BAB 3
ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Sistem

Pada tahapan analisis sistem akan dilakukan berbagai macam analisis yang berhubungan dengan *game* yang akan dibangun, mulai dari analisis terhadap *game* yang sejenis, analisis berbagai kebutuhan baik fungsional dan maupun kebutuhan *no*n fungsional.

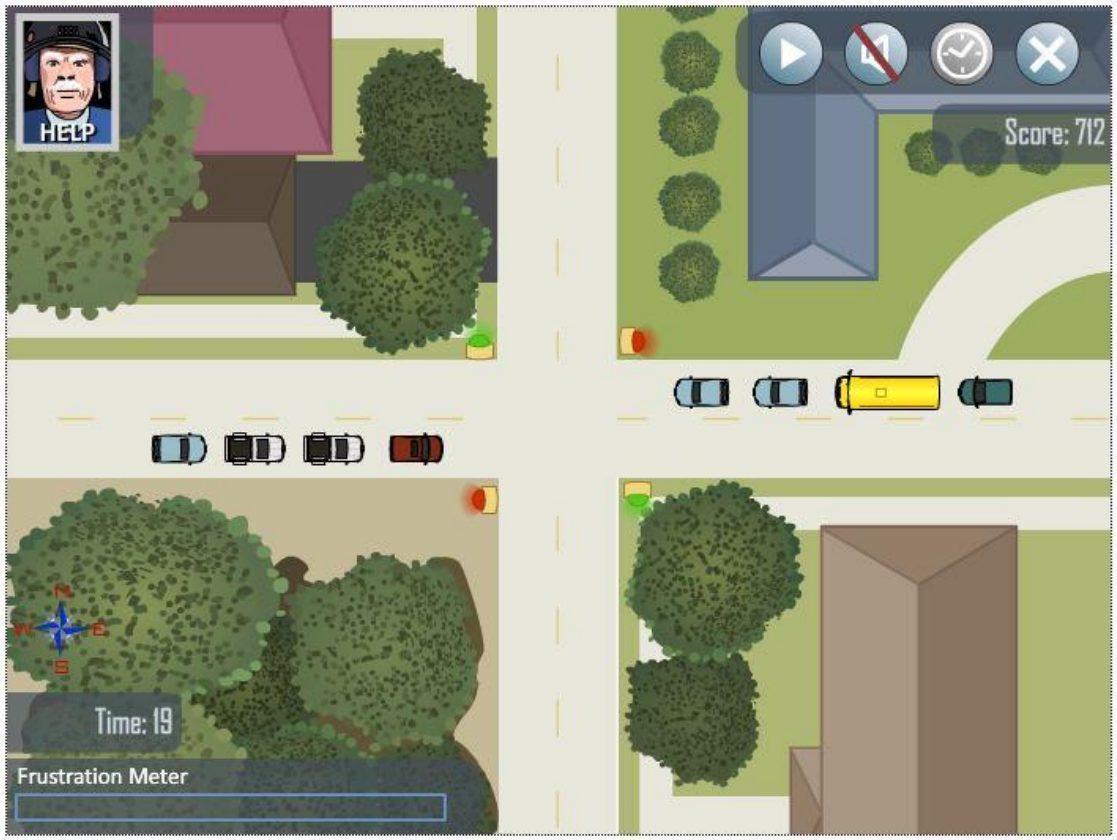
3.2 Analisis *Game* Sejenis

Pada analisis *game* sejenis, akan dilakukan pengematan terhadap *game* yang memiliki kesamaan *genre*, tema, dan *game*play dengan *game* yang akan dibangun. Tujuan dari analisis pada tahap ini yaitu untuk mendapatkan konsep permainan dan perbandingan dari *game* yang telah ada.

3.2.1 *Game Gridlock Buster*

Game Gridlock Buster merupakan salah satu *game* dengan *genre simulasi* yang berlatar belakang pengendalian lampu lalu lintas. *Game* ini dibangun dengan tampilan grafis 2D (dua dimensi) dan dikembangkan untuk pengguna *game* browser. *Game* ini dikembangkan oleh *ITS Institute (Intelligent Transportation System)* dari *University of Minnesota* yang merupakan pusat studi dan pengembangan tek*no*logi transportasi. Ide pembuatan *game* ini muncul berdasarkan dari bagaimana proses keseharian seorang *engineers traffic control* bekerja.

Berikut adalah tampilan dari *game Gridlock Buster* terdapat pada gambar 3.1 :



Gambar 3.1 *Screenshot* Tampilan *Game Gridlock Buster*

1. *Storyline*

Disebuah kota modern terdapat sebuah markas dari lembaga yang bernama *central bureau of traffic control*, markas tersebut merupakan tempat bagi para *engineers traffic* bekerja. Para *engineers traffic* bekerja sebagai pengawas dan pegendali lalu lintas melalui sebuah sistem yang dapat memantau keadaan lalu lintas dari kejauhan.

Pada *game* ini pemain akan berperan sebagai seorang *engineers traffic* baru yang akan menggantikan petugas lamanya yang bernama Crosstown, Crosstown bekerja sebagai *engineer traffic* sejak tahun 1987, dan hingga sekarang dia ditugaskan sebagai *assistant* pemain dalam menjalankan pekerjaan sebagai *engineers traffic*.

