *user* sudah berada di dalam layar permainan (*playing screen*) pada level ketiga. *User* dapat menggerakkan karakter dengan menekan tombol pada *keyboard*, sementara sistem menggerakkan *Non-Playable*

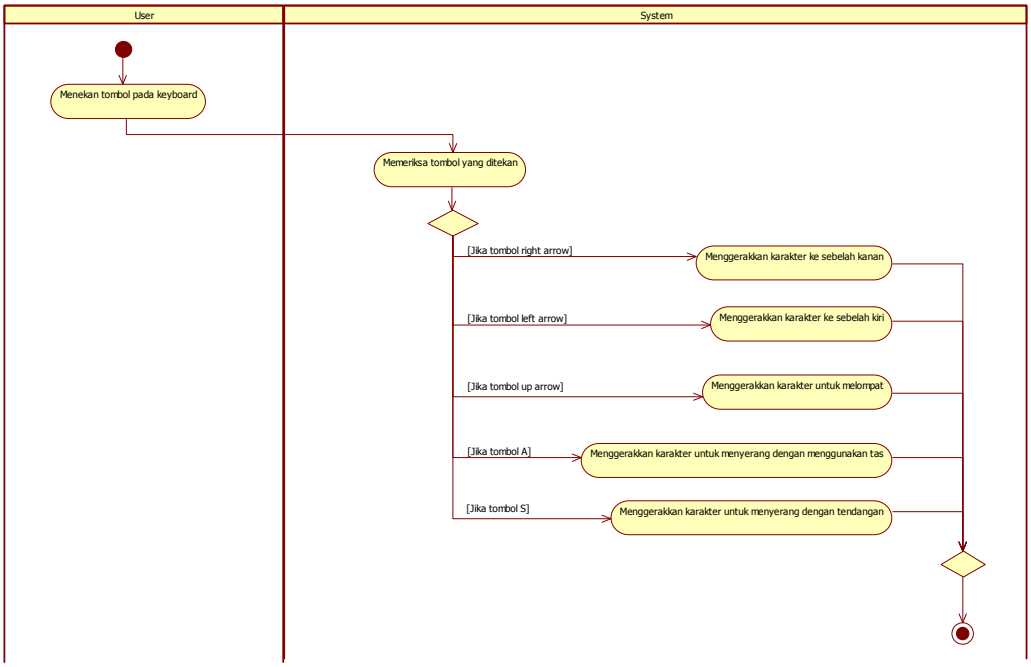
Character (NPC). Jika *user* mengalahkan bos musuh, sistem akan menampilkan Dewi Suda. Jika *user* menekan tombol *enter* pada *keyboard* saat bersentuhan dengan Dewi Suda, sistem kemudian menampilkan komik 6.



Gambar III.13 *Activity Diagram* Mainkan Game Level 3

5. *Activity Diagram* Menggerakkan Karakter

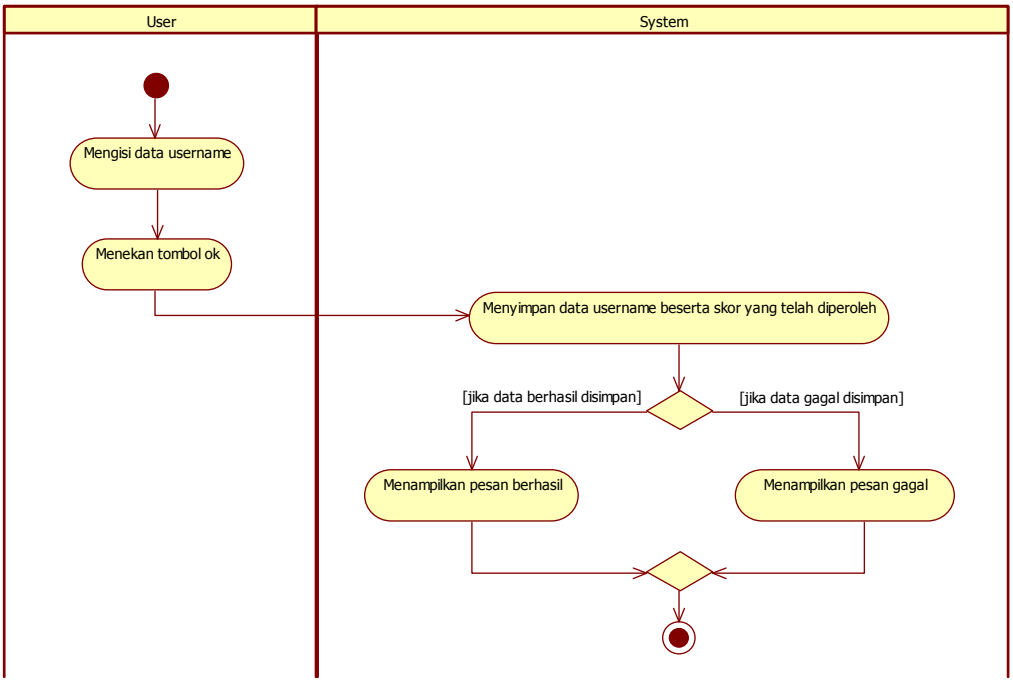
Proses ini terjadi ketika *user* sudah berada di dalam layar permainan (*playing screen*). *User* dapat menggerakkan karakter dengan menekan tombol pada *keyboard*. Jika *user* menekan *right arrow* maka sistem akan menggerakkan karakter ke sebelah kanan. Jika *user* menekan *left arrow* maka sistem akan menggerakkan karakter ke sebelah kiri. Jika *user* menekan *up arrow* maka sistem akan menggerakkan karakter untuk melompat. Jika *user* menekan tombol "A" maka sistem akan menggerakkan karakter untuk menyerang dengan menggunakan tas. Jika *user* menekan tombol "S" maka sistem akan menggerakkan karakter untuk menyerang dengan tendangan.



Gambar III.14 *Activity Diagram* Menggerakkan Karakter

6. *Activity Diagram* Menyimpan Skor Tertinggi

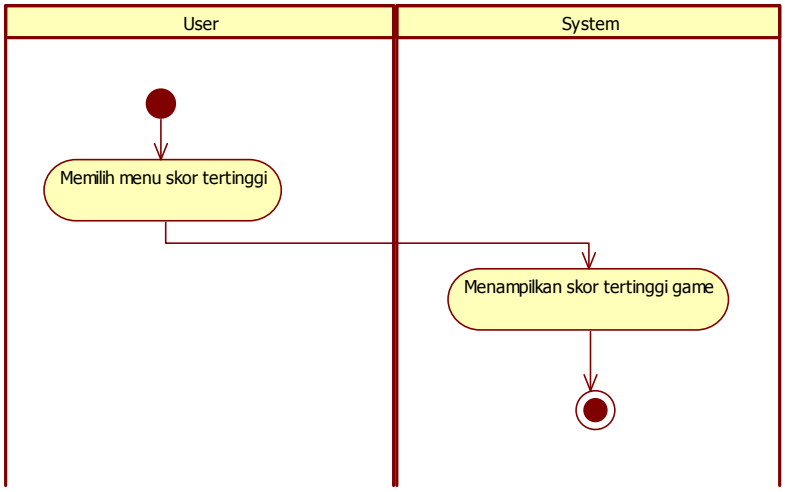
Proses ini terjadi ketika sistem telah melakukan pengecekan terhadap skor yang diperoleh oleh *user* dan dianggap layak untuk dijadikan skor tertinggi. Kondisi awal *user* berada di dalam form pengisian data *username*. *User* mengisi data *username*, lalu *user* menekan tombol *ok.* Sistem akan menyimpan data *username* dan skor yang diperoleh. Jika data berhasil disimpan maka sistem akan menampilkan pesan data berhasil disimpan sedangkan jika data gagal disimpan sistem akan menampilkan pesan data gagal disimpan.



Gambar III.15 *Activity Diagram* Menyimpan Skor Tertinggi

7. *Activity Diagram* Menampilkan Skor Tertinggi

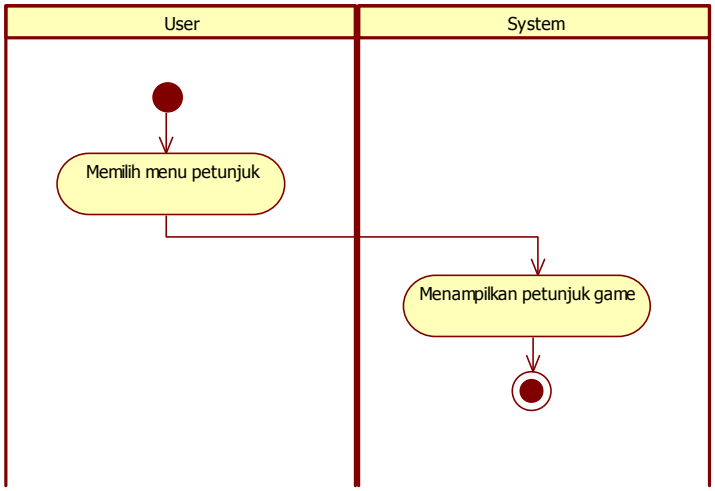
Proses ini terjadi ketika *user* memilih menu skor tertinggi pada halaman utama *game Dasar Jana.* Sistem akan menampilkan layar skor tertinggi *game* yang berisi informasi mengenai skor tertinggi yang pernah didapatkan oleh *user* setelah menyelesaikan *game.*



Gambar III.16 *Activity Diagram* Menampilkan Skor Tertinggi

8. *Activity Diagram* Menampilkan Petunjuk

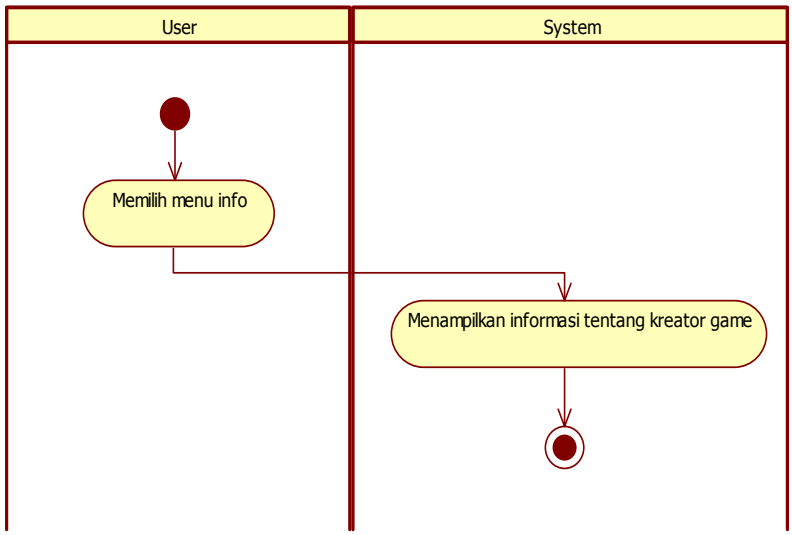
Proses ini terjadi ketika *user* memilih menu petunjuk pada halaman utama *game Dasar Jana.* Sistem akan menampilkan layar petunjuk *game* yang berisi informasi mengenai cara untuk menggerakkan karakter di dalam *game.*



Gambar III.17 *Activity Diagram* Menampilkan Petunjuk

9. *Activity Diagram* Menampilkan Tentang Kreator

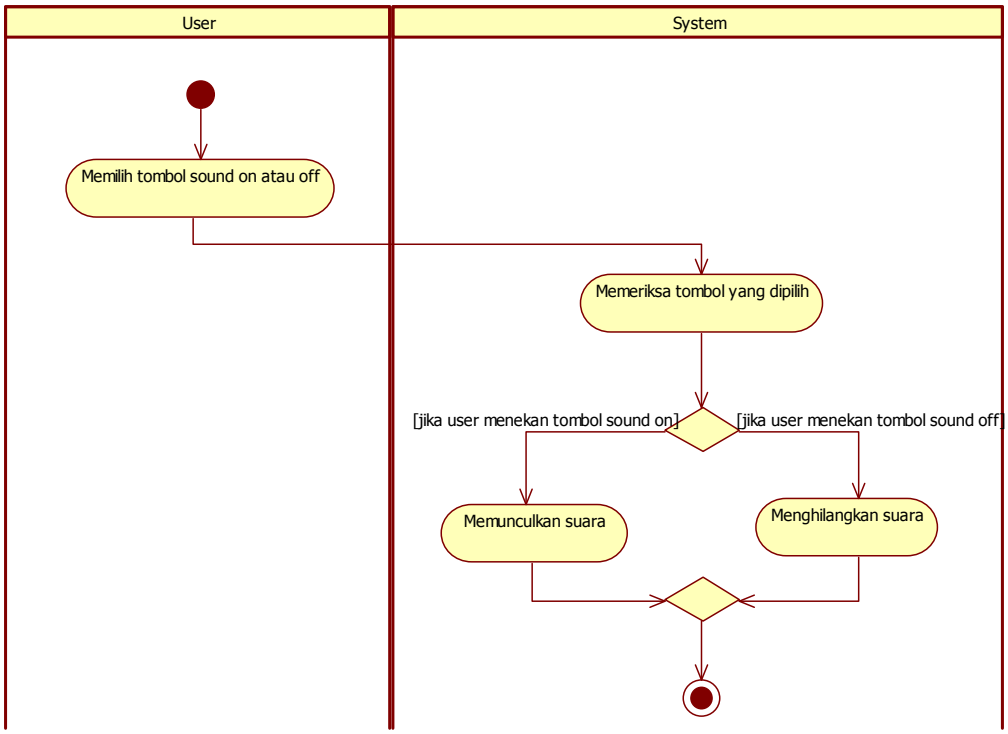
Proses ini terjadi ketika *user* memilih menu info pada halaman utama *game Dasar Jana.* Sistem akan menampilkan layar tentang kreator *game* yang berisi informasi mengenai kreator *game*.



