BAB 3

ANALISIS DAN PERANCANGAN SISTEM

Analisis merupakan proses penguraian konsep ke dalam bagian-bagian yang lebih sederhana, sehingga struktur logisnya menjadi jelas. Analisis merupakan metode untuk menguji, menilai, dan memahami sistem pemikiran yang kompleks dengan memecahnya ke dalam unsur-unsur yang lebih sederhana sehingga hubungan antar unsur-unsur itu menjadi jelas.

3.1 Analisis Sistem

Analisis sistem merupakan penguraian dari suatu sistem informasi yang utuh kebagian-kebagian komponennya yang dimaksudkan untuk mengidentifikasi dan mengevaluasi segala macam permasalahan dan hambatan apa saja yang bisa terjadi dan kebutuhan-kebutuhan yang apa saja yang diharapkan dapat menjadi acuan untuk diusulkannya perbaikan-perbaikan.

Dalam membangun sebuah sistem, diperlukan sebuah analisis terlebih dahulu karena tanpa menganalisis sistem yang ada maka aplikasi tidak akan bisa dibangun. Pada analisis sistem akan mencakup analisis masalah, analisis *game* sejenis ,analisis kebutuhan fungsional dan analisis kebutuhan non fungsional.

3.1.1 Analisis Masalah

Setiap *game* saat ini hanya memiliki satu jenis genre *game* pada setiap *gameplay* permainan seperti hanya berjenis *action* saja atau *adventure* saja. *Game* dengan hanya memiliki satu jenis *genre game* mempunyai kekurangan yaitu menjadikan *gameplay* permainan yang monoton disetiap *level* permainan. Salah

satu *game* yang hanya memiliki satu jenis *genre game* adalah *game* Petualang Kungi.

Game petualangan kungi adalah sebuah *game* yang memiliki jenis *genre tile-based game*, *game* ini mempunya beberapa kekurangan diantara tidak adanya AI dalam *game*, dan mempunyai kontrol pergerakan karakter yang buruk dan audio yang monoton sehingga *game* ini kurang menarik, ini dibuktikan dengan *rating* yang rendah pada *game* Petualang Kungi yang ada di internet[13].



Gambar 3.1 Rating *Game* Petualangan Kungi [13]

Salah satu faktornya adalah AI (*Artificial Intelligence*) dalam sebuah *game*. Karena dengan tanpa adanya AI, *game* akan menjadi sangat membosankan dan tidak menarik untuk dimainkan.

Dalam pembahasan *game* yang akan dibangun adalah menggabungkan kedua jenis *game* yang berbeda menjadi satu judul *game*, diharapkan dengan menggabungkan beberapa jenis *game* yang berbeda akan menjadikan permainan yang lebih menarik. Dalam *game* yang akan dibangun juga akan diterapkan sebuah AI dengan menerapkan algoritma A* dalam objek musuh sehingga musuh mempunyai kemampuan untuk mengejar menuju posisi pemain.

3.1.2 Analisis *Game* Sejenis

Pada tahap ini dilakukan analisis terhadap *game-game* yang sejenis, baik yang sudah dapat dimainkan secara online maupun yang belum. Tujuan dari analisis ini yaitu untuk mendapatkan konsep permainan dan perbandingan dari jenis *game* yang sudah ada.

Game pembanding yang akan dibahas adalah *game TwinBee* dan *game* Petualangan Kungi.

3.1.2.1 Analisis Game TwinBee

TwinBee adalah salah satu *game* klasik yang dikembangkan oleh Konami pada tahun 1985 [12], dalam *game* ini pemain mengendalikan sebuah pesawat dan harus melawat pesawat musuh dengan cara menembaknya. Berikut adalah tampilan *game shooter TwinBee :*



Gambar 3.2 Tampilan *Game* TwinBee [12]

3.1.2.2 Gameplay Game TwinBee

Secara garis besar *gameplay* yang terdapat dalam *game* TwinBee adalah sebagai berikut :

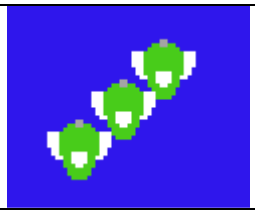
1. Pemain mengendalikan sebuah pesawat.
2. Pemain dapat menembak ke depan untuk melawan musuh.
3. Apabila tertembak oleh pesawat musuh maka pemain akan mati.
4. Pemain harus dapat bertahan hidup untuk dapat melanjutkan ke *Level* berikutnya.

3.1.2.3 Karakter Pada Game TwinBee

Berikut ini merupakan karakter yang ada dalam *game* klasik *TwinBee*, lebih lengkap dapat dilihat pada tabel dibawah ini:

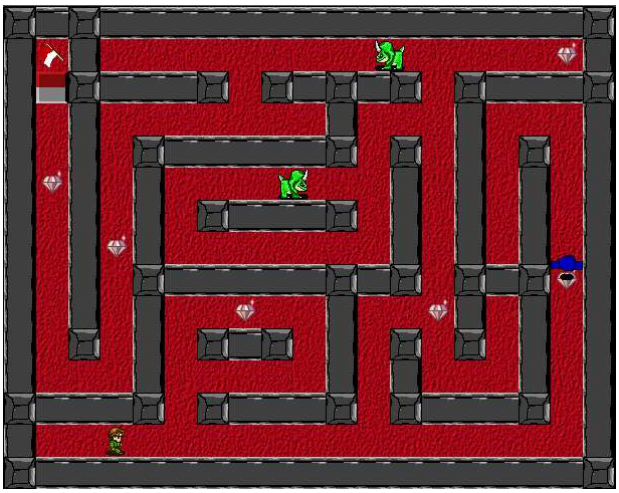
Tabel 3.1 Tabel pengenalan karakter pada game TwinBee

No	Nama	Gambar	Penjelasan
1	Karakter utama		Merupakan tokoh utama dalam *game* yang dimainkan pemain dengan *input keyboard*
2	*Monster* 1		Musu yang terdapat dalam game *Twinbee*, musuh ini bergerak secara diagonal dari layar.
3	*Monster* 2		Musuh yang terdapat dalam *game* sama seperti musuh pertama musuh ini bergerak secara horizontal

4	Diamond			Merupakan item yang harus dikumpulkan kungi untuk membuka penghalang menuju bendera/*goal*

3.1.2.4 Analisis Game Petualangan Kungi

Game Petualangan Kungi adalah salah satu *game* buatan anak bangsa kita, *game* ini termasuk kedalam jenis *game tiled-based game*. *Game* ini bercerita tentang seorang anak laki yang tersesat dalam sebuah kuil di desa majapahit. Berikut ini merupakan tampilan *gameplay* Petualangan Kungi.

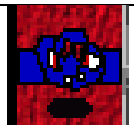
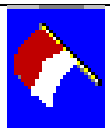


Gambar 3.2 Tampilan *Game* "Petualangan Kungi" [13]

3.1.2.5 Karakter Pada Game Petualangan Kungi

Pengenalan karakter *game* merupakan faktor penting dalam menjaga pemain agar tidak cepat merasa bosan dan tetap terlibat dalam dunia *game*. Berikut adalah konten-konten yang terdapat pada *game* petualangan kungi dapat dilihat dari tabel berikut.

Tabel 3.2 Karakter *Game* Petualangan Kungi

No	Nama	Gambar	Penjelasan
1	Kungi		Merupakan tokoh utama dalam *game* yang dimainkan pemain dengan *input keyboard*
2	*Monster* 1		Musuh yang menghalangi kungi dengan bergerak kekanan dan kekiri
3	*Monster* 2		Musuh yang menghalangi kungi dengan bergerak kekanan dan kekiri
4	Diamond		Merupakan item yang harus dikumpulkan kungi untuk membuka penghalang menuju bendera/*goal*
5	Bendera/*Goal*		Merupakan tujuan akhir pemain, pemain harus menuju bendera untuk melanjutkan ke*Level* berikutnya.
6	*Coin* Emas		Merupkan poin tambahan yang ada dalam *game*.
7	*Block* Penghalang		Blok yang akan terbuka ketika semua diamond berhasil dikumpulkan

3.1.2.6 Gameplay Game Petualangan Kungi

Gameplay dalam *game* Petualangan kungi adalah sebagai berikut :

1. Karakter utama berada dalam kuil tua majapahit dan harus menemukan jalan keluar dengan menyusuri ruangan-ruangan.
2. Didalam arena permainan terdapat *monster* yang siap mengahadang Kungi untuk menuju pintu keluar.

3. Untuk dapat keluar dalam arena kungi harus mengumpulkan batu permata yang tersebar dalam arena *game*.
4. Dengan terkumpulnya semua batu permata makan batu penghalang yang menghalangi pintu keluar akan tebuka.

3.1.3 Pengenalan *Game* V-Moe

Game yang akan dibangun ini berjudul V-Moe attack. Dalam kisahnya nanti *game* ini akan menceritakan petualangan V-chan seorang gadis yang pergi ke markas *monster* untuk membebaskan tawanan yang disandera oleh *monster*.

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

1. *System single player*
2. Karakter utama memiliki kemampuan untuk bergerak 4 arah.
3. Terdapat musuh yang dapat menyerang karakter utama dengan mencari rute terdekat dengan musuh.
4. Memiliki 3 (tiga) *level* yang berkesesuaian dengan tingkat kesulitannya
5. *Control* karakter menggunakan *keyboard*.
6. *Grafik* yang digunakan merupakan *grafik game* 2D

3.1.3.1 Storyline Game V-Moe

Storyline game V-Moe menceritakan tentang seorang anak perempuan yang bernama V-Chan yang pergi ke markas para *monster*. Suatu hari planet bumi diserang oleh para *monster* yang datang dari luar angkasa dan ingin menguasai planet bumi, lalu mereka menculik para anggota keluarga yang tinggal di bersama V-Chan.

V-Chan adalah karakter animasi seorang gadis yang menjadi karakter utama dalam *game* ini. V-Chan dengan berani pergi seorang diri ke markas besar para moster untuk membebaskan anggota keluarga yang diculik.

3.1.3.2 Gameplay Game V-Moe

Secara garis besar *game*play dalam *game* V-moe adalah menjelajahi markas para *monster* yang telah menculik keluarga V-Chan, dalam *game* mencari para anggota keluarga yang disandera. Untuk lebih jelas *gameplay* pada tiap-tiap *level* dalam *game* V-Moe yang akan dibangun seperti berikut.

1. *Gameplay level* 1

Tugas utama dalam *level* ini, pemain mengendalikan karakter V-Chan yang sedang terbang di udara dalam perjalan menuju markas para *monster alien*, dalam perjalan pemain dihadang oleh sekumpulan *monster* yang mencoba menghalangi dan menyerang pemain untuk dapat sampai ke markas para *monster alien.*

Pemain harus dapat menyelesaikan misi ini untuk dapat melanjutkan ke *level* berikutnya, pemain bisa menyerang para *monster* dengan cara menembak *monster* atau mencoba menghindar dari serangan tembakan dan tabrakan dari *monster* itu sendiri.

Setiap menghancurkan *monster*, maka pemain akan mendapatkan skor atau poin. Setelah menyelesaikan misi pada *level* ini pemain akan langsung masuk ke *level* berikutnya.

Tipe dari permainan ini adalah *up-down shooter* dimana karakter pemain dapat digerakan ke kanan, ke kiri, ke atas, ke dan kebawah untuk dapat menghindar dari serangan *monster*.

