BAB 3

ANALISIS DAN PERANCANGAN

3.1 Analisis Sistem

3.1.1 Analisis Masalah

Dalam sebuah *Game* keberadaan AI (*Artificial Intelligence*) sangatlah penting, karena tanpanya *Game* menjadi sangat membosankan dan tidak menarik untuk dimainkan. AI (*Artificial Intelligence*) biasanya terbangun dari sebuah algoritma tertentu. Dalam pembuatan *Game* ini, penulis menggunakan algoritma A* sebagai pencarian solusi terbaik dan *collision detection* sebagai acuan untuk menentukan object yang bertabrakan.

3.2 Analisis dan Kebutuhan Fungsional

Tahap pertama yang dilakukan dalam melakukan analisis berorientasi objek Pemainkan UML adalah menentukan aktor atau Pemain sistem. Kata aktor dalam konteks UML, menampilkan peran (*roles*) yang Pemain (atau sesuatu di luar sistem yang dikembangkan yang dapat berupa perangkat keras, *end user*, sistem yang lain, dan sebagainya).

3.3 Analisis dan Kebutuhan Non-Fungsional

3.3.1 Analisis dan Kebutuhan Perangkat Lunak

Perangkat lunak digunakan dalam sebuah sistem merupakan perintah-perintah yang diberikan kepada perangkat keras agar bisa saling berinteraksi diantara keduanya. Perangkat lunak yang dibutuhkan untuk membangun aplikasi *Eiji : The Lost Canvas* ini adalah sebagai berikut :

1. Sistem operasi Windows XP SP 2.
2. .NET framework 2.0.

3.3.2 Analisis dan Kebutuhan Perangkat Keras

Komputer terdiri dari perangkat keras dan perangkat lunak yang saling berinteraksi. Perangkat lunak memberikan instruksi-instruksi kepada perangkat keras untuk melakukan suatu tugas tertentu, sehingga dapat menjalankan suatu sistem di dalamnya. Pada aplikasi *Eiji : The Lost Canvas* ini, perangkat keras yang digunakan adalah sebagai berikut :

1. Prosesor 1.0 GHz.
2. RAM 128 MB.
3. VGA 32 MB.
4. Hardisk 150 MB.

3.3.3 Analisis Pemain

Sasaran khalayak dari perancangan permainan komputer ini dapat ditinjau dari segi demografis, psikografis, behavioristis, geografis.

a. Demografis
    1. Orang yang berumur 17 tahun keatas.
    2. Ditargetkan kepada laki-laki.
    3. Berstatus siswa, mahasiswa, ataupun sudah berpenghasilan.
    4. Pasif berbahasa inggris.
b. Psikografis

    Secara psikografis ditujukan kepada orang-orang yang suka bermain *Game*, terutama yang bergenre *Thrid person shooter*.
c. Behavioristis

    Ditinjau dari segi behavioristis ditujukan kepada orang yang sudah sering bermain *Game*.
d. Geografis

    Secara geografis berdomisili di Indonesia.

3.3.4 Pembangunan *Game* Eiji : The Lost Canvas

3.3.4.1 Pengenalan

Game yang akan dibuat adalah *Eiji : The Lost Canvast* yang bergenre *Third person shooter*. *Game* ini menceritakan kisah petualangan seorang karakter *Game* bernama Eiji. Eiji mendapat tugas dari presiden Ank untuk mengambil kembali lukisan yang telah dicuri oleh bangsa robot di dunia yang telah ditinggalkan oleh umat manusia.

Game ini bergrafis 3D yang cara permainannya sediri masih menggunakan sistem single *Player*. AI (*Artificial Intelligence*) yang digunakan untuk musuh dalam *Game* ini adalah algoritma *A** dan *collision detection* sebagai pendeteksi tubrukan apabila mengenai object.

3.3.4.2 *Storyline*

Game ini mengambil latar belakang planet bumi tahun 2080. *Eiji : The Lost Canvast* menceritakan petualangan seorang prajurit setengah robot setengah manusia yang mendapatkan tugas dari presiden untuk mengambil kembali lukisan yang telah dicuri oleh bangsa robot untuk mengancurkan planet emperor.

Dan ceritapun dimulai, Eijipun berangkat meninggalkan planet emperor untuk menjalankan tugas beratnya. Dengan hanya bermodalkan machinegun dan berpakaian armor tahan radiasi, Eiji dengan gagah berani menghancurkan robo-

robot jahat. Setelah semua robot hancur, Eijipun mengambil kembali lukisan yang berada di markas besar robot.

3.3.4.3 *Storyboard*

Storyboard adalah sketsa gambar yang disusun berurutan sesuai dengan naskah, dengan *Storyboard* penulis dapat menyampaikan ide cerita penulis kepada orang lain dengan lebih mudah, karena penulis dapat menggiring khayalan seseorang mengikuti gambar-gambar yang tersaji, sehingga menghasilkan persepsi yang sama pada ide cerita penulis. Berikut merupakan *Storyboard* dari Aplikasi *Eiji : The Lost Canvast*.

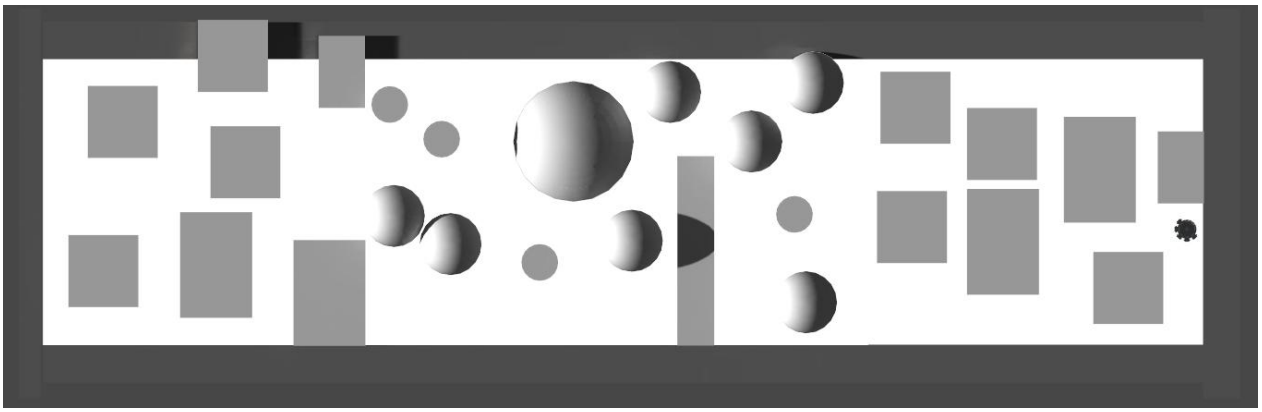


Gambar 3.1 *Storyboard Stage* 1 pada *Game***.**

Stage 1 adalah awal mula perjalanan *Player* dalam menjalankan tugasnya menuju tempat persembunyian *canvas of live* yang dicuri oleh bangsa robot. Pada *stage* 1, permainan dimulai dengan menampilkan *Player* dengan membawa senjata. Pada *stage* ini, musuh banyak yang berkeliaran sehingga *Player* harus memusnahkan semua musuh. Musuh pada *stage* ini memiliki kecepetan yang

cukup lambat dalam mengejar *Player*, jumlah health yang sedikit sehingga mudah dihancurkan, dan demage yang sedikit ketika menembak *Player*.

Fasilitas yang tersedia pada *stage* ini adalah adanya 3 checkpoint yang berguna untuk menghidupkan *Player* kembali setelah mati kehabisan darah. Dan setelah *Player* berhasil memusnahkan musuh yang menghalanginya untuk berpindah *stage*, *Player* akan menemukan sebuah computer box yang berada di akhir *stage* untuk berpindah ke *stage* selanjutnya.



Gambar 3.2 *Storyboard Stage* 2 pada *Game*.

Pada *stage* 2 permainan dimulai di tempat yang cukup terang sehingga musuh yang berada di dekat *Player* bisa terlihat dengan jelas, tidak seperti *stage* 1 yang tempatnya cukup gelap. *Stage* 2 memliki tempat yang cukup kecil untuk dilewati, namun pada *stage* ini kemampuan musuh meningkat dibandingkan *stage* sebelumnya. Kecepatan dalam mengerjar *Player* bertambah, jumlah health bertambah sehingga cukup sulit untuk di musnahkan, dan demage peluru yang ditembakan oleh musuh ke *Player* cukup besar.

Di *stage* ini, hanya tersedia 1 checkpoint yang dimana *Player* harus pintar menghindari peluru-peluru yang datang menerjang. Computer box pada *stage* ini ditempatkan cukup tersembunyi sehingga *Player* harus mencarinya dengan melewati beberapa mecha yang cukup tangguh untuk dihancurkan.

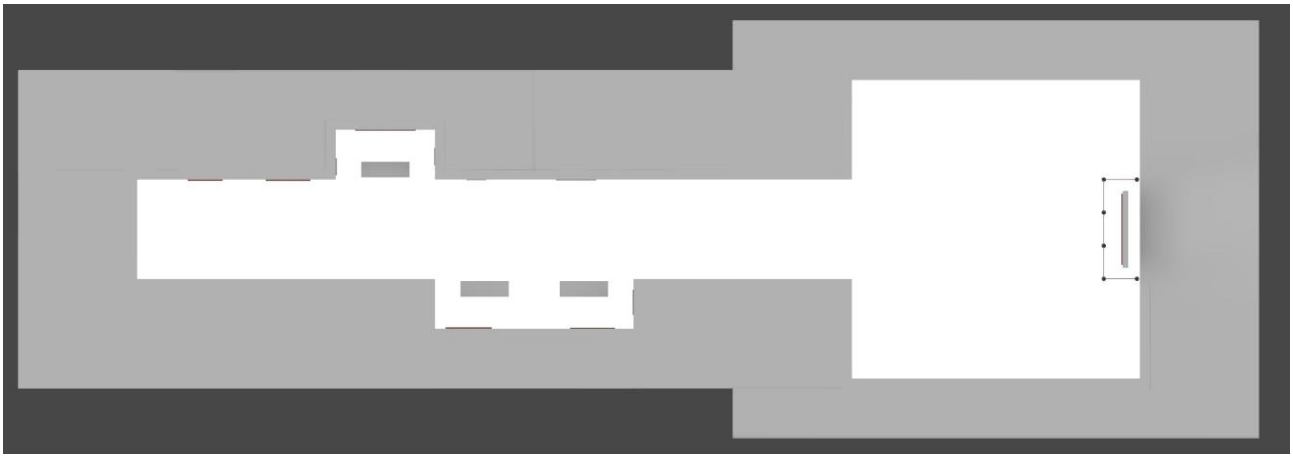


Gambar 3.3 *Storyboard Stage* 3 pada *Game.*

Pada *stage* ini *Player* akan bertemu kembali dengan para musuh-musuh lamanya, namun musuh kali ini mengalami peningkatan kemampuan yang drastis dari *stage-stage* sebelumnya sehinga *Player* harus bisa menghindari serangan musuh dan mencari jalan menuju computer box untuk berpindah ke *Final stage*. Akan tetapi jalan dan tempat di *stage* 3 ini berbeda dengan *stage* sebelumnya. Untuk membuka pintu saja, *Player* harus meng-hack komputer yang berada di

stage ini. Setelah berhasil meng-hack komputernya, pintu otomatis akan terbuka dan *Player*-pun bisa melanjutkan permainan kembali

Stage 3 adalah *stage* terakhir dari permainan. Dimana pada *stage* ini *Player* diberikan kesulitan yang berbeda dengan *stage-stage* sebelumnya dan pada *stage* ini *Player* harus mencari jalan menuju computer box yang cukup rumit. Di *stage* ini memiliki tempat yang luas dan musuh yang bersembunyi di tempat-tempat yang menyulitkan *Player* untuk menuju computer box.



Gambar 3.4 *Storyboard Final Stage* pada *Game.*

Setelah melewati rintangan-rintangan pada *stage* sebelumnya, *Player* akhirnya berada di *Final stage*. *Final stage* adalah tempat disembunyikannya canvas of life.

3.3.4.4 *Gameplay*

Dalam *Game Eiji : The Lost C*avast, pemain diharuskan mengancurkan semua robot-robot yang ada di dalam *Game* untuk berpindah *level*. Dimana setiap bagian *level* menyediakan petualangan dengan menampilkan beberapa karakter

level tersebut maupun karakter musuh yang harus dilawan. Di setiap *stage* tersedia fasilitas *checkpoint* yang berguna ketika *Player* mati, *Player* bisa kembali hidup di tempat dia melewati *checkpoint*.

Setiap lawan yang dihadapi memiliki cara yang berbeda dalam memusnahkan *Player*, bejalan mengejar *Player*, daya tahan serangan dari *Player*, kecepatan meregenerasi *health*. Di setiap *stage*, musuh memiliki kelebihan yang berbeda dengan *stage* sebelumnya.

