BAB 3
ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Sistem

Analisis sistem merupakan penguraian dari suatu sistem informasi yang utuh ke dalam bagian-bagian komponennnya yang dimaksudkan untuk mengidentifkasi dan mengevaluasi segala permasalahan dan hambatan yang terjadi serta kebutuhan yang diharapkan dapat menjadi acuan untuk menjadi perbaikan.

3.1.1 Analisis Masalah

Berdasarkan analisis sistem yang dilakukan dengan mengevaluasi terhadap pengetahuan anak-anak dan remaja tentang makanan tradisional khas Indonesia khususnya di pulau Jawa maka masalah yang dihadapi oleh anak-anak dan remaja di Indonesia khususnya anak-anak yang usianya 10 tahun ke atas dalam melakukan pembelajaran tentang makanan tradisional adalah sebagai berikut :

1. Pentingnya pengetahuan tentang makanan tradisional Indonesia karena dengan mengenal dan mengetahui makanan-makanan tersebut akan menambah wawasan dan melestarikan kebudayaan di Indonesia.
2. Pengenalan secara teoritis kurang optimal untuk mengenalkan makanan-makanan tersebut kepada anak dan remaja, sehingga dibutuhkan sebuah aplikasi *game* agar pengenalan terlihat menarik.
3. Perlunya pengenalan makanan tradisional terhadap anak-anak dan remaja di Indonesia supaya bisa lebih mengenal tentang kebudayaan yang ada di Indonesia.

3.1.2 Analisis*Game* Sejenis

Pada analisis *game* sejenis, akan dilakukan observasi terhadap *game* yang memiliki kesamaan *gameplay* dan *genre* dengan *game* yang akan dibangu. Tujuan dari analisis ini adalah untuk mendapatkan konsep permainan dan perbandingan *game* yang telah ada. *Game* yang akan dibandingkan adalah :

1. *Game* 2D Kabayan *Saving Our Cultures*
   a. Pengenalan

      Game kabayan *Saving Our Cultures* menceritakan perjalanan seorang pemuda dari suku Sunda yaitu Kabayan yang harus bertualang untuk menyelamatkan alat musik tradisional yaitu angklung yang telah dicuri oleh seorang penjahat dari Negara Misterius yang bernama Juragan Asep. Kekurangan dari *game* ini adalah, sisi kebudayaan yang diangkat belum maksimal cuman sebatas alat musik calung dan golok saja, *gameplay* yang terlalu mudah sehingga membuat pemain akan cepat bosan, sedangkan kelebihannya adalah desain yang cukup bagus.



Gambar 3.1.*Game* 2D Kabayan *Saving Our Cultures*

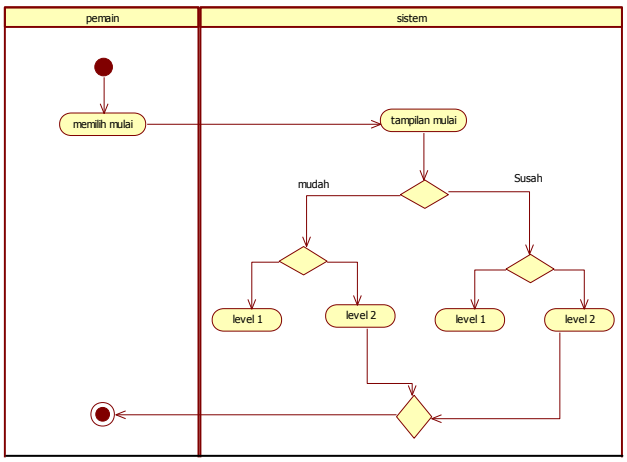
Pada gambar 3.2 bisa dilihat salah satu tampilkan level 1 pada *game* Kabayan *Saving Our Cultures*.



Gambar 3.2.Tampilan Level 1-1

b. Gameplay

Tugas utama pemain dalam memainkan Aplikasi ini yaitu melalui level-level dan juga sub level, dimana setiap bagian level menyediakan petualangan dengan menampilkan beberapa karakter level tersebut maupun karakter musuh yang harus dilawan, sehingga pemain memperoleh *score* (nilai). Adapun alur sistem dari permainan *game* kabayan SOC dapat di gambarkan melalui *activity diagram* berikut ini :



Gambar 3.3. Alur permainan pada *game* SOC

c. Tingkat kesulitan

Pemain dapat memulai permainan dengan memilih pilihan mulai baru, dimana nantinya pemain memasuki sub pilihan yaitu memilih level dengan karakter permainan "Mudah" atau karakter permainan "Susah". Setelah memilih salah satu pilihan, pemain kembali diantarkan pada sub pilihan, yaitu pilihan untuk memilih level, dengan tujuan apakah pemain ingin bermain dari level awal atau pada hal ini level 1, sedangkan pilihan lainnya adalah pemain langsung memulai untuk bermain pada level berikutnya atau level 2.

d. Kekurangan

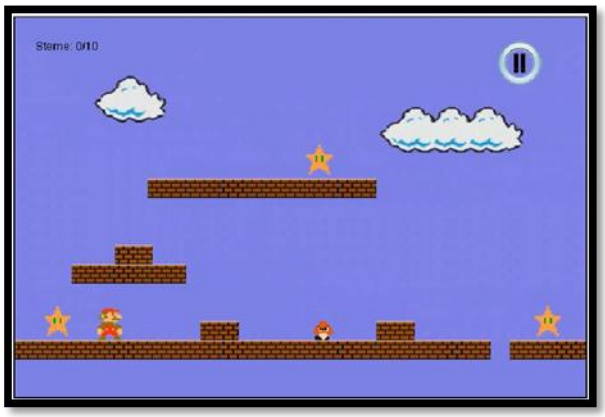
Sisi kebudayaan belum terangkat secara maksimal, masih ada beberapa gameplay yang harus ditambah, tidak memiliki sistem multiplayer,

masih banyak fitur-fitur yang perlu ditambahkan dalam aplikasi game2D Kabayan Saving Our Cultures.

2. Mario klon

a. Pengenalan

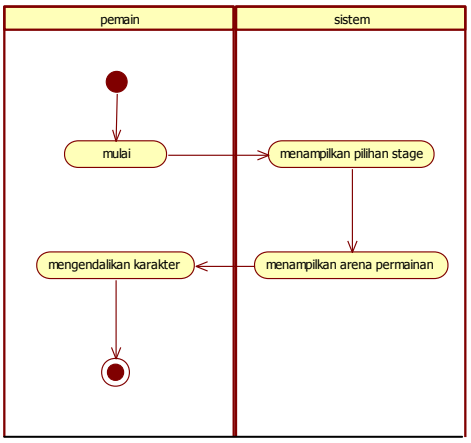
Game Mario klon adalah salah satu *game platform sidescrolling* 2D, *game* ini di adaptasi dari *game* Mario



Gambar 3.4.*Game* Mario Klon

b. Gameplay

Mario menjelajahi suatu tempat untuk mengumpulkan bintang agar mendapat nilai, dan harus mengalahkan musuh Mario yaitu para jamur dan kura-kura. Terdapat 3 pilihan stage denga tingkat kesulitan yang berbeda. Adapun alur sistem dari permainan *game* Mario Klon dapat di gambarkan melalui *activity diagram* berikut ini :



Gambar 3.5. Alur permainan pada *game* Mario Klon

c. Kekurangan

Kekurangan *game* Mario klon ini adalah, *gameplay* yang sangat monoton sehingga sangat tidak menarik, level yang sedikit, sehingga akan mudah untuk di selesaikan.

3. Jsonic

a. Pengenalan

Game Jsonic adalah salah satu *game platform sidescrolling* 2D, *game* ini di adaptasi dari *game* Sonic.



Gambar 3.6.*Game* Jsonic

b. Kekurangan

Kekurangn *Game* Jsonic ini adalah, *gameplay* yang sangat monoton sehingga sangat tidak menarik, tidak memiliki level atau *stage*, karena hanya terdapat satu *stage*.

3.1.3 Analisis *Game* Petualangan Kabayan Menyelamatkan Makanan Tradisional

3.1.3.1 Pengenalan Aplikasi

Game yang akan dibuat berjudul Petualangan Kabayan menyelamatkan makanan tradisional yang *bergenre Adventure Game*. *Game* ini dibangun dengan mengaplikasikan teknologi sebagai jembatan untuk mengenalkan atau menyampaikan pembelajaran khususnya pada anak-anak usia di bawah 10 tahun. *Game* ini dibangun dengan grafis 2D (2 dimensi).*Game* ini menerapkan sistem *single player* dengan data-data dibagikan kedalam objek, karena menggunakan pemrograman berorientasi objek. Berikut ini beberapa kelebihan yang dimiliki yaitu :

1. Memberikan pengenalan sekaligus pembelajaran tentang makanan tradisional pulau Jawa.
2. Ber*gendre adventure game* (*Game* Petualangan) yang lebih interaktif.
3. Aplikasi *game* ini lebih bersifat bermain sambil belajar.

3.1.3.2 Storyline

Game Petualangan Kabayan menyelamatkan makanan tradisional ini ber*gendreAdventure game*. *Game* ini bercerita tentang seorang pemuda yang bernama Kabayan yang mendengar kabar bahwa makanan tradisional dipulau Jawa telah dicuri oleh perampok bernama Rewog dan juga komplotannya. Kabayanpun melakukan perjalanan ke seluruh pulau Jawa untuk menemukan dan merebut kembali makanan tradisional dan mengalahkan Rewog dan komplotannya.

3.1.3.3 Analisis Tingkat Kesulitan

Dalam *game* Petualangan Kabayan Menyelamatkan Makanan Tradisional, terdapat 3 level atau *stage* yang masing-masing level atau *stage* dibedakan berdasarkan alur cerita, tingkat kesulitan rintangan yang harus dihadapi, jumlah dan jenis *enemy* (musuh) yang bertambah. Di setiap peningkatan level atau *stage*, pemainakan dihadapkan pada *enemy* (musuh) dengan tingkat kesulitan yang akan bertambah, mulai dari kecepatan dan jumlah *enemy* (musuh) tersebut.

Di *Stage* pertama, pemain dalam perjalanan menuju daerah pulau Jawa Barat. Di perjalanan, kabayan harus menghadapi bebegig (orang – orangan sawah) yang terdapat di level mudah, pada level sedang musuh bertambah, yaitu guguk dan kalong, dan pada level sulit musuh bertambah yaitu bagong. pemain harus melewati semua musuh dan menuju ke garis akhir untuk dapat melanjutkan permainan ke *stage* selanjutnya.

Di *Stage* kedua, pemain berada di daerah Jawa Tengah. Di *stage* ini pemain harus menghadapi bebegig, dan guguk pada level mudah, pada level sedang musuh bertambah yaitu kalong, dan pada level sulit musuh bertambah yaitu bagong. pemian harus melewati semua musuh dan menuju ke garis akhir untuk dapat melanjutkan permainan ke *stage* selanjutnya.

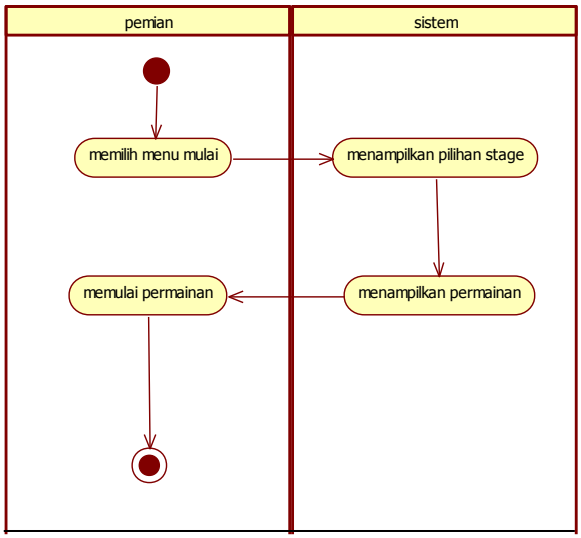
Di *Stage* ketiga, Akhirnya pemain telah sampai di daerah Jawa Timur. Di *stage* ini pemain harus menghadapi bebegig, dan guguk pada level mudah, pada level sedang musuh bertambah yaitu kalong, pemian harus melewati semua musuh dan menuju ke garis akhir untuk dapat melanjutkan permainan ke *stage* selanjutnya, dan pada level sulit musuh bertambah yaitu bagong dan pemian harus melewati semua musuh dan menuju ke garis akhir untuk dapat melanjutkan permainan word boss untuk mengalahkan Bos Utama yang mencuri makanan yaitu Rewog untuk memenangkan *game* ini.

3.1.3.4 Gameplay

Pada permainan ini, tugas utama pemain adalah *actor* akan melakukan perjalanan untuk merebut dan menemukan makanan tradisional di setiap *stage*,

terdapat 3 *stage* di setiap stage akan terdapat tiga pilihan tingkat kesulitan yaitu mudah, sedang dan sulit. Dalam setiap perjalanan akan terdapat beberapa musuh yang harus di kalahkan dan menemukan beberapa makanan tradisional yang tersembunyi untuk mendapatkan *score.*

Pemain dapat mengawali permainan dengan memilih pilihan menu dari ketiga *stage*.. Karakter akan melakukan perjalanan dan melawan musuh yang menghadang, karakter akan menyerang musuh dengan menembak. Pada *stage* ke 2 musuh yang menghadang akan semakin banyak sehingga permainan menjadi semakin sulit.Alur sistem dari permainan *game* Petualangan Kabayan dapat di gambarkan melalui *activity diagram* berikut ini :



Gambar 3.7. Alur permainan pada *game* Petualangan Kabayan

3.1.3.5 Scoring

Berikut cara memperoleh nilai atau *score* dari *game* Petualangan Kabayan Menyelamatkan makanan tradisional yaitu :

1. Untuk mendapatkan nilai karakter kabayan dapat mengambil gambar makanan yang ada disetiap perjalanan.
2. Pada *stage* pertama, terdapat beberapa makanan tradisional khas Jawa Barat yang terletak di tempat-tempat tertentu secara acak. Setiap mendapat makanan akan mendapat *score* 1.

