BAB 3
ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Sistem

Analisis sistem dapat didefinisikan sebagai penguraian dari suatu sistem yang utuh kedalam bagian-bagian komponennya dengan maksud untuk mengidentifikasi dan mengevaluasi kebutuhan-kebutuhan yang diharapkan sehingga dapat diusulkan perbaikan-perbaikannya.

3.1.1 Analisis Masalah

Permainan tradisional merupakan salah satu warisan budaya, Salah satu perwujudan budaya tersebut adalah pada permainan anak-anak. Permainan ini diajarkan dari generasi ke generasi sehingga bisa menjadi sebuah tradisi yang merupakan warisan budaya. Hasil penelitian para ahli, bahwa permainan tradisional yang populer pada masanya dan sering dimainkan adalah permainan kelereng yang menggunakan kelereng sebagai medianya. Peraturan yang sederhana membuat permainan ini terkenal di kalangan masyarakat khususnya anak-anak. Daya tarik dari permainan ini terletak pada adanya unsur prediksi bidikan dan mengumpulkan kelereng sebagai hadiah apabila memenangkan permainan.

Perkembangan teknologi dapat dimanfaatkan sebagai alat yang dapat menjaga keberadaan permainan tradisional. Unsur daya tarik pada permainan tradisional dan teknologi digabungkan, maka dapat diciptakan *Game* yang menggunakan unsur prediksi bidikan permainan tradisional sebagai dasarnya.

3.1.2 Analisis *Game* Sejenis

Pada tahap ini, dilakukan analisis pada *Game* sejenis sebagai perbandingan. *Game* yang dipilih adalah *Game Kelereng (b)* yang biasanya dapat dimainkan secara *mobile* menggunakan mesin Android, *Game* kelereng (b) ini dipopulerkan oleh androidpit.com (The wali studio). Konsep *Game* cukup sederhana dimana tujuannya mengeluarkan kelereng yang ada pada garis

lingkaran dengan menghindari hambatan statis berupa kotak – kotak kecil. Tampilan *Game* kelereng (b) dapat dilihat pada gambar 3.1.

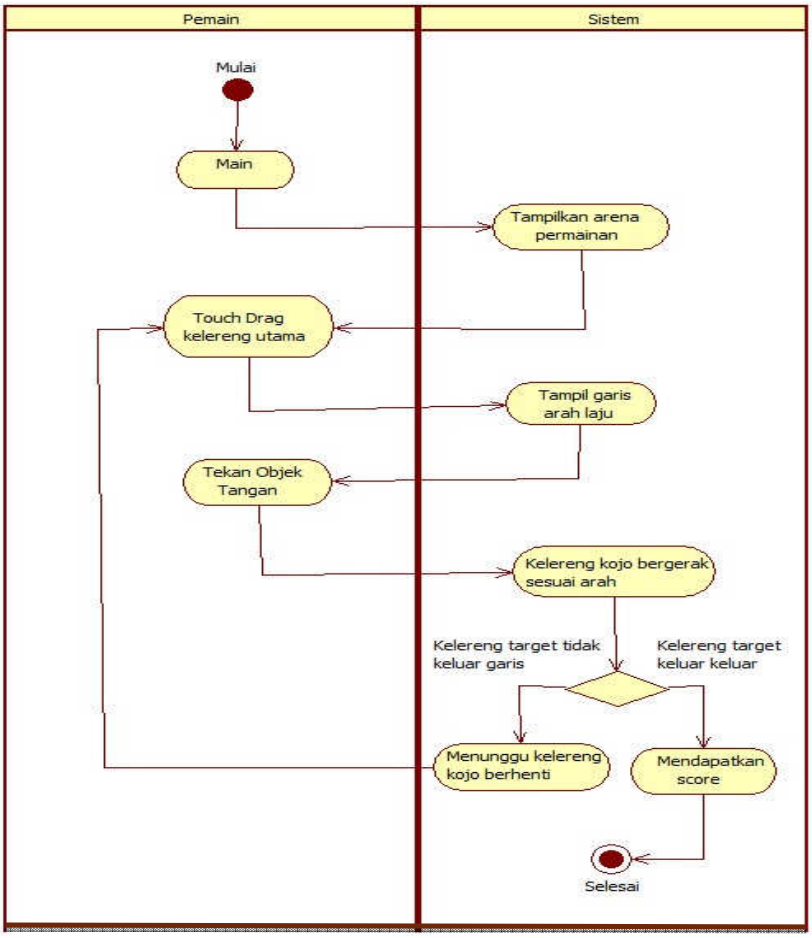


Gambar 3.*1 Game* Kelereng (b)

1. *Game* Play Pada *Game* Sejenis

Game kelereng (b) merupakan sebuah *Game* dengan mode permainan *single player*. Tugas utama pemain dalam memainkan *Game* ini adalah mengeluarkan kelereng yang ada didalam kalang dan menghindari hambatan – hambatan yang ada pada setiap *level*.

Cara bermain *Game* kelereng (b) ini yaitu pemain menggunakan *touchsreen* sebagai media interaksi. *Touchscreen* digunakan sebagai *controller* yaitu untuk mengarahkan dan menjentik kelereng. Pertama *touch drag* objek kelereng utama dan arahkan sesuai keinginan pemain, lalu setelah itu *touch* objek tangan untuk mulai menggerakan objek kelereng utama tersebut. Aktivity *Diagram Game* kelereng (b) dapat dilihat pada gambar 3.2.



Gambar 3.2 *Activity Diagram Game* Kelereng (b)

2. Level *Game* Sejenis

Game ini memiliki dua puluh tujuh *level*, yang disetiap *level*-nya memiliki jumlah kelereng yang berbeda pada kalang dan hambatan yang ada hanya berupa kotak dan jumlah kotak pada setiap *level* berbeda.

3. Scoring *Game* Sejenis

Scoring pada *Game* kelereng (b) ini dihasilkan apabila dapat mengeluarkan kelereng target yang terdapat didalam kalang. Setiap mengeluarkan satu buah kelereng maka pemain akan mendapatkan *score* 2 *point.*

3.1.3 Analisis *Game* yang Akan Dikembangkan

Game yang akan dibangun adalah *Game* Tradisional Kelereng. *Game* ini dibangun dengan mengaplikasikan teknologi sebagai jembatan untuk memainkan permainan kelereng secara *digital*. Berikut batasan – batasan yang ada pada *Game* ini :

1. *Game* Tradisional Kelereng ini dijalankan secara *offline.*
2. Sistem permainan bersifat *single player.*
3. Sasaran pengguna untuk umum.
4. *Game* bersifat bermain dan disetiap *level* memiliki tingkat kesulitan hambatan yang berbeda.
5. Grafis 2D (2 Dimensi).

1. *Game Play* Pada *Game* yang Akan Dikembangkan

Di awal permainan, pemain diberikan pilihan untuk memilih main, cara bermain, tentang dan keluar. Jika pemain memilihan main maka akan langsung memainkan *Game* tradisional kelereng, Jika pemain memilih cara bermain maka akan muncul tampilan cara memainkan *Game* tradisional kelereng, jika pemain memilih tentang maka akan muncul tampilan tentang pembuat *Game* dan pilihan keluar utuk keluar dari *Game* tradisional kelereng.

Cara bermain *Game* tradisional kelereng ini yaitu pemain menggunakan *touchsreen* sebagai media interaksi. *Touchscreen* digunakan sebagai *controller* yaitu untuk mengarahkan dan menggerakan kelereng utama. Pertama *touch drag* objek kelereng utama dan arahkan untuk mengarahkan bidikan, lalu *touch* objek tangan untuk mulai menggerakan objek kelereng utama tersebut. Apabila memenangkan permainan, touch tombol menu pada perangkat untuk melanjutkan kelevel selanjutnya. *Game* tradisional kelereng ini hanya menampilkan garis

batas/kalang berbentuk oval dan apabila kelerreng utama berhenti di dalam batas garis/kalang maka permainan berhenti/kalah.

Tugas utama pemain dalam memainkan *Game* ini adalah mengeluarkan kelereng yang ada didalam kalang dan menghindari hambatan – hambatan yang ada pada setiap *level* dengan posisi kelereng target dan waktu yang telah ditentukan.

2. Level *Game* yang Akan Dikembangkan

Tingkat kesulitan 3 *level.* Pada *level* 1 pemain harus mengeluarkan 5 kelereng yang ada didalam kalang dan harus menghindari hambatan batu yang bisa menghalangi laju kelereng utama dengan diberi waktu selama 3 menit untuk menyelesaikannya. Pada *level* 2 pemain harus mengeluarkan 10 kelereng didalam kalang dan menghindari hambatan dinamis berupa batu dan semut hitam, bila kelereng utama mengenai semut hitam maka akan mengurangi 3 *point,* paada *level* 2 ini hanya diberi waktu 4 menit untuk menyelesaikannya. Pada *level* 3 pemain harus mengeluarkan 15 kelereng didalam kalang, pemain harus lebih *focus* dan hati – hati karena pada *level* 3 ini selain ada batu – batu dan semut hitam akan ada hambatan baru yaitu semut merah, bila kelereng utama mengenai semut merah ini akan mengurangi 5 *point* dan hanya diberi waktu selama 5 menit untuk menyelesaikannya.

3. Scoring *Game* yang Akan Dikembangkan

Scoring pada *Game* ini dilakukan dalam beberapa hal :

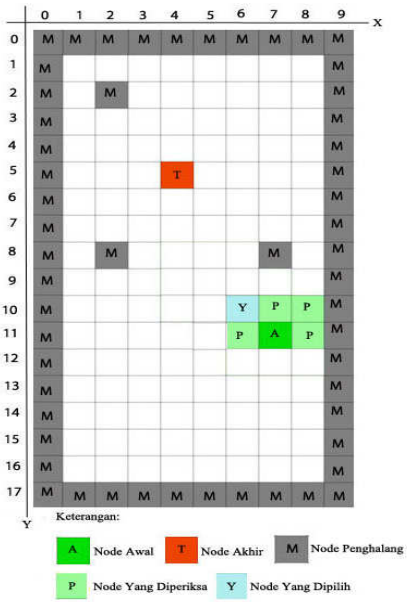
1. Setiap mengeluarkan satu kelereng pada kalang akan menghasilkan *score* dua *point*.
2. Apabila kelereng utama mengenai hambatan berupa semut hitam maka s*core* akan berkurang, minus tiga *point.*
3. Apabila kelereng utama mengenai hambatan berupa semut merah maka *score* akan berkurang, minus Lima *point.*

3.1.4 Analisis Algoritma *Greedy Best-First Search*

Analisis *algoritma greedy best-first search* yang digunakan pada tugas akhir ini digunakan sebagai pengendali untuk menentukan pergerakan hambatan semut (S) untuk mencari jalan menuju objek kelereng target (K). Pencarian jalan terpendek pada *Game* ini akan diasumsikan dengan simpul (*node*) awal dan Simpul (*node*) tujuan.

Gambar 3.3 terlihat bahwa terdapat 3 objek dimana A adalah Objek hambatan semut, T adalah objek kelereng target dan M adalah sebagai batas layar dan batu penghalang. Objek A dan T dipergunakan untuk menunjukkan lokasi *node* awal (A) dan *node* akhir (T). Dalam proses pencarian algoritma *greedy best-first search* terdapat 2 nilai yaitu h dan f. Nilai h adalah biaya perkiraannya dan rumus untuk mendapatkan nilai *heuristic* adalah *h(n) = abs(n.x-goal.x) + abs(n.y-goal.y)*, sedangkan f adalah total biaya yang diperhitungkan dalam notasi matematika algoritma *greedy best-first* dapat dituliskan dengan *f(n) = h(n).*

Berikut adalah langkah perhitungan dalam pencarian jalur menuju *node* akhir (K) dijelaskan pada ilustrasi gambar 3.3.



