BAB 3

ANALISIS DAN PERANCANGAN

3.1 Deskripsi Masalah

3.1.1 *Game Adventure*

Game adventure atau *game* petualangan merupakan *game* yang pemainnya diharuskan untuk bereksplorasi untuk mencapai tujuan tertentu. Tidak ada *combat* ataupun konflik yang berat seperti pertempuran yang luas didalam *game adventure*, hal ini yang membedakan dari *game action* [13]. Terkadang dalam *game adventure*, tantangan hanya untuk mencari tahu apa yang harus dilakukan dan mendapatkan *score* tertinggi. Hal ini merupakan masalah di *game casual*, dimana lingkungan lebih terbatas dan cenderung menghasilkan tugas-tugas yang sering diterapkan oleh *game* pada umumnya dan biasanya item yang diperlukan untuk melakukan tugas tersebut tidak berjarak jauh dengan item yang ditemukan. *Gameplay* maupun *control game* merupakan salah satu kendala juga yang dirasakan oleh *gamer*, karena mereka (*game casual*) cenderung membuat hal yang sama walaupun dengan cerita dan grafik yang berbeda. Hal ini dapat menyebabkan penurunan kualitas permainan karena cenderung monoton. Contohnya seperti *game casual Hedgehog Challenge* yang merupakan salah satu dari *adventure game*.

3.1.2 Konten-Konten Pada Permainan *Hedgehog Callenge*

Konten *game* merupakan faktor penting dalam menjaga pemain agar tidak cepat merasa bosan dan tetap terlibat dalam dunia *game* [2]. Berikut adalah konten-konten yang terdapat pada permainan *game Hedgehog Challenge* dapat dilihat dari tabel 3.1 :

Tabel 3.1 Konten-konten pada *game Hedgehog Challenge*

No	Nama	Gambar	Penjelasan
1	*Hedgehog* / Pemain		*Hedgehog* merupakan karakter utama yang dapat dikendalikan oleh pemain melalui intputan *keyboard*
2	*Pads* (*platform*)		*Pads* merupakan *platform-platform* yang berfungsi sebagai pijakan karakter utama yang di*control* oleh pemain untuk dapat meloncat dan mendapatkan *score*
3	*Clock timer*		*Clock timer* disini merupakan waktu yang berfungsi untuk menambah laju turunnya *pads*.
4	*Clock speed up*		*Clock speed up* berfungsi sebagai tanda dimana laju turunnya *pads* yang bertambah.

5	*Clock max*		*Clock max* merupakan laju maksimum turunnya *pads*
6	*Owl Pads*		*Owl pads* pada *game* ini berfungsi sebagai informasi tingkat *pads* terjauh yang sedang dicapai.
7	*Score*		*Score* pada *game* ini merupakan hasil penjumlahan dari *pads* yang telah dicapai dan bonus *combo* yang telah dibuat oleh pemain

3.1.3 Tampilan *Gameplay* Pada Permainan *Hedgehog Challenge*

Tampilan *game* merupakan *interface* yang akan membuat pengguna merasa tertarik dalam memainkannya. Berikut ini merupakan tampilan *gameplay* permainan *Hedgehog Challenge*.



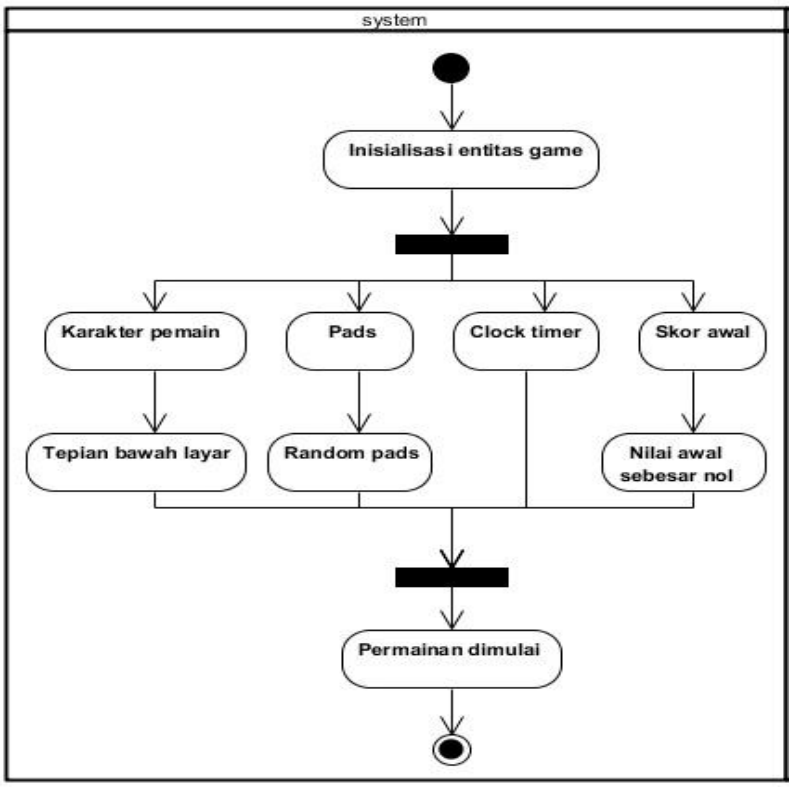
Gambar 3.1 Tampilan *Gameplay Hedgehog Challenge*

3.1.4 Skenario Permainan Pada *Hedgehog Challenge*

Skenario permainan pada *Hedgehog Challenge* dapat disusun oleh beberapa hal sebagai berikut :

1. Alur Sistem

Sistem akan menginisialisasi seluruh entitas yang diperlukan untuk memulai permainan seperti karakter pemain, pads, inisialisasi *score* awal yang masih bernilai nol, dan *clock timer*. Karakter pemain diletakan di tepian bawah layar sebagai area atau tempat pijakan pertama sebelum permainan dimulai. *Pads* yang merupakan *platform* sebagai pijakan karakter untuk melompat di tempatkan secara acak oleh sistem. Setelah seluruh entitas yang diperlukan sudah di inisialisasi, maka permainanpun dapat segera dimulai. Untuk lebih jelasnya, dapat dilihat dari *activity diagram* pada gambar 3.2 :



Gambar 3.2 *Activity Diagram* Alur Sistem *Hedgehog Challenge*

2. *Control* Permainan

Tabel 3.2 *Control* Permainan pada *Hedgehog Challenge*

***Control* Permainan**
1. *Keyboard* merupakan alat input untuk mengontrol karakternya. 2. Pemain dapat meloncat terus dengan menekan spasi pada *keyboard*. 3. Pemain dapat mengontrol karakter ke kiri atau ke kanan dengan menggunakan panah kiri dan kanan yang terdapat pada *keyboard*.

3. Misi Permainan

Tabel 3.3 Misi Permainan pada *Hedgehog Challenge*

Misi Permainan
Pemain harus mendapatkan *score* tertinggi dalam setiap permainannya dengan cara menciptakan *combo* yang beruntun dan jumlah *pads* yang berhasil dicapai.

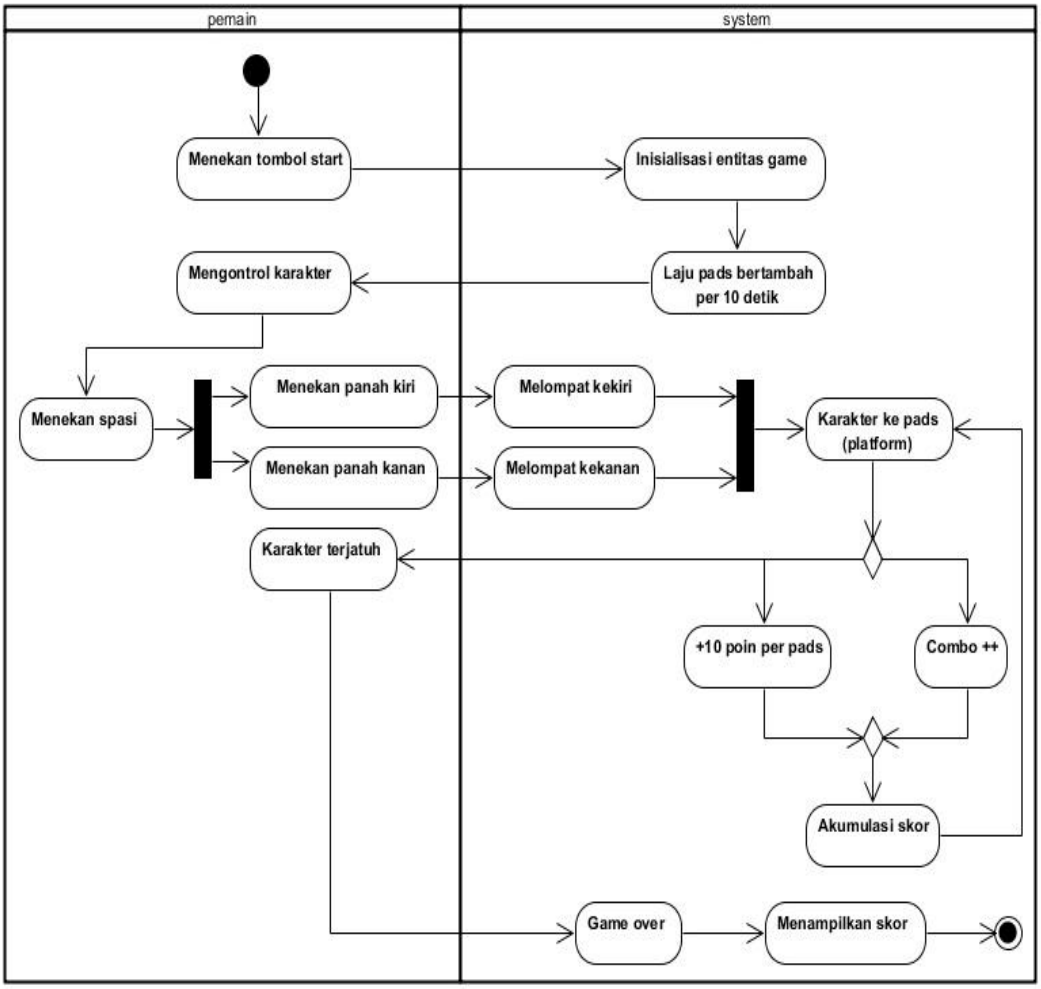
4. *Gameplay Hedgehog Challenge*

Tabel 3.4 *Gameplay Hedgehog Challenge*

Gameplay
1. Pemain berada di tepian paling bawah layar. 2. Pemain harus dapat melompat terus tanpa berhenti dari satu *platform* ke *platform* lain dan berusaha agar hedgehog tetap berada di atas *pads*. 3. Pemain dapat menciptakan *combo* dengan cara menekan dan menahan spasi, dan menekan salah satu arah panah. 4. Bonus *score* akan menjadi lebih besar apabila pemain dapat menciptakan *combo* yang beruntun.

5. *Pads* akan terus turun hingga menghilang ditepan bawah dengan laju yang konstan.
6. Setiap per sepuluh detik, laju *pads* akan menjadi lebih cepat hingga kecepatan maximum.
7. Permainan akan berakhir apabila pemain terjatuh hingga menghilang dari layar.

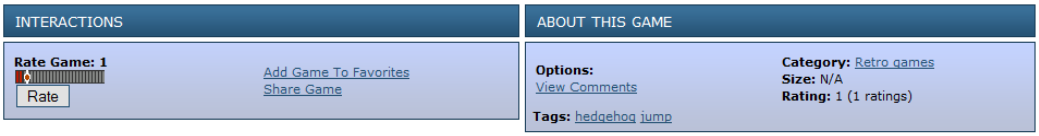
Untuk dapat melihat lebih jelas *gameplay*-nya, di bawah ini merupakan *activity diagram* yang menunjukan *gameplay* dari *game HedgehogChallenge:*



Gambar 3.3 *Activity diagram* Permainan pada *Hedgehog Challenge*

5. Kekurangan permainan

Dari deskripsi masalah di atas, maka diketahui bahwa *game Hedgehog Challenge* kurang menarik dan terkesan monoton yang dapat membuat pengguna merasa cepat bosan dan jenuh untuk kembali memainkannya.Hal ini dibuktikan dari situs *java-gaming.com* bahwa rating untuk permainan *hedgehog challenge* hanya mendapatkan nilai terendah yaitu satu dari lima besaran rating tertinggi.



Gambar 3.4 *Rating Hedgehog Challenge*

Untuk itu, maka penulis merasa tertarik untuk merancang dan membangun aplikasi *game casual adventure* yang sama dengan *gameplay Hedgehog Challenge* dengan *taste* yang berbeda.

3.2 Analisis *Game* Lutung Kasarung *The jumper*

3.2.1 Deskripsi *Game*

Game yang akan dibangun merupakan *game* yang berjenis *adventure* dan *gameplay*-nya hampir sama dengan *game Hedgehog Challenge*. Karakter pada *Game* ini diambil dari salah satu cerita rakyat Jawa Barat yaitu Lutung kasarung. Selain itu, *Game* ini dibangun dengan mengaplikasikan teknologi sensor *accelerometer* sebagai *control* karakter dan menerapkan metode *Procedural content generation* (*PCG*) untuk penyebaran konten-konten yang dibutuhkan.

Berikut ini adalah fitur-fitur yang ada pada *game* ini :

1. Sistem *single player*

2. Karakter utama yang diambil dari cerita rakyat Jawa Barat
3. Karakter utama memiliki kemampuan untuk terus melompat tanpa berhenti
4. Memiliki 3 (tiga) *level* yang berkesesuaian dengan tingkat kesulitannya
5. Disetiap *level* yang sama, penempatan kontennya akan berbeda
6. *Control* karakter hanya memiringkan *smartphone* ke kanan dan ke kiri
7. *Grafik* yang digunakan merupakan *grafik game* 2D

3.2.2 *Storyline*

Lutung kasarung adalah sebuah cerita rakyat yang sangat terkenal dikalangan masyarakat Sunda, Jawa Barat, Indonesia. Cerita ini mengisahkan perjalanan Sanghyang Guruminda dari khayangan ke bumi dalam wujud seekor lutung, yaitu kera hitam berekor panjang. Ketika sampai di bumi, ia tersesat ditengah hutan. Itulah sebabnya ia dipanggil lutung kasarung, yaitu lutung yang tersesat. Entah berapa lama lutung tersesat didalam hutan, ia terus bergerak melompat dari dahan satu ke dahan yang lainnya untuk dapat keluar dari hutan itu. Didalam perjalanannya, ia bertemu dengan seorang putri bernama Purbasari yang juga merupakan misi dari *game* ini. Untuk menyelesaikan misi, lutung harus menjelajahi lokasi-lokasi yang memiliki tiga tingkatan / tiga *level* dengan kesulitan-kesulitan yang berbeda disetiap *level*nya.

3.2.3 *Gameplay*

1. *Gameplay level* 1

Tabel 3.5 *Gameplay Level* 1

No	*Gameplay*
1	Karakter berada di tepian bawah layar
2	Karakter memiliki kemampuan untuk melompat terus dari *platform* satu ke *platform* lainnya
3	Pemain mengendalikan karakter dengan memiringkan ponsel ke kiri atau ke kanan
4	Pemain dapat mengumpulkan poin dengan mengambil pisang yang terdapat didalam *game*
5	Pemain harus berhati-hati karena terdapat *platform* dengan kemunculan acak yang dapat bergerak horizontal dan menghilang setelah karakter menginjaknya
6	Pemain dapat menggunakan *spring* dengan kemunculan acak yang dapat membuat karakter melompat 1,5 lebih tinggi dari lompatan normal
7	Pemain harus berusaha menghindar dari sengatan lebah yang bergerak ke kanan dan ke kiri dengan kecepatan konstan.
8	Permainan akan berakhir apabila karakter telah menyentuh pintu level 2 atau karakter terjatuh dan menghilang di tepian bawah layar

2. *Gameplay level* 2

Tabel 3.6 *Gameplay Level* 2

No	*Gameplay*
1	Karakter berada di tepian bawah layar
2	Karakter memiliki kemampuan untuk melompat terus dari *platform* satu ke *platform* lainnya
3	Pemain mengendalikan karakter dengan memiringkan ponsel

	ke kiri atau ke kanan
4	Pemain dapat mengumpulkan poin dengan mengambil pisang yang terdapat didalam *game*
5	Pemain harus berhati-hati karena terdapat *platform* dengan kemunculan acak yang dapat bergerak horizontal dan menghilang setelah karakter menginjaknya
6	Pemain dapat menggunakan *spring* dengan kemunculan acak yang dapat membuat karakter melompat 1,5 lebih tinggi dari lompatan normal
7	Pemain harus berusaha menghindar dari sengatan lebah yang bergerak ke kanan dan ke kiri dengan kecepatan konstan.
8	Pemain harus berhati-hati dengan bunga beracun yang terdapat di tengah platform statis.
9	Pemain harus berusaha agar tidak terjatuh dan berusaha agar tetap berada di atas *platform* sebagai pijakan
10	Permainan akan berakhir apabila karakter telah menyentuh pintu level 3 atau karakter terjatuh dan menghilang di tepian bawah layar

3. *Gameplay level* 3

Tabel 3.7 *Gameplay Level* 3

No	***Gameplay***
1	Karakter berada di tepian bawah layar
2	Karakter memiliki kemampuan untuk melompat terus dari *platform* satu ke *platform* lainnya
3	Pemain mengendalikan karakter dengan memiringkan ponsel ke kiri atau ke kanan
4	Pemain dapat mengumpulkan poin dengan mengambil pisang yang terdapat didalam *game*
5	Pemain harus berhati-hati karena terdapat *platform* dengan

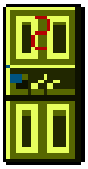
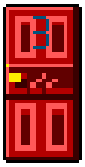
	kemunculan acak yang dapat bergerak horizontal dan menghilang setelah karakter menginjaknya
6	Pemain dapat menggunakan *spring* dengan kemunculan acak yang dapat membuat karakter melompat 1,5 lebih tinggi dari lompatan normal
7	Pemain harus berusaha menghindar dari sengatan lebah yang bergerak ke kanan dan ke kiri dengan kecepatan konstan.
8	Pemain harus berhati-hati dengan bunga beracun yang terdapat di tengah platform statis.
9	Pemain harus berhati-hati dengan kemunculan sekumpulan lebah yang bergerak dari atas ke bawah dan harus menghindarinya.
10	Pemain harus berusaha agar tidak terjatuh dan berusaha agar tetap berada di atas *platform* sebagai pijakan
11	Permainan akan berakhir apabila karakter telah menyentuh purbasari atau terjatuh dan menghilang di tepian bawah layar, dan terkena sengatan lebah

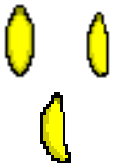
3.2.4 Konten-Konten *Game*

Konten *game* merupakan faktor penting dalam menjaga pemain agar tidak cepat merasa bosan dan tetap terlibat dalam dunia *game* [2]. Berikut adalah konten-konten yang akan digunakan dalam pembangunan aplikasi *game* Lutung kasarung *The jumper* :

