BAB 3
ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Sistem

Analisis sistem adalah teknik pemecahan masalah yang menguraikan bagian-bagian komponen dengan mempelajari seberapa bagus bagian-bagian komponen tersebut berkerja dan berinteraksi untuk mencapai tujuan mereka.

3.1.1 Analisis Masalah

1. Menjaga kesehatan dan kebersihan mulut sangat penting

Menjaga kesehatan dan kebersihan mulut perlu dilakukan karena gigi yang sehat tidak mungkin di peroleh begitu saja, perlu perawatan sejak dini agar gigi selalu bersih, kuat, dan sehat. Mengajarkan anak dalam menjaga kesehatan mulut perlu dilakukan karena pada saat masa pergantian dari gigi susu ke gigi permanen sangatlah penting. Hal ini karena gigi permanen yang baru tumbuh masih sangat rentan dan berisiko tinggi untuk berlubang. Jika gigi permanen rusak, selamanya akan rusak karena tidak ada lagi gigi pengganti. Praktek melakukan penjagaan kebersihan dan kesehatan mulut harus dilakukan dengan benar, jika praktek kebersihan mulut dilakukan secara tidak benar dapat menimbulkan penyakit pada mulut.

2. Dampak dari kesehatan mulut

Dampak dari tidak menjaga kesehatan mulut dapat menimbulkan berbagai macam penyakit mulut seperti bau mulut/*halitosis*, gigi berlubang, penyakit gusi/ *periodontitis* dan penyakit mulut lainnya. Penyakit dalam mulut sering kali dianggap penyakit biasa oleh banyak orang, padahal penyakit gusi/*periodontitis* bila tidak ditangani dapat menyebabkan timbulnya penyakit jantung dan kanker leher dan kepala. Oleh karena itu anak-anak perlu belajar tentang betapa pentingnya menjaga kesehatan mulut sejak dini.

3. Anak-anak sukar dalam menjaga kesehatan mulut

Mengajarkan anak tentang pentingnya menjaga kesehatan mulut dan gigi bukanlah hal yang mudah, diperlukan sebuah media pengajaran yang menyenangkan

dan mudah dipelajari untuk anak-anak. Anak-anak memiliki kecendurangan lebih suka bermain daripada belajar, oleh karena itu *game* edukasi bisa menjadi sebuah media yang tepat untuk mengajarkan anak-anak tentang betapa pentingnya menjaga kesehatan mulut. Dengan *game* edukasi anak-anak akan bermain sambil belajar, di selain itu juga *game* dapat membantu koordinasi visual motorik, mengembangkan visual motorik halus, dan mendorong anak mencapai target tertentu. *Video games* juga membentuk sikap sportif, melatih daya ingat, konsentrasi, kemampuan berlogika dan menyelasaikan masalah. Namun video *games* juga bisa memberi efek buruk, bila dikonsumsi berlebihan dan mengandung materi yang tidak sesuai dengan usianya. Maka dari perancangan game "Clean On Your Mouth" unsur kekerasan dalam *game* tidak akan terlalu dimunculkan, untuk menghindari anak-anak yang cenderung suka mempraktikan atau meniru sikap-sikap kekerasan yang ada dalam *video games*.

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 dibangun. 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*



Gambar 3.1. Game 2D Kabayan Saving Our Cultures

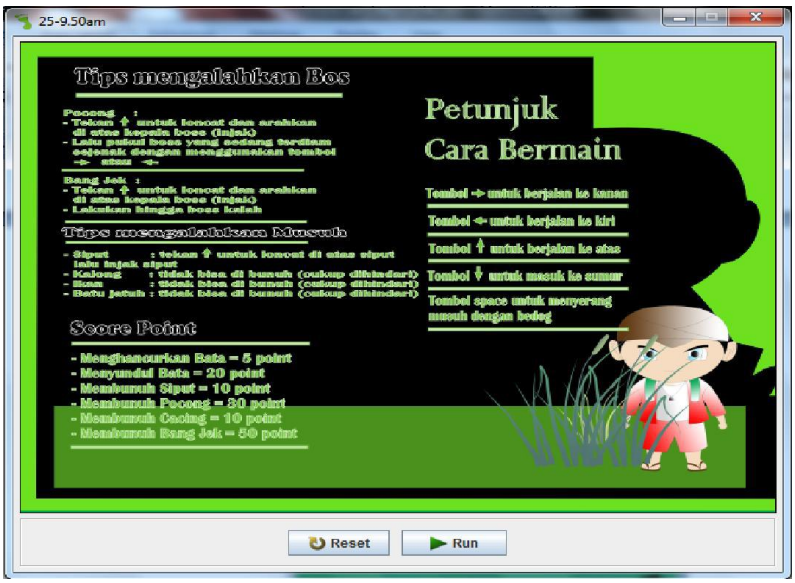
Game kabayan *Saving Our Cultures* menceritakan perjalanan seorang pemuda dari suku Sunda yaitu Kabayan yang harus bertualang untuk menyelamatkan alat musik tradisional yaitu angklung yang telah dicuri oleh seorang penjahat dari Negara Misterius yang bernama Juragan Asep.

Kelebihan

a. Musuh sudah memiliki kecerdasan buatan yang akan membuat permainan semakin menarik

Kekurangan

a. Sisi kebudayaan yang diangkat belum maksimal, hanya sebatas alat musik calung dan golok saja.
b. Unsur edukasi masih sangat kurang, tidak memberi penjelasan tentang alat golok dan calung sebagai alat music dan senjata tradisional
c. Tidak memiliki fitur *save game.*
d. Petunjuk cara bermain terlalu rumit karena hanya menampilkan text, tidak sesuai dengan target user 5-12 tahun, padahal di umur 5-7 tahun masih ada pengguna yang masih belum bisa membaca



Gambar 3.2 Petunjuk cara bermain *Game* 2D Kabayan *Saving Our Cultures*

2. *Game* Sejarah Kemerdekaan Indonesia

   Merupakan game edukasi sejarah kemerdekaan Indonesia yang menceritakan perjalanan seorang anak SMP bernama Dicky. Pada saat liburan sekolah dia berlibur kebeberapa tempat, yaitu rumah kakek, bibi dan paman. Di setiap rumah yang akan dikunjungi Dicky akan disajikan beberapa pertanyaan sejarah tentang kemerdekaan Indonesia.



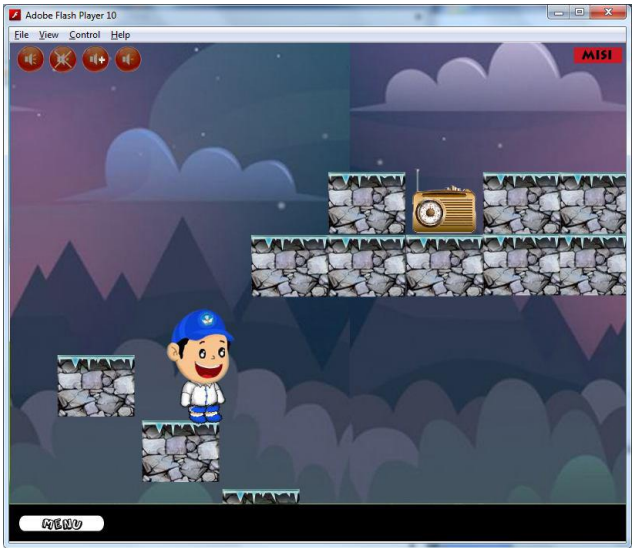
Gambar 3.3 Game 2D Sejarah Kemerdekaan Indonesia

Kelebihan

a. Penyampaian edukasi menarik, yaitu dengan game adventure dan mencari item untuk mendapatkan pertanyaan sejarah

Kekurangan

a. Tidak ada musuh yang harus dilawan, padahal game ini termasuk *adventure games,* sehingga *game* akan lebih cenderung bosan untuk dimainkan karena tidak memiliki tantangan.



Gambar 3.4 *Gameplay Game* 2D Sejarah Kemerdekaan Indonesia

b. Tidak menerapkan kecerdasan buatan
c. Tidak memiliki fitur *save game.*
d. Tidak ada *reward* / hadiah ketika pemain berhasil memenangkan permainan

3. Game War History Kingdom

Game war history kingdom merupakan permainan yang menceritakan tentang 3 kerajaan besar di Indonesia, yaitu Kerajaan Pajajaran, Kerajaan Majapahit dan Kerajaan Sriwijaya



Gambar 3.5 Menu utama game War History Kingdom

Kelebihan

a. Penyampaian edukasi sangat menarik, yaitu dengan cara menyisipkannya kedalam *intro* dalam *game.*



Gambar 3.6 Intro game War History Kingdom

b. Desain *background* dan karakter sangat menarik

Kekurangan

a. *Gameplay* kurang menarik, tidak sesuai dengan judul. Dalam permainan ini pemain akan bertemu dengan musuh lalu diberikan beberapa macam pertanyaan atau misi mencari item tertentu yang harus dicari. Tidak ada pertarungan / perang dengan musuh.
b. Tidak memilikki fitur *save game.*
c. Musuh tidak memiliki kecerdasan buatan.

3.1.3 Analisis *Game* yang Akan Dikembangkan

3.1.3.1 *Story Line*

Game Clean On Your Mouth ini, bercerita tentang seorang pahlawan yang bernama Super Odolman yang berusaha merebut kembali odol-odol dan buku panduan tentang kebersihan dan kesehatan mulut yang telah dicuri oleh Halitosismon. *Game* ini dimulai ketika para monster yang dipimpin oleh Halitosismon datang ke kota untuk mencuri semua odol dan buku panduan menyikat gigi, agar seluruh manusia terserang

penyakit mulut dan gigi. Lalu Super Odolman berusaha untuk mengambil kembali odol-odol dan buku panduan yang telah dicuri oleh Halitosismon.

3.1.3.2 Tingkat Kesulitan

Game Clean On Your Mouth terdiri dari 3 level, dimana di setiap level memiliki tempat / *stage* yang berbeda-beda, dan setiap level memiliki misi yang harus di selesaikan. Di setiap level dijaga oleh *boss* / raja dan memiliki tingkat kesulitan yang berbeda-beda, seperti makin banyaknya rintangan, jurang dan musuh-musuh akan semakin bertambah cepat dan memiliki tingkat kekebelan yang semakin kuat.

3.1.3.3 *Gameplay*

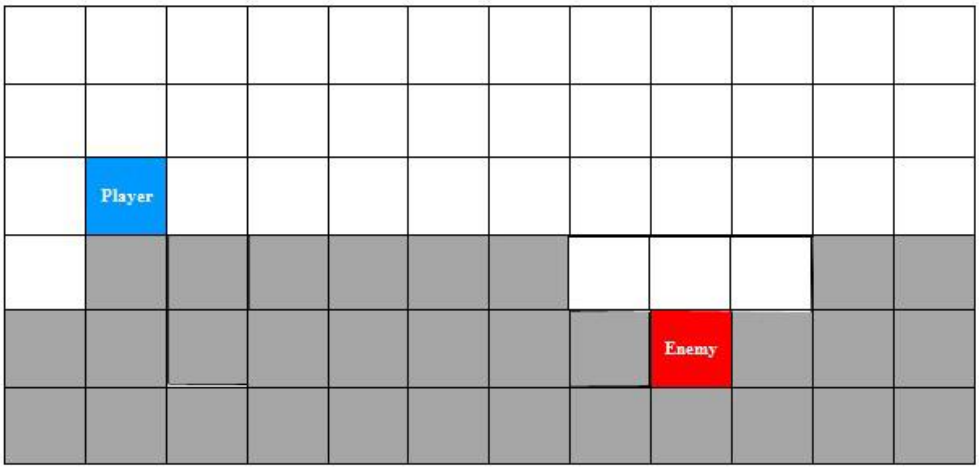
Tugas utama pemain (*player*) dalam memainkan aplikasi ini yaitu melalui level-level, dimana setiap level menyediakan petualangan dengan menampilkan beberapa karakter level tersebut maupun karakter musuh yang harus dilawan. Pemain (*player*) dapat memulai permainan dengan memilih pilihan mulai, dimana nantinya pemain memasuki awal atau pada hal ini level. Di setiap level memiliki rintangan yang berbeda-beda dimana setiap level memiliki tingkat kesulitan yang semakin sulit untuk di lewati, di setiap level juga terdapat *boss* yang harus di kalahkan untuk menuju ke level berikutnya, jika pemain tidak bisa mengalahkan boss pada level tersebut maka pemain harus mengulangnya kembali dan bila pemain mengalahkan boss pada level tersebut maka pemain akan melanjukan ke level berikutnya. *Media Control* yang digunakan pada *game Clean On Your Mouth* menggunakan *keyboard* untuk menggerakan karakter dan *mouse* untuk memilih menu.

3.1.3.4 *Scoring*

Dalam *game* ini untuk menambah *score,* pemaian harus mengumpulkan odol-odol untuk menambah point. Odol-odol tersebar di setiap level, jika pemaian mengambil odol tersebut maka *score*nya akan bertambah. Untuk setiap odol yang diambil *score* akan bertambah 1 poin. Poin tersebut akan habis atau kembali ke keadaan awal jika *user* mati atau meninggalkan permainan.