2. *Game*play

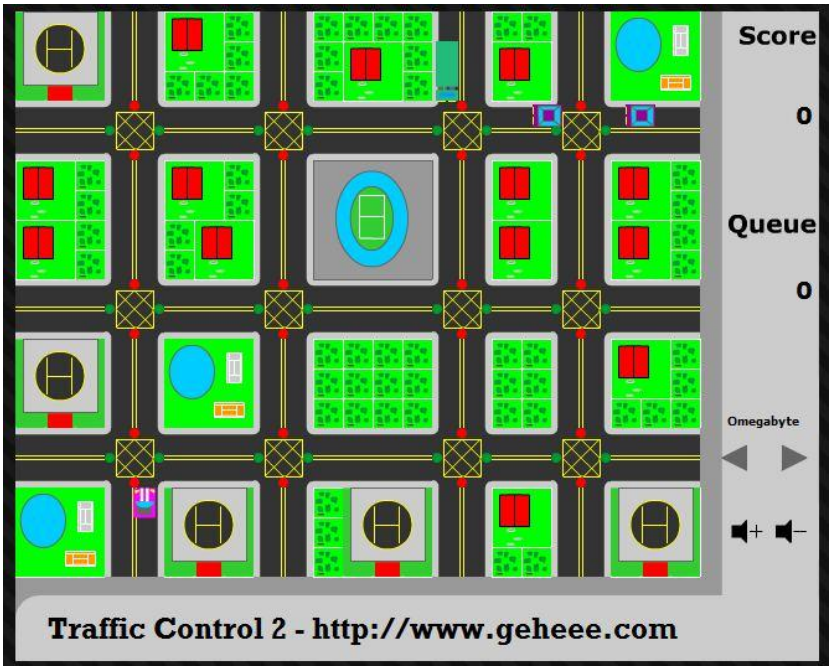
Pada awal permainan ini pemain bertugas menangani lalu lintas di sebuah persimpangan jalan dengan cara mengendalikan lampu lalu lintas pada jalan tersebut. Pemain akan ditugaskan untuk dapat menangani *delay* dan *queue* dari kendaraan yang akan melewati persimpangan tersebut. *Delay* dalam *game* ini digambarkan dengan *frustration bar level* dimana *bar level* ini menggambarkan tingkat stres pengguna jalan ketika menunggu durasi lampu-lalu lintas, apabila *frustaion bar level* ini penuh maka permainan akan dianggap *game over*. Sementara itu, *queue* dalam *game* ini merupakan banyaknya antrian kendaraan pada persimpangan, setiap kendaraan yang berhasil tiba hingga ujung jalan maka pemain akan diberikan sejumlah skor yang nantinya akan di akumulasikan sebagai skor misi. Apabila pemain tidak dapat memenuhi *mission score* yang ditentukan maka pemain tidak dapat meneruskan ke *level* berikutnya. Dalam *game* ini juga terdapat penghitungan waktu yang tersedia pada setiap level, apabila waktu yang ditentukan telah habis maka *game* akan selesai dan memulai pengecekan dari skor misi.

Pada *level* selanjutnya pemain akan dihadapkan dengan masalah yang sama dengan *level* sebelumnya. Namun, setiap tahapan *level* jumlah dari persimpangan jalan akan bertambah satu disetiap *level*. Hal ini akan semakin menambah tingkat kesulitan yang akan dihadapi pemain.

3.2.2 *Game Traffic Control 2*

Sama halnya dengan *game* sejenis yang di analisis sebelumnya *Game Traffic Control 2* juga merupakan *game* dengan *genre simulasi* yang berlatar belakang pengendalian lampu lalu lintas. Bila dibandingkan dengan *game* sejenis yang sebelumnya *Game Traffic Control* juga memiliki tampilan grafis 2D, dan *game*play yang sama. Namun, kualitas *model* karakter pada *game* ini tidak lebih baik dibandingan dengan *game* sejenis sebelumnya.

Berikut adalah tampilan dari *game Traffic Control 2* terdapat pada gambar 3.2 :



Gambar 3.2 *Screenshot* Tampilan *Game Traffic Control 2*

1. *Storyline*

Pada *game* Traffic Control 2 ini tidak terdapat *storyline*, karena tidak terdapat karakter yang dapat dijadikan tokoh utama sebagai alur cerita.

1.

2. *Game*play

Pada awal permainan ini pemain bertugas menangani lalu lintas di sebuah kota dengan cara mengendalikan lampu lalu lintas yang terdapat pada tiap persimpangan jalan yang ada. Pemain ditugaskan untuk menangani antrian dari kendaraan yang akan melewati lampu lalu lintas, pemain akan mendapatkan score yang lebih banyak apabila dapat membuat kendaraan yang muncul dapat tiba ketujuanya dengan cepat. Apabila dalam proses bermain pemain membiarkan terjadinya penumpukan kendaraan sebanyak 6 kendaraan maka permainan akan dianggap selesai *(game over).*

Pada *game* ini hanya terdapat 1 level. Namun, tingkat kesulitanya terdapat pada durasi permainan. Sehingga, semakin lama pemain dapat mengatur antrian kendaraan maka kecepatan waktu munculnya kendaraan juga semakin cepat. Hal ini akan semakin menambah tingkat kesulitan yang akan dihadapi oleh pemain.

3.2.3 Perbandingan *Game* Sejenis

Perbandingan *game* sejenenis dilakukan untuk mengetahui kelebihan dan kekurangan masing-masing yang terdapat pada *game* pembanding. Penilaian dilakukan pada beberapa komponen tertentu, seperti tampilan grafis, *storyline*, penerapan kecerdasan buatan, jenis tantangan, dan penerapan konsep simulasi. Perbandingan dari kedua *game* sejenis dapat dilihat pada tabel 3.1 :