Tabel 3.8 Konten-konten Pada *Game* Lutung *The jumper*

No	Nama	Gambar	Penjelasan
1	Lutung kasarung		Lutung kasarung merupakan tokoh utama dalam *game* yang akan dibangun dan dapat di*control* oleh

			pemain. Gambar disamping menunjukan gambar dengan lima *frame* animasi yang dibuat semenarik mungkin agar dapat menimbulkan kesan lucu dan unik. Pada gambar di atas terdapat dua *frame* untuk ekspresi melompat, dua frame untuk ekspresi terjatuh, dan satu frame lagi untuk ekspresi mati. Maka disini terlihat bahwa lutung memiliki tiga *state*, yaitu melompat, jatuh, dan menjadi mati. Masing-masing *frame* memiliki background yang transparan.
2	Pintu *Level 2*		Pintu level ini berwarna hijau dan bertuliskan 2 (dua), maksudnya pintu ini sebagai tanda dimana level akan berlanjut dari level 1 ke level 2. Konten ini terdapat pada *level* 1 saja.
3	Pintu *Level 3*		Pintu level ini berwarna merah dan bertuliskan 3 (tiga), maksudnya pintu ini sebagai tanda dimana level akan berlanjut dari level 2 ke level 3. Konten ini terdapat pada *level* 2 saja.
4	Purbasari		Purbasari merupakan tokoh karakter yang akan dicapai dalam permainan ini. Jika pemain sampai pada tempat purbasari berada, maka misi telah berhasil dicapai. Konten ini hanya terdapat pada level 3 saja.
5	*Platform*		Gambar disamping menunjukan gambar *platform* dengan empat frame animasi *platform* yang lenyap atau menghilang. *Platform* ini merupakan

			tempat pijakan karakter untuk setiap lompatannya. Konten ini terdapat pada *level* 1, *level* 2, dan *level* 3.
6	Pisang (poin)		Gambar disamping menunjukan gambar pisang dengan tiga frame animasi yang tampak seperti sedang menari agar lebih menarik penglihatan pemain untuk mendekatinya. Pisang disini berfungsi sebagai poin yang akan didapatkan oleh pemain. Konten ini terdapat pada *level* 1, *level* 2, dan *level* 3.
7	Pelontar (spring)		Konten ini berfungsi untuk membuat lutung kasarung dapat melompat satu setengah kali lebih tinggi dari lompatan normal. Konten ini terdapat pada *level* 1, *level* 2, dan *level* 3.
8	Pelontar (spring)		Bunga beracun berfungsi untuk membuat lutung menjadi kondisi mati apabila lutung meyentuhnya. Letak bunga beracun terdapat di tengah *platform* yang statis. Karenanya, lutung masih dapat melompat di pinggiran *platform* sebagai pijakan. Konten ini terdapat pada *level* 2 dan *level* 3.
9	Lebah Terbang (enemy)		Gambar dissamping menunjukan gambar dengan dua frame animasi yang mengibaskan sayap. Lebah disini merupakan karakter jahat, apabila lutung mengenainya maka lutung akan menjadi kondisi mati. Dia akan terbang bolak-balik secara *vertical*

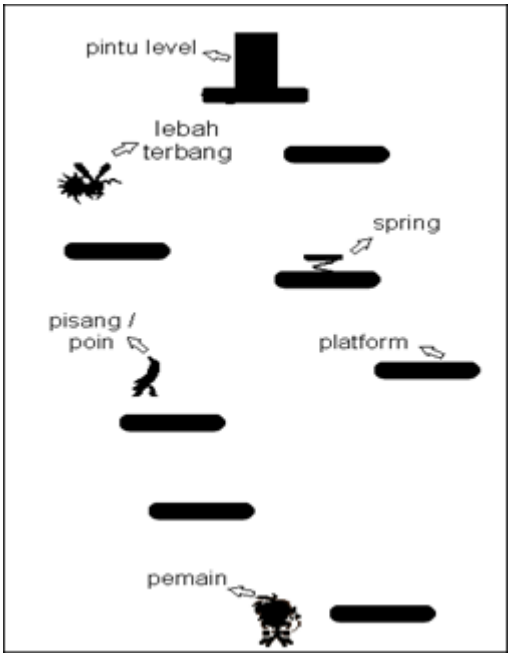
			dengan kecepatan yang konstan. Konten ini terdapat pada *level* 2 dan *level* 3.
10	Sekumpulan Lebah (enemy)		Konten ini merupakan karakter jahat yang akan mengganggu pemain dalam menyelesaikan permainan. Dia akan terbang menurun secara berkelompok maupun per-ekor secara *horizontal* dengan kecepatan yang konstan dan apabila lutung mengenainya, maka lutung akan menjadi kondisi mati. Konten ini hanya terdapat pada level 3 saja.

3.2.5 *Story Board*

Story board adalah sketsa gambar yang disusun berurutan sesuai dengan naskah, dengan *story board* kita dapat menyampaikan ide cerita kita kepada orang lain dengan lebih mudah, karena kita dapat menggiring khayalan seseorang mengikuti gambar-gambar yang tersaji, sehingga menghasilkan persepsi yang sama pada ide cerita kita. Berikut merupakan *story board* dari *game* lutung kasarung *the jumper* :

1. *Story Board Level* 1

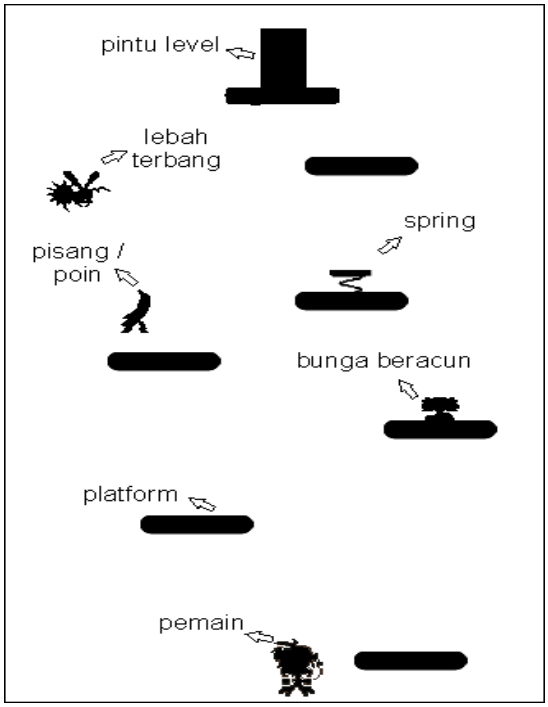
Story board untuk *level* 1 pada aplikasi ini dapat dilihat sebagai berikut :



Gambar 3.5 *Story board Level* 1

2. *Story Board Level* 2

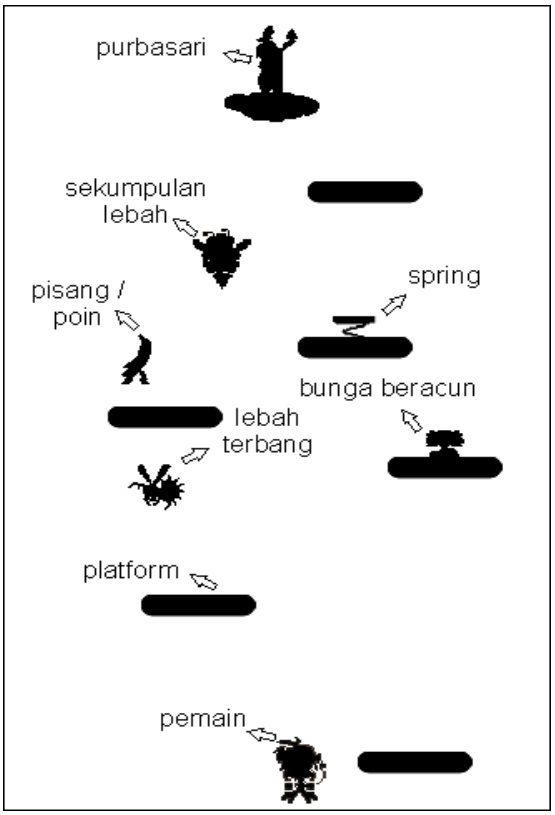
Story board untuk *level* 2 pada aplikasi ini dapat dilihat seagai berikut :



Gambar 3.6 *Story board* Sub *Level* 2

3. *Story Board Level* 3

Story board untuk *level* 3 pada aplikasi ini dapat dilihat sebagai berikut :



Gambar 3.7 *Story board* Sub *Level* 3

3.3 Penerapan *Accelerometer* dan *Procedural content generation*

Accelerometer merupakan salah satu teknologi sensor yang akan di terapkan untuk *control* karakter dalam *game* yang akan dibangun dan *Procedural content generation* sebagai metode untuk penyebaran konten-konten yang dibutuhkan.

3.3.1 Penerapan *Accelerometer*

Dalam penerapannya, dibutuhkan perhitungan untuk mendefinisikan kecepatan dan percepatan dalam permainan dan *AccelerometerHandler class* untuk penanganan *accelerometer*nya.

1. Perhitungan Kecepatan dan Percepatan

Nilai-nilai *accelerometer* terkadang akan mendapatkan nilai yang lebih dari jangkauan besaran *accelerometer*. Hal ini disebabkan karena ketidakakuratan nilai-nilai yang di laporkan oleh *Listener*, sehingga harus melakukan perhitungan nilai-nilainya agar dapat sesuai [11]. Berikut adalah perkiraan besaran percepatan dan kecepatan yang akan digunakan dalam pembangunan *game* lutung kasarung *the jumper* :

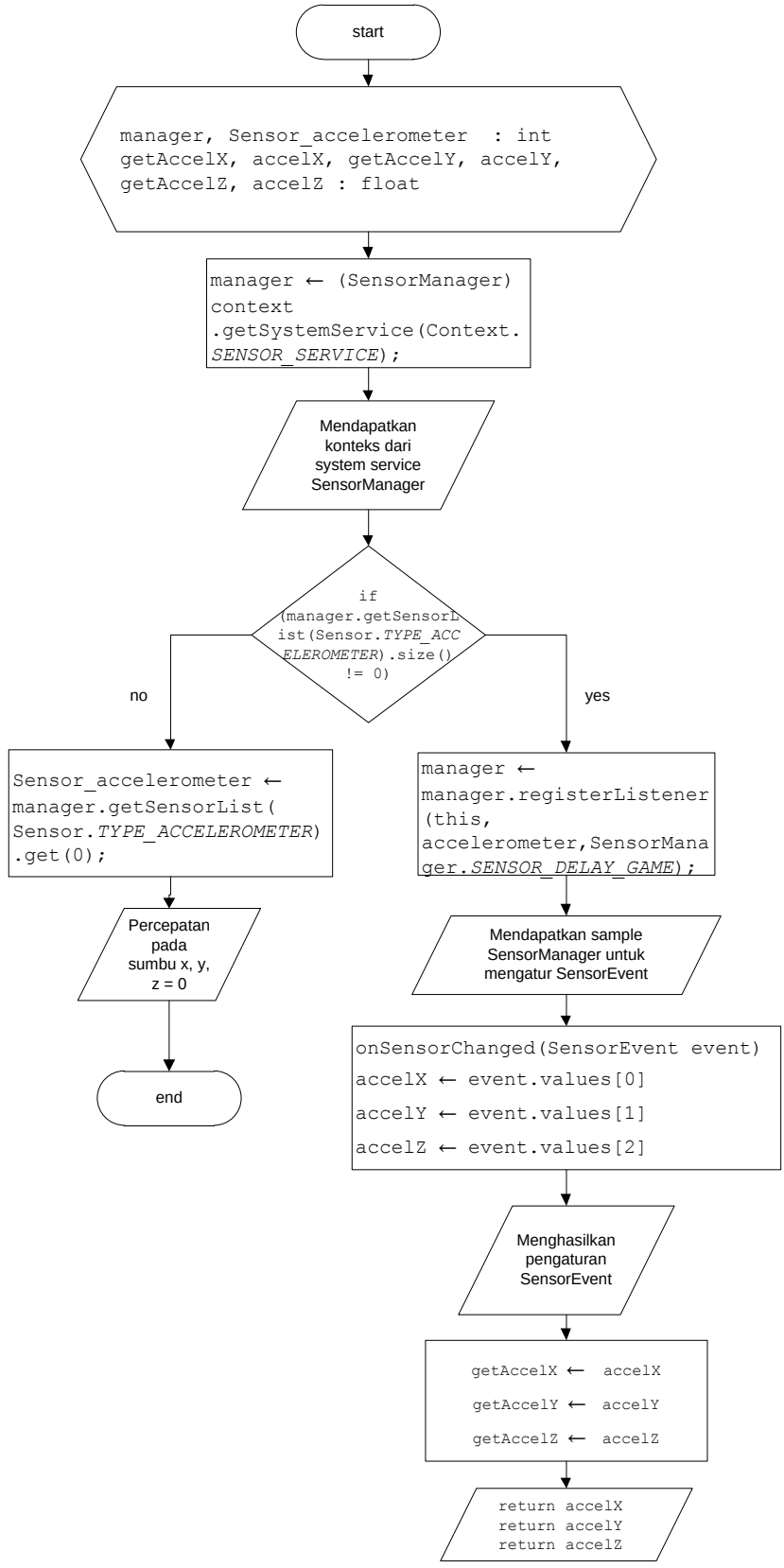
Tabel 3.9 Perkiraan Besaran Kecepatan dan Percepatan

No	Besaran kecepatan (*velocity*)
1	Percepatan gravitasi sebesar 0-13 m/s. sedikit lebih besar dari dunia nyata, hal ini dikarenakan agar lutung kasarung terlihat normal ketika melompat yang dipengaruhi dengan gravitasi
2	Ketika lutung kasarung melompat, didefinisikan kecepatan awalnya adalah sekitar 0-11m/s
3	Ketika lutung kasarung melompat dengan pijakan benda pegas, maka kecepatan saat melompat didefinisikan sebesar 1,5 kali dari lompatan normal
4	Ketika sepenuhnya miring ke kiri maka sumbu x akan melaporkan percepatan sebesar -10 m/s^2 dan ketika sepenuhnya miring ke kanan maka sumbu x akan melaporkan percepatan sebesar 10 m/s^2
5	Kecepatan untuk gerakan horizontal lutung kasarung didefinisikan sebesar 0-20 m/s^2

Semua perhitungan digunakan untuk menormalkan pembacaan *accelerometer* dengan membagi nilai *absolute* maksimum yaitu 10 dan kemudian dikalikan dengan kecepatan maksimum *horizontal* lutung kasarung. Dengan demikian, maka akan didapat percepatan sebesar 20 m/s di sumbu-x. Setiap lompatan yang dilakukan lutung kasarung, merupakan hasil penggabungan dari kondisi nilai gerakan *horizontal* dan gerakan *vertical.*

2. *AccelerometerHandler Class*

AccelerometerHandler merupakan *class* yang berfungsi untuk menangani sensor *accelerometer*. Pertama kali yang akan dilakukan adalah memeriksa ketersediaan sensor *accelerometer* terhadap *device*. Apabila *device* memiliki sensor *accelerometer*, maka sistem akan mendapatkan sample *SensorManager* untuk mengatur *EventListener*. Sebaliknya, jika sensor *accelerometer* tidak terinstal pada *device*, maka percepatan akan bernilai 0 (nol) untuk semua sumbu sepanjang hidupnya.Untuk lebih jelas, berikut ini merupakan *flowchart* dari *AccelerometerHandler class* :



Gambar 3.8 *Flowchart AccelerometerHandler*

Kelas *AccelerometerHandler* menerapkan antarmuka *SensorEventListener* yang menyimpan tiga anggota dengan tipe data *float* yaitu *accelX*, *accelY*, dan *accelZ*. Ketiga anggota itu akan mengadakan percepatan pada masing-masing sumbu *accelerometer*. Dalam satu detik, akan mengambil nilai-nilai *accelerometer* dari *SensorEvent* dan menyimpannya kedalam tiga anggota sumbu. Terakhir merupakan pengembalian percepatan saat ini kepada setiap sumbu.

3.3.2 *Penerapan Procedural content generation (PCG)*

Metode dari *Procedural content generation* digunakan dalam penyebaran konten-konten yang dibutuhkan untuk meng-*generate* dunia *game* [2]. Dalam *game* yang akan dibangun, yang perlu diperhatikan adalah memastikan bahwa setiap *platform* berikutnya harus dapat dicapai ketika melompat. Mengingat kecepatan awal lompatan lutung kasarung sebesar 11 m/s secara *vertical*, maka dapat dihitung seberapa besar jarak antara *platform* yang memungkinkan karakter utama dapat menjangkaunya dengan perhitungan sebagai berikut :

$$tinggi = kecepatan \times \frac{kecepatan}{2 \times gravitasi} = 11 \times \frac{11}{2 \times 13} = 4{,}6m$$

Ini berarti jarak antara *platform* harus memiliki jarak sebesar 4,6 meter antara masing-masing *platform* sehingga lutung kasarung masih dapat mencapainya. Untuk memastikan semua *platform* dapat dijangkau, maka digunakanlah nilai yang sedikit lebih kecil dari tinggi lompatan maksimal. Dengan begitu akan menjamin setiap *platform* dapat dijangkau oleh lutung kasarung. Objek yang lainnya diciptakan secara kebetulan dengan memanfaatkan perhitungan *Random.nextFloat() method* yang terdapat pada *library*

java.util.Random class dapat mengembalikan *pseudorandom* dengan nilai *float* antara 0,0 dan 1,0. Pada setiap pemanggilan konten yang diperlukan, dimana setiap angka memiliki probabilitas kemunculan yang sama. Lebah hanya dihasilkan ketika nomor acak lebih besar dari 0,8, ini berarti kemunculan lebah akan menghasilkan probabilitas sebesar 20% (1–0,8). Hal yang serupa berlaku untuk semua konten yang dibuat secara acak dengan probabilitasnya masing-masing. Berikut ini merupakan konten-konten yang dibutuhkan dan ide umum dari metode *PCG* disetiap *level*-nya.