Gambar III.18 *Activity Diagram* Menampilkan Tentang Kreator

10. *Activity Diagram* Mengatur Musik Latar

Proses ini terjadi ketika *user* berada pada aplikasi *game.* Setiap layar pada aplikasi *game* terdapat dua buah tombol untuk mengatur suara. Jika *user* menekan tombol *sound on* maka sistem akan memunculkan suara di dalam permainan, tetapi jika *user* menekan tombol *sound off* maka sistem akan menghilangkan suara di dalam permainan.

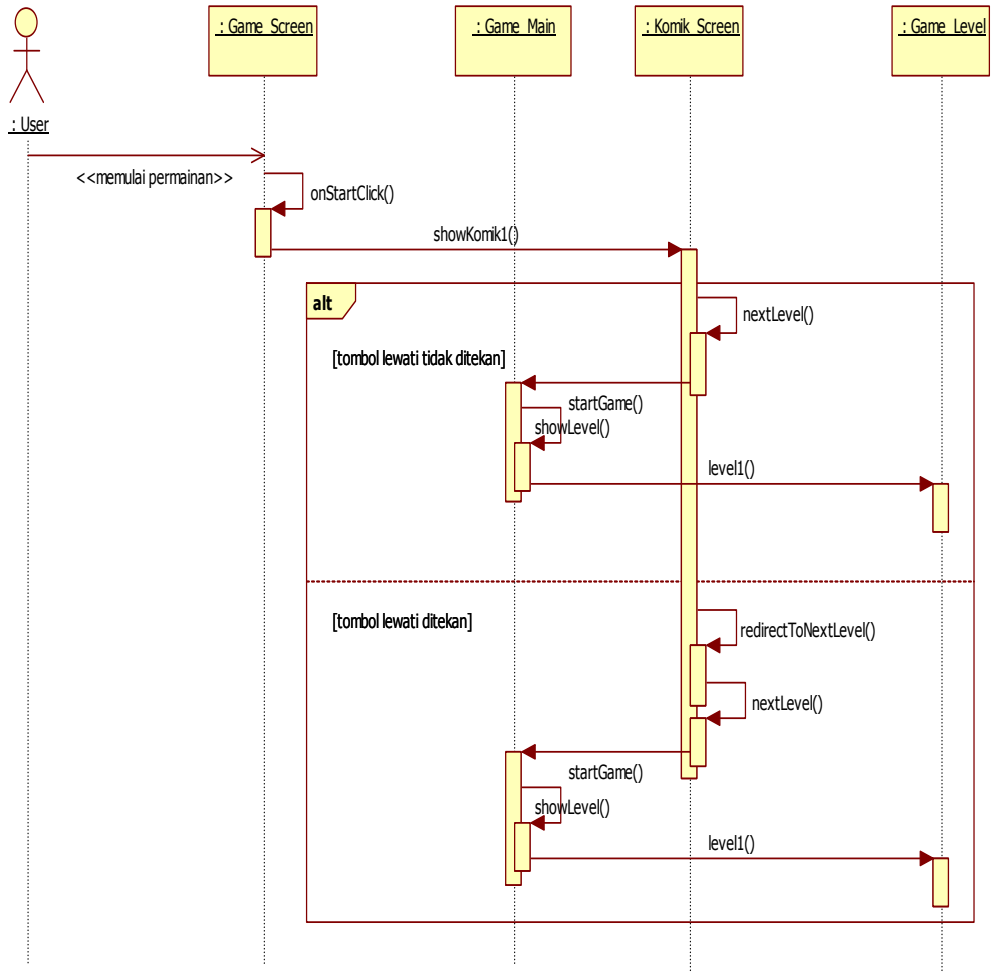


Gambar III.19 *Activity Diagram* Mengatur Suara

III.1.5.3 *Sequence Diagram*

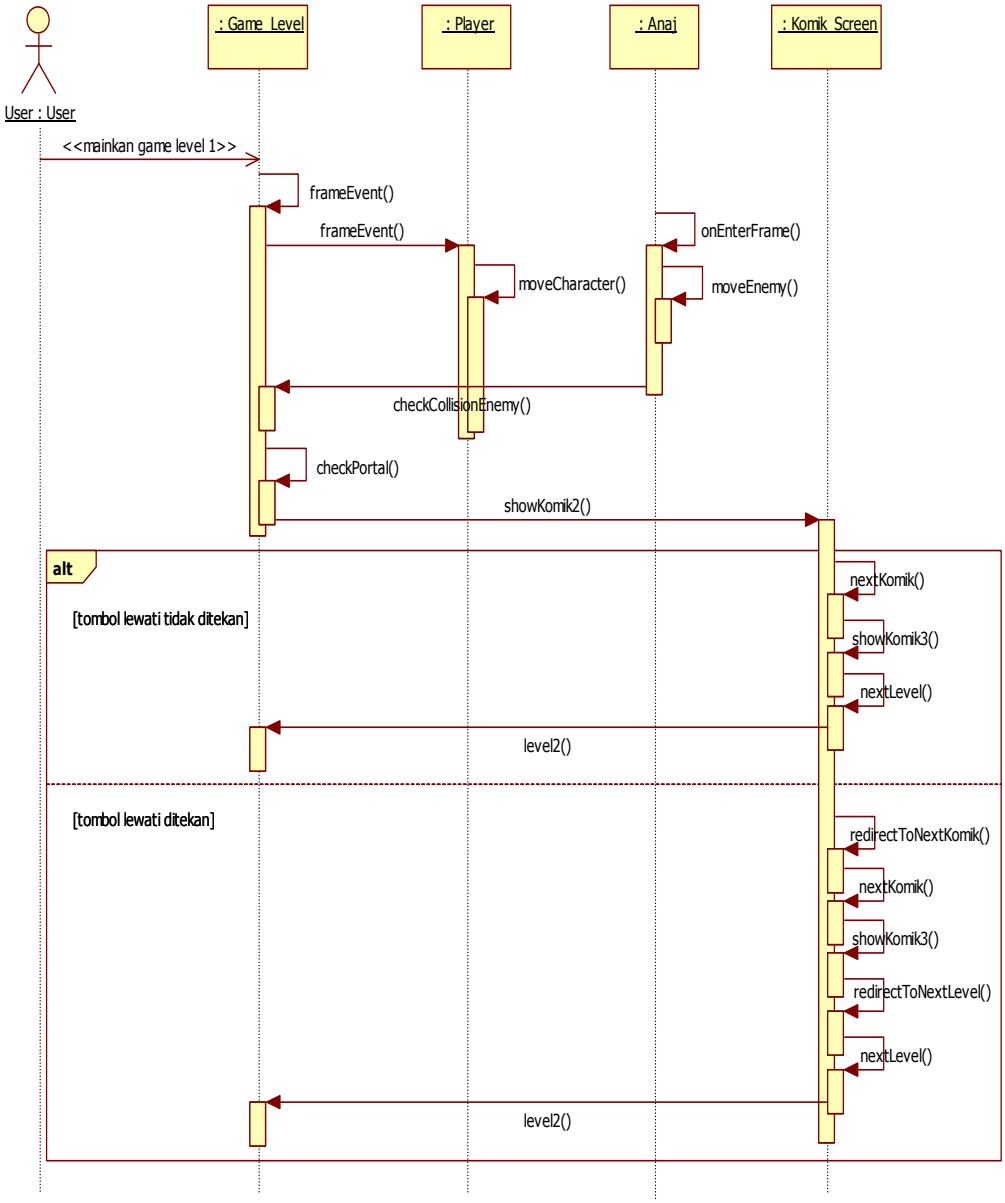
Sequence Diagram menjabarkan sebuah skenario tunggal. Diagram tersebut menunjukkan sejumlah objek contoh dan pesan-pesan yang melewati objek-objek ini di dalam *use case*. Perancangan *sequence diagram* aplikasi *game* dapat dilihat pada Gambar III.20 sampai dengan Gambar III.29.

1. *Sequence Diagram* Memulai Permainan



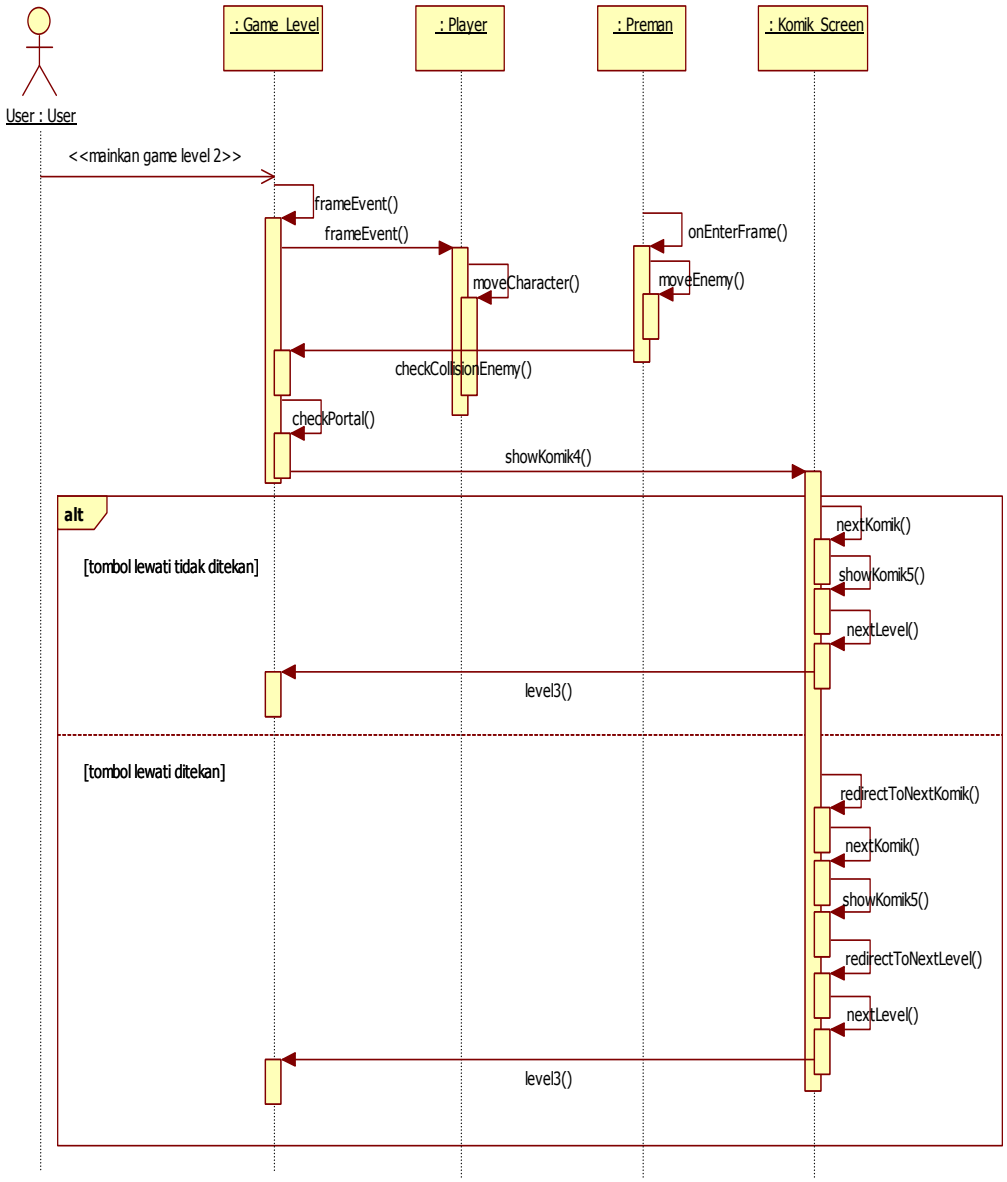
Gambar III.20 *Sequence Diagram* Memulai Permainan

2. *Sequence Diagram* Mainkan *Game* Level 1



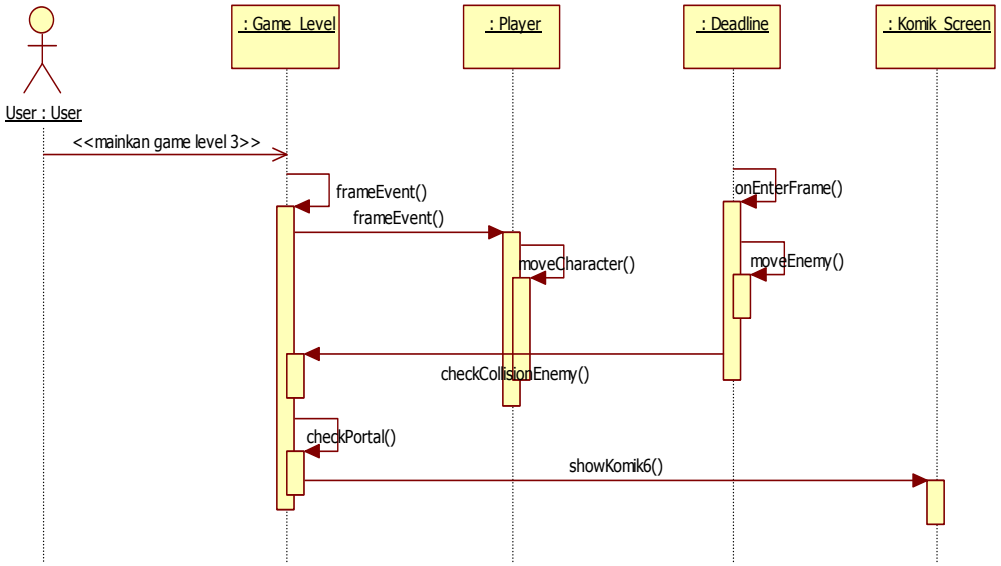
Gambar III.21 *Sequence Diagram* Mainkan *Game* Level 1