3. Pada *stage* 2, terdapat makanan tradisional khas Jawa Tengah. *Score* yang didapat tetap sama yaitu 1.
4. Pada *stage* 3, terdapat makanan tradisional khas Jawa Timur yang tersembunyi diberbagai tempat dalam perjalanan. *Score* nya masih tetap sama yaitu 1*point*.
5. Setelah menyelesaikan *stage* 3 kemudian, akan mengalahkan bos terakhir untuk menyelesaikan permainan

3.1.3.6 Elemen *Game*

Elemen permianan yang terdapat dalam g*ame* Petualangan Kabayan Menyelamatkan Makanan Tradisional diantaranya adalah sebagai berikut :

1. Arena permainan

   Pada permainan g*ame* Petualangan Kabayan Menyelamatkan Makanan Tradisional arena permainannya adalah dari pedesaan, pegunungan sampai ke kota. Dalam arena tersebut memiliki banyak rintangan atau halangan yang harus dilewati oleh pemain khususnya musuh-musuh yang selalu menghadang pemain.
2. Pemain

   Pemain bernama Kabayan merupakan *actor* utama yang menggunakan aplikasi *game*.Kabayan memiliki kemampuan untuk melompat, menghindar, dan juga menembak musuh. Misi dari Kabayan adalah untuk mengumpulkan makanan tradisional yang tersebar.
3. Musuh

   Pada g*ame* Petualangan Kabayan Menyelamatkan Makanan Tradisionalterdapat lima musuh yaitu bebegig, kalong, guguk, bagong dan Rewog sebagai bos terakhir.
4. Makanan Tradisional

   Makanan merupakan entitas yang harus dikumpulkan oleh pemain.

3.1.3.7 Skenario *Game* Petualangan Kabayan Menyelamatkan Makanan Tradisional

Skenario pada *Game*Petualangan Kabayan Menyelamatkan Makanan Tradisional merupakan alur-alur yang terdapat pada sebuah aplikasi permainan agar setiap pemain dapat mengetahui bagaimana cara untuk memainkan permainan ini dan untuk menyelesaikan misi pada aplikasi permainan tersebut.

Skenario permainan dari *Game*Petualangan Kabayan Menyelamatkan Makanan Tradisional terdapat pada tabel 3.1

Tabel 3.1 Skenario Aplikasi Permainan

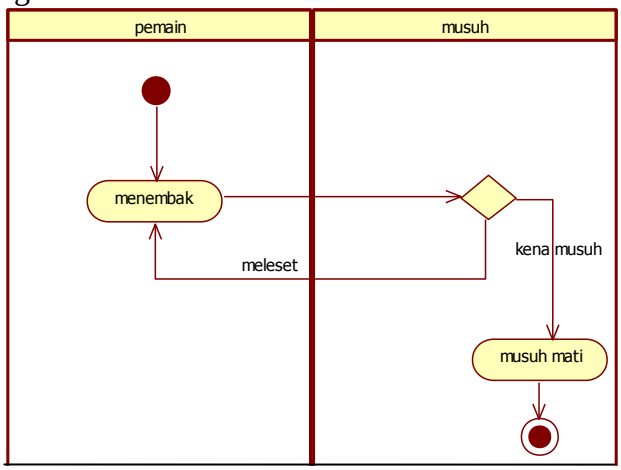
Alur Permainan
1. Pemain bergerak dari kiri ke kanan sesuai dengan jalur yang telah disediakan 2. Pemain harus mengumpulkan makanan tradisional yang tersebar di area permainan 3. Pemain harus menghindari musuh yang berada di area permainan

3.1.4 Penerapan Metode *Collision Detection*

Agar aplikasi permainan Petualangan Kabayan Menyelamatkan Makanan Tradisionalini menjadi lebih menarik, maka akan diterapkan metode *collision detection* yang berupa metode dimana yang bersinggungan merupakan objek secara langsung, baik itu *sprite* dengan *sprite* maupun *sprite* dengan objek lain.

Pada aplikasi permainan Petualangan Kabayan Menyelamatkan Makanan Tradisional, metode yang akan diterapkan pada peluru yang ditembakan oleh para pemain untuk membunuh setiap musuh yang ada adalah metode *collision* antara objek dengan objek yang terjadi apabila objek tersebut saling bersentuhan pada titik yang sama.

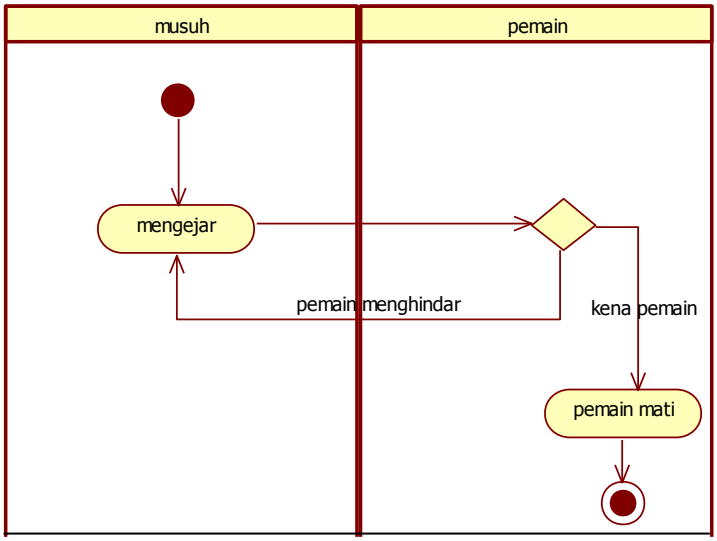
Diagram Activity metode *collision* tembakan pemain menyerang musuh ditunjukkan pada gambar 3.8



Gambar 3.8 Diagram Activity Tembakan Pemain Menyerang Musuh

Selain itu Pada aplikasi permainan Petualangan Kabayan Menyelamatkan Makanan Tradisional, metode yang akan diterapkan pada musuh yang menyerang pemain adalah metode *collision* antara objek dengan objek yang terjadi apabila objek tersebut saling bersentuhan pada titik yang sama.

Diagram Activity metode *collision* musuh menyerang pemain ditunjukkan pada gambar 3.9

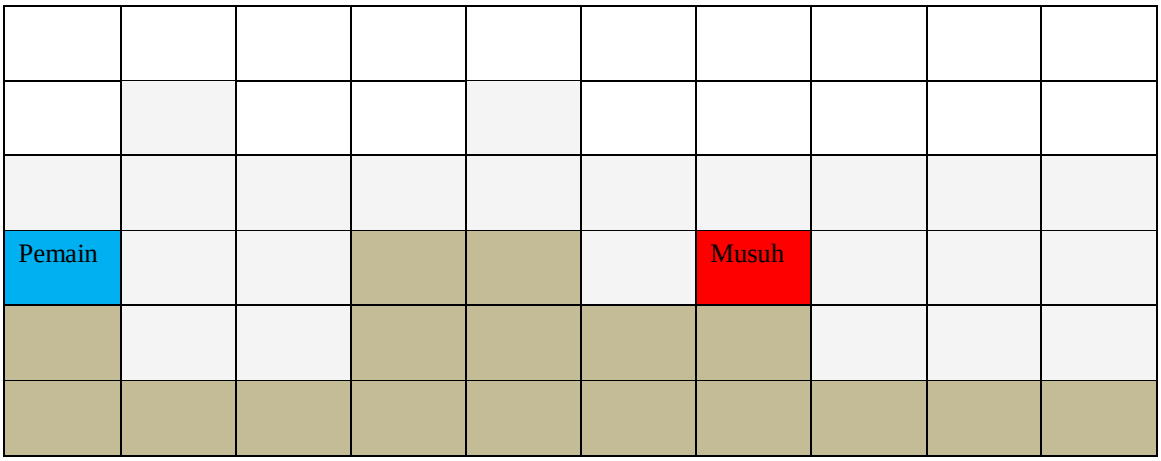


Gambar 3.9 Diagram Activity Metode Collision Musuh Menyerang Pemain

3.1.5 Analisis Algoritma A* (A *Star*)

Untuk membuat permainan *game* menjadi lebih menarik dapat dilakukan dengan meningkatkan kualitas *gameplay* bagi pengguna yang memainkannya. AI merupakan faktor yang penting dalam meningkatkan *gameplay* pada semua jenis permainan *game*, termasuk pada *game* Petualangan Kabayan Menyelamatkan Makanan Tradisional yang akan dibangun. Hal penting dalam menentukan keberhasilan suatu aplikasi *game* yang menerapkan AI adalah kesuksesan dalam pencarian dan pelacakan. Pada dasarnya ada dua teknik pencarian dan pelacakan yang digunakan, yaitu pencarian buta (*blind search*) dan pencarian terbimbing (*heuristic search*). Dalam pencarian terbimbing ada beberapa algoritma yang dapat digunakan salah satunya algoritma A*.

Algoritma A* melakukan pencarian jarak terpendek dengan biaya yang diperhitungkan didapat dari biaya sebenarnya ditambah dengan biaya perkiraan. Dalam notasi matematika dituliskan sebagai : $f(n) = g(n) + h(n)$. Dengan perhitungan biaya seperti ini, Algoritma A* adalah *complete* dan optimal. Untuk penerapannya dapat dilihat pada gambar di bawah ini



Gambar 3.10 Contoh kasus pencarian jalur terpendek menggunakan algoritma A*

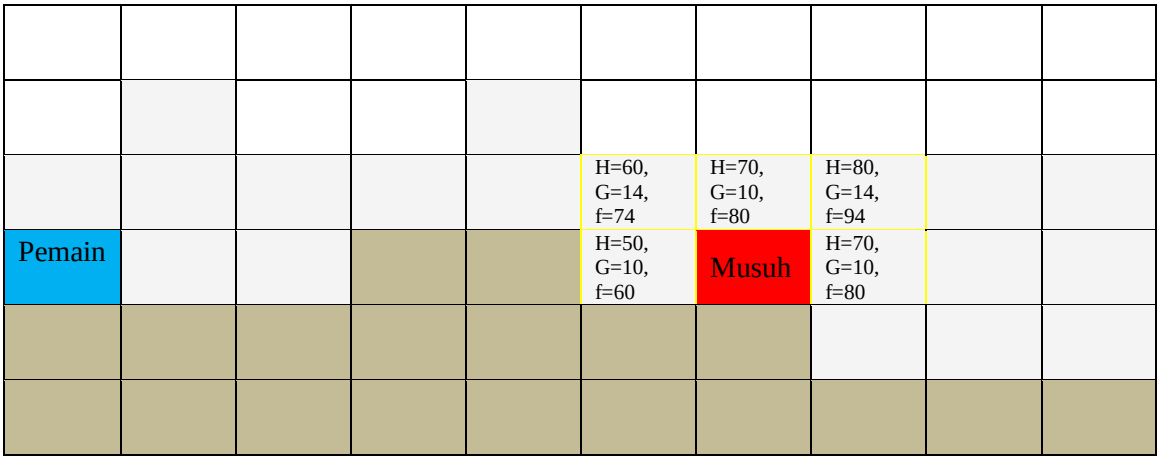
Pada gambar 3.10 , Diasumsikan *node* berwarna biru sebagai pemain,*node* berwarna merah sebagai musuh, sedangkan *node berwarna* coklat muda sebagai

penghalang (*obstacle)*. Pada algoritma A*, *actual cost* atau G di dapat dari jarak perkiraan jarak dari *node* musuh ke *node* pemain.

Dalam algoritma A* terdapat 3 nilai yaitu : f, g dan h. Nilai f adalah nilai yang menyatakan seberapa bagus sebuah *node*, nilai g adalah *actual cost* atau total jarak menuju ke n *node*, sedangkan nilai h adalah *heuristic cost* atau biaya perkiraan. Untuk mendapatkan nilai heuristik menggunakan *diagonal distance* yang memungkinkan pergerakan ke 8 arah maka digunakan notasi matematika

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

Berikut ini adalah langkah pencarian rute terpendek menggunakan algoritma A*



Gambar 3.11 Pembangkitan suksesor *node* musuh

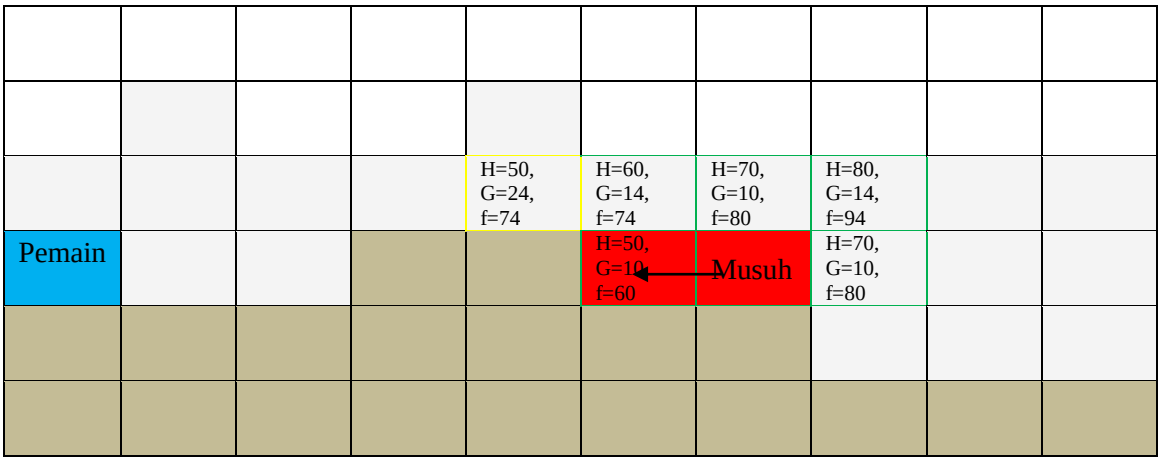
Langkah pertama seperti pada gambar 3.11, musuh memiliki 5 *node*, baik arah horizontal, vertikal, maupun diagonal, kemudian kelima *node* tersebut di masukan ke *open list* dengan warna garis *node* berwarna kuning, unuk di cek nilai setiap *nodenya*. *Node* pertama dengan arah horizontal kanan memiliki nilai G = 10, nilai H = 70, sehingga f = G + H = 80. *Node* kedua dengan arah diagonal kanan atas memiliki nilai G = 14, H = 80 sehingga f = G + H = 94. *Node* ketiga dengan arah horizontal atas memiliki nilai G = 10, H = 70 sehingga f = G + H = 80. *Node* keempat dengan arahdiagonal kiri atas memiliki nilai G = 14, H = 60 sehingga f = G + H = 74. *Node* kelima dengan arah horizontal kiri memiliki nilai G = 10, H = 85 sehingga f = G + H = 60. Untuk perhitungah nilai H digunakan fungsi *heuristic*, metode yang digunakan di dalam contoh ini adalah metode *Manhattan*

dimana perhitungan jumlah *node*hanya yang bergerak secara *vertical* dan *horizontal* menuju tujuannya serta mengabaikan penghalang atau *collision*, yang kemudian nilainya dikalikan dengan 10. Atau dirumuskan dengan: H = 10*(abs(*current*X-*target*X) + abs(*current*Y-*target*Y))

Sebagai contoh *node* yang terletak disebelah kanan dari *node* musuh diberikan nilai H = 50 yang didapatkan dari perhitungan 5 langkah atau 5 *node* menuju *node* pemain dan dikalikan 10, 5*10 = 50.