Tabel 3.1 Hasil Perbangingan Game Sejenis

Komponen	Gridlock Buster	Traffic Control 2
Tampilan Grafis	2D, Dengan gambar-gambar karakter, *back*ground kota, model mobil yang menyerupai bentuk aslinya.	2D, Model mobil tidak mirip dengan aslinya
Storyline	*Storyline* tentang seorang pekerja yang berprofesi sebagai traffic engineer	Tidak terdapat *storyline*
Kecerdasan buatan	Tidak terdapat kecerdasan buatan pada mobil agar dapat menetukan jalur sendiri	Tidak terdapat kecerdasan buatan pada mobil agar dapat menetukan jalur sendiri
Tantangan	Terdapat stress meter, score, timer.	Jumlah antrian maksimal
Kemiripan cara kerja lampu lalu lintas dengan kondisi aslinya	Pengendalian lampu lalu lintas hanya bersifat manual, sementara lampu lalu lintas dalam keadaan aslinya memiliki timer.	Pengendalian lampu lalu lintas hanya bersifat manual, sementara lampu lalu lintas dalam keadaan aslinya memiliki timer.
Kemiripan sifat dinamis pengguna jalan dengan	Mobil yang bermunculan hanya bersifat statis.	Mobil yang bermunculan berdasarkan score, bukan berdasarkan jam sibuk.

kondisi aslinya		

Dari hasil tabel perbandingan *game* sejenis yang telah dilakukan maka disimpulkan komponen-komponen yang akan diterapkan pada *game* yang akan dibangun nantinya. Berikut penjelasan komponen-komponen yang akan digunakan :

1. Tampilan grafis : Menggunakan model 3D, dengan tampilan *back*ground perkotaan, dan model grafik mobil yang memiliki kemiripan dengan aslinya.
2. *Storyline* : Tidak terdapat *storyline*, karena tidak terdapatnya model karakter yang digunakan sebagai tokoh utama.
3. Kecerdasan buatan : Menggunakan algoritma A* (A-STAR) pathfinding yang akan diterapkan pada *NPC* mobil agar dapat menentukan jalur terpendek dimulai dari posisi spawn hingga ke tujuan.
4. Tantangan : Menggunakan *Score*, misi keadaan darurat yang harus terpenuhi, misi *score* dan menggunakan *timer* untuk keadaan darurat.
5. Komponen konsep simulasi :

a. Lampu lalu lintas yang digunakan berbasis *timer* yang terurut di setiap persimpanganya. Namun, tetap dapat dikendalikan secara *manual* agar dapat mengurangi kemacetan dan penanganan keadaan darurat lebih cepat.

b. Menerapkan jam-jam sibuk pengguna jalan seperti keadaan aslinya, dimana mobil yang akan muncul sebagai tantangan akan semakin banyak pada jam-jam yang telah ditentukan.

3.3 Analisis *Game* yang dibangun

Pada tahapan ini akan dilakukan analisis terhadap *game* yang akan dibangun. Bagian ini terdiri dari pengenalan, *storyline*, tingkat kesulitan, *game*play dan *scoring*.

3.3.1 Pengenalan

Game simulasi pengendali lampu lalu lintas terpusat merupakan *game* yang dengan *genre* simulasi. Dalam *game* yang dibangun ini pemain berperan sebagai pengendali lampu lalu lintas pada sebuah kota yang memiliki banyak aktifitas atau kejadian-kejadian yang umumnya sering terjadi pada kehidupan nyata, misalnya kebakaran dan panggilan ambulans. *Game* ini merupakan *game* dengan sistem single player dan dibangun dengan grafis 3D.

3.3.2 *Gameplay*

Dalam *game* yang akan dibangun ini tugas utama pemain adalah mengendalikan lampu lalu lintas yang terdapat pada tiap persimpangan jalan, dimana tiap tahapan waktu memiliki tingkat kesulitan yang berbeda, semakin tinggi tahapan yang dilewati maka tingkat kesulitan akan semakin bertambah. Untuk tantangan keadaan darurat pemain harus dapat membuat mobil pemadam kebakaran atau mobil ambulans dapat tiba pada tujuanya dalam waktu yang sudah ditentukan, apabila gagal dalam menyelesaikan tantangan tersebut maka *gold* pemain akan dikurangi. Permainan pada tiap tahapan akan selesai apabila dalam tahapan waktu pada yang sudah ditentukan pemain tidak dapat memenuhi ketentuan misi keadaan darurat dan score yang dibutuhkan maka pemain tidak dapat melanjutkan ketahapan berikutnya dan pemain harus mengulang kembali tahapan tersebut agar dapat meneruskan ke tahapan berikutnya.

3.3.3 Tingkat Kesulitan

Dalam *game* ini, tingkat kesulitan akan semakin bertambah berdasarkan waktu permainan. Tingkat kesulitan pada *game* yang dibangun dijelaskan sebagai berikut :

1. Waktu 00.00 - 05.00. Pada tahapan ini kecepatan waktu jeda kemunculan mobil adalah 4 detik untuk 1 mobil yang dimunculkan secara acak pada tiap persimpangan jalan, dan akan terdapat 5 keadaan emergensi yang muncul secara acak selama waktu berjalan. Selain itu pemain juga harus dapat memenuhi tantangan score sebanyak 1000.

2. Waktu 05.00 – 12.00. Pada tahapan ini kecepatan waktu jeda kemunculan mobil adalah 3 detik untuk 1 mobil yang dimunculkan secara acak pada tiap persimpangan jalan, dan akan terdapat 10 keadaan emergensi yang muncul secara acak selama waktu berjalan. Selain itu pemain juga harus dapat memenuhi tantangan score sebanyak 1800.
3. Waktu 12.00 – 18.00. Tahapan ini merupakan tahapan tersulit dalam *game* ini karena kecepatan waktu jeda kemunculan mobil adalah 2 detik untuk 1 mobil yang dimunculkan secara acak pada tiap persimpangan jalan, dan akan terdapat 15 keadaan emergensi yang muncul secara acak selama waktu berjalan. Selain itu pemain juga harus dapat memenuhi tantangan score yang telah ditentukan sebanyak 2500.