Tabel 3.1 Pejelasan kemampuan *enemy* dan *Player*.

Model Karakter	Kemampuan
Spider 	Pada musuh ini, spider mempunyai kemampuan untuk menghancurkan diri ketika mendekati *Player*. Dan apabila *Player* menjauhi spider, maka spider akan kembali ke pos pertama dia diam. Spider juga mempunyai kemampuan untuk meregenerasi health.
Mecha 	Pada musuh ini, mecha bertugas untuk membunuh *Player* dengan misil yang keluar dari senjatany yang berada di pinggir kepalanya. Mecha mempunyai kemampuan patroli, mengejar *Player* hingga mati dan menemembakan misil bertubi-tubi secara diagonal. Namun ketika *Player*

	bersembunyi dibalik dinding, mecha akan berhenti mengejar dan akan kembali ke waypoint dimana dia berpatroli. Mecha juga mempunyai kemampuan untuk meregenerasi health.
 Buzzer	Pada musuh ini, buzzer bertugas membunuh *Player* dengan sengatan listrik yang keluar dari mulutnya. Buzzer mempunyai kemampuan untuk membunuh *Player* dengan arah random secara diagonal. Buzzer juga mempunyai kemampuan untuk meregenerasi health.
Eiji 	Ini adalah karakter Eiji yang *user* mainkan untuk menjalan tugasnya. Eiji mempunyai kemampuan menembak musuh bertubi-tubi hingga mush hancur. Eiji memiliki health yang berada dipunggungnya, sehingga ketika health akan habis maka indikator healthnya akan berkedap-kedip. Ketika Eiji mati, dia akan kembali ke checkpoint yang telah dilwatinya. Eiji juga mempunyai kemampuan untuk meregenerasi *health*.

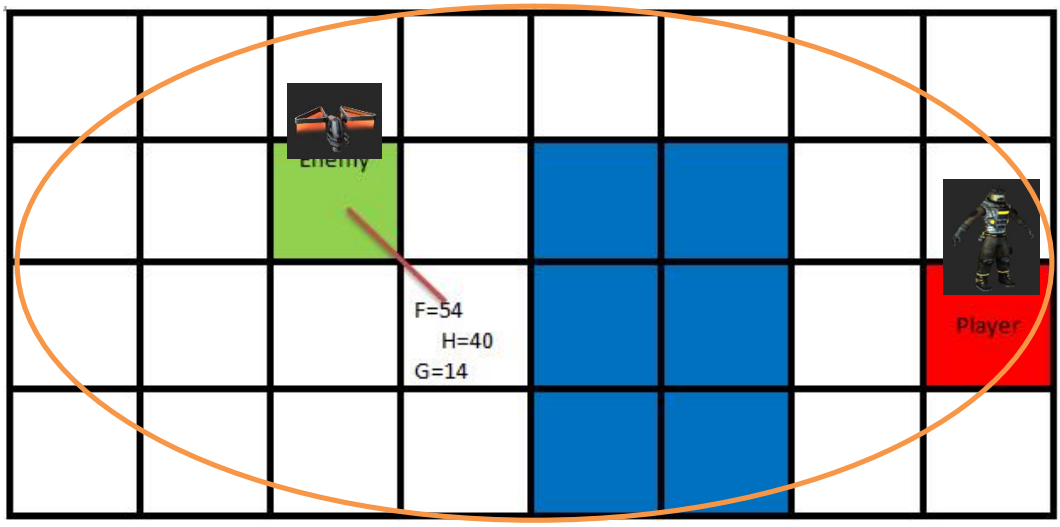
3.3.5 Representasi AI menggunakan Algoritma A*

Terdapat beberapa hal yang perlu didefinisikan terlebih dahulu dalam kasus *Game* ini dengan penerapan algoritma A* *(A Star)*. Adapun istilah-istilah yang akan dibahas yaitu *path, open list, closed list,* nilai f, g dan n.

Algoritma A* menggunakan dua senarai yaitu *OPEN* dan *CLOSED*. *OPEN* adalah senarai *(list)* yang digunakan untuk menyimpan simpul-simpul yang pernah dibangkitkan dan nilai heuristiknya telah dihitung tetapi belum terpilih sebagai simpul terbaik *(best node)* dengan kata lain, *OPEN* berisi simpul-simpul masih memiliki peluang untuk terpilih sebagai simpul terbaik, sedangkan *CLOSED* adalah senarai untuk menyimpan simpul-simpul yang sudah pernah dibangkitkan dan sudah pernah terpilih sebagai simpul terbaik. Artinya, *CLOSED* berisi simpul-simpul yang tidak mungkin terpilih sebagai simpul terbaik (peluang untuk terpilih sudah tertutup).

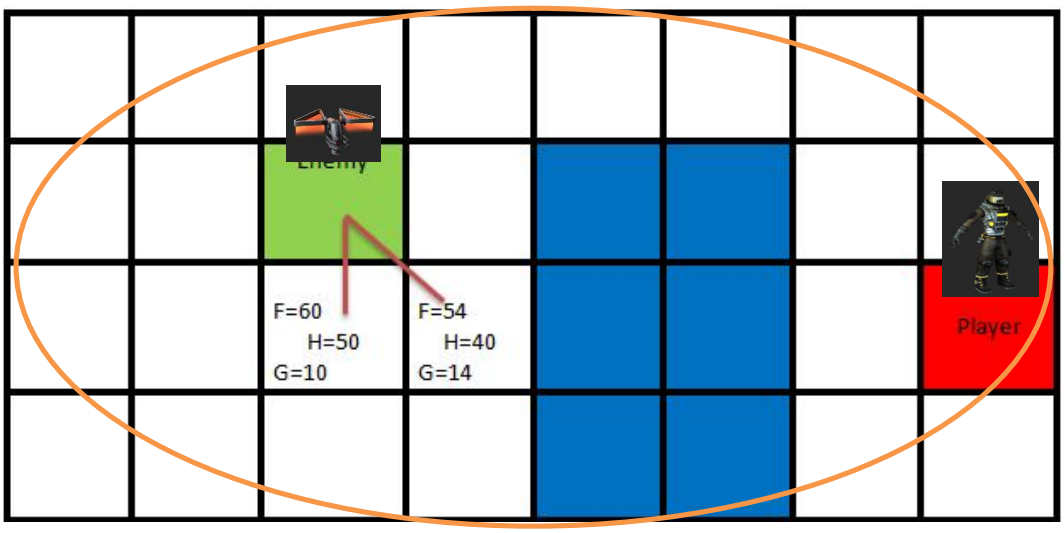
Berikut ini adalah penjelasan AI yang digunakan musuh dalam pengejaran *play*er**,** dalam pengujian ini asumsi bahawa *Play*er terdeteksi oleh musuh dan *player* dalam keadaan diam ditempat. Dalam pengejaran ini, musuh menggunakan algoritma A* untuk pencarian rute terdekat agar *play*er bisa ditembak..

Apabila *play*er telah keluar dari batas pendeteksian, maka musuh akan kembali ke tempat semula dia berdiri melalui jejak pencarian yang telah musuh buat dan apabila *play*er terhalang oleh objek yang besar maka pencarian akan diakhiri dan musuhpun kembali ke tempat dia semula berdiri.



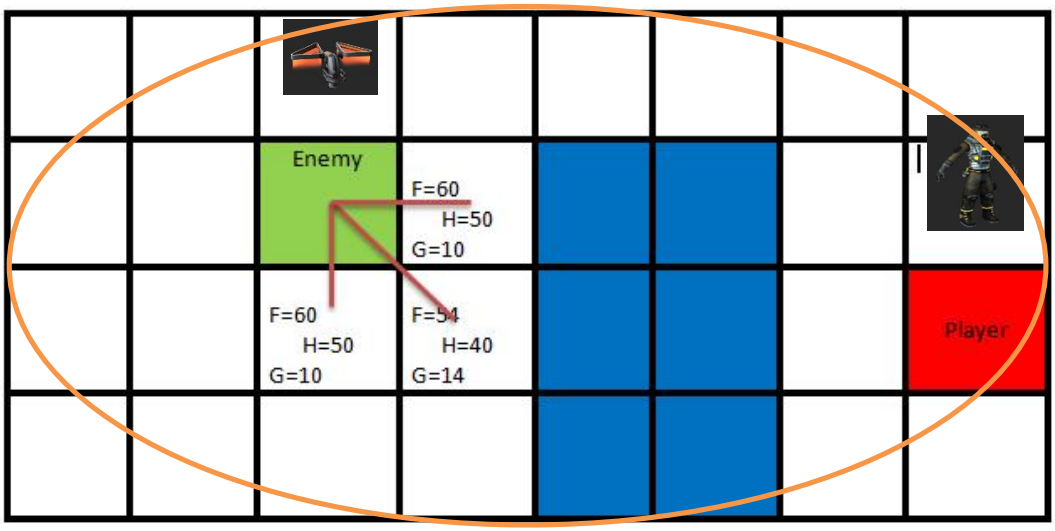
Gambar 3.5 Pencarian menggunakan algoritma A* langkah 1.

Dari *enemy* mencari secara diagonal dengan nilai G=14, dan H=40 maka dengan rumus F=G+H , dihasilkanlah nilai F=54.



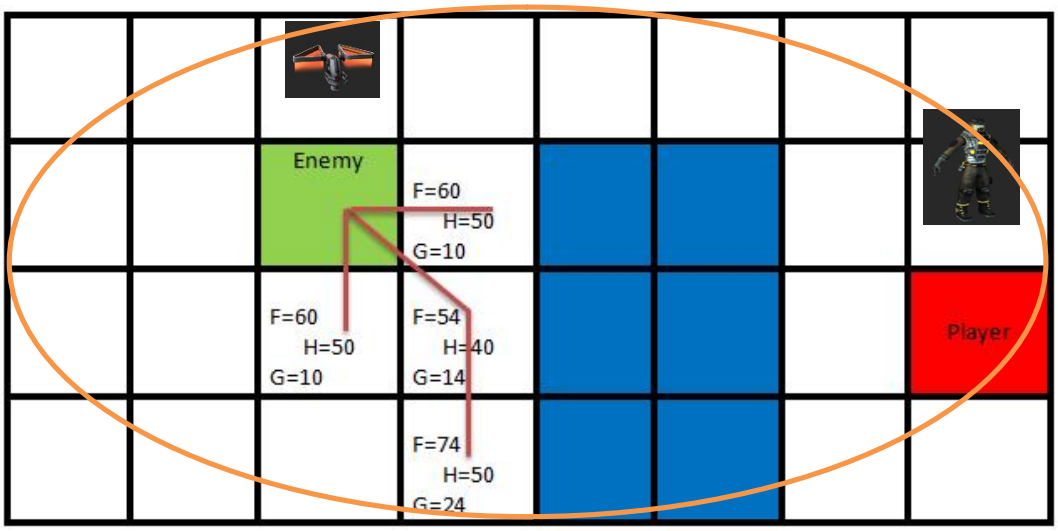
Gambar 3.6 Pencarian menggunakan algoritma A* langkah 2.

Dari *enemy* mencari secara vertikal dengan nilai G=10, dan H=50 maka dengan rumus F=G+H , dihasilkanlah nilai F=60.



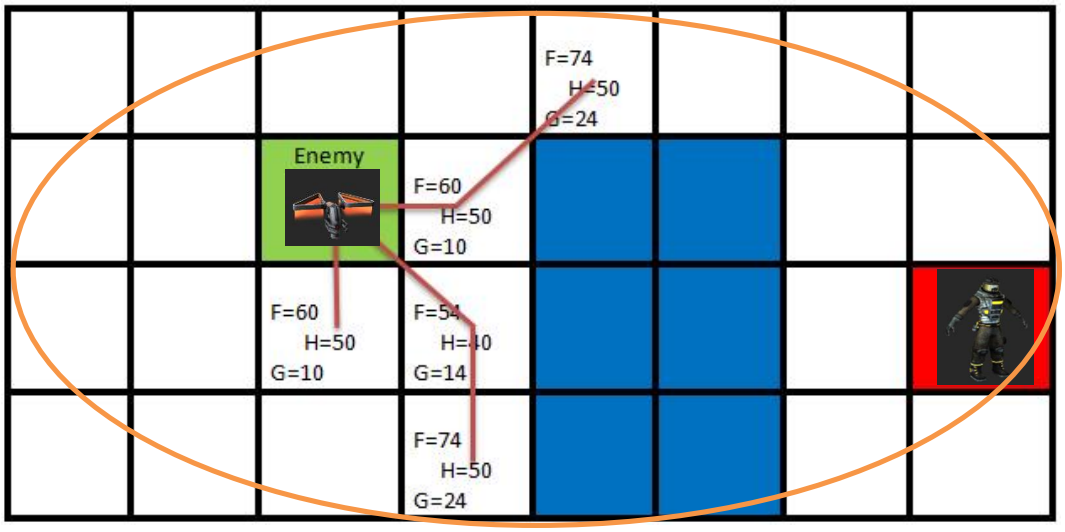
Gambar 3.7 Pencarian menggunakan algoritma A* langkah 3**.**

Kemudian dari *enemy* mencari secara horizontal dengan nilai G=10, dan H=50 maka dengan rumus F=G+H , dihasilkanlah nilai F=60.