Gambar 3.3 Kondisi awal pencarian *Greedy Best-First Search*

Posisi *node* awal = currentX = 7, currentY =11

Posisi *node* tujuan = targetX = 4, targetY = 5

Langkah pertama :

H (7,11) = (abs(currentX-targetX) + abs(currentY-targetY))
= (abs(7-4) + abs(11-5))
= 3+6
= 9

F (7,11) = H(7,11)
= 9

H (8,11) = (abs(currentX-targetX) + abs(currentY-targetY))
= (abs(8-4) + abs(11-5))
= 4+6
= 10

F (8,11) = H(8,11)
= 10

H (6,11) = (abs(currentX-targetX) + abs(currentY-targetY))
= (abs(6-4) + abs(11-5))
= 2+6
= 8

F (6,11) = H(6,11)
= 8

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

F (8,10) = H(8,10)
= 9

H (7,10) = (abs(currentX-targetX) + abs(currentY-targetY))
= (abs(7-4) + abs(10-5))
= 3+5
= 8

F (7,10) = H(7,10)

= 8

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

= (abs(6-4) + abs(10-5))

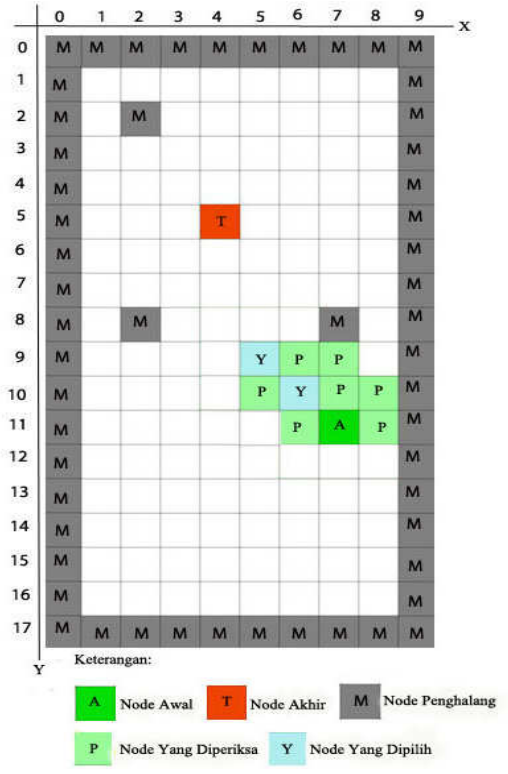
= 2+5

= 7

F (6,10) = H(6,10)

= 7

Gambar 3.3 terdapat lima *node* yang mungkin menjadi *best node* yaitu (8,11) dengan F(n) = 10, (6,11) dengan F(n) = 8, (8,10) dengan F(n) = 9, (7,10) dengan F(n) = 8 dan (6,10) dengan F(n) = 7 maka yang dipilih adalah *node* (6,10) dengan biaya 7.



Gambar 3.4 Langkah Kedua Pencarian *Best Node*

Langkah kedua :

H (5,10) = (abs(currentX-targetX) + abs(currentY-targetY))
= (abs(5-4) + abs(10-5))
= 1+5
= 6

F (5,10) = H(5,10)
= 6

H (7,9) = (abs(currentX-targetX) + abs(currentY-targetY))
= (abs(7-4) + abs(9-5))
= 3+4
= 7

F (7,9) = H(7,9)
= 7

H (6,9) = (abs(currentX-targetX) + abs(currentY-targetY))
= (abs(6-4) + abs(9-5))
= 2+4
= 6

F (6,9) = H(6,9)
= 6

H (5,9) = (abs(currentX-targetX) + abs(currentY-targetY))
= (abs(5-4) + abs(9-5))
= 1+4
= 5

F (5,9) = H(5,9)
= 5

Gambar 3.4 terdapat empat *node* yang mungkin menjadi *best node* yaitu (5,10) dengan F(n) = 6, (7,9) dengan F(n) = 7, (6,9) dengan F(n) = 6 dan (5,9) dengan F(n) = 5 maka yang dipilih adalah *node* (5,9) dengan biaya 5.



Gambar 3.5 Langkah Ketiga Pencarian *Best Node*

Langkah ketiga :

H (4,9) = (abs(currentX-targetX) + abs(currentY-targetY))

= (abs(4-4) + abs(9-5))

= 0+4

= 4

F (4,9) = H(4,9)

= 4

H (6,8) = (abs(currentX-targetX) + abs(currentY-targetY))

= (abs(6-4) + abs(8-5))

= 2+3

= 5

F (6,8) = H(6,8)

= 5

H (5,8) = (abs(currentX-targetX) + abs(currentY-targetY))

= (abs(5-4) + abs(8-5))

= 1+3

= 4

F (5,8) = H(5,8)

= 4

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

= (abs(4-4) + abs(8-5))

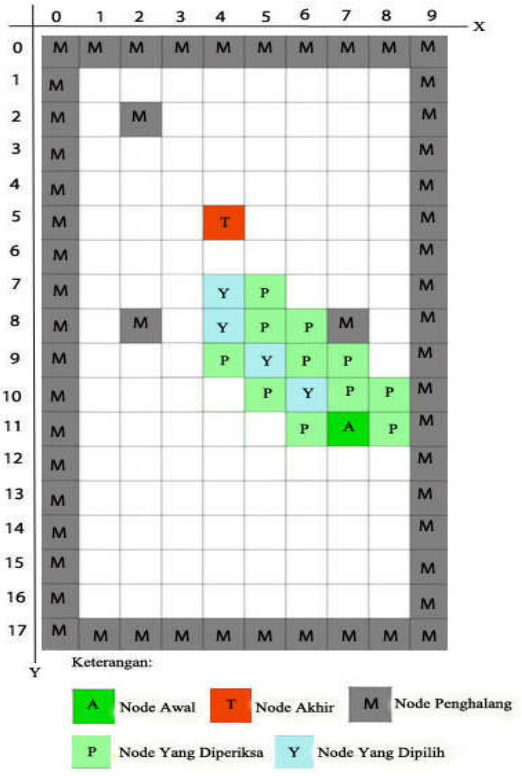
= 0+3

= 3

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

= 3

Gambar 3.5 terdapat empat *node* yang mungkin menjadi *best node* yaitu (4,9) dengan F(n) = 4, (6,8) dengan F(n) = 5, (5,8) dengan F(n) = 4 dan (4,8) dengan F(n) = 3 maka yang dipilih adalah *node* (4,8) dengan biaya 3.



Gambar 3.6 Langkah Keempat Pencarian *Best Node*

Langkah keempat :

H (5,7) = (abs(currentX-targetX) + abs(currentY-targetY))

= (abs(5-4) + abs(7-5))

= 1+2

= 3

F (5,7) = H(5,7)

= 3

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

= (abs(4-4) + abs(7-5))

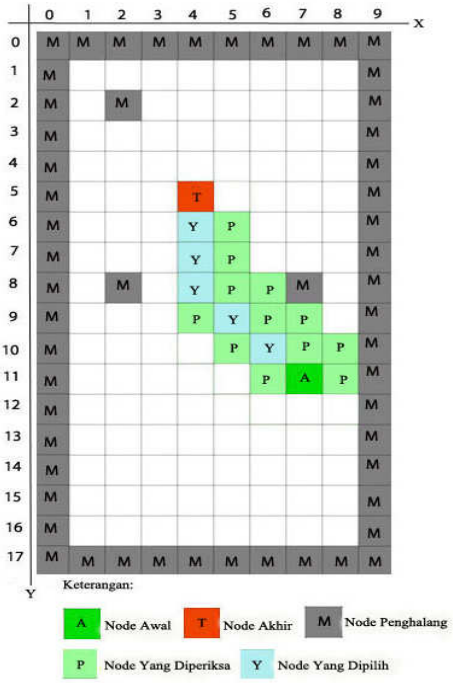
= 0+2

= 2

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

= 2

Gambar 3.6 terdapat dua *node* yang mungkin menjadi *best node* yaitu (5,7) dengan F(n) = 3 dan (4,7) dengan F(n) = 2, maka yang dipilih adalah *node* (4,7) dengan biaya 2.



Gambar 3.7 Langkah KelimaPencarian *Best Nod*

Langkah kelima :

H (5,6) = (abs(currentX-targetX) + abs(currentY-targetY))

= (abs(5-4) + abs(6-5))

= 1+1

= 2

F (5,6) = H(5,6)

= 2

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

= (abs(4-4) + abs(6-5))

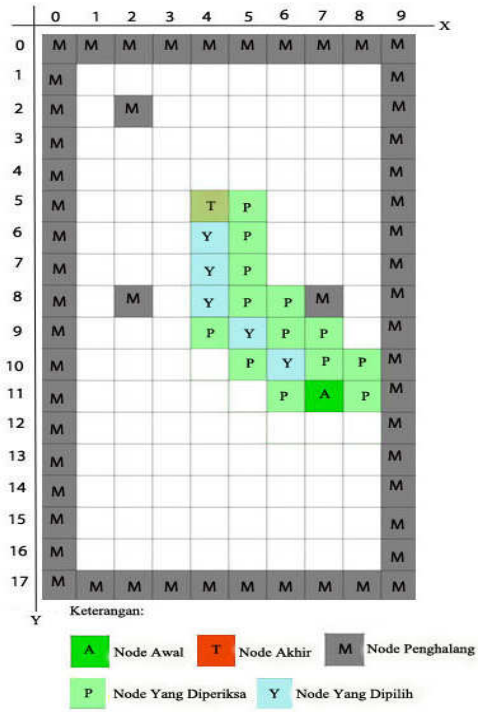
= 0+1

= 1

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

= 1

Gambar 3.7 terdapat dua *node* yang mungkin menjadi *best node* yaitu (5,6) dengan F(n) = 2 dan (4,6) dengan F(n) = 1, maka yang dipilih adalah *node* (4,6) dengan biaya 1.



Gambar 3.8 Langkah Keenam Pencarian *Best Node*

Langkah keenam :

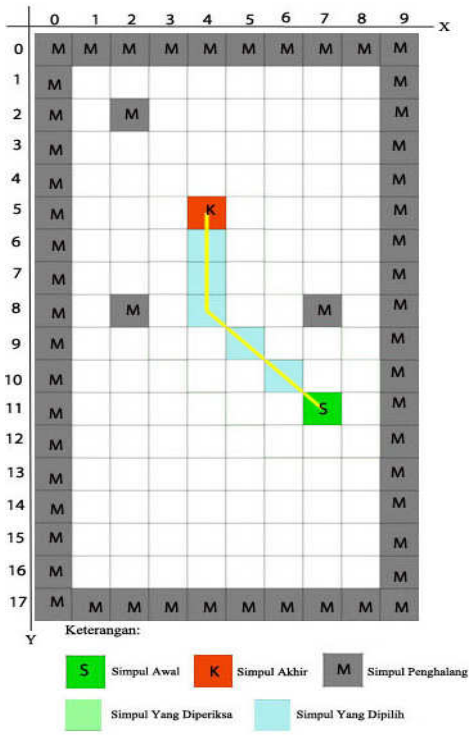
H (5,5) = (abs(currentX-targetX) + abs(currentY-targetY))
= (abs(5-4) + abs(5-5))
= 1+0
= 1

F (5,5) = H(5,5)
= 1

H (4,5) = (abs(currentX-targetX) + abs(currentY-targetY))
= (abs(4-4) + abs(5-5))
= 0+0
= 0

F (4,5) = H(4,5)
= 0

Gambar 3.8 terdapat dua *node* yang mungkin menjadi *best node* yaitu (5,5) dengan F(n) = 1 dan (4,5) dengan F(n) = 0, maka yang dipilih adalah *node* (4,6) dengan biaya 0 dan dikenali sebagai *node* tujuan.