1. *Level* 1

   Konten-konten *game* yang dibutuhkan dalam pembentukan dunia *game level* 1 ini yaitu :

   a. *Platform* yang statis maupun dinamis

   b. Pisang (poin)

   c. Spring (pelontar)

   d. Pintu *level* 2

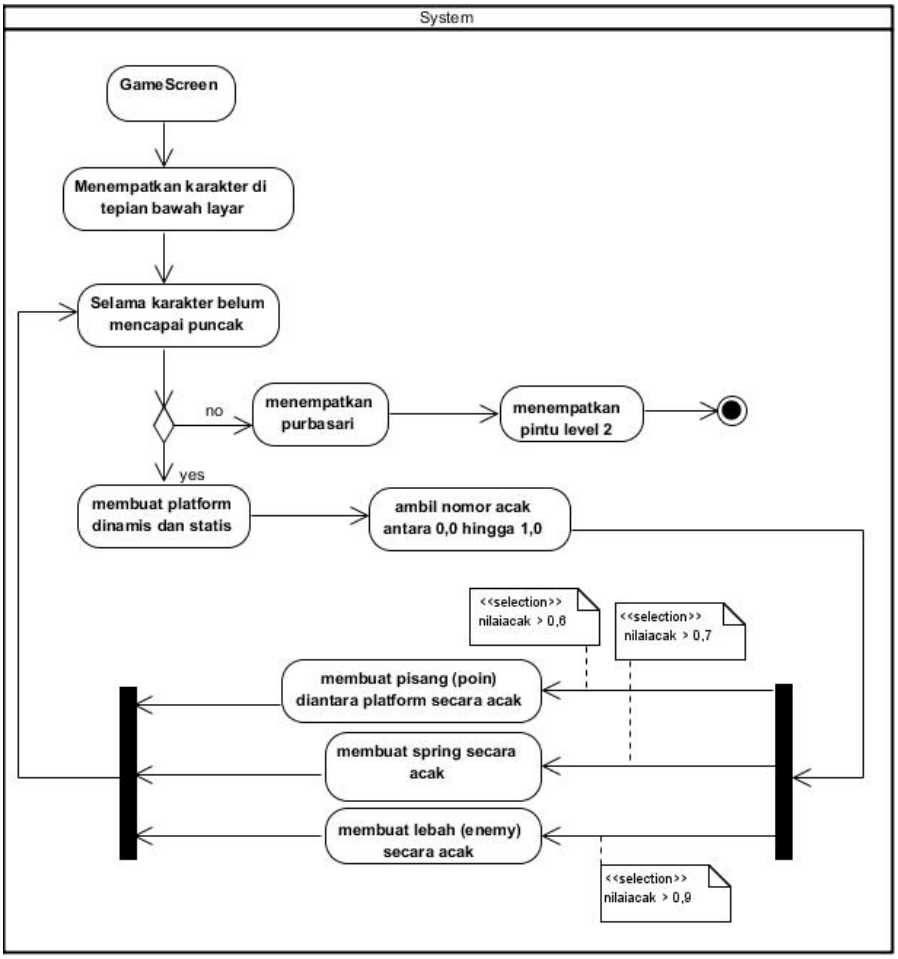
   Untuk lebih jelasnya mengenai konten-konten *game*, dapat dilihat kembali pada tabel 3.8. Adapun penguraian ide umum *level* 1 dari metode *PCG* dapat dilihat dari tabel 3.10 :

Tebel 3.10 Penguraian Ide Umum *PCG level* 1

No	Uraian
1	Dimulai dari dunia *game* paling dasar yaitu pada saat y = 0
2	Selama karakter belum mencapai puncak, maka dilakukan hal-hal berikut : a. Buat *platform* baik yang bergerak atau stasioner pada posisi y karakter berada dengan mengacak posisi x

	b. Ambil no.acak antara 0,0 hingga 1,0 c. Jika no.acak lebih besar dari 0,6, maka dibuat pisang (poin) secara acak diantara posisi *platform*. d. Jika no.acak lebih besar dari 0,7, maka dibuat spring (pelontar) secara acak diantara *platform* statis. e. Jika no.acak lebih besar dari 0,9, maka dibuat lebah terbang secara acak diantara *platform* statis.
3	Mengulang langkah 2 hingga karakter berada di puncak.
4	Terakhir menempatkan pintu level 2 pada posisi y maksimum sebagai akhir dari dunia *game level* 1 dan berlanjut ke *level* 2

Untuk lebih jelasnya, dapat dilihat dari *activity diagram* pada gambar 3.9.



Gambar 3.9 *Activity Diagram PCG Level* 1

2. *Level* 2

Berbeda dengan *level* 1, terdapat tambahan satu konten *game* untuk dunia *level* 2 yaitu ditambahkannya konten *spring* atau benda pegas yang akan membuat karakter melompat 1,5 kali lebih tinggi dari lompatan normal. Konten-konten *game* yang dibutuhkan dalam pembentukan dunia *game level* 2 ini yaitu :

a. *Platform* yang statis maupun dinamis
b. Pisang (poin)
c. *Spring*
d. Bunga Beracun
e. Lebah Terbang
f. Pintu *level* 3

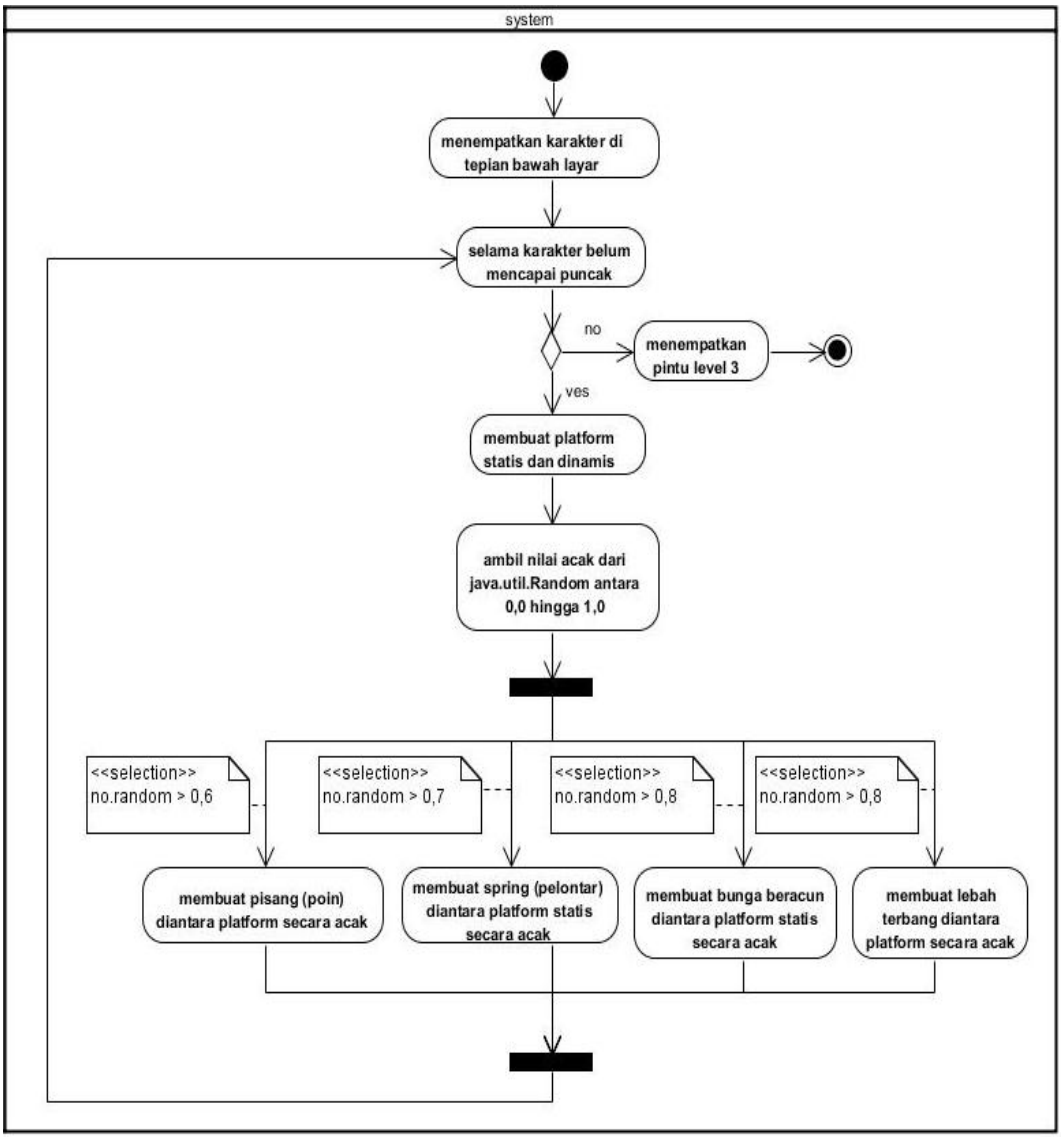
Untuk lebih jelasnya mengenai konten-konten *game*, dapat dilihat kembali pada tabel 3.8. Adapun penguraian ide umum *level* 2 dari metode *PCG* dapat dilihat dari tabel 3.11 :

Tabel 3.11 Penguraian Ide Umum *PCG level* 2

No	Uraian
1	Dimulai dari dunia *game* paling dasar yaitu pada saat y = 0
2	Selama karakter belum mencapai puncak, maka dilakukan hal-hal berikut : a. Buat *platform* baik yang bergerak atau stasioner pada posisi y karakter berada dengan mengacak posisi x b. Ambil nomor acak antara 0,0 hingga 1,0. c. Jika no.acak lebih besar dari 0,6, maka dibuat pisang (poin) secara acak diantara posisi *platform*. d. Jika no.acak lebih besar dari 0,7, maka dibuat spring

	(pelontar) secara acak diantara *platform* statis. e. Jika no.acak lebih besar dari 0,8, maka dibuat lebah terbang dan bunga beracun secara acak diantara *platform*.
3	Mengulang langkah 2 hingga karakter berada di puncak
4	Terakhir menempatkan pintu level 3 pada posisi y maksimum sebagai akhir dari dunia *game level* 2.

Untuk lebih jelasnya, dapat dilihat dari *activity diagram* pada gambar 3.10 :



Gambar 3.10 *Activity diagram PCG level* 2

3. *Level* 3

Berbeda dengan *level-level* sebelumnya, *level* 3 merupakan *level* yang tersulit untuk dilalui. Dengan penambahan satu konten lebah yang apabila karakter menyentuhnya, maka karakter akan menjadi mati dan permainanpun berakhir. Konten-konten *game* yang dibutuhkan dalam pembentukan dunia *game level* 3 ini yaitu :

a. *Platform* yang statis maupun dinamis
b. Pisang (poin)
c. *Spring*
d. Bunga Beracun
e. Lebah Terbang
f. Sekumpulan Lebah
g. Purbasari

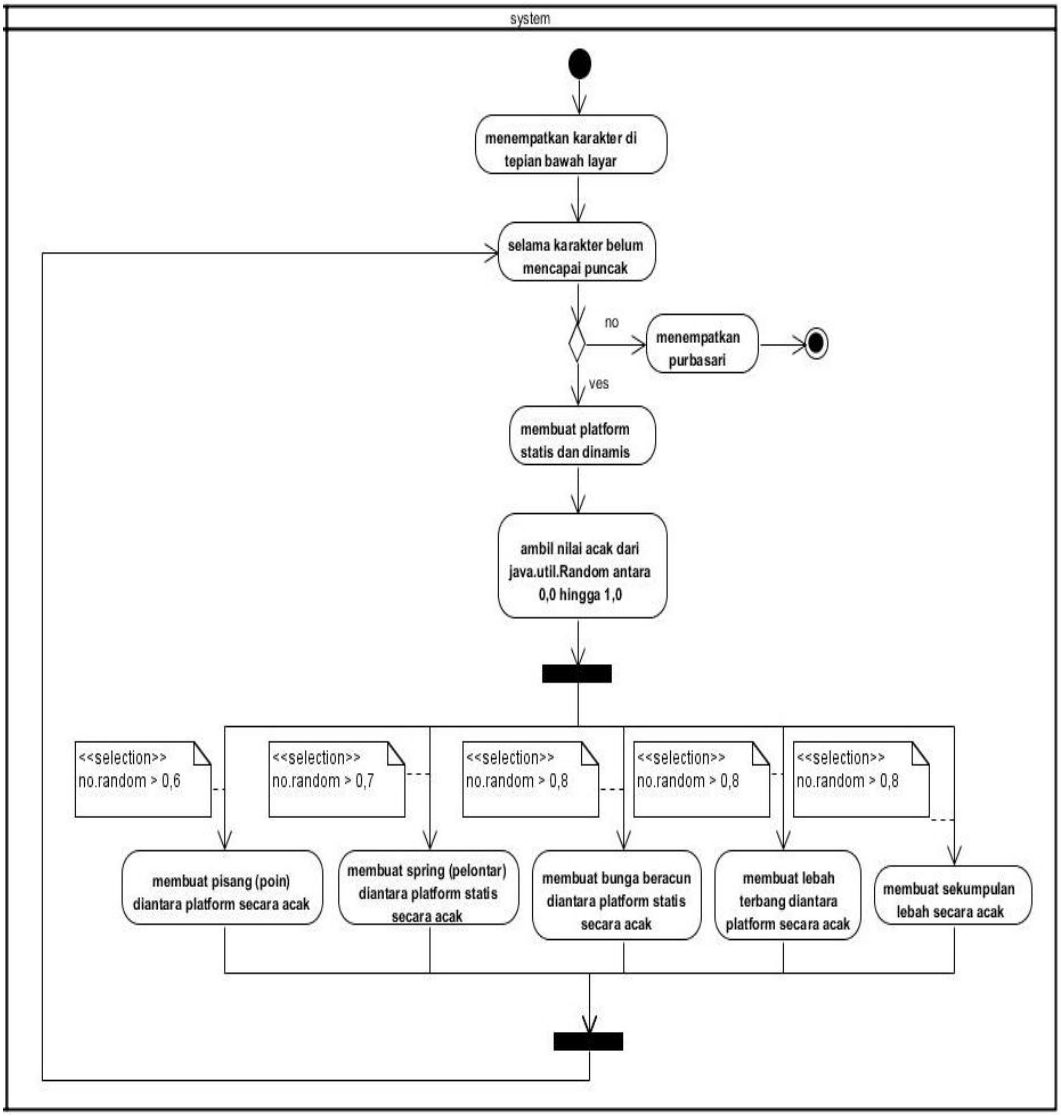
Untuk lebih jelasnya mengenai konten-konten *game*, dapat dilihat kembali pada tabel 3.8. Adapun Penguraian ide umum *level* 3 dari metode *PCG* dapat dilihat dari tabel 3.12 :

Tabel 3.12 Penguraian Ide Umum *PCG level* 3

No	Uraian
1	Dimulai dari dunia *game* palling dasar yaitu pada saat y = 0
2	Selama karakter belum mencapai puncak, maka dilakukan hal-hal berikut : a. Buat *platform* baik yang bergerak atau stasioner pada posisi y karakter berada dengan mengacak posisi x b. Ambil no.acak antara 0,0 hingga 1,0 c. Jika lebih besar dari 0,7, dan jika *platform* tidak bergerak, maka dibuat *spring* di atas *platform*

	d. Jika nomor acak di atas 0,8, maka dibuat lebah terbang, bunga beracun dan sekumpulan lebah secara acak.
3	Mengulang langkah 2 hingga karakter berada di puncak
4	Terakhir menempatkan karakter purbasari pada posisi y maksimum sebagai akhir dari dunia *game*.

Untuk lebih jelasnya, dapat dilihat dari *activity diagram* pada gambar 3.11 :

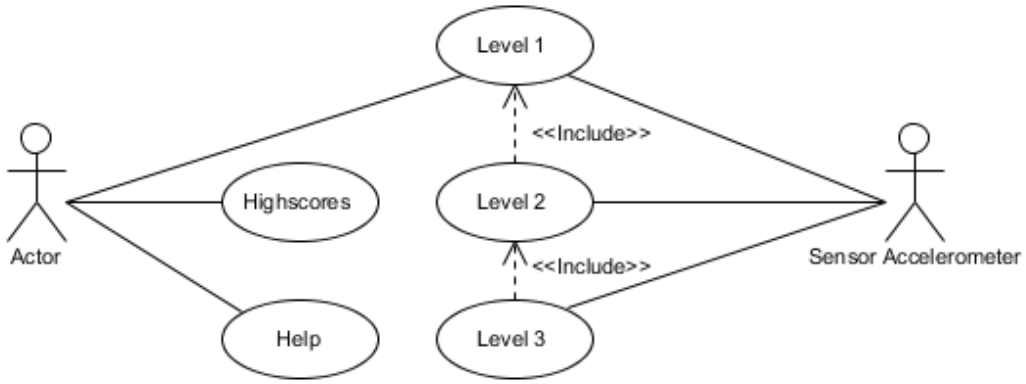


Gambar 3.11 *Activity diagram PCG level* 3

3.4 Analisis Kebutuhan Fungsional

3.4.1 *Use case Diagram*

Use case atau diagram *use case* merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat [5]. Berikut ini adalah perancangan proses-proses yang terdapat pada *game* Lutung kasarung *The jumper*, yang digambarkan dengan *Use case Diagram* di bawah ini:



Gambar 3.12 *Use Case Diagram game* Lutung kasarung *The jumper*.

3.4.2 Definisi *Actor*

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

Tabel 3.13 Definisi *Actor*

No	*Actor*	Deskripsi
1	*Actor*	Orang yang akan memainkan *game*
2	*Accelerometer Hardware*	*Hardware* yang memiliki pendeteksi gerakan percepatan

3.4.3 Definisi *Use Case*

Definisi *use case* berfungsi intuk menjelaskan fungsi *use case* yang terdapat pada *use case* diagram. Definisi *use case* diterangkan pada tabel 3.14 :

Tabel 3.14 Definisi *Use case*

No	*Use case*	Deskripsi
1	*Level* 1	Proses permainan pada *level* 1
2	*Level* 2	Proses permainan pada *level* 2
3	*Level* 3	Proses permainan pada *level* 3
4	*Highscores*	Proses untuk menampilkan pencapaian 5 *score* tertinggi
5	*Help*	Proses untuk menampilkan petunjuk tentang cara bermain

3.4.4 Skenario *Use case* dan *Activity diagram*

Skenario pada diagram *use case* berisi mengenai alur kejadian yang di dalamnya menjelaskan urutan interaksi antara *actor* dengan *use case* tersebut. Sedangkan diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis.