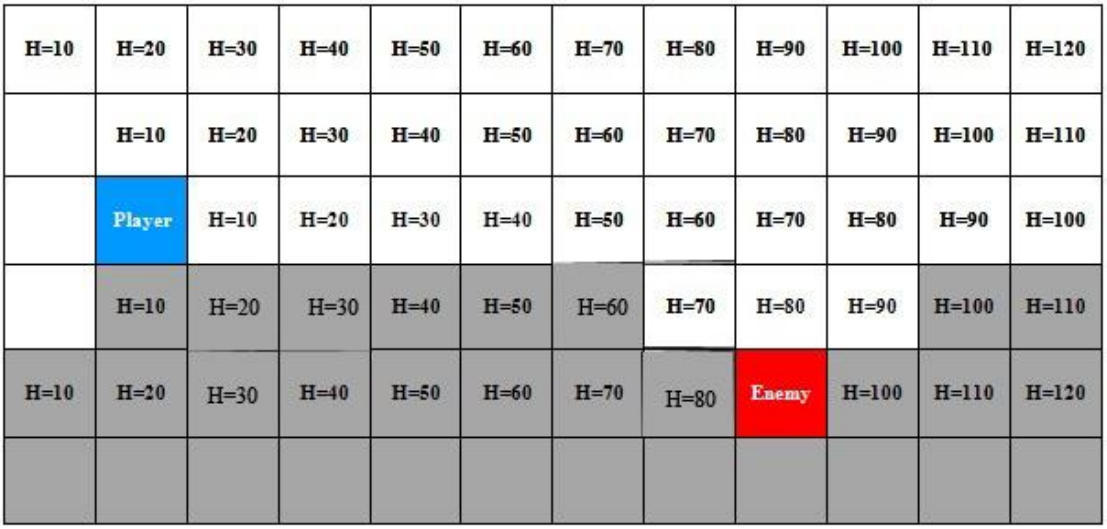
3.1.4 Analisis Algoritma

Algoritma ini melakukan pencarian jalur terpendek dengan biaya yang di perhitungkan dari penjumlahan biaya sebenarnya ditambah dengan biaya perkiraan. Dalam notasi matematika algoritma ini dapat dituliskan dengan *f(n) = g(n)+h(n).* Dengan perhitungan biaya seperti ini, maka algoritma A* (A *Star*) adalah complete dan optimal. Contoh kasus untuk mencari jalur terpendek yang akan di tempuh oleh enemy / NPC (*non player character*), dapat dilihat di gambar 3.7 :



Gambar 3.7 Kasus pencarian jalur terpendek dengan algoritma A* (A Star)

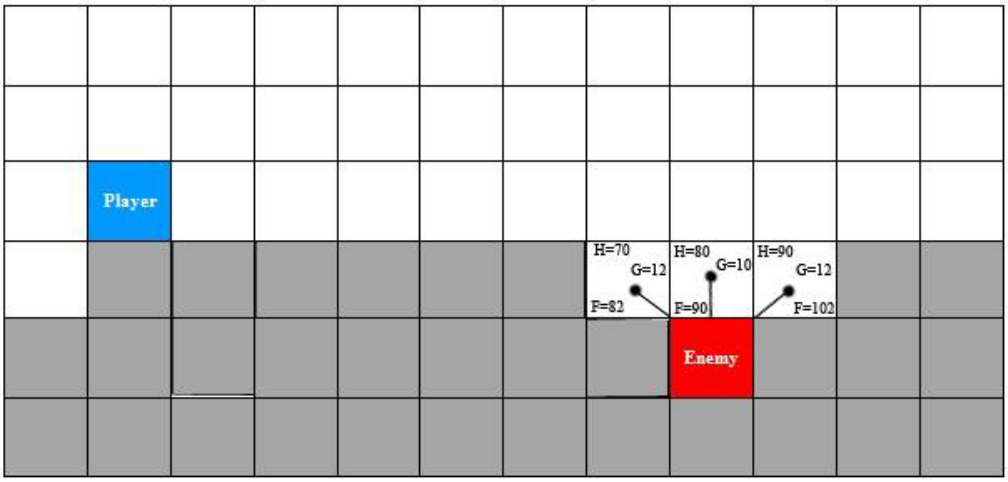
Pada gambar 3.7 dapat dilihat, bahwa node berwarna biru melambangkan *player* yang merupakan node akhir, sedangkan yang berwarna merah melambangkan enemy / NPC (*non player character*) yang merupakan node awal, dan node berwarna abu merupakan penghalang dalam pencarian. Nilai *heuristic* atau *h* nya di dapat dari jarak perkiraan dari node biru / *player* ke node merah / *enemy*. Untuk lebih jelasnya dapat dilihat di gambar 3.8 :



Gambar 3.8 nilai heuristic pada kasus pencarian jalur terpendek

Terdapat 3 nilai yaitu *f, g* dan *h*.Nilai *f* adalah total biaya yang diperhitungkan, nilai *g* adalah biaya sebenarnya, dan nilai *h* adalah biaya perkiraannya. Untuk pergerakan horizontal *cost* nya adalah 10 sedangkan untuk diagonal adalah 12, dan notasi matematika untuk mendapatkan nilai *heuristic diagonal distance* adalah *h_diagonal(n)* = min (abs(n.x-goal.x) + abs(n.y-goal.y)). Berikut ini adalah langkah langkah dalam pencarian jalur terpendek menggunakan algorima A* (A *Star*) yang dapat dilihat pada gambar 3.9 :

Langkah pertama :



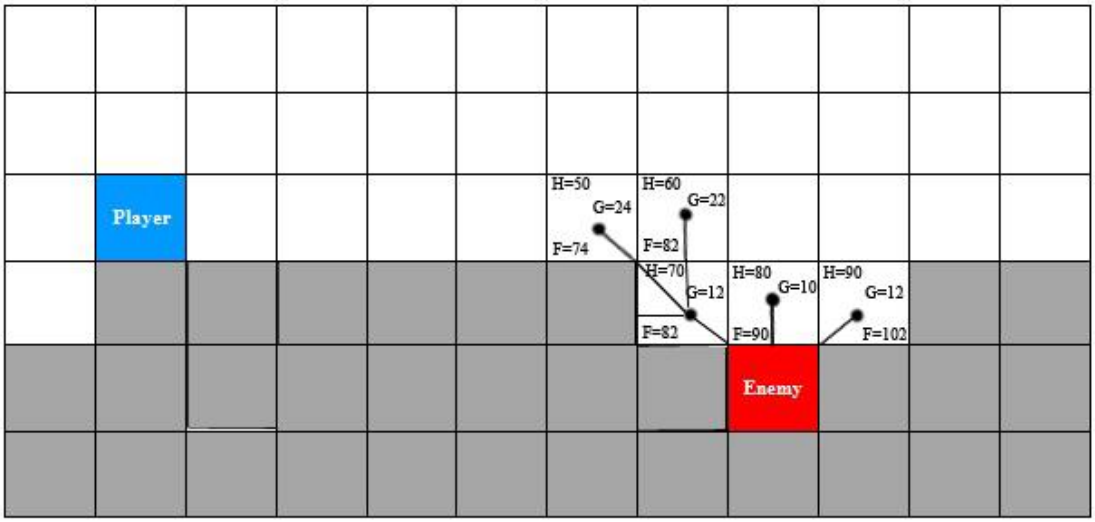
Gambar 3.9 Pencarian jalur terpendek dengan algoritma A* (A Star) 1

Dari node merah / *enemy* akan melakukan pencarian secara horizontal, vertkal, dan diagonal menuju ke arah node berwarna biru / *player*, langkahnya sebagai berikut :

1. Vertikal atas, nilai $g = 10$, $h = 80$, dan $f = g + h = 90$
2. Diagonal kanan atas, nilai $g = 12$, $h = 90$, dan $f = g + h = 102$
3. Diagonal kiri atas, nilai $g = 12$, $h = 70$, dan $f = g + h = 82$

Langkah yang di ambil diagonal kiri atas karena memiliki nilai cost yang paling murah, $f = 82$

Langkah Kedua :



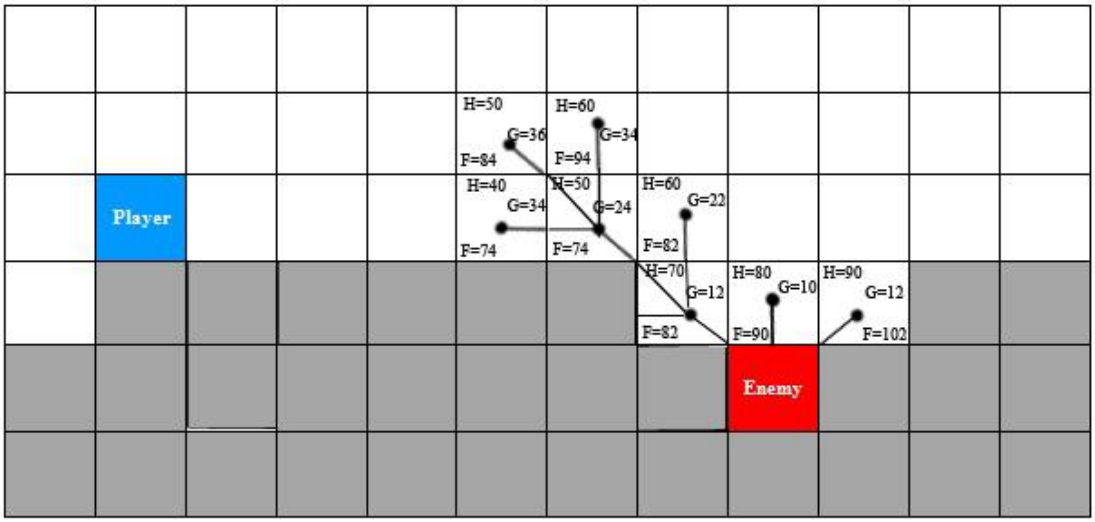
Gambar 3.10 Pencarian jalur terpendek dengan algoritma A* (A *Star*) 2

langkahnya sebagai berikut (nilai *g* akan di tambah nilai *g* di node sebelumnya *g* = 12):

1. Vertikal atas nilai *g* = 22, *h* =60, dan *f* = *g* + *h* = 82
2. Diagonal kiri atas nilai *g* = 24, *h* =50, dan *f* = *g* + *h* = 74

Langkah yang di ambil diagonal kiri atas karena memiliki nilai cost yang paling murah, *f* = 74

Langkah ketiga :



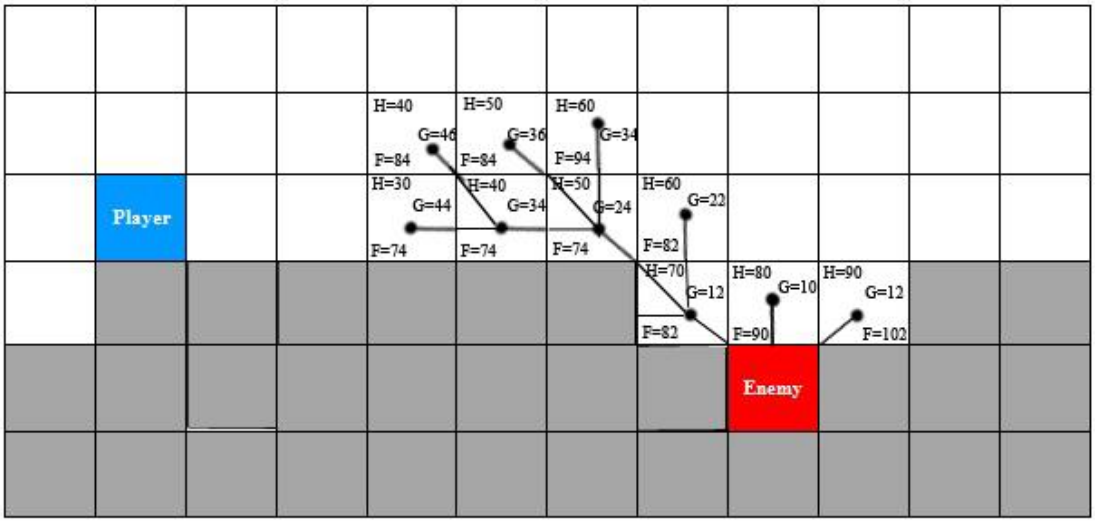
Gambar 3.11 Pencarian jalur terpendek dengan algoritma A* (A *Star*) 3

langkahnya sebagai berikut (nilai *g* akan di tambah nilai *g* di node sebelumnya *g* = 24):

1. Horizontal kiri, nilai *g* = 34, *h* = 40, dan *f* = *g* + *h* = 74
2. Vertikal atas, nilai g = 34, h = 60, dan f = g + h =94
3. Diagonal kiri atas g = 36, h = 50, dan f = g + h = 84

Langkah yang di ambil diagonal kiri atas karena memiliki nilai cost yang paling murah, *f* = 74

Langkah 4 :



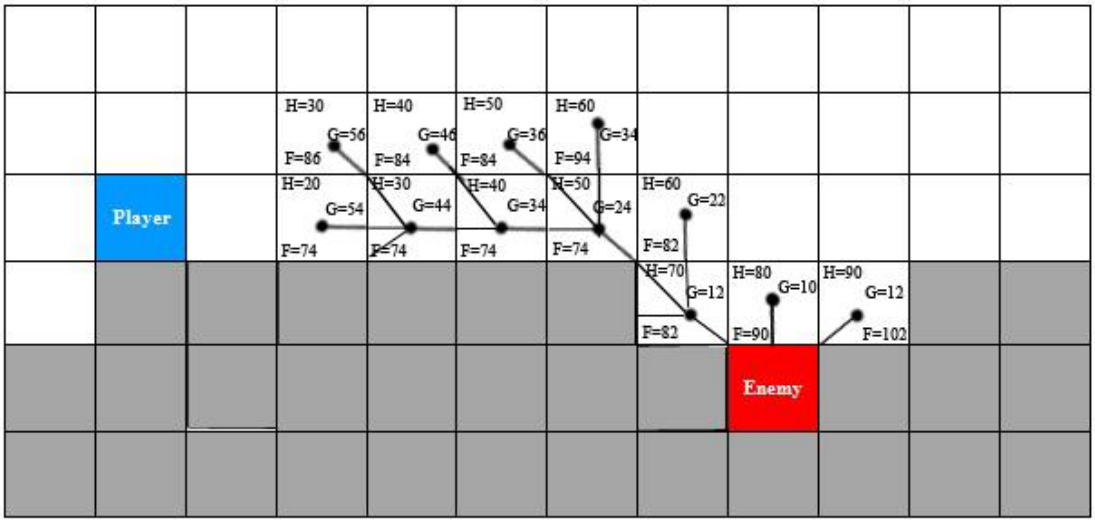
Gambar 3.12 Pencarian jalur terpendek dengan algoritma A* (A *Star*) 4

langkahnya sebagai berikut (nilai *g* akan di tambah nilai *g* di node sebelumnya *g* = 34):

1. Horizontal kiri, nilai *g* = 44, *h* =30, dan *f* = *g* + *h* = 74
2. Diagonal kiri atas g = 44, h = 40, dan f = g + h = 84

Langkah yang di ambil horizontal kiri atas karena memiliki nilai cost yang paling murah, *f* = 74

Langkah 5 :



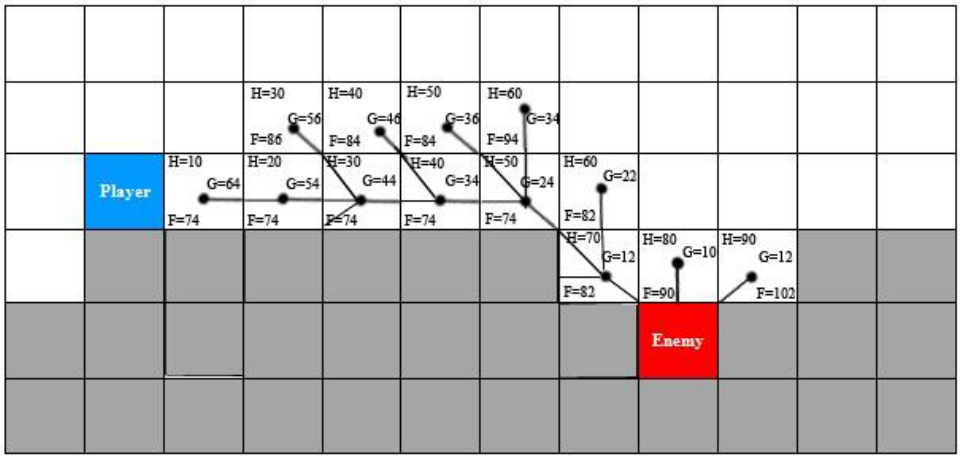
Gambar 3.13 Pencarian jalur terpendek dengan algoritma A* (A *Star*) 5

langkahnya sebagai berikut (nilai *g* akan di tambah nilai *g* di node sebelumnya *g* = 44):

1. Horizontal kiri, nilai *g* = 54, *h* =20, dan *f* = *g* + *h* = 74
2. Diagonal kiri atas, nilai g = 30, h = 56 dan f = g+ 5 = 86