3. *Sequence Diagram* Mainkan *Game* Level 2



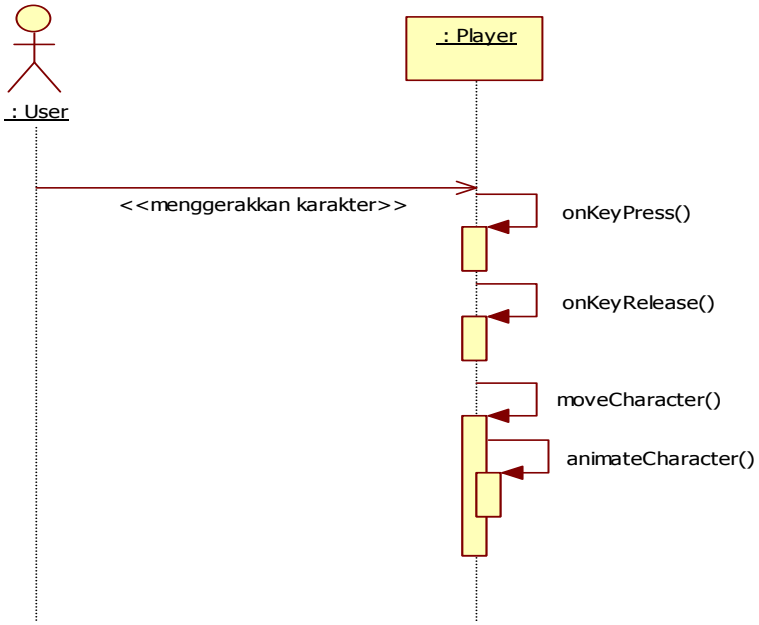
Gambar III.22 *Sequence Diagram* Mainkan *Game* Level 2

4. *Sequence Diagram* Mainkan *Game* Level 3



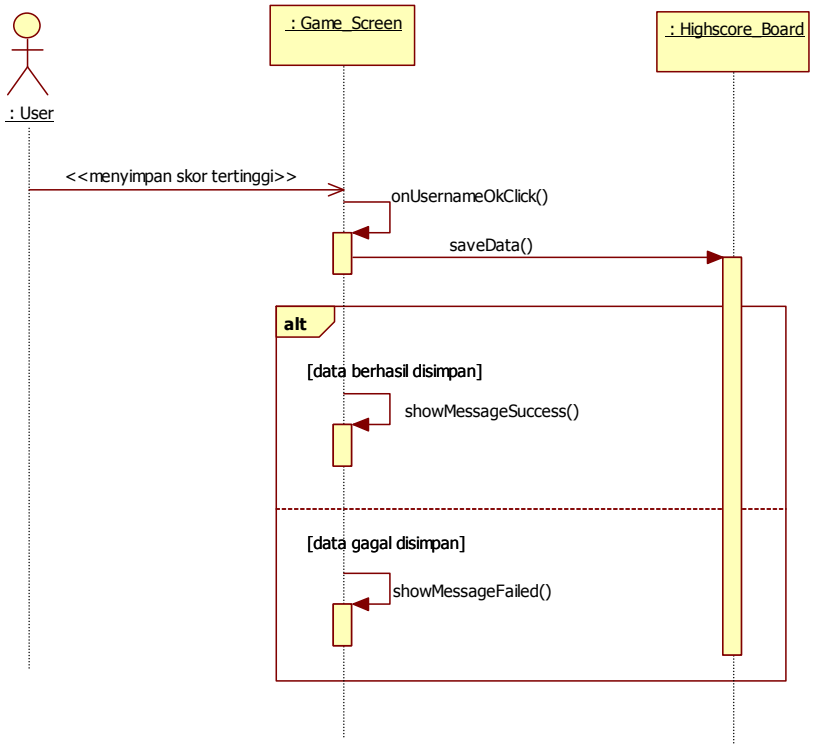
Gambar III.23 *Sequence Diagram* Mainkan *Game* Level 3

5. *Sequence Diagram* Menggerakkan Karakter



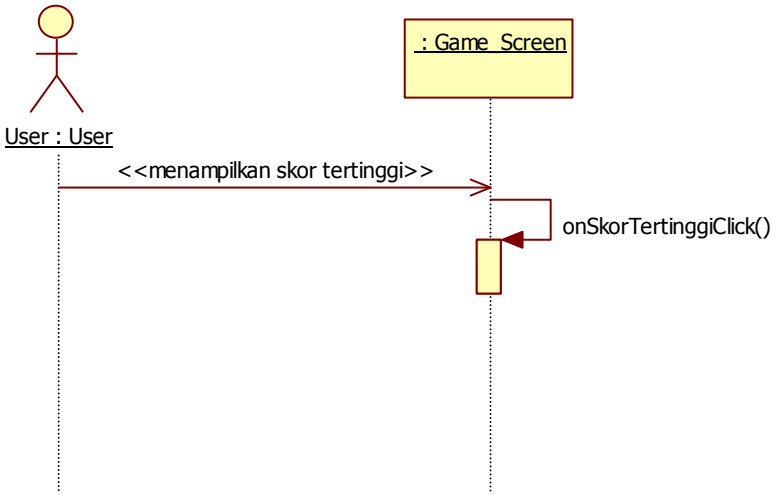
Gambar III.24 *Sequence Diagram* Menggerakkan Karakter

6. *Sequence Diagram* Menyimpan Skor Tertinggi



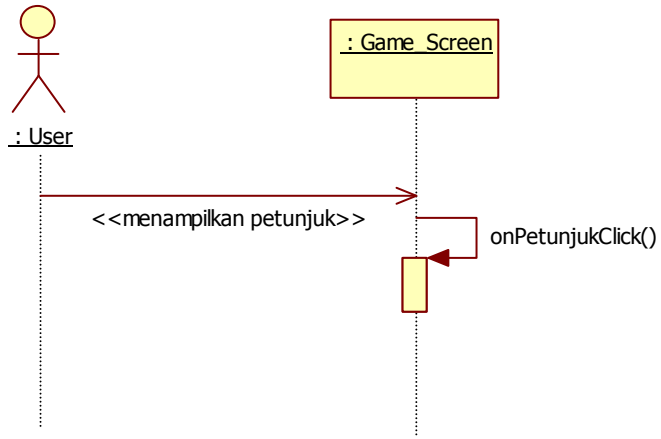
Gambar III.25 *Sequence Diagram* Menyimpan Skor Tertinggi

7. *Sequence Diagram* Menampilkan Skor Tertinggi



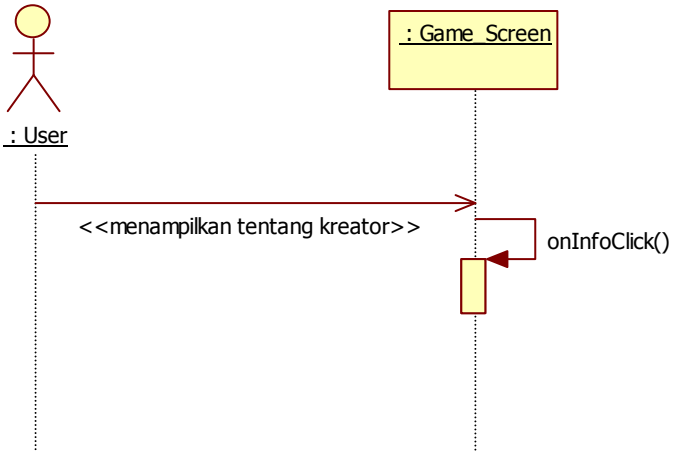
Gambar III.26 *Sequence Diagram* Menampilkan Skor Tertinggi

8. *Sequence Diagram* Menampilkan Petunjuk



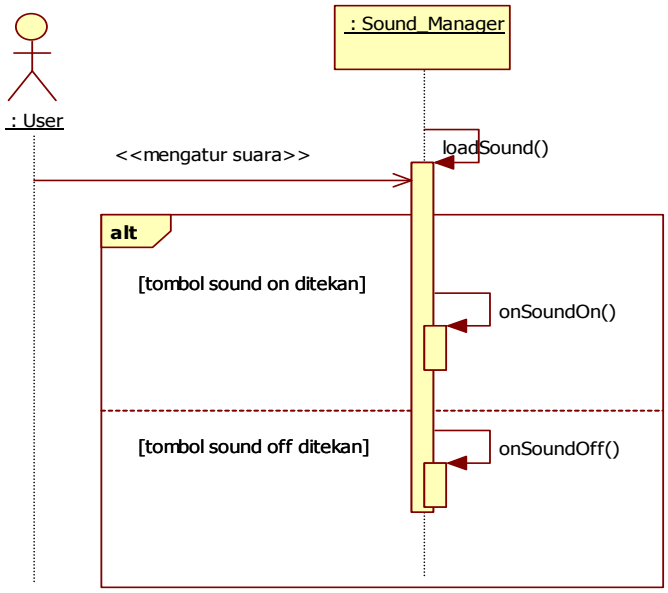
Gambar III.27 *Sequence Diagram* Menampilkan Petunjuk

9. *Sequence Diagram* Menampilkan Tentang Kreator



Gambar III.28 *Sequence Diagram* Menampilkan Tentang Kreator

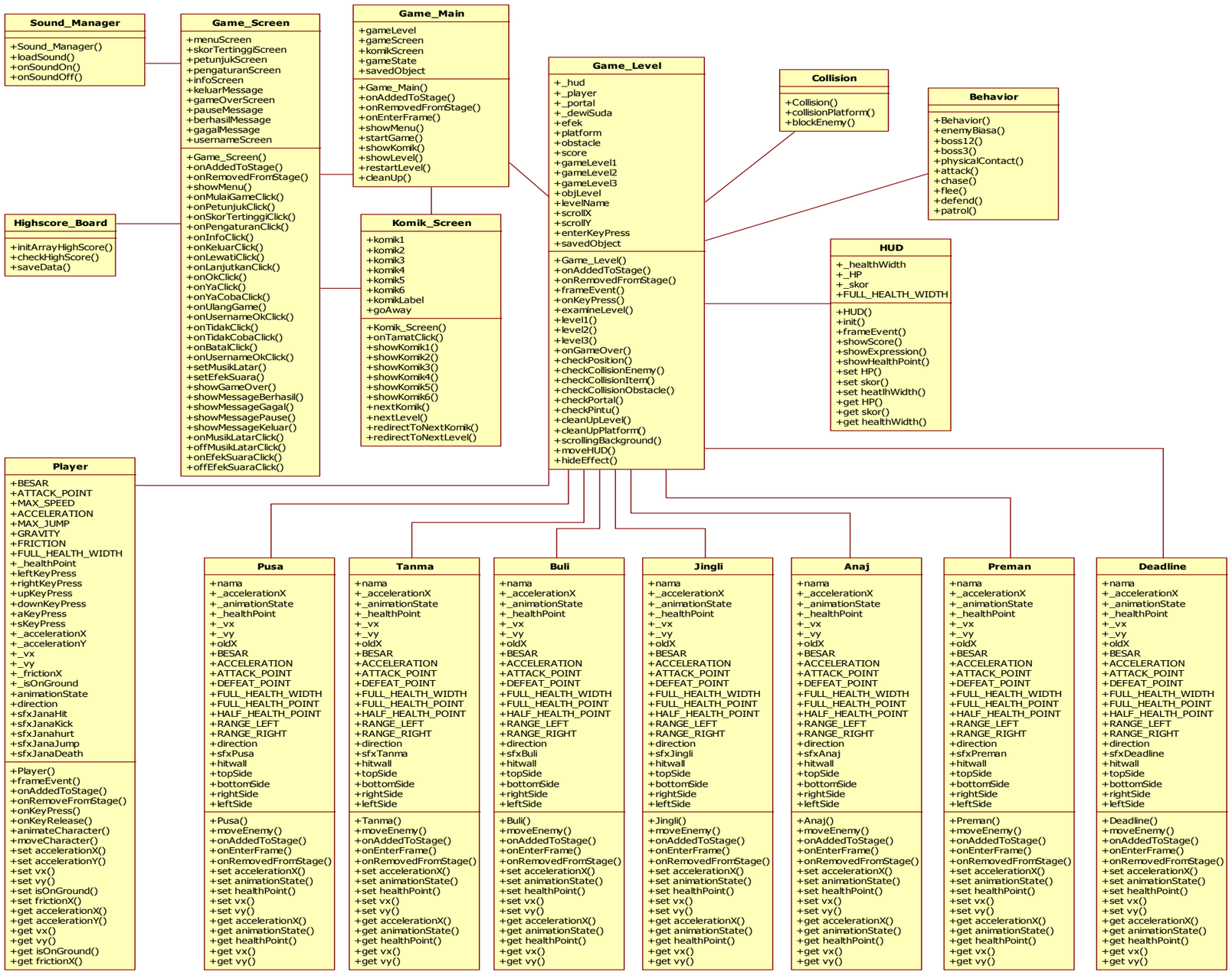
10. *Sequence Diagram* Mengatur Suara



Gambar III.29 *Sequence Diagram* Mengatur Suara

III.1.5.4 *Class Diagram*

Class Diagram mendeskripsikan jenis-jenis objek dan berbagai macam hubungan statis yang terdapat diantara mereka. *Class diagram* juga menunjukkan properti dan operasi sebuah *class* dan batasan-batasan yang terdapat dalam hubungan – hubungan objek tersebut. Perancangan *class diagram game* dapat dilihat pada Gambar III.30 sedangkan penjelasan mengenai *class* dapat dilihat pada Tabel III.18.