Gambar 3.9 Hasil Pencarian Jalur Dengan *Greedy Best-Firs Search*

Gambar 3.9 merupakan hasil pencarian jalur dari *node* awal (A) menuju *node* akhir (T).

Pada algoritma *greedy best first search* tedapat beberapa prosedur didalamnya. *Berikut ini kamus global algoritma greedy best first Search yang diterapkan pada Game* tradisional kelereng dengan menggunakan struktur data single linked list.

Kamus global *single linked list*

{Algoritma}

```
Type
      A_node  ⟶      simpul
      simpul = record
      <xCoor : integer
        yCoor : integer
        ratiox : real
        ratioy : real
        F : real
        Openlist : integer
```

```
        Closelist : integer>
End record
bSemutM = A_node
```

Berikut ini adalah *pseudocode* pembuatan *node* dapat dilihat pada table 3.11.

Tabel 3. 1 *Pseudocode* Prosedur *node*

	Isi Pseudocode
1	`Procedure Node (input A_node : x, y)`
2	`{I.S : koordinat node sudah terdefinisi}`
3	`{F.S : menghasilkan node}`
4	`Kamus :`
5	`     Type`
6	`     xCoor : integer`
7	`     yCoor : integer`
8	`     ratiox : real`
9	`     ratioy : real`
10	`     F : real`
11	
12	`Algoritma :`
13	`  xCoor : x`
14	`  yCoor : y`
15	`  ratioX = (xCoor / Math.abs(xCoor))`
16	`            + Math.min(1, Math.abs(xCoor /yCoor))`
17	`  ratioY = (yCoor / Math.abs(yCoor))`
18	`            + Math.min(1, Math.abs(yCoor / xCoor))`
19	`  F = ratiox + ratioy`

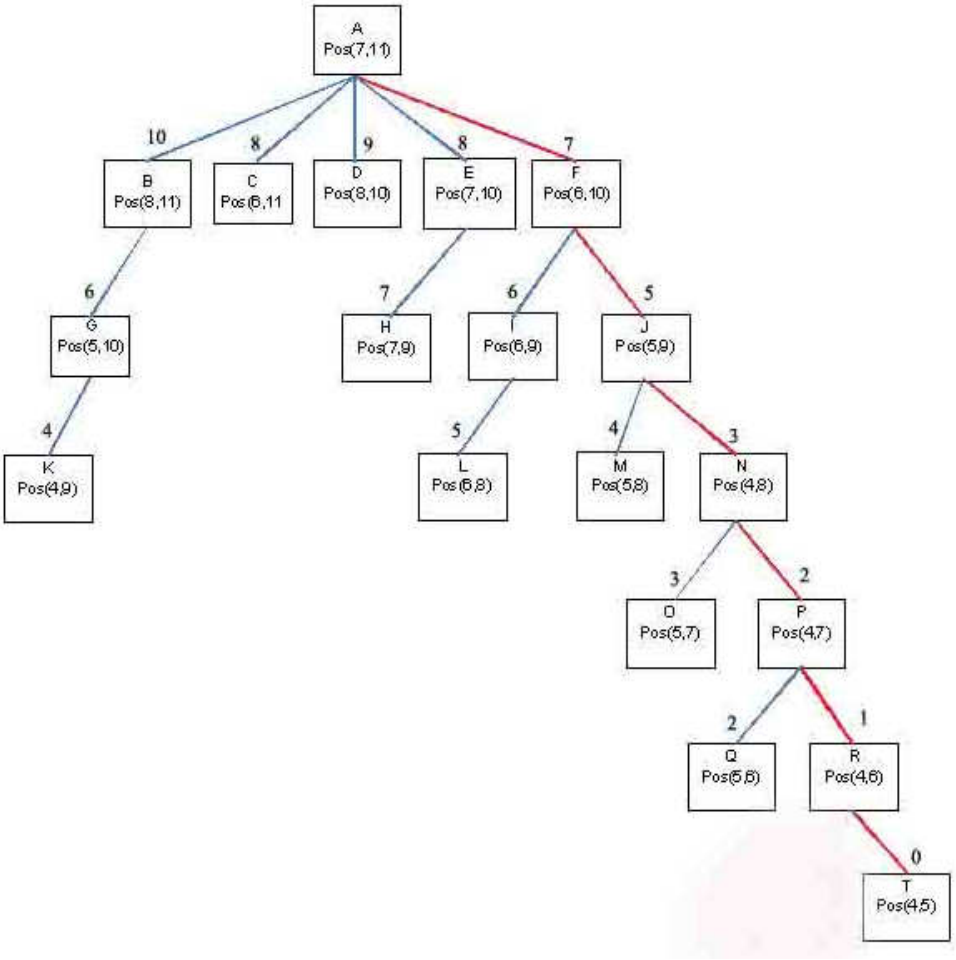
Berikut ini adalah pseudocode pembangkit suksesor dapat dilihat pada tabel 3.2.

Tabel 3. 2 *Pseudocode* Prosedur *checkhNode*

	Isi Pseudocode
1	`Procedure checknode`
2	`{I.S : node sudah terdefinisi}`
3	`{F.S : membangkitkan node sekitar}`
4	
5	`Kamus :`
6	`   Type`
7	`   openlist : integer`
8	`   closelist : integer`
9	`   ratiox : real`
10	`   ratioy : real`
11	`   F : real`
12	
13	`Algoritma :`
14	`  Iter = openlist.iterator()`
15	`  lastCheckedF : float`
16	`while (iter.hasnext())`
17	`  {`

```
12      lastCheckednode = iter.next()
13      lastCheckedF = lastCheckedNode.F;
        if (lastCheckedF <= smallestFfound) {F lebih kecil}
14      {
15       If(!.contains (lastCheckedNode)){
         {tidak berada di close list}
16        smallestFfound = lastCheckedF
         }
17    }
18
```

Gambar 3.10 adalah penjelasan langkah algoritma *greedy best first search.*



Gambar 3.10 Langkah Pencarian Algoritma *Greedy Best-First Search*

Langkah – Langkah urutan pencarian dari *node* A (awal) ke *node* T (Tujuan).

1. Open = [A]; closed=[]
2. Open = [B,C,D,E,F]; closed=[A]
3. Open = [I,J]; closed=[A,F]
4. Open = [M,N]; closed=[A,F,J]
5. Open = [O,P]; closed=[A,F,J,N]
6. Open = [Q,R]; closed=[A,F,J,N,P]
7. Open = [S,T]; closed=[A,F,J,N,P,R]
8. Open = [S]; closed=[A,F,J,N,P,R,T].

Langkah pertama, dimulai dari *node* awal (A) yang ditambahkan ke dalam *Open list* yang merupakan *list* untuk *node-node* yang harus diperiksa..

Langkah kedua, suksesor A dibangkitkan yaitu *node* B,C,D,E,F, cari biaya paling minimal dan didapatkan hasil :

f- *estimated cost*(B) = 10

f- *estimated cost*(C) = 8

f- *estimated cost*(D) = 9

f- *estimated cost*(E) = 8

f- *estimated cost*(F) = 7

Node F terpilih dengan biaya f- *estimated cost*(F) = 7

Langkah ketiga, suksesor F dibangkitkan yaitu *node* I dan J, cari biaya paling minimal dan didapatkan hasil :

f- *estimated cost*(I) = 6

f- *estimated cost*(J) = 5

Node J terpilih dengan biaya f- *estimated cost*(J) = 5.

Langkah keempat, suksesor J dibangkitkan yaitu *node* M dan N, cari biaya paling minimal dan didapatkan hasil :

f- *estimated cost*(M) = 4

f- *estimated cost*(N) = 3

Node N terpilih dengan biaya f- *estimated cost*(N) = 3.

Langkah kelima, suksesor N dibangkitkan yaitu *node* O dan P, cari biaya paling minimal dan didapatkan hasil :

f- *estimated cost*(O) = 3

f- *estimated cost*(P) = 2

Node P terpilih dengan biaya f- *estimated cost*(P) = 2.

Langkah keenam, suksesor P dibangkitkan yaitu *node* Q dan R, cari biaya paling minimal dan didapatkan hasil :

f- *estimated cost*(Q) = 2

f- *estimated cost*(R) = 1

Node R terpilih dengan biaya f- *estimated cost*(R) = 1.

Langkah ketujuh, suksesor R dibangkitkan yaitu *node* S dan T, cari biaya paling minimal dan didapatkan hasil :

f- *estimated cost*(S) = 1

f- *estimated cost*(T) = 0

Node T terpilih dengan biaya f- *estimated cost*(T) = 0.

Karena *node* telah sampai ke tujuan maka pencarian dihentikan dengan menghasilkan jalur *node* A-F-J-N-P-R-T.

3.1.5 Analisis Kebutuhan Non Fungsional

Analisis kebutuhan nonfungsional meliputi analisis kebutuhan perangkat lunak, analisis kebutuhan perangkat keras, serta analisis user.

3.1.5.1 Analisis Perangkat Keras (*Hardware*)

Berdasarkan analisis spesifikasi perangkat keras minimum yang dibutuhkan untuk membangun aplikasi ini adalah :

1. Processor minimal 2 Ghz
2. Hardisk minimal 5 GB
3. RAM minimal 512 MB
4. Grapic Card minimal 256 MB
5. Mouse & Keyboard.

Sedangkan spesifikasi perangkat keras pada smartphone Android minimum yang dibutuhkan adalah :

1. Processor minimal 500 mhz
2. RAM minimal 256 MB

3. Layar *Touchscreen*
4. Menggunakan layar dengan dimensi 320x480 pixel
5. Baterai standar
6. Sistem Operasi minimal Android versi 2.3 (Gingerbread), karena Os Android versi 2.3 (Gingerbread) tela mendukung library grafis 2D dan grafis 3D yang terkostumisasi berdasarkan spesifikasi openGL yang membuat tampilan grafis lebih halus.

Spesifikasi minimal perangkat keras yang tercantum pada umumnya perangkat smartphone yang beredar dipasaran sekarang ini telah memenuhi persyaratan minimum untuk menjalankan aplikasi sehingga tidak memerlukan penambahan software penunjang untuk memainkan *Game* tradisional kelereng.

3.1.5.2 Analisis Perangkat Lunak (*Software*)

Perangkat lunak diguanakan dalam sebuah sistem merupakan perintah-perintah yang diberikan kepada perangkat keras agar bisa saling berinteraksi di antara keduanya. Perangkat lunak yang dibutuhkan untuk membangun *Game* tradisional kelereng ini adalah sebgai berikut :

1. Sistem Operasi Windows 7
2. Java Runtime Edition
3. Eclipse versi 3.6 (Helios)
4. ADT (Android Development Tools)
5. AVD (Android Virtual Device)
6. Adobe Photoshoap CS5
7. Adobe Illustrator CS5.