2. *Gameplay level* 2

Dalam *level* ini, pemain berada dalam markas *monster*, tugas utama dalam *level* ini adalah pemain harus membebaskan anggota keluarga yang ditahan oleh para *monster*. Untuk dapat membebaskan para tawanan yang disandera dalam sebuah sel tahanan, pemain harus mengumpulkan kunci-kunci yang tersebar dalam arena permainan. Akan ada *monster* yang menghadang dalam l*evel* ini, tapi kepintaran musuh dalam *Level* ini masih sederhana, yaitu hanya berpatroli di sekitar tawanan.

Apabila karakter pemain bertabrakan dengan musuh maka karakter pemain akan mati dan harus mengulang dari awal permainan.

Setelah menemukan dan mengumpulkan semua kunci yang ada, maka pemain dapat melanjutkan ke *level* berikutnya.

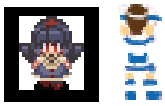
3. *Gameplay level* 3

Tugas utama dalam *level* ini hampir sama dengan *level* ke-2, tetapi AI *enemy* pada *level* ini lebih susah. Karakter *Enemy* akan menyerang pemain dengan mengejar ke arah karakter pemain. Pemainan akan selesai ketika semua anggota keluarga telah diselamatkan oleh V-chan.

3.1.3.3 Pengenalan Karakter V-Moe

Berikut ini adalah pengenalan karakter yang ada dalam *game* V-Moe, yang dapat dilihat pada tabel 3.3:

Tabel 3.3 Konten *Game* V moe

No	Nama	Gambar	Penjelasan
1	V-Chan		Karakter utama dalam *game* ini, user menggerakkan dengan input keyboard. Dalam *Level* 1 V-Chan memiliki jumlah *life* 3 dan *healthbar*.
2	*Monster* 1		Karakter disamping adalah karakter musuh yang akan menghadang pemain dalam *level* 3, *monster* ini akan bergerak mendekati pemain dengan mencari jalan terpendek dalam arena permainan.
3	*Monster* 2		*Monster* disamping adalah musuh yang ada dalam permainan *level* 2 yang menjaga kunci dan bergerak kekanan-kekiri
4	*Monster* 3		*Monster* disamping adalah musuh terbang yang muncul pada *level* 1
5	*Monster* 4		*Monster* disamping adalah musuh terbang yang muncul pada *level* 1
6	*Monster* 5		*Monster* disamping adalah musuh terbang yang muncul pada *level* 1 dan mempunyai kemampuan menembak peluru ke arah *player*.
7	*Monster* 6		*Monster* disamping adalah

			musuh yang ada dalam permainan *level* 2 yang bergerak mengelilingi dalam arena permainan.
8	Kunci		Kunci adalah item yang harus pemain dapatkan untuk dapat membuka gembok dan membebaskan sandra.
9	Pintu		Pintu merupakan gerbang menuju *level* berikutnya setelah pemain mendapatkan kunci dan membebaskan tawanan.
10	Koin		Koin adalah *item* untuk mendapatkan poin dalam *game* ini, setiap *player* mendapatkan koiin maka nilai *score* akan bertambah 10.
11	Lock		Adalah gembok yang akan terbuka ketika setelah *player* mendapatkan *item* kunci yang ada dalam arena permainan.
12	Warga		Adalah keluarga yang harus diselamatkan oleh v-chan dalam permainan *evel* 2 dan *evel* 3.
13	Item Nyawa		*Item* ini berfungsi untuk menambahkan jumlah nyawa yang dimiliki oleh karakter utama, item ini akan terdapat pada game level 2 dan 3.

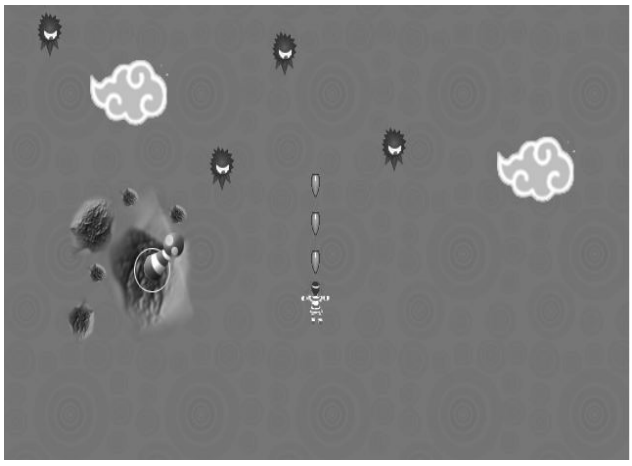
3.1.3.4 Storyboard Game V-Moe

Storyboard adalah sketsa gambar yang disusun berurutan sesuai dengan naskah, dengan *Storyboard* kita dapat menyampaikan ide cerita kita kepada orang lain dengan lebih mudah, karena kita dapat menggiring khayalan seseorang

mengikuti gambar-gambar yang tersaji, sehingga menghasilkan persepsi yang sama pada ide cerita kita.

1. *Storyboard Level 1*

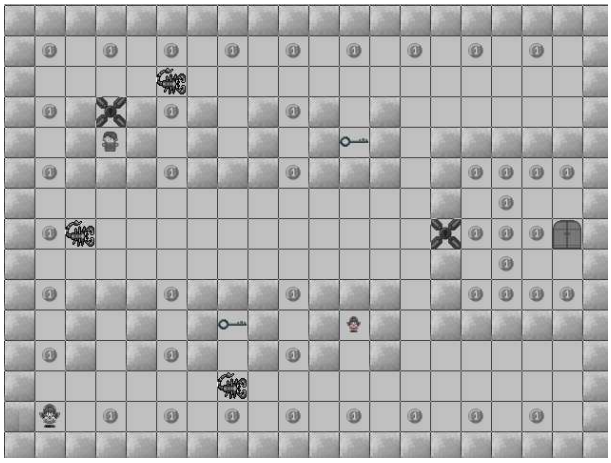
Storyboard Level 1 pada *game* V-Moe dapat dilihat sebagai berikut :



Gambar 3.3 *Storyboard Game V-Moe Level 1*

2. Storyboard Level 2

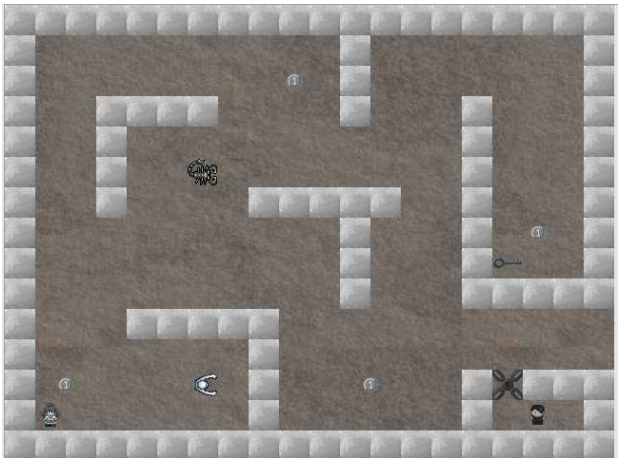
Storyboard Level 2 pada *game* V-Moe dapat dilihat sebagai berikut :



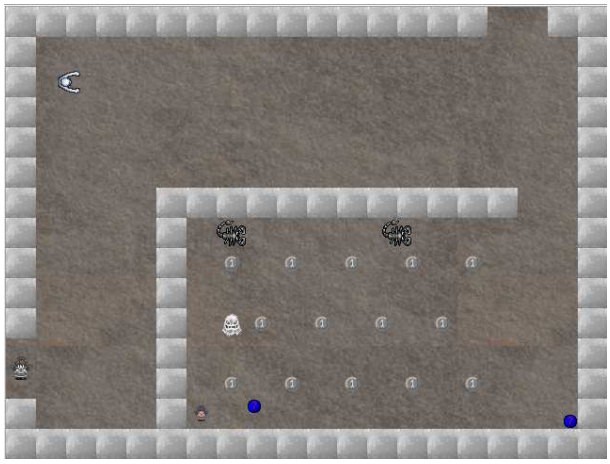
Gambar 3.4 *Storyboard Game V-Moe Level 2*

3. Storyboard Level 3

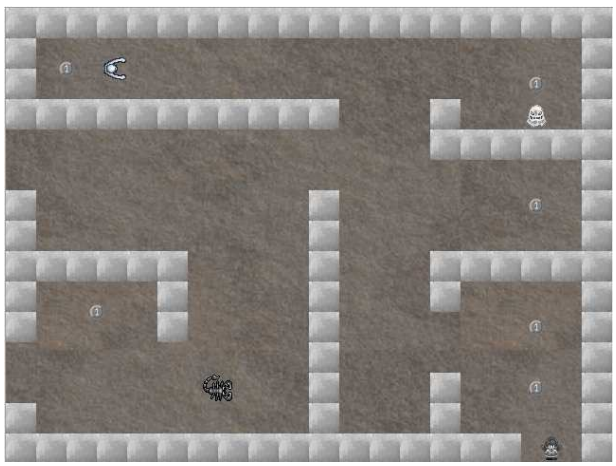
Storyboard Level 3 pada *game* V-Moe dapat dilihat sebagai berikut :



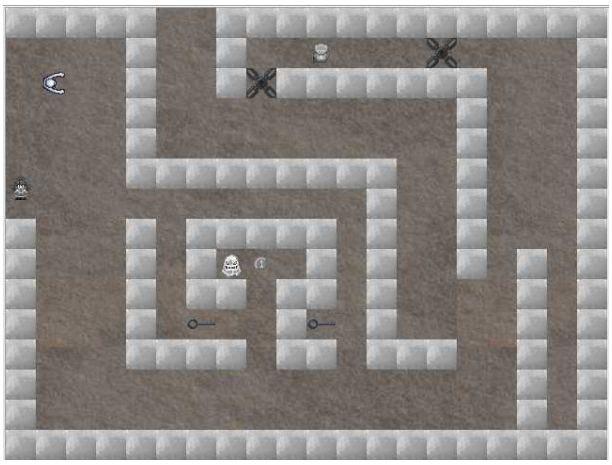
Gambar 3.5 ***Storyboard Game V-Moe Level 3 Frame 1***



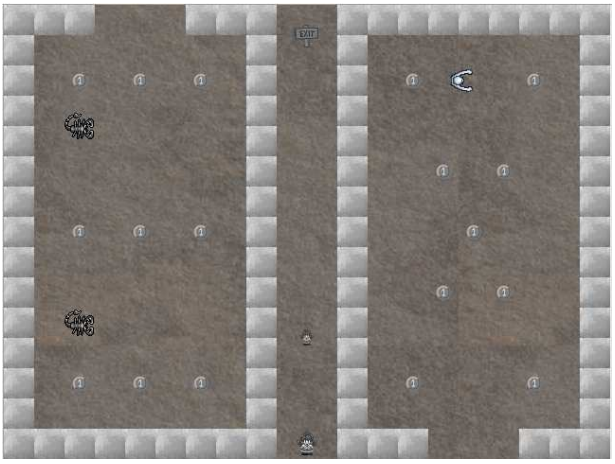
Gambar 3.6 ***Storyboard Game V-Moe Level 3 Frame 2***



Gambar 3.7 ***Storyboard Game V-Moe Level 3 Frame 3***



Gambar 3.8 ***Storyboard Game V-Moe Level 3 Frame 4***



Gambar 3.9 ***Storyboard Game V-Moe Level 3 Frame 5***

3.1.3.5 Perbandingan Game Petualangan Kungi dengan V-Moe

Dari penjelasan diatas maka dapat disimpulkan beberapa perbandingan *Game* Petualangan Kungi dengan *Game* V-Moe yang akan dibangun, yang dijelaskan dalam tabel dibawah ini :

Tabel 3.4 Perbandingan *game*

No	Aspek	Petualangan Kungi	V-Moe
1	*Game*play	Hanya memiliki 1 jenis *gameplay*	Mempunyai 2 jenis *gameplay*
2	Konten AI	*Enemy* bergerak statis	*Enemy* bergerak statis dan dinamis (menerapkan A*)
3	NPC	Tidak ada	Ada

3.1.4 Analisis Algoritma A*

Algoritma A* (A Star) merupakan perbaikan dari metode BFS dengan memodifikasi fungsi heuristiknya. Algoritma ini melakukan pencarian jalur terpendek dengan mencari biaya dari penjumlahan biaya sebenarnya ditambah dengan biaya perkiraan. Dalam notasi matematika dapat dituliskan : $f(n) = g(n) + h(n)$. Pada kondisi yang tepat, A* akan memberikan solusi yang terbaik dalam waktu yang optimal.