Langkah yang di ambil horizontal kiri atas karena memiliki nilai cost yang paling murah, *f* = 74

Langkah 6 :



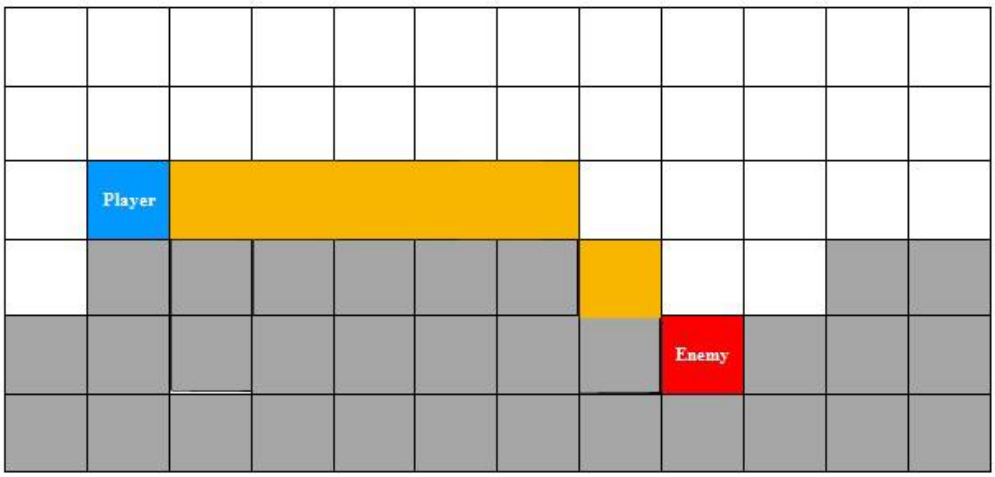
Gambar 3.14 Pencarian jalur terpendek dengan algoritma A* (A *Star*) 5

langkahnya sebagai berikut (nilai *g* akan di tambah nilai *g* di node sebelumnya *g* = 54):

1. Horizontal kiri, nilai *g* = 64, *h* =10, dan *f* = *g* + *h* = 74

Langkah yang di ambil horizontal kiri atas karena memiliki nilai cost yang paling murah, *f* = 74

Dari node berwarna merah / *enemy* menuju node berwarna biru / *player* maka di dapatkan lah jalur terpendek dengan menggunakan algoritma A* (A Star) sebagai berikut :



Gambar 3.15 Hasil pencarian jalur terpendek dengan algoritma A* (A *Star*) 5

1. Dari node berwarna merah / *enemy* mengambil jalur ke arah node diagonal kiri atas dengan nilai *f* = 82
2. Dari node diagonal kiri atas dengan nilai *f* = 82, *enemy* akan mengambil jalur ke diagonal kiri atas dengan nilai *f=74*
3. Dari node diagonal kiri atas dengan nilai *f* = 74, *enemy* akan mengambil jalur ke horizontal kiri dengan nilai *f* = 74
4. Dari node diagonal kiri atas dengan nilai *f* = 74, *enemy* akan mengambil jalur ke horizontal kiri dengan nilai *f* = 74
5. Dari node diagonal kiri atas dengan nilai *f* = 74, *enemy* akan mengambil jalur ke horizontal kiri dengan nilai *f* = 74
6. Dari node diagonal kiri atas dengan nilai *f* = 74, *enemy* akan mengambil jalur ke horizontal kiri dengan nilai *f* = 74 dan sampailah di node berwarna biru / *player*

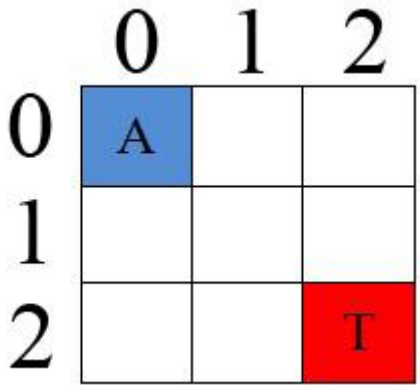
Untuk menghitung nilai heuristic diagonal distance bisa dilihat di kasus dibawah ini berikut ini adalah penjelasannya

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

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

h(n) = h_diagonal(n) + (h_straight(n) – 2*h)_diagonal(n)))

bahwa ordo dapat disesuaikan dengan rentang X + Y , salah satu contoh perhitungan pada game *pathfinding* ini akan di jelaskan dengan ordo minimal yaitu 3 x 3 di karenakan perhitungan pada ordo berapa pun hasilnya akan sama. Pada penentuan bobot setiap node akan diberikan nilai sesuai jarak terdeket ke tujuan missal node yang terjauh dari tujuan maka di beri bobot kecil sedangkan node yang terdekat dengan tujuan di beri bobot lebih besar. Perhitungan yang dilakukan dengan algoritma A* (A *Star*), dengan kondisi tanpa penghalang dapat di lihat di gambar 3.10



Gambar 3.16 Contoh Kondisi Ruang Keadaan

Keterangan :

A : Simpul Awal

T : Simpul Tujuan

Posisi simpul awal = Ax : 0, Ay : 0

Posisi simpul tujuan = goal x : 2, goal y : 2

Langkah Pertama :

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

h_ straight (1,1) = (abs(1-2) + abs(1-2))

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

= 2

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

h(1,1) = -2 + (2+4)

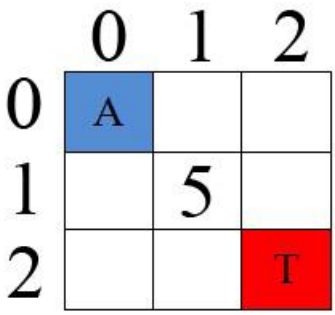
= 4

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

= 1+4

= 5

Dari hasil perhitungan heuristic diagonal pertama hasilnya 5, seperti pada gambar 3.11 :



Gambar 3.17 Hasil perhitungan heuristic diagonal distance tahap 1

Langkah kedua :

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

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

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

= 0

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

h_ straight (2,2) = (abs(2-2) + abs(2-2))

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

= 0

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

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

= 0 + (0 - 0)

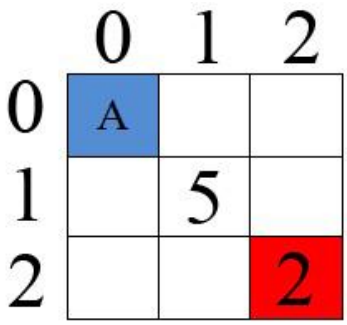
=0

f(2,2) = g(1,1) + h(1,1)

= 2+0

= 2

Dari hasil perhitungan heuristic diagonal pertama hasilnya2, seperti pada gambar 3.12



Gambar 3.18 Hasil perhitungan heuristic diagonal distance tahap 2

3.1.5 Analisis Kebutuhan Non Fungsional

Analisis kebutuhan non fungsional biasanya dilakukan untuk mengetahui spesifikasi kebutuhan untuk sebuah sistem. Dan Spesifikasi kebutuhan biasanya melibatkan analisis perangkat keras/*hardware*, analisis perangkat lunak/*software*, dan juga analisis pengguna/*user*.

3.1.5.1 Analisis Kebutuhan Perangkat Keras

Perangkat keras yang dibutuhkan untuk menjalankan aplikasi *game Clean On Your Mouth* menggunakan spesifikasi sebagai berikut :

Tabel 3.1 Spesifikasi Kebutuhan Perangkat Keras

Nama Perangkat	Spesifikasi
Prosesor	1.00 Ghz
RAM	512 MB
Harddisk	10 GB
Vga	128 MB
Display	Resolusi 800 x 600 pixels
Keyboard	Standar
Mouse	Standar
Speaker	Standar

3.1.5.2 Analisis Kebutuhan Perangkat Lunak

Perangkat lunak yang dipergunakan dalam pembangunan aplikasi *game* yang akan dibangun sebagai berikut :

Tabel 3.2 Spesifikasi Kebutuhan Perangkat Lunak

Nama perangkat lunak	Jenis
Sistem operasi	Windows, Linux, dan Macintosh
IDE	JDK 1.6

3.1.5.3 Analisis Pengguna

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

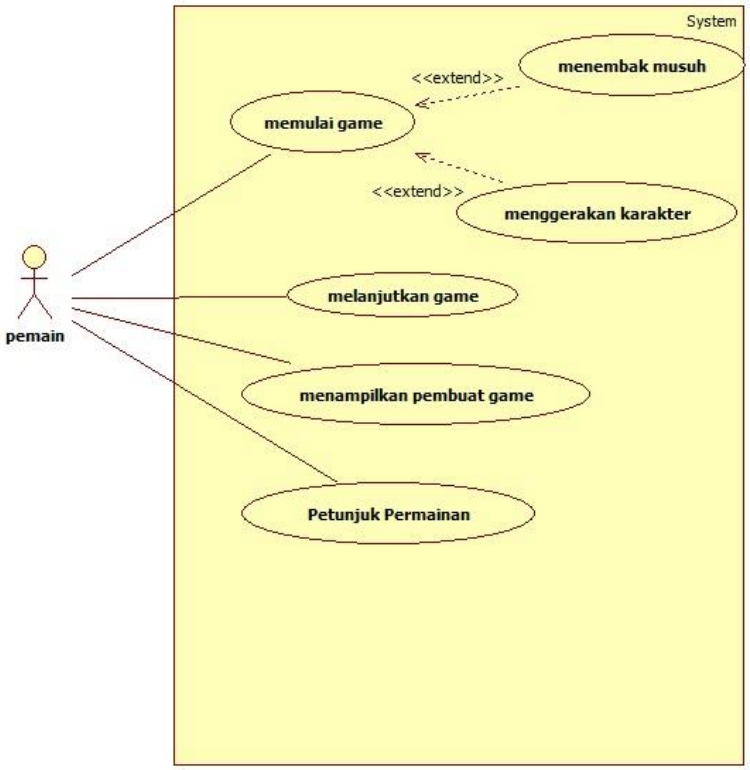
Tabel 3.3 Karakteristik Pengguna

Tingkat pengetahuan dan pengalaman	
Kecakapan menggunakan komputer	Pengalaman menggunakan dekstop komputer / notebook
Pengalaman sistem	Pernah bermain *pc game*
Tingkat baca	Sudah bisa membaca
Kemampuan mengetik	Tidak diperlukan
Fisik	Tidak memiliki kelainan (buta warna, tuna netra, tuna rungu dan cacat fisik)
Bahasa	Bahasa Indonesia dan bahasa Inggris

3.1.6 Analisis Kebutuhan Fungsional

3.1.6.1 *Use Case Diagram*

Berikut ini adalah rancangan dari *use case* diagram *game clean on your mouth. Use case* diagram memberikan gambaran tentang bagaimana pengguna berinteraksi dengan sistem.



Gambar 3.19 *Use Case* Diagram *Game*

3.1.6.1.1 Identifikasi *Aktor*

Definisi aktor berfungsi untuk menjelaskan peranan aktor dalam pengoperasian game. Dalam *game* ini, hanya terdapat seorang aktor yaitu pemain. Berikut ini adalah definisi dari aktor yang ada dalam perancangan *game*.

Tabel 3.4 Definisi Aktor

No	Aktor	Deskripsi
1.	Pemain	Pemain merupakan orang yang memainkan *game* dan berinteraksi langsung dengan antar muka *game.*

3.1.6.1.2 Definisi *Use Case*

Definisi *Use Case* berfungsi untuk menjelaskan proses yang terdapat pada setiap *use case.*

Tabel 3.5 Definisi *Use Case*

No	*Use Case*	Deskripsi
1.	Memulai *game*	Pada use case ini, aktor bisa memulai game. Memulai game memiliki arti mulai permain baru sehingga ketika button memulia game di klik maka akan ditampilkan *stage* pertama
2.	Melanjutkan *game*	Melanjutkan game merupakan kemampuan yang diberikan kepada aktor untuk meneruskan game yang tersimpan. Meneruskan dalam hal ini berarti meneruskan dari stage terakhir yang telah dimainkan oleh pemain.
3.	Menggerakan karakter	Memindahkan karakter dengan cara menekan papan tombol.
4.	Menembak musuh	Odolman memiliki kemampuan untuk menembak musuh. Odolman akan mengeluarkan tembakan ketika pengguna menekan tombol tertentu pada keyboard
5.	Petunjuk permainan	Menampilkan tentang cara bermain *game*.
6.	Menampilkan pembuat *game*	Menampilkan tentang pembuat *game*.

3.1.6.1.3 Skenario *Use Case*

Tabel 3.6 *Use Case Scenario* Memulai *Game*

Identifikasi Masalah	
Nama	**Memulai *game***
Tujuan	Menampilkan *permainan clean on your mouth*
Deskripsi	Memilih menu mulai pada menu permainan untuk memulai permainan
Aktor	Pemain
Skenario Utama	
Kondisi Awal	Tampilkan menu mulai pada game
Aksi Aktor	**Reaksi Sistem**
1.Memilih menu mulai game	2.Menampilkan intro 1
3.Menekan tombol next	4.Menampilkan level 1
5.Menyelesaikan level 1	6.Menampilkan intro 2
7.Menekan tombol next	8.Menampilkan level 2
9. Menyelesaikan level 2	10.Menampilkan intro 3
11.Menekan tombol next	12.Menampilkan level 3
13. Menyelesaikan level 3	14.Menampilkan intro 4
15.Menekan tombol next	15.Menampilkan menu utama

Tabel 3.7 *Use Case Scenario* Menembak

Identifikasi Masalah	
Nama	Menembak musuh
Tujuan	Mengeluarkan tembakan
Deskripsi	Mengeluarkan tembakan untuk menyerang musuh

Aktor	Pemain
Skenario Utama	
Kondisi Awal	Tampilan permainan
Aksi Aktor	**Reaksi Sistem**
1.menekan tombol “f” pada keyboard	2. mengeluarkan tembakan
Kondisi akhir	Karakter mengeluarkan tembakan

Tabel 3.8 *Use Case Scenario* Menggerakan Karakter

Identifikasi Masalah	
Nama	Menggerakan karakter
Tujuan	Menggerakan karakter, Menghindari musuh
Deskripsi	Menggerakan karakter untuk memainkan permainan dan menghindarai serangan musuh
Aktor	Pemain
Skenario Utama	
Kondisi Awal	Tampilan permainan
Aksi Aktor	**Reaksi Sistem**
1.Menekan tombol ← pada *keyboard*	2.Karakter bergerak ke kiri
3.Menekan tombol → pada *keyboard*	4.Karakter bergerak ke kanan
5.Menekan tombol ↑ pada *keyboard*	6.Karakter loncat
Kondisi akhir	Karakter bergerak sesuai yang di *input* kan oleh pemain

Tabel 3.9 *Use Case Scenario* Melanjutkan *Game*

Identifikasi Masalah	
Nama	Melanjutkan *game*
Tujuan	Melanjutkan permainan
Deskripsi	Melanjutkan permainan pada level dimana *user* kalah
Aktor	Pemain
Skenario Utama	
Kondisi Awal	Tampilan menu game
Aksi Aktor	**Reaksi Sistem**
1.Memilih menu lanjutkan pada menu game	2.Menampilkan level terakhir
Kondisi akhir	Menampilkan level terakhir