3.1.5.3 Analisis Pengguna (*User*)

Pada dasarnya *Game* ini diperuntukkan untuk semua kalangan. Namun untuk spesifikasi pengguna atau kriteria minimal yang harus dimiliki oleh pengguna adalah sebagai berikut :

1. Mengerti operasional *Smartphone* Android
2. Dapat mengoperasikan aplikasi pada *Smartphone* Android
3. Memiliki *Smartphone* Android dengan minimal versi 2.3 (*Gingerbread*).

3.1.5.4 Karakteristik Pengguna

Karakteristik pengguna untuk *Game* tradisional kelereng ini adalah pemain. Dapat dilihat di tabel 3.3.

Tabel 3. 3 Karakteristik Pengguna

Pengguna	Tanggung Jawab	Hak Akses	Tingkat Pendidikan	Tingkat Keterampilan	Pengalaman
Pemain	Melakukan ermainan	Menjalankan aplikasi dan seluruh proses didalamnya	SD (yang tertarik dengan permainan kelereng	Bisa mengoprerasikan *Smartphone* Android	-

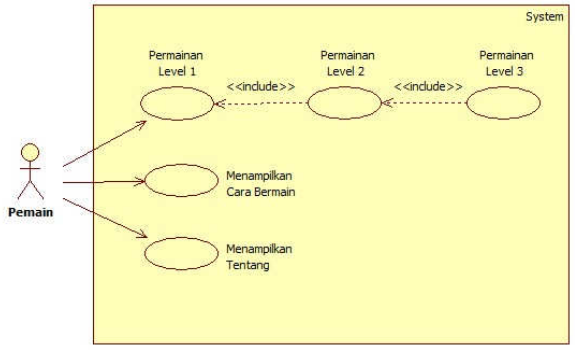
3.1.6 Analisis Kebutuhan Fungsional

Tahap pertama yang dilakukan dalam melakukan analisis berorientasi objek menggunakan UML adalah menentukan aktor atau pengguna sistem. Aktor menampilkan peran (*roles*) yang dilakukan pada sistem sebagai aksi dan respon dari sistem.

3.1.6.1 *Use Case Diagram*

Use case Diagram merupakan gambaran fungsional dari interaksi antara aktor dengan sistem. Sebuah *Diagram use case* menggambarkan keterhubungan antara aktor dengan aksi yang dapat dilakukannya terhadap sistem.

Berikut ini perancangan proses-proses yang terdapat pada *Game* Tradisional Kelereng dapat dilihat pada gambar 3.11.



Gambar 3.11 *Use Case Diagram Game* Tradisional Kelereng

3.1.6.2 Identifikasi Aktor

Identifikasi aktor adalah aktor yang berperan dalam menjalankan sistem, aktor yang berperan dapat dilihat pada tabel 3.4.

Tabel 3. 4 Identifikasi Aktor

No.	Aktor	Deskripsi
1	Pemain	Merupakan aktor yang berperan dalam memainkan *Game* tradisional kelereng.

3.1.6.3 Definisi *Use Case*

Definisi *use case* berfungsi untuk menjelaskan proses yang terdapat pada setiap *use case*. Definisi Use Case dapat dilihiat pada table 3.5.

Tabel 3. 5 Definisi *Use Case*

NO	*Use Case*	Deskripsi
1	Permainan *Level* 1	Proses bermain permainan *level* 1
2	Permainan *Level* 2	Proses bermain permainan *level* 2
3	Permainan *Level* 3	Proses bermain permainan *level* 3
4	Menampilkan Cara Bermain	Proses menampilkan tampilan cara bermain
5	Menampilkan Tentang	Proses menampilkan tampilan tentang

3.1.6.4 Skenario *Use Case*

Skenario setiap bagian pada *use case* menunjukkan proses apa yang terjadi pada setiap bagian di dalam *use case* tersebut, di mana pengguna memberikan perintah pada setiap bagian dan respon apa yang diberikan oleh sistem kepada pengguna setelah pengguna memberikan perintah pada setiap bagian-bagian *use case*. Skenario use case permainan *level* 1 akan dijelaskan pada tabel 3.6.

Tabel 3. 6 Skenario *Use Case* Permainan *Level* 1

Identifikasi	
Nomor	1
Nama	Permainan *Level* 1
Tujuan	Memenangkan permainan pada *level* 1
Deskripsi	Proses bermain permainan *level* 1
Aktor	Pemain
Skenario Utama	
Kondisi awal	Pemain berada di tampilan Menu Utama

Aksi Aktor	Reaksi Sistem
1. Memilih tombol main pada menu utama	
	2. Menampilkan tampilan permainan
	3. Menampilkan objek kelereng utama
	4. Menampilkanobjek kelereng target
	5. Menampilkan objek batu
	6. Menampilkan waktu
	7. Menampilkan point
8. *Touch drag* objek kelereng utama	
	9. Menampilkan garis arah laju
10. Touch objek tangan	
	11. Objek kelereng utama bergerak
Kondisi Akhir	Kelereng utama *collision* dengan kelereng target dan kelereng target keluar semua dari garis batas waktu masih tersedia dan pemain menang atau kelereng target tidak keluar semua dari garis batas waktu habis dan pemain kalah

Skenario *use case* permainan *level* 2 menjelaskan interaksi antara aktor yaitu pemain dengan *use case* permainan *level* 2 yang akan dijelaskan pada tabel 3.7.

Tabel 3. 7 Skenario *Use Case* Permainan *Level* 2

Identifikasi	
Nomor	2
Nama	Permainan *Level* 2
Tujuan	Memenangkan permainan pada *level* 2
Deskripsi	Proses Masuk ketampilan permainan *level* 2
Aktor	Pemain
Skenario Utama	
Kondisi awal	Pemain berada di tampilan permainan *level* 2
Aksi Aktor	**Reaksi Sistem**
	1. Menampilkan tampilan permainan
	2. Menampilkan objek Kelereng utama
	3. Menampilkan objek kelereng target
	4. Menampilkan objek batu
	5. Menampilkan waktu
	6. Menampilkan point
	7. Menampilkan objek semut hitam
8. *Touch drag* objek kelereng utama	
	9. Menampilkan garis arah gerak kelereng

	utama
10.*Touch* objek tangan	
	11.Objek kelereng utama bergerak
Kondisi Akhir	Kelereng utama *collision* dengan kelereng target dan kelereng target keluar semua dari garis batas waktu masih tersedia dan pemain menang atau kelereng target tidak keluar semua dari garis batas waktu habis dan pemain kalah. Kelereng utama *collision* dengan objek semut point berkurang, point habis dan pemain kalah

Skenario *use case* permainan *level* 3 menjelaskan interaksi antara actor yaitu pemain dengan *use case* permainan *level* 3 yang akan dijelaskan pada table 3.8.

Tabel 3. 8 Skenario Use Case Permainan *Level* 3

Identifikasi	
Nomor	3
Nama	Permainan *Level* 3
Tujuan	Memenangkan permainan pada *level* 3
Deskripsi	Proses Masuk ketampilan permainan *level* 3
Aktor	Pemain
Skenario Utama	
Kondisi awal	Pemain berada di tampilan permainan *level* 3
Aksi Aktor	**Reaksi Sistem**
	1. Menampilkan tampilan permainan
	2. Menampilkan objek kelereng utama
	3. Menampilkan objek kelereng target
	4. Menampilkan objek batu
	5. Menampilkan waktu
	6. Menampilkan point
	7. Menampilkan objek semut hitam
	8. Menampilkan objek semut merah

9. Pemain menekan dan mengarahkan *objek* kelereng utama	
	10.Menampilkan garis arah gerak kelereng utama
11.Pemain menekan objek tangan	
	12.Objek kelereng utama bergerak
Kondisi Akhir	Kelereng target *collision* dengan kelereng target dan kelereng target keluar semua dari garis batas waktu masih tersedia dan pemain menang atau kelereng target tidak keluar semua dari garis batas waktu habis dan pemain kalah. Kelereng utama *collision* dengan objek semut point berkurang, point habis dan pemain kalah

Skenario *use case* menampilkan cara bermain menjelaskan interaksi antara aktor yaitu pemain dengan *use case* menampilkan cara bermain yang akan dijelaskan pada tabel 3.9.

Tabel 3. 9 Skenario *Use Case* Menampilkan Cara Bermain

Identifikasi	
Nomor	4
Nama	Menampilkan Cara Bermain
Tujuan	Masuk ke tampilan cara bermain
Deskripsi	Proses untuk menampilkan cara bermain
Aktor	Pemain
Skenario Utama	
Kondisi awal	Pemain berada di menu utama permainan *Game tradisional kelereng*
Aksi Aktor	**Reaksi Sistem**
1. Menekan tombol cara bermain	
	2. Masuk ke tampilan cara bermain
Kondisi Akhir	Masuk ke tampilan cara bermain

Skenario *use case* menampilkan tentang menjelaskan interaksi antara aktor yaitu pemain dengan *use case* menampilkan tentang yang akan dijelaskan pada tabel 3.10.

Tabel 3. 10 Skenario *Use Case* Menampilkan Tentang

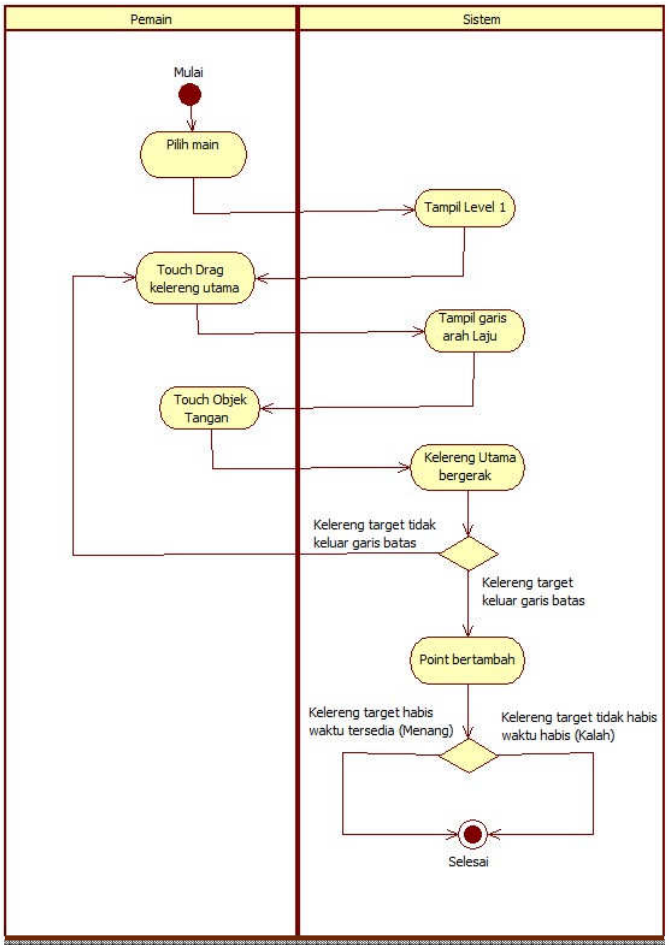
Identifikasi	
Nomor	5
Nama	Menampilkan Tentang
Tujuan	Masuk ke tampilan tentang
Deskripsi	Proses untuk menampilkan tentang
Aktor	Pemain
Skenario Utama	
Kondisi awal	Pemain berada di menu utama permainan *Game tradisional kelereng*
Aksi Aktor	**Reaksi Sistem**
1. Menekan tombol tentang	
	2. Masuk ke tampilan tentang pembuat *Game*
Kondisi Akhir	Masuk ke tampilan tentang pembuat *Game*

3.1.6.5 *Activity Diagram*

Activity Diagram menggambarkan alur aktivitas di dalam sebuah proses (use case). *Activity Diagram* menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum. Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih.