Sama seperti algoritma BFS, algoritma A* ini juga menggunakan dua senarai (*list*) : *OPEN* dan *CLOSED*. Terdapat tiga kondisi bagi setiap suksesor yang dibangkitkan, yaitu : sudah berada di *OPEN*, sudah berada di *CLOSED* , dan tidak berada di *OPEN* maupun di *CLOSED*.

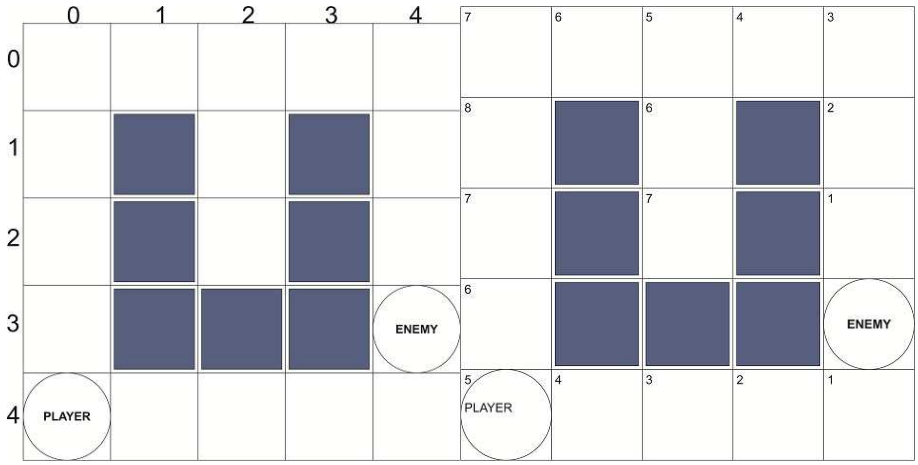
Jika suksesor sudah pernah berada di *OPEN*, maka lakukan pengecekan apakah perlu pengubahan *parent* atau tidak tergantung nilai g-nya melalui *parent* lama atau *parent* baru. Jika melalui *parent* baru memberikan nilai g yang lebih kecil, maka dilakukan pengubahan *parent*, jika pengubahan *parent* dilakukan, maka dilakukan pula pembaruan nilai g dan nilai f pada suksesor tersebut. Dengan perbaruan ini, suksesor tersebut memiliki kesempatan yang lebih besar untuk terpilih sebagai simpul terbaik (*bestnode*)

Jika suksesor sudah pernah berada di *CLOSED*, maka dilakukan pengecekan apakah perlu pengubahan *parent* atau tidak. Jika ya, maka lakukan pembaruan nila g dan f pada suksesor tersebut serta pada semua "anak cucunya" yang sudah pernah berada di *OPEN*. Dengan perbaruan ini maka anak cucunya terssebut memiliki kesempatan lebih besar untuk terpilihn sebagai simpul terbaik(*bestnode*).

Jika suksesor tidak berada di *OPEN* maupun *CLOSED*, maka suksesor tersebut dimasukkan ke dalam *OPEN*. Tambahkan suksesor tersebut sebagai suksesornya *bestnode*. Hitung biaya suksesor tersebut dengan rumus *f=g+h*

3.1.4.1 Penerapan A Pada Game V-Moe*

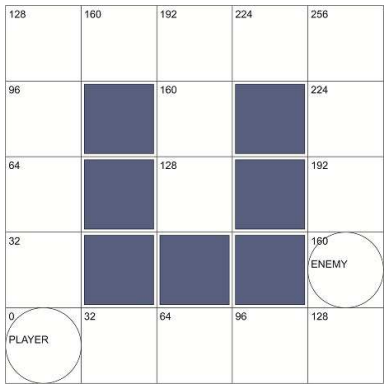
Berikut contoh penerapan algoritma A* pada *game* v-moe yang akan dbangun :



Gambar 3.10 Kondisi Awal dan Nilai G pada Map

Pada gambar dapat dilihat kondisi *nodeEnemy* berada pada posisi di atas, dan posisi *nodeplayer* berada di bawah *nodeEnemy* dan terhalang oleh dinding pada *game*. Lalu dilakukan pemberian biaya sebenarnya (g).Nilai heuristic atau h

didapat dari jarak perkiraan dari *nodeplayer* menuju *nodeEnemy*. Untuk lebih jelas dapat dilihat pada gambar dibawah ini :

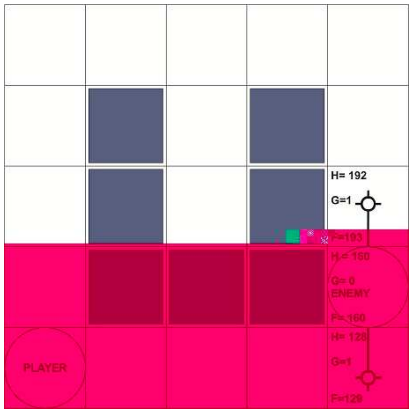


Gambar 3.11 Pemberian Nilai Heuristic

Dalam pencarian jalur terpendek dengan menggunakan algoritma A* akan terdapat 3 nilai, yaitu *g*, *h*, dan *f*. nilai *g* adalah biaya sebenarnya jarak dari *nodeEnemy* ke *nodeplayer*, nilai *h* adalah biaya perkiraan, dan *f* adalah total biaya yang diperhitungkan dengan menjumlahkan nilai *g* dan *h*. Notasi matematika untuk mendapatkan nilai h:

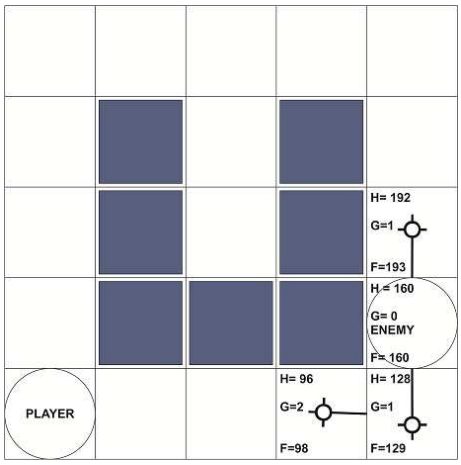
Persamaan 3.1 h = abs(x.x-destination.x) + abs(y.y-destination.y).

Dibawah ini langkah-langkah pencarian jalur terpendek menggunakan algoritma A* yang diterapkan pada *game* seperti gambar dibawah ini.



Gambar 3.12 Langkah 1 pencarian algoritma A*

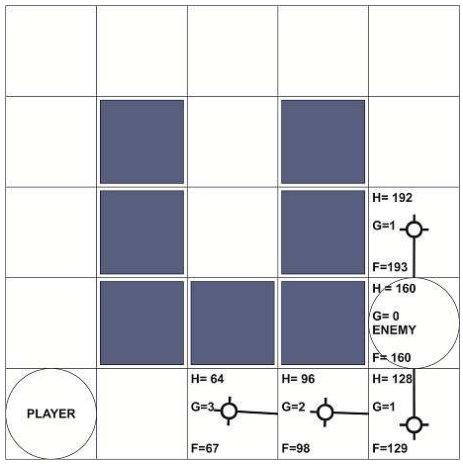
Dari *nodeenemy* akan melakukan pencarian jalur terpendek dengan membuka semua kemungkinan pergerakan secara *horizontal* untuk menuju ke arah *nodeplayer*. Pertama-tama pencarian ke arah verical atas, nilai g = 1, h =192, dan f =193. Karena disebelah kiri *node enemy* adalah penghalang yang tidak bisa dilewati, maka pencarian dilakukan ke arah *vertikal* bawah. Nilai g pada posisi ini adalah 1, nilai h = 128, sehingga nilai f= g+h= 129. Setelah itu dilakukan pengecekan *bestnode* untuk menjadi posisi baru dengan mencari nilai terkecil dari 2 kemungkinan langkah tadi, setelah dibandingkan didapat bahwa langkah vertikal bawah adalah *bestnode*. Posisi *node enemy* yang sebelumnya berada pada posisi awal sekarang pindah ke posisi *bestnode* untuk pencarian berikutnya. langkah berikutnya dalam pencarian jalur terpendek dengan algoritma A* dapat dilihat gambar dibawah ini :



Gambar 3.13 Langkah 2 pencarian algoritma A*

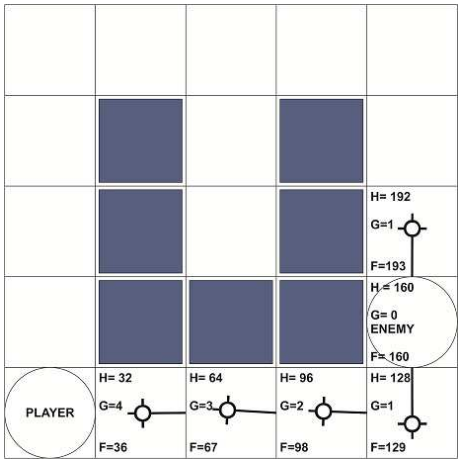
Selanjutnya setelah posisi *nodeEnemy* pindah ke *bestnode* yang baru yaitu pada f= 129, maka dari posisi ini *node Enemy* melakukan kembali pencarian dengan membuka kemungkinan jalur yang dapat dilalui, karena sebelah kanan

node Enemy terdapat penghalang sehingga kemungkinan jalur yang dapat dilalui hanya ke arah horizontal kiri dengan nilai g=2, nilai h=96, sehingga nilai f=98. Selanjutnya dilakukan kembali pengecekan *bestnode* denga *node* sebelumnya. Didapat bahwa *node* yang baru merupakan *bestnode* dengan f=96. Sehingga posisi baru menjadi *bestnode* dan posisi *node Enemy* pindah ke posisi ini. Setelah itu dilanjutkan langkah berikutnya seperti gambar dibawah ini.