Setelah didapatkan nila F dari setiap *node* yang ada di *open list*, maka akan dipilih *node* yang memiliki nilai F paling rendah yang kemudian disebut dengan *current node*, selanjutnya langkahnya adalah sebagai berikut.

1. Hapus *current node* tersebut dari *open list* dan masukkan ke dalam *closed list* dengan warna garis *node* hijau.
2. Periksa semua *node* yang berdekatan dengan *current node*, abaikan *node* yang *unwalkable* atau dalam contoh ini adalah *node* penghalang. Jika tidak ada pada *open list*, tambahkan ke dalam *open list.* Dan tambahkan "*parent*" untuk *node* tersebut.
3. Jika *node* sudah ada pada *open list*, periksa apakah ini jalan yang lebih baik dari *node* awal, dengan kata lain periksa apakah nilai G-nya lebih rendah apabila kita menggunakan *node* ini, jika lebih rendah maka hitung ulang nilai F dan G serta merubah arah *pointer*, jika tidak maka jangan lakukan apa-apa.

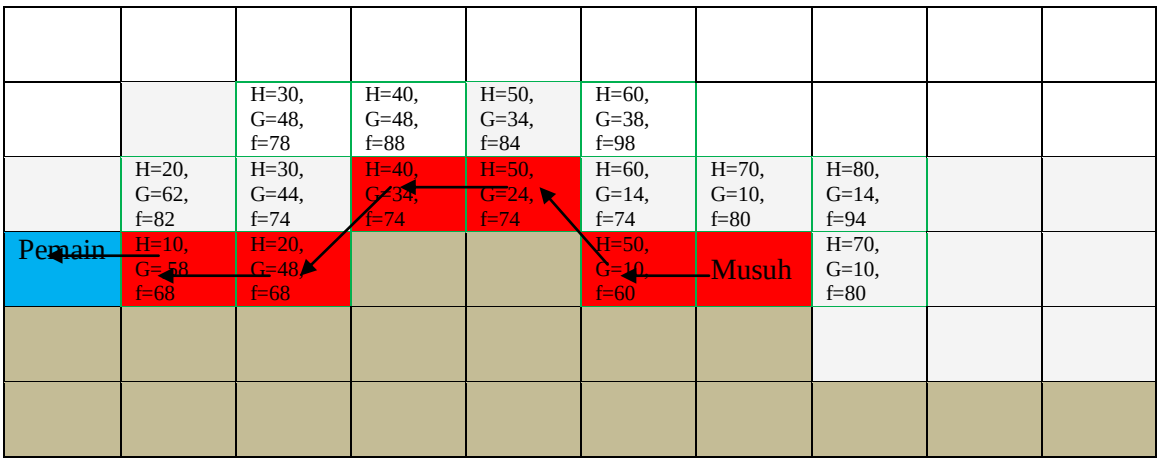


Gambar 3.12 Langkah 1 pencarian rute terpendek menggunakan algoritmaA*

Karena *node* dengan arah horizontal kiri memiliki biaya terkecil (f=60) terpilihlah sebagai *bestnode* seperti pada gambar 3.12, dan pencarian selanjutnya dimulai dari *node* ini, dengan membuka *node-node* yang ada di sekitarnya dan memasukan *none-node* sebelumnya ke *closed list* dengan warna garis *node* hijau. Pencarian dimulai dengan menelusuri node dengan arah diagonal kiri atas memiliki nilai G yang dulu tambah 14 = 24, H = 50, sehingga f = G + H = 74.

Lakukan perhitungan lagi seperti langkah sebelumnya, hapus dari *open list* dan tambahkan ke dalam *closed list*, periksa semua *node* yang berdekatan, masukkan ke dalam *open list* dan tambahkan *parent*nya, dan periksa lagi jalan terbaik dan *parent*nya. Terus ulangi sampai *target* ditambahkan ke dalam *closed list* atau di dalam contoh ini adalah *node* pemain.

Perhitungannya diilustrasikan dengan gambar berikut.



Gambar 3.13 Hasil pencarian rute terpendek menggunakan algoritma A*

Karena *node* dengan arah horizontal kiri memiliki biaya terkecil (f = 68), dan *node* pemain ketemu seperti pada gambar 3.13.

Jika dua atau lebih node yang diperiksa mempunyai harga *f(n)*yang sama,maka terjadi keadaan sepadan (*tie*). Hal ini dimungkinkan karena *f(n)* bergantung pada duafungsi, yaitu fungsi *g(n)* dan *h(n)*. Hal ini sangat mungkin terjadi antara node-node yang letaknya berjauhan, dan kemungkinan besar node

yang satu terletak dekat node tujuan sedangkan yang lainnya terletak jauh dari node tujuan. Karena algoritma A* memberikan prioritas berdasarkan harga *f(n)*, maka jika keadaan sepadan terjadi, terdapat lebih dari satu node dengan prioritas sama. Akibatnya adalah node-node tersebut akan diperiksa lebih dulu, yang mungkin node tersebut terletak berjauhan dengan node tujuan. Hal ini berakibat turunnya kinerja algoritma A*.

Dari berwarna merah atau musuh menuju *node* berwarna biru atau pemain maka didapatkan jalur terpendek dengan menggunakan algoritma A* sebagai berikut :

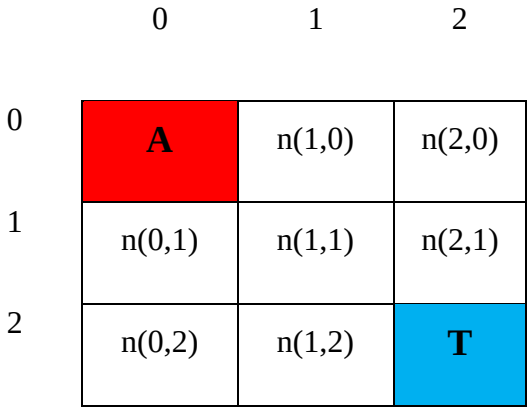
1. Dari *node* berwarna merah atau musuh mengambil jalur ke arah horizontal kiri memiliki biaya terkecil (f=60).
2. Dari *node*horizontal kanan memiliki biaya terkecil (f=60) mengambil jalur ke arah diagonal kiri atas memiliki biaya terkecil (f = 74).
3. Dari *node*diagonal kiri atas memiliki biaya terkecil (f = 74) mengambil jalur ke arah horizontal kiri memiliki biaya terkecil (f = 74).
4. Dari *node*horizontal kiri memiliki biaya terkecil (f = 74) mengambil jalur ke arah *node*diagonal kiri bawah memiliki nilai terkecil (f = 68).
5. Dari *node* diagonal kiri bawah memiliki nilai terkecil (f = 68). mengambil jalur ke arah horizontal kiri memiliki biaya terkecil (f = 68) dan sampailah di *node* berwarna biru atau pemain.

Karena objek musuh dapat bergerak secara diagonal di arena permainan, maka digunakan fungsi *heuristic Non-Manhattan Distance*, dengan fungsi *heuristic* yang digunakan adalah sebagai berikut:

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

berikut adalah perhitungan yang akan dilakukan dengan algoritma A* dengan kondisi tanpa penghalang yang terlihat pada gambar. Pada penentuan bobot setiap *node* diberikan nilai sesuai dengan jarak terdekat

ke tujuan, misal *node* yang terjauh dari tujuan maka diberi bobot yang kecil sedangkan *node* yang terdekat dengan tujuan diberi bobot yang lebih besar. Perhitungan yang dilakukan dengan algoritma A* dengan kondisi tanpa penghalang yang terlihat pada gambar berikut :



Gambar 3.14 Hasil perhitungan *heuristic diagonal distance* tahap 1

Keterengan :

A = simpul awal

T = simpul akhir

Langkah tahap pertama :

1. Koordinat n(1,0)

   Nilai n.x = 1

   Nilai n.y = 0

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

   g (1,0) = 1

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

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

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

   = 3

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

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

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

   = min 3

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

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

= -3 + 9

= 6

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

= 1 + 6

= 7

2. Koordinat n(0,1)

Nilai n.x = 0

Nilai n.y = 1

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

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

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

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

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

= 3

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

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

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

= min 3

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

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

= -3 + 9

= 6

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

= 1 + 6

= 7

3. Koordinat (1,1)

Nilai n.x = 1

Nilai n.y = 1

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

g (1,1) = 1

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

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

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

= 2

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

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

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

= min 2

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

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

= -2 + 6

= 4

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

= 1 + 4

= 5

Dari hasil perhitungan heuristic diagonal pertama maka didapatlah hasil seperti gambar

	0	1	2
0	**A**	**7**	
1	**7**	**5**	
2			**T**

Gambar 3.15 Hasil perhitungan *heuristic diagonal distance* tahap 2

Langkah tahap pertama :

1. Koordinat n(2,1)

Nilai n.x = 2

Nilai n.y = 1

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

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

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

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

= 1

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

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

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

= min 1

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

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

= -1 + 3

= 2

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

= 2 + 2

= 4

2. Koordinat n(1,2)

Nilai n.x = 1

Nilai n.y = 2

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

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

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

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

= 1

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

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

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

= min 1

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

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

= -1 + 3

= 2

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

= 2 + 2

= 4

3. Koordinat n(2,2)

Nilai n.x = 2

Nilai n.y = 2

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

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

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

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

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

= 0

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

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

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

= min 0

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

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

= 0 + 0

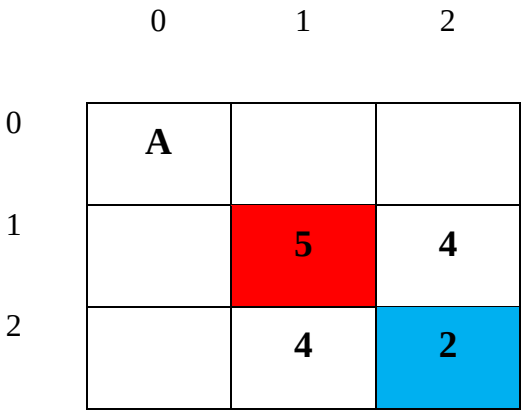
= 0

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

= 2 + 0

= 2

Dari hasil perhitungan *heuristic diagonal* kedua maka didapatlah hasil seperti gambar



Gambar 3.16 Hasil perhitungan *heuristic diagonal distance* tahap 3

3.1.6 Analisis Kebutuhan Non Fungsional

Pada tahap ini meliputi elemen-elemen yang diperlukan dalam membuat aplikasi *Game* Petualangan Kabayan menyelamatkan makanan tradisional. Kebutuhan non fungsional terbagi menjadi tiga yaitu analisis perangkat keras, analisis perangkat lunak dan analisis pengguna.

3.1.6.1 Analisis Perangkat Keras

Perangkat keras atau *hardware* merupakan salah satu hal yang penting karena tanpa *hardware* yang memenuhi syarat, program yang akan dibuat tidak akan dapat berjalan. Berikut spesifikasi standar perangkat keras yang dapat dipergunakan untuk membangun aplikasi *game* ini dan spesifikasi perangkat keras yang dapat dipergunakan untuk menjalankan alplikasi *game* ini yaitu :

A. Spesifikasi standar perangkat keras untuk pembangunan aplikasi *game* tersebut adalah :
   1. Processor2.6 Ghz
   2. RAM 1 GB
   3. VGA 512 MB
   4. *Harddisk*26MB

5. Monitor
6. *Mouse, keyboard dan speaker.*

B. Spesifikasi standar perangkat keras untuk dapat menjalankan aplikasi *game* tersebut adalah :
   1. Processor 2.6 Ghz
   2. RAM 1 GB
   3. VGA 512 MB
   4. *Harddisk* 26MB
   5. Monitor
   6. *Mouse, keyboard dan speaker.*

3.1.6.2 Analisis Perangkat Lunak

Perangkat lunak atau *software* merupakan hal yang terpenting dalam mendukung kinerja sebuah sistem. Perangkat lunak digunakan dalam sebuah sistem merupakan perintah-perintah yang diberikan kepada perangkat keras agar dapat salung berinteraksi diantara keduanya.

Perangkat lunak yang dapat dibutuhkan untuk membangun aplikasi *game* Petualangan Kabayan menyelamatkan makanan tradisional ini adalah sebagai berikut :

1. Sistem Operasi Windows XP, Windows 7 dan sistem operasi lainnya.
2. Greenfoot-windows-2.2.1
3. Java Jdk-6u2-windows-i586-p ke atas (minimal JDK 1.5).