3.4.4.1 *Level* 1

1. **Skenario *Use case Level* 1**

   Skenario *use case* ini menjelaskan interaksi antara *actor* yaitu pemain, dengan *use case level* 1 yang akan dijelaskan pada tabel 3.15 :

Tabel 3.15 Skenario *Use case Level* 1

Identifikasi	
No	4
Nama	*Level* 1

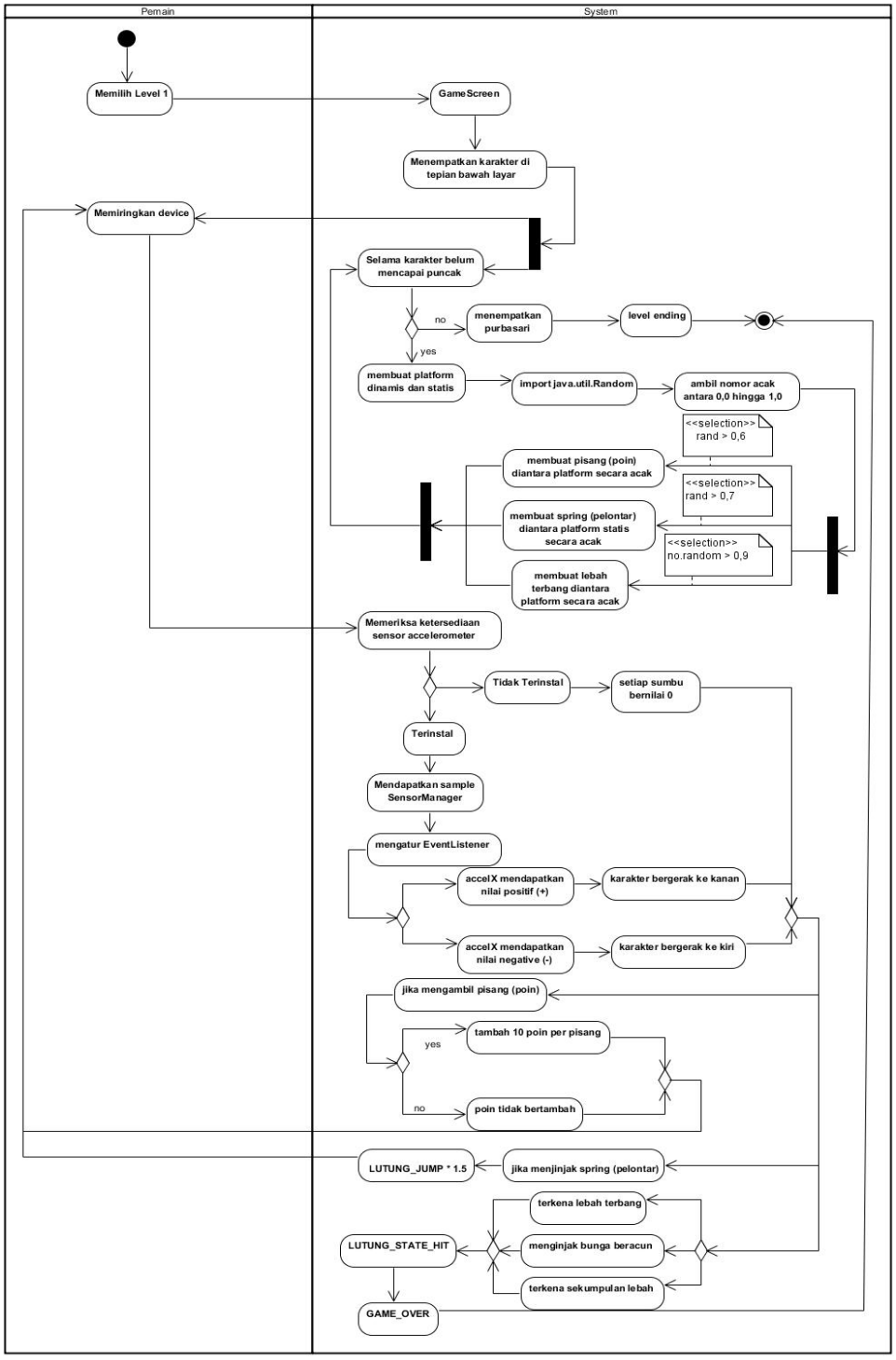
Tujuan	Untuk dapat memulai permainan dengan *level* 1
Actor	Pemain
Skenario Utama	
Kondisi Awal	Lutung kasarung berada di tepian bawah layar dengan kondisi karakter melompat-lompat

Aksi Aktor Pemain	**Reaksi sistem**
Memilih *level* 1	
	Masuk ke *level* 1
	Inisialisasi seluruh konten-konten yang dibutuhkan untuk *level* 1
	Menmpatkan karakter di tepian layar bawah
	Selama karakter belum mencapai puncak maka ditambahkan *platform* statis dan dinamis secara acak
	Panggil java.util.*Random*, kemudian mengambil nomor acak antara 0 hingga 1 dengan tipe data *float*
	Jika nomor acak lebih besar dari 0,6 maka dibuatkan pisang (poin) secara acak diantara *platform*
	Jika nomor acak lebih besar dari 0,7 maka dibuatkan spring (pelontar) secara acak diantara *platform* statis
	Jika nomor acak lebih besar dari 0,69 maka dibuatkan lebah terbang secara acak diantara *platform*
Mengambil pisang (poin)	
	Tambah 10 poin per pisang
Pemain menginjak *spring* (pelontar)	
	Lompatan menjadi 1,5 kali lebih tinggi dari lompatan normal

Aksi Aktor *Accelerometer* Hardware	**Reaksi Sistem**
	Memeriksa ketersediaan sensor *accelerometer*
Sensor *accelerometer* tidak terinstal	
	Setiap *accel*X, *accel*Y, dan *accel*Z akan diberikan nilai 0 (nol) sepanjang hidupnya
Sensor *accelerometer* terinstal	
	Mendapatkan sample *SensorManager* untuk mengatur EventListener.
Device cenderung miring ke kiri	
	Mendapatkan nilai negative (-) pada *accel*X untuk karakter yang bergerak ke kiri sebesar kemiringan.
Device cenderung miring ke kanan	
	Mendapatkan nilai positive (+) pada *accel*X untuk karakter yang bergerak ke kanan sebesar kemiringan.
Kondisi akhir	Menampilkan permainan dengan dunia *game level* 1

2. *Activity Diagram Level* 1

Activity diagram level 1 pada aplikasi *game* ini dapat dilihat dari gambar berikut :



Gambar 3.13 *Activity diagram Level* 1

3.4.4.2 *Level* 2

1. **Skenario *Use case Level* 2**

   Skenario *use case* ini menjelaskan interaksi antara *actor* yaitu pemain, dengan *use case level* 2 yang akan dijelaskan pada tabel 3.16 :

Tabel 3.16 Skenario *Use case Level* 2

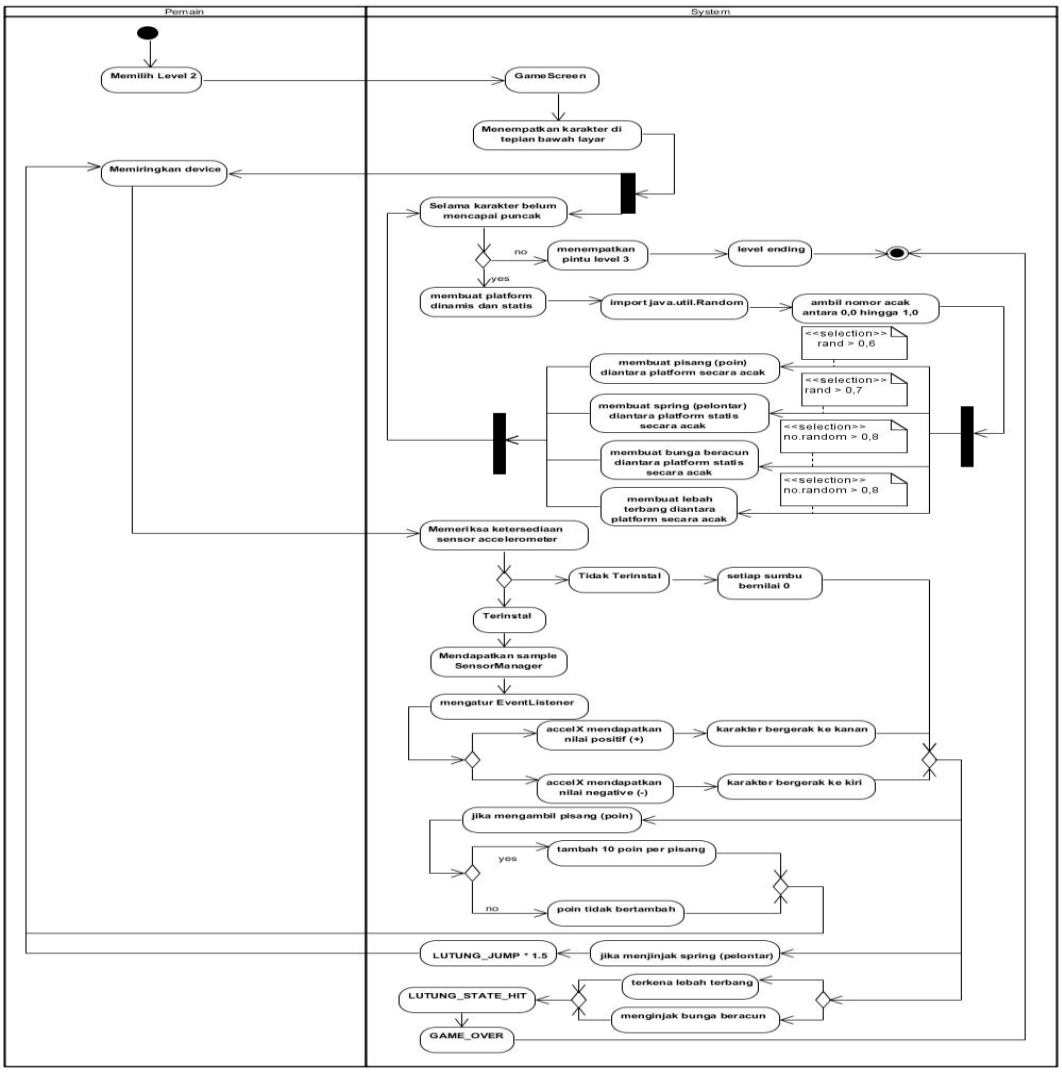
Identifikasi		
No	5	
Nama	*Level* 2	
Tujuan	Untuk dapat memulai permainan dengan *level* 2	
Actor	Pemain	
Skenario Utama		
Kondisi Awal	Lutung kasarung berada di tepian bawah layar dengan kondisi karakter melompat-lompat	
Aksi Aktor		**Reaksi sistem**
Memilih *level* 2		
		Jika *level* 1 belum berhasil diselesaikan, maka pemain tidak dapat masuk ke dunia *game level* 2
		Jika *level* 1 sudah berhasil diselesaikan, maka dapat masuk ke dunia *game level* 2
		Menmpatkan karakter di tepian layar bawah
		Selama karakter belum mencapai puncak maka ditambahkan *platform* statis dan dinamis secara acak
		Panggil java.util.*Random*, kemudian mengambil nomor acak antara 0 hingga 1 dengan tipe data *float*
		Jika nomor acak lebih besar dari 0,6 maka dibuatkan pisang (poin) secara

	acak diantara *platform*
	Jika nomor acak lebih besar dari 0,7 maka dibuatkan *spring* (pelontar) secara acak
	Jika nomor acak lebih besar dari 0,8 maka dibuatkan lebah terbang dan bunga beracun secara acak
Pemain menginjak *spring* (benda pegas)	
	Lompatan menjadi 1,5 kali lebih tinggi dari lompatan normal
Pemain mengambil pisang (poin)	
	Menambahkan poin sebesar 10 untuk pengambilan satu pisang (poin)
Pemain terkena lebah terbang	
	Kondisi lutung akan menjadi mati
Pemain menginjak bunga beracun	
	Kondisi lutung akan menjadi mati
Aksi Aktor *Accelerometer* Hardware	**Reaksi Sistem**
	Memeriksa ketersediaan sensor *accelerometer*
Sensor *accelerometer* tidak terinstal	
	Setiap *accel*X, *accel*Y, dan *accel*Z akan diberikan nilai 0 (nol) sepanjang hidupnya
Sensor *accelerometer* terinstal	
	Mendapatkan sample *SensorManager* untuk mengatur EventListener.
Device cenderung miring ke kiri	
	Mendapatkan nilai negative (-) pada *accel*X untuk karakter yang bergerak

		ke kiri sebesar kemiringan.
Device cenderung miring ke kanan		
		Mendapatkan nilai positive (+) pada *accel*X untuk karakter yang bergerak ke kanan sebesar kemiringan.
Kondisi akhir	Menampilkan permainan dengan dunia *game level* 2	

2. Activity *Diagram Level* 2

Activity diagram level 2 pada aplikasi *game* ini dapat dilihat dari gambar berikut ini :



Gambar 3.14 *Activity diagram Level* 2

3.4.4.3 *Level* 3

1. **Skenario *Use case Level* 3**

   Skenario *use case* ini menjelaskan interaksi antara *actor* yaitu pemain, dengan *use case level* 3 yang akan dijelaskan pada tabel 3.17 :

Tabel 3.17 Skenario *Use case Level* 3

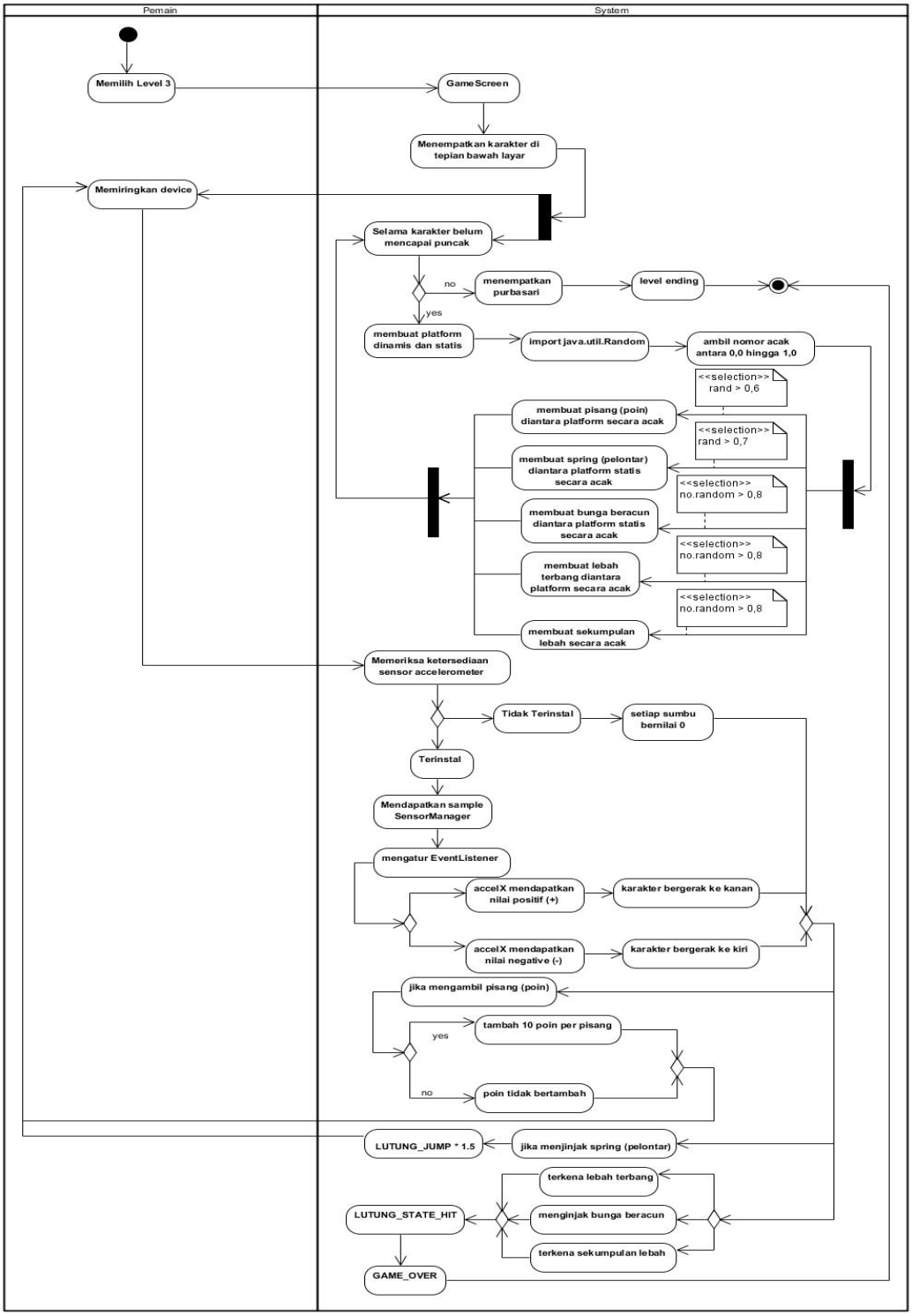
Identifikasi		
No	6	
Nama	*Level* 3	
Tujuan	Untuk dapat memulai permainan dengan *level* 3	
Actor	Pemain	
Skenario Utama		
Kondisi Awal	Lutung kasarung berada di tepian bawah layar dengan kondisi karakter melompat-lompat	
Aksi Aktor		**Reaksi sistem**
Memilih *level* 3		
		Jika *level* 2 belum berhasil diselesaikan, maka pemain tidak dapat masuk ke dunia *game level* 3
		Jika *level* 2 sudah berhasil diselesaikan, maka dapat masuk ke dunia *game level* 3
		Menmpatkan karakter di tepian layar bawah
		Selama karakter belum mencapai puncak maka ditambahkan *platform* statis dan dinamis secara acak
		Panggil java.util.*Random*, kemudian mengambil nomor acak antara 0 hingga 1 dengan tipe data *float*
		Jika nomor acak lebih besar dari 0,6 maka dibuatkan pisang (poin) secara

	acak diantara *platform*
	Jika nomor acak lebih besar dari 0,7 maka dibuatkan *spring* (pelontar) secara acak
	Jika nomor acak lebih besar dari 0,8 maka dibuatkan lebah terbang, bunga beracun dan sekumpulan lebah dengan kemunculan secara acak
Pemain menginjak *spring* (benda pegas)	
	Lompatan menjadi 1,5 kali lebih tinggi dari lompatan normal
Pemain mengambil pisang (poin)	
	Menambahkan poin sebesar 10 untuk pengambilan satu pisang (poin)
Pemain terkena lebah terbang	
	Kondisi lutung akan menjadi mati
Pemain menginjak bunga beracun	
	Kondisi lutung akan menjadi mati
Pemain terkena sekumpulan lebah	
	Kondisi lutung akan menjadi mati
Kondisi akhir	Menampilkan permainan dengan dunia *game level* 3
Aksi Aktor *Accelerometer* Hardware	**Reaksi Sistem**
	Memeriksa ketersediaan sensor *accelerometer*
Sensor *accelerometer* tidak terinstal	
	Setiap *accel*X, *accel*Y, dan *accel*Z akan diberikan nilai 0 (nol) sepanjang hidupnya
Sensor *accelerometer* terinstal	
	Mendapatkan sample *SensorManager*

	untuk mengatur EventListener.
Device cenderung miring ke kiri	
	Mendapatkan nilai negative (-) pada *accel*X untuk karakter yang bergerak ke kiri sebesar kemiringan.
Device cenderung miring ke kanan	
	Mendapatkan nilai positive (+) pada *accel*X untuk karakter yang bergerak ke kanan sebesar kemiringan.
Kondisi akhir	Menampilkan permainan dengan dunia *game level* 3

2. *Activity Diagram Level* 3

Activity diagram Level 3 pada aplikasi *game* ini dapat dilihat dari gambar berikut :



Gambar 3.15 *Activity diagram Level* 3

3.4.4.4 Menu *Highscores*

1. Skenario *Use Case Highscores*

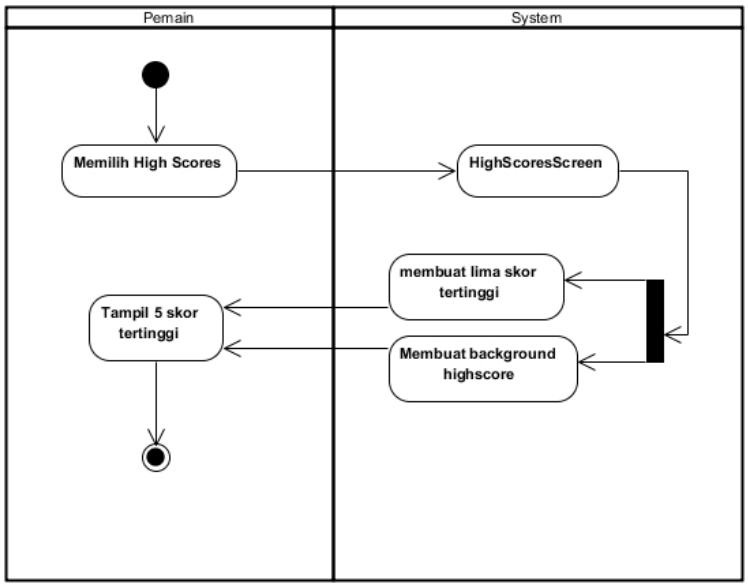
Skenario *use case* ini menjelaskan interaksi antara *actor* yaitu pemain, dengan *use case score* tertinggi yang akan dijelaskan pada tabel 3.18 :