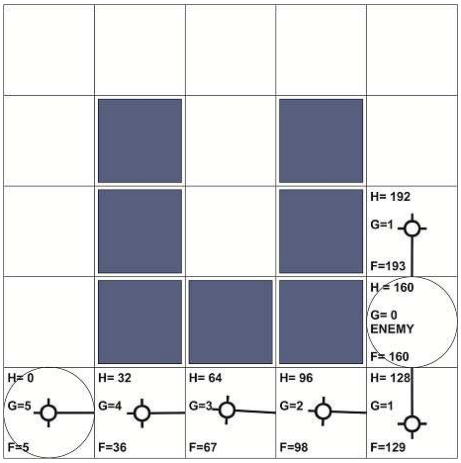
Gambar 3.14 Langkah 3 pencarian algoritma A*

Selanjutnya pencarian dimulai dari current_*node* yang baru kemudian pencarian dimulai kembali dengan mencari kemungkinan jalur yang dapat dilalui, karena pada posisi vertical atas terdapat penghalang sehingga kemungkinan jalur yang ada hanya horizontal kiri, Karena hanya satu-satunya *node* yang diperiksa, maka pencarian selanjutnya dimulai dari sini dengan nilai g = 3, h =64, dan f = g + h = 67. Setelah itu dilakukan kembali pengecekan posisi *bestnode* dengan nilai yang paling kecil dengan membandingkan *node* sebelumnya didapatkan hasil bahwa current_*node* mempunyai nilai terkecil sehingga posisi ini merupakan *bestnode* dan posisi *Enemy* pindah keposisi ini. Selanjutnya pecarian algoritma A* dilakukan pada langkah ke 4 seperti gambar dibawah ini :



Gambar 3.15 Langkah 4 pencarian algoritma A*

Selanjutnya pencarian dimulai dari current_*node* yang baru kemudian pencarian dimulai kembali dengan mencari kemungkinan jalur yang dapat dilalui, karena pada posisi vertical atas terdapat penghalang sehingga kemungkinan jalur yang ada hanya horizontal kiri, Karena hanya satu-satunya *node* yang diperiksa, maka pencarian selanjutnya dimulai dari sini dengan nilai g = 4, h =32, dan f = g + h = 36. Setelah itu dilakukan kembali pengecekan posisi *bestnode* dengan nilai yang paling kecil dengan membandingkan *node* sebelumnya didapatkan hasil bahwa *node* yang baru mempunyai nilai terkecil sehingga posisi ini merupakan *bestnode* dan posisi *Enemy* pindah keposisi ini. Selanjutnya pecarian algoritma A* dilakukan pada langkah ke 5 seperti gambar dibawah ini :



Gambar 3.16 Langkah 5 pencarian algoritma A*

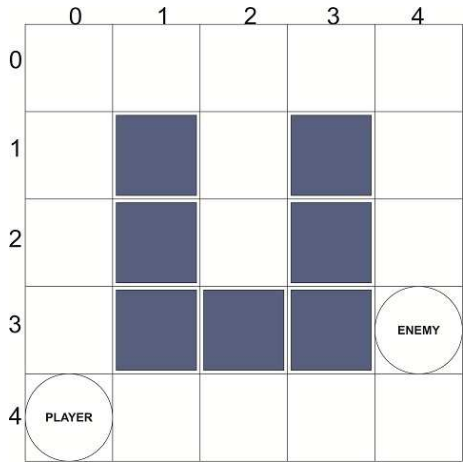
Pada posisi ini pencarian dilakukan kembali, kemungkinan jalur yang ada adalah ke arah *horizontal* kiri, dengan nilai g= 5, h=0 sehingga f=5. Setelah itu dilakukan pengecekan apakah current_*node=goal*, setelah dilakukan didapatkan bahwa current_*node=goal* sehingga pencarian selesai.

Dari *nodeEnemy* menuju *nodeplayer* maka didapatkan jalur terpendek dengan menggunakan algoritma A* seperti berikut :

1. Dari *nodeEnemy* mengambil jalur ke arah vertical bawah dengan nilai f = 129.
2. Lalu dilanjutkan mengambil jalur horizontal kiridengan nilai f = 98.
3. Kemudian dilanjutkan kembali ke arah horizontal kiri dengan nilai f =67.
4. Setelah itu dilanjutkan dengan mengambil jalur horizontal kiri dengan nilai f=36.

Pencarian selesai setelah didapat current_*node = goal* yaitu pada posisi *node player* dengan nilai f = 5.

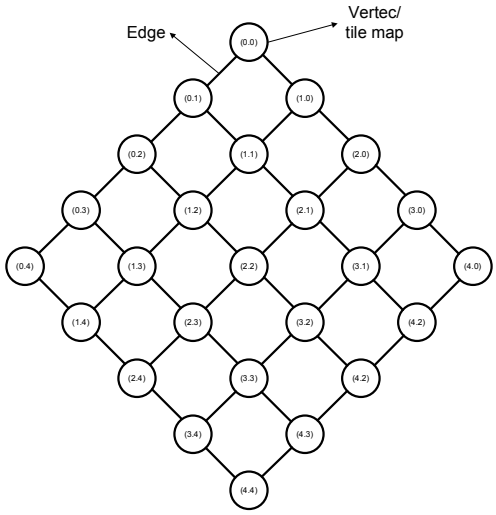
Game V-Moe adalah *game* yang tersusun atas kumpulan tile-tile yang saling terhubung dan memiliki ordo X x Y, sehingga berbentuk array dua dimensi.



Gambar 3.17 Arena permainan *game* V-Moe

Struktur data yang digunakan dalam *game* V-Moe dapat direpresentasikan oleh *graph* karena mempunyai hubungan antara objek dengan objek lain. Struktur data graph adalah struktur data yang berbentuk network/ jaringan, hubungan antar elemenya adalah *many - to – many.*

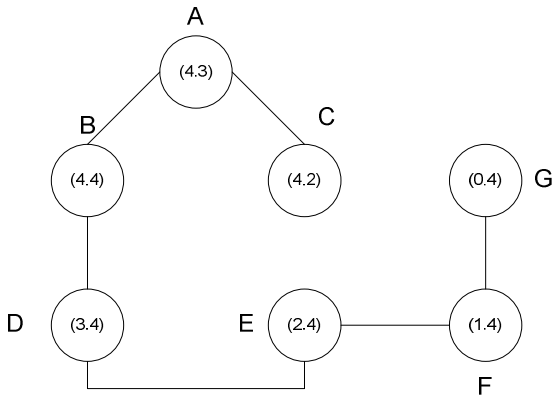
Graph terdiri dari *Node (Vertex)* dan ARC *(Edge). Node* adalah elemen graph yang berisi informasi sedangkan ARC *(Edge)* adalah *path* yang menghubungkan dua buah *node*.



Gambar 3.18 Arena permainan dengan graph pada *game* V-Moe

Secara umum terdapat dua macam representasi dari struktur data graph yang dapat diimplementasikan. Pertama, disebut *adjacency list*, yaitu diimplementasikan dengan menampilkan masing-masing simpul sebagai sebuah struktur data yang mengandung senarai dari semua simpul yang saling berhubungan. Yang kedua adalah representasi berupa *adjacency matrix* dimana baris dan kolom dari matriks tersebut merepresentasikan simpul awal dan simpul tujuan dan sebuah entri di dalam senarai yang menyatakan apakah terdapat sisi di antara kedua simpul tersebut.

Pada *game* V-Moe ini mengunakan representasi struktur data graph *adjacency matrix* dimana baris dan kolom dari matriks tersebut merepresentasikan simpul awal dan simpul tujuan dan sebuah entri di dalam senarai yang menyatakan apakah terdapat sisi di antara kedua simpul tersebut.



Gambar 3.19 Graph Pencarian Jalur Terpendek Pada *Game* V-Moe

Setelah itu *node-node* dirubah kedalam bentuk *adjacency matrix,* berikut adalah bentuk matrix dari *node* pencarian rute menuju *player* :

Tabel 3.5 *Adjacency Matrix Game V-Moe*

	A	B	C	D	E	F	G
A	0	1	1	0	0	0	0
B	1	0	0	1	0	0	0
C	1	0	0	0	0	0	0
D	0	1	0	0	1	0	0
E	0	0	0	1	0	1	0
F	0	0	0	0	0	0	1
G	0	0	0	0	0	1	0

Setelah itu dilakukan proses pecarian jalur terpendek dengan memeriksa simpul-simpul yang berhubungan dari objek musuh yang berada pada posisi (4,3). Pada saat dicek, ternyata ada 2 simpul yang dapat dilalui oleh objek musuh yaitu simpul (4,2) dan simpul (4,4), selanjutnya dicek simpul mana yang mempunyai nilai f paling kecil. Setelah dilakukan perhitungan maka siperoleh simpul (4,4) mempunyai nilai f paling kecil. Sehingga objek musuh pindah ke posisi simpul (4,4), setelah berada pada posisi (4,4) pengecekan nilai di hitung kembali sampai proses ditemukan posisi simpul = posisi *player*. Sehingga ditemukan bahwa objek *player* berada pada posisi simpul (0,4) dan pencarian jalur terpendek selesai.

Tabel 3.6 Simpul yang diperiksa dalam pencarian jalur terpendek

Langkah	*OPEN*	*CLOSED*
1	[(4,2),(4,4)]	[(4,3)]
2	[(3,4)]	[(4,3)] ,(4,4)]
3	[(2,4)]	[(4,3)] ,(4,4) ,(3,4)]
4	[(1,4)]	[(4,3)] ,(4,4) ,(3,4),(2,4)]
5	[(0,4)]	[(4,3)] ,(4,4) ,(3,4), (2,4)] ,(1,4)]

3.1.5 Analsis Kebutuhan *Non-Fungsional*

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

3.1.5.1 Analisis Kebutuhan User

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

Tabel 3.7 Analisis Kebutuhan User

1	Peran	Pemain
2	Usia	Anak berusia lebih dari 8 tahun atau dewasa dan menyukai *game.*
3	Tingkat Keterampilan	Dapat membaca dengan baik mengoperasikan komputer, menggunakan *keyboard* dan *mouse.*
4	Pengalaman	Pernah memainkan aplikasi *game* komputer atau mengoperasikan komputer.

3.1.5.2 Analisis Kebutuhan Hardware (Developer)

Perangkat keras yang digunakan dalam membangun *game* V-Moe ini adalah sebagai berikut:

Tabel 3.8 Spesifikasi Hardware *(Developer)*

Nama Perangkat	**Spesifikasi**
Processor	Dual Core 2,8 Ghz
Memory	DDR3 2GB
HDD	80 GB
Display	Resolusi 1360 x 768 VRAM 1GB

Mouse	1 Unit
Keyboard	1 Unit

3.1.5.3 Analisis kebutuhan Hardware (User)

Agar aplikasi dapat berjalan dengan baik, maka dibutuhkan perangkat keras yang sesuai dengan kebutuhan aplikasi. Spesifikasi minimum perangkat keras yang dibutuhkan agar dapat membangun *game* V-Moe ini adalah sebagai berikut:

Tabel 3.9 Spesifikasi Hardware *(User)*

Nama Perangkat	**Spesifikasi**
Prosesor	Prosessor dengan Kecepatan 1,8Ghz
RAM	512Mb
Harddisk	1Gb
Display	Resolusi 1024 x 768
Mouse	USB Standard
Keyboard	USB Standard

3.1.5.4 Analisis kebutuhan Software (Developer)

Perangkat lunak yang dipergunakan dalam pembuatan aplikasi yang akan dibangun menggunakan *software* dan *tools* sebagai berikut :

Tabel 3.10 Spesifikasi Perangkat Lunak *(Developer)*

Nama Perangkat Lunak	**Jenis**
Sistem Operasi	Microsoft Wndows XP Prof SP3
Utilitas	*Game maker plugins*
SDK	*Game* Maker 8.0