3.3.4 Scoring

Pada *game* yang akan dibangun terdapat *reward gold* yang diberikan setiap menyelasaikan tantangan. Jumlah *gold* yang didapat tingkat kesulitan tantangannya sebagai berikut :

1. Tantangan mobil umum : +10 *gold*
2. Tantangan mobil pemadam kebakaran : +50 *gold*
3. Tantangan mobil ambulans : +50 *gold*

Pada *game* ini juga terdapat pengurangan *gold* apabila pemain melakukan kesalahan dalam proses permainan :

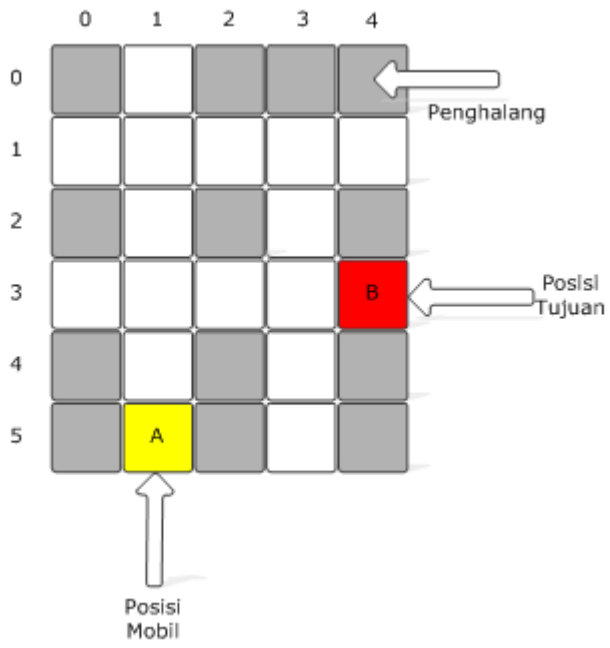
1. Gagal tantangan mobil pemadam kebakaran : -50 *gold*
2. Gagal tantangan mobil ambulans : -50 *gold*

3.4 Analisis Algoritma

Pada analisis algoritma ini dilakukan analisis tentang algoritma yang nantinya digunakan pada *game* yang akan dibangun. Analisis algoritma yang digunakan adalah analisis algoritma A*(A *Star*) dan akan diterapkan pada *NPC* mobil agar dapat menentukan jalur terpendek berdasarkan posisi kemunculan dengan posisi tujuanya.

3.4.1 Analisis A*(A *Star*)

Algoritma A* akan diterapkan pada *NPC* mobil dalam melakukan penentuan jalur terpendek atau *pathfinding* menuju target dalam arena permainan. Cara kerja algortima A* dalam *game* yang akan dibangun terdapat pada Gambar 3.3.



Gambar 3.3 Cara Kerja Algoritma A*

Pada gambar 3.3 terdapat beberapa kotak yang memiliki koordinat dari (0,1) sampai (4,4). Kotak atau *node* yang masing-masing telah memiliki koordinat merupakan *node* yang digunakan untuk melakukan pencarian jalur terpendek /

pathfinding. Terdapat beberapa *node* yang memiliki warna berbeda, berikut penjelasan dari masing-masing *node* tersebut :

1. *Node* warna biru : *Node* awal dari posisi mobil yang berada pada *node* di koordinat (5,1).
2. *Node* warna abu : *Node* penghalang yang berada pada *node* di koordinat (0,0), (0,2), (0,4),(0,5),(2,0),(2,2), (2,4),(3,5), (4,0), (4,2),(4,4) dan (4,5).
3. *Node* warna merah : *Node* dari posisi tujuan yang berada pada *node* di koordinat (3,4).

Untuk nilai-nilai yang digunakan untuk perhitungan a-*star*, terdapat 3 nilai yaitu :

1. Nilai f, nilai yang diperoleh dari penjumlahan nilai g, jumlah nilai tiap simpul dalam jalur terpendek dari *Starting point* ke *current node*.
2. Nilai g, nilai atau biaya yang dikeluarkan dari *node* awal / *node* A ke *node* yang ada di sekitarnya.
3. Nilai h, nilai *heuristic* yang digunakan untuk menghitung perkiraan biaya yang dikeluarkan dari *node* awal / *node* A menuju *node* tujuan / *node* B.

Perhitungan nilai f didapat dari persamaan f= g + h. Sedangkan untuk nilai g diberikan nilai satu untuk gerakan *vertical* maupun *horizontal*. Untuk perhitungan nilai h digunakan fungsi *heuristic*, metode yang digunakan di dalam contoh ini adalah metode *Manhattan* dimana perhitungan jumlah *node* hanya yang bergerak secara *vertical* dan *horizontal* menuju tujuannya serta mengabaikan penghalang. Atau dirumuskan dengan:

$$h = (\text{abs}(current\text{X} - target\text{X}) + \text{abs}(current\text{Y} - target\text{Y}))$$

Pada pencarian jalur terpendek oleh A *star* ada beberapa istilah yang digunakan seperti *open list, closed list, parrent, current node* dan *pointer*.

1. *Open list* adalah *list* untuk memeriksa *node-node*.

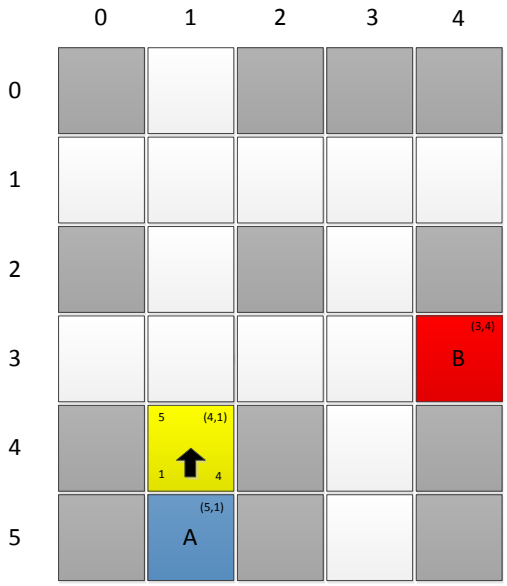
2. *Closed list* adalah *node-node* yang diabaikan atau *node-node* yang tidak perlu diperiksa.
3. *Parrent* adalah digunakan untuk menelusuri jalan.
4. *Current Node* adalah simpul yang sedang dijalankan dalam algoritma pencarian jalan terpendek.
5. *Pointer* adalah penunjuk jalan terhadap *Node* A *(back* pointer) ketika *node* B (target) telah ditemukan.