Gambar III.30 ***Class Diagram Game***

Penjelasan mengenai kelas-kelas yang terdapat pada *class diagram* di atas dapat dilihat pada tabel III.17.

Tabel III.17 Definisi *Class*

Nama *Class*	Deskripsi
Game_Main	Kelas yang pertama kali di-*running* ketika aplikasi *game* dijalankan.
Game_Screen	Kelas yang bertanggung jawab dalam memunculkan tampilan yang terdapat di dalam *game*, seperti tampilan petunjuk, pengaturan, dan sebagainya.
Game_Level	Kelas yang mengatur interaksi objek – objek yang terlibat ketika permainan berlangsung.
Komik_Screen	Merupakan kelas yang bertanggung jawab untuk menampilkan komik.
Player	Kelas yang bertanggung jawab untuk menggerakkan karakter sesuai interaksi yang dilakukan oleh *User*.
HUD	Kelas yang bertanggung jawab untuk menampilkan kondisi karakter yang dimainkan oleh *User*.
Pusa	Spesialisasi dari kelas *enemy* yang bertanggung jawab untuk menggerakkan NPC Pusa.
Tanma	Spesialisasi dari kelas *enemy* yang bertanggung jawab untuk menggerakkan NPC Tanma.
Anaj	Spesialisasi dari kelas *enemy* yang bertanggung jawab untuk menggerakkan NPC Anaj.
Buli	Spesialisasi dari kelas *enemy* yang bertanggung jawab untuk menggerakkan NPC Buli.
Jingli	Spesialisasi dari kelas *enemy*yang bertanggung jawab untuk menggerakkan NPC Jingli.
Preman	Spesialisasi dari kelas *enemy* yang bertanggung jawab untuk menggerakkan NPC Preman.
Deadline	Spesialisasi dari kelas *enemy* yang bertanggung jawab untuk menggerakkan NPC Jingli.
Collision	Kelas yang bertanggung jawab untuk mengelola *collision*
Sound_Manager	Kelas yang bertanggung jawab untuk memunculkan atau menghilangkan suara (baik musik latar maupun efek suara).
Highscore_Board	Kelas yang bertanggung jawab untuk melakukan penyimpanan skor tertinggi.
Behavior	Kelas yang bertanggung jawab untuk mengatur tingkah laku *enemy*.

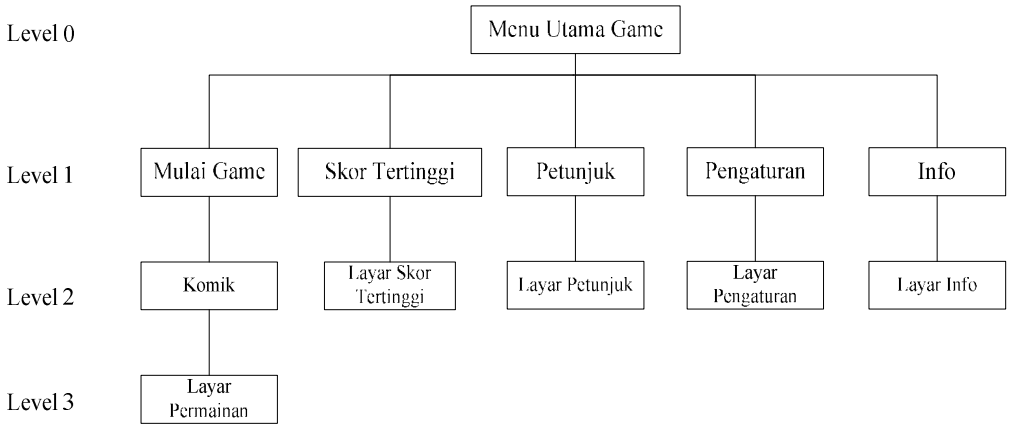
III.2 Perancangan Sistem

Pada bagian perancangan sistem akan dilakukan berbagai macam perancangan *game* yang akan dibangun, dimulai dari perancangan struktur menu,

perancangan antarmuka, perancangan pesan, jaringan semantik serta perancangan *method*.

III.2.1 Perancangan Struktur Menu

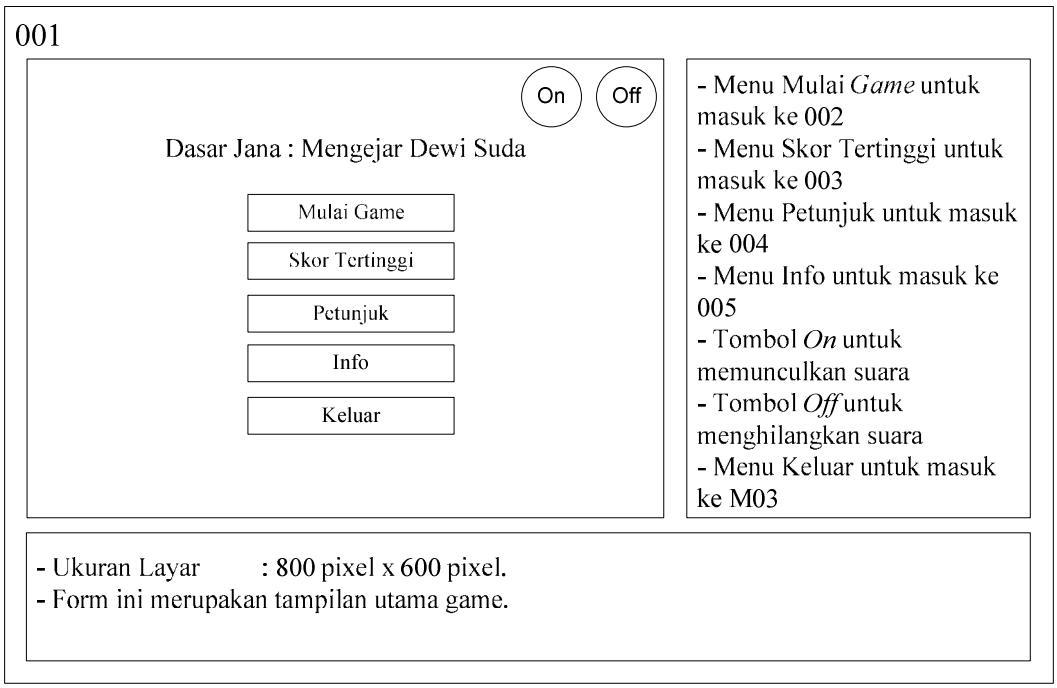
Aplikasi *game* yang akan dibuat terdiri dari lima buah menu yakni, mulai *game,* skor tertinggi, petunjuk, pengaturan, dan info. Sedangkan perancangan struktur menu dapat dilihat pada Gambar III.31.



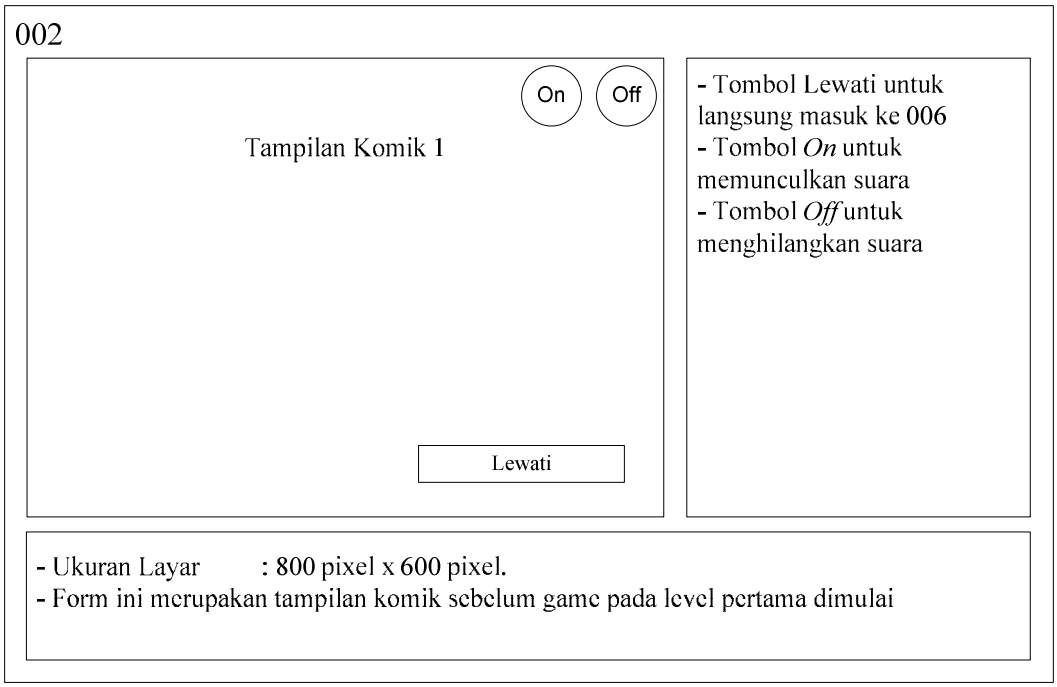
Gambar III.31 Perancangan Struktur Menu

III.2.2 Perancangan Antarmuka

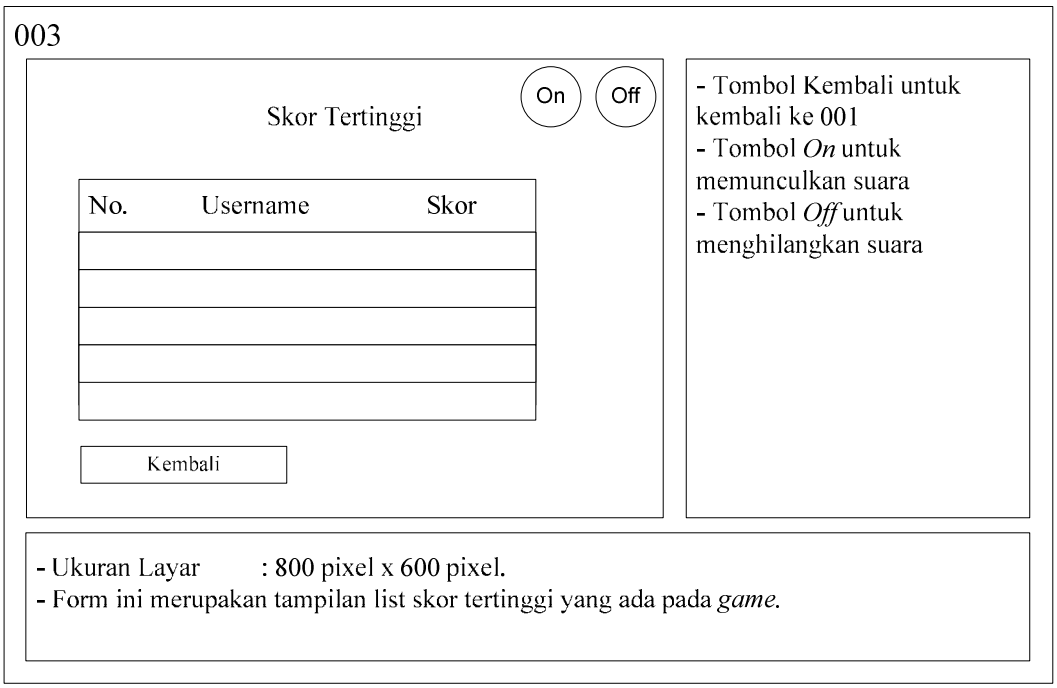
Perancangan antarmuka dibutuhkan untuk mewakili keadaan sebenarnya dari aplikasi yang akan dibangun. Perancangan dari aplikasi yang akan dibangun dapat dilihat pada Gambar III.32 sampai dengan III.46.