1. *Activity Diagram* Permainan *level* 1

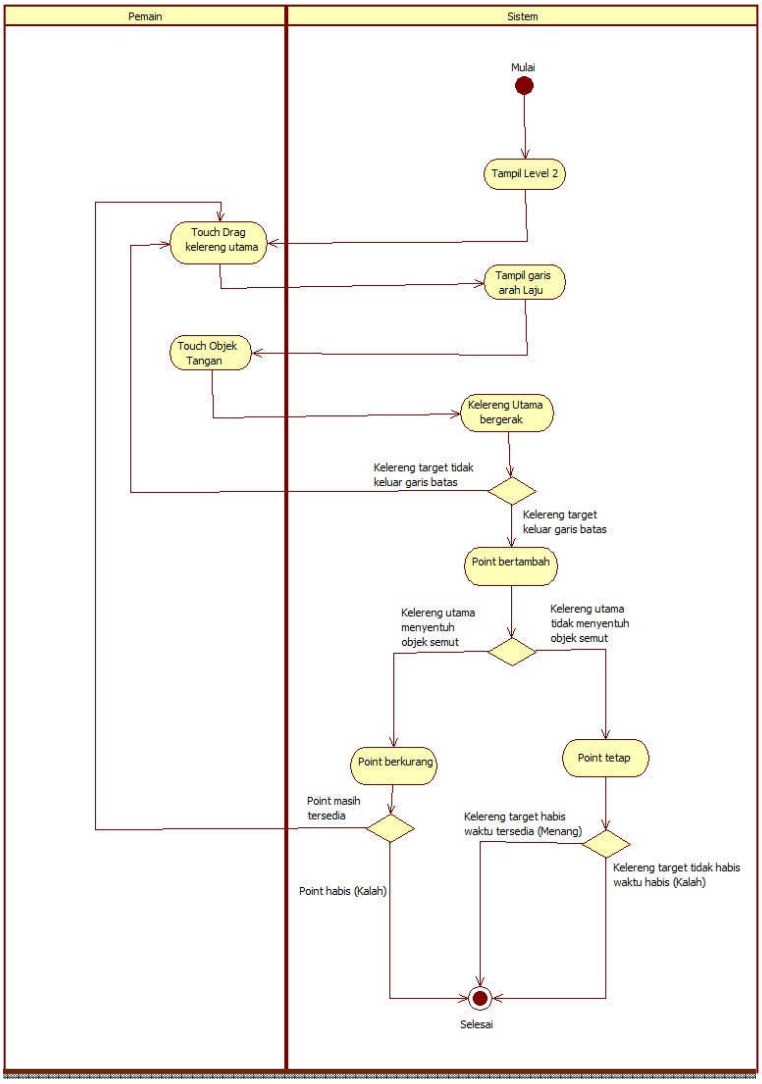
Aktivitas ini terjadi saat pemain memainkan permainan pada *level* 1. *Diagram activity* Permainan *level* 1 dapat dilihat pada gambar 3.12.



Gambar 3.12 *Activity Diagram* Pada Proses Permainan *Level* 1

2. *Activity Diagram* Permainan *Level* 2

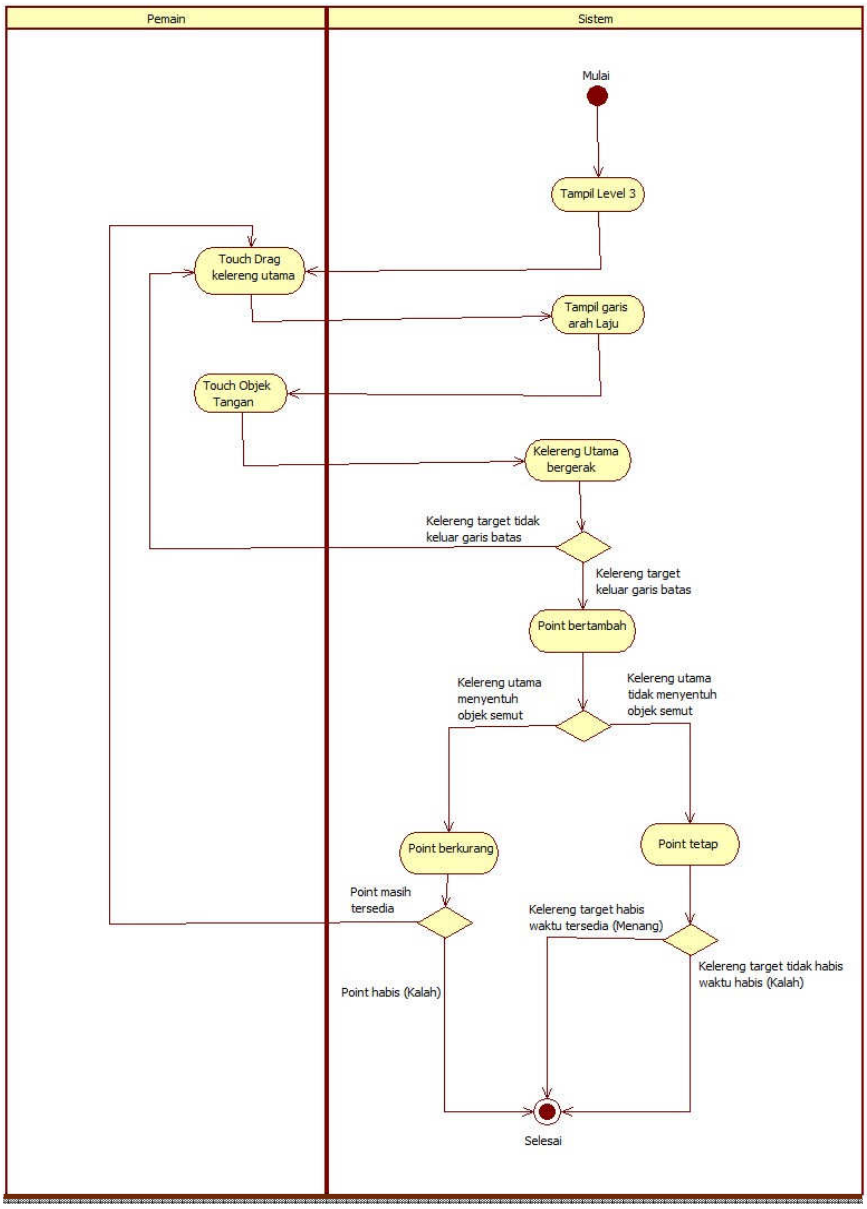
Aktivitas ini terjadi saat pemain memainkan permainan pada *level 2. Diagram activity* Permainan *level* 2 dapat dilihat pada gambar 3.13.



Gambar 3.13 *Activity Diagram* Pada Proses Permainan *Level* 2

3. *Activity Diagram* Permainan *Level* 3

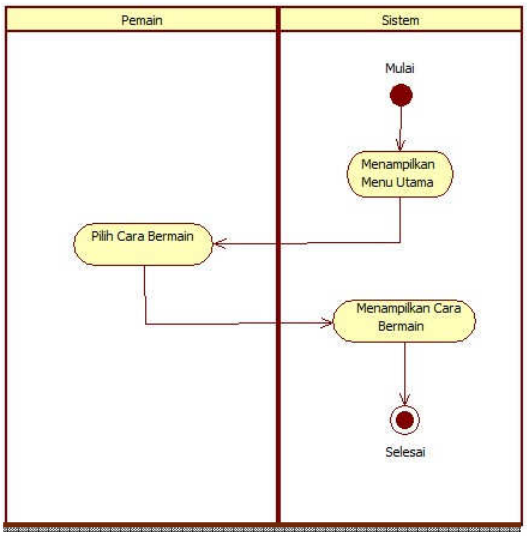
Aktivitas ini terjadi saat pemain memainkan permainan pada *level 3. Diagram activity* Permainan *level* 3 dapat dilihat pada gambar 3.14.



Gambar 3.14 *Activity Diagram* Pada Proses Permainan *Level* 3

4. *Activity Diagram* Menampilkan Cara Bermain

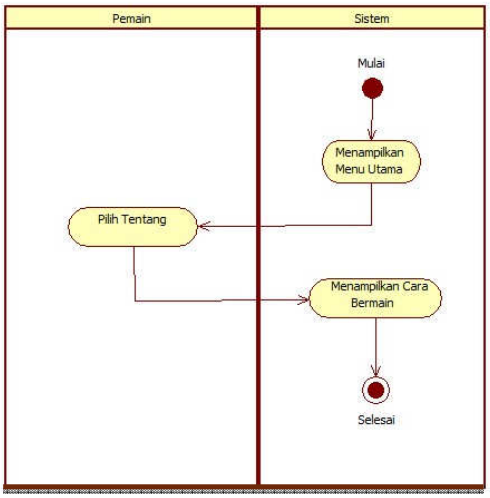
Aktivitas ini terjadi saat pemain menekan tombol cara bermain pada tampilan menu utama. *Diagram activity* menampilkan cara bermain dapat dilihat pada gambar 3.15.



Gambar 3.15 *Activity Diagram* Pada Proses Menampilkan Cara Bermain

5. *Activity Diagram* Menampilkan Tentang

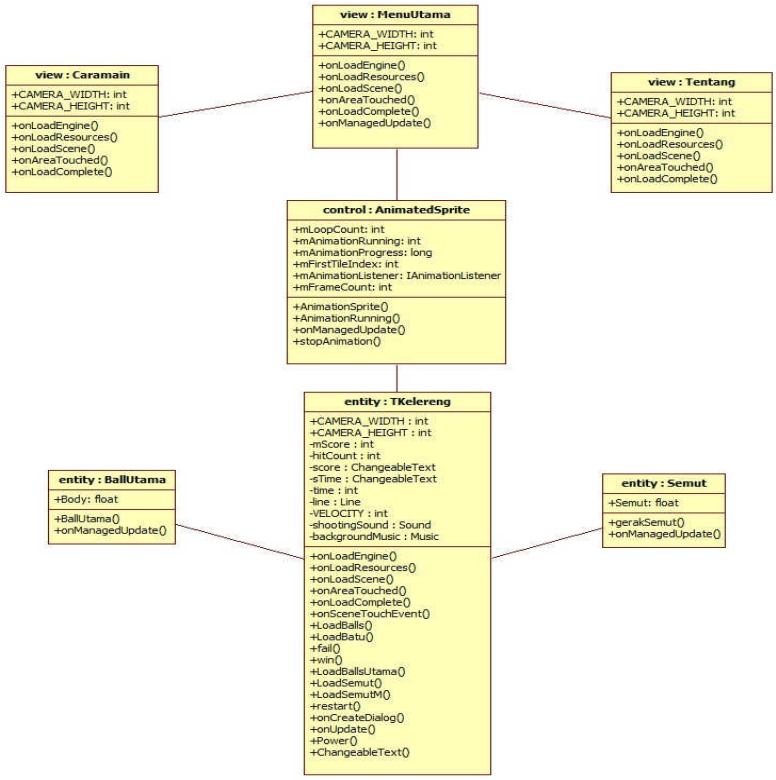
Aktivitas ini terjadi saat pemain menekan tombol tentang pada tampilan menu. *Diagram activity* menampilkan tentang dapat dilihat pada gambar 3.16.