Langkah berikutnya ketika telah mendapatkan *node* awal adalah menentukan rute terpendek dengan cara memeriksa setiap *node* yang berdekatan dengan *node* A sampai *node* B. Di bawah ini merupakan penjelasan langkah-langkahnya.

1. Langkah pertama adalah memasukan *node* A kedalam *open list*.
2. Langkah kedua adalah memeriksa *node-node* yang berdekatan dengan *node* A, abaikan *node* dengan penghalang lalu tambahkan semua *node* tersebut kedalam *open list*. *Node* A sendiri nantinya akan menjadi "*parent*" dan digunakan untuk menelusuri jalan.
3. Langkah ketiga adalah menghapus *node* A dari *open list*, dan dimasukan ke dalam *closed list*.

Pada langkah pertama, *node* (5,1) dimasukan ke dalam open list dan selanjutnya memeriksa *neighbor node* atau *node* yang berdekatan dengan *node* (5,1) yaitu, *node* (4,1). Setelah diperiksa *node* tersebut dimasukan ke dalam *open list*. Hapus *node* (5.1) dari *open list* dan masukan *node* tersebut kedalam *closed list.* Lalu dari *node-node* yang berada di dalam *open list* pilih *node* dengan nilai F yang paling rendah, yaitu *node* (4,1) dengan nilai F = 5.

Berikut ini perhitungan pada langkah satu yang berada pada *node* dengan koordinat (5,1) terdapat pada gambar 3.4.



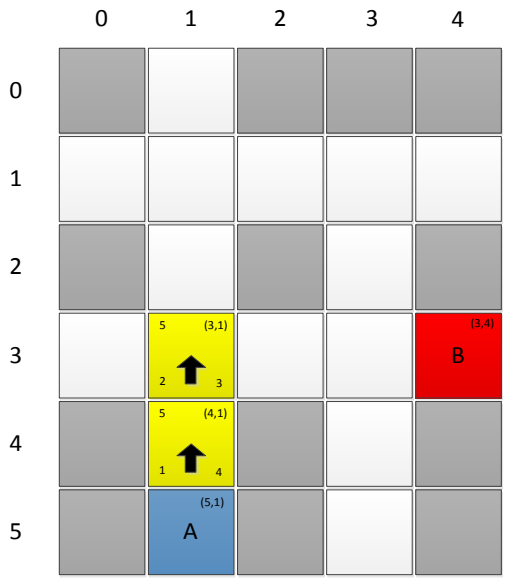
Gambar 3.4 Langkah Satu Perhitungan Algoritma A*

Langkah satu menghasilkan *node-node* di dalam *open list* dan *closed list* dijelaskan dengan Tabel 3.2.

Tabel 3.2 Perhitungan Langkah Satu

Open List	(5,1)
Closed list	(4,1)
Parent	(4,1)
Current node	(4,1)

Berikut ini perhitungan pada langkah dua yang berada pada *node* dengan koordinat (4,1) terdapat pada gambar 3.5.



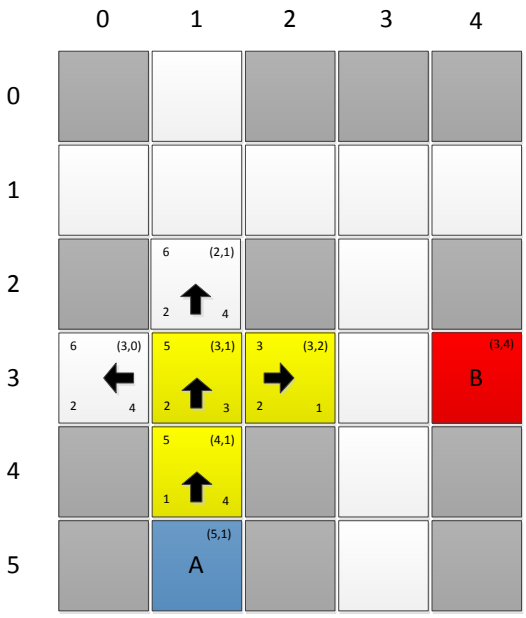
Gambar 3.5 Langkah Ke-Dua Perhitungan Algoritma A*

Langkah dua menghasilkan *node-node* di dalam *open list* dan *closed list* dijelaskan dengan Tabel 3.3.

Tabel 3.3 Perhitungan Langkah Dua

Open List	(3,1)
Closed list	(4,1)
Parent	(4,1)
Current node	(4,1)

Berikut ini perhitungan pada langkah tiga yang berada pada *node* dengan koordinat (3,1) terdapat pada gambar 3.6.



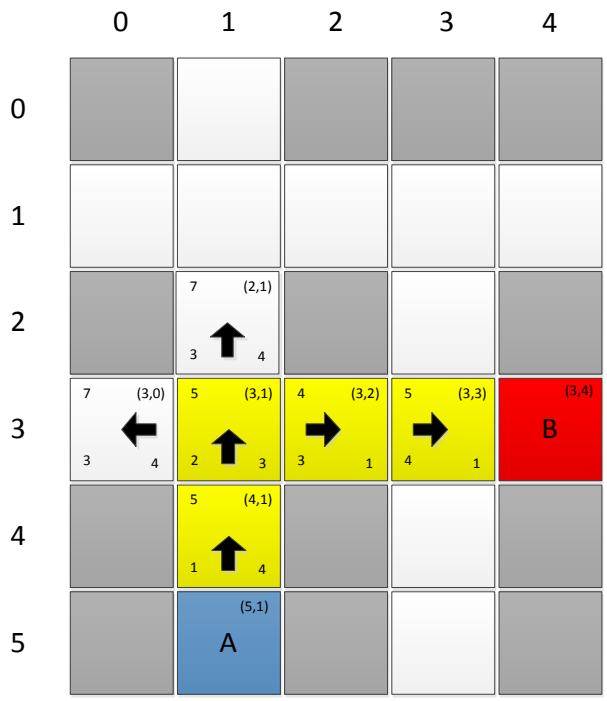
Gambar 3.6 Langkah Ke-Tiga Perhitungan Algoritma A*

Langkah tiga menghasilkan *node-node* di dalam *open list* dan *closed list* dijelaskan dengan Tabel 3.4.