Sedangkan perangkat lunak yang dapat dibutuhkan untuk menjalankan aplikasi *game* Petualangan Kabayan menyelamatkan makanan tradisional ini adalah sebagai berikut :

1. Sistem Operasi Windows XP, Windows 7 dan sistem operasi lainnya.
2. Greenfoot-windows-2.2.1
3. Java Jdk-6u2-windows-i586-p ke atas (minimal JDK 1.5)

3.1.6.3 Analisis Pengguna

Berdasarkan hasil penelitian, spesifikasi minimal bagi pengguna yang dapat menggunakan aplikasi ini adalah sebagai berikut :

Tabel 3.2 Karakteristik Pengguna

Tingkat pengetahuan dan pengalaman	
Kecakapan menggunakan komputer	Pengalaman menggunakan komputer / notebook
Pengalaman sistem	Pernah bermain *pc game*
Pengalaman aplikasi	Sering bermain *pc game*
Pengalaman tugas	Sering bermain *game*
Menggunakan sistem lain	Sering
Edukasi	Minimal murid SD kelas 4, atau usia 10 tahun, karena memiliki sudah memiliki sedikit sugesti (pengaruh psikis baik yang datang dari diri sendiri)
Tingkat baca	Minimal murid SD kelas 4
Kemampuan mengetik	Tidak diperlukan
Bahasa yang biasa digunakan	Bahasa Indonesia
Handedness	Kanan dan Kiri
Color Blind	Pemain yang tidak bisa membedakan warna yang satu dengan yang lainnya (buta warna) masih mampu menggunakan aplikasi ini, karena tidak ada indikator warna-warna khusus yang membedakan antara fungsional yang satu dengan fungsional yang lainnya.

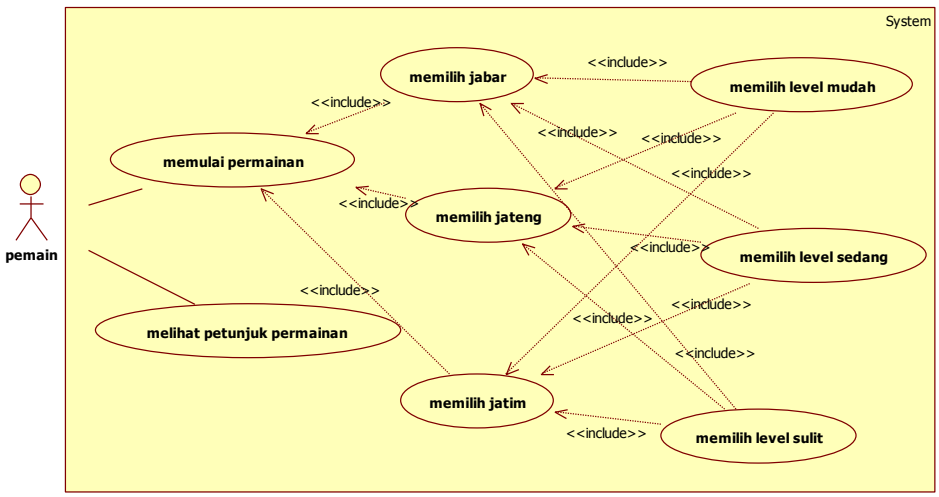
3.1.7 Analisis Kebutuhan Fungsional

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

3.1.7.1 Use Case Diagram

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

Secara umum, interaksi antara pengguna dengan aplikasi permainan *game* Petualangan Kabayan menyelamatkan makanan tradisional digambarkan pada diagram *use case* pada gambar 3.17



Gambar 3.17 Use Case Diagram Menu Utama (User)

3.1.7.2 Definisi Use Case

Definisi *use case* berfungsi untuk menjelaskan proses yang terdapat pada setiap *use case*. Berikut dibawah ini tabel dari definisi *use case* yaitu :

Tabel 3.3 Definisi *Use Case*

No.	*Use Case*	Deskripsi
1.	Memulai Permainan	Proses untuk memulai permainan
2.	Melihat petunjuk permainan	Proses untuk melihat petunjuk permainan
3.	Memilih Jabar	Proses untuk memulai *stage* Jawa Barat
4.	Memilih Jateng	Proses untuk memulai *stage* Jawa Tengah
5.	Memilih Jatim	Proses untuk memulai *stage* Jawa Timur
6.	Memilih level mudah	Proses untuk memilih level mudah
7.	Memilih level sedang	Proses untuk memilih level sedang
8.	Memilih level sulit	Proses untuk memilih level sulit

3.1.7.3 Skenario Use Case

Skenario *use case* merupakan bagian pada *use case* yang menunjukkan proses apa saja yang terjadi pada setiap bagian di dalam *use case*, dimana 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*. Berikut ini beberapa skenario *use case* berdasarkan *use case* yang ada yaitu :

Tabel 3.4 Skenario*Use Case* Memulai Permainan

Identifikasi	
Nomor	1
Nama	Memulai Permainan
Tujuan	Untuk Memulai permainan
Deskripsi	Proses untuk memulai permainan
Aktor	Pengguna
Skenario Utama	
Kondisi awal	Pengguna berada pada menu utama aplikasi *Game* Petualangan Kabayan Menyelamatkan Makanan Tradisional
Aksi Aktor	Reaksi Sistem
Memulai Permainan	Aplikasi merespon dengan cara masuk ke tampilan pilih *stage*
Kondisi akhir	Aplikasi masuk ke dalam menu pilih *stage*

Tabel 3.5 Skenario *Use Case* Melihat petunjuk permainan

Identifikasi	
Nomor	2
Nama	Melihat petunjuk permainan
Tujuan	Melihat informasi mengenai cara bermain
Deskripsi	Proses untuk melihat informasi petunjuk cara bermain
Aktor	Pengguna
Skenario Utama	
Kondisi awal	Pengguna berada pada menu utama aplikasi *Game* Petualangan Kabayan Menyelamatkan Makanan Tradisional
Aksi Aktor	Reaksi Sistem

Melihat petunjuk permainan	Aplikasi merespon dengan cara memanggil dan milihat petunjuk
Kondisi akhir	Aplikasi masuk ke dalam infrormasi petunjuk aplikasi *game*

Tabel 3.6 Skenario *Use Case* Memilih Jabar

Identifikasi	
Nomor	3
Nama	Memilih Jabar
Tujuan	Memilih *stage* yg akan di mainkan
Deskripsi	Proses untuk memilih *stage* Jawa Barat
Aktor	Pengguna
Skenario Utama	
Kondisi awal	Pengguna berada pada menu utama aplikasi *Game* Petualangan Kabayan Menyelamatkan Makanan Tradisional
Aksi Aktor	Reaksi Sistem
Memilih Jabar	Aplikasi merespon dengan cara masuk ke sistem dari aplikasi permainan
Kondisi akhir	Aplikasi masuk ke dalam menu pilih *level* Jawa Barat

Tabel 3.7 Skenario *Use Case* Memilih Jateng

Identifikasi	
Nomor	4
Nama	Memilih Jateng
Tujuan	Memilih *stage* yg akan di mainkan
Deskripsi	Proses untuk memilih *stage* Jawa Tengah

Aktor	Pengguna
Skenario Utama	
Kondisi awal	Pengguna berada pada menu utama aplikasi *Game* Petualangan Kabayan Menyelamatkan Makanan Tradisional
Aksi Aktor	Reaksi Sistem
Memilih Jateng	Aplikasi merespon dengan cara masuk ke sistem dari aplikasi permainan
Kondisi akhir	Aplikasi masuk ke dalam menu pilih *level* Jawa Tengah

Tabel 3.8 Skenario *Use Case* Memilih Jatim

Identifikasi	
Nomor	5
Nama	Memilih Jatim
Tujuan	Memilih *stage* yg akan di mainkan
Deskripsi	Proses untuk memilih *stage* Jawa Timur
Aktor	Pengguna
Skenario Utama	
Kondisi awal	Pengguna berada pada menu utama aplikasi *Game* Petualangan Kabayan Menyelamatkan Makanan Tradisional
Aksi Aktor	Reaksi Sistem
Memilih Jatim	Aplikasi merespon dengan cara masuk ke sistem dari aplikasi permainan
Kondisi akhir	Aplikasi masuk ke dalam menu pilih *level* Jawa Timur

Tabel 3.9 Skenario *Use Case* Memilih level mudah

Identifikasi	
Nomor	6
Nama	Memilih level mudah
Tujuan	Memilih tingkatan *level*mudah untuk dimainkan
Deskripsi	Proses untuk memainkan permainan di *level* mudah
Aktor	Pengguna
Skenario Utama	
Kondisi awal	Pengguna berada pada tampilan permainan aplikasi *Game* Petualangan Kabayan Menyelamatkan Makanan Tradisional
Aksi Aktor	Reaksi Sistem
Memilih level mudah	Aplikasi merespon dengan cara menjalankan aplikasi permainan di *level* mudah
Kondisi akhir	Aplikasi masuk tampilan permainan *level* mudah

Tabel 3.10 Skenario*Use Case* Memilih level sedang

Identifikasi	
Nomor	7
Nama	Memilih level sedang
Tujuan	Memilih tingkatan *level*sedanguntuk dimainkan
Deskripsi	Proses untuk memainkan permainan di *level* sedang
Aktor	Pengguna
Skenario Utama	
Kondisi awal	Pengguna berada pada tampilan permainan aplikasi *Game* Petualangan Kabayan Menyelamatkan Makanan Tradisional

Aksi Aktor	Reaksi Sistem
Memilih level sedang	Aplikasi merespon dengan cara menjalankan aplikasi permainan di *level* sedang
Kondisi akhir	Aplikasi masuk tampilan permainan *level* sedang

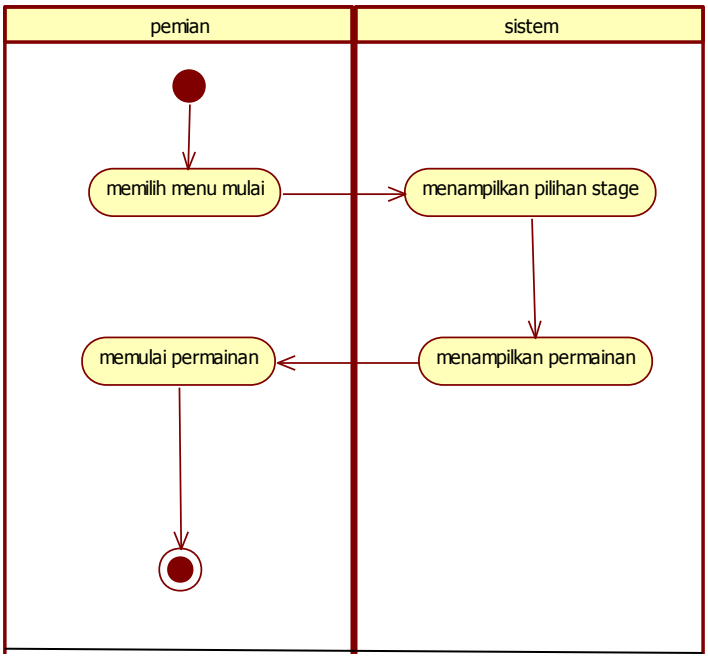
Tabel 3.11 Skenario *Use Case* Memilih level sulit

Identifikasi	
Nomor	8
Nama	Memilih level sulit
Tujuan	Memilih tingkatan *level*sulituntuk dimainkan
Deskripsi	Proses untuk memainkan permainan di *level* sulit
Aktor	Pengguna
Skenario Utama	
Kondisi awal	Pengguna berada pada tampilan permainan aplikasi *Game* Petualangan Kabayan Menyelamatkan Makanan Tradisional
Aksi Aktor	Reaksi Sistem
Memilih level sulit	Aplikasi merespon dengan cara menjalankan aplikasi permainan di *level* sulit
Kondisi akhir	Aplikasi masuk tampilan permainan *level* sulit

3.1.7.4 Activity Diagram

Activity diagram menggambarkan berbagai aliran aktivitas dalam sistem yang sedang dirancang.Berikut ini beberapa *activity* diagram yang terdapat pada *game* tersebut yaitu :

1. *Activity* Proses Memulai Permainan

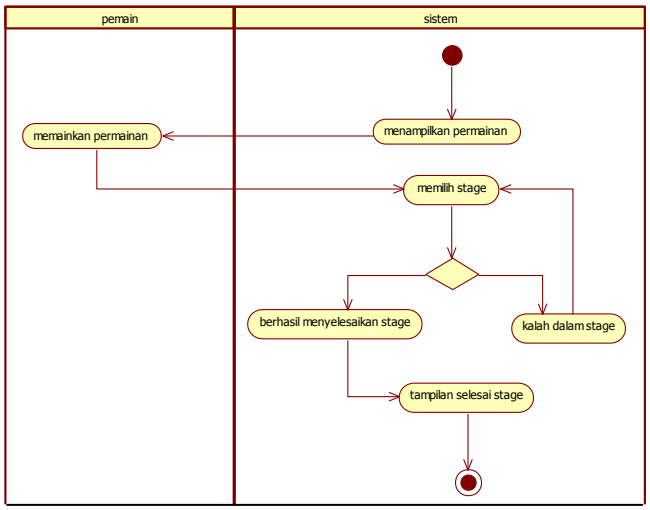


Gambar 3.18 *Activity* Proses memulai permainan

Tabel 3.12 Penjelasan Proses Memulai permainan

Deskripsi	Tampilan ini menampilkan pilihan menu mulai, dimana pemain akan memulai permainan
Normal *Flow*	1. Pemain memilih pilihan menu mulai 2. Pemain memilih *stage* 3. Pemain memainkan permainan 4. Pemain selesai memainkan *stage* yang di mainkan