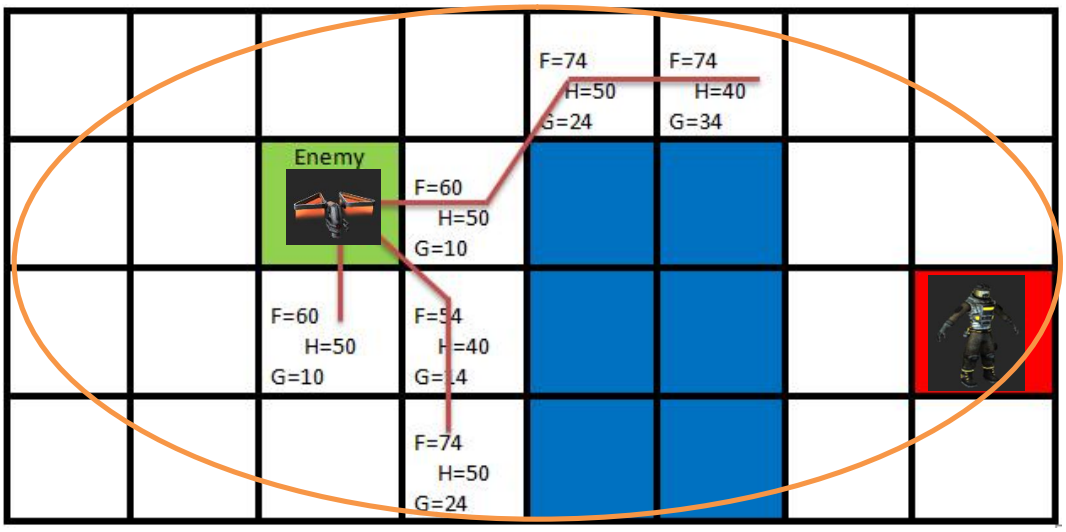
Gambar 3.8 Pencarian menggunakan algoritma A* langkah 4**.**

Kemudian dari node yang bernilai F=54, mencari secara vertikal dengan nilai G=24 yang didapatkan dari langakah 1 ditambah dengan niai G vertikal yaitu 10, dan H=50 maka dengan rumus F=G+H , dihasilkanlah nilai F=74.



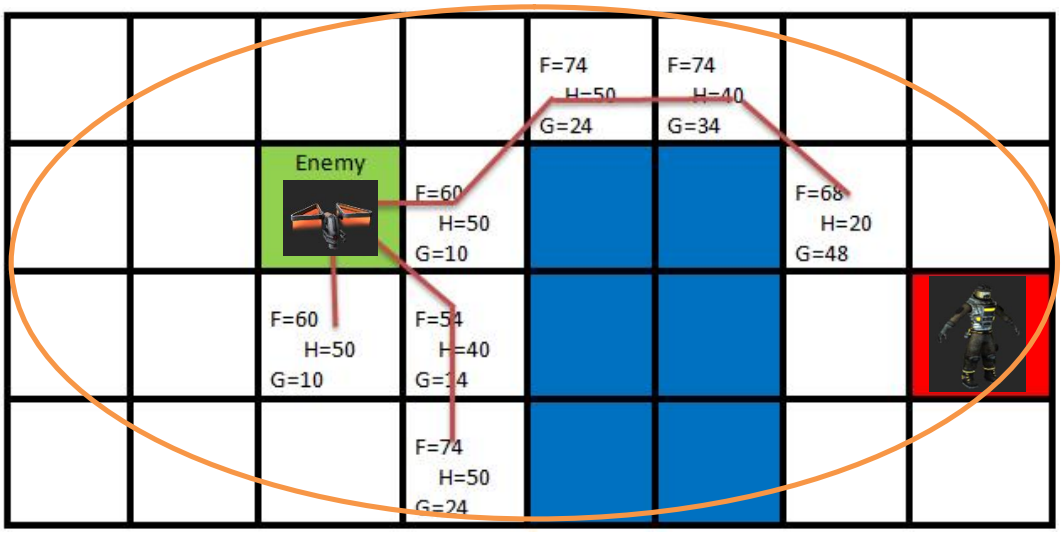
Gambar 3.9 Pencarian menggunakan algoritma A* langkah 5**.**

Kemudian dari node yang bernilai F=60, mencari secara diagonal dengan nilai G=24 yang didapatkan dari langakah 1 ditambah dengan niai G vertikal yaitu 10, dan H=50 maka dengan rumus F=G+H , dihasilkanlah nilai F=74.



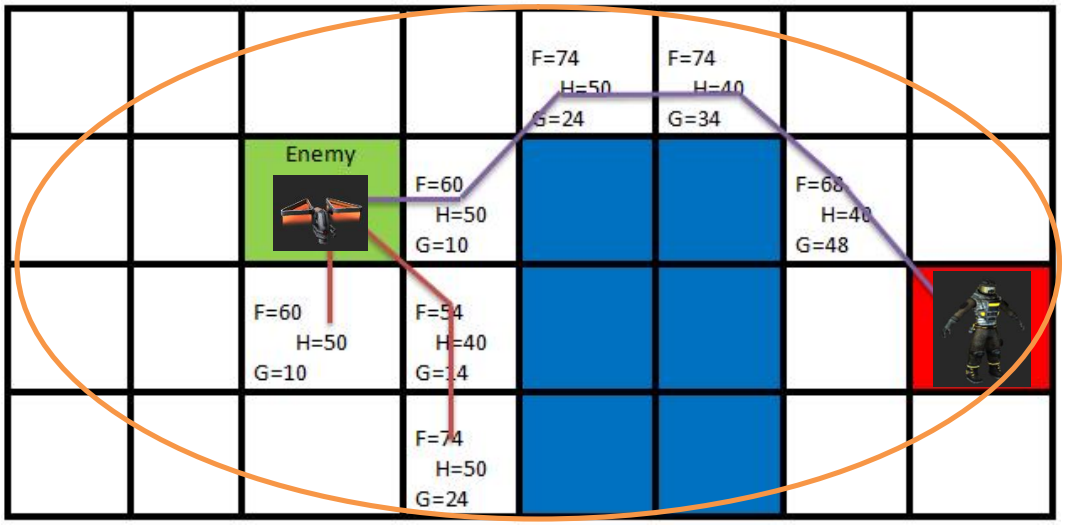
Gambar 3.10 Pencarian menggunakan algoritma A* langkah 6.

Kemudian dari node yang bernilai F=74, mencari secara horizontal dengan nilai G=34 yang didapatkan dari langakah 5 ditambah dengan niai G vertikal yaitu 10, dan H=40 maka dengan rumus F=G+H , dihasilkanlah nilai F=74.



Gambar 3.11 Pencarian menggunakan algoritma A* langkah 7.

Kemudian dari node yang bernilai F=74, mencari secara diagonal dengan nilai G=48 yang didapatkan dari langakah 6 ditambah dengan niai G vertikal yaitu 10, dan H=50 maka dengan rumus F=G+H , dihasilkanlah nilai F=68.



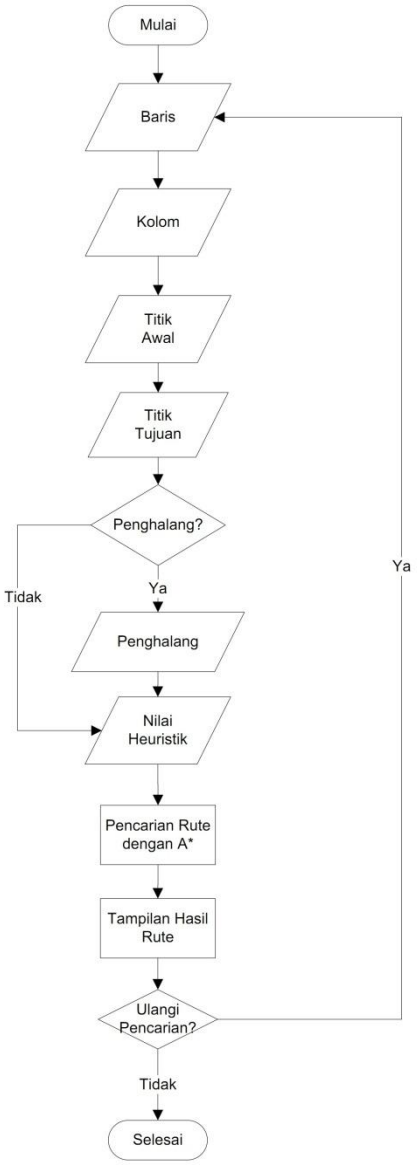
Gambar 3.12 Pencarian menggunakan algoritma A* langkah 8**.**

Maka dihasilkan jalur optimum untuk pencarian *enemy* menuju *Player* dengan menggunakan algoritma A*.

Bagan alir sistem *(flowchart system)* merupakan bagan-bagan yang mempunyai arus yang menggambarkan langkah-langkah penyelesaian suatu masalah dan merupakan cara penyajian dari suatu algoritma, sehingga dapat menjelaskan urutan prosedur-prosedur yang ada dalam system.

Pada gambar 3.13. dijelaskan bahwa cara kerja sistem yang akan dibangun yaitu dimulai dari memasukkan data berupa titik awal, titik tujuan dan penghalang maupun tanpa penghalang, kemudian masuk kedalam pencarian *rute* dengan

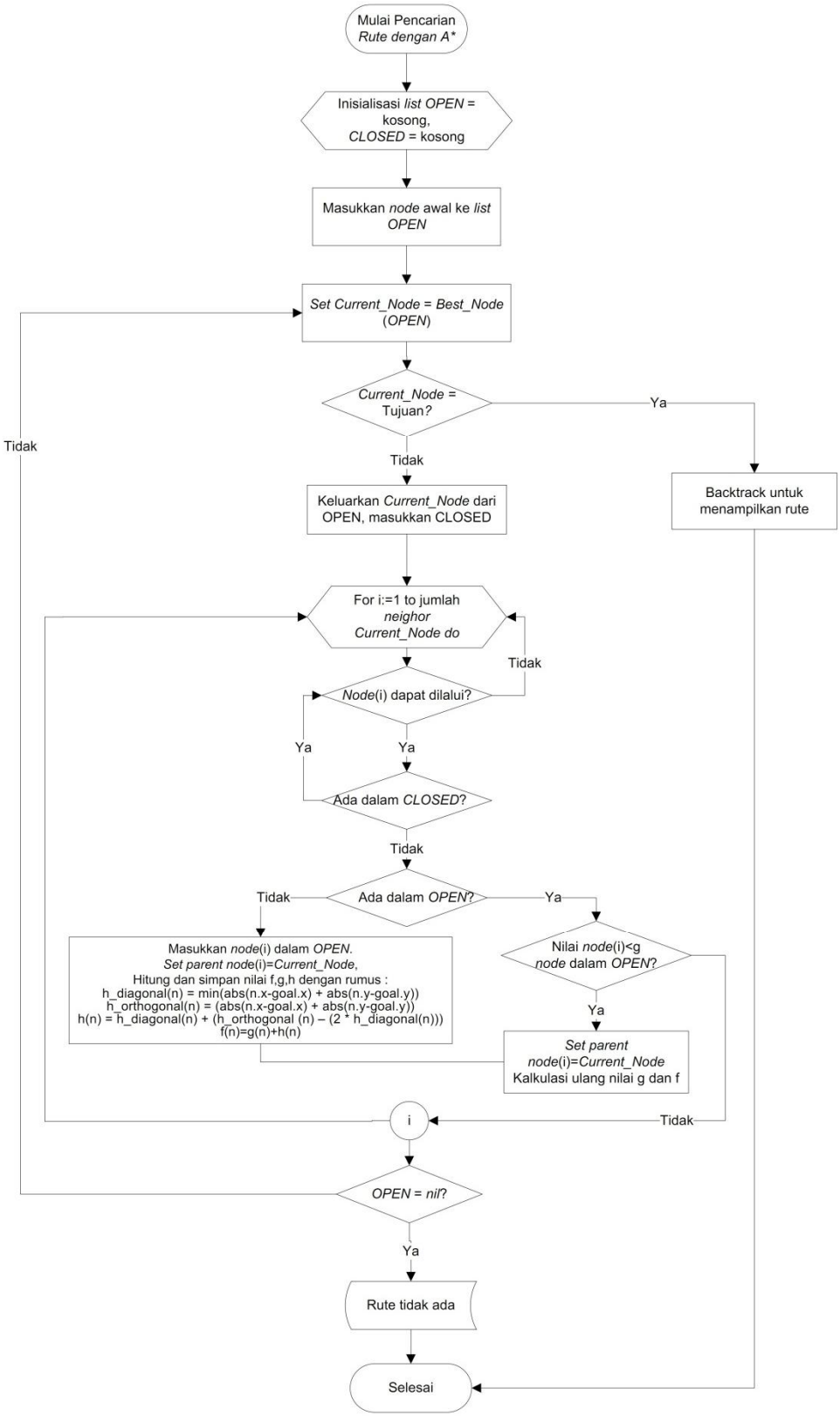
algoritma A* *search.* Setelah pencarian *rute* yang dipilih A* maka akan muncul jalan yang ditemukan oleh algoritma A*, jika sekaligus dicentang "lihat cabang pencarian" maka akan muncul pelacakan cabangnya. Setelah selesai musuh menghapus *rute* dan dapat memasukkan masukan baru.