Tabel 3.4 Perhitungan Langkah Tiga

Open List	(3,0),(2,1),(3,2)
Closed list	(3,1)
Parent	(3,1)
Current node	(3,2)

Berikut ini perhitungan pada langkah empat yang berada pada *node* dengan koordinat (3,2) terdapat pada gambar 3.7.



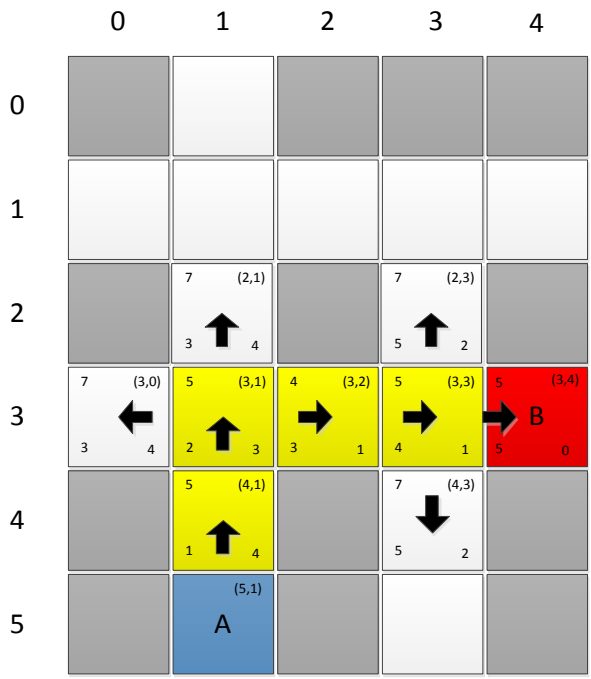
Gambar 3.7 Langkah Empat Perhitungan Algoritma A*

Langkah empat menghasilkan *node-node* di dalam *open list* dan *closed list* dijelaskan dengan Tabel 3.5.

Tabel 3.5 Perhitungan Langkah Empat

Open List	(3,3)
Closed list	(3,2)
Parent	(3,2)
Current node	(3,3)

Berikut ini perhitungan pada langkah lima yang berada pada *node* dengan koordinat (3,3) terdapat pada gambar 3.8.



Gambar 3.8 Langkah Lima Perhitungan Algoritma A*

Langkah lima menghasilkan *node-node* di dalam *open list* dan *closed list* dijelaskan dengan Tabel 3.6.

Tabel 3.6 Perhitungan Langkah Lima

Open List	(2,3),(4,3),(3,4)
Closed list	(3,3)
Parent	(3,3)
Current node	(3,4)

Penentuan jalan terbaik dengan melakukan pemeriksaan dimulai dari tujuan atau *node* B, dalam ilustrasi adalah kotak berwarna kuning. Dengan mengikuti arah *pointer* yang berarti akan membawanya kembali ke *node* awal, maka akan didapatkan jalan terbaiknya, Maka jika diimplementasikan pada *game*,

nantinya mobil akan menghindari halangan dan akhirnya akan menemukan rute terpendek ke tujuan.

Pseudo Code

	//Teruskan pencarian selama tidak terjadi eror dan node akhir belum ditemukan
1	while (StatusComplete == BelumComplete) {
	//Close atau tutup current node, jika current node sama dengan node tujuan maka pencarian jalur selesai
2	if (current.node == endNode) {
3	StatusComplete = Complete;
4	break;
5	}
	//apabila
6	if (current.node.h < nodetargetsementara.h) {
7	nodetargetsementara = currrent;
8	}
	//Ulangi pada semua node tetangga yang dapat dilewati dan masukan kedalam open list
9	current.node.Open (data,current,htarget);
10	//apakah terdapat node untuk dilakukan pencarian?
11	if (runData.open.numberOfItems <= 1) {
12	Error ();
13	LogError ("No open points, whole area searched");
14	return;
15	}
16	//pilih node dengan nilai F score terkecil dan hapus node tersebut dari open list

```
17  current = runData.open.Remove ();
18  }
```

Gambar 3.9 *Pseudo Code*

3.5 Analisis Kebutuhan *Non* Fungsional

Analisis kebutuhan *non*-fungsional dilakukan untuk mengetahui spesifikasi kebutuhan untuk sistem. Spesifikasi kebutuhan melibatkan analisis perangkat keras/hardware, analisis perangkat lunak/software.

3.6 Analisis Kebutuhan Perangkat Keras

Analisis perangkat keras digunakan untuk mengetahui spesifikasi perangkat keras yang dibutuhkan dalam merancang *game* yang dibangun dan spesifikasi bagi pengguna agar dapat menjalankan *game* yang dibangun**.** Berikut adalah spesifikasi perangkat keras yang dibutuhkan bagi pengembang dan pengguna :

1. Pengembang

Spesifikasi kebutuhan perangkat keras yang dibutuhkan untuk pengembangan *game* yang dibangun terdapat pada Tabel 3.7.