3.1.5.5 Analisis kebutuhan Software (User)

Ada beberapa hal penting yang harus diperhatikan agar sistem yang akan dirancang dapat berjalan dengan baik dan sesuai dengan yang diharapkan. Pengguna diharapkan dapat memenuhi kebutuhan perangkat lunak yang

telah direkomendasikan agar aplikasi dapat berjalan dengan baik. *Game* V-moe memiliki kebutuhan minimal perangkat lunak sebagai berikut :

Tabel 3.11 Spesifikasi Perangkat Lunak *(User)*

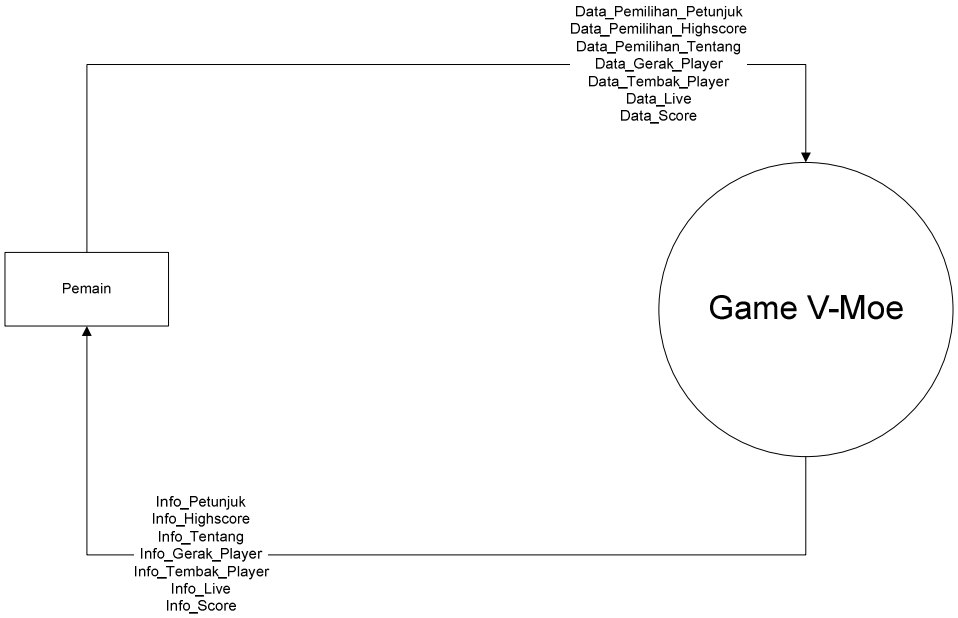
Nama Perangkat Lunak	**Jenis**
Sistem Operasi	Microsoft Wndows XP Prof SP3
Utilitas	*Game maker plugins*
SDK	*Game* Maker 8.0

3.1.6 Analisis Kebutuhan Fungsional

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

3.1.6.1 Diagram Konteks

Diagram konteks adalah alat struktur analisis yang menggambarkan hubungan dan keterkaitan suatu sistem dengan entitas serta aliran data dari entitas menuju sistem dan dari sistem menuju entitas. Berikut adalah gambar DFD dari pembangunan aplikasi *game* V-Moe :



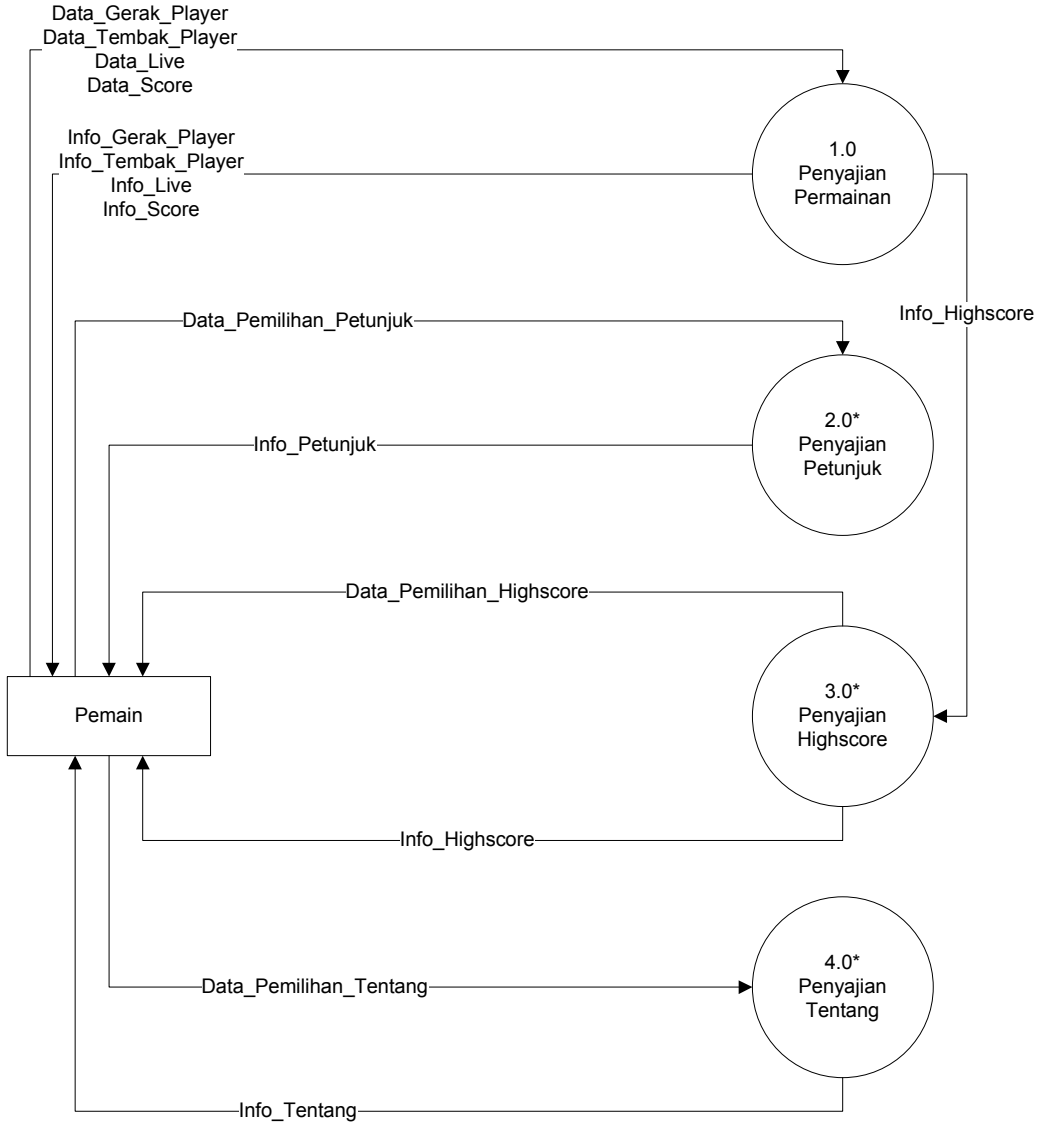
Gambar 3.20 Diagram konteks *game* V-Moe

3.1.6.2 Data Flow Diagram

Data Flow Diagram (DFD) adalah suatu model logika data atau proses yang dibuat untuk menggambarkan dari mana asal data dan kemana tujuan data yang keluar dari sistem, dimana data disimpan, proses apa saja yang menghasilkan data tersebut dan interaksi antara data yang disimpan dan proses yang dikenakan pada data tersebut.

3.1.6.3 DFD Level 1

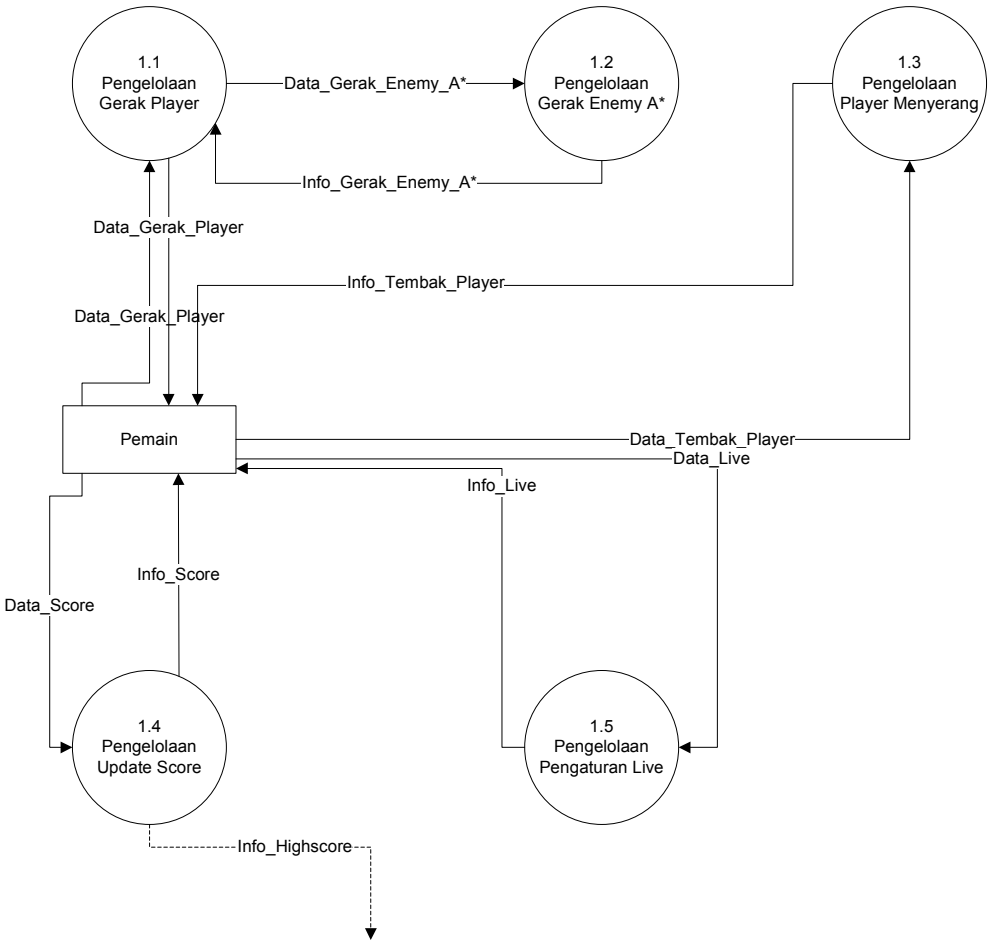
Data Flow Diagram (DFD) *Level* 1 mempunyai 4 proses, diantaranya adalah proses penyajian permainan berjalan, penyajian petunjuk, penyajian high*score* dan proses penyajian tentang. Berikut adalah gambar DFD *Level* 1 aplikasi *game* V-Moe :



Gambar 3.21 DFD level 1 *game* V-Moe

3.1.6.4 DFD Level 2 Proses 1 Proses Penyajian Permainan

Data Flow Diagram *Level* 2 proses 1, proses penyajian permainan



Gambar 3.22 DFD level 2 proses 1 mulai permainan

3.1.6.5 Spesifikasi Proses

Spesifikasi proses digunakan untuk menjelaskan lebih lanjut tentang proses-proses yang ada di dalam diagram alir data atau DFD. Spesifikasi proses merupakan deskripsi dari setiap elemen proses yang terdapat dalam aplikasi, yang meliputi nama proses, masukan, keluaran dan keterangan dari proses. Adapun spesifikasi proses untuk diagram alir data aplikasi *game* V-Moe dapat dilihat pada tabel berikut :