Tabel 3.18 Skenario *Use case High Scores*

Identifikasi		
No	2	
Nama	*Score* Tertinggi	
Tujuan	Untuk dapat melihat daftar *score* tertinggi yang telah dicapai selama permainan	
Actor	Pemain	
Skenario Utama		
Kondisi Awal	Menu Utama ditampilkan	
Aksi Aktor		**Reaksi sistem**
Memilih *highscores*		
		Masuk ke kelas *HighscoresScreen*
		Membuat *background* dan lima *score* tertinggi
Kondisi akhir	Menampilkan 5 daftar *score* tertinggi	

2. *Diagram Activity Highscores*

Activity diagram Highscores pada aplikasi *game* ini dapat dilihat dari gambar berikut:



Gambar 3.16 *Diagram Activity High Scores*

3.4.4.5 Menu *Help*

1. **Skenario *Use case Help***

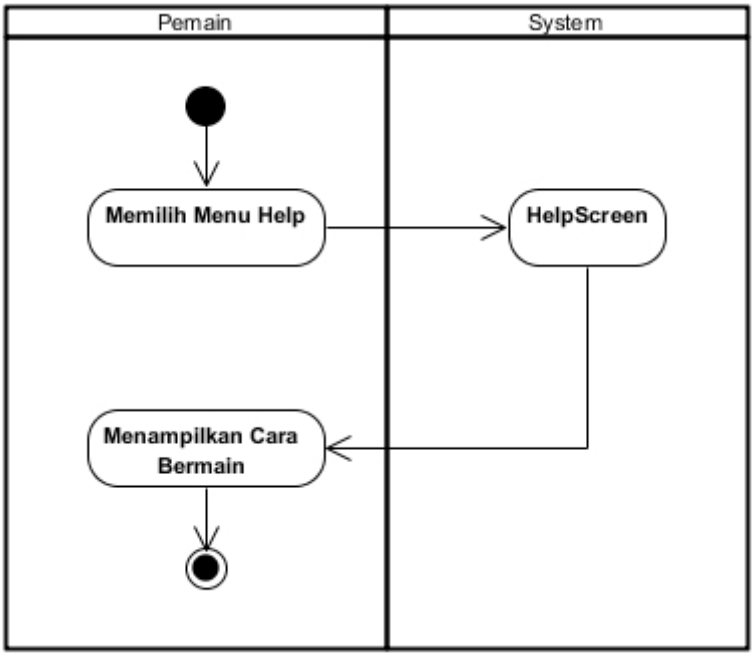
Skenario *use case* ini menjelaskan interaksi antara *actor* yaitu pemain, dengan *use case* cara bermain yang akan dijelaskan pada tabel 3.19 :

Tabel 3.19 Skenario *Use case Help*

Identifikasi		
No	3	
Nama	Cara Bermain	
Tujuan	Untuk dapat mengetahui bagaimana cara *game* ini dimainkan	
Actor	Pemain	
Skenario Utama		
Kondisi Awal	Menu Utama ditampilkan	
Aksi Aktor		**Reaksi sistem**
Memilih *help*		
		Masuk ke kelas *HelpScreen*
		Tampil informasi cara bermain
Kondisi akhir	Menampilkan informasi mengenai petunjuk bermain	

2. *Activity Diagram Help*

Activity diagram Help pada palikasi *game* ini dapat dilihat dari gambar berikut :



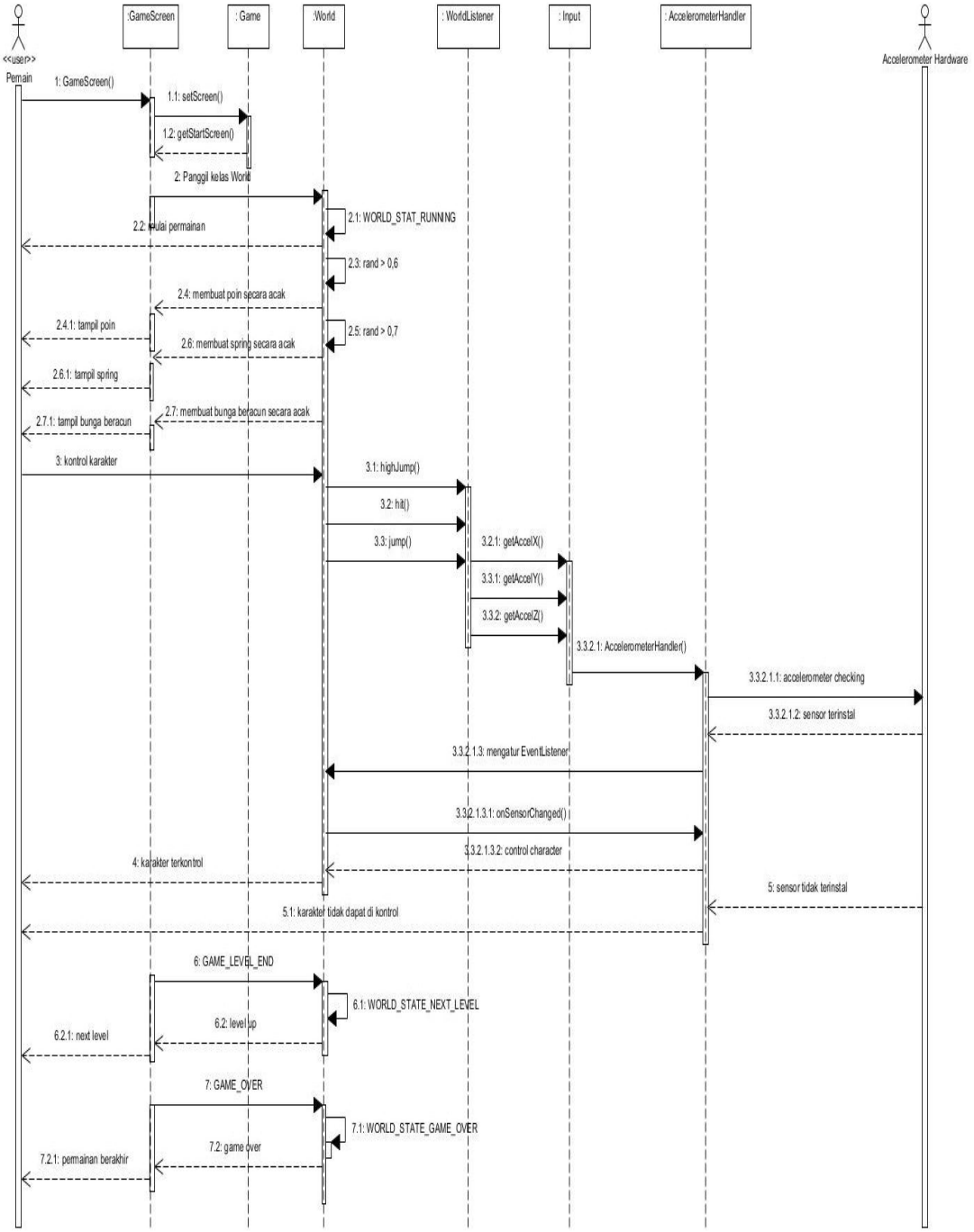
Gambar 3.17 *Activity Diagram Help*

3.4.5 *Sequence Diagram*

Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek [5]. Berikut adalah gambar *sequence diagram* pada *game* lutung kasarung *the jumper* :

1. ***Sequence Diagram Level 1***

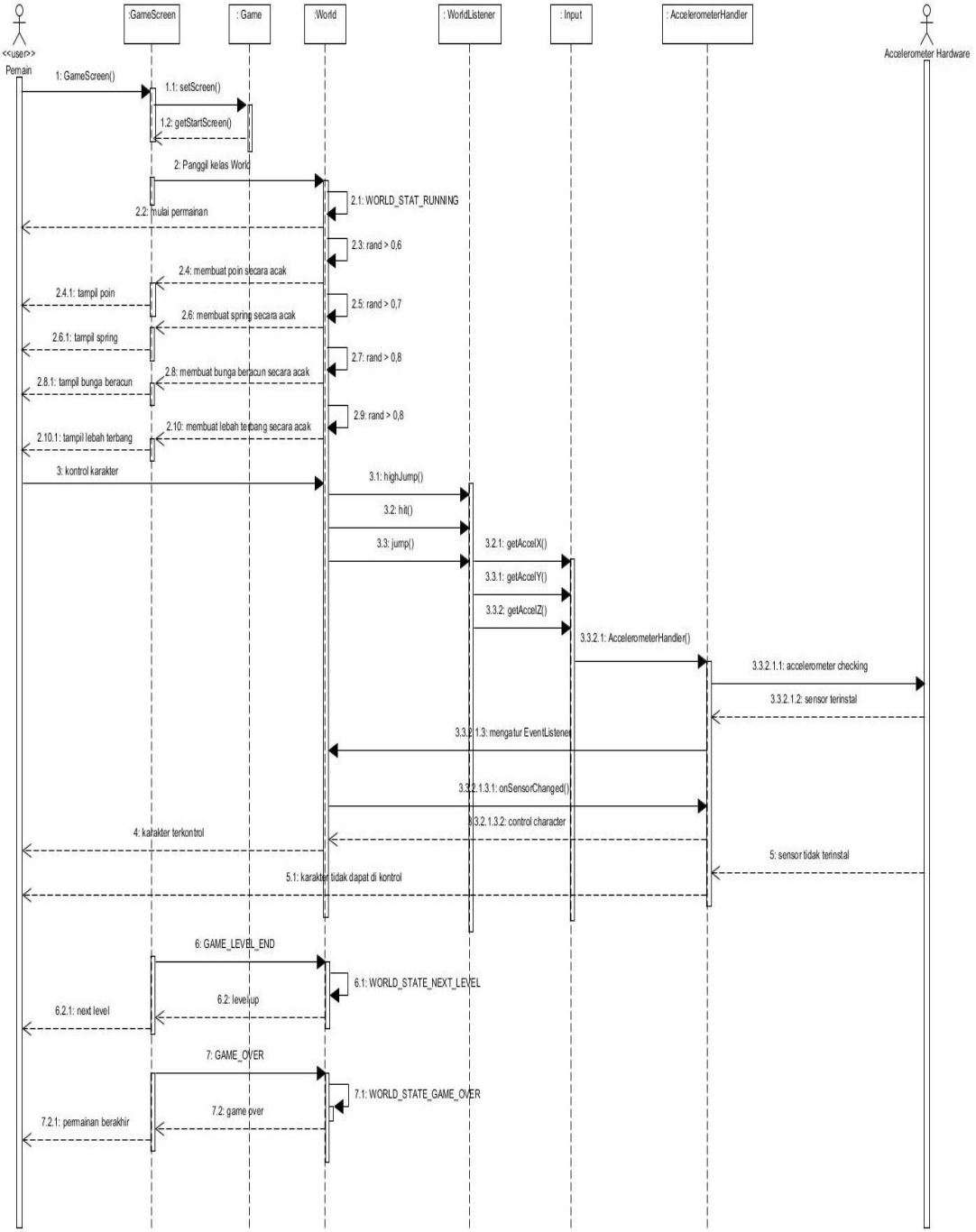
Sequence diagram level 1 pada aplikasi ini dapat dilihat pada gambar 3.18:



Gambar 3.18 *Sequence Diagram Level 1*

2. *Sequence Diagram Level 2*

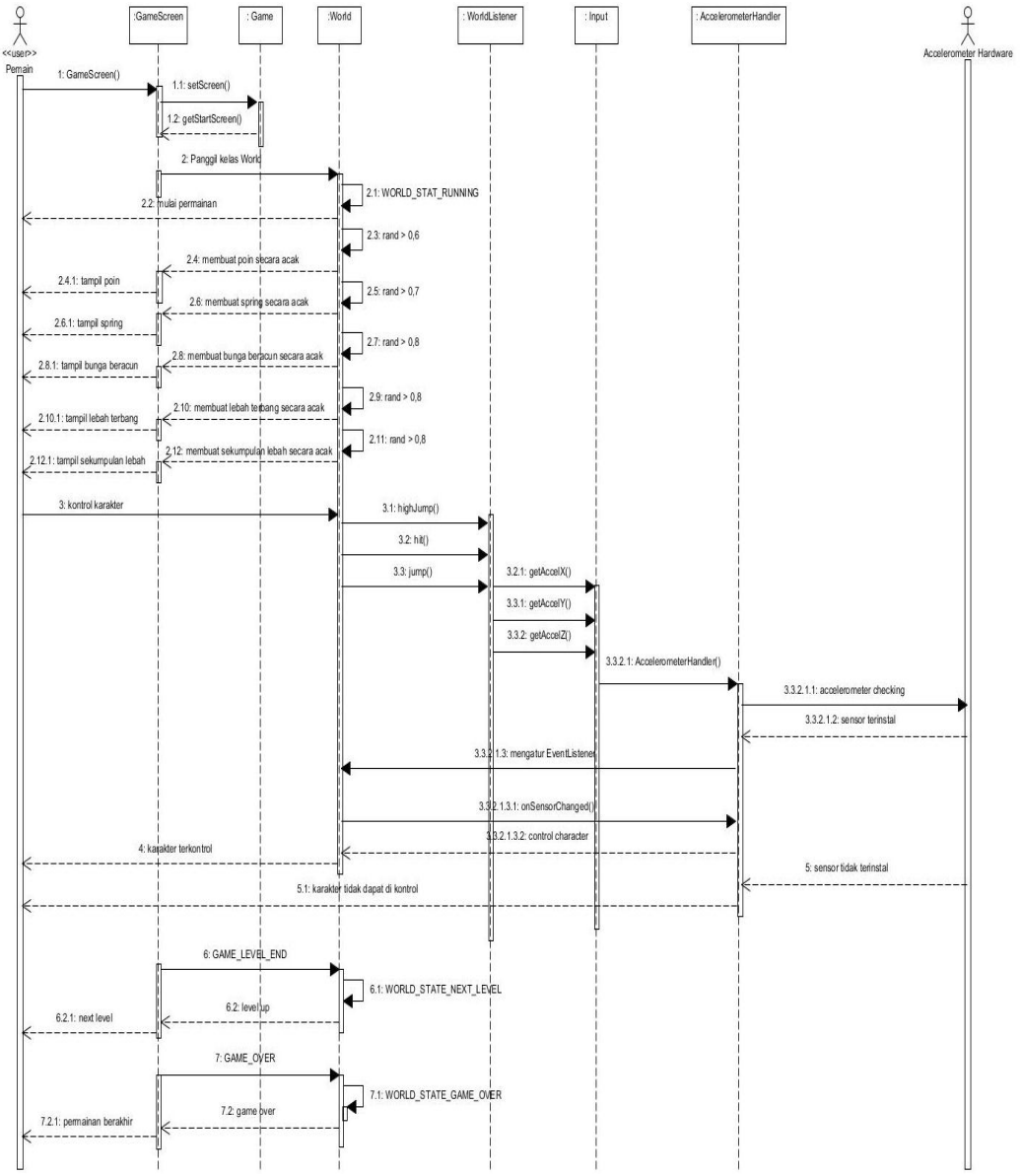
Sequence diagram level 2 pada aplikasi ini dapat dilihat sebagai berikut :



Gambar 3.19 *Sequence Diagram Level 2*

3. *Sequence Diagram Level 3*

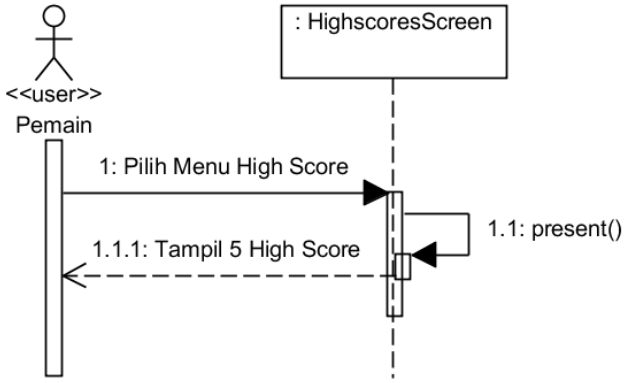
Sequence diagram level 3 pada aplikasi ini dapat dilihat pada gambar 3.20:



Gambar 3.20 Sequence Diagram Level 3

4. Sequence Diagram Highscores

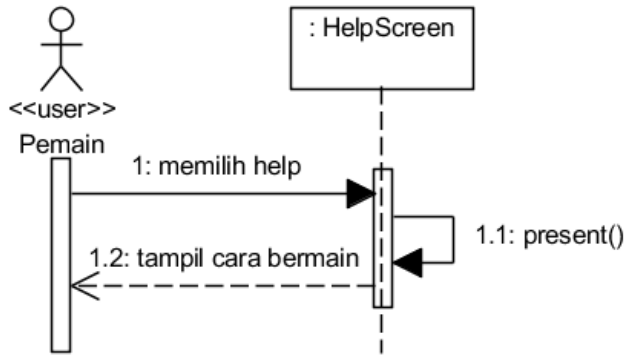
Sequence diagram highscores pada aplikasi ini dapat dilihat pada gambar 3.21:



Gambar 3.21 *Sequence Diagram Highscores*

5. *Sequence Diagram Help*

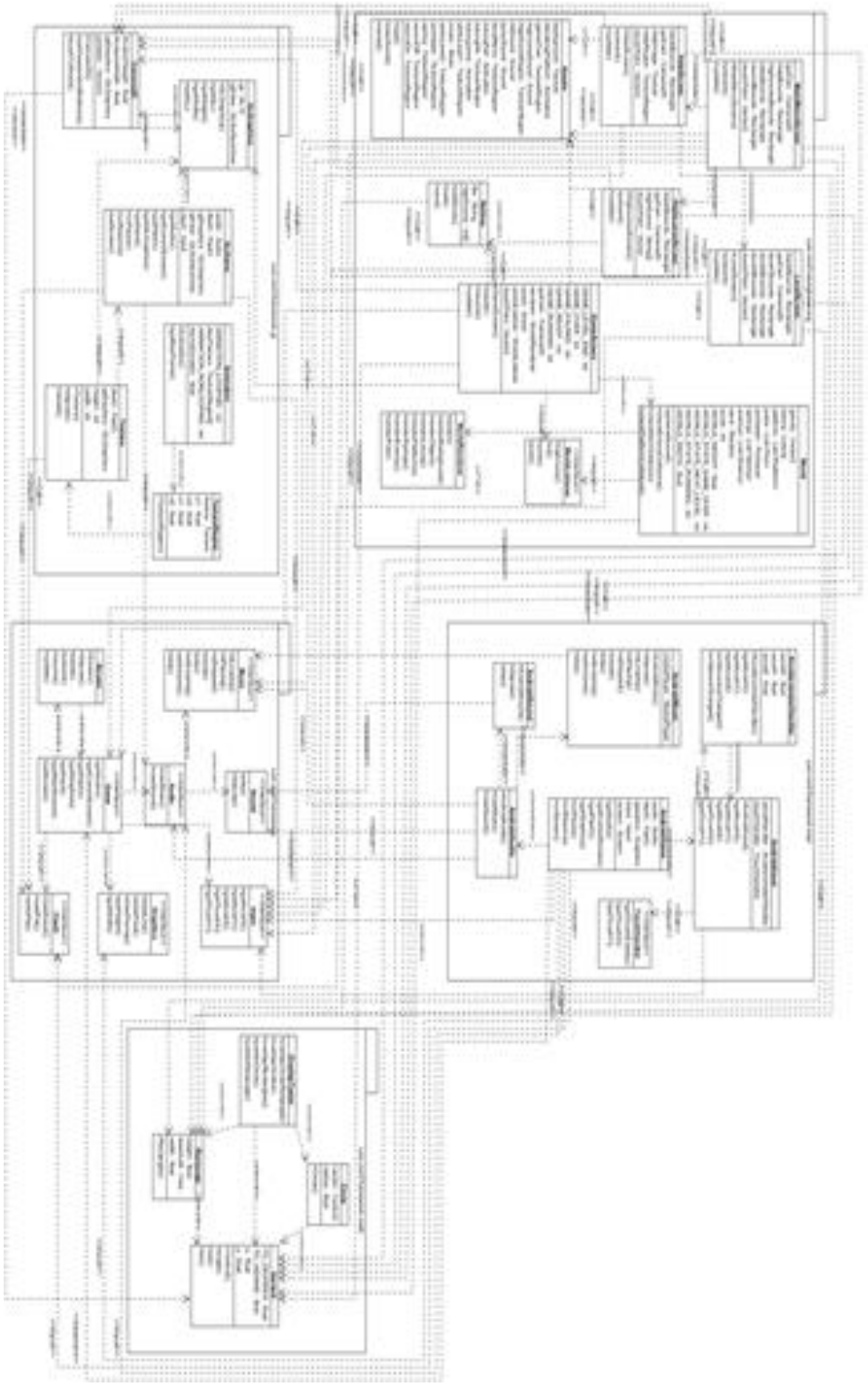
Sequence diagram help pada aplikasi ini dapat dilihat pada gambar 3.22 :



Gambar 3.22 *Sequence Diagram Help*

3.4.6 *Class Diagram*

Class diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang dibuat untuk membangun sistem. Berikut ini adalah perancangan struktur sistem yang terdapat pada *game* lutung kasarung *the jumper*, yang digambarkan dengan *class* diagram *package* :



Gambar 3.23 *Diagram Class Game* Lutung Kasarung *The Jumper*

3.5 Analisis Kebutuhan Non Fungsional

Analisis kebutuhan non-fungsional merupakan analisis yang dibutuhkan untuk menentukan spesifikasi kebutuhan sistem [5]. Spesifikasi ini juga meliputi elemen atau komponen-komponen apa saja yang dibutuhkan untuk sistem yang akan dibangun sampai dengan sistem tersebut dapat diimplementasikan. Analisis kebutuhan ini juga menentukan spesifikasi masukan yang diperlukan sistem, keluaran yang akan dihasilkan dan proses yang dibutuhkan untuk mengolah masukan sehingga menghasilkan suatu keluaran yang diinginkan.

3.5.1 Analisis Kebutuhan Perangkat Lunak

Perangkat lunak digunakan dalam sebuah sistem merupakan perintah-perintah yang diberikan kepada perangkat keras agar dapat saling berinteraksi diantara keduanya. Perangkat lunak yang dibutuhkan untuk membangun aplikasi *game* Lutung kasarung *The jumper* ini adalah sebagai berikut :

1. Sistem *operasi Windows* 7 *Ultimate* 64-bit
2. *Eclipse Hellios*
3. *Adobe Photoshop CS6*
4. *Corel Draw Graphics Suite X6*
5. *VisualParadigm for UML 9.0*

3.5.2 Analisis Kebutuhan Spesifikasi Perangkat Keras

Komputer terdiri dari perangkat keras dan perangkat lunak yang saling berinteraksi. Perangkat lunak memberikan instruksi-instruksi kepada

perangkat keras untuk melakukan tugas tertentu. Adapun perangkat keras yang digunakan dalam membangun aplikasi *game* Lutung kasarung *The jumper* adalah sebagai berikut :

1. *Intel® Core™ I5-2430M CPU @ 2.40Ghz*
2. *Harddisk 640GB*
3. *Ram 4096MB*
4. *VGA AMD Radeon HD 6730M*
5. *Display LCD Monitor* dengan *resolusi* 1366 x 768

3.5.3 Analisis Pengguna

Spesifikasi minimal bagi pengguna yang dapat menggunakan aplikasi ini adalah sebagai berikut :

1	Peran	:	Pemain
2	Usia	:	10 – 15 tahun
3	Tingkat Keterampilan	:	- Dapat menggunakan smartphone Android - Sedikitnya mengerti Bahasa Inggris
4	Pengalaman	:	- Pernah memainkan aplikasi *game mobile* - Sedikitnya mengetahui tentang sensor accelerometer pada perangkat mobile
5	Minat terhadap *game casual*	:	Baik

3.5.4 Analisis Perangkat Keras Pengguna

Pada sisi pengguna, perangkat keras yang dibutuhkan untuk menjalankan aplikasi ini adalah sebagai berikut :

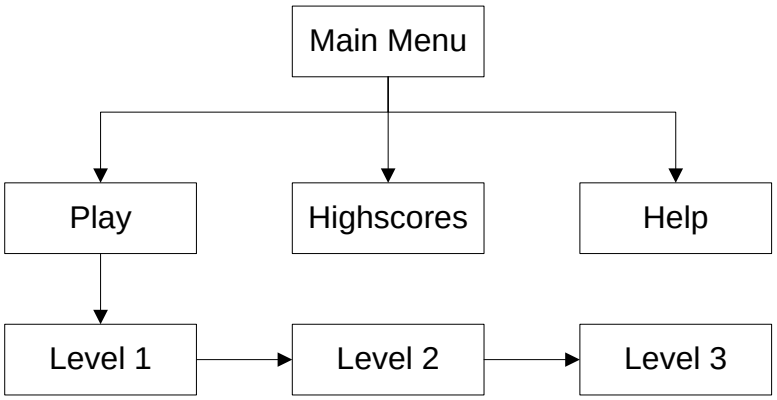
1. *Smartphone Android* 2.2 *(froyo)* keatas
2. Adanya perangkat Sensor *Accelerometer*

3.6 Perancangan Perangkat Lunak

Perancangan perangkat lunak merupakan kegiatan untuk merancang aplikasi yang akan di bangun, tahapannya dimulai dari perancangan arsitektur menu, kemudian berlanjut ke perancangan antar muka dan jaringan semantik.

3.6.1 Perancangan Arsitektur Menu

Perancangan struktur menu merupakan jalur pemakaian sistem yang mudah dipahami dan mudah digunakan. Untuk lebih jelasnya dapat dilihat pada gambar di bawah ini, yaitu :



Gambar 3.24 Arsitektur Menu Lutung kasarung *The jumper*

1. Menu *Play* : merupakan submenu untuk memulai permainan baru
2. Menu *Highscores* : merupakan submenu yang menampilkan informasi lima *score* tertinggi

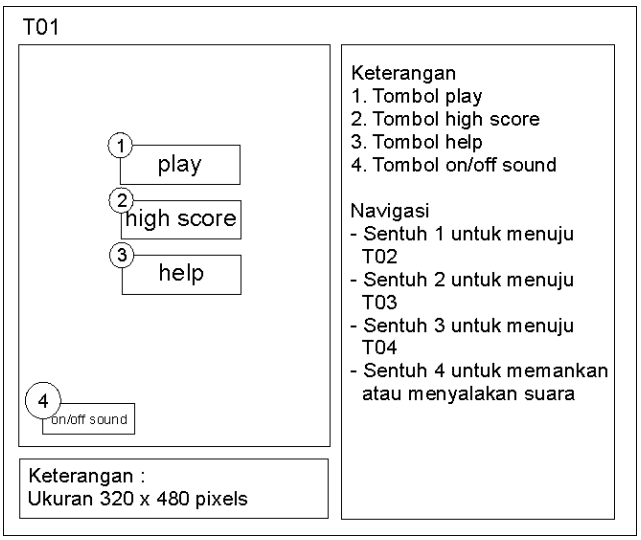
3. Menu *Help* : merupakan submenu yang menampilkan informasi petunjuk permainan
4. *Level* 1 : merupakan submenu *level* yang menampilkan dunia *game level* 1
5. *Level* 2 : merupakan submenu *level* yang menampilkan dunia *game level* 2
6. *Level* 3 : merupakan submenu *level* yang menampilkan dunia *game level* 3

3.6.2 Perancangan Antar Muka

Perancangan antarmuka bertujuan untuk memberikan gambaran tentang aplikasi yang akan dibangun, sehingga akan mempermudah dalam mengimplementasikan aplikasi serta akan memudahkan pembuatan aplikasi. Berikut ini merupakan perancangan antar muka pada *game* Lutung Kasarung *The Jumper* :

1. Perancangan Antar Muka Menu Utama

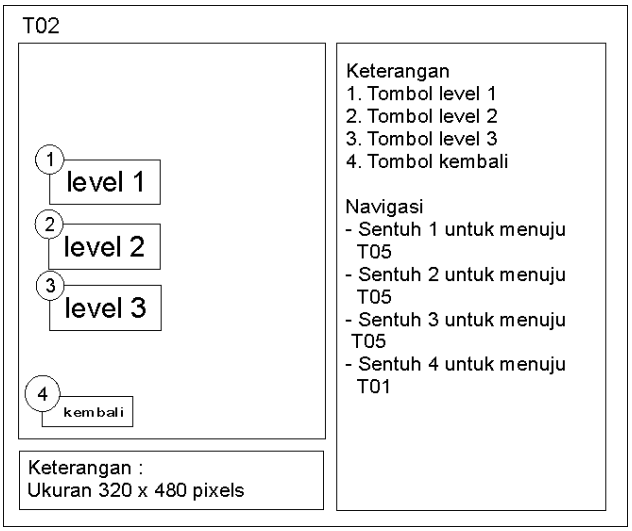
Perancangan antar muka Menu Utama pada aplikasi ini dapat dilihat sebagai berikut :



Gambar 3.25 Perancangan Antar Muka Menu Utama

2. Perancangan Antar Muka Menu *Level*

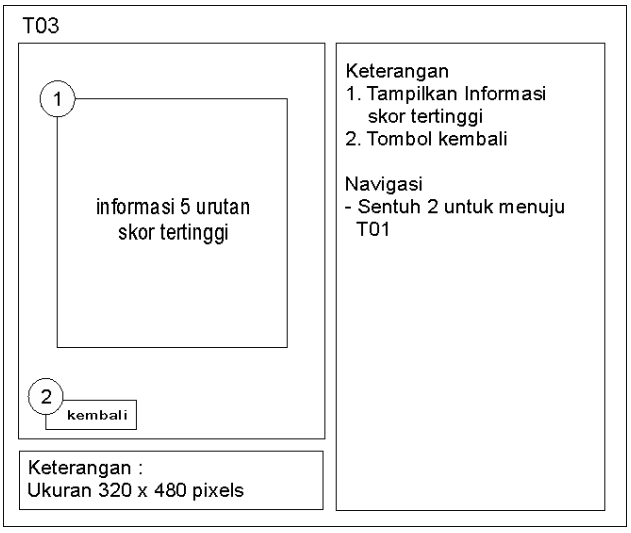
Perancangan antar muka menu *level* pada aplikasi ini dapat dilihat sebagai berikut :



Gambar 3.26 Perancangan Antar Muka Menu *Level*

3. Perancangan Antar Muka Menu *Highscores*

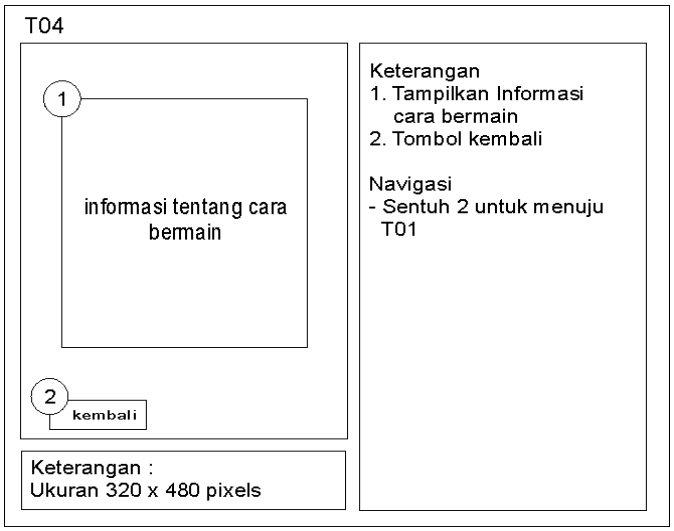
Perancangan antar muka menu *highscores* pada aplikasi ini dapat dilihat sebagai berikut :



Gambar 3.27 Perancangan Antar Muka Menu *Highscores*

4. Perancangan Antar Muka Menu *Help*

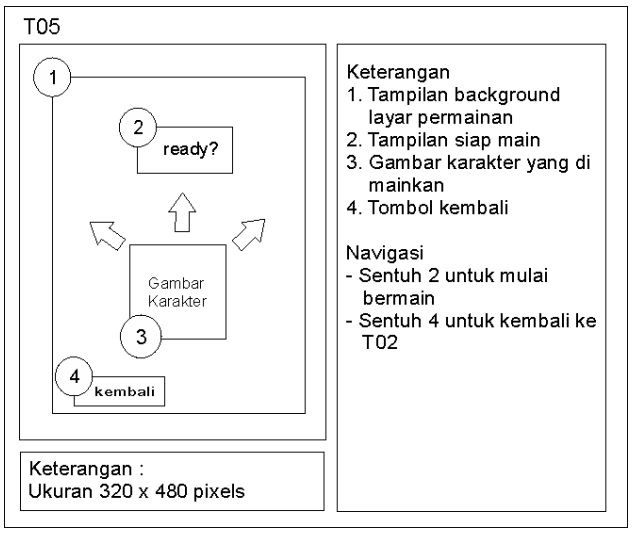
Perancangan antar muka menu *help* pada aplikasi ini dapat dilihat sebagai berikut :



Gambar 3.28 Perancangan Antar Muka Menu *Help*

5. Perancangan Antar Muka Ready?

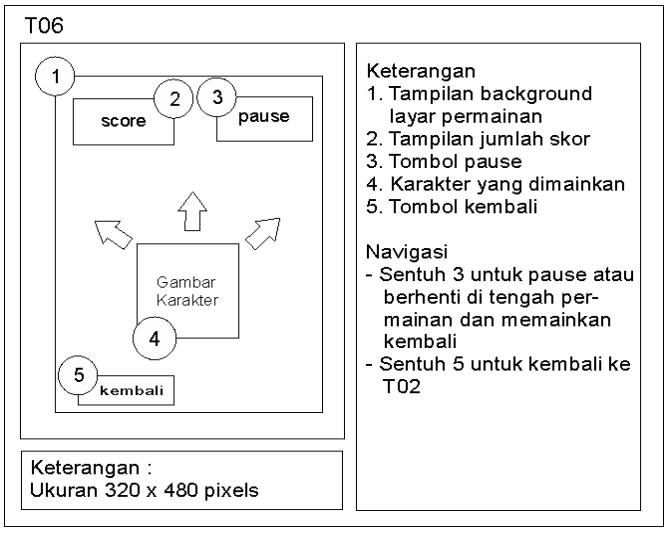
Perancangan antar muka "*ready?"* atau siap bermain pada aplikasi ini dapat dilihat sebagai berikut :



Gambar 3.29 Perancangan Antar Muka "Ready?"

6. Perancangan Antarmuka Permainan

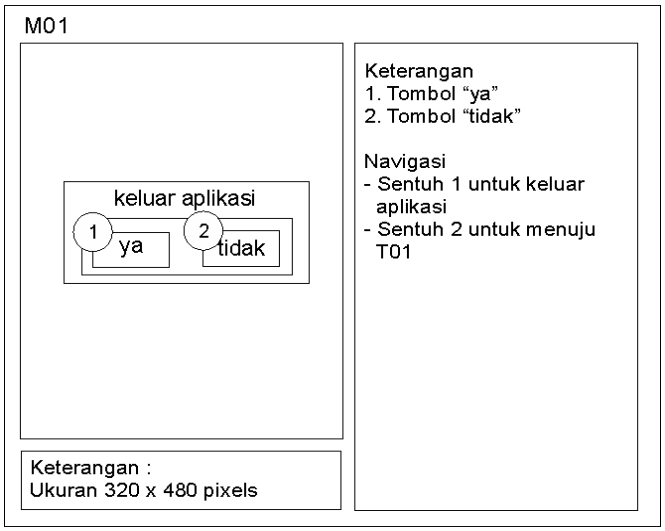
Perancangan antar muka permainan pada aplikasi ini dapat dilihat sebagai berikut :



Gambar 3.30 Perancangan Antar Muka Permainan

7. Perancangan Antarmuka Pesan Keluar Aplikasi

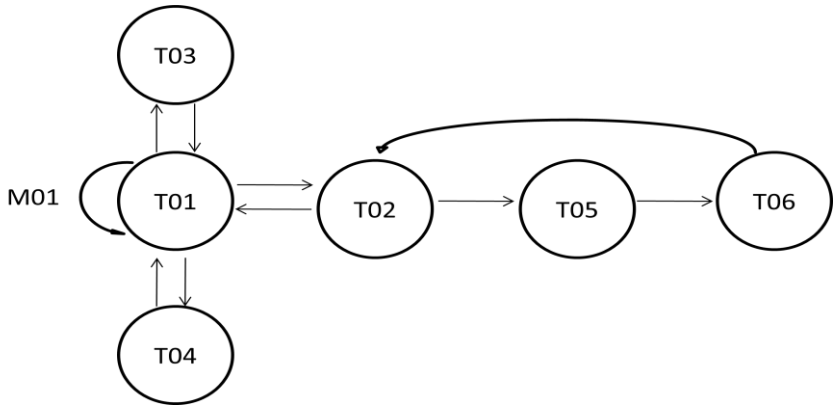
Perancangan antar muka pesan keluar pada aplikasi ini dapat dilihat sebagai berikut :



Gambar 3.31 Perancangan Antar Muka Pesan Keluar Aplikasi

3.6.3 Jaringan Semantik

Jaringan semantik adalah gambaran tentang relasi dari masing- masing halaman yang bisa diakses oleh seorang pengguna. Jaringan semantik ini, digunakan sebagai panduan penggunaan menu-menu dalam sebuah aplikasi. Terdapat 6 tampilan pada aplikasi ini, yaitu T01 untuk tampilan menu utama aplikasi, T02 untuk tampilan submenu *level*, T03 untuk tampilan informasi *score* tertinggi, T04 untuk tampilan informasi cara bermain, T05 untuk tampilan "ready" sebelum permainan dimulai, dan terakhir T06 merupakan tampilan permainan. Perancangan semantik ini dapat dilihat pada gambar 3.32 berikut :