Gambar 3.16 *Activity Diagram* Pada Proses Menampilkan Tentang

3.1.6.6 *Class Diagram*

Class Diagram digunakan untuk menampilkan kelas - kelas dan paket-paket di dalam sistem. *Class Diagram* memberikan gambaran sistem secara statis dan relasi antar mereka. Beberapa *Diagram* akan menampilkan subset dari kelas-kelas dan relasinya. *Class Diagram Game* tradisional kelereng dapat dilihat pada gambar 3.17.



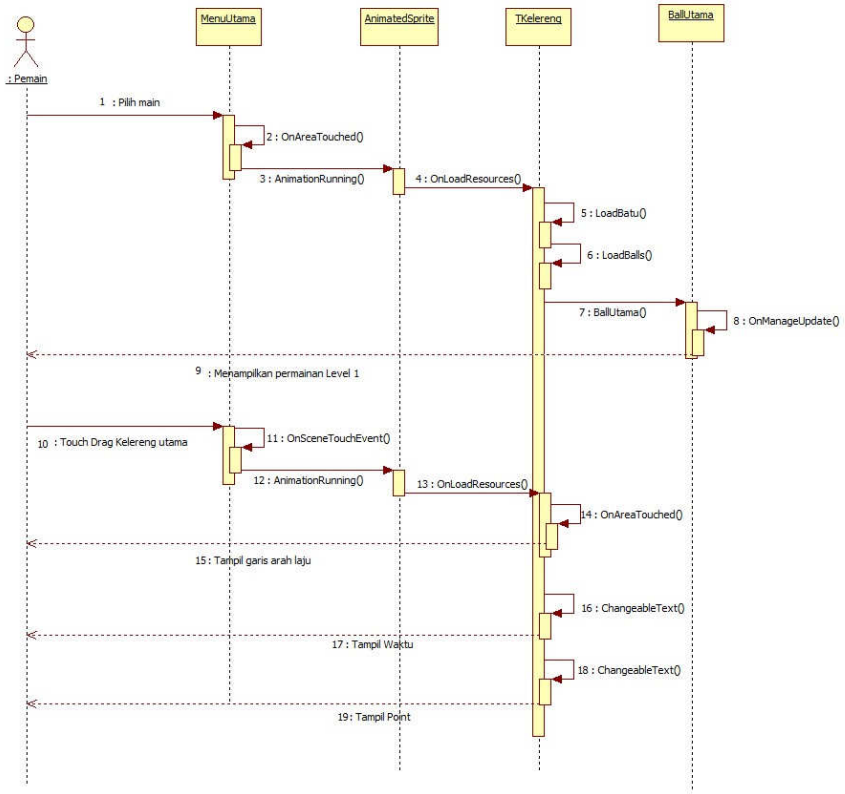
Gambar 3.17 *Class Diagram Game* Tradisional Kelereng

3.1.6.7 Sequence *Diagram*

Sequence Diagram menggambarkan aksi objek atau aktor pada *usecase* dengan mendeskripsikan waktu hidup objek dan pesan respon yang dikirimkan dan diterima antar objek.

1. *Sequence Diagram* Permainan *Level* 1

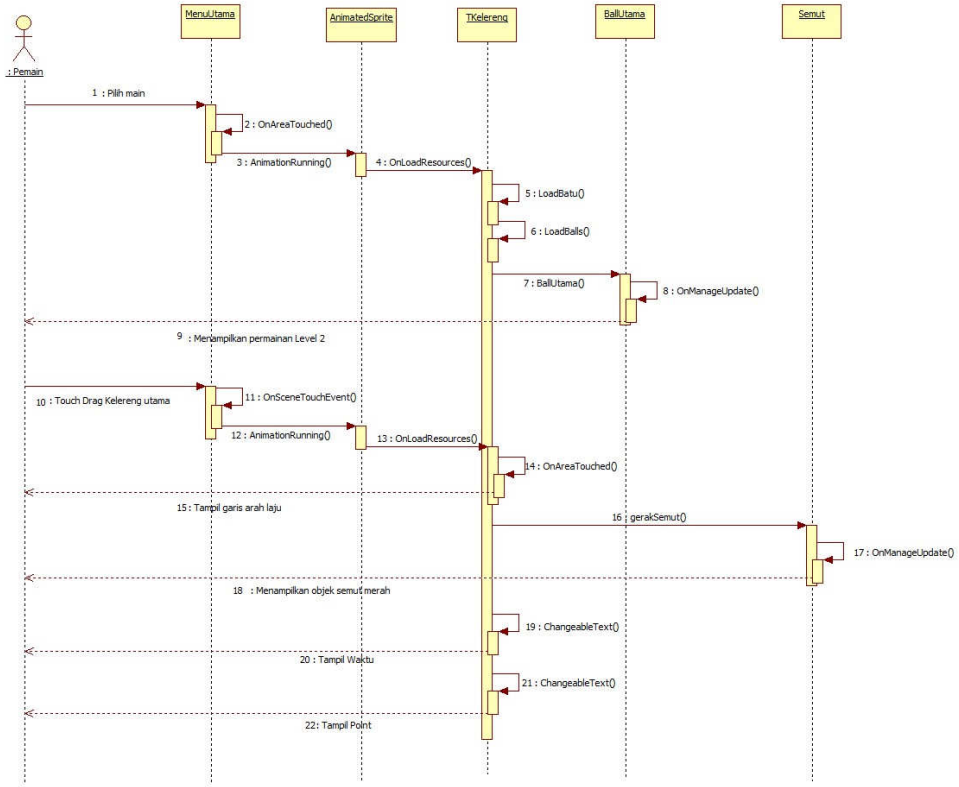
   Sequence Diagram permainan *level* 1 dapat dilihat pada gambar 3.18.



Gambar 3.18 *Sequence Diagram* Permainan *Level* 1

2. Sequence *Diagram* Permainan *Level* 2

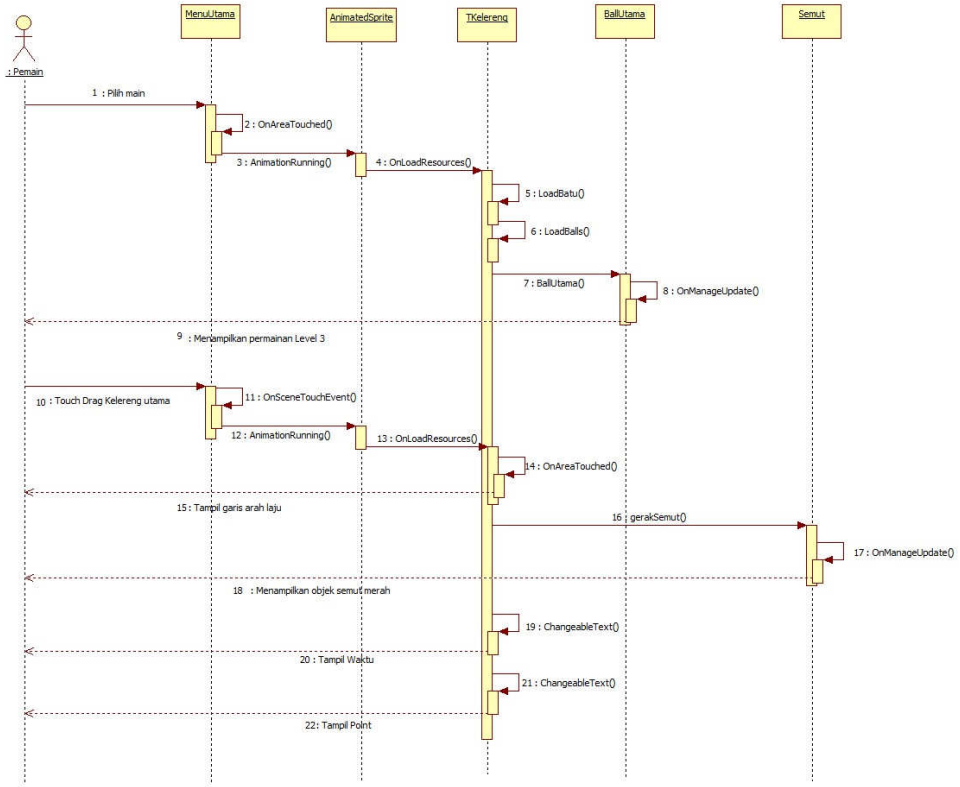
Sequence Diagram permainan *level* 2 dapat dilihat pada gambar 3.19.



Gambar 3.19 *Sequence Diagram* Permainan *Level* 2

3. Sequence *Diagram* Permainan *Level* 3

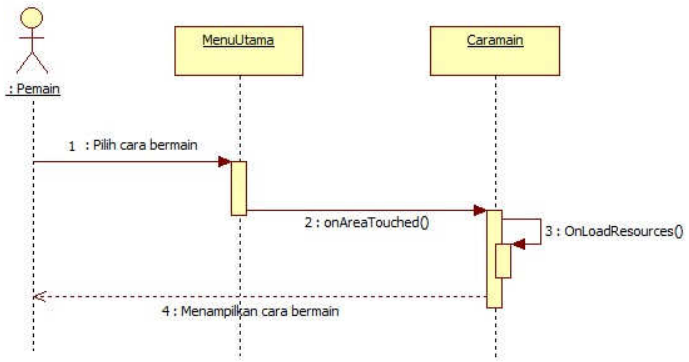
Sequence Diagram permainan *level* 3 dapat dilihat pada gambar 3.20.



Gambar 3.20 *Sequence Diagram* Permainan *Level* 3

4. Sequence *Diagram* Menampilkan Cara Bermain

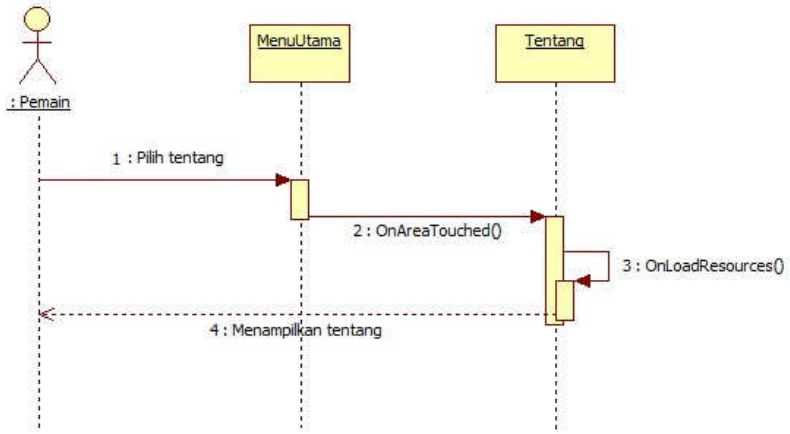
   Sequence Diagram menampilkan cara bermain dapat dilihat pada gambar 3.21.



Gambar 3.21 *Sequence Diagram* Menampilkan Cara Bermain

5. Sequence *Diagram* Menampilkan Tentang

   Sequence Diagram menampilkan tentang dapat dilihat pada gambar 3.22



Gambar 3.22 *Sequence Diagram* Menampilkan Tentang

3.2 Perancangan Sistem

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

3.2.1 Perancangan Komponen *Game*

Perancangan Komponen *Game* ini adalah untuk mengetahui komponen-komponen yang dibutuhkan dalam merancang *Game* yang akan dibangun seperti karakter, storyboard, perancangan menu, perancangan antar muka dan perancangan method.