Tabel 3.12 Spesifikasi Proses

No	Proses	Keterangan
1	No. Proses	1
	Nama Proses	Penyajian Permainan
	Deskripsi	Proses untuk memulai permainan.
	Source	Pemain
	Input	Data_Gerak_*Player*,Data_Tembak_*Player*,Data_ *Life,* Data_Tambah_*Score*
	Output	Info_Gerak_*Player,* Info_Tembak_*Player,* Info_Tambah_*Life,* Info_Tambah_*Score*
	Destination	Pemain
	Logika Proses	1. Pemain memilih menu mulai permainan untuk memulai aplikasi permainan. 2. Sistem akan menampilkan arena permainan. 3. Jika pemain berhasil melewati level 1 4. Maka akan tampil arena permainan level 2 5. Jika tidak maka permainan selesai dan tampil *score* permainan. 6. Jika pemain berhasil melewati level 2 7. Maka akan tampil arena permainan level 3 8. Jika tidak maka permainan selesai. 9. Jika pemain berhasil menyelesaikan level 3 10. Maka permainan berakhir
2	No. Proses	2
	Nama Proses	Penyajian Petunjuk
	Deskripsi	Proses untuk melihat petunjuk permainan.
	Source	Pemain
	Input	Data pemilihan petunjuk
	Output	Info petunjuk
	Destination	Pemain
	Logika Proses	1. Pemain memilih menu petunjuk permainan. 2. Sistem akan menampilkan info petunjuk permainan.
3	No. Proses	3.
	Nama Proses	Penyajian *Highscore*
	Deskripsi	Proses untuk melihat *Highscore*
	Source	Pemain
	Input	Data pemilihan *Highscore*
	Output	Info *Highscore*
	Destination	Pemain
	Logika Proses	1. Pemain memilih menu *highscore* 2. Sistem akan menampilkan informasi *highscore* permainan.
4	No. Proses	4
	Nama Proses	Penyajian Tentang
	Deskripsi	Proses untuk melihat informasi permainan

	Source	Pemain
	Input	Data pemilihan tentang
	Output	Info tentang
	Destination	Pemain
	Logika Proses	1. Pemain memilih menu tentang 2. Sistem akan menampilkan informasi aplikasi permainan.
5	No. Proses	1.1
	Nama Proses	Pengelolaan Gerak *Player*
	Deskripsi	Proses untuk menampilkan gerakan karakter
	Source	Pemain
	Input	Data gerak *player*
	Output	Info gerak *player*
	Destination	Pemain
	Logika Proses	1. Pemain menggerakkan karakter dengan menekan tombol pada keyboard. 2. Sistem akan menampilkan gerak karakter utama dalam permainan.
6	No. Proses	1.2
	Nama Proses	Pengelolaan Gerak *Enemy* A*
	Deskripsi	Proses untuk menampilkan gerakan *Enemy* A*
	Source	Pemain
	Input	Data gerak *Enemy* A*
	Output	Info gerak *Enemy A**
	Destination	Sistem
	Logika Proses	1. Jika pemain bergerak dalam arena permainan 2. Objek musuh mencari rute terpendek menuju pemain dengan menggunakan algoritma A* 3. Maka sistem akan menampilkan gerak musuh mengejar karakter utama dalam permainan.
7	No. Proses	1.3
	Nama Proses	Pengelolaan *Player* Menyerang
	Deskripsi	Proses untuk menampilkan menyerang pada karakter
	Source	Pemain
	Input	Data tembak *player*
	Output	Info tembak *player*
	Destination	Pemain
	Logika Proses	1. Pemain memasukkan input tombol pada *keyboard* 2. Sistem akan menampilkan objek peluru dalam permainan.
8	No. Proses	1.4
	Nama Proses	Pengelolaan update *score*
	Deskripsi	Proses untuk pengelolaan *score* permainan
	Source	Pemain

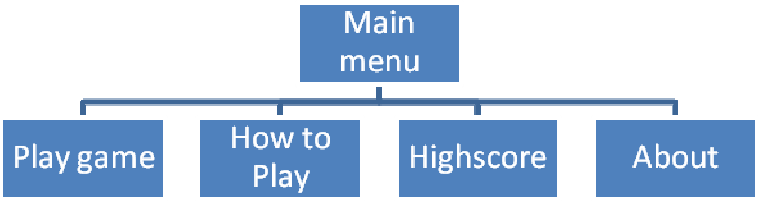
	Input	Data *score*
	Output	Info *score*
	Destination	Pemain
	Logika Proses	1. Jika posisi karakter pemain sama dengan item coin dalam permainan. 2. Maka sistem menambahkan *score* sebanyak 10 poin 3. Jika tidak, maka *score* tetap.
9	No. Proses	1.5
	Nama Proses	Pengelolaan Pengaturan *life*
	Deskripsi	Proses untuk menambah *life* pada karakter
	Source	Pemain
	Input	Data *life*
	Output	Info *life*
	Destination	Pemain
	Logika Proses	1. Jika posisi karakter pemain sama dengan item *life*. 2. Maka sistem akan menambahkan jumlah *life* sebanyak 1 pada karakter pemain dalam permainan. 3. Jika tidak maka *life* tetap.

3.2 Perancangan sistem

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

3.2.1 Perancangan Arsitektur Menu

Untuk mempermudah pembuatan sistem diperlukan perancangan struktur menu program yang akan dibangun. Perancangan struktur menu program ini membantu dalam merancang bagian-bagian dari sistem yang sebenarnya dan untuk mengetahui bagian mana yang terlebih dahulu nantinya yang akan diakses setelah program tersebut selesai. Gambar di bawah ini menunjukkan perancangan Arsitektur menu dari sistem yang akan dibangun.



Gambar 3.26 *Arsitektur Menu*

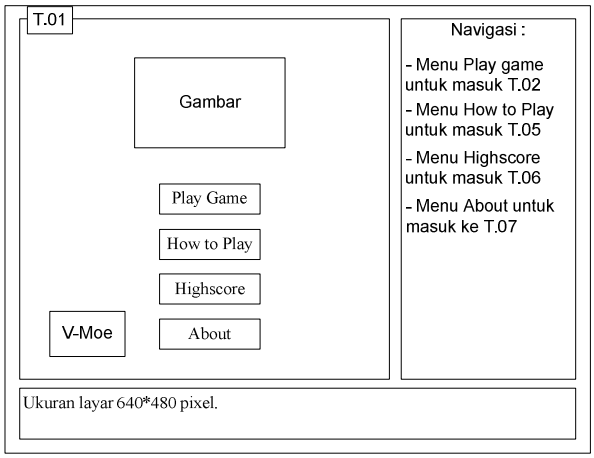
1. Menu *Play* : merupakan submenu untuk memulai permainan baru
2. Menu *How to Play* : merupakan submenu yang menampilkan informasi petunjuk *game.*
3. Menu *Highscore* : merupakan submenu yang menampilkan informasi nilai tertinggi dalam permainan.
4. Menu *About* : merupakan submenu yang menampilkan informasi *game.*

3.2.2 Perancangan Antar Muka *Game*

Perancangan antarmuka bertujuan untuk memberikan gambaran tentang aplikasi yang akan dibangun, sehingga akan mempermudah dalam mengimplementasikan aplikasi serta akan memudahkan pembuatan aplikasi. Berikut ini merupakan perancangan antar muka pada *game V-Moe Attack* :

1. Perancangan Antar Muka Menu Utama

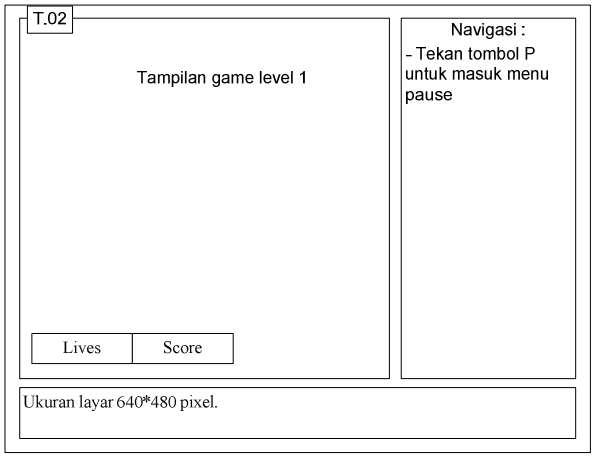
   Berikut adalah gambar perancangan antar muka menu utama *game* V-Moe yang dapat dilihat pada gambar 3.21 :



Gambar 3.23Tampilan Menu Utama

2. Perancangan Antar Muka *Level* 1

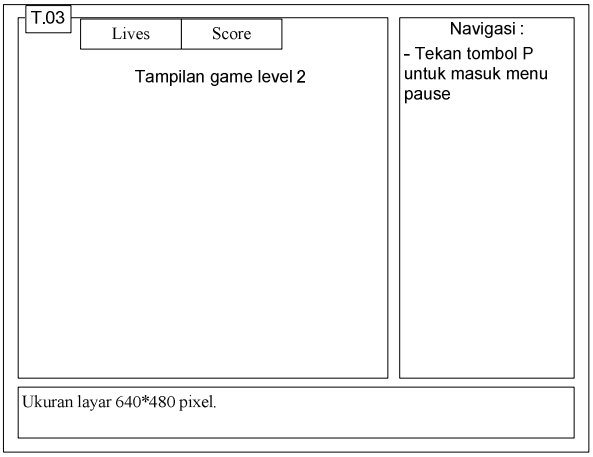
   Berikut adalah gambar perancangan antar *level* 1 utama *game* V-Moe yang dapat dilihat pada gambar 3.22 :



Gambar 3.24 Tampilan Level 1

3. Perancangan Antar Muka *Level* 2

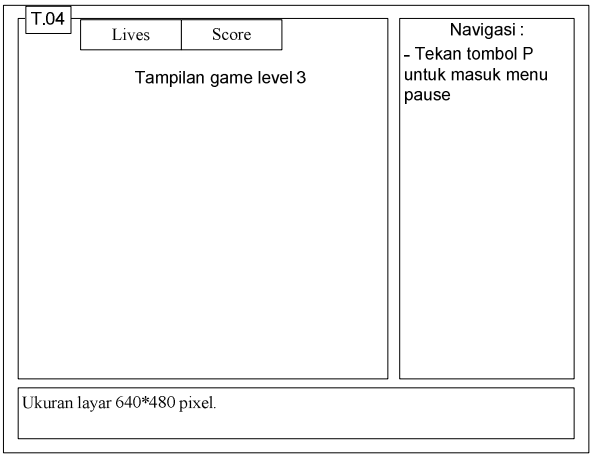
   Berikut adalah gambar perancangan antar muka *level* 2 *game* V-Moe yang dapat dilihat pada gambar 3.23 :



Gambar 3.25 Tampilan Level 2

4. Perancangan Antar Muka *Level* 3

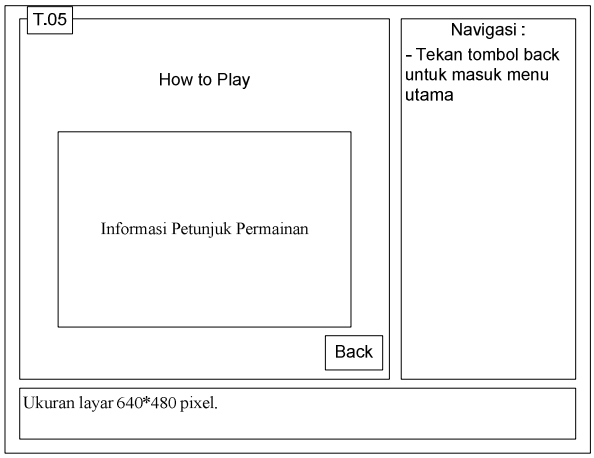
   Berikut adalah gambar perancangan antar muka *menu level* 3 *game* V-Moe yang dapat dilihat pada gambar 3.24 :