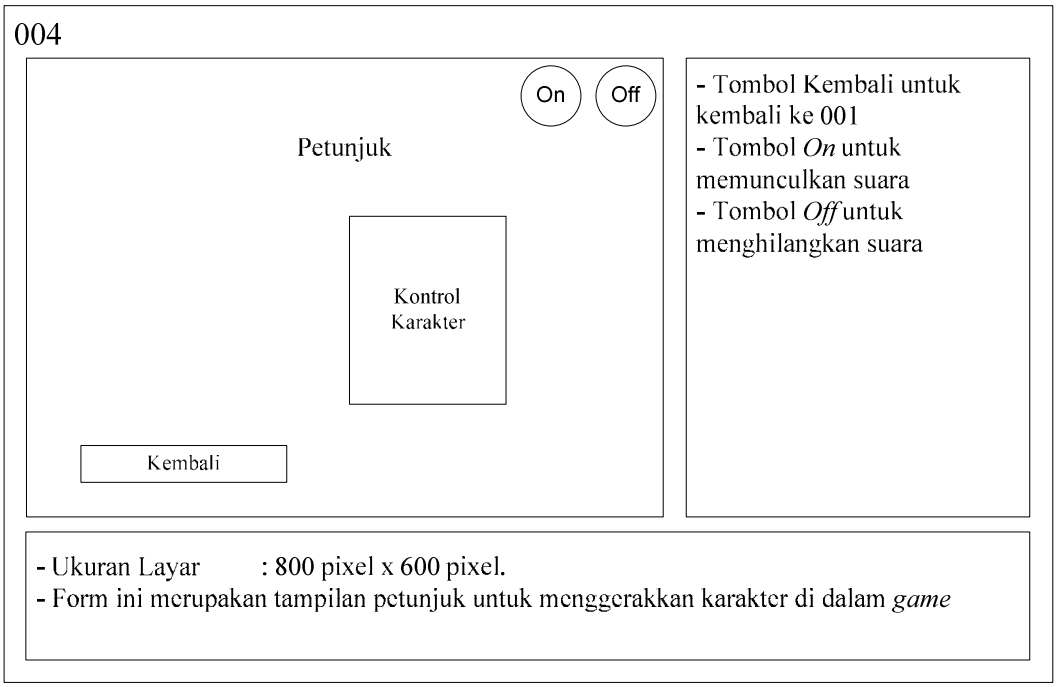
Gambar III.32 Tampilan Menu Utama G*ame* 001



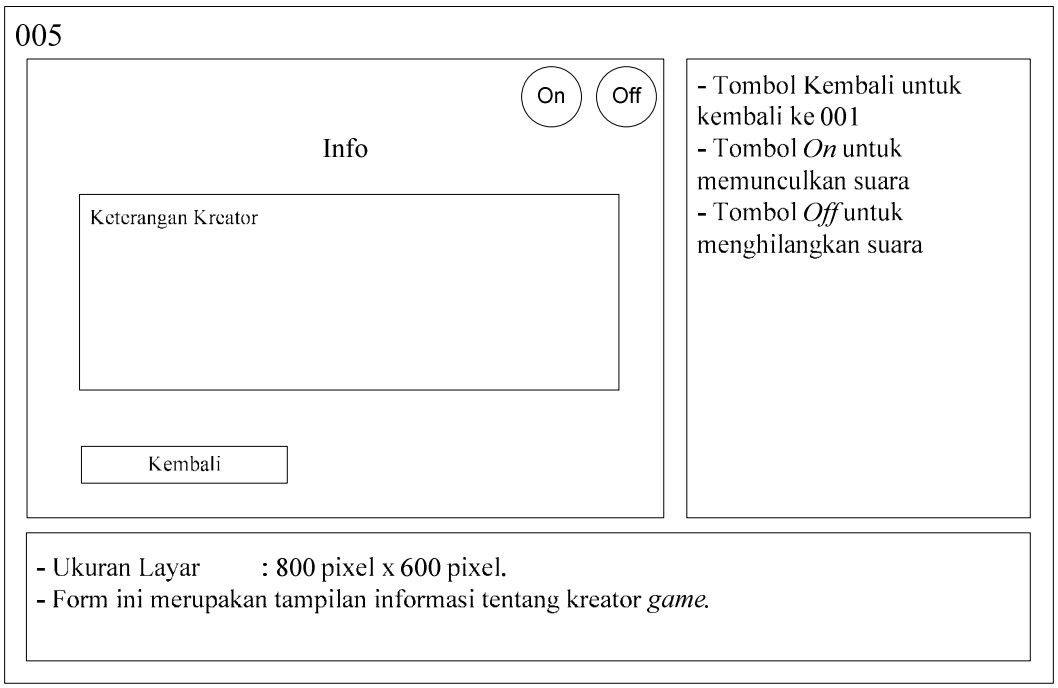
Gambar III.33 Tampilan Komik 1 002



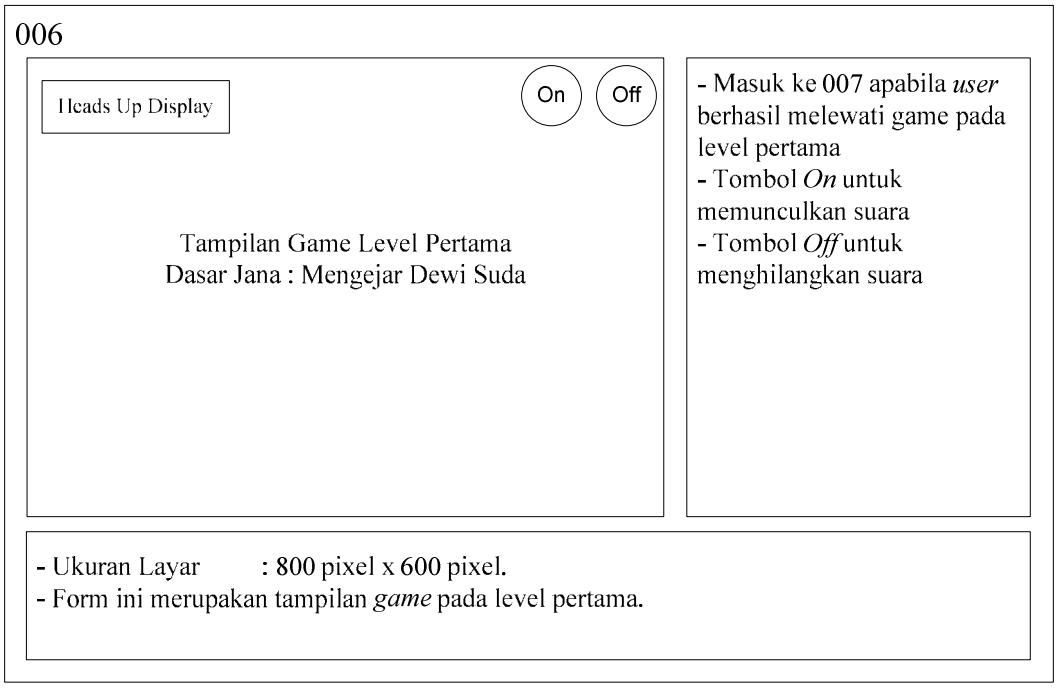
Gambar III.34 Tampilan Skor Tertingi 003



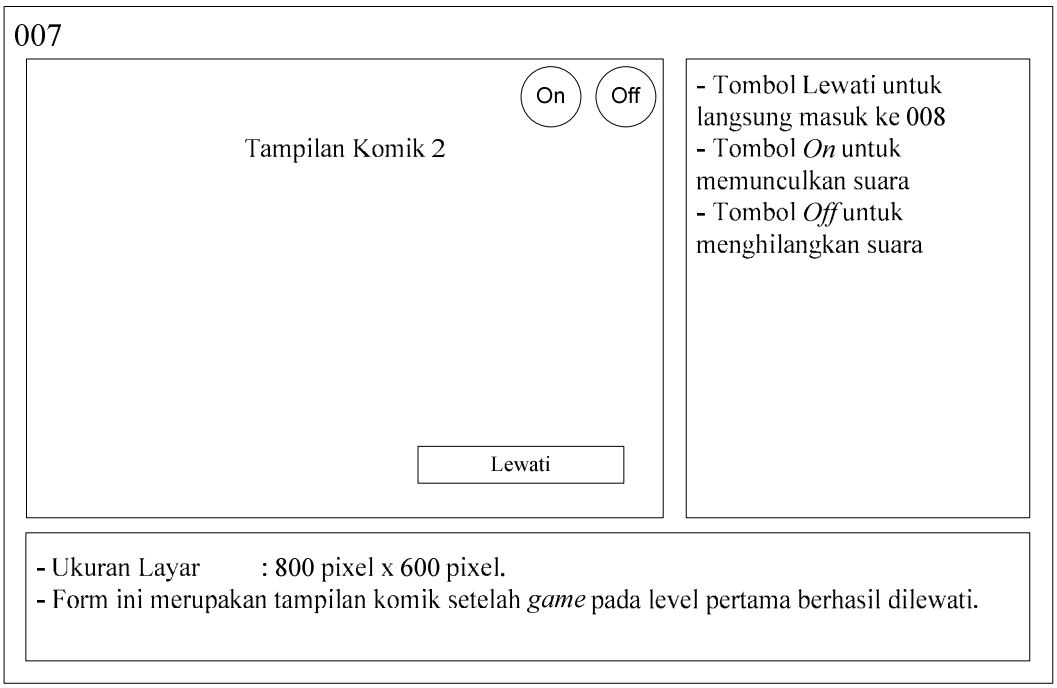
Gambar III.35 Tampilan Petunjuk 004



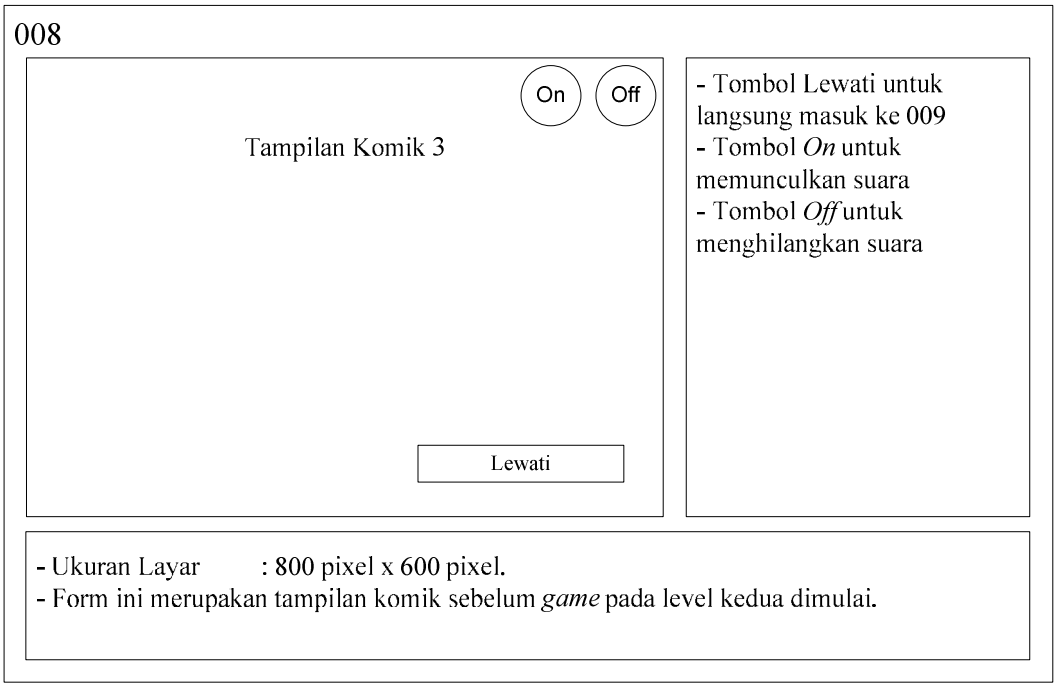
Gambar III.36 Tampilan Info 005



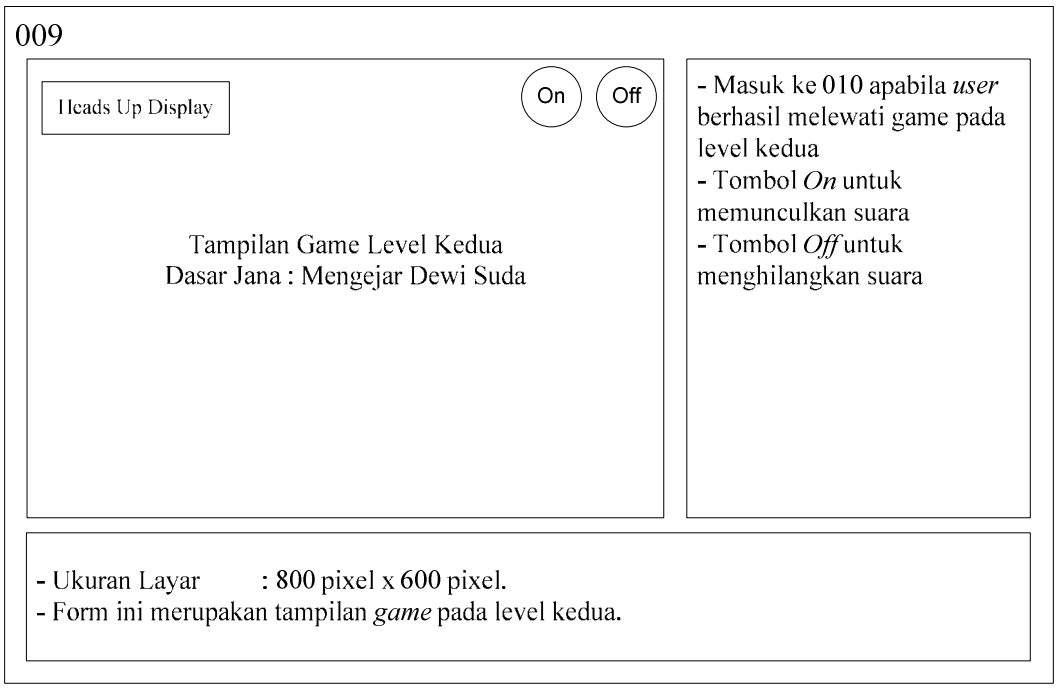
Gambar III.37 Tampilan *Game* Level Pertama 006