3.2.1.1 Perancangan Objek *Game*

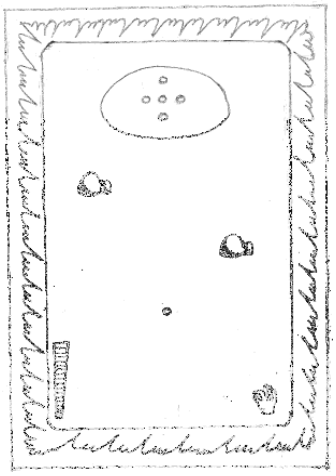
Objek dalam sebuah *Game* merupakan komponen terpenting sebagai pelaku atau objek di dalam sebuah *Game*. Objek yang terdapat dalam *Game* Tradisional Kelereng ini dapat dilihat pada table 3.11.

Tabel 3.11 Pengenalan Objek

No	Nama	Gambar	Keterangan
1	Kelereng Target		- Merupakan kelereng Target - Diletakan didalam lingkaran/kalang
2	Kelereng Utama		Kelereng utama yang dipakai pemain
3	Batu		Merupakan hambatan yang statis
4	Semut		Merupakan hambatan yang dinamis

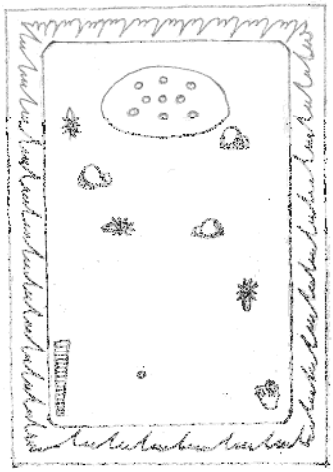
3.2.1.2 Story Board

Story board adalah sketsa gambar yang disusun berurutan sesuai dengan naskah. Gambar 3.23 merupakan *story board level 1*dari *Game* tradisional kelereng.



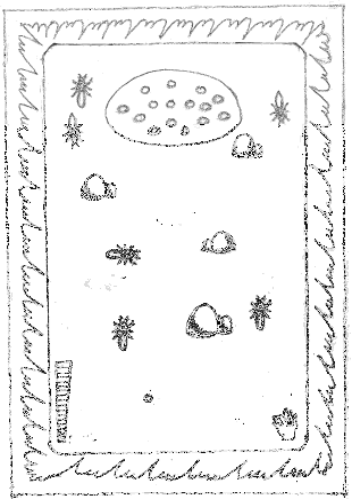
Gambar 3.23 ***Story Board level 1***

Pada *level* 1 ini akan terdapat 5 kelereng dikalang dan 2 buah batu sebagai hambatan.



Gambar 3.24 ***Story Board level*** **2**

Gambar 3.24 merupakan *story board level* 2, *pada level 2* ini akan terdapat 10 kelereng dikalang dan hambatannya ada 3 batu dan 2 semut hitam dan 1 semut merah.



Gambar 3.25 *Story Board level* 3

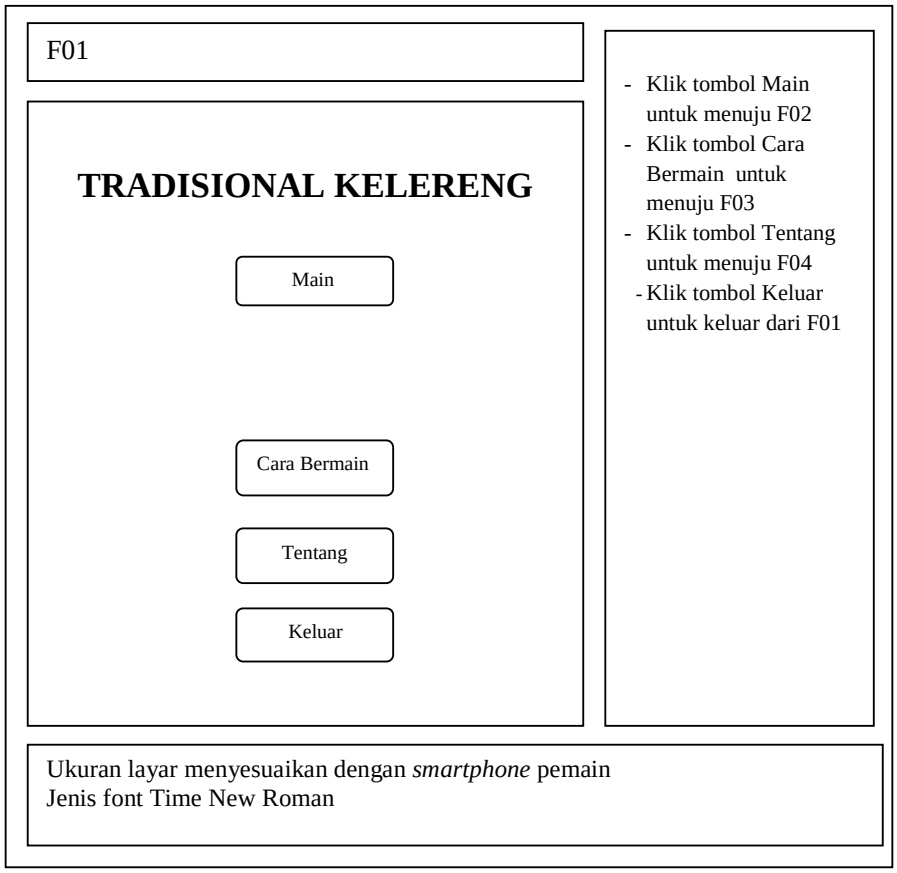
Gambar 3.25 merupakan *story board level* 3, *pada level 3* ini akan terdapat 10 kelereng dikalang dan hambatannya ada 4 batu, 3 semut hitam dan 1 semut merah.

3.2.2 Perancangan Antar Muka

Perancangan antarmuka merupakan sebuah penggambaran, perencanaan, dan pembuatan sketsa atau pengaturan dari beberapa elemen yang terpisah ke dalam satu kesatuan yang utuh dan berfungsi. Adapun perancangan antarmuka *Game* tradisional kelereng adalah sebagai berikut :

1. Perancanagn Antar Muka Menu Utama

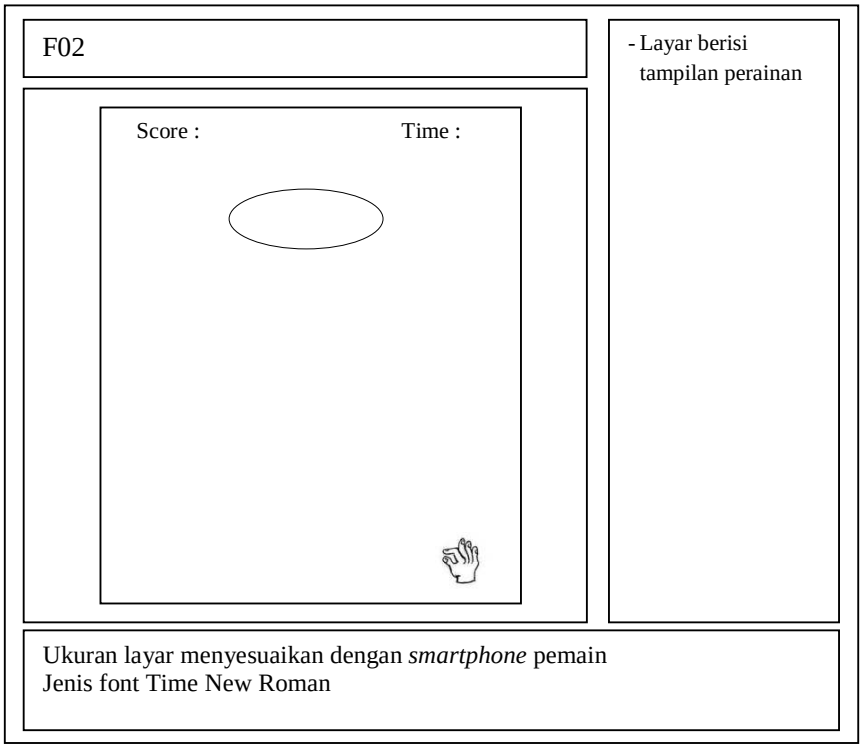
Form menu utama adalah *form* yang digunakan sebagai tampilan awal ketika pemain membuka aplikasi. Tampilan *form* menu utama dapat dilihat pada gambar 3.26.



Gambar 3.26 Perancangan Antar Muka Menu Utama

2. Perancanagn Antar Muka Permainan

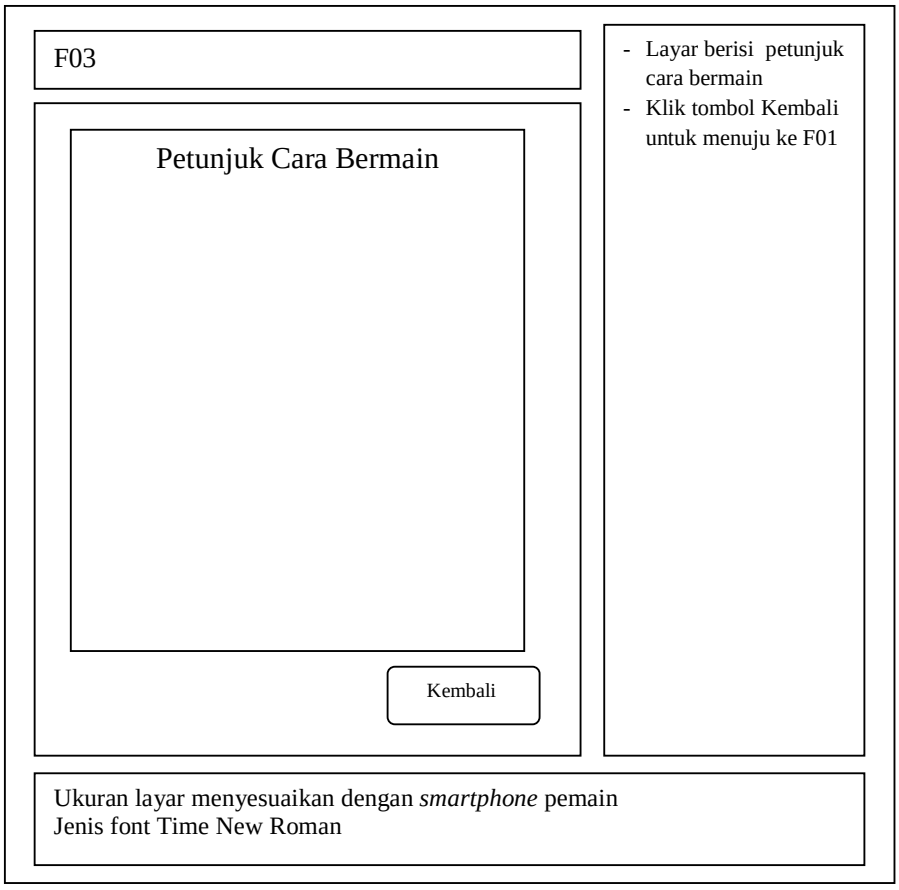
Form permainan adalah *form* yang digunakan sebagai tampilan area bermain dalam *Game* ini. Tampilan *form* permainan dapat dilihat pada gambar 3.27.