Gambar 3.13 Flowchart pencarian rute

Diagram alir algoritma A* dapat dilihat pada gambar 3.14. Pada gambar 3.14 menjelaskan cara kerja algoritma A* bekerja dalam mencari *rute* paling optimum pada peta. Algoritma ini bekerja dari titik awal ke titik tujuan dengan mencari jalan dengan dengan akumulasi bobot terendah (dilambangkan dengan notasi g) dijumlah dengan perkiraan jalan terendah bobotnya untuk mencapai tujuan (dilambangkan dengan notasi h). *Rute* yang memiliki *node-node* dengan nilai f(n) = g(n) + h(n) yang paling kecil akan dikembangkan lebih dulu, karena itu proses pemberian bobot pada *node-node* sangatlah penting dalam menentukan hasil pencarian.

Proses mencari *node* dengan nilai terkecil ini dilakukan berulang-ulang sampai menemukan *node* tujuan. Apabila *node* yang menjadi titik tujuan telah ditemukan, program akan melakukan *backtrack* ke *parent* dari tiap *node* untuk mendapatkan rangkaian *node* yang membentuk *rute* yang akan paling optimum yang diinginkan. Setelah musuh memilih tujuan, maka pencarian pun dipilih yaitu dengan menggunakan algoritma A*.



Gambar 3.14 *Flow chart* algoritma A*.

3.3.6 Representasi *Collision Detection*

Segitiga adalah primitif yang paling banyak digunakan dalam pembuatan objek 3 dimensi, hal ini disebabkan karena sifat segitiga yang selalu konveks sehingga dapat lebih cepat diproses oleh kartu grafis. Karena sebagian besar objek dibentuk oleh segitiga maka untuk deteksi tubrukan dilakukan dengan memeriksa apakah ada segitiga segitiga dalam dua buah objek yang saling memotong, apabila ada maka pastilah terjadi tubrukan. Metode interval overlap yang dikembangkan oleh Tomas Moller merupakan salah satu metode yang digunakan untuk mendeteksi perpotongan segitiga. Misal terdapat dua segitiga T_1 dan T_2 , titik titik pada segitiga T_1 dan T_2 adalah V_0^1, V_1^1, V_2^1 dan V_0^2, V_1^2, V_2^2 dan bidang yang ditempati adalah π_1 dan π_2. Langkah pertama adalah menentukan persamaan bidang π_2 : $N_2.X + d_2 = 0$, dengan X adalah titik sembarang pada bidang. N_2 adalah persamaan garis normal pada bidang π_2. Untuk menghitung N_2 dan d_2 digunakan persamaan:

$$N_2 = (V_1^2 - V_0^2) \times (V_2^2 - V_0^2)$$

$$d_2 = N_2 \times V_0^2$$

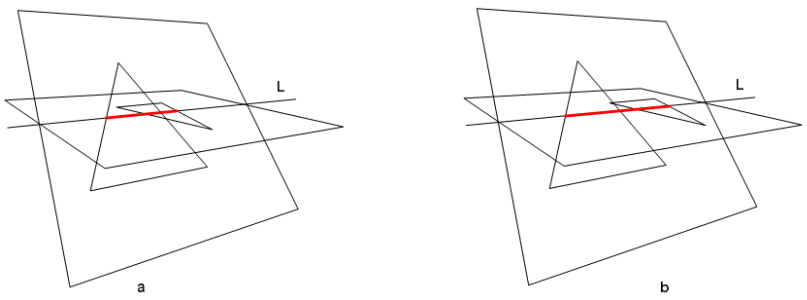
Langkah selanjutnya adalah menghitung jarak dari titik titik pada segitiga T_1 ke bidang π_2. Penghitungan jarak dilakukan dengan memasukkan titik titik T_1 pada persamaan bidang, sehingga didapat

$$dV_i^1, i = 0, 1, 2.$$

$$dV_i^1 = N_2. V_i^1 + d_2, i = 0, 1, 2.$$

Jika semua $dV_i^1 \neq 0$,i = 0,1,2 dan semua memiliki tanda yang sama (semuanya positif atau semuanya negatif) maka T_1 terletak pada satu sisi bidang yang sama maka tidak akan terjadi perpotongan. Hal yang sama dilakukan pula kepada segitiga T_2 dan bidang Jika semua $dV_i^1 = 0$, i = 0, 1, 2 maka segitiga terletak dalam bidang yang sama sehingga cukup dilakukan pemeriksaan perpotongan segitiga 2 dimensi.

Jika semua persyaratan diatas tidak dipenuhi maka pada bidang π_1 dan π_2 terdapat suatu garis perpotongan yaitu: L = 0 + tD, dimana D = N_1 X N_2 adalah arah dari garis dan O adalah suatu titik pada garis itu. Karena perhitungan yang telah dilakukan di atas tadi, kedua segitiga dipastikan memotong garis L. Perpotongan tersebut membentuk interval pada L, jika kedua interval tersebut berimpit maka dapat dipastikan bahwa kedua segitiga berpotongan. Kedua kemungkinan tersebut digambarkan pada gambar dibawah ini:



Gambar 3.15 a. Kedua segitiga berpotongan

b. Kedua segitiga tidak berpotongan.

Gambar diatas menunjukkan kedua segitiga dan bidang dimana segitiga tersebut berada, perpotongan antara kedua bidang ditunjukkan dengan garis L. Gambar a menunjukkan saat kedua segitiga berpotongan. Perpotongan antara kedua segitiga tersebut digambarkan dengan garis merah. Langkah selanjutnya adalah menghitung garis perpotongan antara T_1 dan L. Misalkan V_0^1, dan V_2^1 terdapat pada sisi yang sama terhadap bidang π_2 dan V_1^1 terletak pada sisi yang lain. Untuk menemukan garis yang merupakan perpotongan dari sisi $V_0^1\ V_1^1$ dan $V_1^1\ V_2^1$ dan L, pertama tama semua titik diproyeksikan ke L:

$$pv_i^1 = D.(V_i^1 - 0)$$

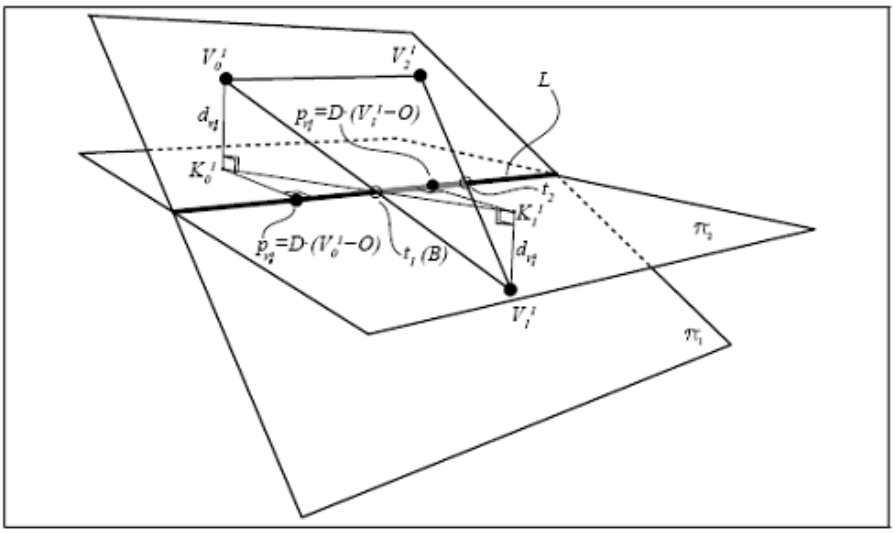
Langkah selanjutnya adalah menghitung nilai parameter garis, t_1, untuk

$$B = V_0^1\ V_1^1 \cap L = O + t_1 D$$

K_i^1 menunjukkan proyeksi dari V_i^1 ke π_2, terlihat bahwa $\Delta v_0^1 B K_0^1$ dan $\Delta v_1^1 B K_1^1$ adalah sama, sehingga :

$$t_1 = pv_0^1 + (pv_1^1 - pv_0^1)\frac{dv_0^1}{dv_0^1 - dv_1^1}$$

Perhitungan juga dilakukan untuk segitiga T2, apa bila kedua garis saling berpotongan maka kedua segitiga juga berpotongan. Situasi diatas tampak pada gambar dibawah ini:



Gambar 3.16 Perhitungan garis perpotongan antara T1 dan L.

Jika Kedua segitiga terletak pada bidang yang sama, maka kedua segitiga tersebut diproyeksikan kebidang yang terletak pada salah satu sumbu. Pemilihan sumbu diatur sehingga luas segitiga menjadi maksimal. Selanjutnya tes perpotongan segitiga 2 dimensi dilakukan. Pertama tes semua sisi dari T1 untuk perpotongan dengan T2. Jika perpotongan ditemukan maka kedua segitiga berpotongan. Jika tidak maka harus diperiksa apakah T1 berada dalam T2 atau sebaliknya.

Algoritma dari metode ini adalah:

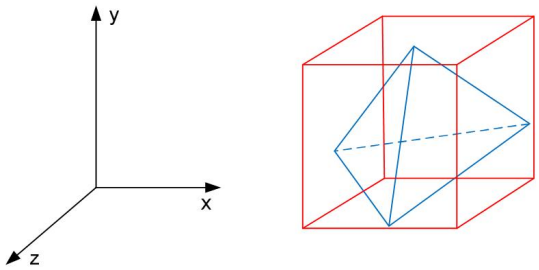
1. Hitung persamaan bidang segitiga 2
2. Hitung jarak semua titik segitiga 1 terhadap bidang segitiga 2
3. Jika jarak titik semua bertanda positif atau semua bertanda negatif, maka segitiga 1 terletak pada satu sisi bidang segitiga 2, maka segitiga tidak berpotongan.

4. Hitung persamaan bidang segitiga 1
5. Hitung jarak semua titik segitiga 2 terhadap bidang segitiga 1
6. Jika jarak titik semua bertanda positif atau semua bertanda negatif, maka segitiga 2 terletak pada satu sisi bidang segitiga 1, maka segitigatidak berpotongan.
7. Hitung garis perpotongan antara bidang 1 dan bidang 2
8. Periksa apakah segitiga 1 dan 2 terletak pada bidang yang sama, jika ya, maka lakukan tes perpotongan segitiga 2 dimensi, jika tidak lanjut ke langkah selanjutnya.
9. Proyeksikan garis perpotongan ke sumbu terbesar
10. Hitung interval untuk masing masing segitiga
11. Jika interval berimpit, maka segitiga berpotongan, jika tidak, segitiga tidak berpotongan

Dalam aplikasi 3D biasanya jumlah primitif yang digunakan sangat besar. Sehingga untuk melakukan perhitungan deteksi perpotongan antara seluruh segitiga yang ada dalam satu frame tidaklah mungkin karena akan memerlukan kecepatan perhitungan yang sangat besar. Untuk itu diperlukan suatu cara untuk mengurangi jumlah perhitungan dalam satu frame, dengan cara membuat suatu pembatas antara objek objek yang akan bertabrakan, sehingga deteksi perpotongan hanya dilakukan antara objek objek yang sudah berdekatan.

Axis Aligned Bounding Box (AABB) merupakan salah satu bentuk dari Bounding Volume AABB berbentuk kotak yang semua sisinya searah dengan sumbu, dan setiap sisinya tegak lurus dengan salah satu sumbu koordinat. Karena

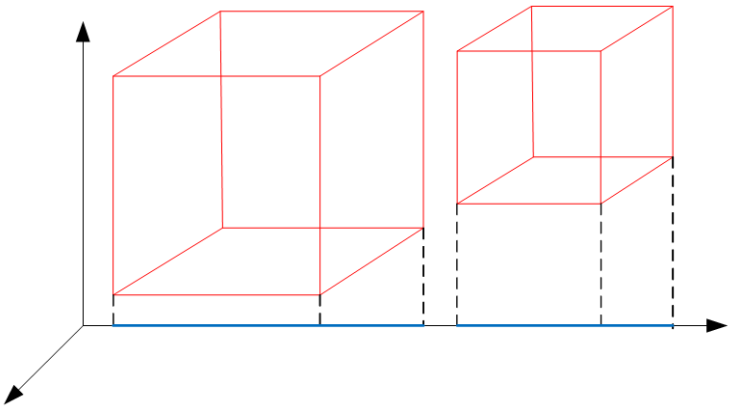
AABB selalu searah dengan sumbu, maka ketika objek ditransformasikan, misalnya berotasi maka AABB tidak bias dirotasikan juga, AABB harus dihitung ulang setiap frame. Perhitungan AABB relatif sederhana oleh karena itu hanya sedikit berdampak pada kecepatan proses keseluruhan.