2. *Activity Diagram* Permainan

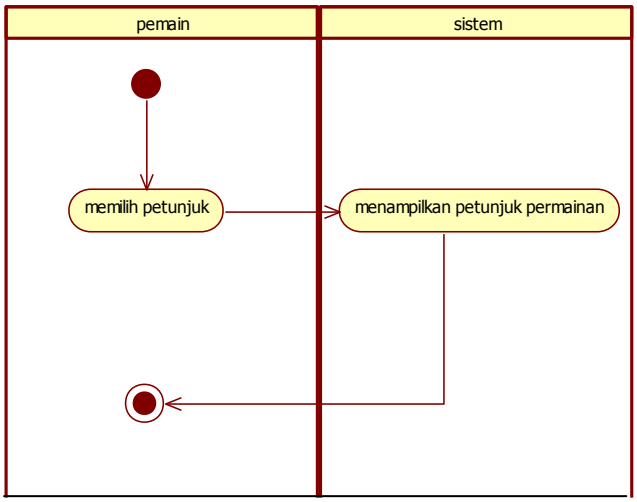


Gambar 3.19 *Activity Diagram* Permainan

Tabel 3.13 Penjelasan Proses permainan

Deskripsi	Permainan merupakan kondisi pemain ketika telah berada di dalam permainan.
Normal Flow	1. Sistem menampilkan permainan 2. Pemain mulai memainkan permainan 3. Pemain berhasil menyelesaikan *stage* yang dihadapi 4. Sistem menampilkan tampilan selesai *stage* 5. Pemain gagal menyelesaikan *stage* 6. Sistem kembali ke awal permainan

3. *Activity* Proses Tampil Petunjuk

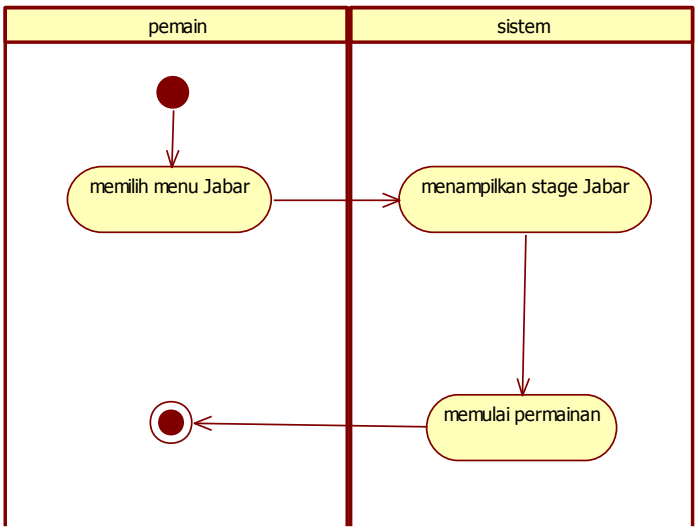


Gambar 3.20 *Activity* Proses memilih petunjuk permainan

Tabel 3.14 Penjelasan Proses memilih petunjuk permainan

Deskripsi	Menu petunjuk digunakan pemain untuk melihat petunjuk permainan
Normal *Flow*	1. Memilih Menu petunjuk di menu awal 2. Pemain melihat petunjuk permainan

4. *Activity* Proses Memilih Menu Jawa Barat

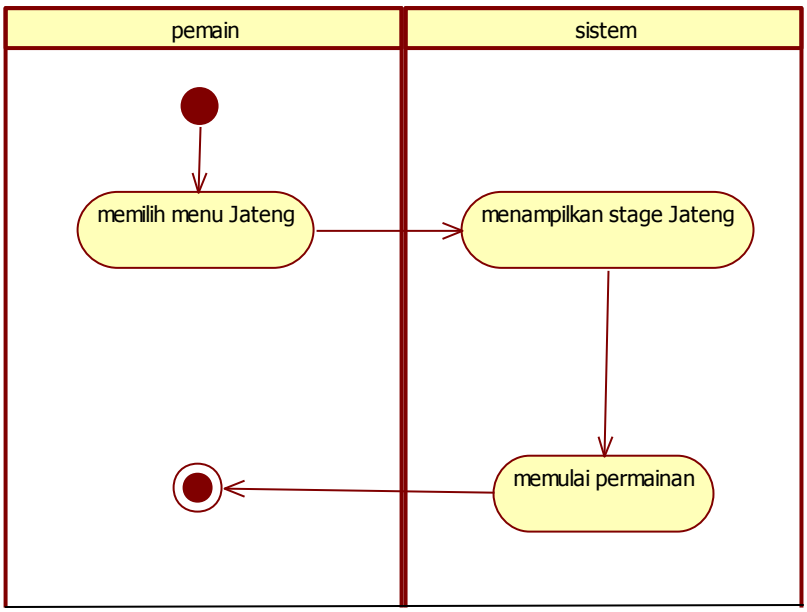


Gambar 3.21 *Activity* Proses Memilih Menu Jawa Barat

Tabel 3.15 Penjelasan Proses Memilih Menu Jawa Barat

Deskripsi	Tampilan ini menampilkan pilihan menu Jabar, dimana pemain akan memilih *stage* Jawa Barat untuk dimainkan.
Normal *Flow*	1. Pemain memilih pilihan menu Jabar 2. Pemain memulai permainan. 3. Pemain keluar dari permainan.

5. *Activity* Proses Memilih Menu Jawa Tengah



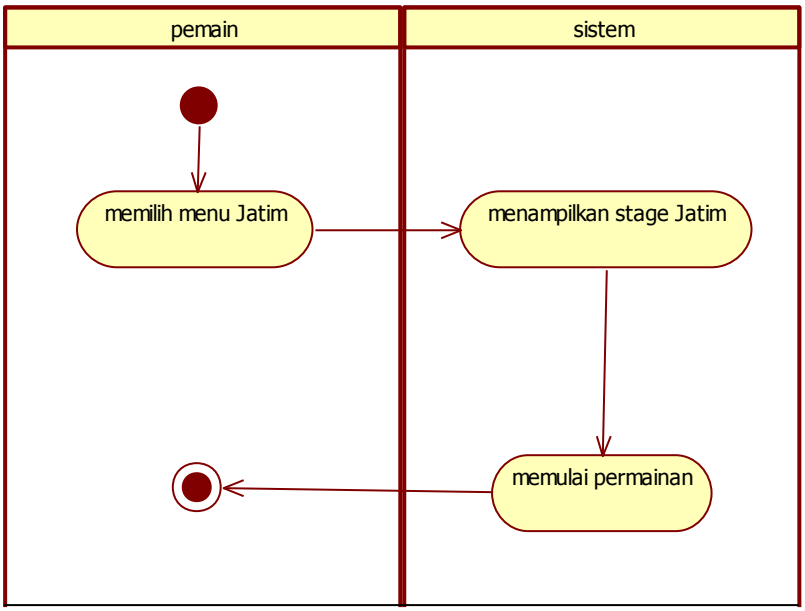
Gambar 3.22 *Activity* Proses Memilih Menu Jawa Tengah

Tabel 3.16 Penjelasan Proses Memilih Menu Jawa Tengah

Deskripsi	Tampilan ini menampilkan pilihan menu Jateng, dimana pemain akan memilih *stage* Jawa Tengah untuk dimainkan.

Normal *Flow*	1. Pemain memilih pilihan menu Jateng 2. Pemain memulai permainan. 3. Pemain keluar dari permainan.

3. *Activity* Proses Memilih Menu Jawa Timur



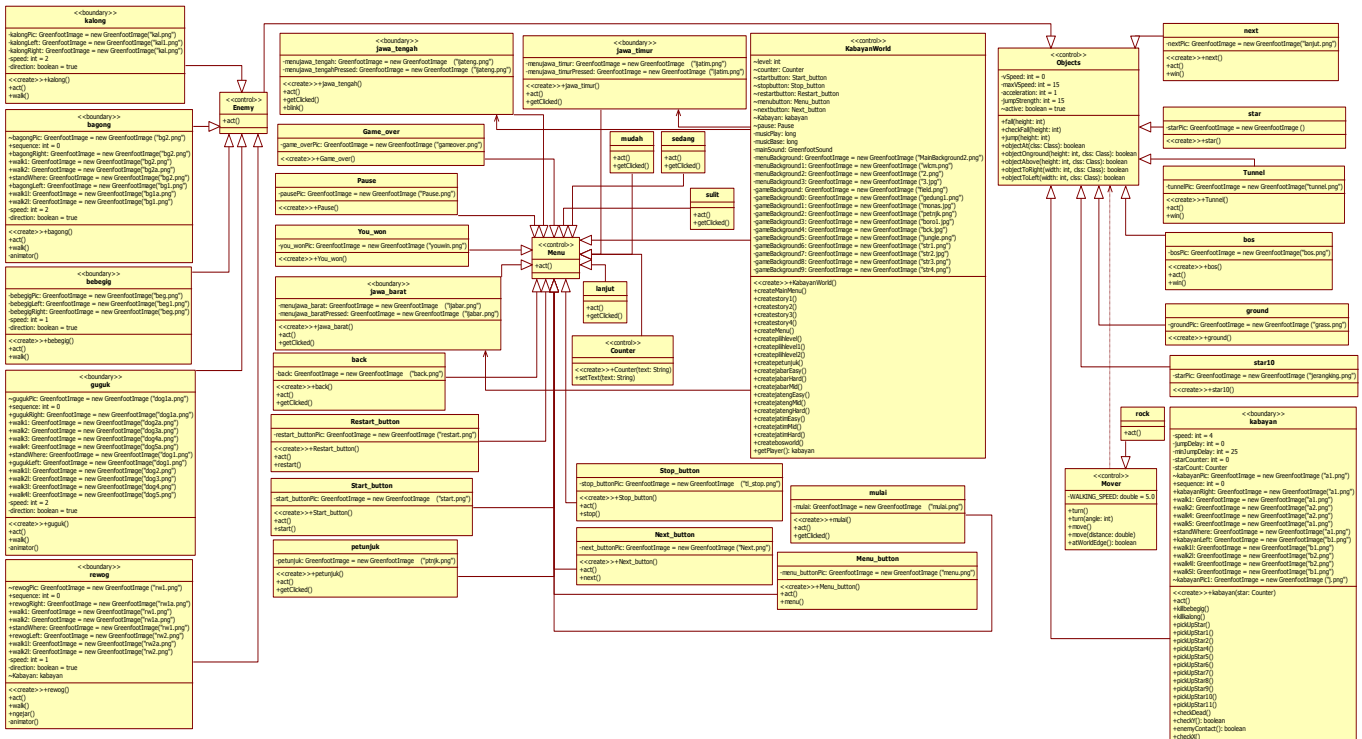
Gambar 3.23 *Activity* Proses Memilih Menu Jawa Timur

Tabel 3.17 Penjelasan Proses Memilih Menu Jawa Timur

Deskripsi	Tampilan ini menampilkan pilihan menu Jatim, dimana pemain akan memilih *stage* Jawa Timur untuk dimainkan.
Normal *Flow*	1. Pemain memilih pilihan menu Jatim 2. Pemain memulai permainan. 3. Pemain keluar dari permainan.

3.1.7.5 Class Diagram

Class Diagram menggambarkan keadaan suatu sistem (*atribut*), dan memberikan pelayanan untuk menyelesaikan keadaan tersebut (metoda).

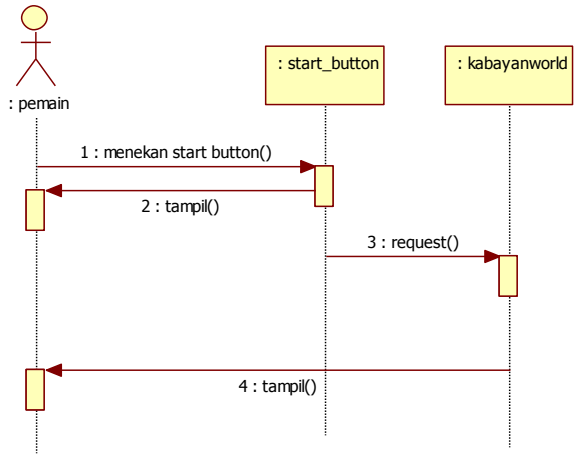


Gambar 3.24 Class Diagram

3.1.7.6 *Sequence* Diagram

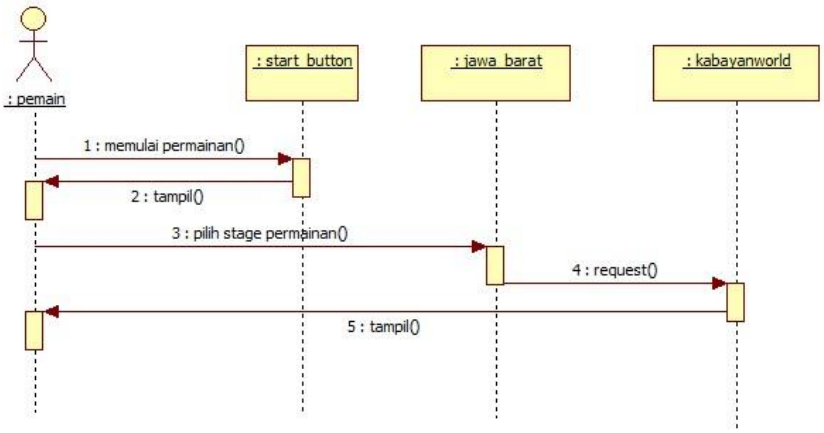
Diagram *sequence* atau *sequence* diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message*yang dikirimkan dan diterima antar objek

1. *Sequence Diagram* mulai permainan



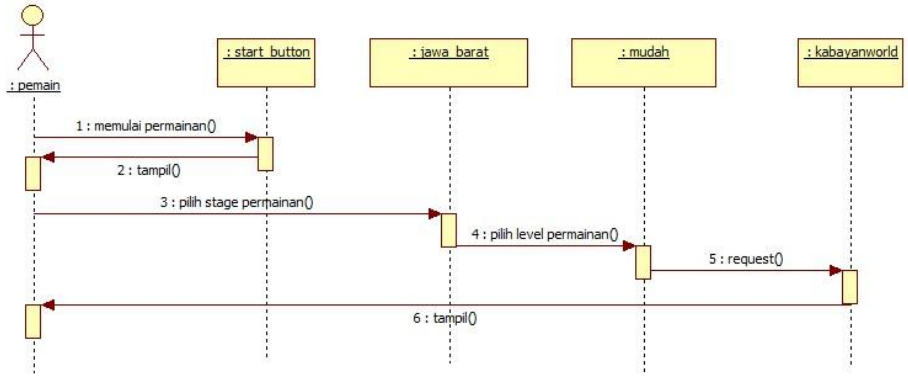
Gambar 3.25 *Sequence Diagram* Mulai Permainan