Gambar 3.26 Tampilan level 3

5. Perancangan Antar Muka Menu How to Play

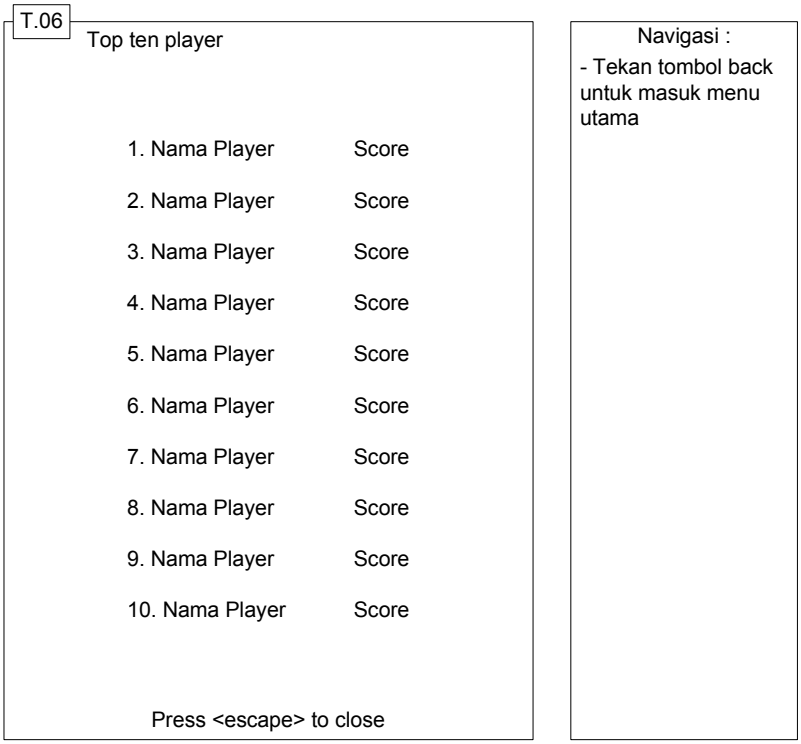
   Berikut adalah gambar perancangan antar muka menu utama *game* V-Moe yang dapat dilihat pada gambar 3.25 :



Gambar 3.27 Tampilan menu petunjuk

6. Perancangan Antar Muka *Menu Highscore*

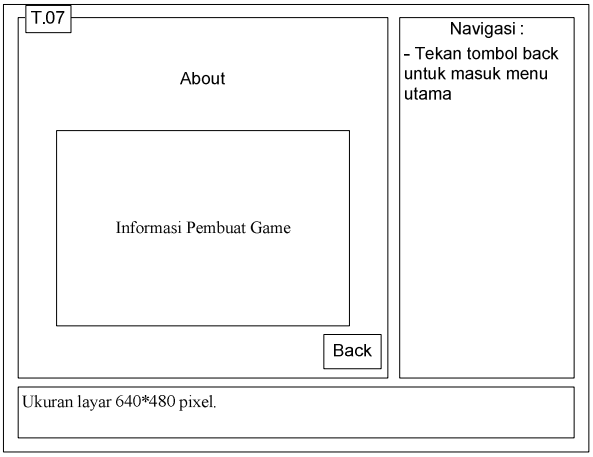
Berikut adalah gambar perancangan antar muka menu *highscore game* V-Moe yang dapat dilihat pada gambar 3.26 :



Gambar 3.28 Tampilan Menu High*score*

7. Perancangan Antar Muka Menu About

Berikut adalah gambar perancangan antar muka menu about *game* V-Moe yang dapat dilihat pada gambar 3.27 :

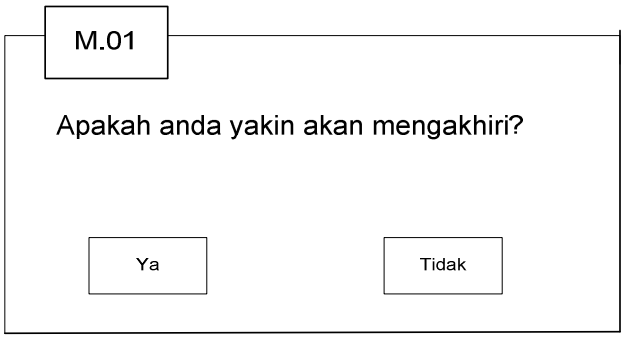


Gambar 3.29 Tampilan menu about

3.2.3 Perancangan pesan

1. Pesan Peringatan Keluar Permainan

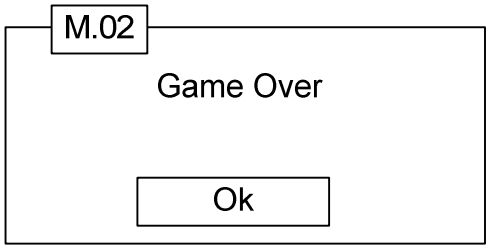
Berikut adalah gambar perancangan pesan keluar permainan *game* V-Moe yang dapat dilihat pada gambar 3.27 :



Gambar 3.30 Pesan Peringatan Keluar Permainan

2. Pesan Peringatan *Game* Over

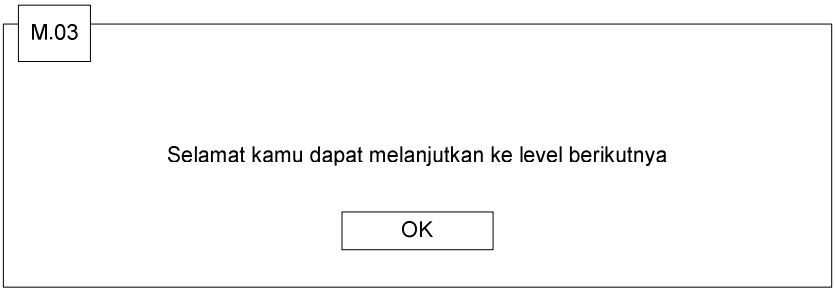
Berikut adalah gambar perancangan pesan *game over* permainan *game* V-Moe yang dapat dilihat pada gambar 3.28 :



Gambar 3.31 Pesan Peringatan *Game* Over

3. Perancangan pesan sukses pada tiap *Level*

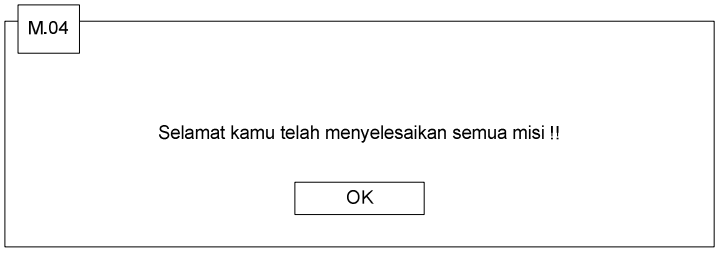
Berikut adalah gambar perancangan pesan sukses menyelesaikan *Level* permainan *game* V-Moe yang dapat dilihat pada gambar 3.29 :



Gambar 3.32 Pesan lanjut level berikutnya

4. Perancangan pesan sukses menyelesaikan *game*

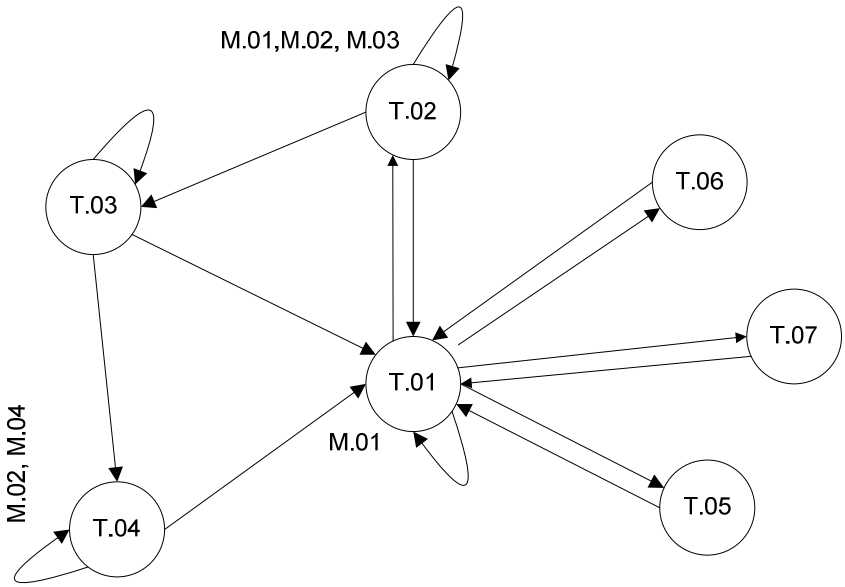
Berikut adalah gambar perancangan pesan menyelesaikan semua misi permainan *game* V-Moe yang dapat dilihat pada gambar 3.30 :



Gambar 3.33 Sukses selasai *game*

3.2.4 Jaringan Semantik

Jaringan semantik adalah gambaran tentang relasi dari masing- masing halaman yang bisa diakses oleh seorang pengguna. Jaringan semantik ini, digunakan sebagai panduan penggunaan menu-menu dalam sebuah aplikasi. Perancangan semantik ini dapat dilihat pada gambar 3.33 berikut:



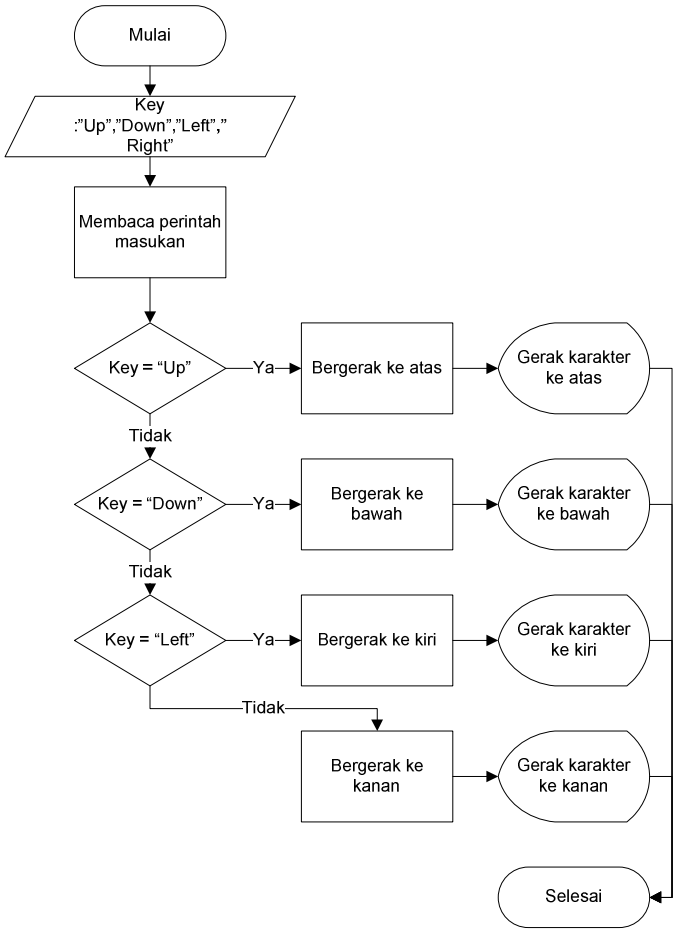
Gambar 3.34 Jaringan Semantik *Game* V-Moe

3.2.5 Perancangan *Prosedural*

Merupakan kumpulan prosedural-prosedural yang ada pada sistem yang akan dibangun dan berfungi untuk menjalankan sistem. Dan berikut ini adalah perancangan prosedural-prosedural yang berada pada sistem.