Tabel 3.7 Spesifikasi Perangkat Keras - Pengembang

Nama Perangkat	Spesifikasi
Prosesor	1.90 GH
RAM	4 GB
VGA	2 GB
Harddisk	500 GB
Monitor	Resolusi 1366 x 768
Keyboar & Mouse	Standar

Dari spesifikasi perangkat keras yang digunakan untuk pengembangan didapatkan hasil FPS *(Frame per second)* sebesar 60 dari *game* yang dibangun. Dengan FPS 60 *game* yang dibangun dapat dimainkan dengan baik dan lancar.

2. Pengguna

Spesifikasi kebutuhan perangkat keras yang dibutuhkan bagi pengguna ini agar dapat menjalankan aplikasi *game* yang dibangun terdapat pada Tabel 3.8.

Tabel 3.8 Spesifikasi Perangkat Keras - Pengguna

Nama Perangkat	Spesifikasi
Prosesor	1.80 GHz
RAM	1 GB
VGA	1 GB
Harddisk	1 GB
Monitor	Resolusi 1366 x 768
Keyboar & Mouse	Standar
Speaker	Optional

Spesifikasi perangkat keras yang ditentukan untuk pengguna merupakan spesifikasi minimum yang dapat digunakan agar pengguna dapat memainkan *game* yang dibangun dengan baik dan lancar. Semakin rendah spesifikasi perangkat keras yang digunakan oleh pengguna maka proses permainan akan tidak lancar karena beratnya proses grafik dan perhitungan yang dilakukan ketika permainan dijalankan tidak dapat terpenuhi secara maksimal dengan spesifikasi yang rendah.

3.7 Analisis Kebutuhan Perangkat Lunak

Analisis perangkat lunak digunakan untuk mengetahui spesifikasi perangkat lunak yang dibutuhkan untuk merancang *game* yang akan dibangun. Berikut adalah spesifikasi perangkat lunak yang dibutuhkan bagi pengembang dan pengguna.

1. Pengembang

Spesifikasi kebutuhan perangkat lunak yang dibutuhkan untuk pengembangan *game* yang dibangun terdapat pada Tabel 3.9.

Tabel 3.9 Analisisi Kebutuhan Perangkat Lunak

Nama Perangkat Lunak	Spesifikasi
Sistem Operasi	Microsoft Windows 7 Home
Tools	1. Unity 3D 4.1.5 2. iClone 3DXchange v5.5 Pipeline 3. Blender 2.68a 4. 3ds MAX 2014 5. SketchUp 2013

2. Pengguna

Spesifikasi kebutuhan perangkat lunak yang dibutuhkan bagi pengguna agar dapat menjalankan aplikasi *game* yang dibangun terdapat pada Tabel 3.10.

Tabel 3.10 Analisisi Kebutuhan Perangkat Lunak

Nama PerangkatLunak	Spesifikasi
Sistem Operasi	Microsoft Windows XP, Windows 7, dan Windows 8.

3.8 Analisis *User*

Ada 2 hal yang akan memudahkan pemain ketika memainkan *game* ini yaitu pengetahuan dan pengalaman, terutama pengetahuan dan pengalaman dalam memainkan *game* yang akan dibangun.

1. User K*no*wledge and Experience

Berikut ini klasifikasi *knowledge and experience* dari pengguna aplikasi terdapat pada Tabel 3.11.

Tabel 3.11 Knowledge and Experience

Education Level	Bisa digunakan oleh berbagai kalangan, seperti pelajar, mahasiswa hingga masyarakat awam.
Reading Level	Bisa digunakan oleh berbagai *level* pendidikan dengan reading *level* yang *moderate*.
Typing Skills	Tidak memerlukan *typing skills* yang tinggi.
Computer Literacy	*Moderate* (menengah)
Task Experience	Bisa digunakan oleh pengguna dengan pengalaman penggunaan komputer dan *game* yang *moderate*.
System Experience	Bisa digunakan oleh pengguna dengan pengalaman penggunaan komputer dan *game* yang *moderate*.
Application Experience	Bisa digunakan oleh pengguna yang biasa menggunakan sistem operasi Windows XP, Windows 7, dan Windows 8.

2. User Physical Characteristic

Keadaan fisik seseorang mungkin akan berpengaruh pada penggunaan aplikasi *game* ini. Ada hal-hal yang harus diperhatikan juga terhadap user dari karakteristik fisiknya untuk dapat menggunakan aplikasi ini yaitu, *Color Blind, Handednes, dan Gender.* Berikut ini klasifikasi *Users Physical Characteristic* dari pengguna aplikasi terdapat pada Tabel 3.12.

Tabel 3.12 Users Physical Characteristic

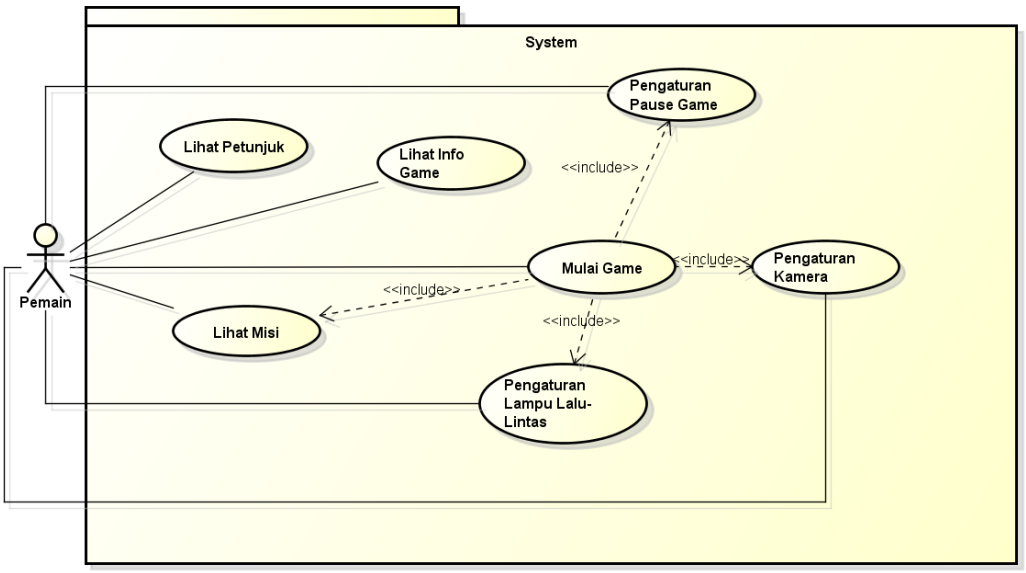
Age	11 tahun ke atas
Gender	Pria dan Wanita
Handedness	Kanan dan kiri