Tabel 3.10 *Use Case Scenario* Menampilkan Pembuat *Game*

Identifikasi Masalah	
Nama	Menampilkan pembuat game
Tujuan	Menampilkan informasi pembuat game
Deskripsi	Menampilkan informasi pembuat game pada game *clean on your mouth*
Aktor	Pemain
Skenario Utama	
Kondisi Awal	Tampilan menu *game clean on your mouth*
Aksi Aktor	**Reaksi Sistem**
1.Memilih menu info pembuat game	2. Menampilkan informasi pembuat game
3. Memilih tombol kembali	4.menampilkan menu utama
Kondisi akhir	Kembali ke menu utama

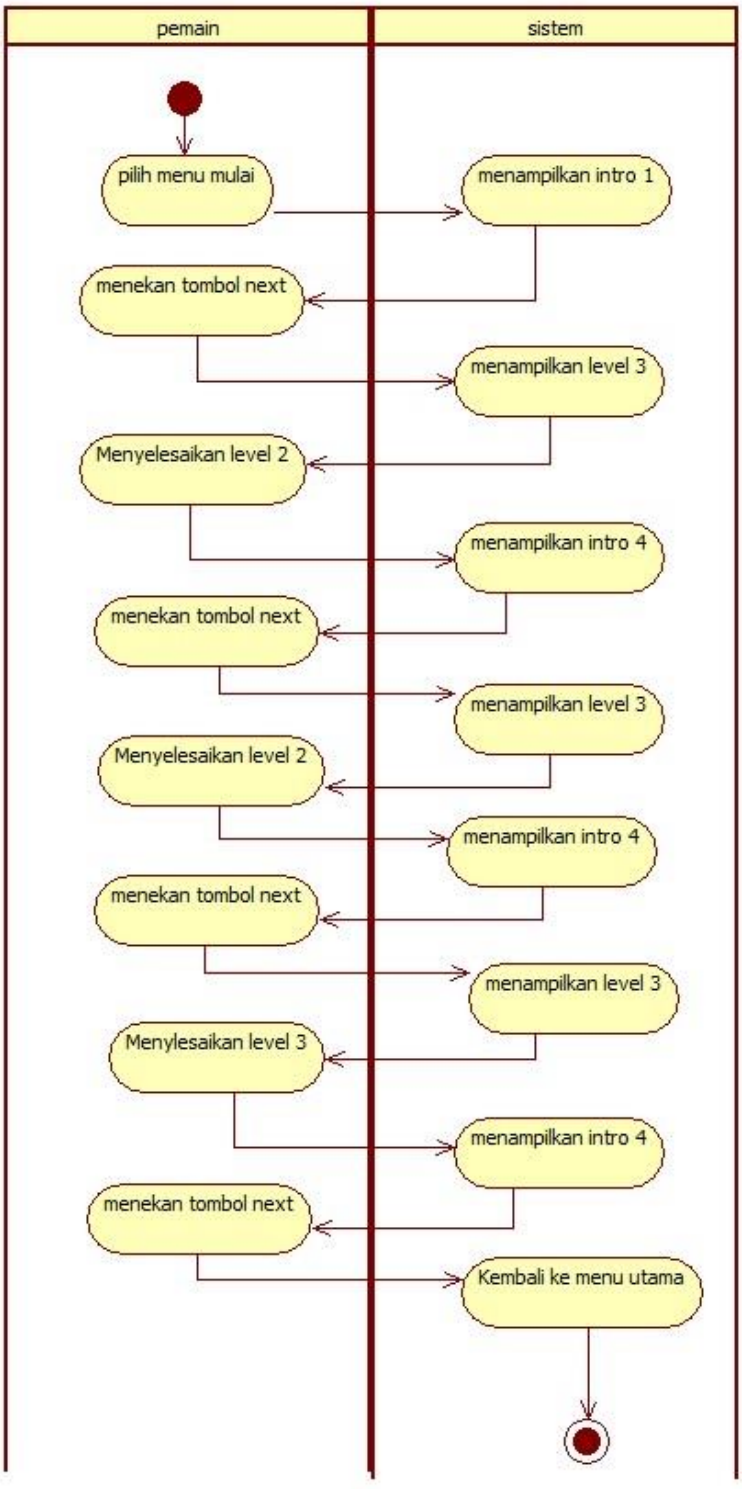
Tabel 3.11 *Use Case Scenario* Petunjuk Permainan

Identifikasi Masalah	
Nama	**Petunjuk permainan**
Tujuan	Menampilkan informasi petunjuk permainan
Deskripsi	Menampilkan informasi petunjuk permainan pada *game clean on your mouth*
Aktor	Pemain
Skenario Utama	
Kondisi Awal	Tampilan menu *game clean on your mouth*
Aksi Aktor	**Reaksi Sistem**
1.Memilih menu petunjuk permainan	2. Menampilkan informasi petunjuk permainan
3.Memilih tombol kembali	4.Menampilkan menu utama
Kondisi Akhir	Kembali ke menu utama

3.1.6.2 *Acitivity* Diagram

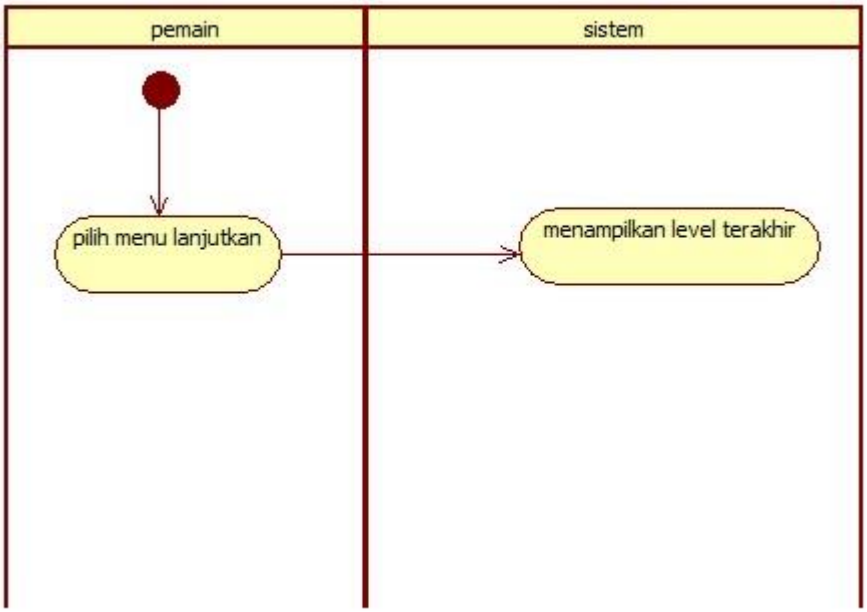
Activity diagram adalah teknik untuk mendeskripsikan logika *procedural*, proses bisnis dan aliran kerja dari urutan aktivitas dalam suatu proses yang mengacu pada *use case* diagram.

3.1.6.2.1 *Activity* Diagram Memulai *Game*



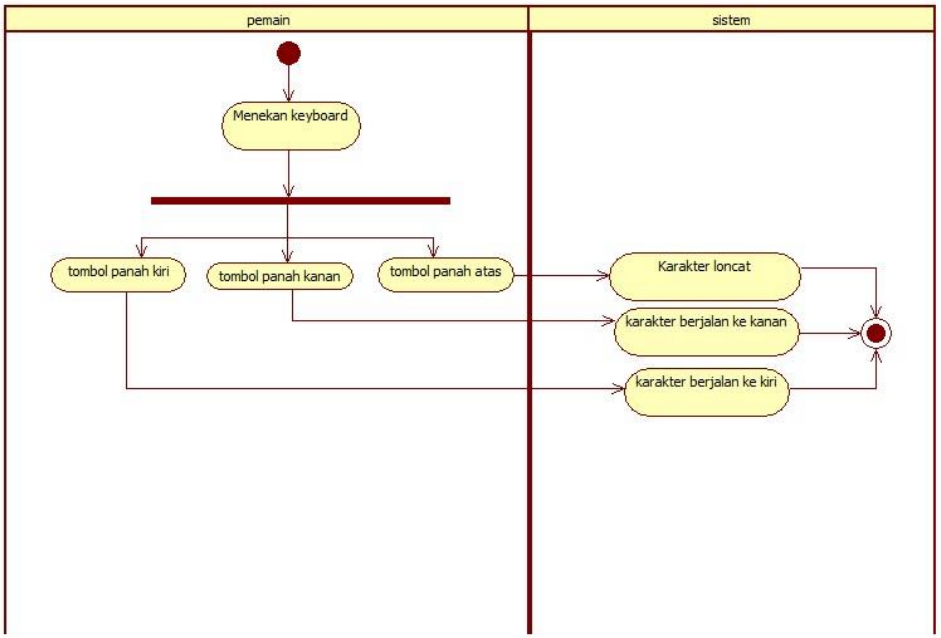
Gambar 3.20 *Activity* Diagram Memulai *Game*

3.1.6.2.2 *Activity* Diagram Melanjutkan *Game*



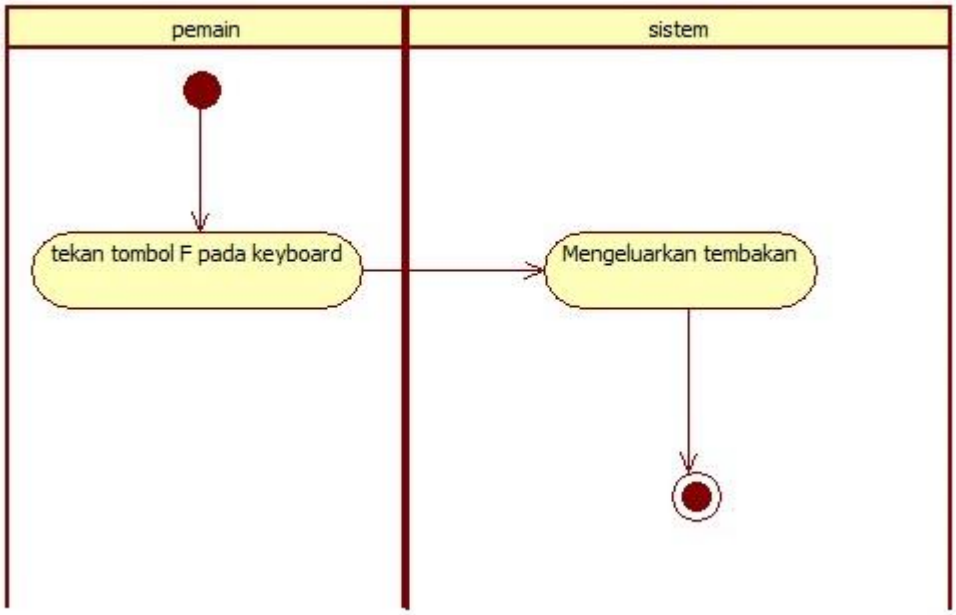
Gambar 3.21 *Activity* Diagram Melanjutkan *Game*

3.1.6.2.3 *Activity* Diagram Menggerakan karakter :



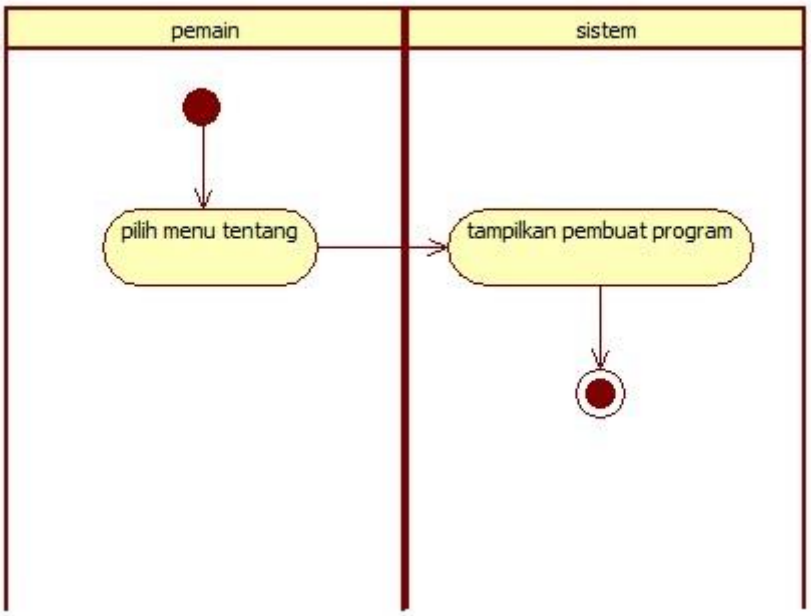
Gambar 3.22 Activity Diagram Menggerakan Karakter

3.1.6.2.4 *Activity* Diagram Menembak :



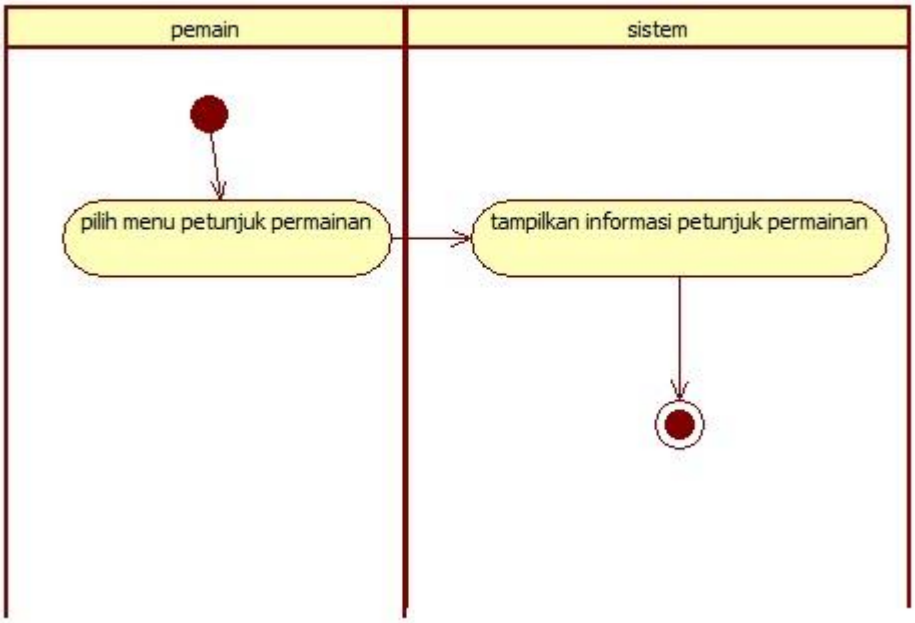
Gambar 3.23 *Activity* Diagram Menembak

3.1.6.2.5 *Activity* Diagram Menampilkan Tentang *Game* :



Gambar 3.24 *Activity* Diagram Menampilkan Tentang *Game :*

3.1.6.2.6 *Activity* Diagram Menampilkan Panduan *Game* :