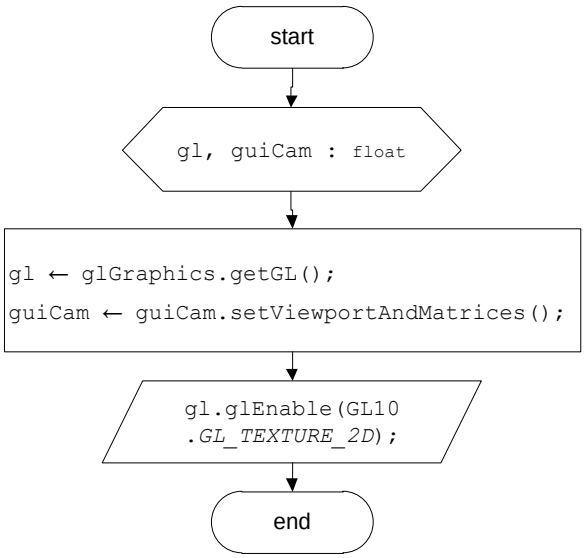
Gambar 3.32 Jaringan Semantik

3.6.4 Perancangan *Method*

Perancangan *method* merupakan perncangan yang berfungsi untuk mendeskripsikan *method-method* yang berada di dalam aplikasi. *Method* dapat dipanggil dengan menyertakan variabel, baik hanya satu variabel, banyak variabel atau bahkan tidak ada sama sekali .Adapun *method-method* yang terdapat dalam *game* lutung kasarung yang akan dibangun adalah sebagai berikut :

1. *Method present*

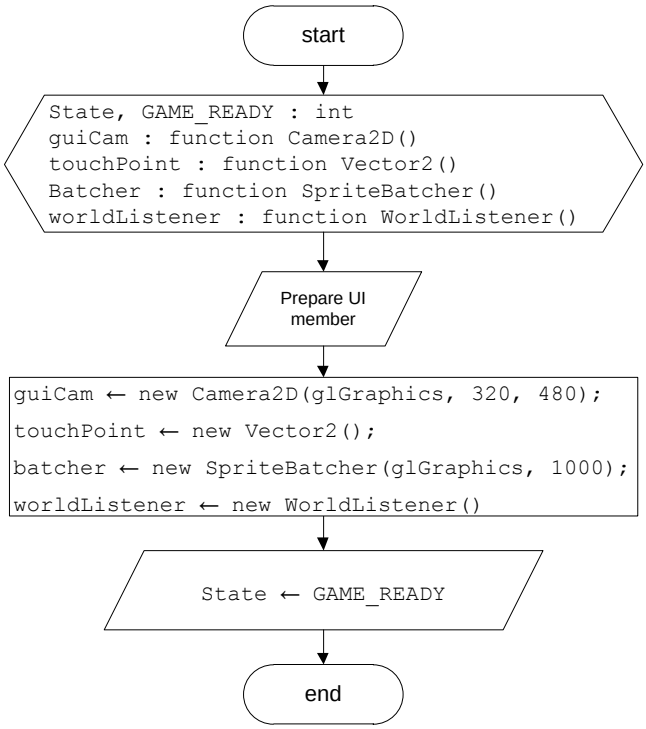
Method ini digunakan untuk mengatur kamera dan grafik 2D yang dibutuhkan dalam permainan. Untuk lebih jelasnya dapat dilihat dari *flowchart* pada gambar berikut ini :



Gambar 3.33 *Flowchart Method present*

2. *Method GameScreen*

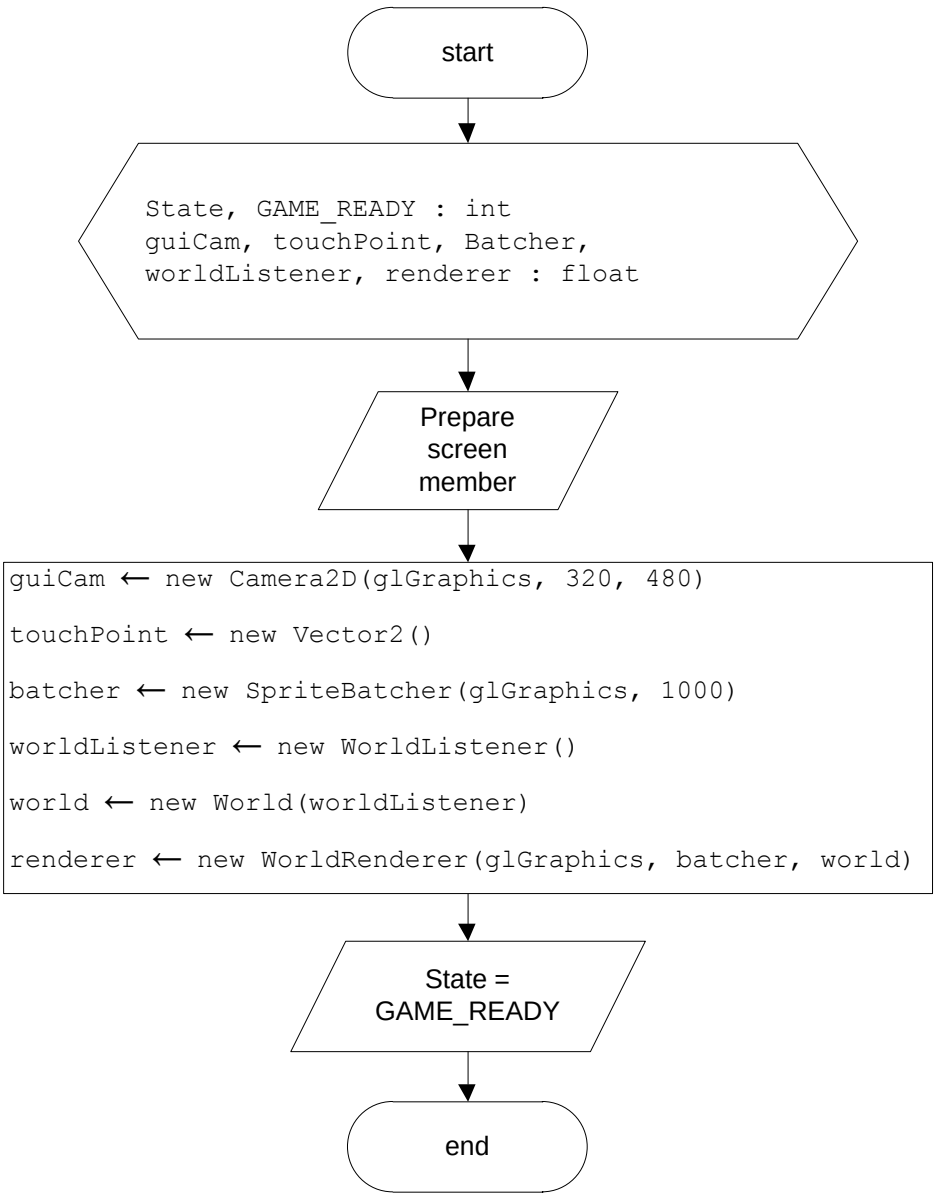
Method ini digunakan untuk menginisialisasi semua variabel anggota dalam *screen* permainan. Untuk lebih jelasnya dapat dilihat dari *flowchart* pada gambar berikut ini :



Gambar 3.34 *Flowchart Method GameScreen*

3. *Method getStartScreen*

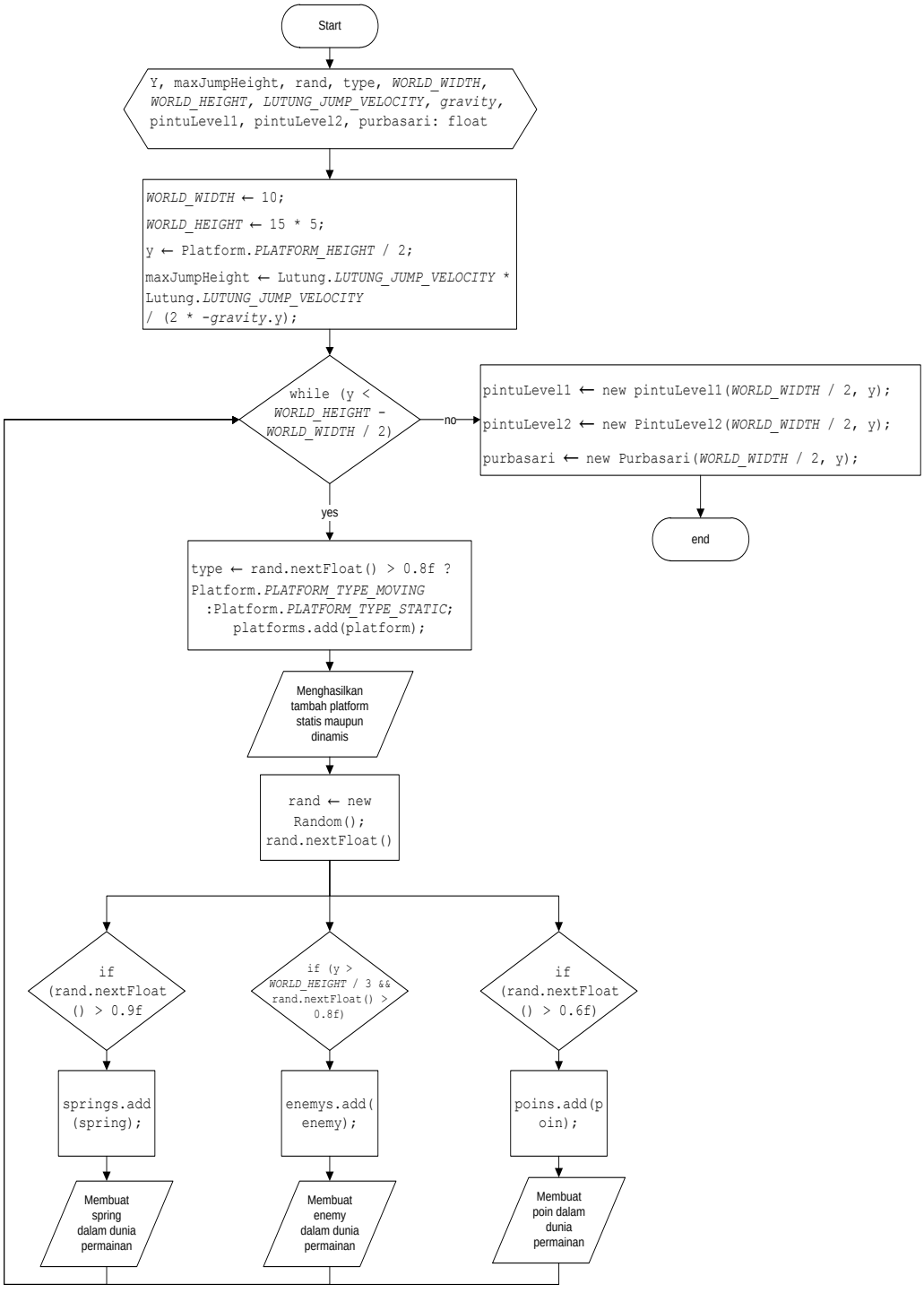
Method ini digunakan untuk mendapatkan *state game ready* pada *screen* awal permainan. Untuk lebih jelasnya dapat dilihat dari gambar *flowchart* berikut ini :



Gambar 3.35 *Flowchart Method getStartScreen*

4. *Method generatelevel*

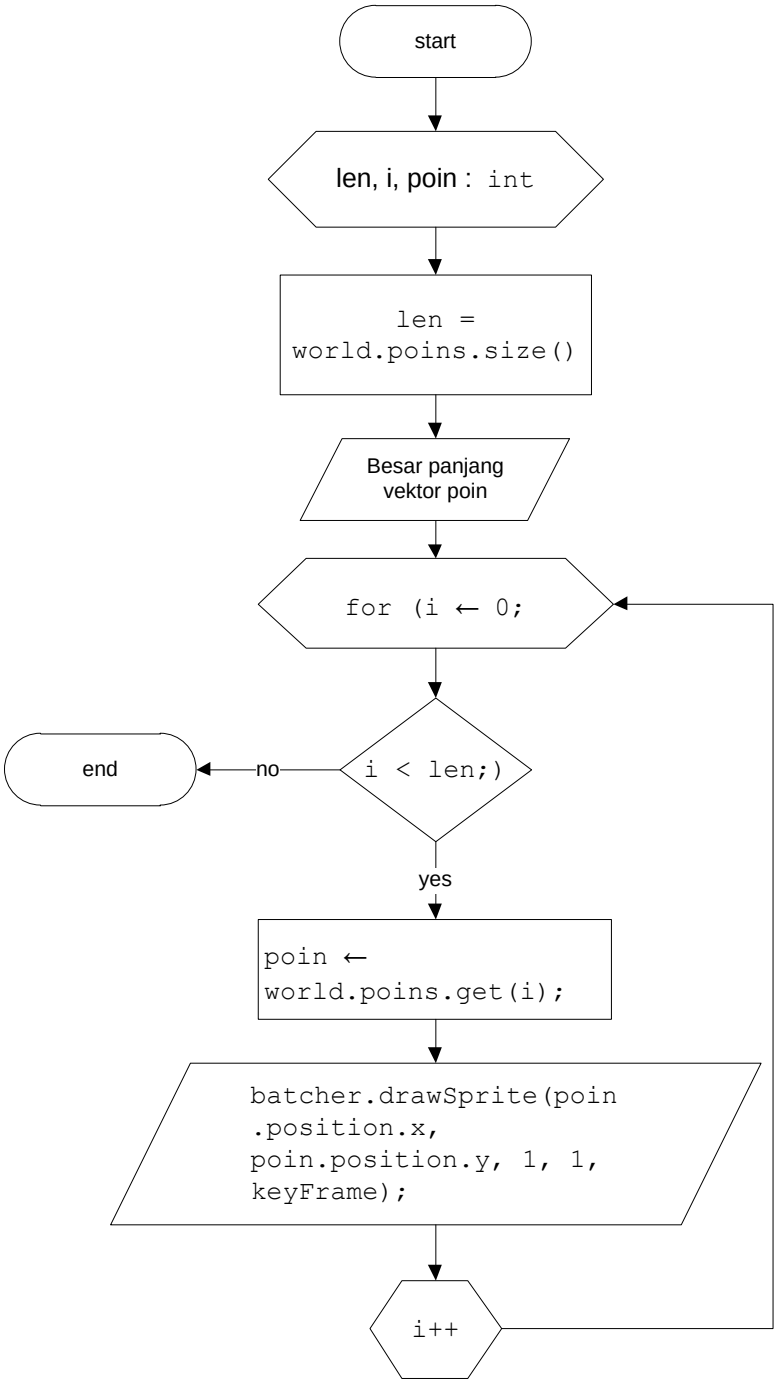
Method ini digunakan untuk membuat dan menempatkan konten-konten yang dibutuhkan kedalam dunia permainan. Dalam penempatannya, digunakanlah metode *procedural content generation*. Untuk lebih jelasnya dapat dilihat dari gambar *flowchart* berikut ini :



Gambar 3.36 *Flowchart Method generateLevel*

5. *Method renderPoin*

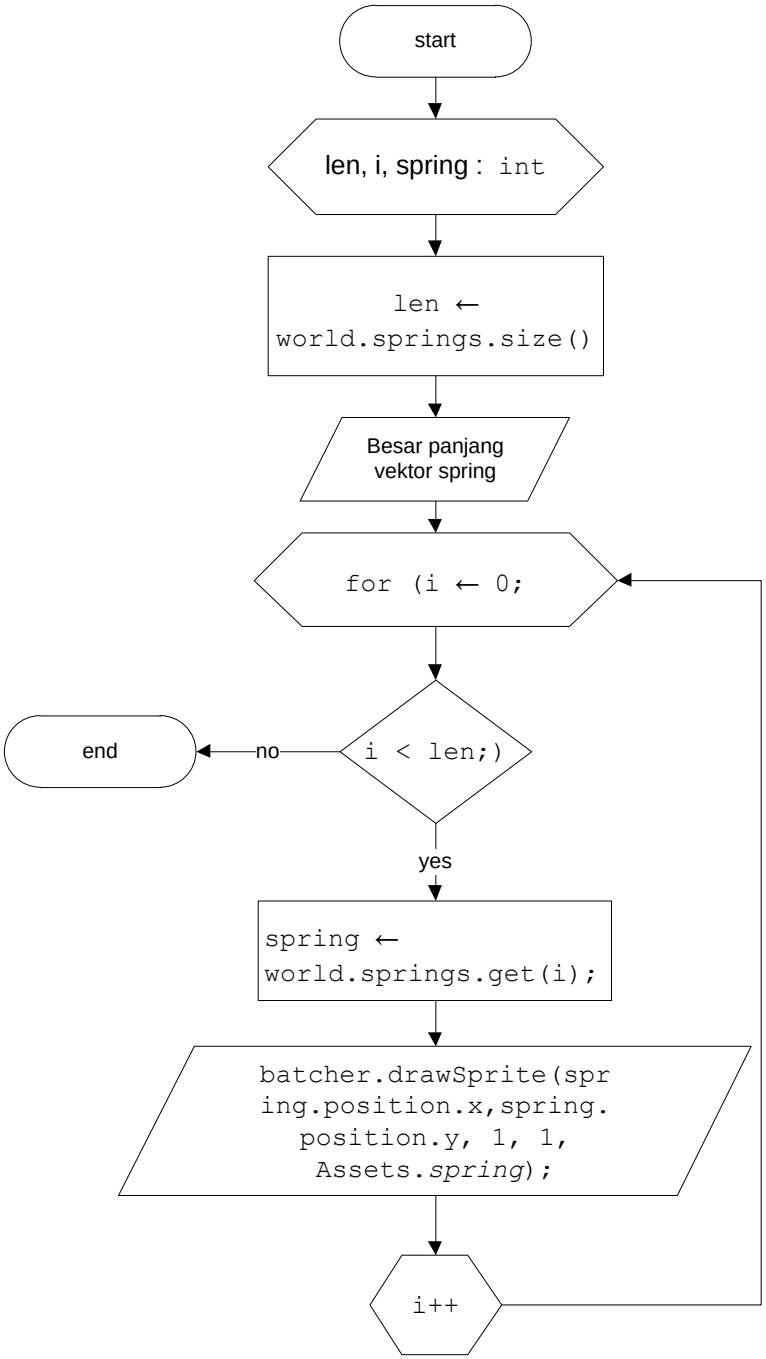
Method ini berfungsi untuk merender kemunculan dari setiap poin. Untuk lebih jelasnya dapat dilihat dari *flowchart* berikut ini :