Gambar 3.17 Sebuah objek 3D (biru) dan AABB nya (merah).

Untuk mengetahui apakah 2 AABB saling bertubrukan maka digunakan langkah perhitungan seperti di bawah ini:

1. Proyeksikan semua sisi ke sumbu x.

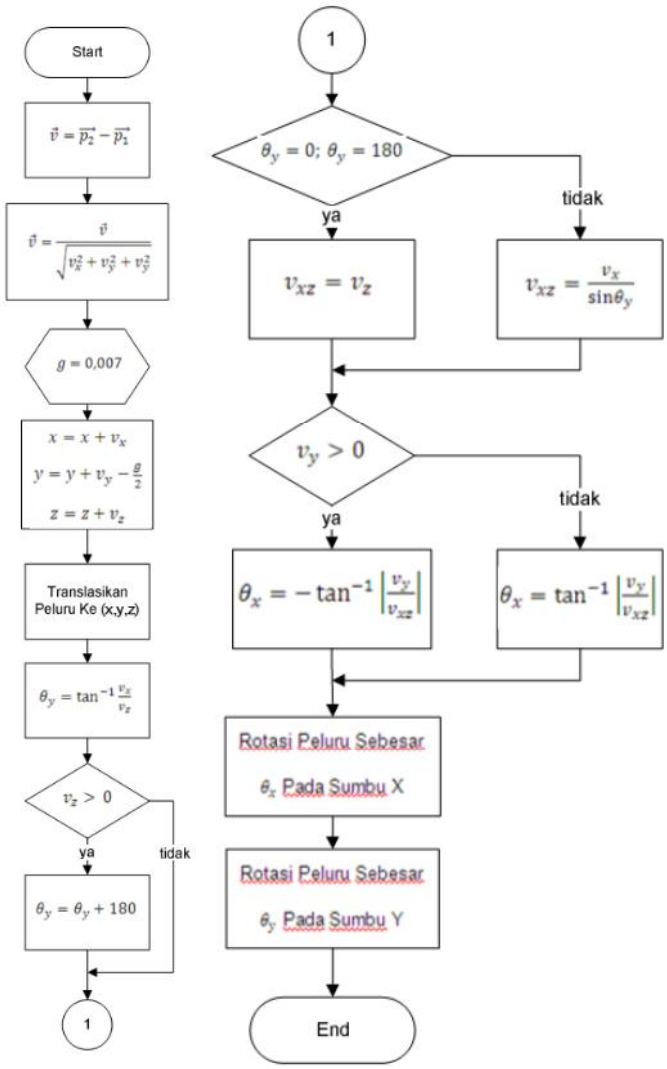


Gambar 3.18 Proyeksi AABB ke sumbu x.

2. Apabila kedua garis tidak bertindihan maka AABB tidak berpotongan. Apabila ya, lakukan langkah diatas pada sumbu selanjutnya

3. Apabila pada ketiga sumbu tidak ada garis yang bertindihan maka AABB tidak berpotongan.

Sebelum digambar pada layar, terlebih dahulu peluru ditranslasi dan dirotasi sesuai dengan posisinya. Diagram alir pergerakan peluru tampak pada gambar dibawah ini.

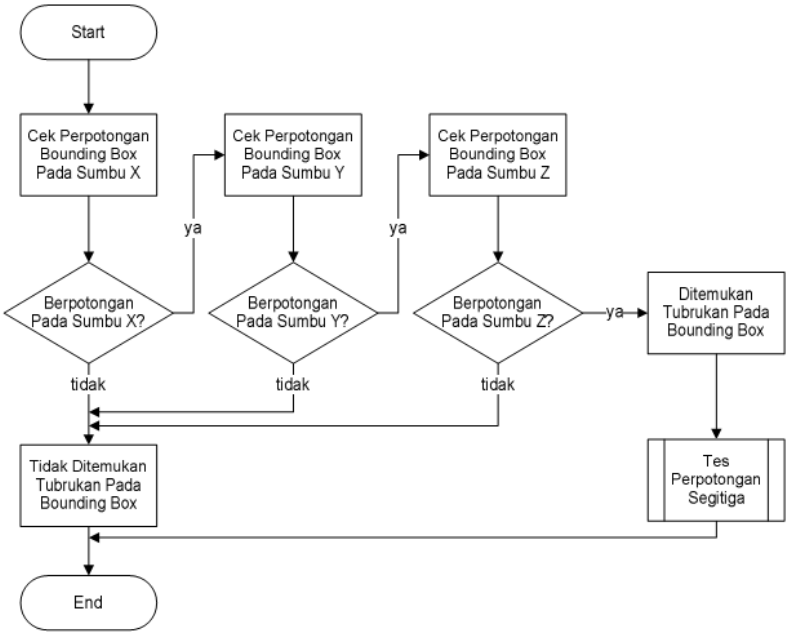


Gambar 3.19 Flow chart proses pergerakan peluru.

Diagram ini menggambarkan proses mencari posisi peluru, dan sudut rotasi peluru, sehingga peluru dapat digambarkan secara tepat.

3.3.6.1 Diagram Alir Deteksi Tubrukan

Gambar dibawah ini menunjukkan diagram alir dari proses deteksi tubrukan.

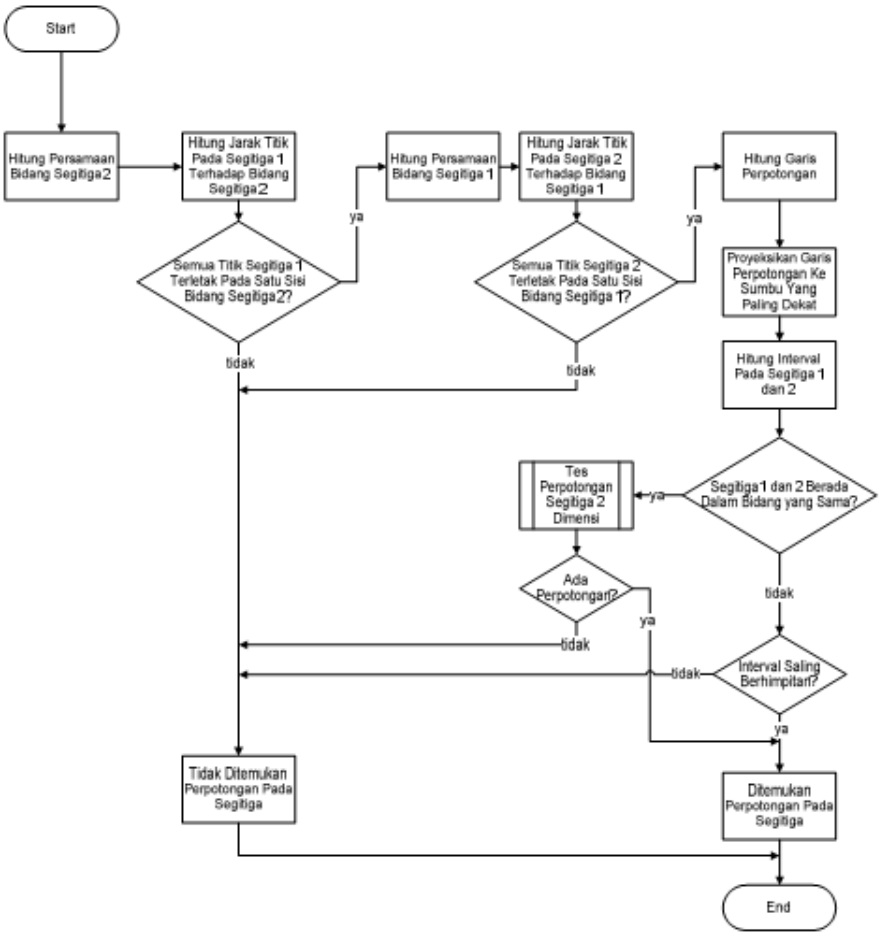


Gambar 3.20 Flow chart deteksi tubrukan.

Deteksi Tubrukan digunakan untuk menentukan apakah peluru mengenai daratan, kendaraan, atau tidak mengenai apapun. Karena proses perhitungan tes perpotongan segitiga membutuhkan waktu yang lama, maka setiap objek memiliki sebuah Bounding Volume. Langkah pertama dari deteksi tubrukanadalah menghitung tabrakan antara Bounding Volume

3.3.6.2 Diagram Alir Tes Perpotongan Segitiga

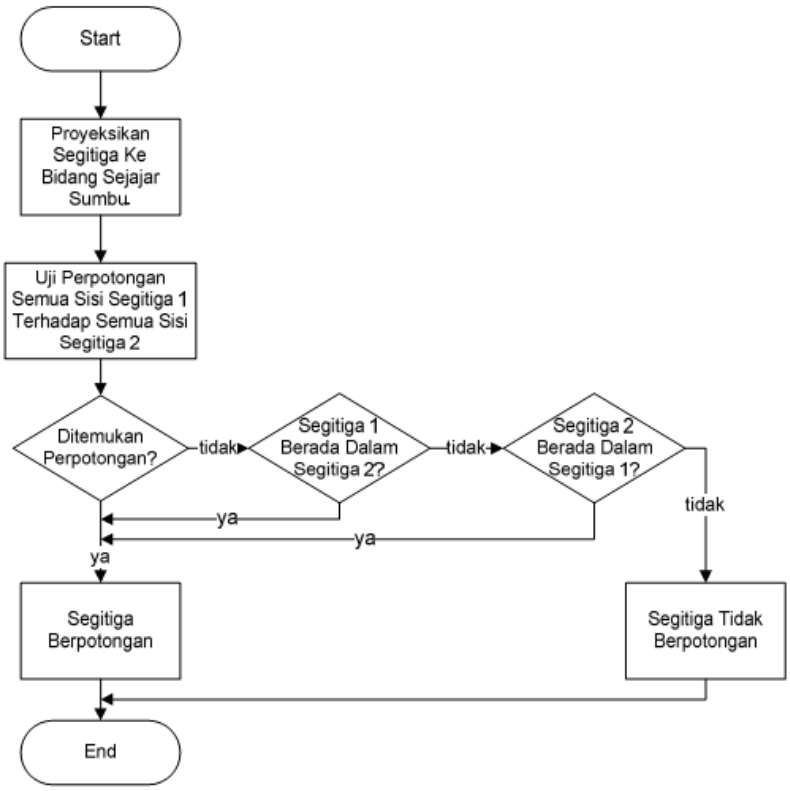
Tes perpotongan segitiga adalah proses terakhir untuk menentukan apakah dua buah objek benar benar bertubrukan. Metode tes perpotongan segitiga yang digunakan dalam aplikasi ini mengacu pada metode interval overlap oleh Moller. Gambar dibawah ini mennunjukkan diagram alir tes perpotongan segitiga.



Gambar 3.21 Flow chart tes perpotongan segitiga.

3.3.6.3 Diagram Alir Tes Perpotongan Segitiga 2 Dimensi

Dalam aplikasi 3 dimensi, tetap diperlukan tes perpotongan segitiga 2 dimensi, tes ini digunakan untuk menangani keadaan dimana kedua segitiga yang di tes berada dalam bidang yang sama atau nyaris berimpit. Apabila kedua segitiga terletak pada bidang yang sama, maka tidak dapat dicari garis perpotongan antar bidang seperti pada langkah tes perpotongan segitiga 3 dimensi, oleh karena itu keadaan seperti ini, walaupun jarang terjadi perlu ditangani secara khusus agar deteksi tubrukan tetap akurat. Dalam tes ini digunakan metode "Faster Line Segment Intersection" oleh Franklin Antonio untuk menemukan perpotongan antara dua buah sisi segitiga



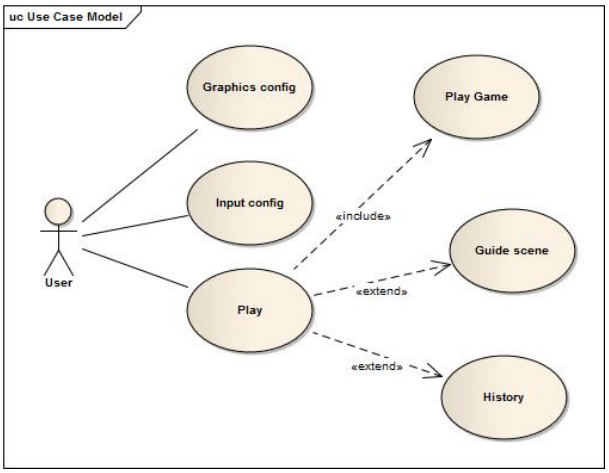
Gambar 3.22 Flow chart tes perpotongan segitiga 2 dimensi.