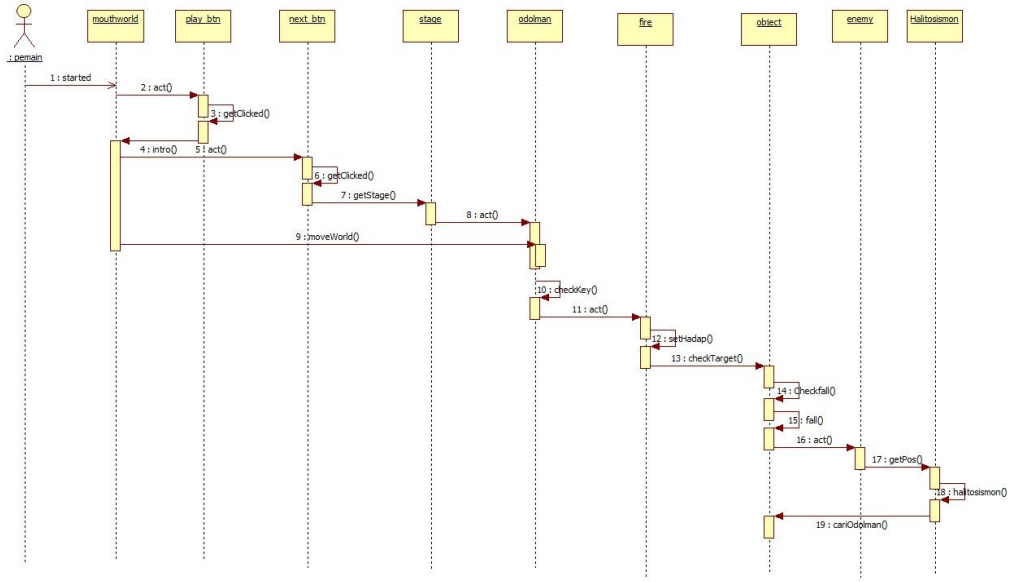


Gambar 3.25 *Activity* Diagram Menampilkan Panduan *Game*

3.1.6.3 *Sequence* Diagram

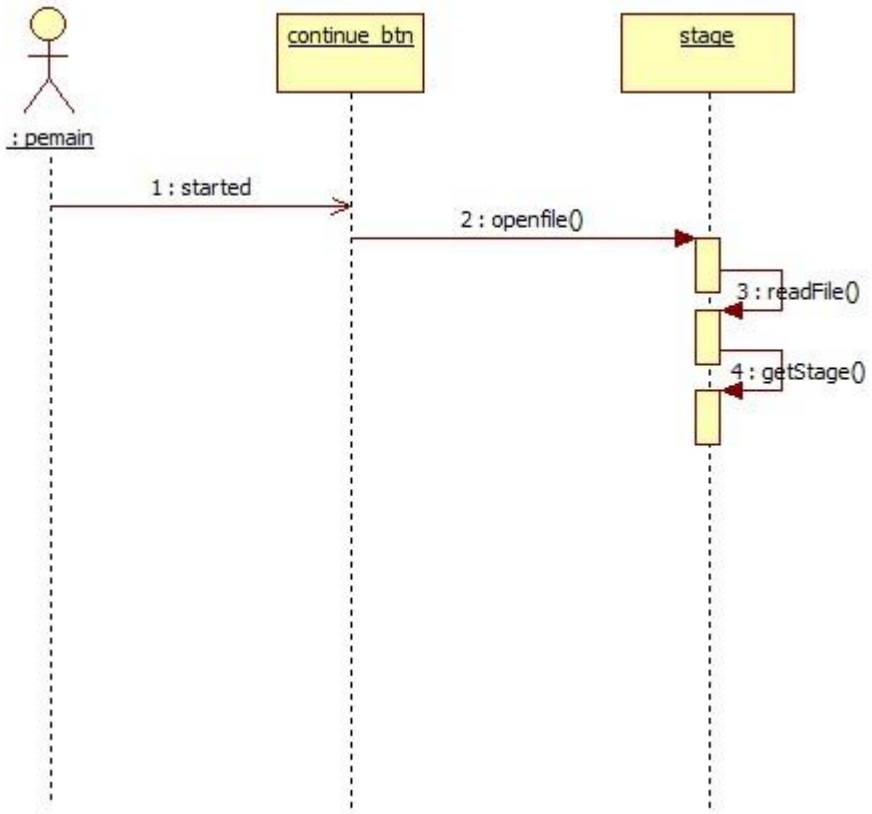
S*equence* diagram menggambarkan interaksi antar masing-masing objek pada setiap *use case* dalam urutan waktu. Interaksi ini berupa pengiriman serangkaian data antar objek-objek yang saling berinteraksi.

3.1.6.3.1 *Sequence Diagram* Memulai *Game*



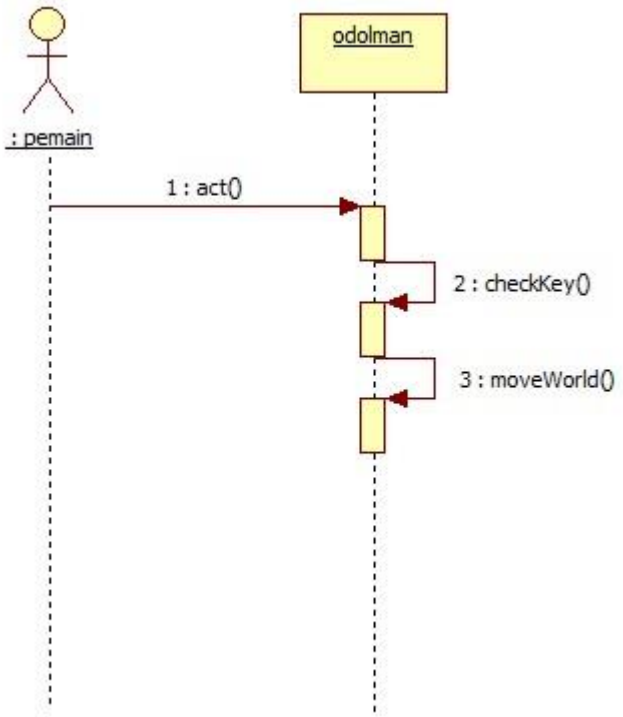
Gambar 3.26 *Sequence* Diagram Memulai *Game*

3.1.6.3.2 *Sequence* Diagram Melanjutkan *Game*



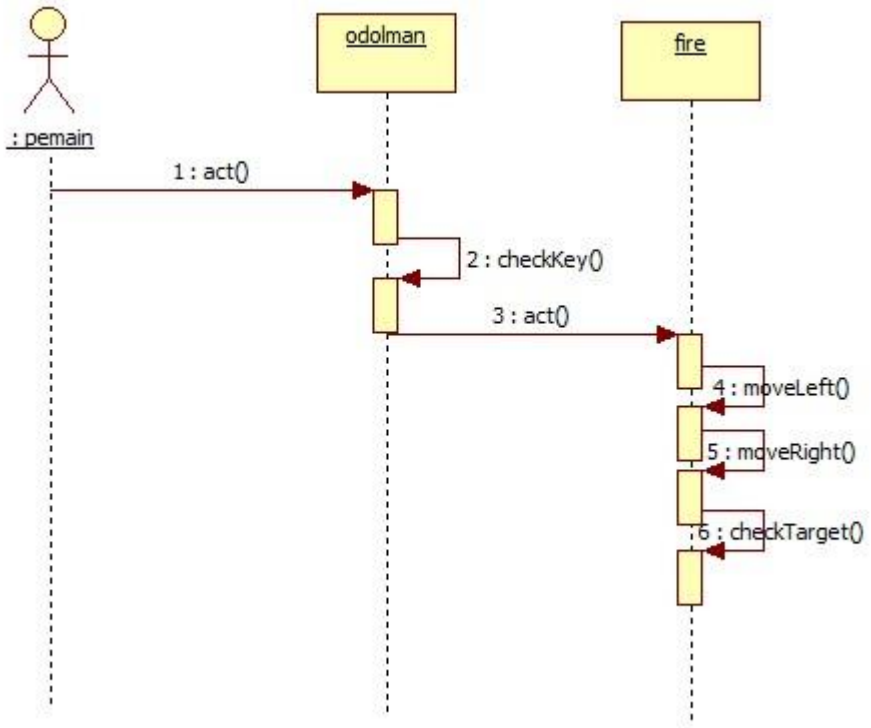
Gambar 3.27 *Sequence* Diagram *m*elanjutkan *Game*

3.1.6.3.3 *Sequence Diagram* Menggerakan karakter



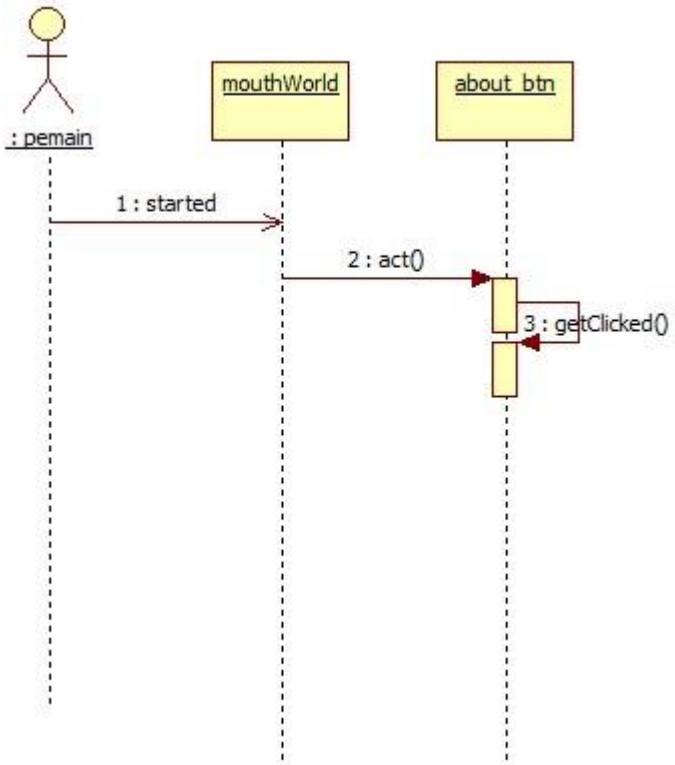
Gambar 3.28 *Sequence Diagram* Menggerakan Karakter

3.1.6.3.4 *Sequence Diagram* Menembak



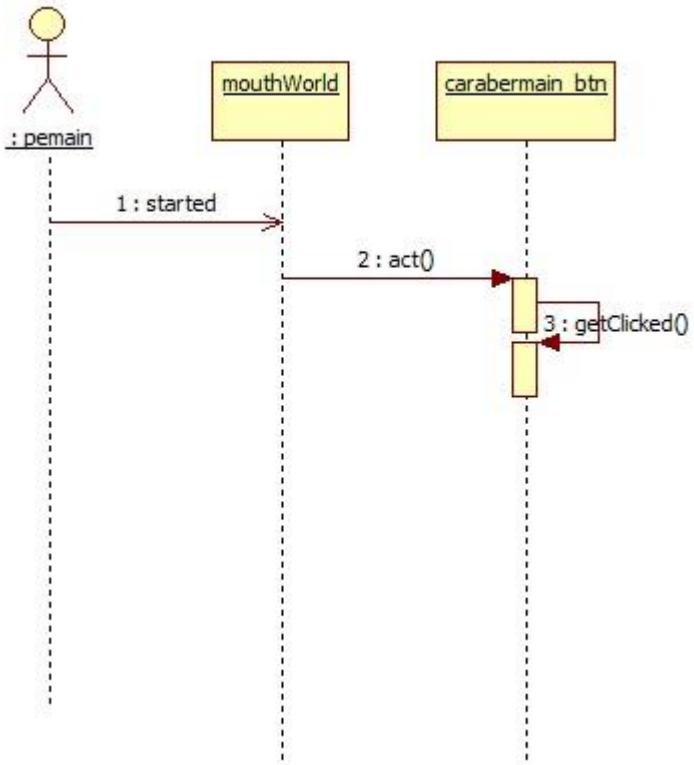
Gambar 3.29 *Sequence* Diagram Menembak Musuh

3.1.6.3.5 *Sequence* Diagram Menampilkan Pembuat *Game*



Gambar 3.30 Sequence Diagram Tentang *Game*

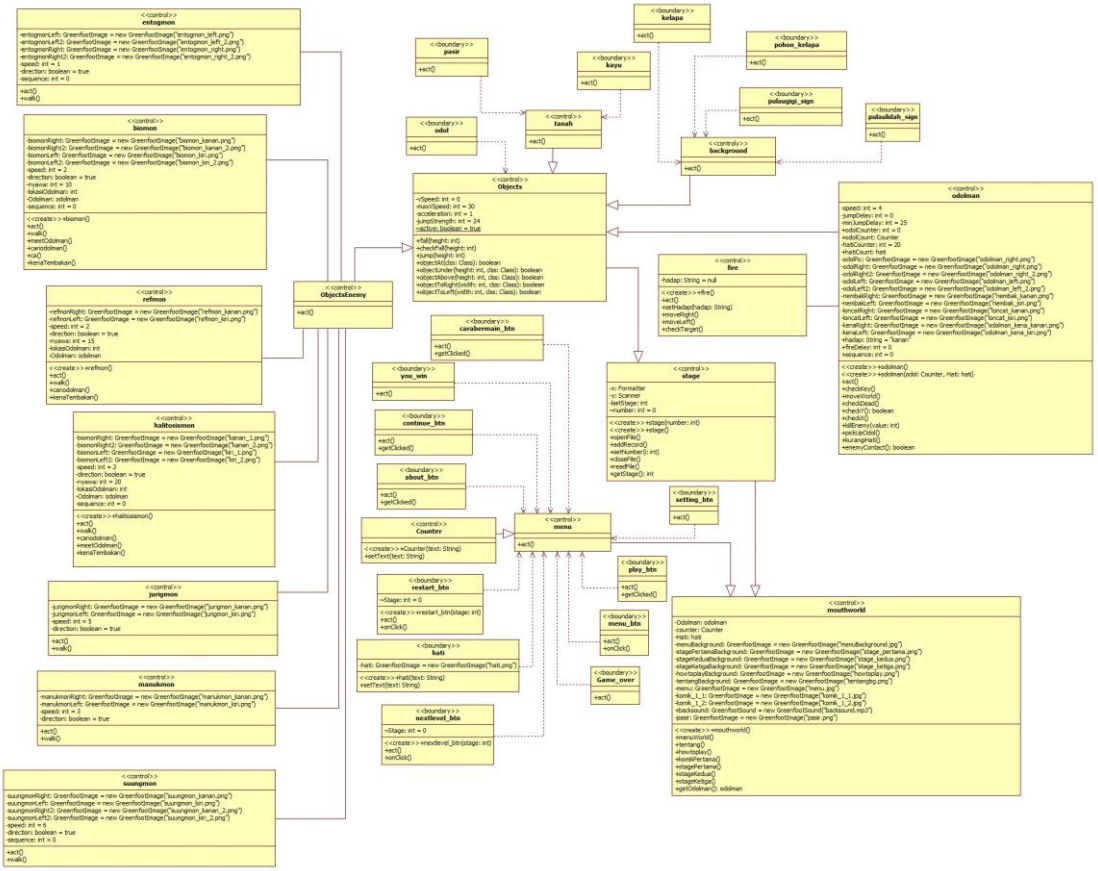
3.1.6.3.6 *Sequence* Diagram Menampilkan Panduan *Game* :



Gambar 3.31 Sequence Diagram Panduan *Game*

3.1.6.4 Class Diagram

Class Diagram menggambarkan struktur dan hubungan antar objek-objek yang ada pada sistem. Struktur itu meliputi atribut-atribut dan metode-metode yang ada pada masing-masing *class*. Hubungnnya dari masing-masing *class* yang ada tersebut digambarkan seperti pada gambar berikut:



Gambar 3.32 *Class* Diagram

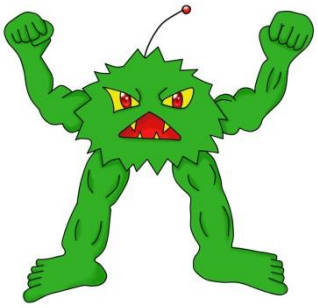
3.2 Perancangan Sistem

Perancangan sistem adalah merancang atau mendesain suatu sistem yang isinya adalah langkah langkah operasi dalam proses pengolahan data dan proses prosedur-prosedur untuk mendukung operasi sistem.