Gambar 3.27 Perancangan Antar Muka Permainan

3. Perancangan Antar Muka Cara Bermain

Form Cara Bermain adalah *form* yang digunakan sebagai tampilan ketika pemain ingin mengetahui petunjuk bermain *Game* ini. Tampilan *form* cara bermain dapat dilihat pada gambar 3.28.



Gambar 3.28 Perancangan Antar Muka Cara Bermain

4. Perancangan antar Muka Tentang

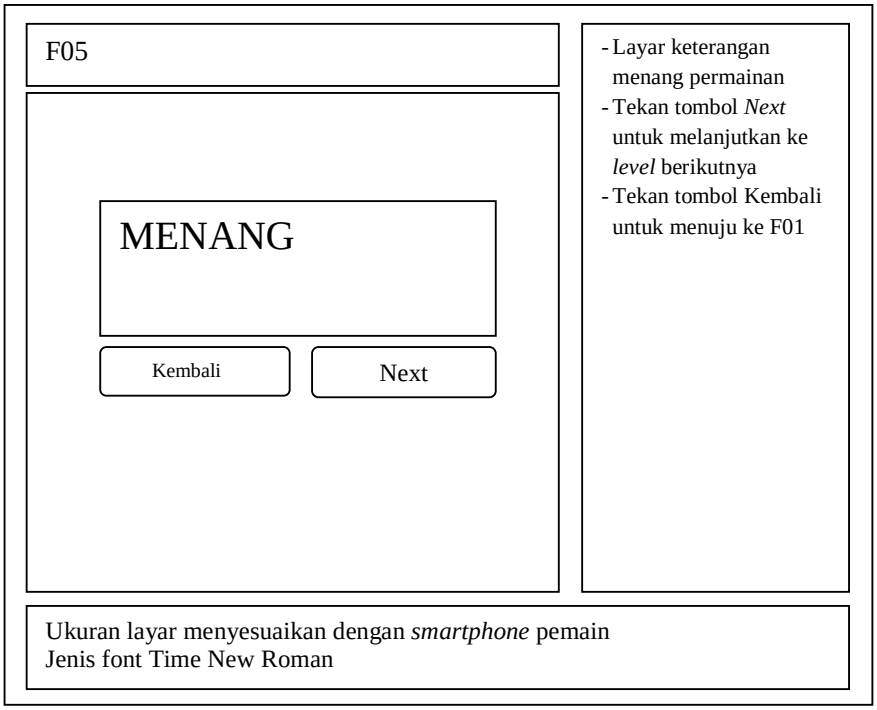
Form tentang adalah *form* yang digunakan sebagai tampilan ketika pemain ingin mengetahui pembuat dari *Game* ini. Tampilan *form* tentang dapat dilihat pada gambar 3.29.



Gambar 3.29 Perancangan Antar Muka Tentang

5. Perancangan antar Menang

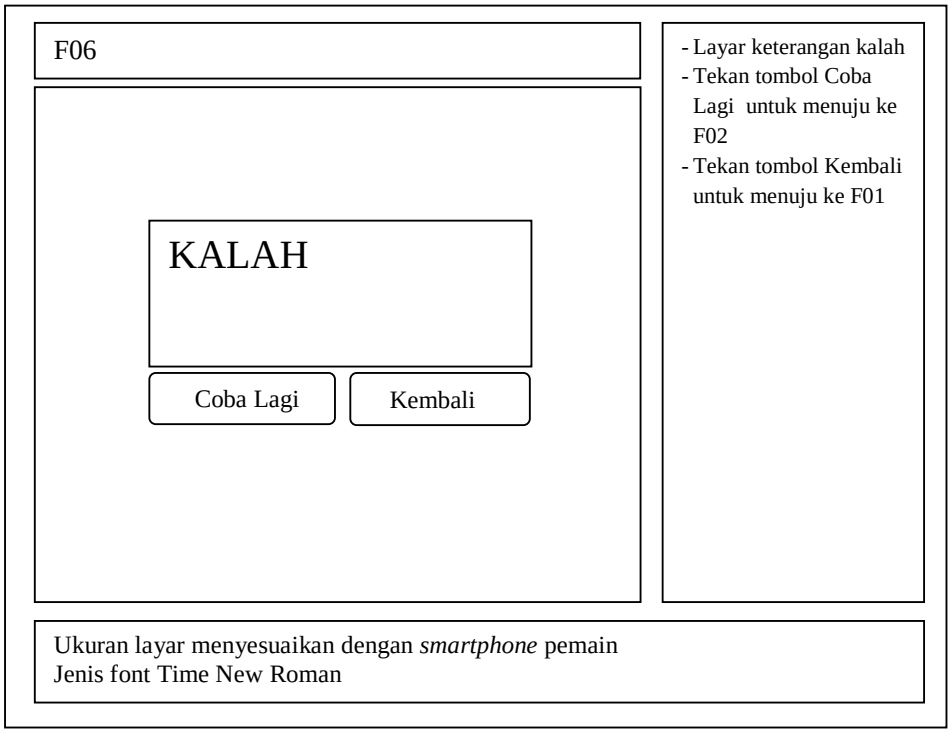
Form menang adalah *form* yang digunakan sebagai tampilan menang bermain kelereng dan dapat melanjutkan ke *level* berikutnya dalam *Game* ini. Tampilan *form* menang dapat dilihat pada gambar 3.30.



Gambar 3.30 Perancangan Antar Muka Tentang

6. Perancangan antar Muka Kalah

Form kalah adalah *form* yang digunakan sebagai tampilan kalah bermain kelereng dan tidak dapat melanjutkan ke stage berikutnya dalam *Game* ini. Tampilan *form* kalah dapat dilihat pada gambar 3.31.



Gambar 3.31 Perancangan Antar Muka Kalah

3.2.3 Perancangan *Method*

Perancangan *method* menjelaskan penjabaran alur proses-proses yang ada pada sistem yang dibangun menggunakan *Diagram* flowchart. Adapun perancangan method untuk aplikasi yang akan dibangun adalah sebagai berikut.

3.2.3.1 Perancangan *Method win()*

Method untuk memenangkan permainan terdapat pada *class* TKelereng, tiap *level* memiliki point maksimum yang berbeda, *level* 1 point maksimumnyanya adalah 10, *level* 2 point maksimumnyanya adalah 20 dan *level* 3 point maksimumnyanya adalah adalah 20. *Method Win()* ini akan deksekusi ketika seluruh kelereng target yang ada keluar dari batas garis/kalang. Flowchart yang di contohkan adalah perhitungan nilai *level* 1 dapat dilihat di gambar 3.32.



Gambar 3.32 ***Flowchart Method win()***

3.2.3.2 Perancangan *Method fail()*

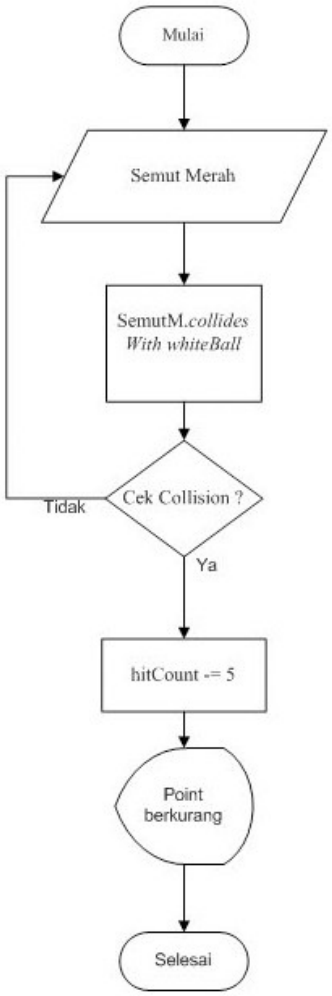
Method ini terdapat pada *class* TKelereng, *method* ini dieksekusi ketika pemain kalah. *Method fail()* ini bisa dilihat pada gambar 3.33.



Gambar 3.33 *Flowchart Method fail()*

3.2.3.3 Perancangan *Method onUpdate()*

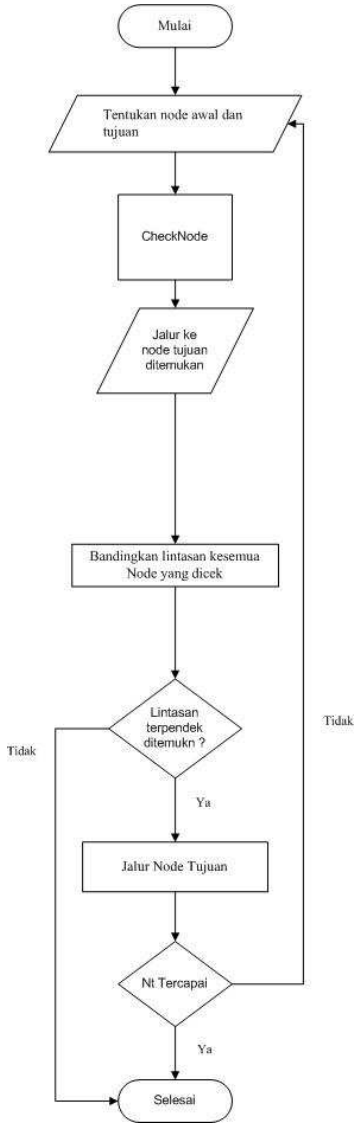
Method ini terdapat pada *class* TKelereng, *method ini* dieksekusi apabila objek kelereng utama menyentuh objek semut merah. *Method* ini bisa dilihat pada gambar 3.34.



Gambar 3.34 *Flowchart Method onUpdate()*

3.2.3.4 Perancangan *Method* gerakSemut*()*

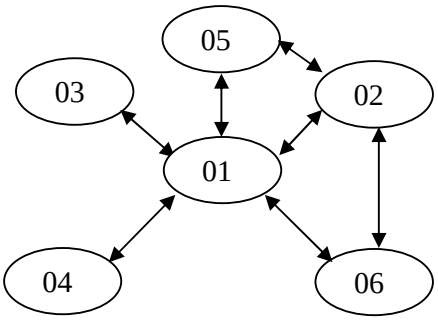
Method ini terdapat pada *class* gerakSemut, *method ini* dieksekusi ketika objek kelereng target bergerak , *method* ini berfungsi untuk menentukan pergerakan objek semut. *Method* ini bisa dilihat pada gambar 3.36.



Gambar 3.35 *Flowchart Method* geraksemut*()*

3.2.4 Jaringan Semantik

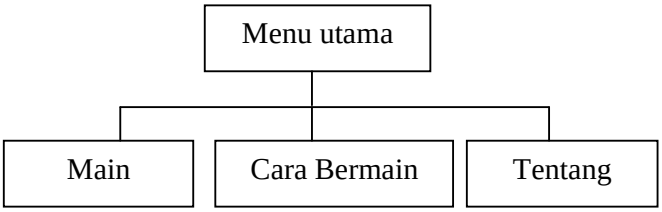
Perancangan jaringan semantik menggambarkan alur informasi dari setiap antarmuka aplikasi yang dijelaskan pada gambar 3.36.



Gambar 3.36 Jaringan Semantik

3.2.5 Struktur Menu

Struktur menu merupakan fitur-fitur yang disediakan pada system yang dirancang dan untuk mempermudah dalam penggunaan system nantinya. Perancangan struktur menu dapat dilihat pada gambar 3.38.



Gambar 3.37 Struktur Menu *Game* Tradisional Kelereng