2. *Sequence Diagram* pilih stage Jawa Barat



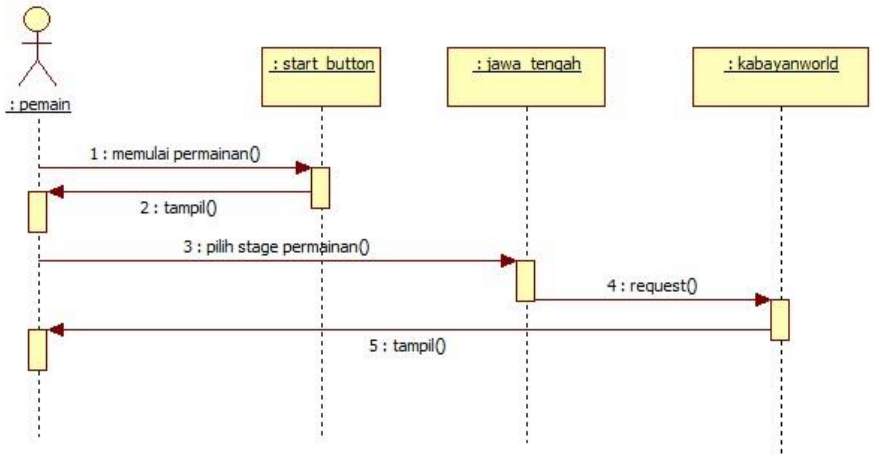
Gambar 3.26 *Sequence Diagram* pilih stage Jawa Barat

3. Sequence Diagram pilih stage Jawa Barat level mudah



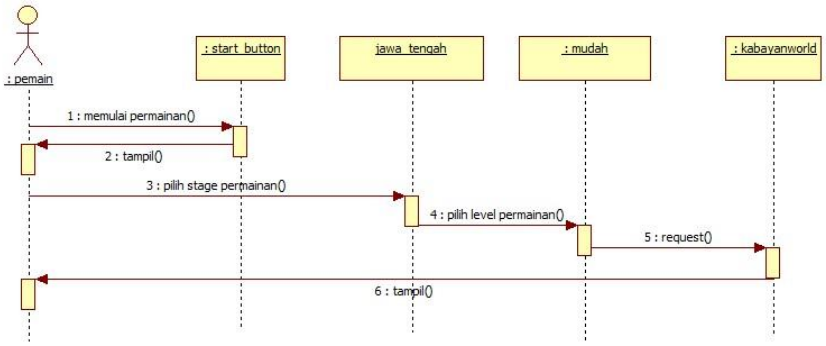
Gambar 3.27 *Sequence Diagram* pilih stage Jawa Barat level mudah

4. *Sequence Diagram* pilih stage Jawa Tengah



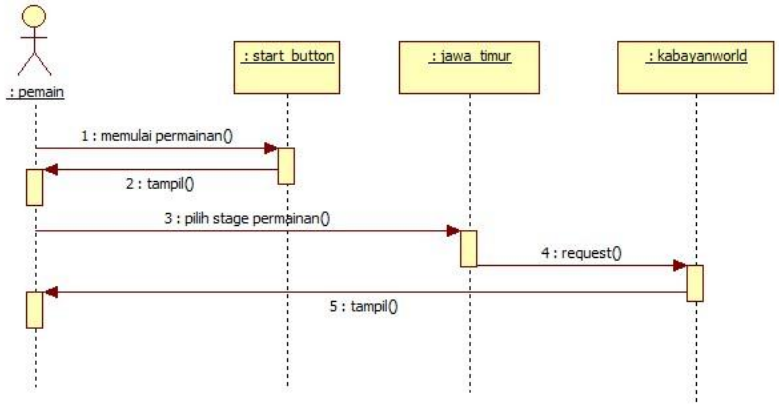
Gambar 3.28 *Sequence Diagram* stage Jawa Tengah

5. *Sequence Diagram* pilih stage Jawa Tengah level mudah



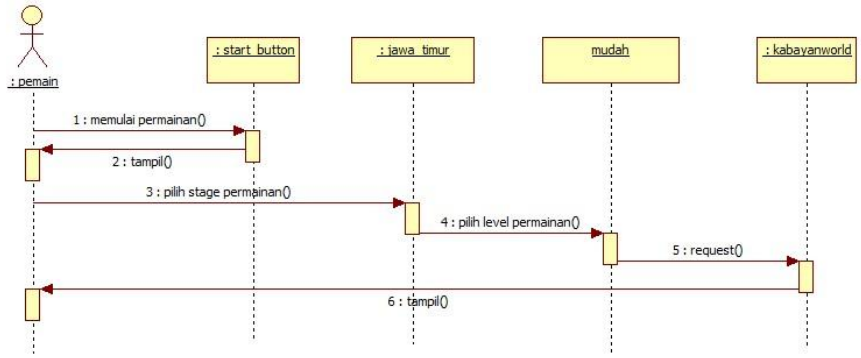
Gambar 3.29 *Sequence Diagram* pilih stage Jawa Tengah level mudah

***6. Sequence Diagram* pilih stage Jawa Timur**



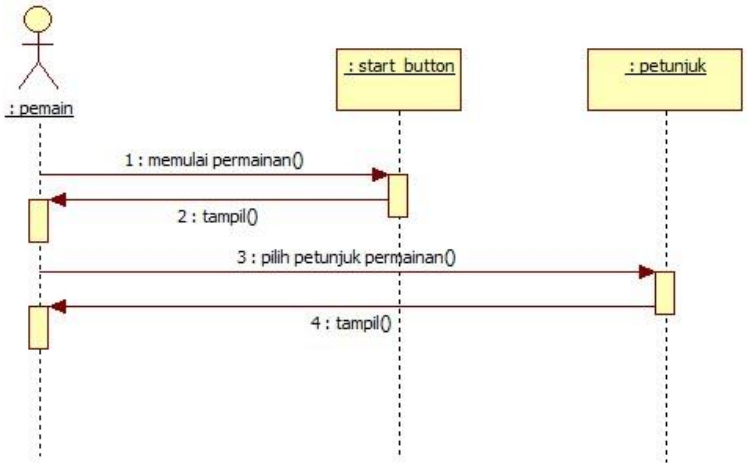
Gambar 3.30 *Sequence Diagram* pilih stage Jawa Timur

***7. Sequence Diagram* pilih stage Jawa Timur level mudah**



Gambar 3.31 *Sequence Diagram* pilih stage Jawa Timur level mudah

***8. Sequence Diagram* Lihat Petunjuk**



Gambar 3.32 *Sequence Diagram* Lihat Petunjuk

3.2 Perancangan

Setelah melakukan perancangan data pada sistem yang dibangun, maka dilakukanlah perancangan arsitektur. Perancangan arsitektur yang telah dibuat meliputi beberapa perancangan diantaranya perancangan komponen permainan, struktur menu, perancangan antarmuka.

3.2.1 Perancangan Komponen Permainan

Deskripsi karakter *game* merupakan penjelasan dari setiap karakter yang terdapat pada *game*. Barikut beberapa karakter yang terdapat pada *game* tersebut yaitu :

3.2.1.1 Karakter *Game*

Tabel 3.18 Deskripsi Karakter *Game*

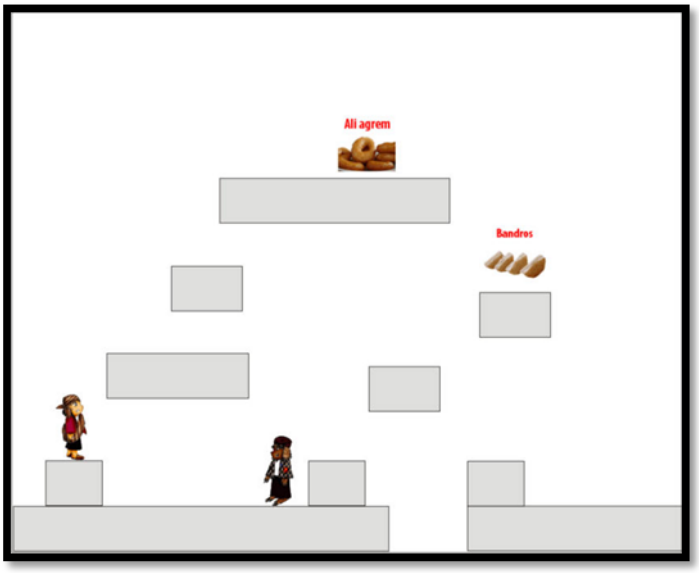
No	Gambar Karakter	Nama Karakter	Deskripsi
1		Kabayan	Kabayan adalah sebagai tokoh utama dalam *game* petualangan ini. Kabayan memiliki kemampuan menembak dengan ketepel, loncat kanan kiri, maju ke kanan dan kiri.
2		Rewog	Rewog adalah bos terakhir yang akan di kalahkan di *stage* ke 3, memiliki kemampuan menyerang, loncat ke kanan dan kiri, maju ke kanan dan kiri.
3		Kalong	Kalong merupakan salah satu musuh yang terdapat pada setiap *stage*. Memiliki kemampuan terbang mengejar kabayan.
4		Bebegig	Bebegig salah satu musuh yang akan terdapat pada setiap *stage*. Memiliki kemampuan bergerak ke kiri dan ke kanan
5		Guguk	Guguk musuh yang terdapat di *stage* 2 dan 3. Memiliki kemampuan mengejar.
6		Bagong	Bagong musuh biasa di *stage* 3. Memiliki kemampuan mengejar dan menabrak.

3.2.1.2 Storyboard

Storyboard adalah sketsa gambar yang disusun berurutan sesuai dengan naskah, dengan menggunakan *storyboard* dapat menyampaikan ide cerita kepada orang lain dengan lebih mudah karena dapat dengan *storyboard* dapat mengimajinasikan khayalan yang sesuai dengan gambar-gambar yang ada sehingga dapat menghasilkan persepsi yang sama pada ide cerita yang dibuat.

Berikut beberapa *storyboard* dari Aplikasi *Game* Petualangan Kabayan Menyelamatkan Makanan Tradisional yaitu :

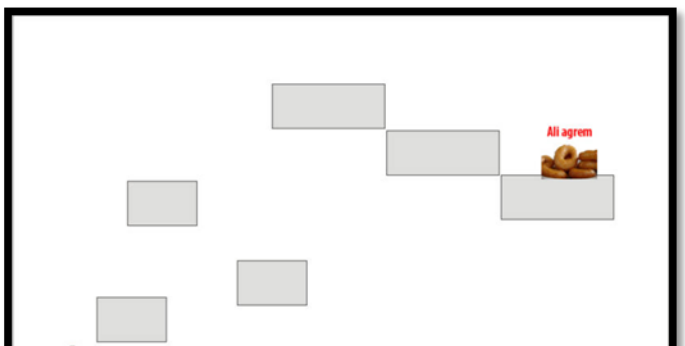
1. *Storyboard stage* jabar *level* mudah



Gambar 3.33 *Storyboard stage* jabar *level* mudah

Pada *stage* ini dimulai dari pulau Jawa Barat, kabayan akan mencari makanan tradisional khas jawa barat. Pada tingkatan level mudah, terdapat bebegig sebagai musuh yang menghadang seperti pada gambar 3.33.

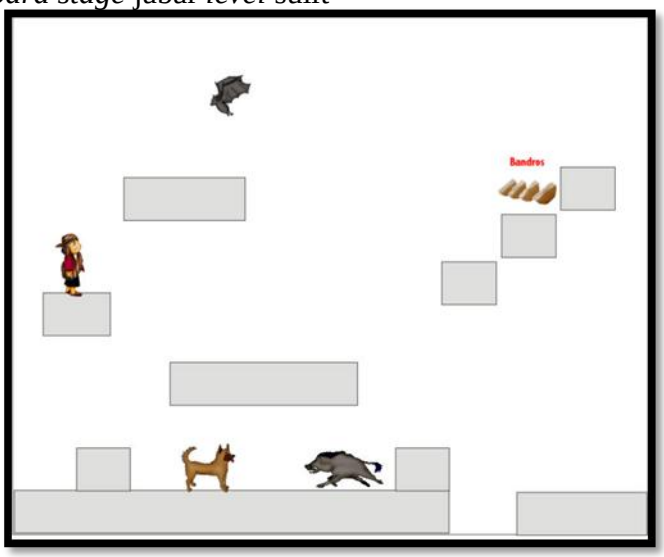
2. *Storyboard stage* jabar *level* sedang



Gambar 3.34 *Storyboard stage* jabar *level* sedang

Pada tingkatan level sedang musuh yang menghadang berambah yaitu guguk. Dan arena pun bertambah sulit. Kabayan harus mencari makanan tradisional yang tersebar seperti pada gambar 3.34.

3. *Storyboard stage* jabar *level* sulit



Gambar 3.35 *Storyboard stage* jabar *level* sulit

Pada *stage* jabar level sulit musuh yang menghadang bertambah yaitu bagong dan kalong yang berputar di atas seperti pada gambar 3.35.

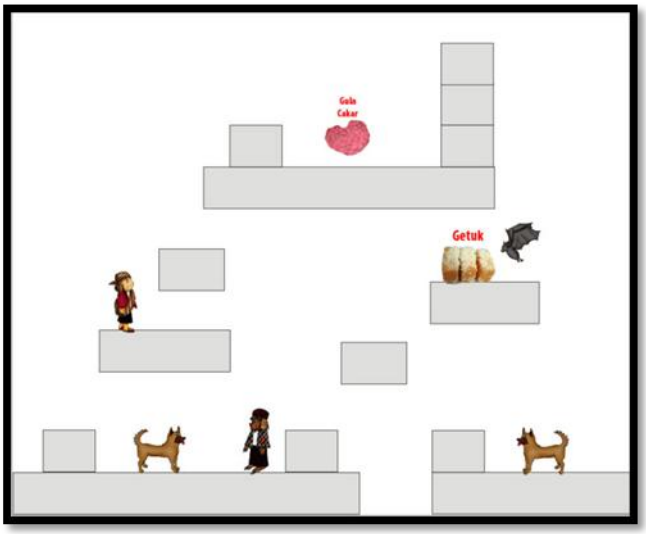
4. *Storyboard stage* jateng *level* mudah



Gambar 3.36 *Storyboard stage* jateng *level* mudah

Pada *stage* ini adalah *stage* ke dua, yaitu *stage* Jawa Tengah, terdapat makanan khas jawa tengah yang tersebar. Musuh pada *stage* ini tetap sama yaitu bebegig seperti pada gambar 3.36.