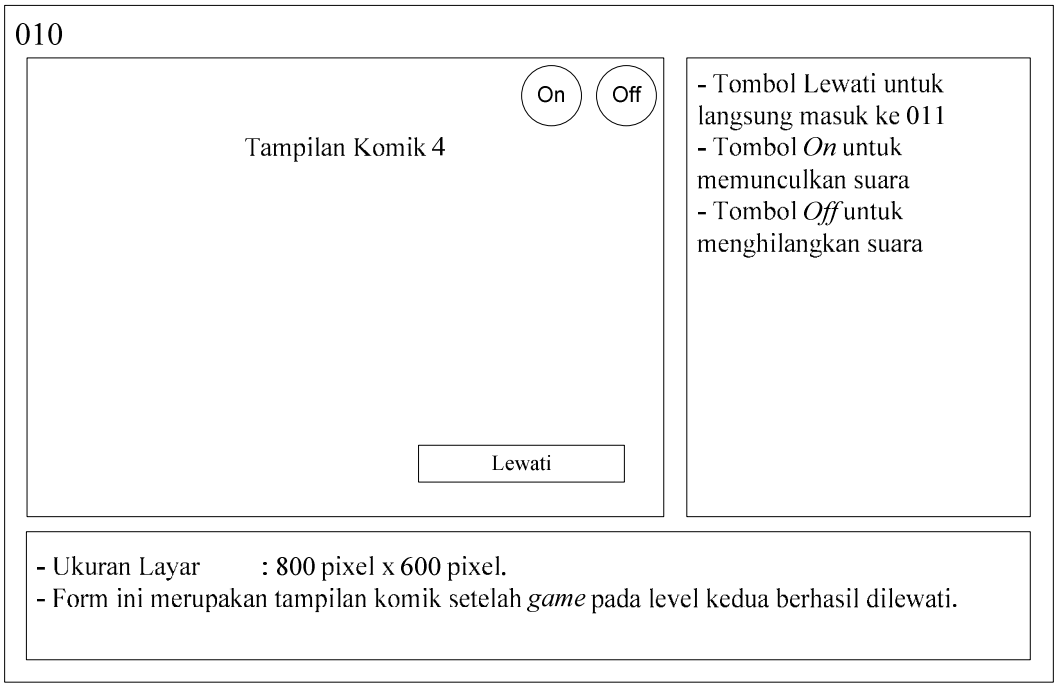
Gambar III.38 Tampilan Komik 2 007



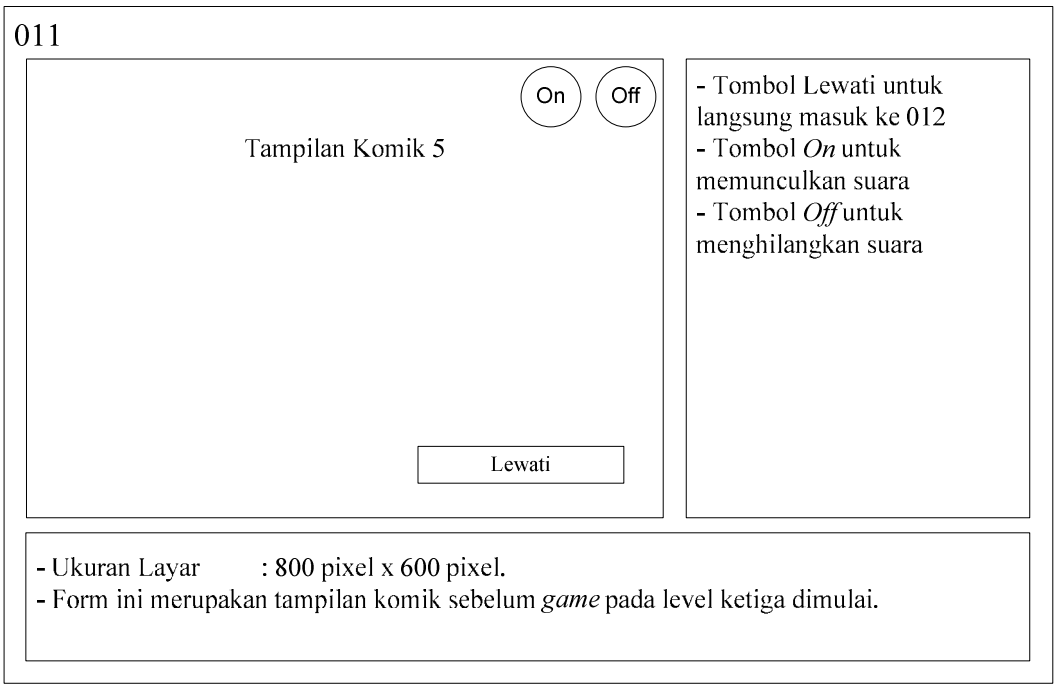
Gambar III.39 Tampilan Komik 3 008



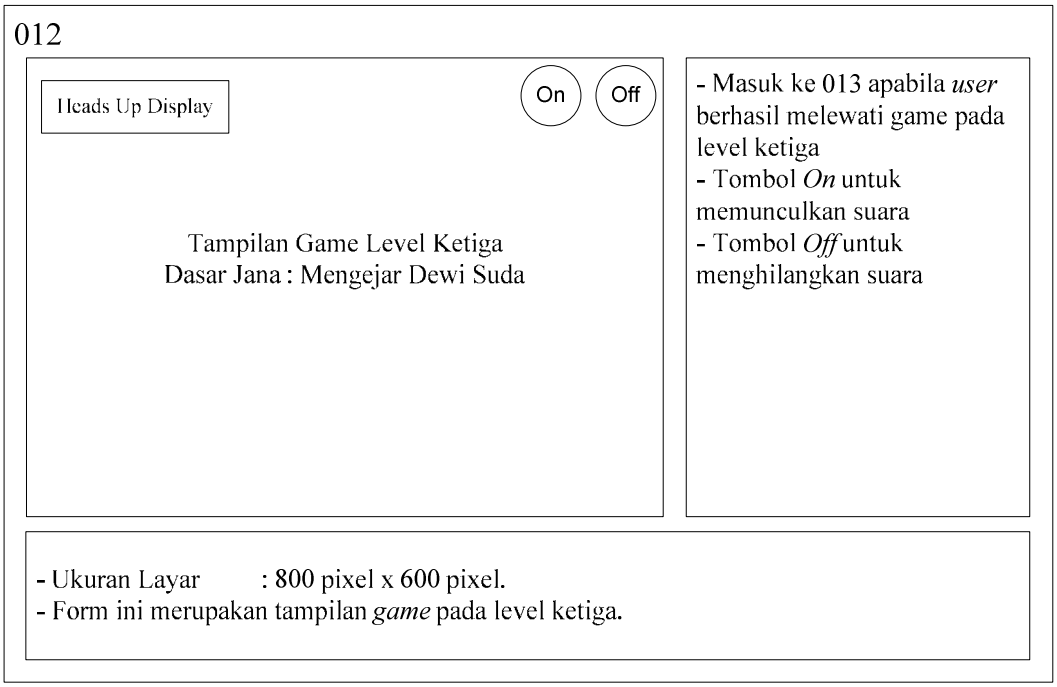
Gambar III.40 Tampilan *Game* Level Kedua 009



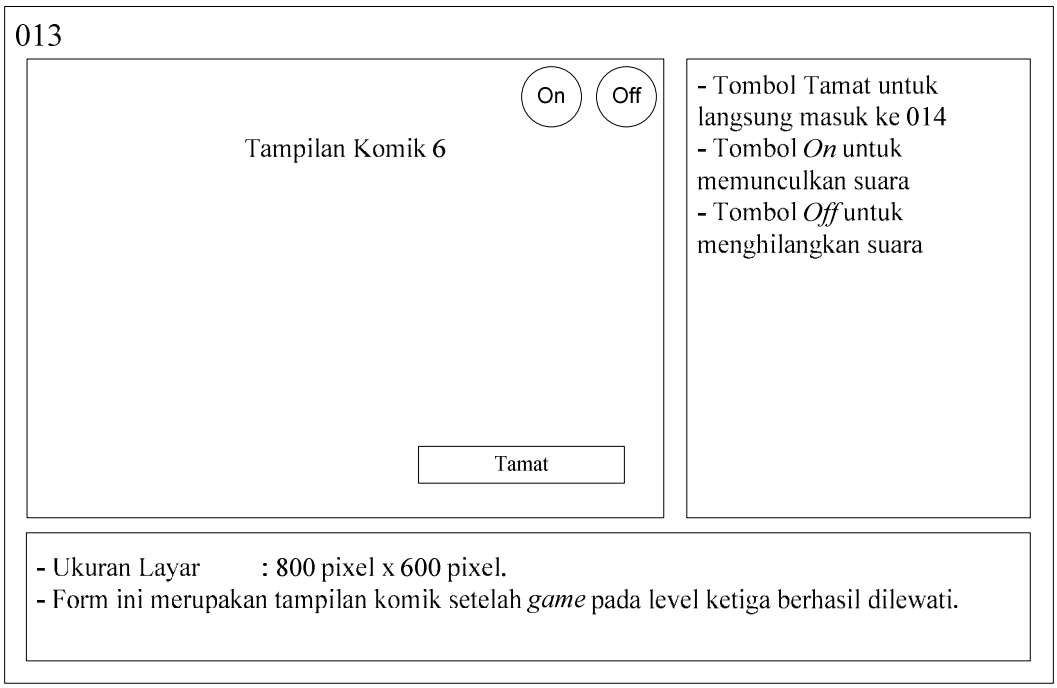
Gambar III.41 Tampilan Komik 4 010



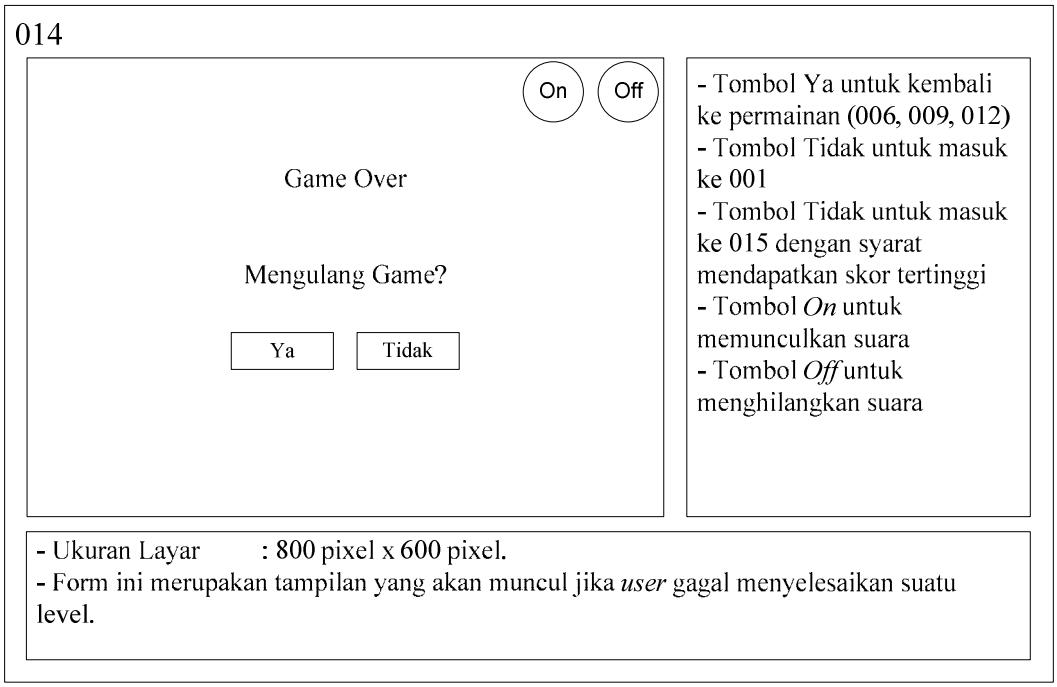
Gambar III.42 Tampilan Komik 5 011



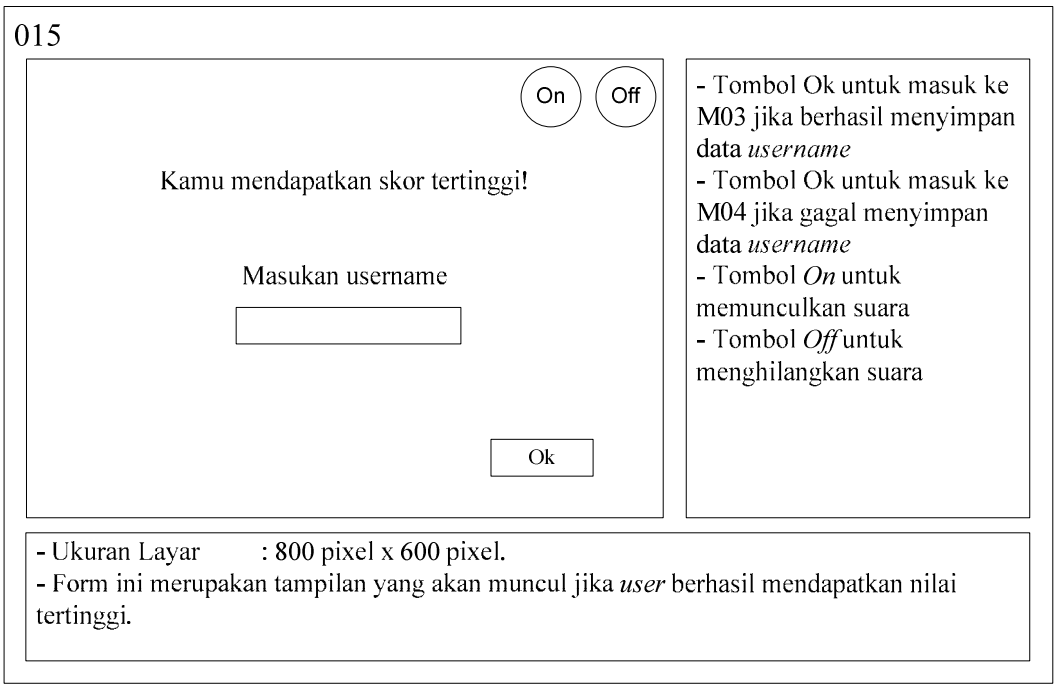
Gambar III.43 Tampilan *Game* Level Ketiga 012