3.2.5.1 Perancangan Prosedural Gerak Player

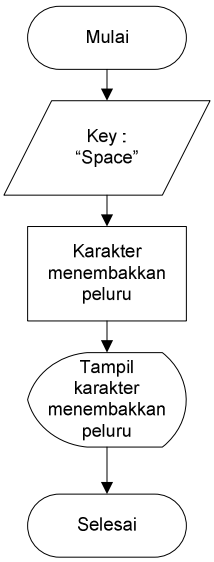
Prosedural ini dijalankan untuk membuat *player* dapat bergerak 4 arah. Untuk lebih jelasnya dapat dilihat gambar dibawah ini :



Gambar 3.35 Prosedural gerak player

3.2.5.2 Perancangan prosedural karakter menyerang

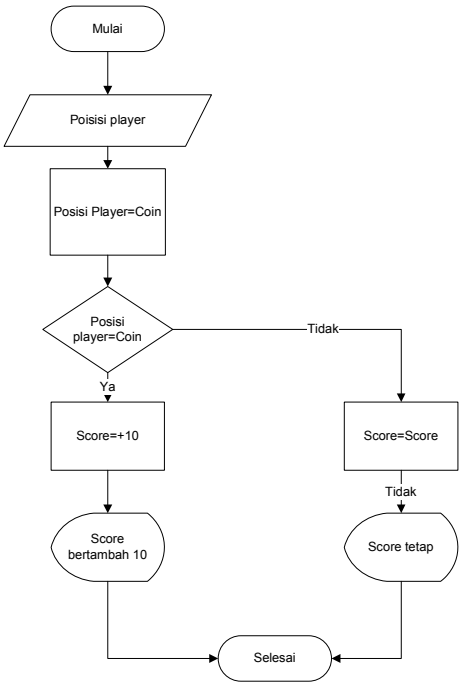
Prosedural ini dijalankan untuk membuat karakter utama dapat menyerang dengan menembak peluru. Untuk lebih jelasnya dapat dilihat gambar dibawah ini :



Gambar 3.36 Prosedural Karakter Menyerang

3.2.5.3 Perancangan Prosedural Update Score

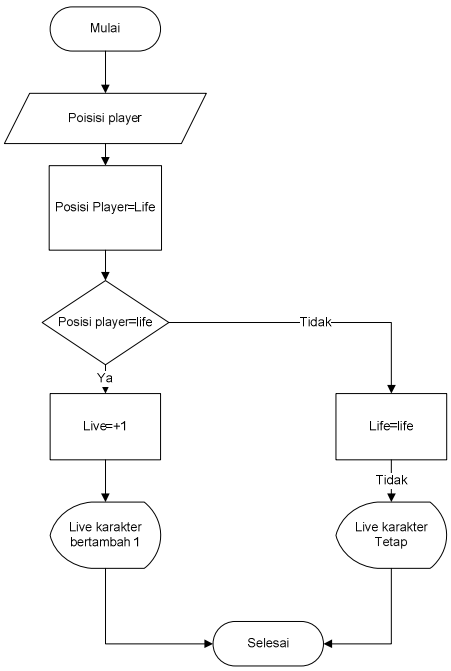
Prosedural ini dijalankan untuk membuat *player* menambah *score* ketika player mendapatkan *coin*. Untuk lebih jelasnya dapat dilihat gambar dibawah ini :



Gambar 3.37 Prosedural *Update Score*

3.2.5.4 Perancangan Prosedural Pengaturan Life

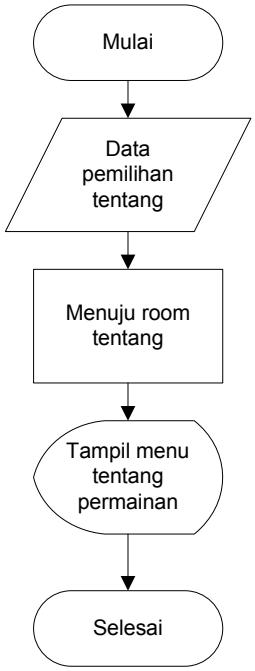
Prosedural ini dijalankan untuk menambah *life* pada *player* ketika mendapatkan item *life* dalam arena permainan . Untuk lebih jelasnya dapat dilihat gambar dibawah ini :



Gambar 3.38 Prosedural Pengaturan *Life*

3.2.5.5 Perancangan Prosedural Tampil Menu about

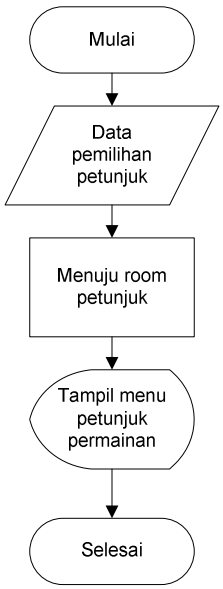
Prosedural ini dijalankan untuk menampilkan menu petunjuk. Untuk lebih jelasnya dapat dilihat gambar dibawah ini :



Gambar 3.39 Prosedural Tampil Menu About

3.2.5.6 Perancangan Prosedural Tampil Menu Petunjuk

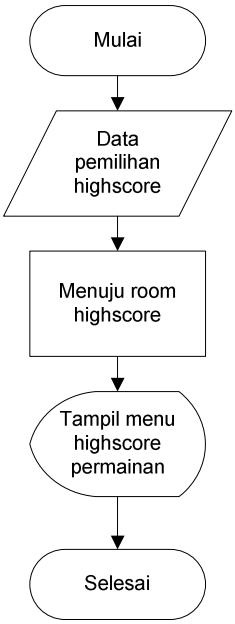
Prosedural ini dijalankan untuk menampilkan menu petunjuk. Untuk lebih jelasnya dapat dilihat gambar dibawah ini :



Gambar 3.40 Prosedural Tampil Petunjuk

3.2.5.7 Perancangan Prosedural Menu Highscore

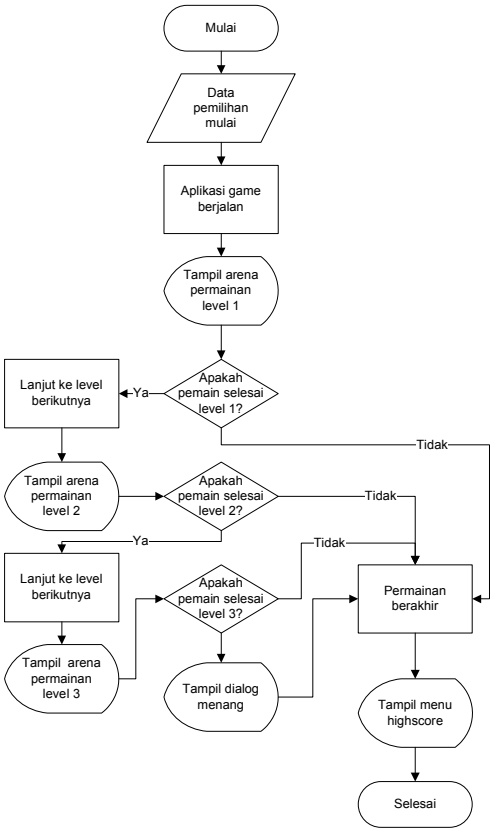
Prosedural ini dijalankan untuk menampilkan menu high*score*. Untuk lebih jelasnya dapat dilihat gambar dibawah ini :



Gambar 3.41 Prosedural Tampil High*score*

3.2.5.8 Percanganan Prosedural Mulai Permainan

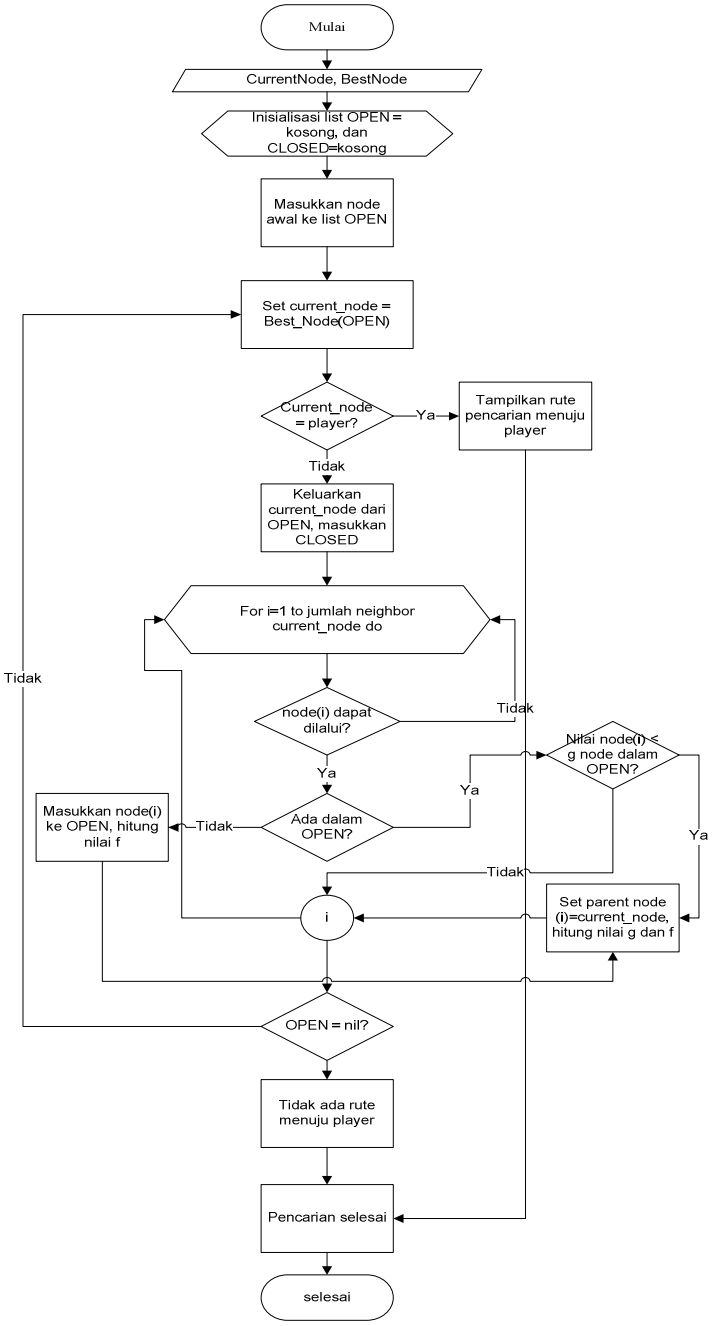
Prosedural ini dijalankan untuk memulai permainan. Untuk lebih jelasnya dapat dilihat gambar dibawah ini :



Gambar 3.42 Prosedural Mulai Permainan

*3.2.5.9 Perancangan Prosedural Gerak Enemy A**

Berikut adalah perancangan prosedural gerak *Enemy* A*. Untuk lebih jelasnya dapat dilihat gambar dibawah ini :



Gambar 3.43 Prosedural Gerak *Enemy* A*