Gambar 3.37 *Flowchart Method renderPoin*

6. *Method renderSpring*

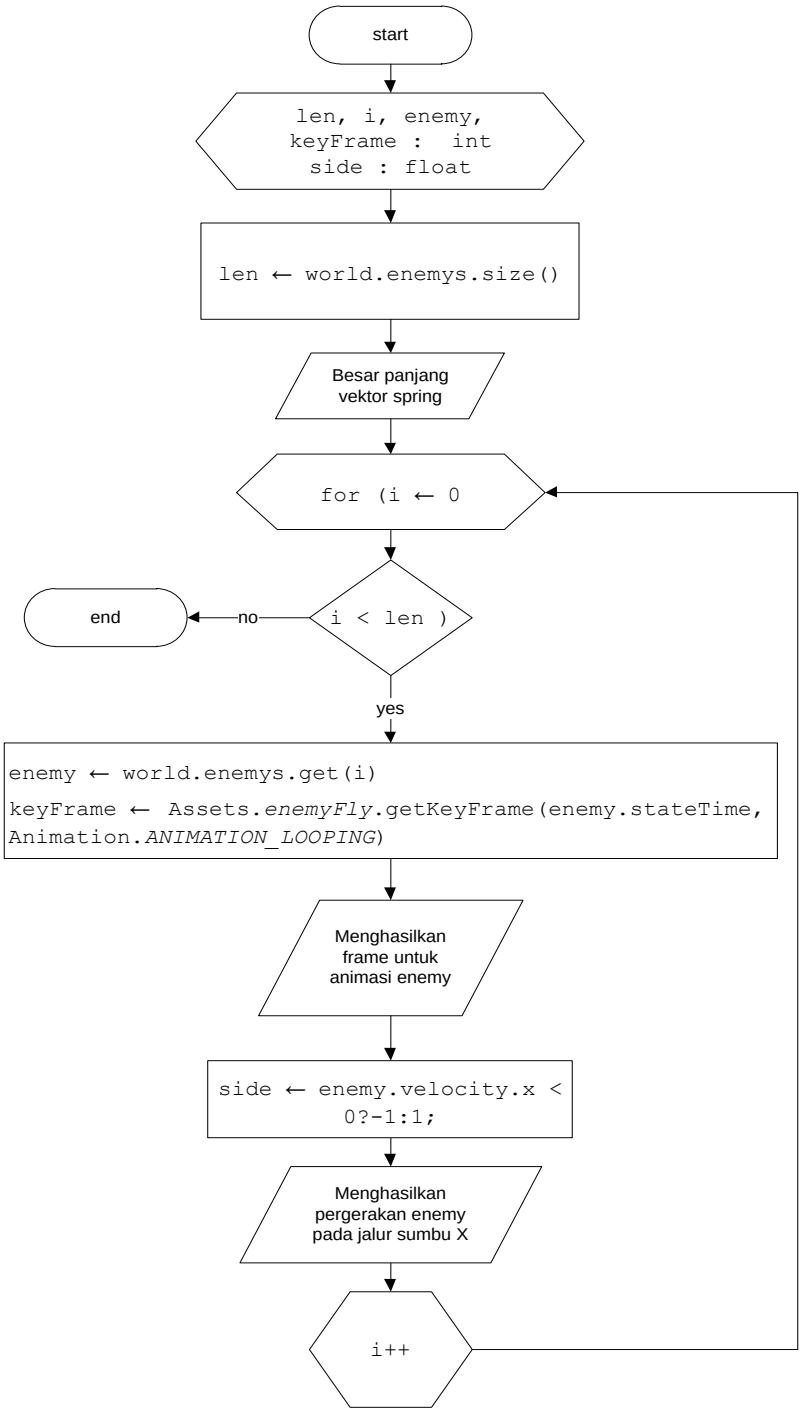
Method ini berfungsi untuk merender kemunculan dari setiap spring. Untuk lebih jelasnya dapat dilihat dari *flowchart* berikut ini :



Gambar 3.38 *Flowchart Method renderSpring*

7. *Method renderEnemy*

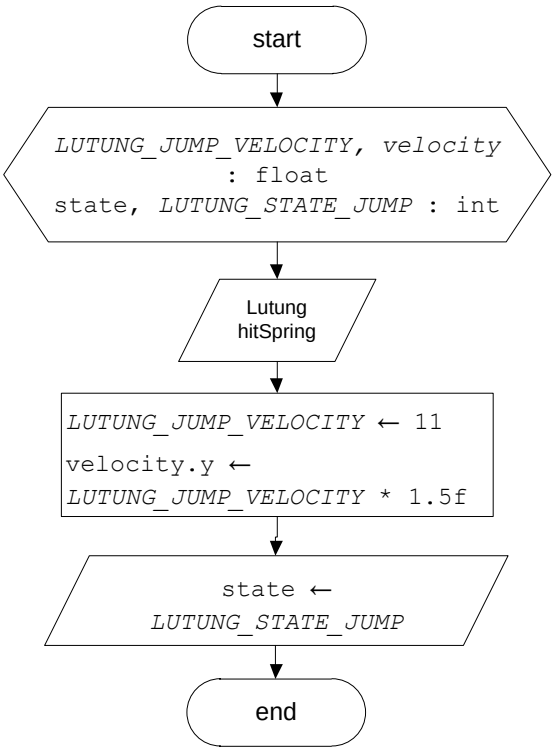
Method ini berfungsi untuk merender kemunculan dari setiap *enemy*. Untuk lebih jelasnya dapat dilihat dari gambar *flowchart* berikut ini :



Gambar 3.39 *Flowchart renderEnemy*

8. *Method highJump*

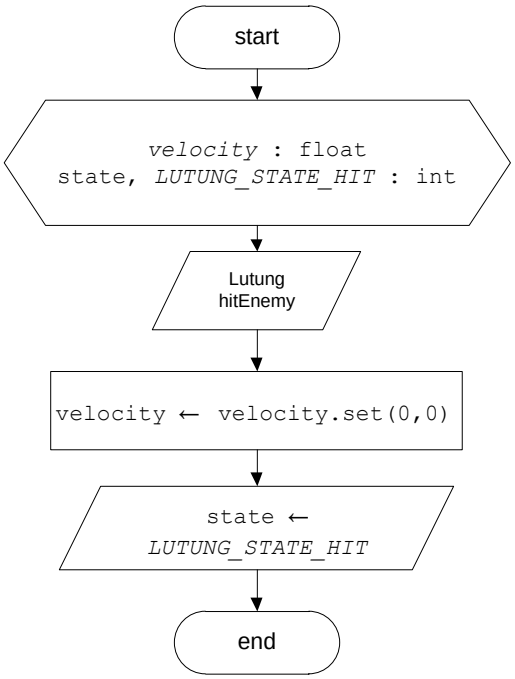
Method ini digunakan untuk karakter lutung yang mengalami *event hitSpring*, sehingga lompatannya akan menjadi 1,5 lebih tinggi dari lompatan normal. Untuk lebih jelasnya dapat dilihat dari gambar *flowchart* berikut ini :



Gambar 3.40 *Flowchart Method highJump*

9. *Method hit*

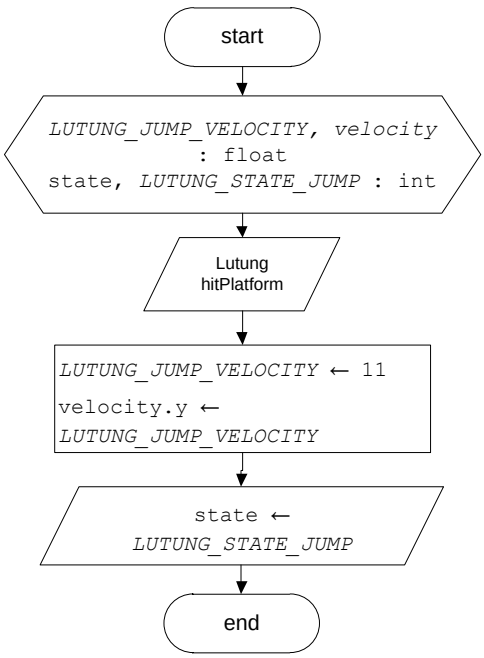
Method ini digunakan untuk karakter lutung yang mengalami *event hitEnemy*. Untuk lebih jelasnya dapat dilihat dari gambar *flowchart* berikut ini :



Gambar 3.41 *Flowchart Method hit*

10. *Method jump*

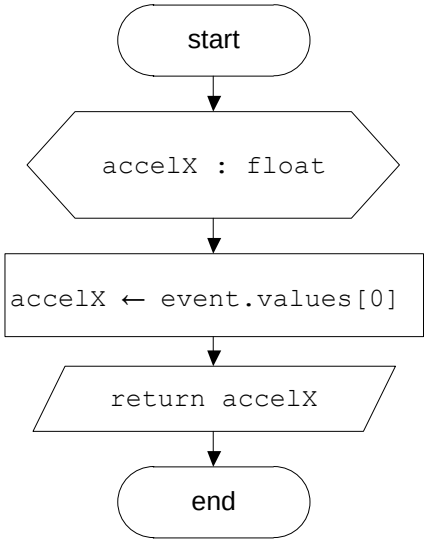
Method ini digunakan untuk karakter lutung yang mengalami *event hitPlatform*. Untuk lebih jelasnya dapat dilihat dari gambar *flowchart* berikut ini :



Gambar 3.42 *Flowchart Method jump*

11. *Method getAccelX*

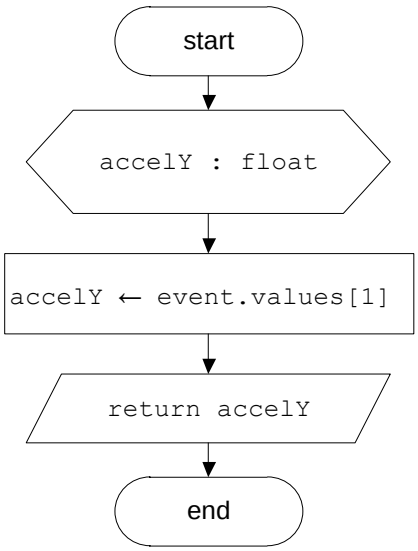
Method ini berfungsi untuk mengembalikan nilai percepatan kepada sumbu X. Untuk lebih jelasnya dapat dilihat pada gambar *flowchart* berikut ini :



Gambar 3.43 *Flowchart Method getAccelX*

12. *Method getAccelY*

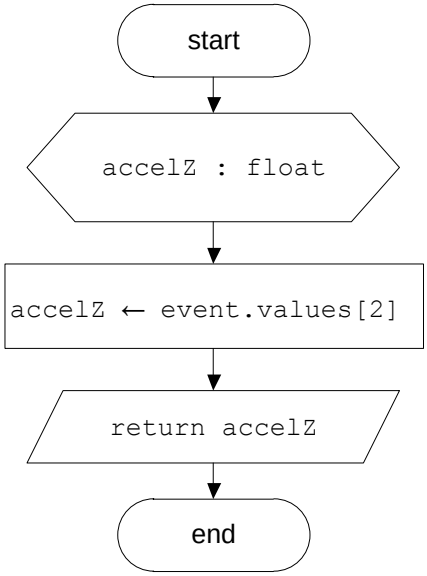
Method ini berfungsi untuk mengembalikan nilai percepatan kepada sumbu Y. Untuk lebih jelasnya dapat dilihat pada gambar *flowchart* berikut ini :



Gambar 3.44 *Flowchart Method getAccelY*

13. *Method getAccelZ*

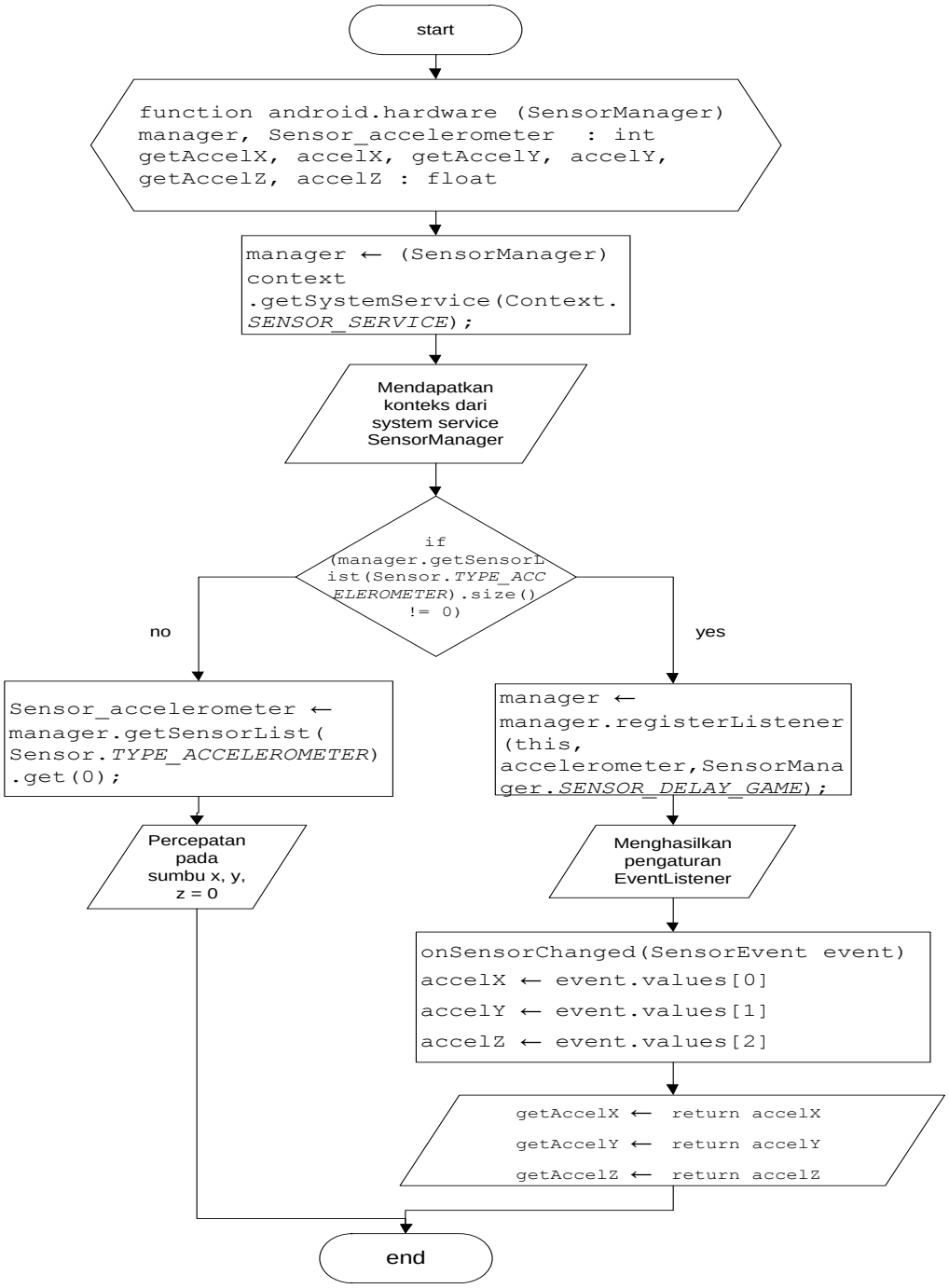
Method ini berfungsi untuk mengembalikan nilai percepatan kepada sumbu Z. Untuk lebih jelasnya dapat dilihat pada gambar *flowchart* berikut ini :



Gambar 3.45 *Flowchart Method getAccelZ*

14. *Method AccelerometerHandler*

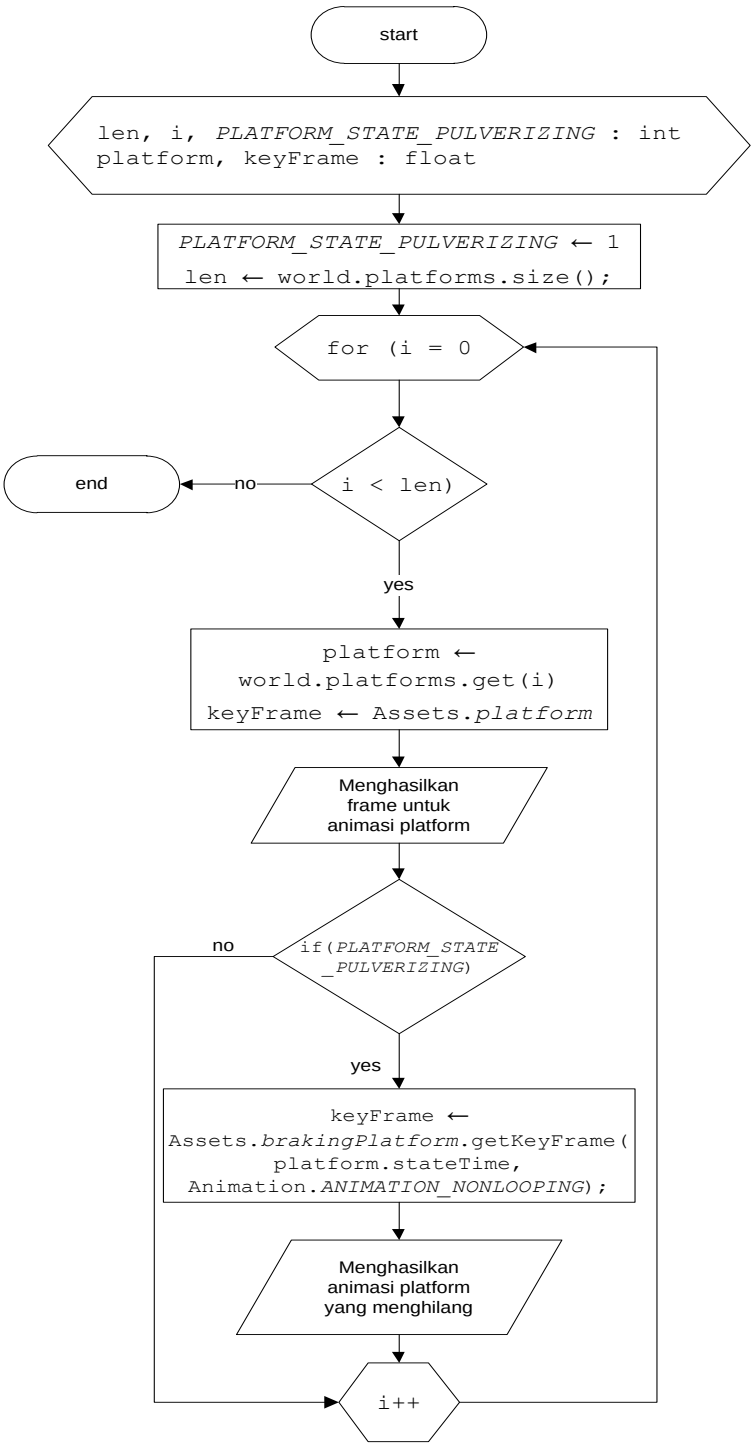
Method ini berfungsi untuk melakukan pemeriksaan sensor *accelerometer* terhadap device pengguna. Untuk lebih jelasnya dapat dilihat dari *flowchart* berikut ini :



Gambar 3.46 *Flowchart Method AccelerometerHandler*

15. *Method renderPlatform*

Method ini berfungsi untuk me-render kemunculan dari setiap platform. Untuk lebih jelasnya dapat dilihat dari gambar *flowchart* 3.46 :



Gambar 3.47 *Flowchart Method renderPlatform*