Gambar III.44 Tampilan Komik 6 013



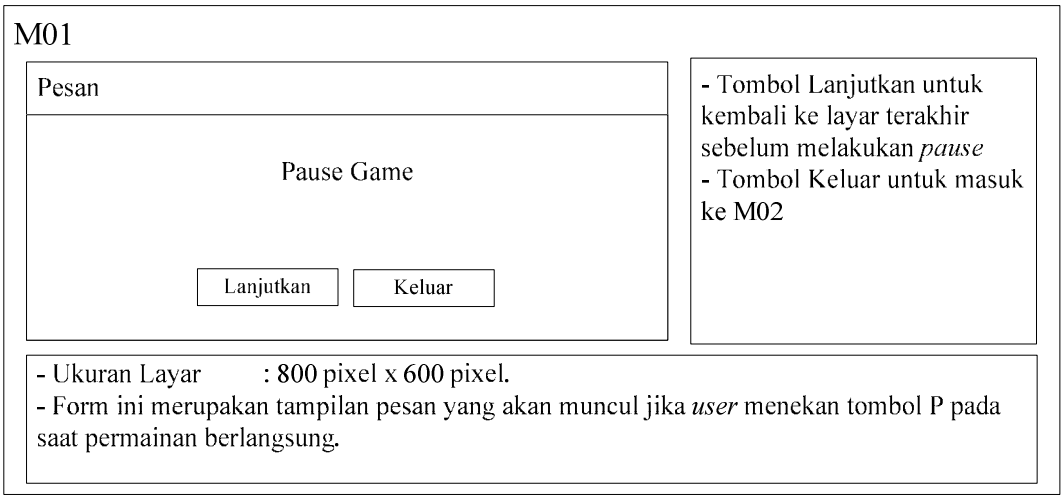
Gambar III.45 Tampilan *Game Over* 014



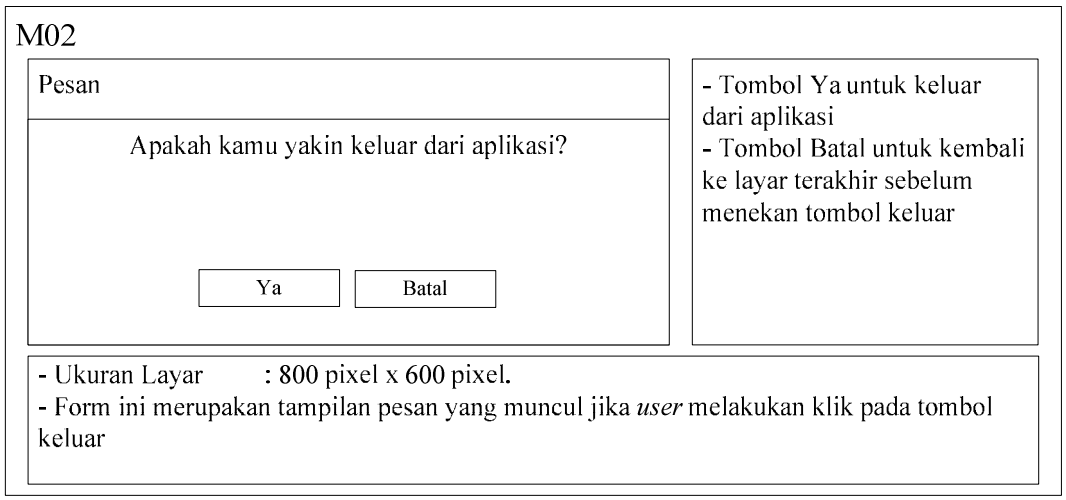
Gambar III.46 Tampilan Masukan *Username* 015

III.2.3 Perancangan Pesan

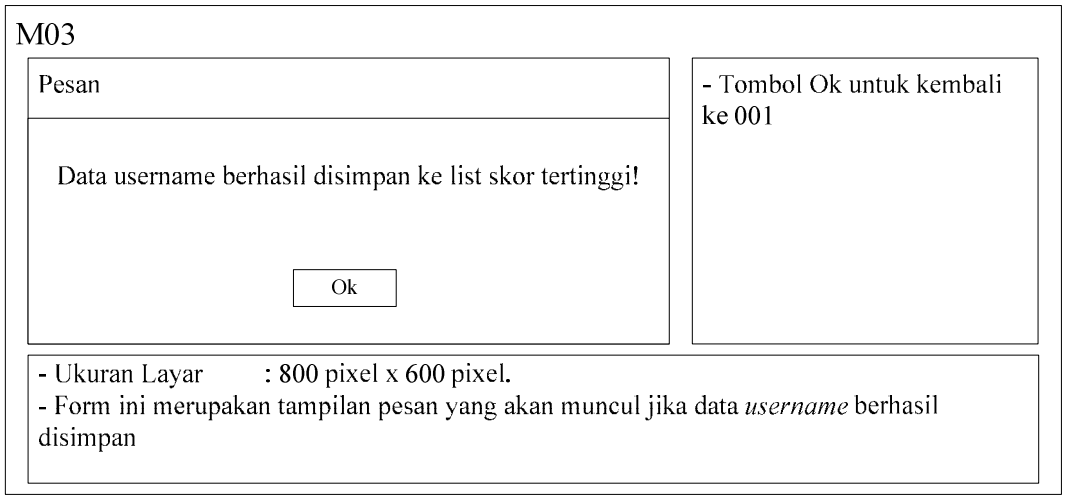
Pesan yang dirancang dalam konteks aplikasi yang dikembangkan merupakan pesan peringatan yang muncul jika *user* melakukan aktivitas tertentu yang akan memicu event tertentu. Perancangan pesan di dalam aplikasi dapat dilihat pada Gambar III.47 sampai dengan Gambar III.50.



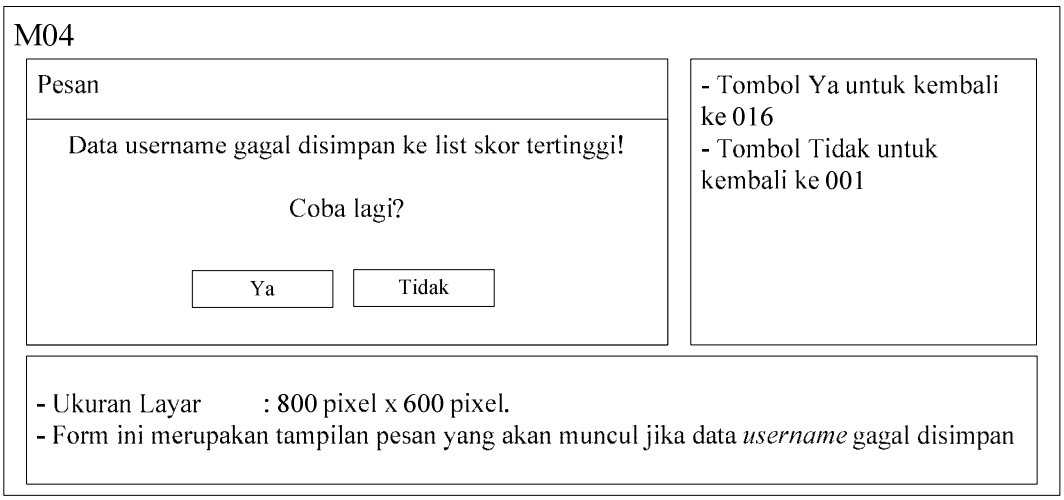
Gambar III.47 Tampilan Pesan *Pause Game* M01



Gambar III.48 Tampilan Pesan *Exit Game* M02



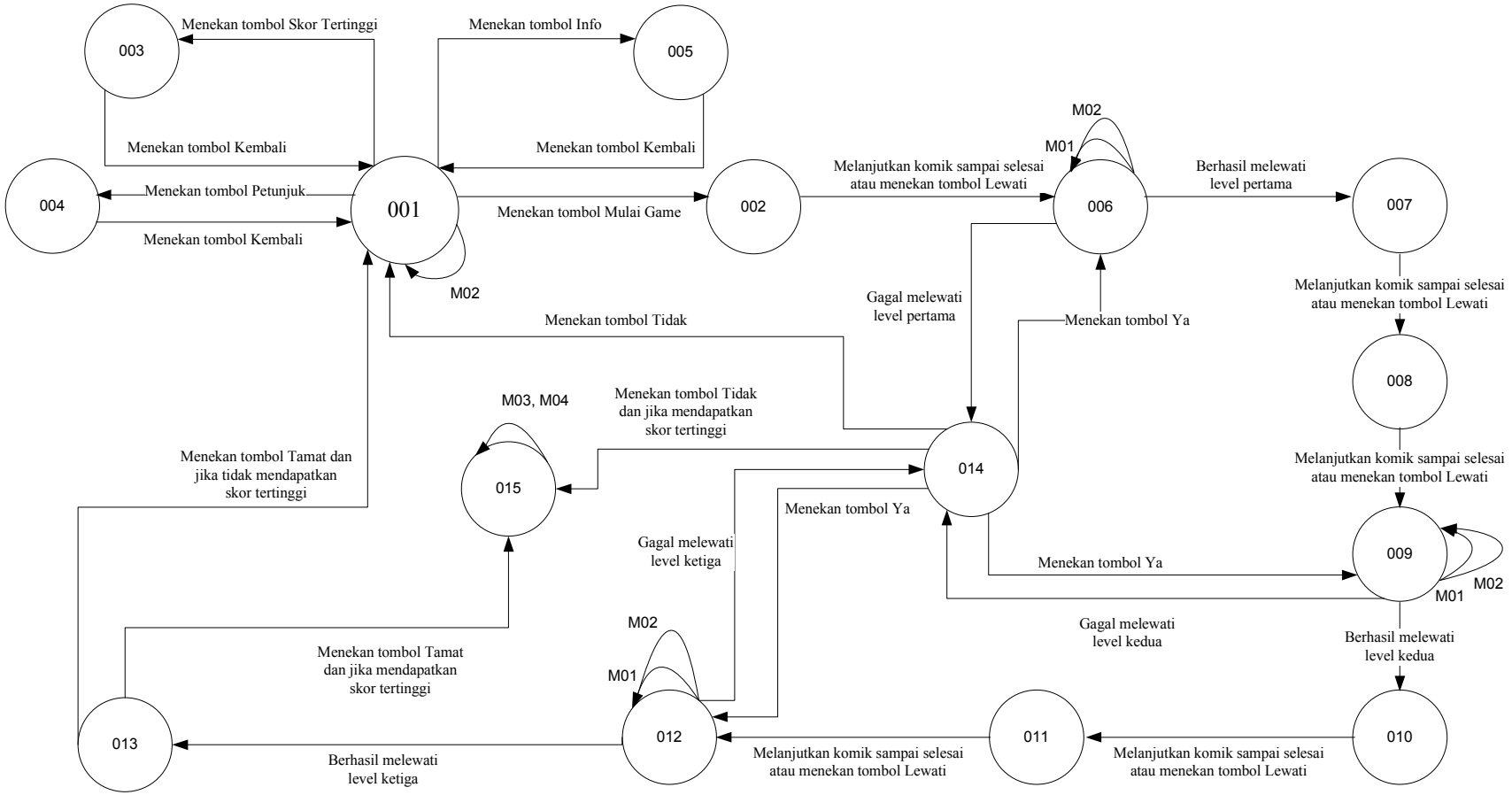
Gambar III.49 Tampilan Pesan Data Berhasil Disimpan M03



Gambar III.50 Tampilan Pesan Data Gagal Disimpan M04

III.2.4 Jaringan Semantik

Jaringan semantik digunakan untuk memperlihatkan hubungan antara perancangan antarmuka dengan perancangan pesan yang telah dibuat sebelumnya. Jaringan semantik aplikasi *game* dapat dilihat pada Gambar III.51.