3.4 Perancangan

Tahap pertama yang dilakukan dalam melakukan analisis berorientasi objek menggunakan UML adalah menentukan aktor atau Pemain sistem. Kata aktor dalam konteks UML, menampilkan peran (*roles*) yang Pemain (atau sesuatu di luar sistem yang dikembangkan yang dapat berupa perangkat keras, *end user*, sistem yang lain, dan sebagainya).

3.4.1 *Use Case* Diagram

Use case merupakan gambaran skenario dari interaksi antara *user* dengan sistem. Sebuah diagram *Use case* menggambarkan hubungan antara aktor dan kegiatan yang dapat dilakukannya terhadap aplikasi.



Gambar 3.23 *Scenario Use case* Diagram Eiji The Lost Canvas**.**

Definisi *Use case* berfungsi untuk menjelaskan proses yang terdapat pada setiap *Use case*. Definisinya dapat dilihat pada tabel 3.2.

Tabel 3.2 Definis *Use case.*

No.	*Use case*	Deskripsi
1.	Graphics Config	Proses untuk pengaturan kualitas dan resolusi yang akan digunakan dalam permainan.
2.	Input Config	Proses untuk pengaturan tonbol yang akan digunakan dalam permainan.
3.	*Play*	Proses untuk memulai aplikasi *Game*.
4.	*Play Game*	Proses untuk memulai permainan.
5.	*Guide scene*	Menampilkan scene *Guide*.
6.	*History*	Menampilkan scene *History*.

3.4.2 *Scenario Use Case*

Skenario setiap bagian pada *Use case* menunjukkan proses apa yang terjadi pada setiap bagian didalam *Use case* tersebut, dimana user memberikan perintah pada setiap bagian dan respon apa yang diberikan oleh sistem kepada user setelah user memberikan perintah pada setiap bagian – bagian *Use case*.

Tabel 3.3 Skenario *use case graphics config.*

Identifikasi	
Nomor	**1**
Nama	*Eiji : The Lost Canvas graphics config*
Tujuan	Dapat mengkonfigurasikan graphics pada *Game*
Deskripsi	proses konfigurasi *dari Game Eiji : The Lost Canvas*
Aktor	User
Skenario utama	
Kondisi awal	Pemain berada di dekstop
Aksi aktor	**Reaksi sistem**
1. Pemain mengeksekusi aplikasi.	2. Pemain akan disuguhkan tampilan konfigurasi grafis.
3. Pemain memilih konfigurasi grafis.	4. Sistem menyimpan konfigurasi grafis.
Kondisi akhir	Pemain bisa memulai permainan atau

	mengakhiri aplikasi.

Tabel 3.4 Skenario *use case input config*.

Identifikasi	
Nomor	**2**
Nama	*Eiji : The Lost Canvas input config*
Tujuan	Dapat mengkonfigurasikan iput control pada *Game*
Deskripsi	proses konfigurasi *input dari Game Eiji : The Lost Canvas*
Aktor	User
Skenario utama	
Kondisi awal	Pemain berada di dekstop
Aksi aktor	**Reaksi sistem**
1. Pemain mengeksekusi aplikasi.	2. Pemain akan disuguhkan tampilan konfigurasi input.
3. Pemain memilih konfigurasi	4. Sistem menyimpan konfigurasi input.

input.	
Kondisi akhir	Pemain bisa memulai permainan atau mengakhiri aplikasi.

Tabel 3.5 Skenario *use case Play*.

Identifikasi	
Nomor	**3**
Nama	*Play*
Tujuan	Menampilkan main menu utama dalam *Game*
Deskripsi	proses memulai aplikasi *Game*
Aktor	User
Skenario utama	
Kondisi awal	Pemain berada di dekstop
Aksi aktor	**Reaksi sistem**
1. Pemain mengeksekusi aplikasi.	2. Pemain akan disuguhkan tampilan

	konfigurasi input.
3. Pemain klik tombol *Play*.	4. Sistem menyimpan konfigurasi input dan menampilkan main menu utama dalam *Game*.
Kondisi akhir	Pemain memasuki menu utama untuk memilih menu yang ingin ditampilkan.

Tabel 3.6 Skenario *use case Play Game***.**

Identifikasi	
Nomor	**4**
Nama	*Play Game*
Tujuan	Dapat menjalakan permainan dari *stage* 1 hingga *Final stage*
Deskripsi	proses dari permainan *Game Eiji : The Lost Canvas*
Aktor	user
Skenario utama	
Kondisi awal	Pemain berada di *Main Menu*
Aksi aktor	**Reaksi sistem**

1. Pemain klik tombol *Play Game*.	2. Menerima perintah, dan menampilkan *stage* 1.
3. Memainkan *stage* 1 dan melawan musuh yang ada.	4. Menyimpan data checkpoint yang dilewati pemain.
5. Belanjut ke *stage* 2 hinggan *stage* 3 dan melawan musuh yang ada.	6. Memindahkan *stage* yang telah dilewati pemain hingga *Final stage*.
Kondisi akhir	Permainan berakhir dan kemabali ke *Main Menu*.

Tabel 3.7 Skenario *use case Guide scene***.**

Identifikasi	
Nomor	**5**
Nama	*Guide scene*
Tujuan	Menampilkan *Guide scene*
Deskripsi	Proses pemilhan untuk menampilkan Guide
Aktor	User

Skenario utama	
Kondisi awal	Pemain berada di *Main Menu*
Aksi aktor	**Reaksi sistem**
1. Pemain klik tombol Guide	2. Menampilkan *Guide scene*
	3. Menampilkan petunjuk *Game*
Kondisi akhir	Pemain kembali ke *Main Menu*.

Tabel 3.8 Skenario *use case History*.

Identifikasi	
Nomor	**6**
Nama	*History*
Tujuan	Menampilkan *History*
Deskripsi	Proses pemilhan untuk menampilkan *History*
Aktor	User
Skenario utama	

Kondisi awal	Pemain berada di *Main Menu*
Aksi aktor	**Reaksi sistem**
1. Pemain klik icon *History*	2. Menampilkan *History Game*
Kondisi akhir	Pemain kembali ke *Main Menu*

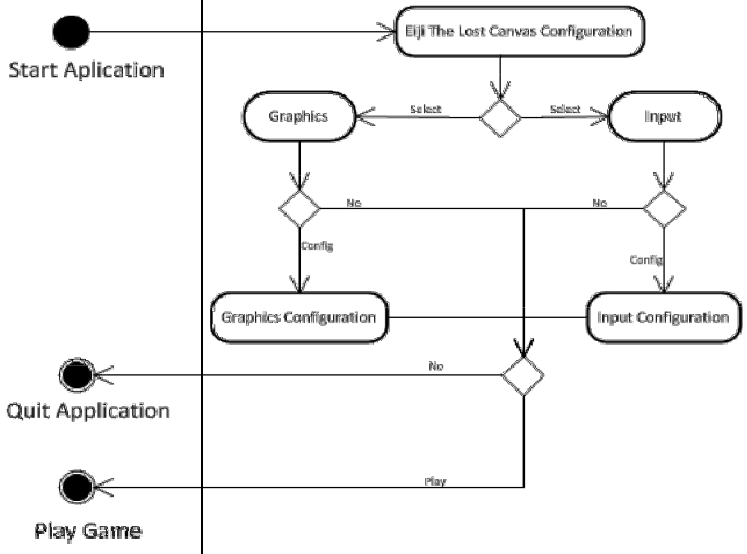
3.4.3 *Activity Diagram*

Activity Diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity Diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. *Activity Diagram* merupakan *state diagram* khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *Activity Diagram* tidak menggambarkan behaviour internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

1. **Proses *Eiji : The Lost Canvas configuration***

Tabel 3.9 Proses konfigurasi permainan**.**

Deskripsi	proses *konfigurasi dari Game Eiji : The Lost Canvas.*
Nama Flow	1. Pemain mulai menjalakan aplikasi. 2. Pemain akan disuguhkan tampilan konfigurasi. 3. Pemain bisa memilih kualitas grafis yang akan dipakai. 4. Pemain bisa memilih resolusi *Game* yang akan dipakai. 5. Pemain bisa mengatur tombol-tombol yang akan dipakai. 6. Pemain bisa memulai permainan. 7. Pemain bisa mengakhiri aplikasi.

User	**System**
	

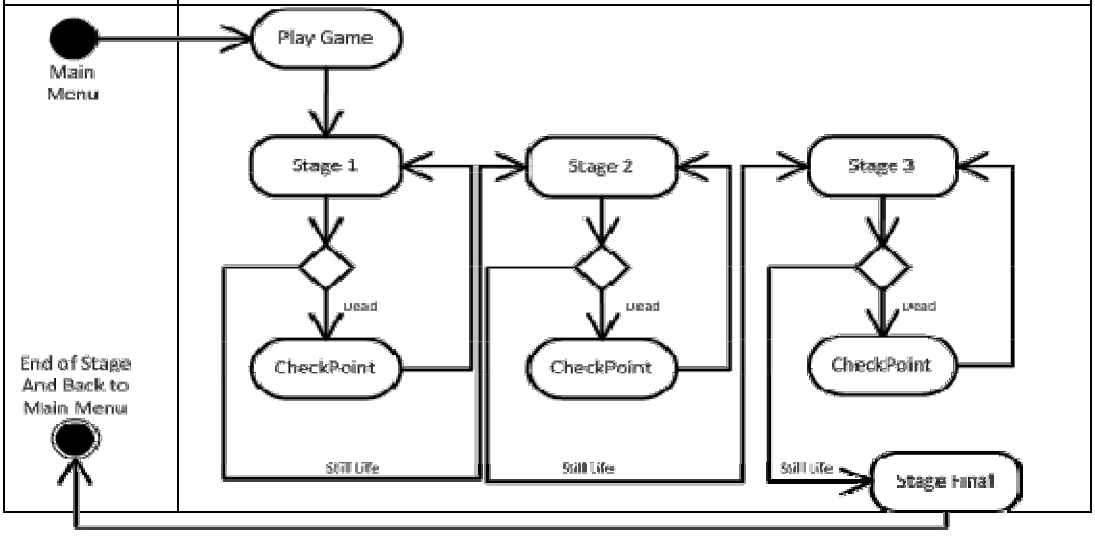
Gambar 3.24 *Activity Diagram* Eiji : The Lost Canvas configuration**.**

2. **Proses *Play Game***

Tabel 3.10 Proses *Play Game***.**

Deskripsi	proses dari permainan *Game Eiji : The Lost Canvas.*
Nama Flow	1. Pemain mulai permainan. 2. Pemain akan disuguhi tempilan loading. 3. Pemain memasuki *Stage* 1. 4. Pemain mulai berpetualang. 5. Ketika *Player* mati, maka akan kembali ke checkpoint yg telah dilewati. 6. *Stage* 1 berakhir dan berpindah ke *Stage* 2. 7. Pemain mulai berpetualang. 8. Ketika *Player* mati, maka akan kembali ke checkpoint yg telah dilewati. 9. *Stage* 2 berakhir dan berpindah ke *Stage* 3.

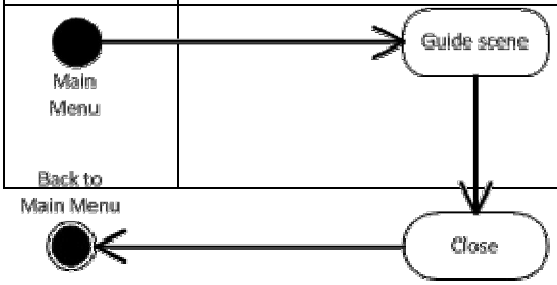
	10. Pemain mulai berpetualang. 11. Ketika *Player* mati, maka akan kembali ke checkpoint yg telah dilewati. 12. *Stage* 3 berakhir dan akan berpindah ke *Final Stage*. 13. Sesudah *Final Stage* dilewati, maka permainan berakhir. 14. Kembali ke *Main Menu*

User	**System**
	

Gambar 3.25 *Activity Diagram Play Game***.**

3. **Proses pemilihan *Guide***

Tabel 3.11 Proses pemilihan *Guide***.**

Deskripsi	Proses pemilihan untuk menampilkan Guide
Nama Flow	1. Pemain klik tombol *Guide*. 2. Pemain akan disuguhkan dengan tampilan *Guide Game*. 3. Ketika pemain klik tombol Close, maka pemain akan kembali ke menu utama.
User	**System**
	

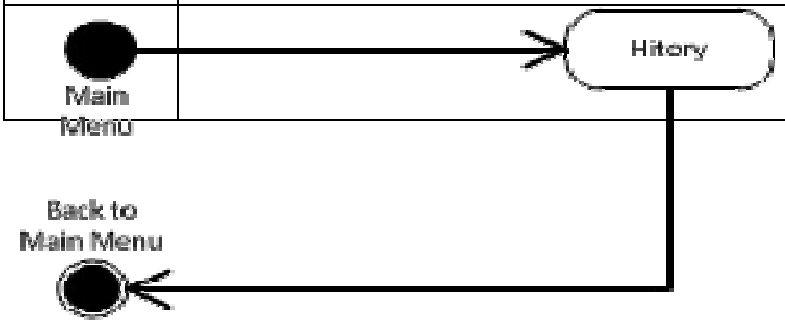
Gambar 3.26 *Activity Diagram* Guide**.**

4. **Proses Pemilihan *History***

Tabel 3.12 Proses pemilihan *History***.**

Deskripsi	Proses pemilhan untuk menampilkan *History*
Nama Flow	1. Pemain klik tombol *History*. 2. Pemain akan disuguhkan dengan tampilan *History Game*. 3. Ketika pemain klik tombol Menu, maka pemain akan kembali ke menu utama.