3.2.1 Perancangan Komponen Permainan

3.2.1.1 Karakter

Tabel 3.12 Karakter pada game clean on your mouth

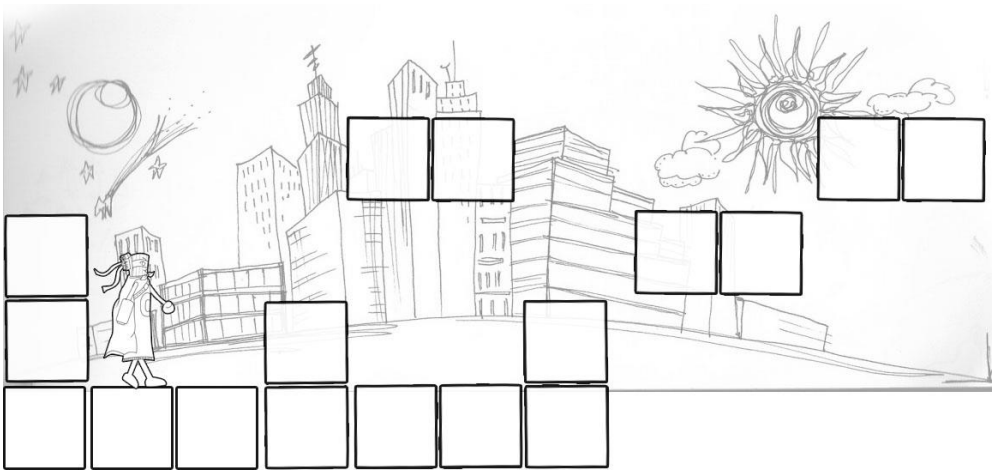
Nama Karakter	Kemampuan
 **Super Odolman**	Super Odolman merupakan karakter utama, memiliki kemampuan untuk melompat, menghindar, dan menembak
 **Biomon**	Biomon merupakan bos pada level 1, memiliki kemampuan untuk melompat, menghindar, menembak. Ai yang digunakan untuk musuh ini adalah algoritma A* (A *Star*)
 **Refmon**	Refmon merupakan bos pada level 2, memiliki kemampuan untuk melompat, menghindar, menembak. Ai yang digunakan untuk musuh ini adalah algoritma A* (A *Star*)

 **Halitosismon**	Halitosismon merupakan bos pada level 3, memiliki kemampuan untuk melompat, menghindar, menembak. Ai yang digunakan untuk musuh ini adalah algoritma A* (A *Star*)
 **Entogmon**	Entogmon merupakan musuh yang akan muncul pada level 1,2 dan 3 hanya sebagai rintangan , memiliki kemampuan untuk berjalan ke kanan dan ke kiri Tidak memiliki kecerdasan
 **Suungmon**	Entogmon merupakan musuh yang akan muncul pada level 1,2 dan 3 hanya sebagai rintangan , memiliki kemampuan untuk berjalan ke kanan dan ke kiri Tidak memiliki kecerdasan
 **Jurigmon**	Jurigmon merupakan musuh yang akan muncul pada level 1,2 dan 3 hanya sebagai rintangan , memiliki kemampuan untuk berjalan ke kanan dan ke kiri Tidak memiliki kecerdasan

3.2.1.2 Story Board

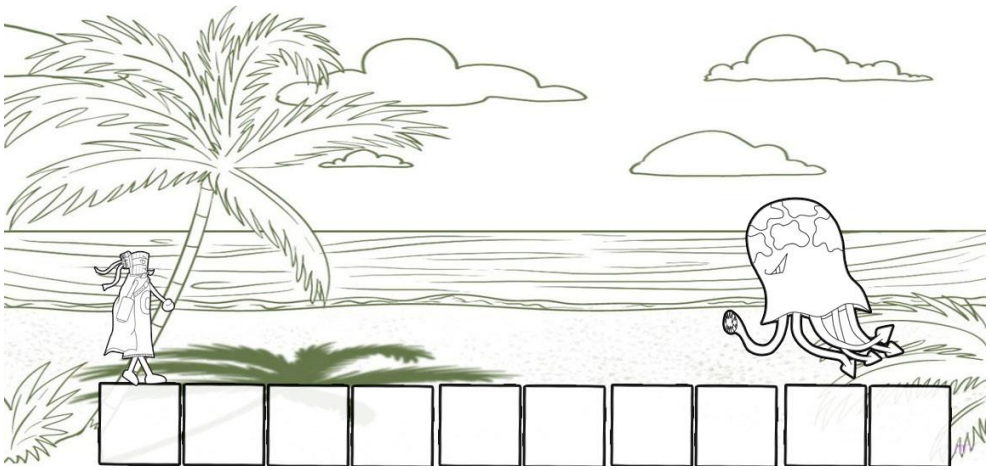
Storyboard adalah skesta gambar yang disusun seperti halnya sebuah naskah cerita. Dengan *storyboard* kita dapat menyampaikan ide cerita yang kita punya kepada orang lain dengan cara lebih mudah, karena dengan *storyboard* kita dapat menggirng imajinasi seseorang menuju persepsi yang kita inginkan melalui gambar-gambar yang sudah kita buat.

1. *Storyboard* pada level 1 background nya berada di perkotaan, di tempat ini pemaian akan menghadapi boss level 1 yang bernama Refmon.



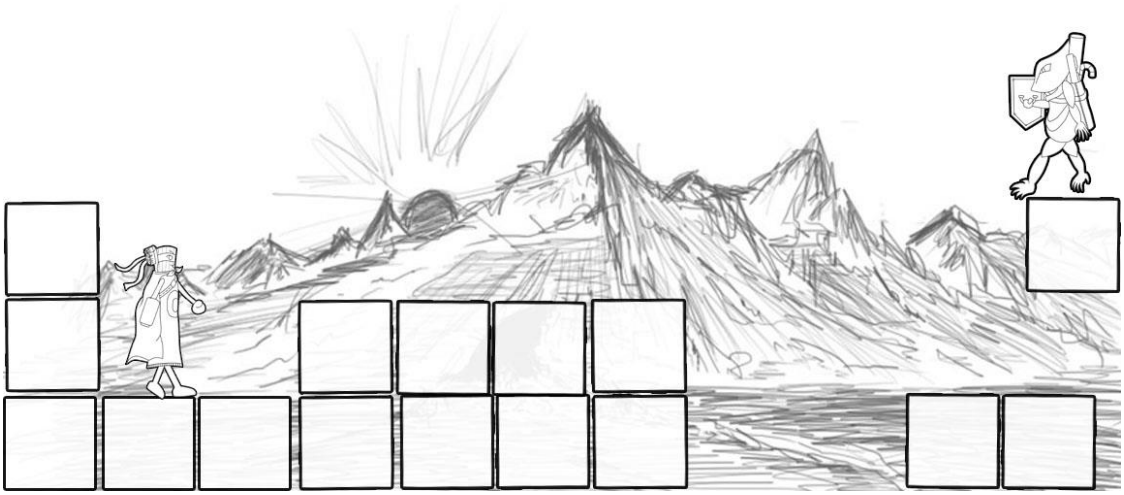
Gambar 3.33 Storyboard level 1

2. *Storyboard* pada level 2 background nya berada di pantai, di tempat ini pemaian akan menghadapi boss level 2 yang bernama Biomon.



Gambar 3.34 Storyboard level 2

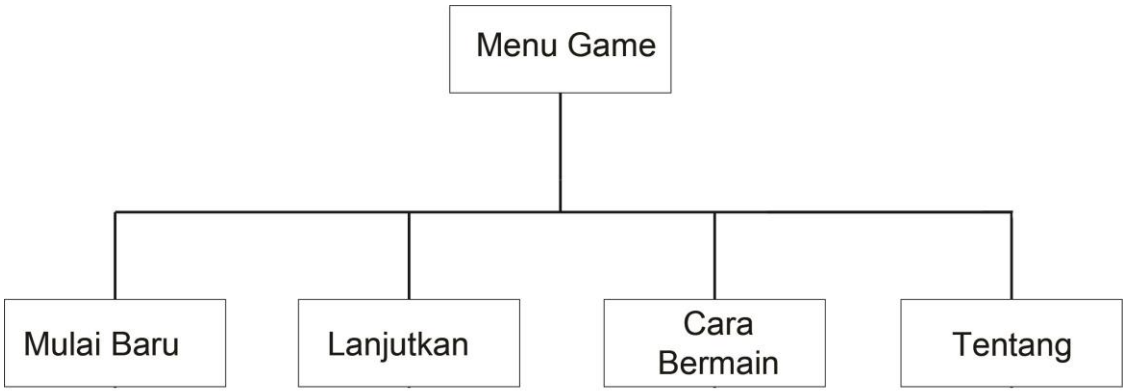
3. *Storyboard* pada level 3 background nya berada di bukit, di tempat ini pemaian akan menghadapi boss level 3 yang bernama Halitosismon.



Gambar 3.35 *Storyboard* level 3

3.2.2 Perancangan Struktur Menu

Merupakan fitur-fitur yang disediakan pada sistem yang dirancang dan untuk mempermudah dalam penggunaan sistem nantinya. Perancangan struktur menu dapat dilihat pada gambar berikut :

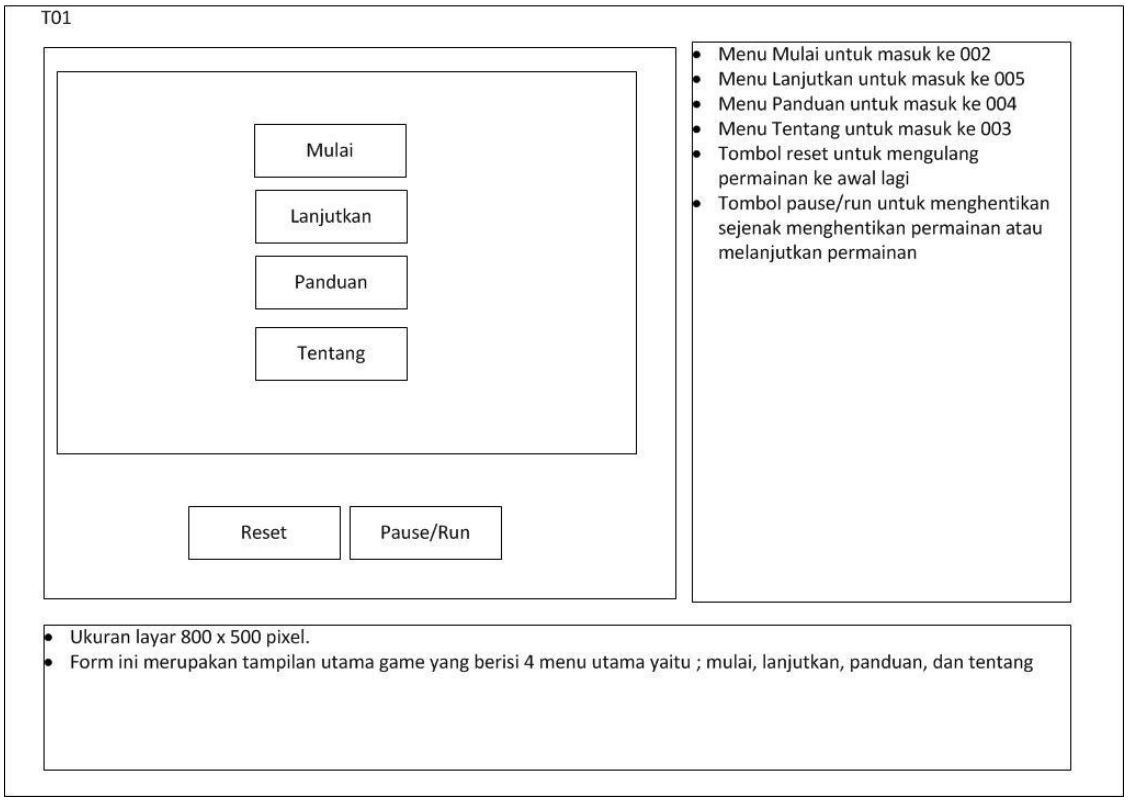


Gambar 3.36 Struktur Menu

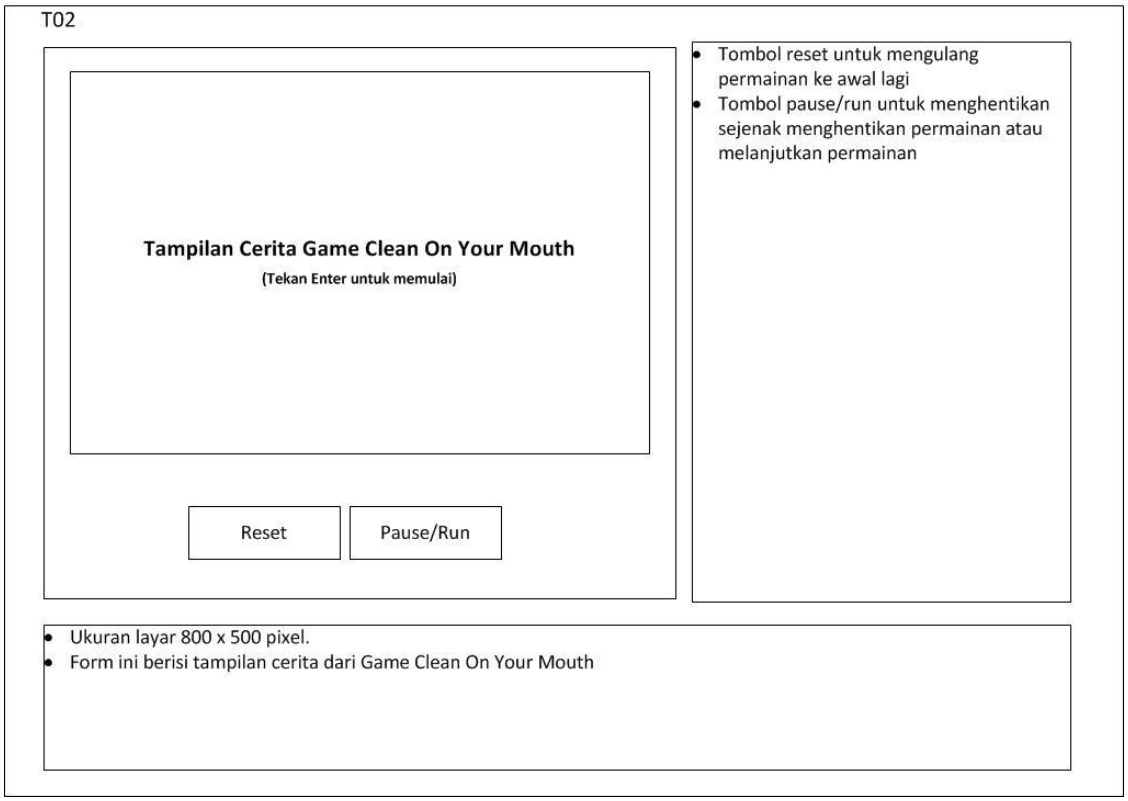
3.2.3 Perancangan Antar Muka

Merancang antarmuka merupakan bagian yang paling penting dari merancang sebuah system. Biasanya hal tersebut juga merupakan bagian yang paling sulit, karena dalam merancang antarmuka harus memenuhi tiga persyaratan yaitu ; antarmuka harus