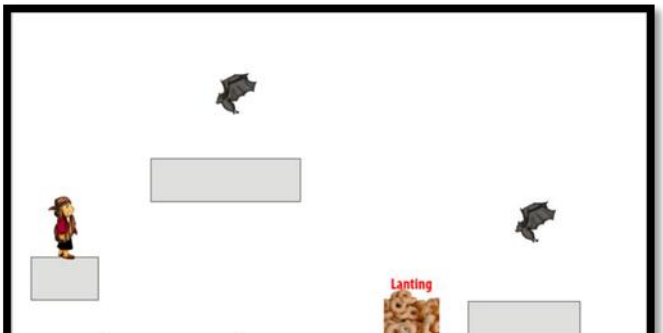
5. *Storyboard stage* jateng *level* sedang



Gambar 3.37 *Storyboard stage* jateng *level* sedang

Pada gambar 3.37 adalah gambar pada *stage* Jawa Tengah dengan tingkatan level sedang, musuh yang menghadang bertambah yaitu guguk dan kalong

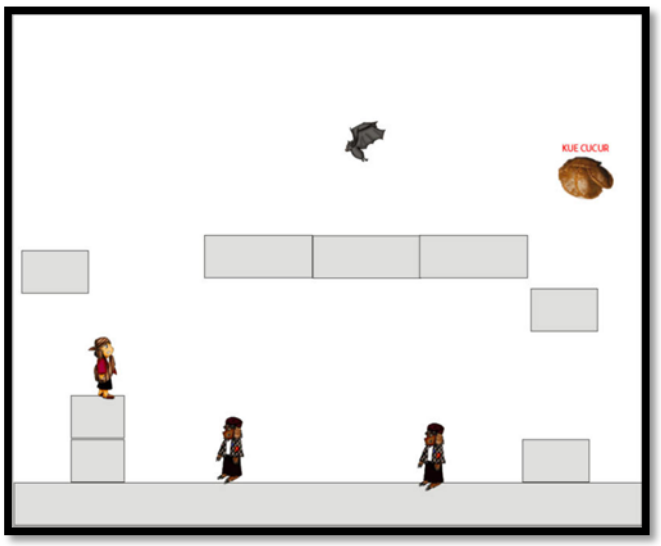
6. *Storyboard stage* jateng *level* sulit



Gambar 3.38 *Storyboard stage* jateng *level* sulit

Pada gambar 3.38 adalah gambar *stage* Jawa Tengah dengan tingkatan level sulit, musuh yang menghadang bertambah yaitu bagong dan kalong yg bertambah banyak sehingga cukup menyulitkan pemain.

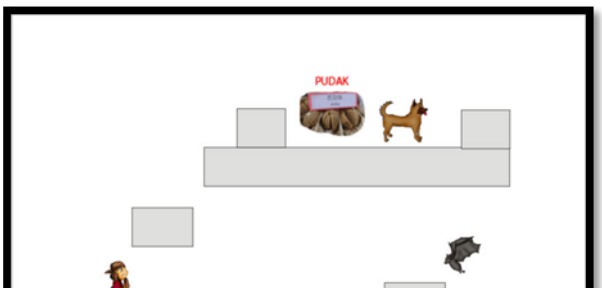
7. *Storyboard stage* jatim *level* mudah



Gambar 3.39 *Storyboard stage* jatim *level* mudah

Pada gambar 3.39 adalah *stage* Jawa Timur dengan tingkat kesulitan mudah, makanan yang tersebar adalah makanan khas Jawa Timur yg harus di dapatkan pemain. Musuh yang menghadang adalah bebegig dan kalong.

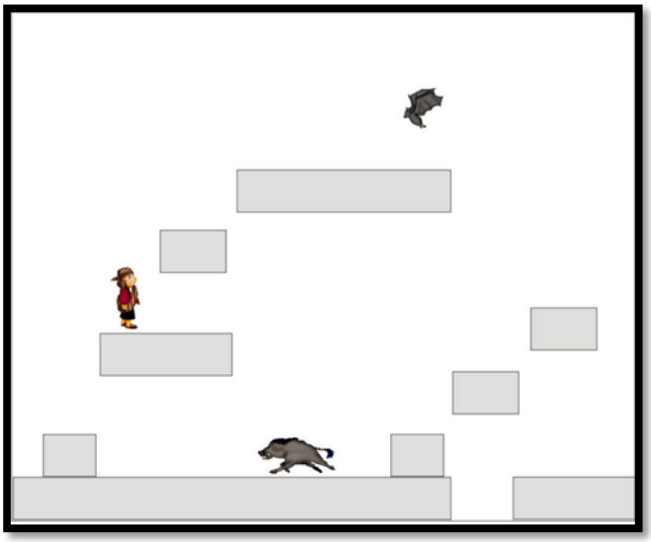
8. *Storyboard stage* jatim *level* sedang



Gambar 3.40 *Storyboard stage* jatim *level* sedang

Pada gambar 3.40 adalah tingkatan level sedang pada *stage* jawa timur, musuh yang menghadang adalah bebegig, kalong, dan guguk, pemain tetap harus mengumpulkan makanan khas Jawa Timur

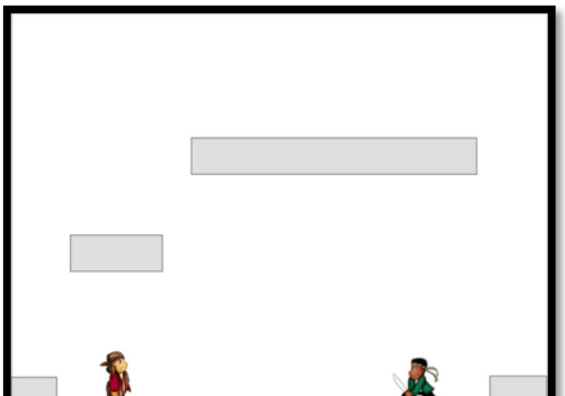
9. *Storyboard stage* jatim *level* sulit



Gambar 3.41 *Storyboard stage* jatim *level* sulit

Pada gambar 3.41 adalah tingkatan level sulit pada stage Jawa Timur. Musuh bertambah yaitu bagong. Kabayan harus mengumpulkan makanan tradisional untuk menuju ke *stage* bos.

10. *Storyboard stage* bos terakhir

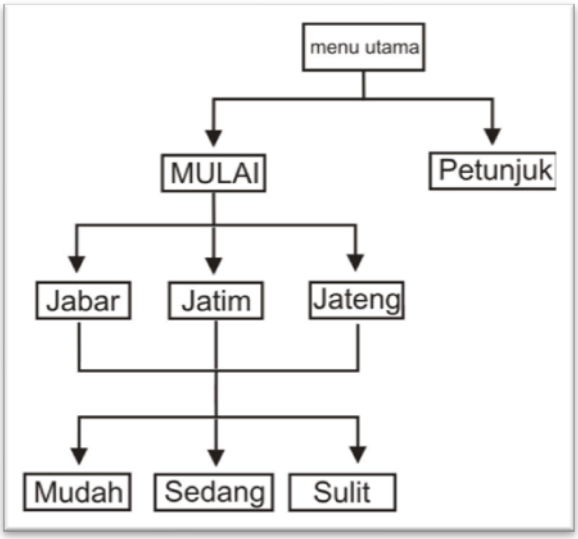


Gambar 3.42 *Storyboard stage* bos terakhir

Pada gambar 3.42 adalah gambar pada *stage* bos terakhir yaitu rewog yang harus dikalahkan oleh kabayan untuk memenangkan *game* ini.

3.2.2 Perancangan Struktur Menu

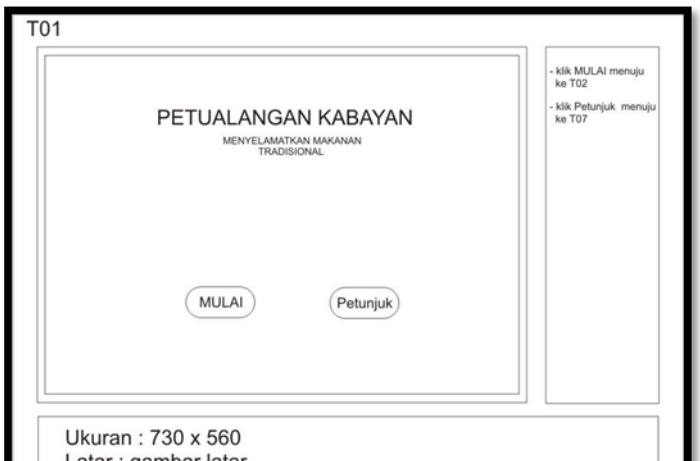
Berikut dapat dilihat perancangan struktur menu dari *game*Petualangan Kabayan Menyelamatkan Makanan Tradisional dapat dilihat pada gambar 3.43 :



Gambar 3.43 Stuktur Menu

3.2.3 Antar Muka Menu Utama

Berikut dapat dilihat perancangan antarmuka menu utama dari *game*Petualangan Kabayan Menyelamatkan Makanan Tradisional dapat dilihat pada gambar 3.44 :



Gambar 3.44 Antar Muka Menu Utama

3.2.4 Antar Muka Tampil Petunjuk

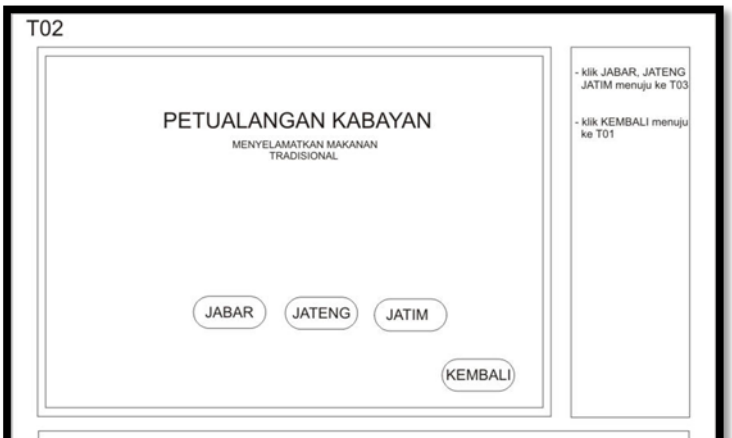
Berikut dapat dilihat perancangan antarmuka petunjuk dari *game* Petualangan Kabayan Menyelamatkan Makanan Tradisional dapat dilihat pada gambar 3.45.



Gambar 3.45 Antar Muka Petunjuk

3.2.5 Antar Muka Pilih *Stage*

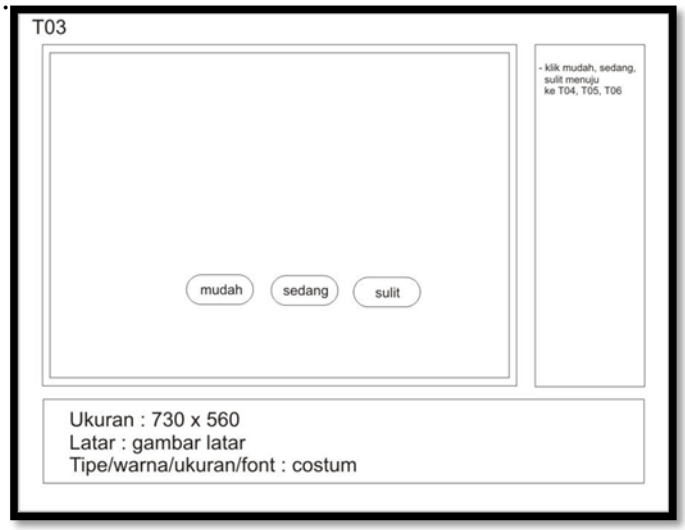
Berikut dapat dilihat perancangan antarmuka pilih *Stage* dari *game*Petualangan Kabayan Menyelamatkan Makanan Tradisional dapat dilihat pada gambar 3.46



Gambar 3.46 Antar Muka Pilih *Stage*

3.2.6 Antar Muka Pilih *Level*

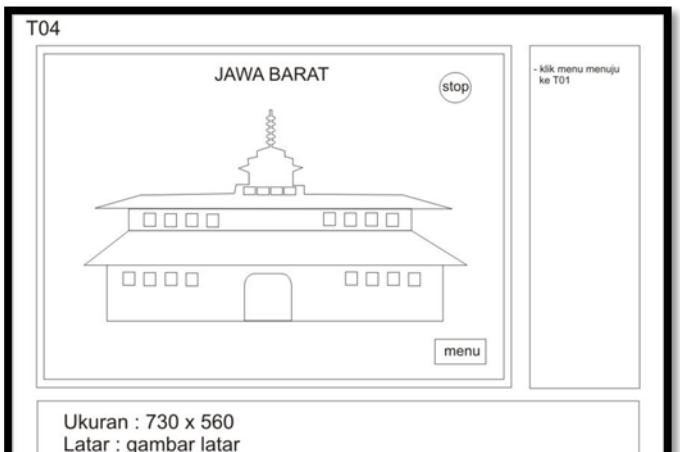
Berikut dapat dilihat perancangan antarmuka pilih *level* dari *game*Petualangan Kabayan Menyelamatkan Makanan Tradisional dapat dilihat pada gambar 3.47.