User	**System**

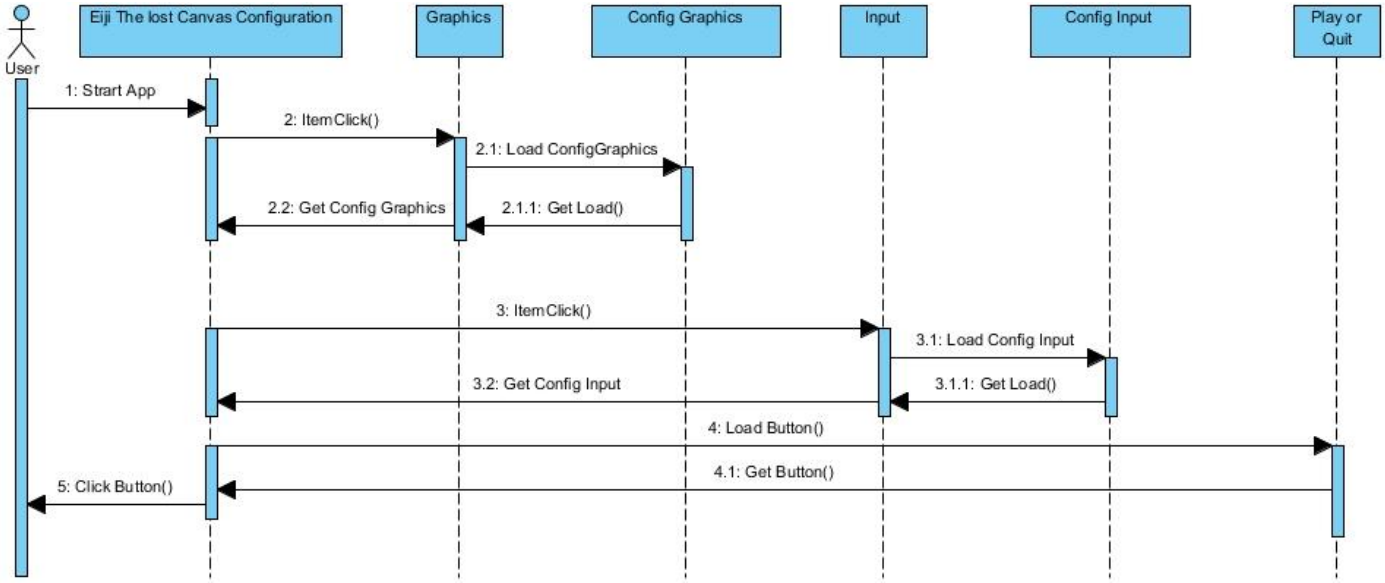


Gambar 3.27 *Acitivity Diagram History***.**

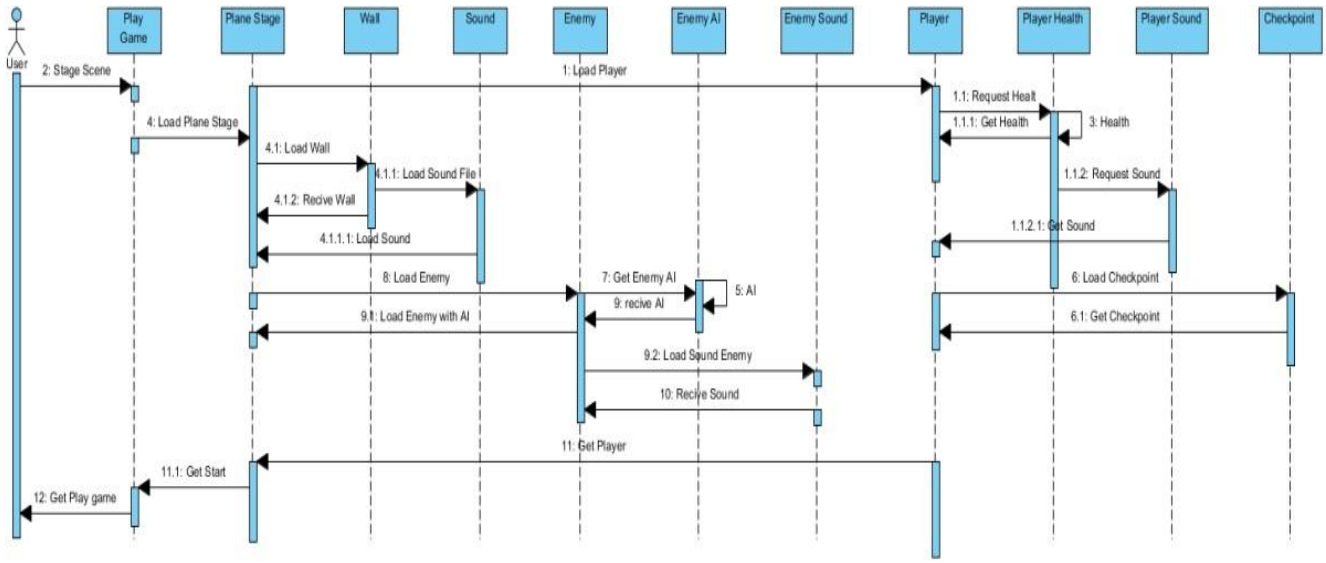
3.4.4 *Sequence Diagram*

Sequence Diagram menggambarkan interaksi antar objek di dalam dan disekitar sistem (termasuk Pemain, display, dan sebagainya). *Sequence Diagram* terdiri atas dimensi vertical (waktu) dan dimensi horizontal (objek-objek yang terkait).

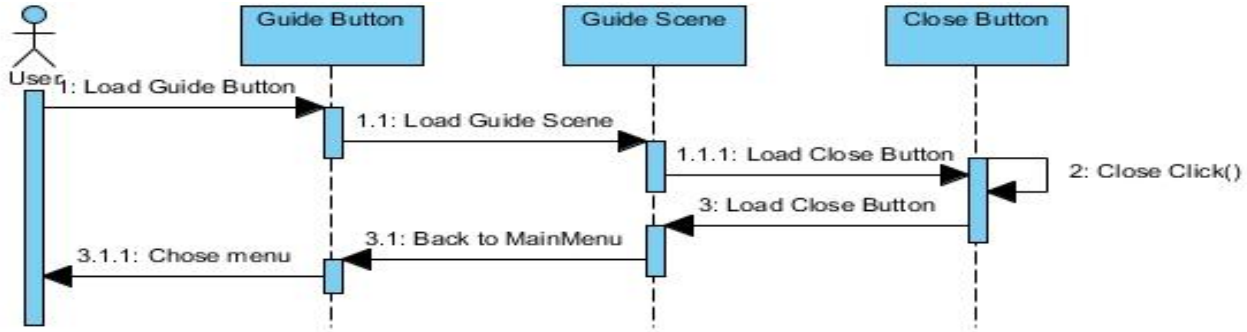
Pada *Sequence Diagram* ini menggambarkan rangkaian langkah-langkah yang dilakukan pada sistem sebagai respon dari sebuah *event* untuk menghasilkan *output* tertentu.



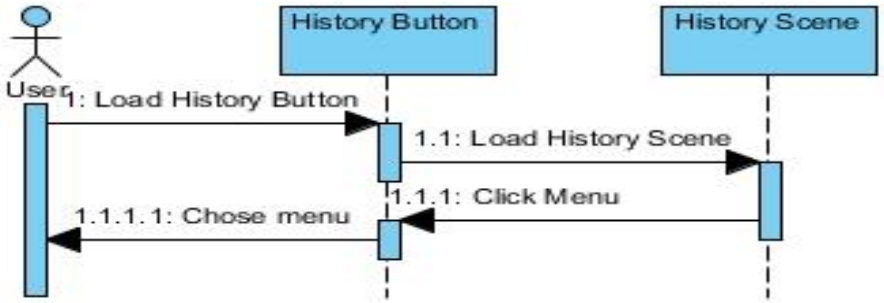
Gambar 3.28 *Sequence Diagram* Eiji The lost Canvas Configuration**.**