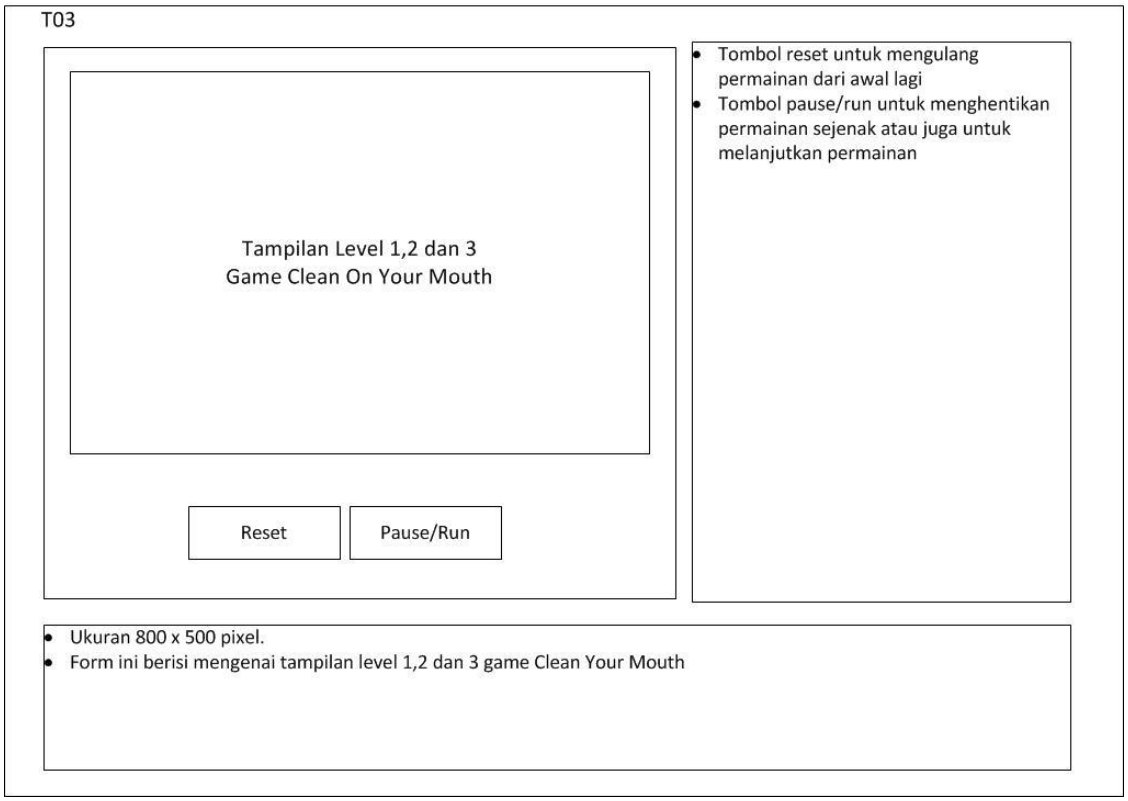
sederhana, antarmuka harus lengkap dan antarmuka harus memiliki kinerja yang cepat. Perancangan antarmuka dari game dapat dilihat pada gambar berikut ;



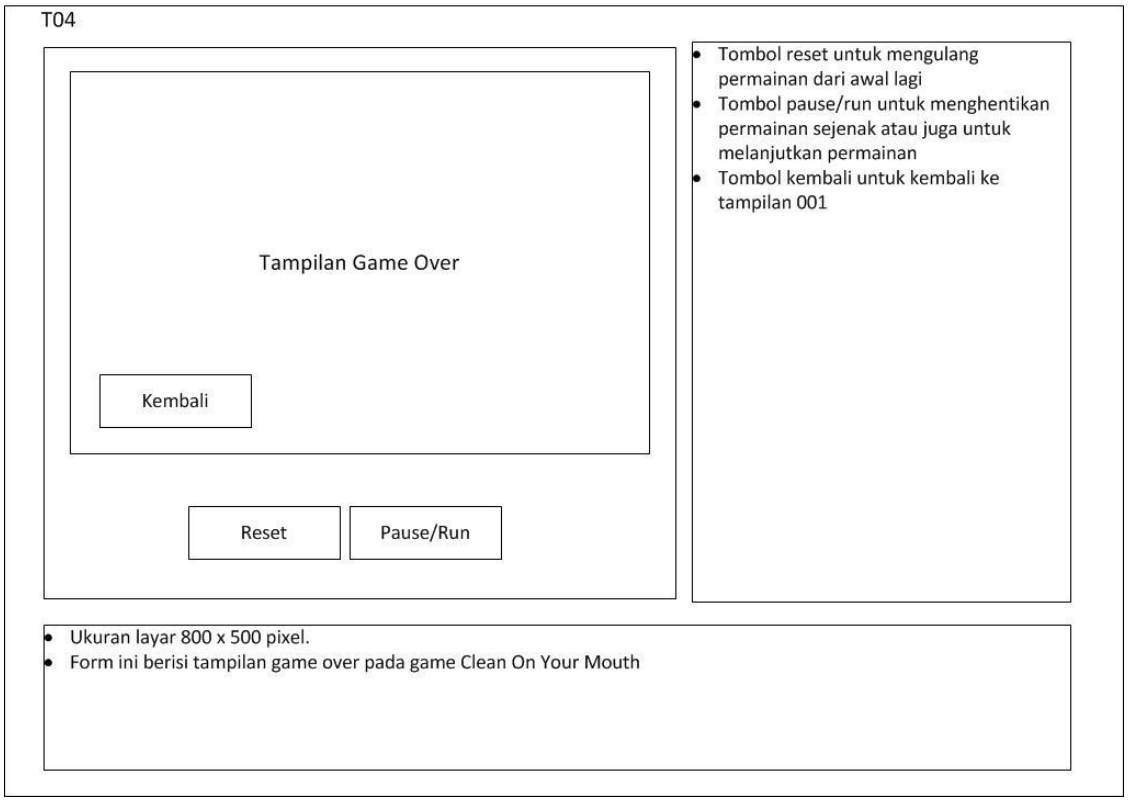
Gambar 3.37 Tampilan antarmuka menu *game*



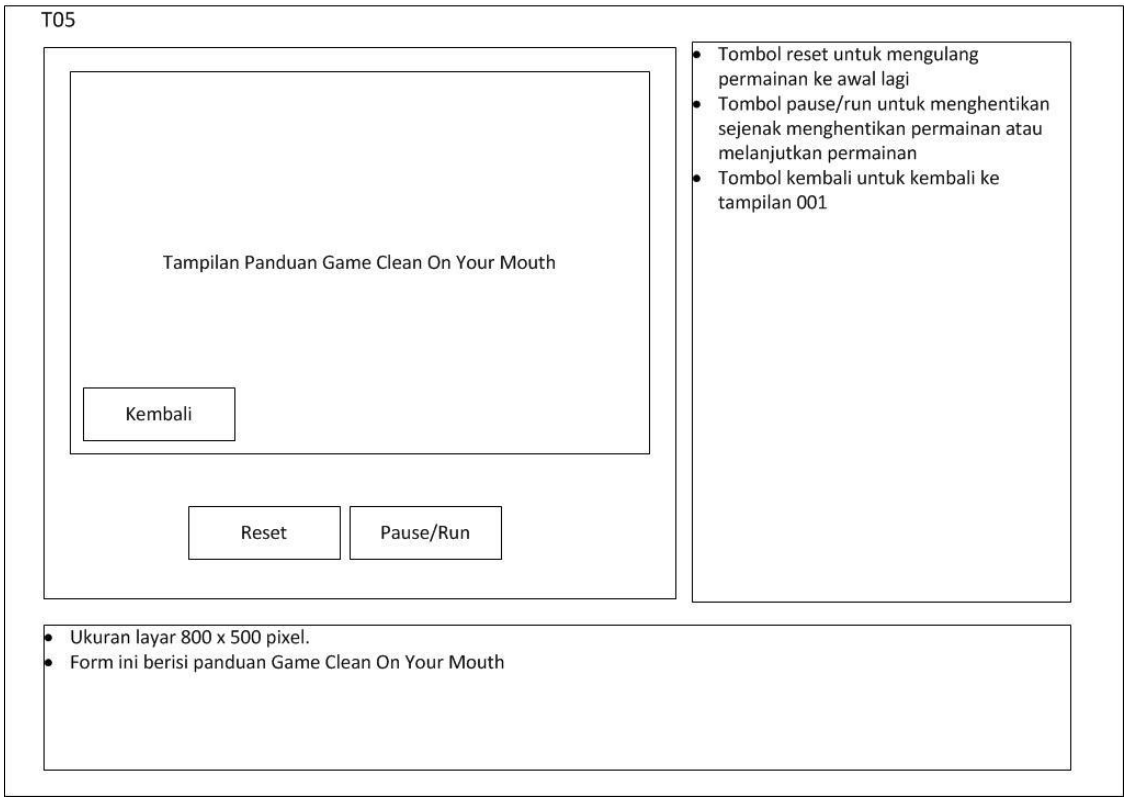
Gambar 3.38 Tampilan Antarmuka Cerita *Game*



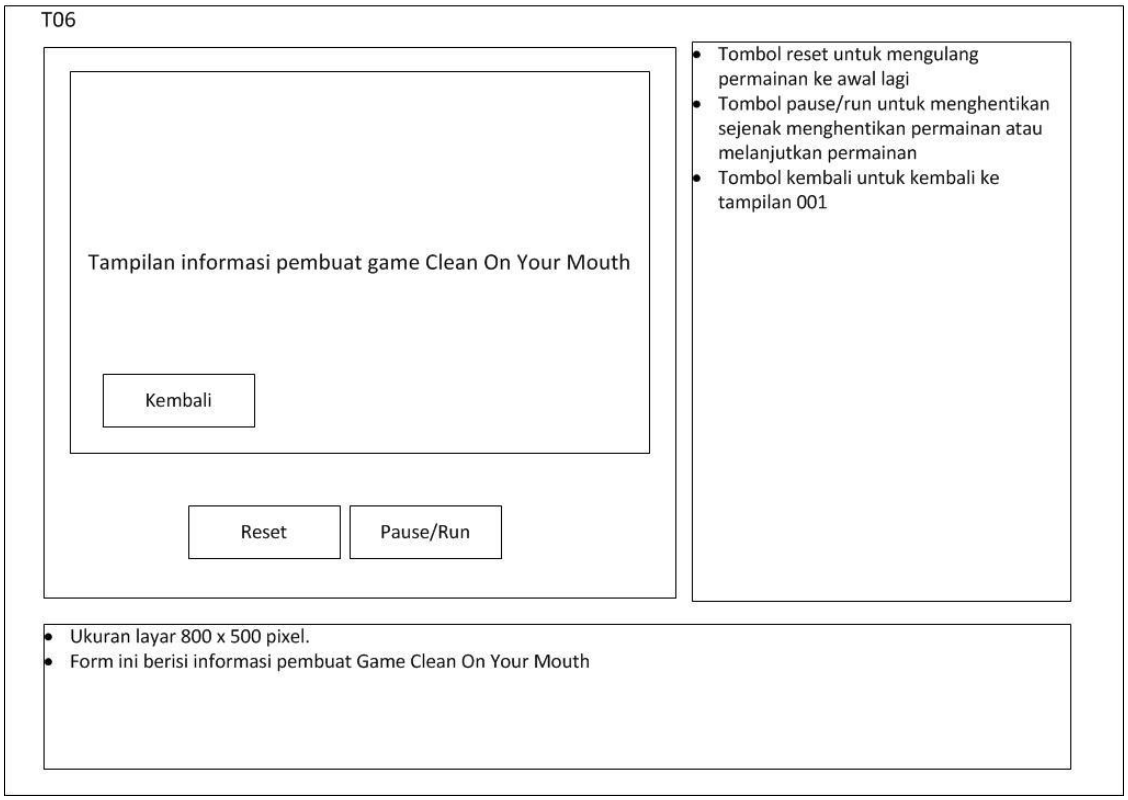
Gambar 3.39 Tampilan Antarmuka *Level* 1,2 dan 3 *Game*



Gambar 3.40 Tampilan antarmuka *game over*



Gambar 3.41 Tampilan antarmuka panduan *game*



Gambar 3.42 Tampilan antarmuka informasi pembuat *game*

3.2.4 Perancangan *Method*

Perancangan *method* menjelaskan penjabaran alur proses-proses yang ada pada sistem dan dibangun menggunakan diagram *flowchart*. *Method* yang ada pada aplikasi *game* ini terdiri dari dua jenis *method*, yaitu *method* yang sudah ada di dalam *tools* pembangun aplikasi (hanya perlu dipanggil sesuai kebutuhan) dan *method* yang dirancang berdasarkan kebutuhan aplikasi.