Gambar 3.47 Antar Muka Pilih *Level*

3.2.7 Antar Muka *Stage* Jawa Barat

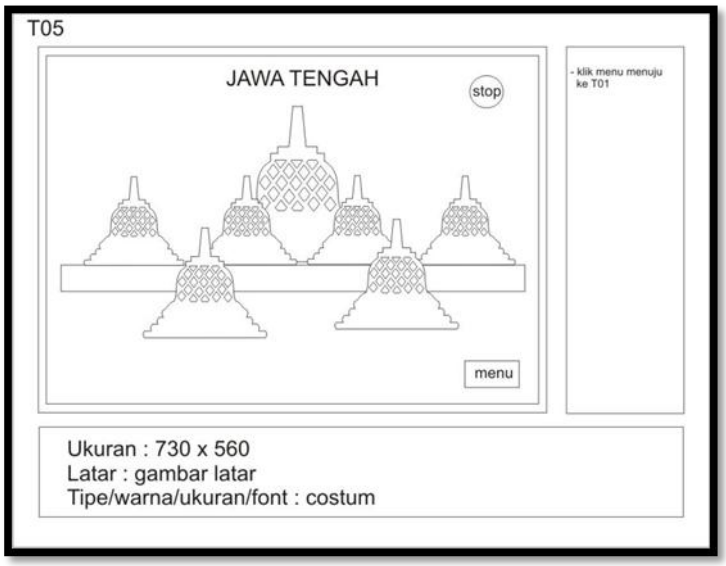
Berikut dapat dilihat perancangan antarmuka *stage* Jawa Barat dari *game* Petualangan Kabayan Menyelamatkan Makanan Tradisional dapat dilihat pada gambar 3.48.



Gambar 3.48 Antar Muka *Stage*Jawa Barat

3.2.8 Antar Muka *Stage* Jawa Tengah

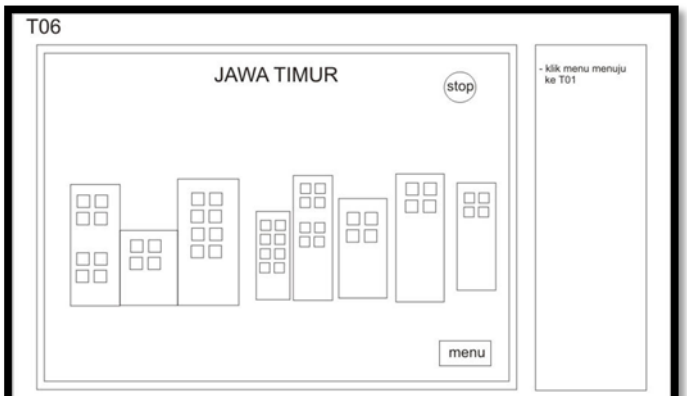
Berikut dapat dilihat perancangan antarmuka *stage* Jawa Tengah dari *game* Petualangan Kabayan Menyelamatkan Makanan Tradisional dapat dilihat pada gambar 3.49.



Gambar 3.49 Antar Muka *Stage* Jawa Tengah

3.2.9 Antar Muka *Stage* Jawa Timur

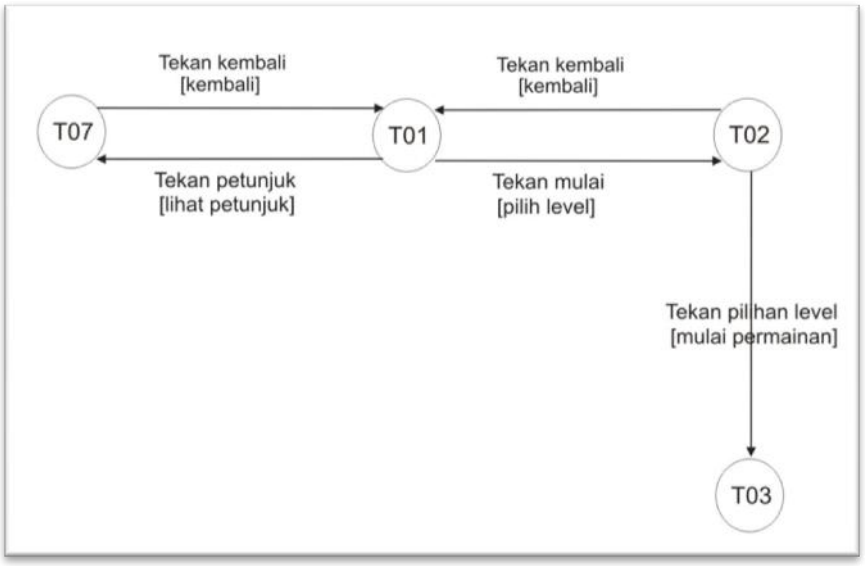
Berikut dapat dilihat perancangan antarmuka *stage* Jawa Timur dari *game* Petualangan Kabayan Menyelamatkan Makanan Tradisional dapat dilihat pada gambar 3.50.



Gambar 3.50 Antar Muka *Stage* Jawa Timur

3.2.10 Jaringan Semantik

Tahap selanjutnya adalah menggambarkan hubungan antar menu pada aplikasi *game* Petualangan Kabayan Menyelamatkan Makanan Tradisional. Penggmbaran tersebut diwujudkan melalui jaringan semantik seperti gambar 3.51.



Gambar 3.51 Jaringan Semantik

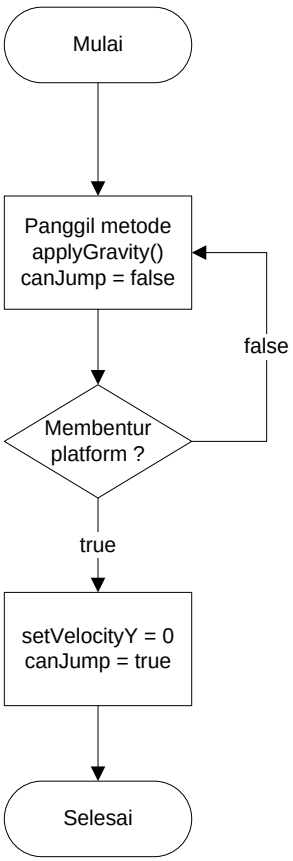
3.2.11 Perancangan *Method*

Perancangan komponen *method* merupakan perancangan yang dibuat setelah perancangan arsitektur menu, antarmuka, pesan, dan jaringan semantik. Perancangan ini berfungsi untuk mendeskripsikan *method-method* yang berada di dalam aplikasi permainan Petualangan kabayan menyelamatkan makanan

tradisional. Adapun *method-method* yang terdapat dalam aplikasi permainan Petualangan kabayan menyelamatkan makanan tradisional. yang akan dibangun adalah sebagai berikut :

1. **Perancangan method *manageJumping***

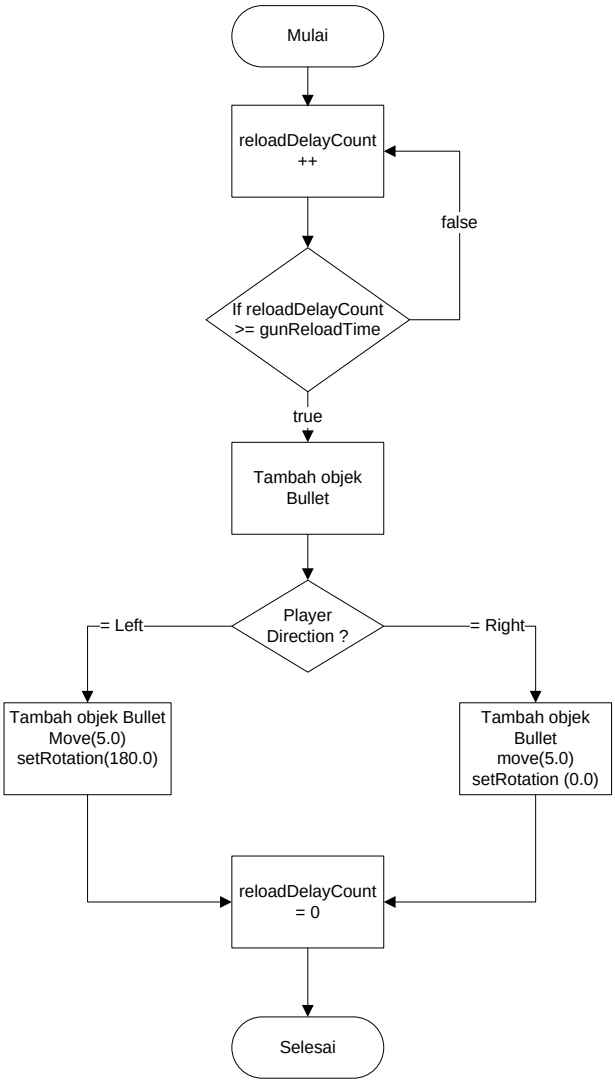
   Metode ini berfungsi untuk penanganan gravitasi terhadap *player*. Untuk lebih jelas dapat dilihat pada gambar 3.52



Gambar 3.52 Perancangan metode *manageJumping*

2. Perancangan metode *fire*

Metode ini berfungsi untuk penanganan senjata pada *player*. Untuk lebih jelas bisa dilihat pada gambar 3.53

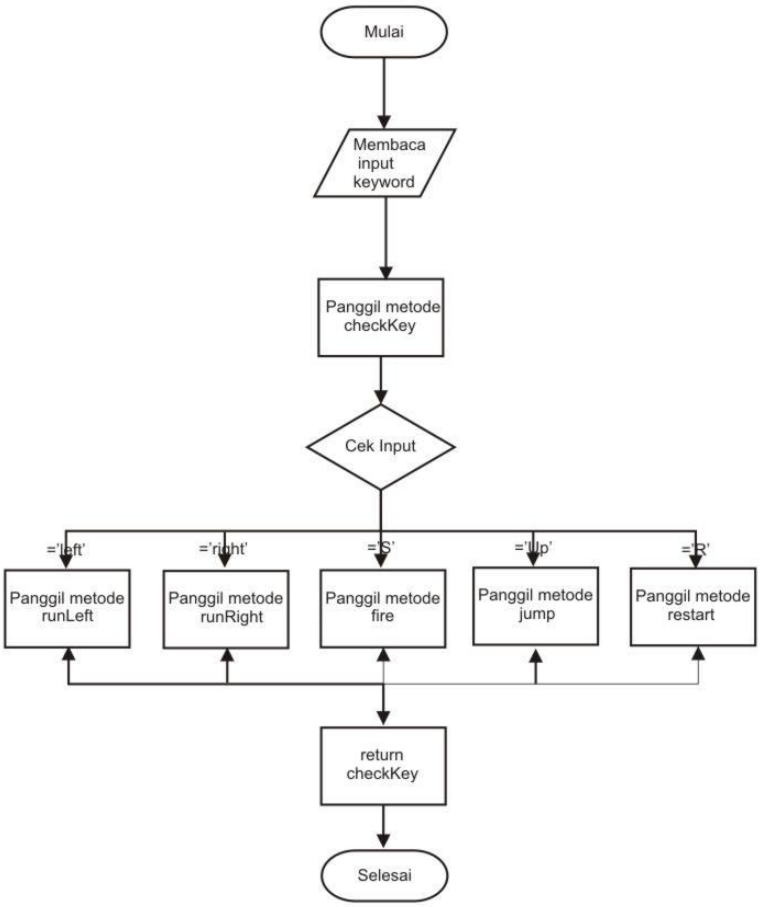


Gambar 3.53 Perancangan metode *fire*

3. Perancangan metode *check Keys*

Metode ini berfungsi untuk penanganan pergerakan pada *player*.

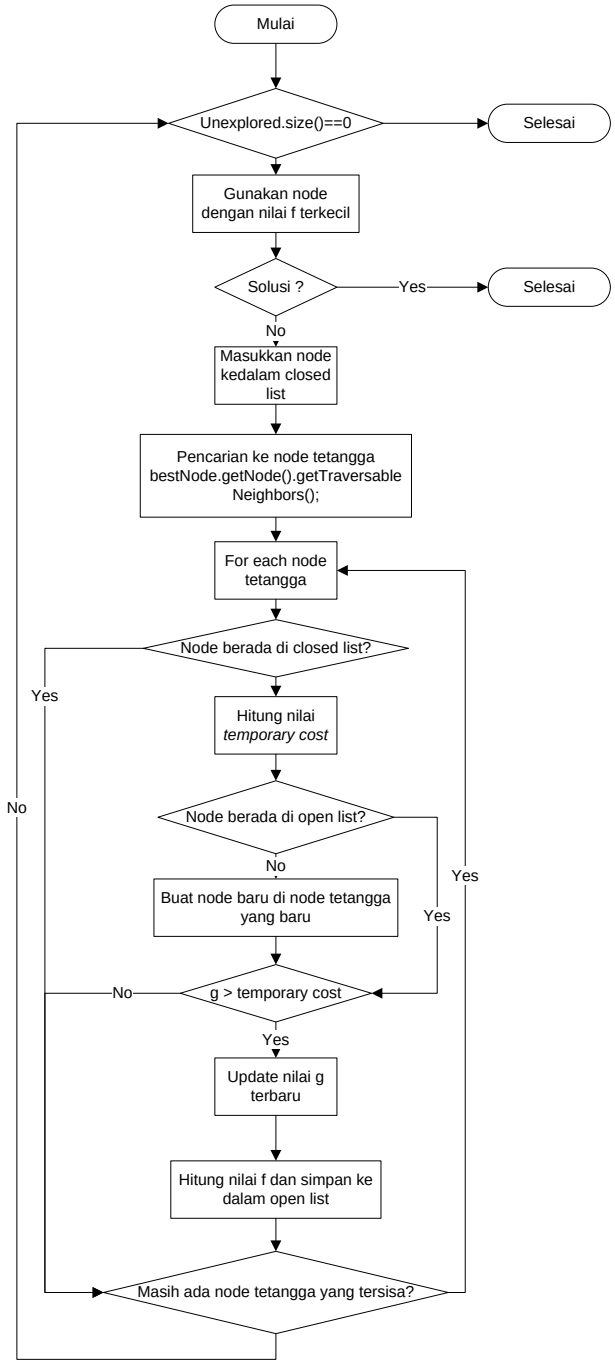
Untuk lebih jelas bisa dilihat pada gambar 3.54



Gambar 3.54 Perancangan metode *check Keys*

4. Perancangan metode *findpath*

Metode ini berfungsi untuk *enemy* menemukan rute untuk mendekati *player*. Untuk lebih jelas bisa dilihat pada gambar 3.55



Gambar 3.55 Perancangan metode *findPath*