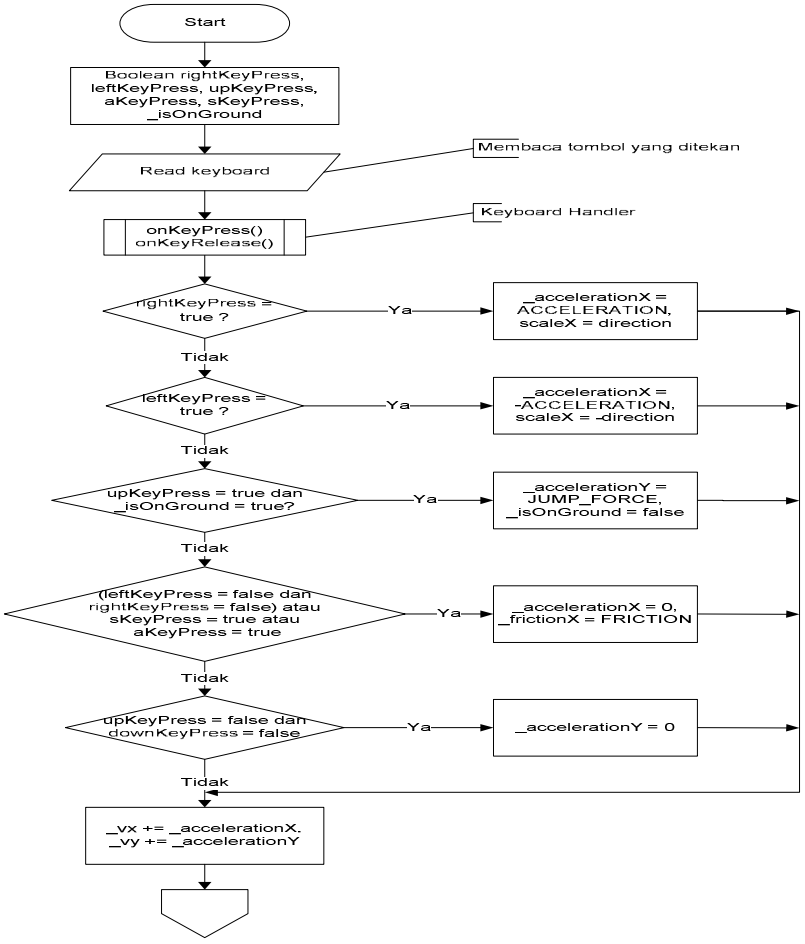
Gambar III.51 Jaringan Semantik Aplikasi *Game*

III.2.5 Perancangan *Method*

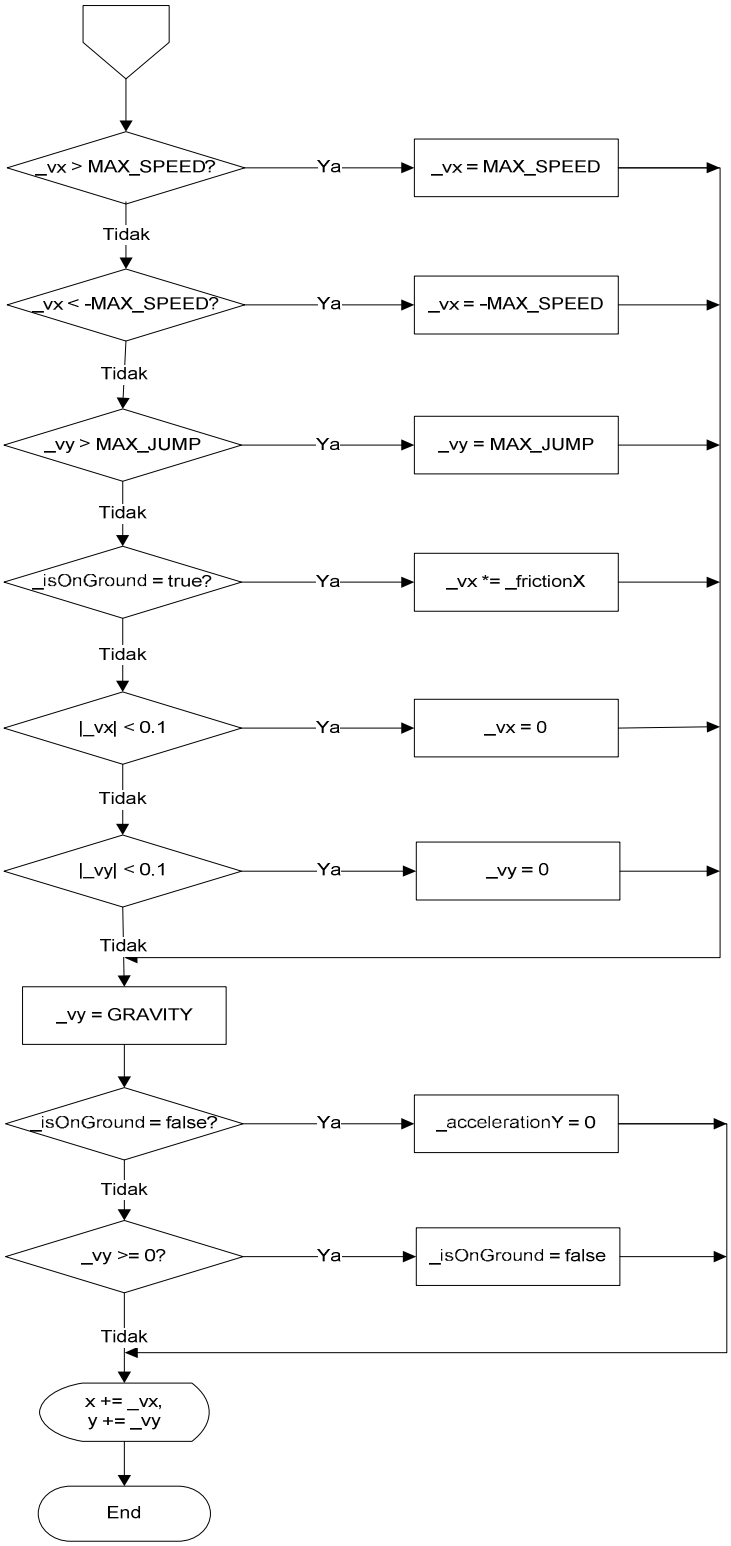
Perancangan *method* merupakan sebuah proses untuk merancang sekumpulan metode yang ada dan berfungsi untuk menjalankan sistem. *Method – method* yang terdapat dalam *game Dasar Jana: Mengejar Dewi Suda* adalah sebagai berikut :

1. *Method* moveCharacter

Method moveCharacter merupakan *method* yang berada pada *class* Player, berfungsi untuk menggerakkan karakter secara matematis berdasarkan inputan dari *keyboard*.



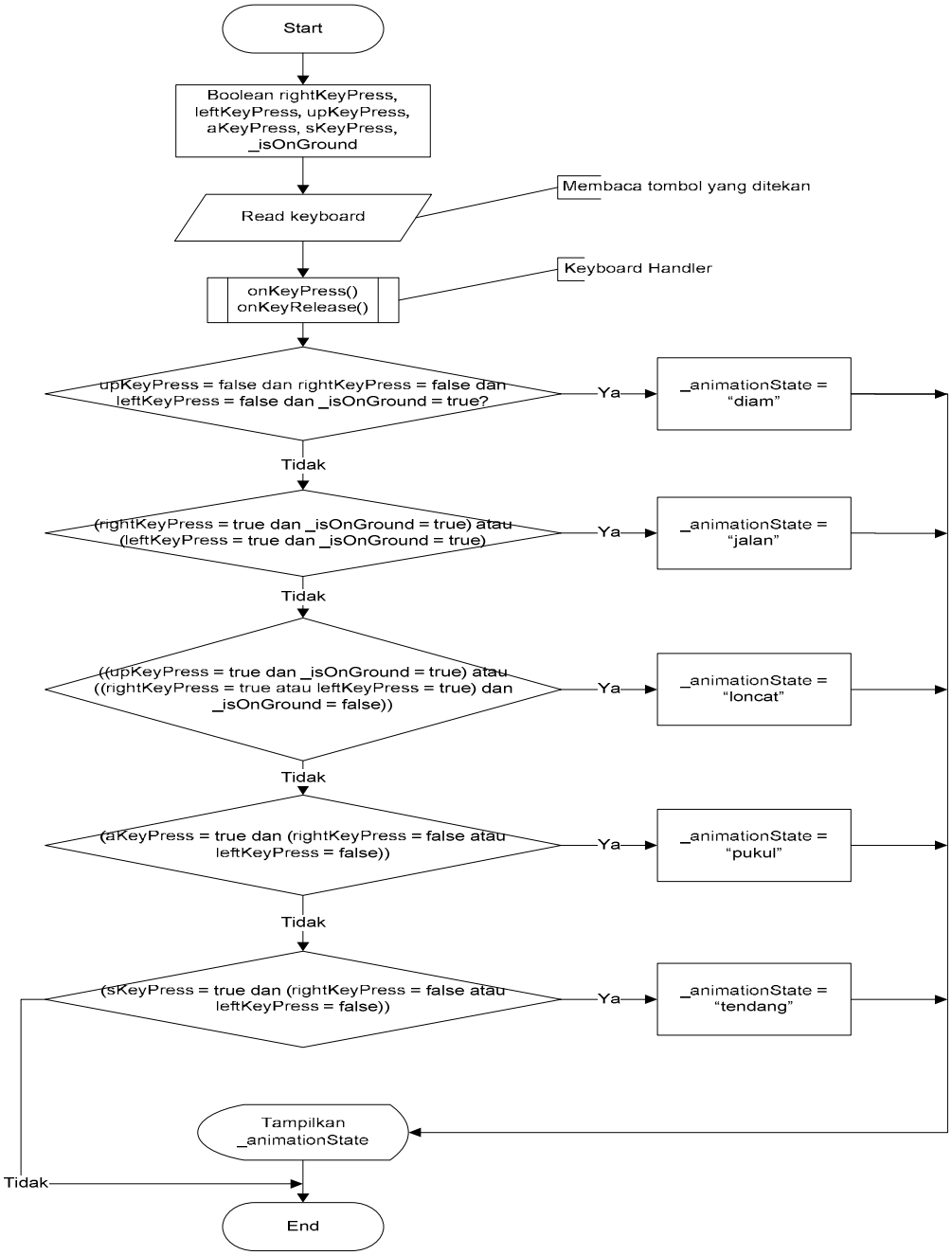
Gambar III.52 *Method* moveCharacter



Gambar III.53 *Method* moveCharacter – Lanjutan

2. *Method* animateCharacter

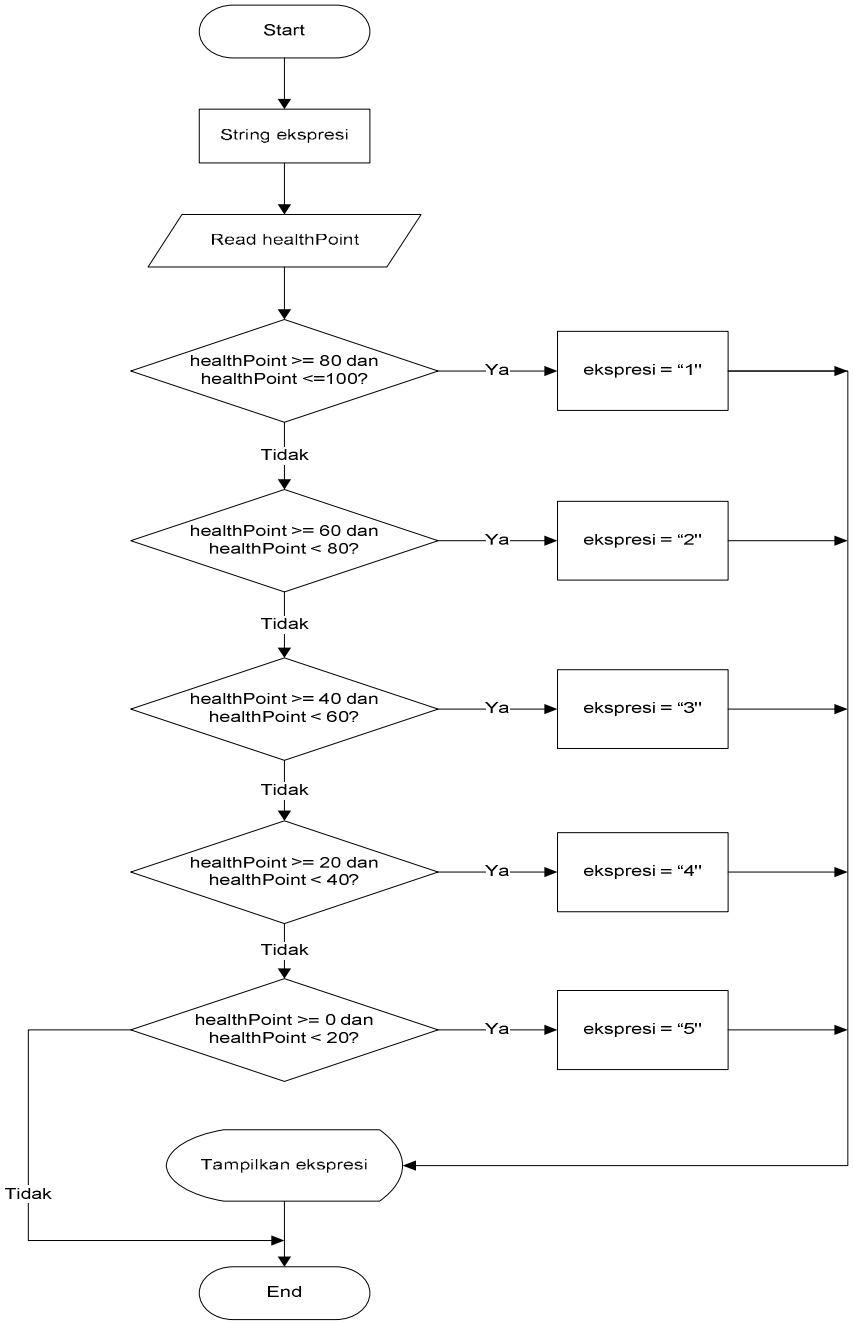
Method animateCharacter merupakan *method* yang berada pada *class* Player, berfungsi untuk menampilkan animasi pergerakan karakter berdasarkan inputan dari *keyboard*.



Gambar III.54 *Method* animateCharacter

3. *Method* showExpression

Method showExpression merupakan *method* yang berada pada *class* HUD, berfungsi untuk menampilkan animasi perubahan ekspresi karakter yang terdapat pada *Heads Up Display* (HUD).



Gambar III.55 *Method* showExpression