Gambar 3.29 *Sequence Diagram* Proses *Play Game***.**

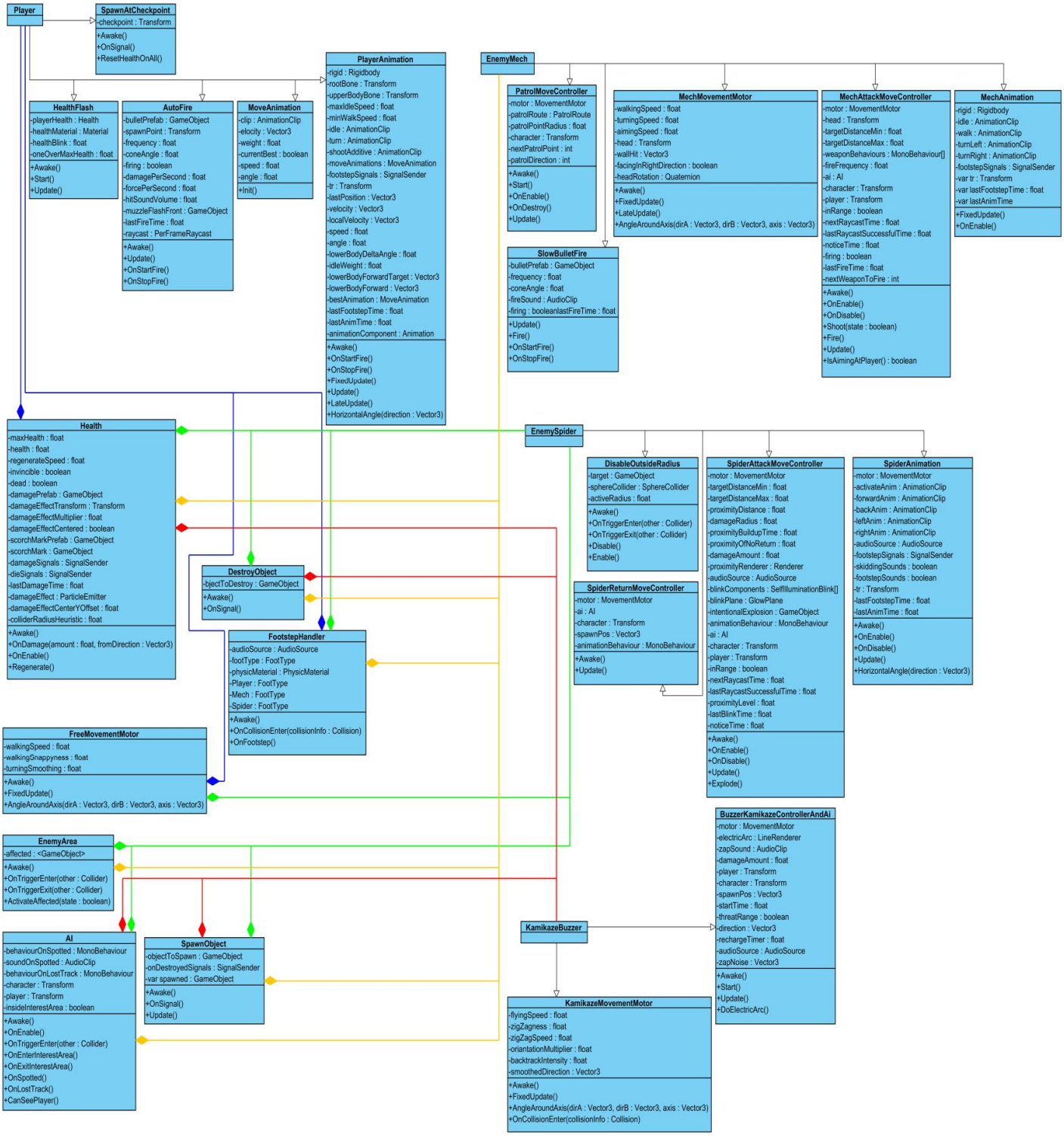


Gambar 3.30 *Sequence Diagram* Guide**.**



Gambar 3.31 *Sequence Diagram History***.**

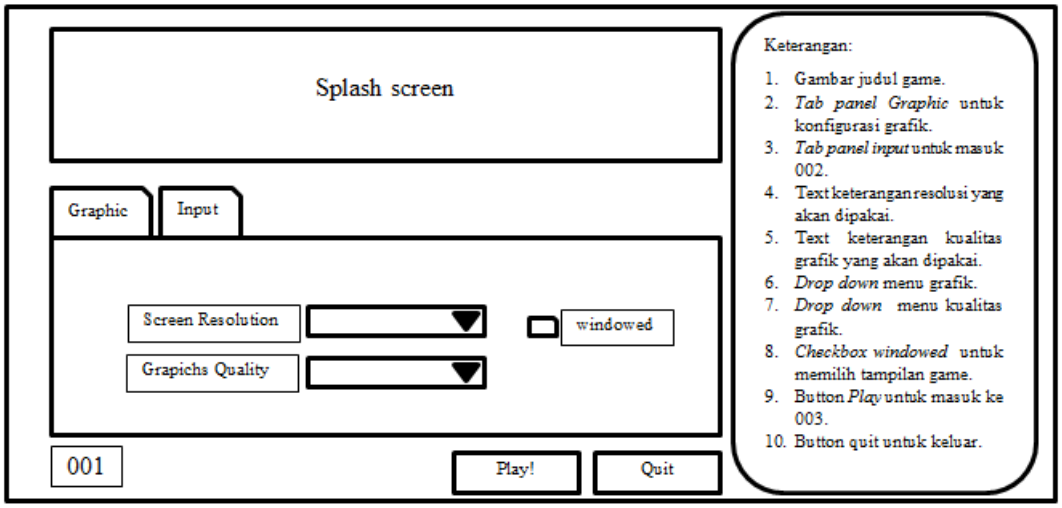
3.4.5 *Class Diagram*



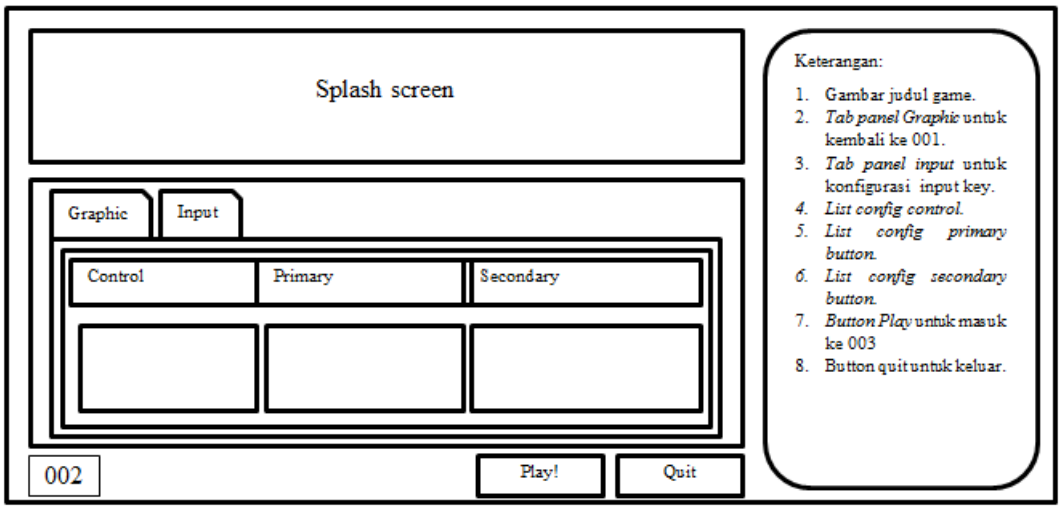
Gambar 3.32 *Class Diagram* Eiji The Lost Canvas**.**

3.5 Perancangan Antar Muka

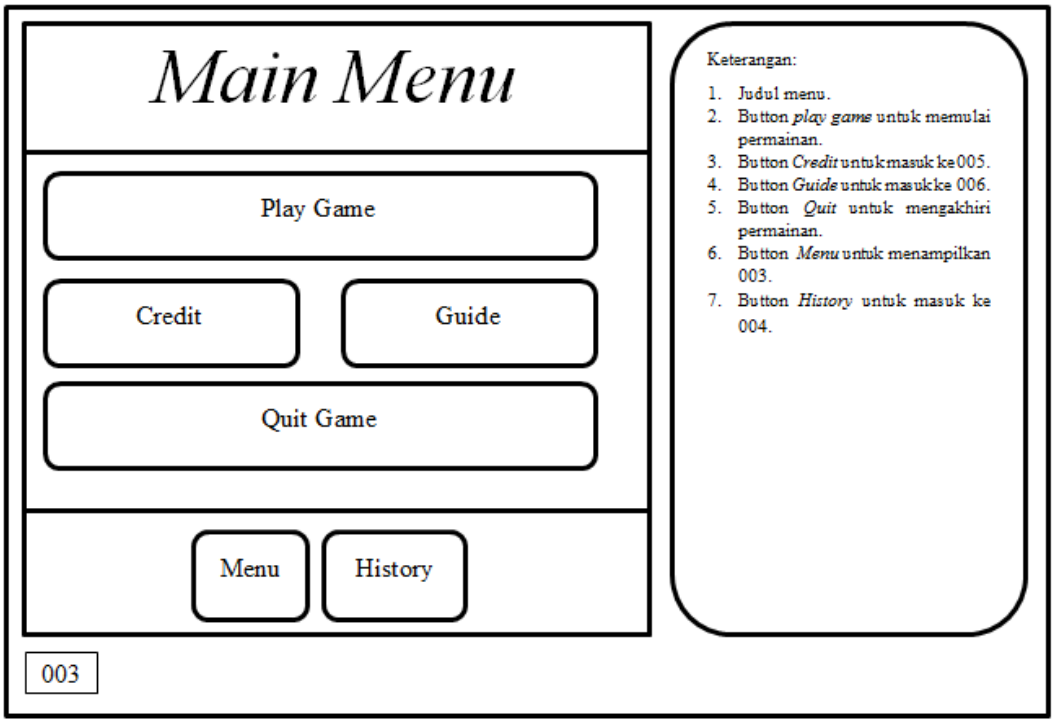
Perancangan antarmuka dibutuhkan untuk mewakili keadaan sebenarnya dari aplikasi yang akan dibangun, berikut akan disajikan perancangan antarmuka dari aplikasi yang akan dibangun:



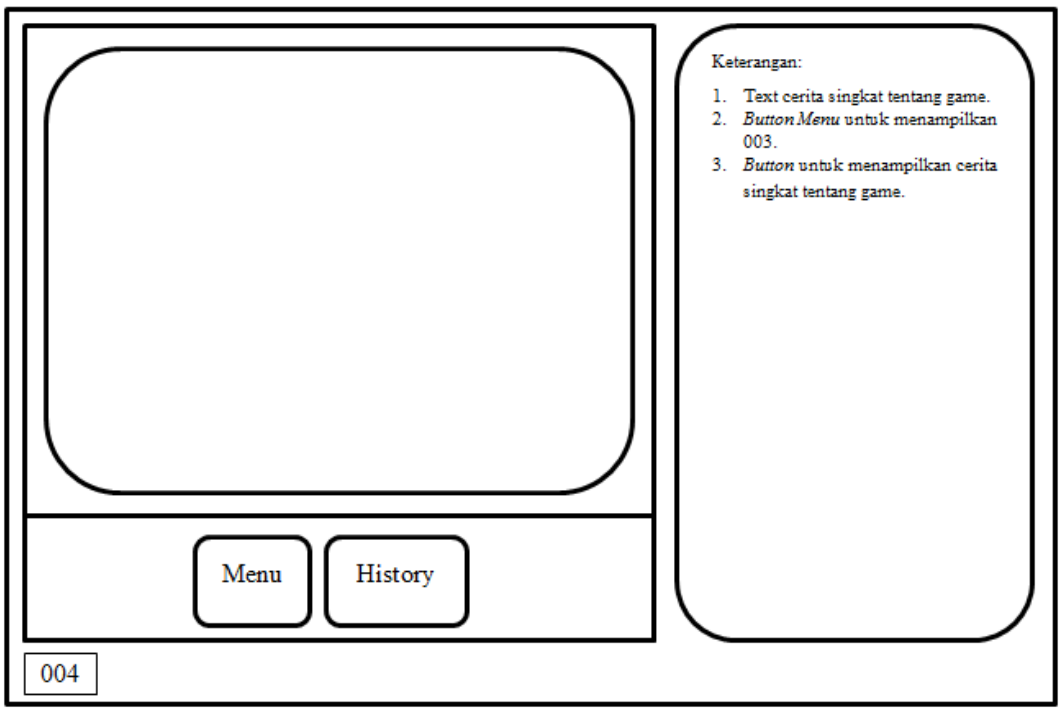
Gambar 3.33 Tampilan awal dan konfigurasi Graphic 001.



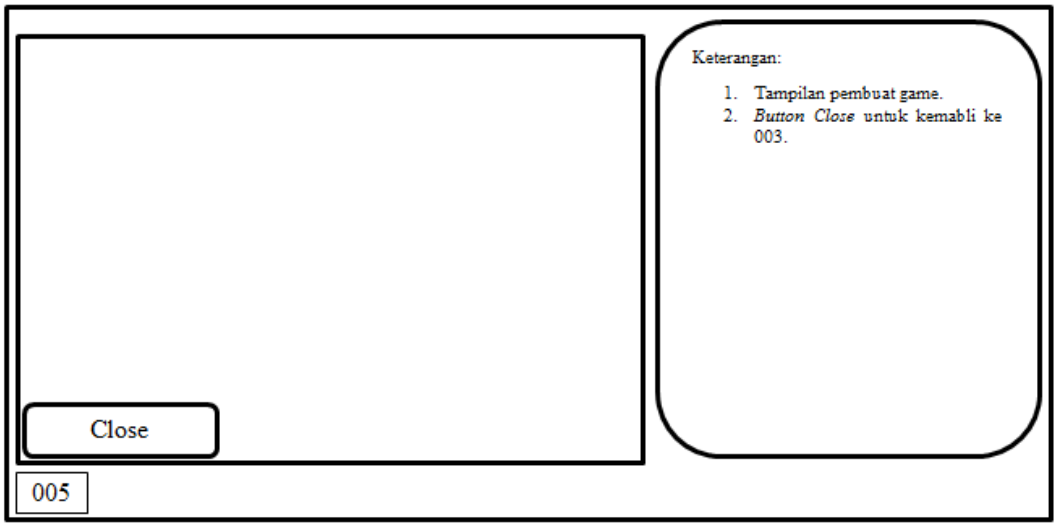
Gambar 3.34 Tampilan konfigurasi Input 002.



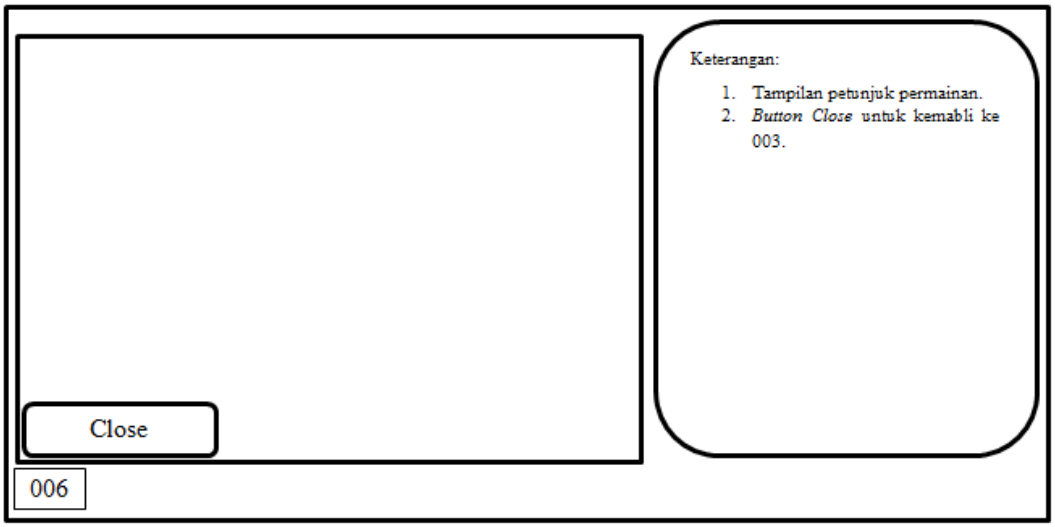
Gambar 3.35 Tampilan Main Menu 003.



Gambar 3.36 Tampilan History 004.



Gambar 3.37 Tampilan Credit 005.



Gambar 3.38 Tampilan Guide 006.