3.2.4.1 Perancangan *Method* yang Tersedia di Dalam *Tools* Pembangun

Method yang digunakan dari *tools* pembangun aplikasi dalam pembangunan *game Clean On Your Mouth* ini adalah sebagai berikut :

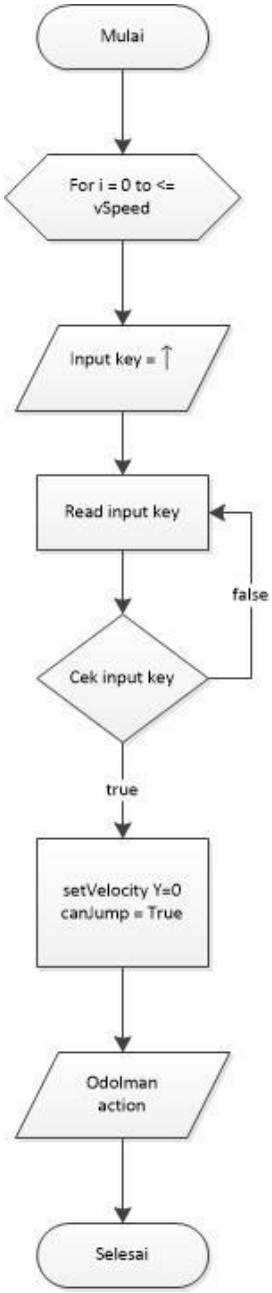
Tabel 3.13 Penggunaan Method yang Sudah Ada Dalam *Tools* Pembangun

No	Nama *Method*	Keterangan
1	Started()	Untuk menjalankan *world*
2	Stopped()	Untuk menghentikan *world*
3	act()	Untuk melakukan tindakan pada objek di *world*
4	addObject()	Untuk menambahkan objek
5	removeObject()	Untuk menghapus objek
6	getBackground()	Untuk membuat gambar menjadi background
7	Collison()	Untuk memeriksa tabrakan antar objek
8	getHeight()	Untuk menentukan tinggi world
9	getWidht()	Untuk menentukan lebar world
10	getObject()	Untuk mengambil objek di *class* tertentu
11	setBackground()	Untuk mengatur *background*
12	getClicked()	Untuk membaca input dari mouse

3.2.4.2 Perancangan *Method* yang di Buat

1. Perancangan *method fall*

   Method ini dijalankan ketika pemain atau musuh yang sudah meloncat atau jatuh dari tempat yang lebih tinggi. Untuk lebih jelasnya dapat di lihat pada gambar berikut :

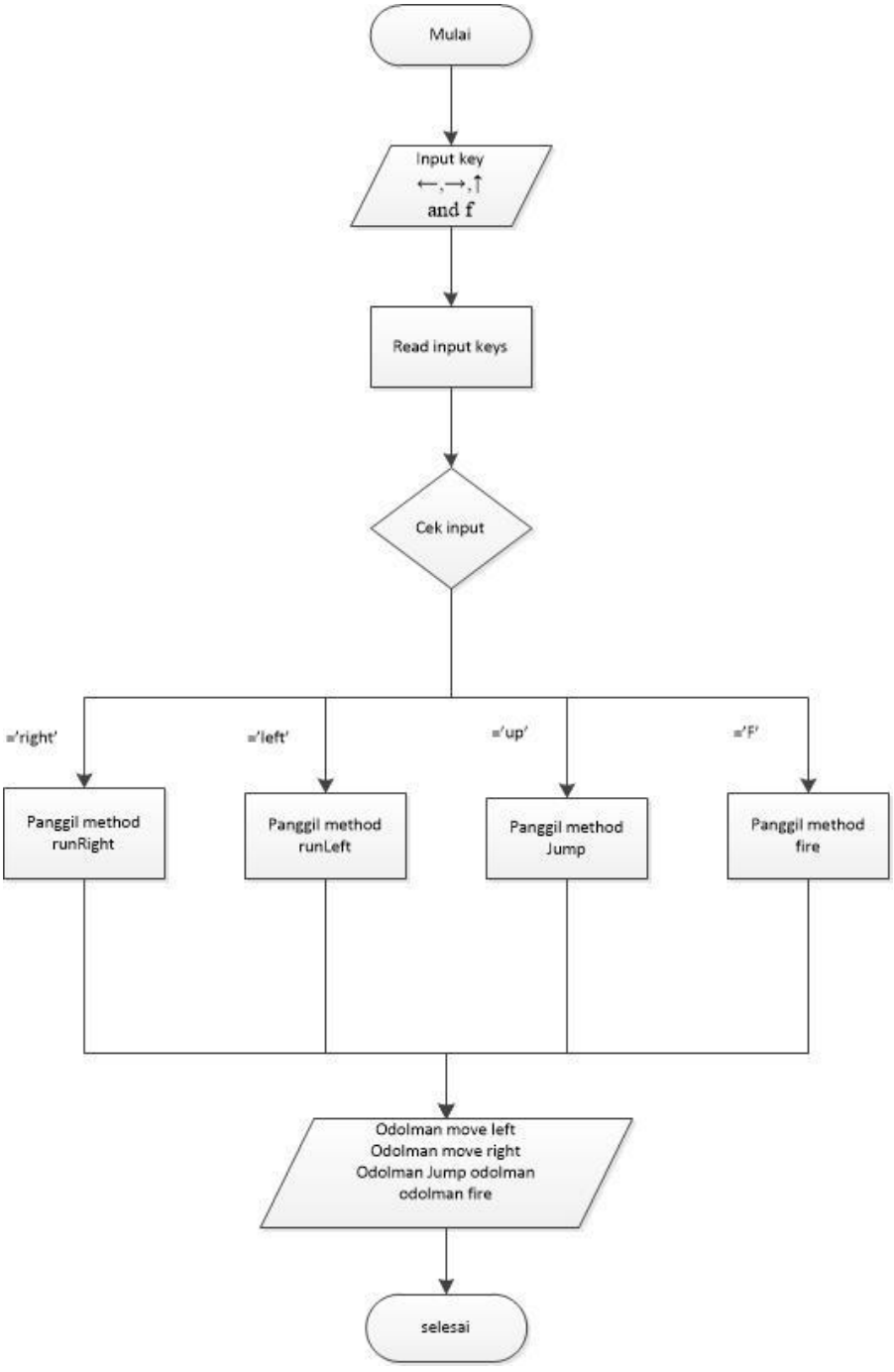


Gambar 3.43 Perancangan *Method Fall*

2. Perancangan *checkkey()*

Method ini dijalankan untuk mengecek input keyboard yang di *input* pemaian.

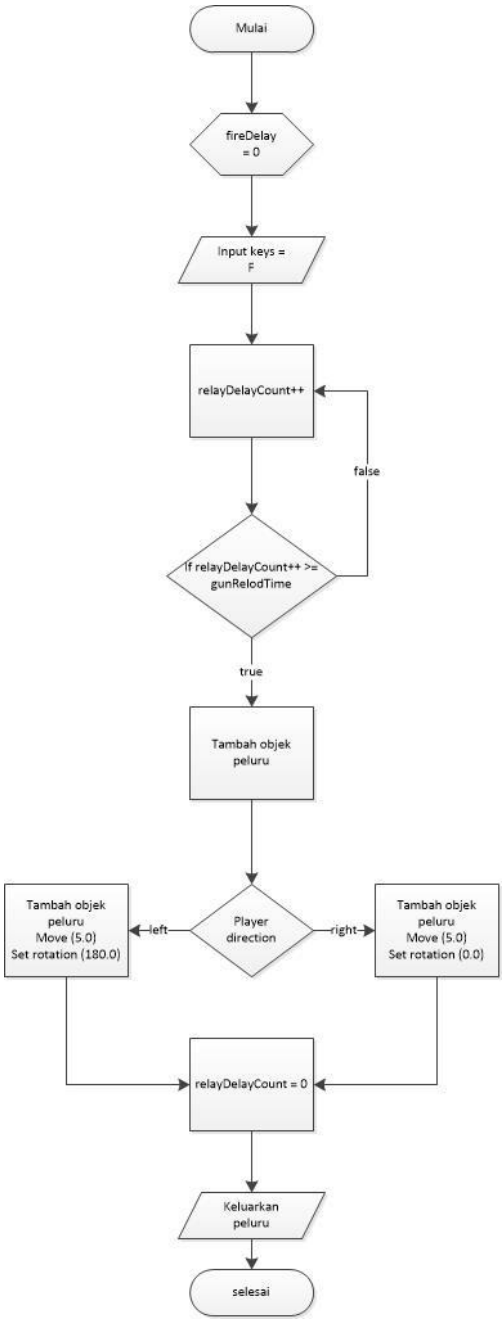
Untuk lebih jelasnya dapat dilihat pada gambar berikut :



Gambar 3.44 Perancangan *Method Check Keys*

3. Perancangan *method fire* di kelas Odolman

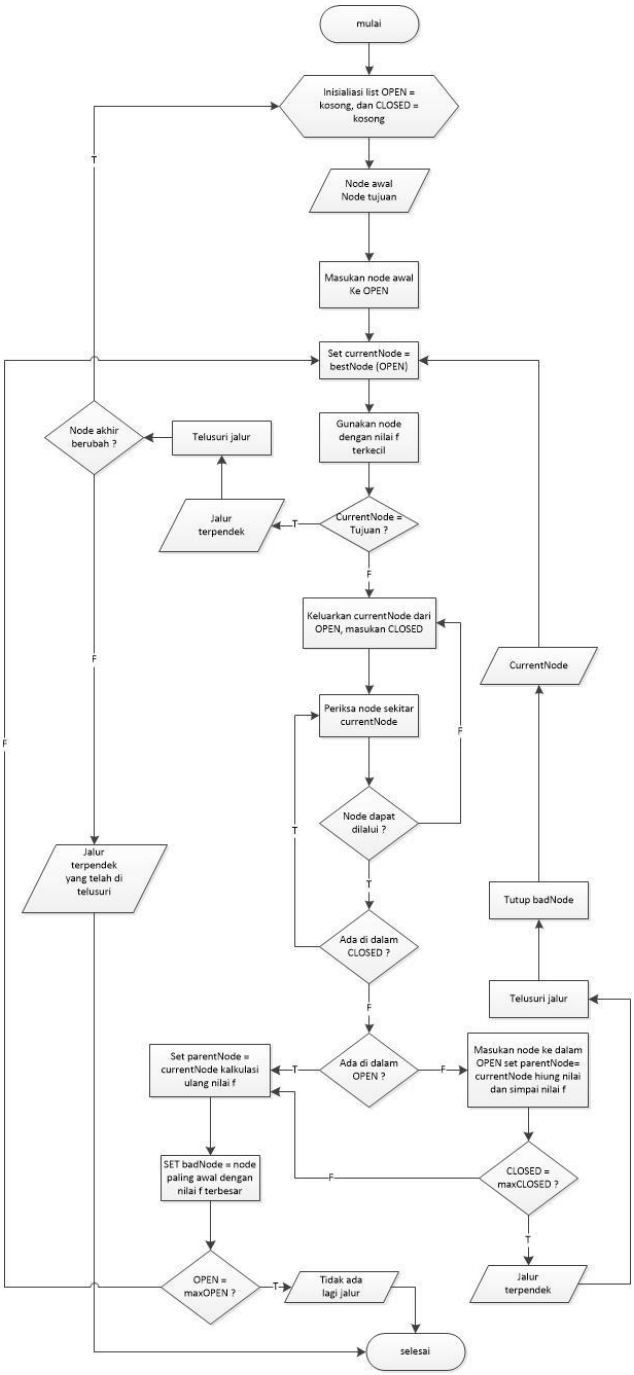
   Method ini dijalankan untuk membuat Odolman menembak. Untuk lebih jelasnya dapat dilihat pada gambar berikut :



Gambar 3.45 Perancangan *Method Fire*

4. Perancangan *method cariodolman* di kelas *enemy*

Method ini dijalankan untuk membuat musuh mencari keberadaan Odolman. Untuk lebih jelasnya dapat dilihat pada gambar berikut :

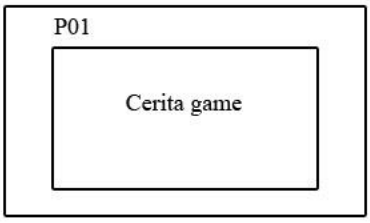


Gambar 3.46 Perancangan *method cariodolman*

3.2.5 Perancangan Pesan

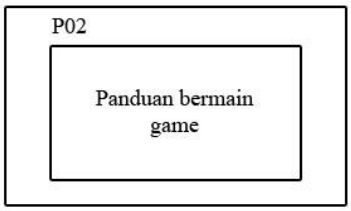
Merupakan kumpulan pesan yang ada pada sistem yang dirancang dan berfungsi, agar pengguna dapat mengerti maksud perancang dalam menggunakan aplikasi. Pesan-pesan yang muncul dalam aplikasi sebagai berikut ;

1. *User* memilih menu *game* akan mengeluarkan intro (*storyline game*).



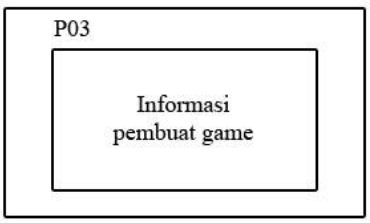
Gambar 3.47 Perancangan Pesan Cerita *Game*

2. *User* memilih menu panduan akan mengeluarkan tampilan panduan bermain *game*



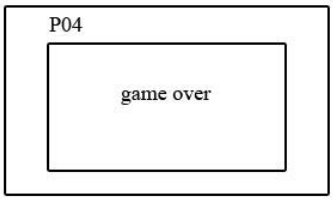
Gambar 3.48 Perancangan Pesan Panduan Bermain *Game*

3. *User* memilih menu tentang akan mengeluarkan tampilan informasi pembuat *game*



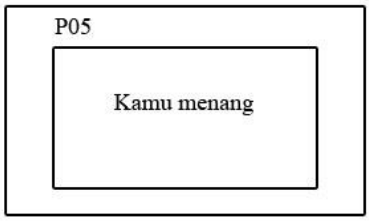
Gambar 3.49 Perancangan Pesan Informasi Pembuat *Game*

4. *U*ser kalah dalam permainan akan mengeluarkan pesan *game over*.



Gambar 3.50 Perancangan Pesan *Game Over*

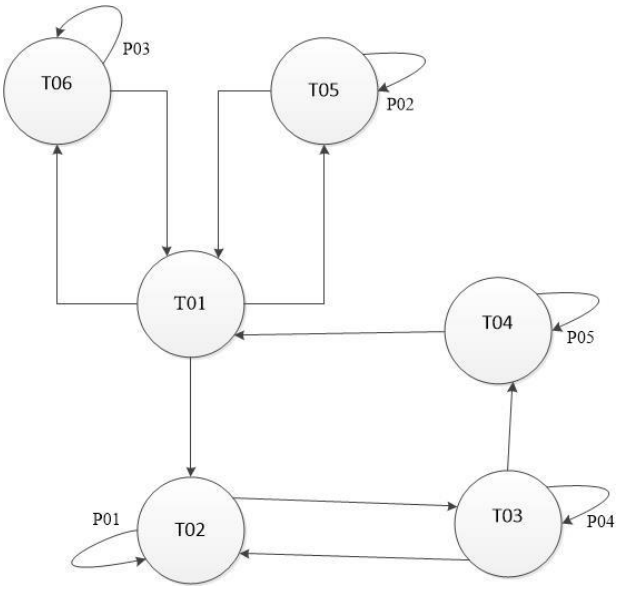
5. *User* menyelesaikan level *game* akan mengeluarkan pesan kamu menang dan akan melanjutkan untuk menuju level berikutnya



Gambar 3.51 Perancangan Pesan Kamu Menang

3.2.6 Jaringan Semantik

Setelah melakukan perancangan antar muka dan perancangan pesan maka dilakukanlah perancangan terhadap hubungan antar form dan pesan. Untuk menggambarkan hubungannya digambarkan dalam sebuah jaringan semantik.



Gambar 3.52 Jaringan Semantik