3.9 Analisis Kebutuhan Fungsional

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

3.9.1 *Use case Diagram*

Use case merupakan permodelan untuk perilaku *(behavior)* sebuah sistem yang akan dibangun. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem yang akan dibangun. Secara umum, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut [2]. Interaksi antara aktor dengan aplikasi *game* yang akan dibangun digambarkan pada *diagram use case* pada Gambar 3.10.



Gambar 3.10 Use case *Diagram Game* yang akan dibangun

3.9.2 Definisi Aktor

Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem sistem yang akan dibangun [2]. Definisi aktor diterangkan pada Tabel 3.13.

Tabel 3.13 Definisi Aktor

No	Aktor	Deskripsi
1.	Pemain	Pemain atau orang yang memainkan *game*

3.9.3 Definisi *Use case*

Use case merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor [2]. Definisi *use case* diterangkan pada Tabel 3.14.

Tabel 3.14 Definisi Use case

No.	*Use case*	Deskripsi
1.	Mulai *Game*	Proses untuk memulai *game* baru
2.	Lihat Petunjuk	Proses untuk melihat petunjuk atau panduan cara bermain
3.	Lihat *Info Game*	Proses untuk melihat informasi pembuat *game*
4.	Lihat Misi	Proses untuk melihat misi dalam *game*
5.	Pengaturan *Pause Game*	Proses untuk melakukan *Pause* atau *Unpause* dalam *game*
6.	Pengaturan Kamera	Proses untuk melakukan pindah kamera pada setiap persimpangan jalan dalam *game*
7.	Pengaturan Lampu Lalu lintas	Proses untuk mengatur *object* lampu lalu lintas dalam *game*

3.9.4 Skenario *Use case*

Setiap *use case* dilengkapi dengan skenario. Skenario *use case* adalah alur jalannya proses *use case* dari sisi aktor dan sistem [2]. Berikut adalah skenario jalannya masing-masing *use case* yang telah didefinisikan sebelumnya :

Nama *Use case* : Mulai *Game*

Skenario :

Tabel 3.15 Scenario Use case Mulai *Game*

Aksi Aktor	Reaksi Sistem
Skenario *No*rmal	
1. Mulai aplikasi	
	2. Menampilkan *Menu* utama
3. Pilih *Menu* play	
	4. Menampilkan layar *game*
	5. Mengatur misi pada level
	6. Mengatur timer pada level
	7. Mengatur spawnrate pada level
	8. Menghitung timer pada level hingga selesai
	9. Check misi
	10. Menampilkan pesan gagal dan ulangi level apabila misi gagal terpenuhi
11. Pilih *Menu* ulangi	
	12. Mengulang level dan ulangi proses pengaturan
	13. Menampilkan pesan sukses dan lanjut level apabila misi terpenuhi
14. Pilih *Menu* lanjut	
	15. Update level dan ulangi proses pengaturan

Nama *Use case* : Lihat Petunjuk

Skenario :

Tabel 3.16 Scenario Use case Lihat Petunjuk

Aksi Aktor	Reaksi Sistem
Skenario *N*ormal	
	1. Menampilkan *Menu* utama
2. Pilih *Menu* lihat petunjuk	
	3. Menampilkan layar petunjuk permainan

Nama *Use case* : Lihat Info *Game*

Skenario :

Tabel 3.17 Scenario Use case Lihat Info *Game*

Aksi Aktor	Reaksi Sistem
Skenario *N*ormal	
	1. Menampilkan *Menu* utama
2. Pilih *Menu* lihat info *game*	
	3. Menampilkan layar info *game*

Nama *Use case* : Lihat Misi

Skenario :

Tabel 3.18 Scenario Use case Lihat Misi

Aksi Aktor	Reaksi Sistem
Skenario *N*ormal	
	1. Menampilkan layar *game*
	2. Mengatur misi pada level
3. Pilih *Menu* lihat misi	
	4. Menampilkan informasi misi pada level
5. Pilih *Menu* tutup layar misi	

	6. *Menu*tup layar informasi misi

Nama *Use case* : Pengaturan *Pause Game*

Skenario :

Tabel 3.19 Scenario Use case Pengaturan Pause *Game*

Aksi Aktor	Reaksi Sistem
Skenario *N*ormal	
	1. Menampilkan layar *game*
2. Pilih *Menu* pause *game*	
	3. Menampilkan pesan pause / keluar
4. Pilih *Menu* resume *game*	
	5. Menampilkan layar *game* kembali
6. Pilih *Menu* keluar *game*	
	7. Kembali ke *Menu* utama

Nama *Use case* : Pengaturan Kamera

Skenario :

Tabel 3.20 Scenario Use case Pengaturan Kamera

Aksi Aktor	Reaksi Sistem
Skenario *N*ormal	
	1. Menampilkan layar *game*
	2. Menampilkan current camera
3. Tekan tombol keyboard ‘A’	
	4. Menampilkan kamera selanjutnya
5. Tekan tombol keyboard ‘D’	
	6. Menampilkan kamera sebelumnya

Nama *Use case* : Pengaturan Lampu Lalu lintas

Skenario :

Tabel 3.21 Scenario Use case Pengaturan Lampu Lalu lintas

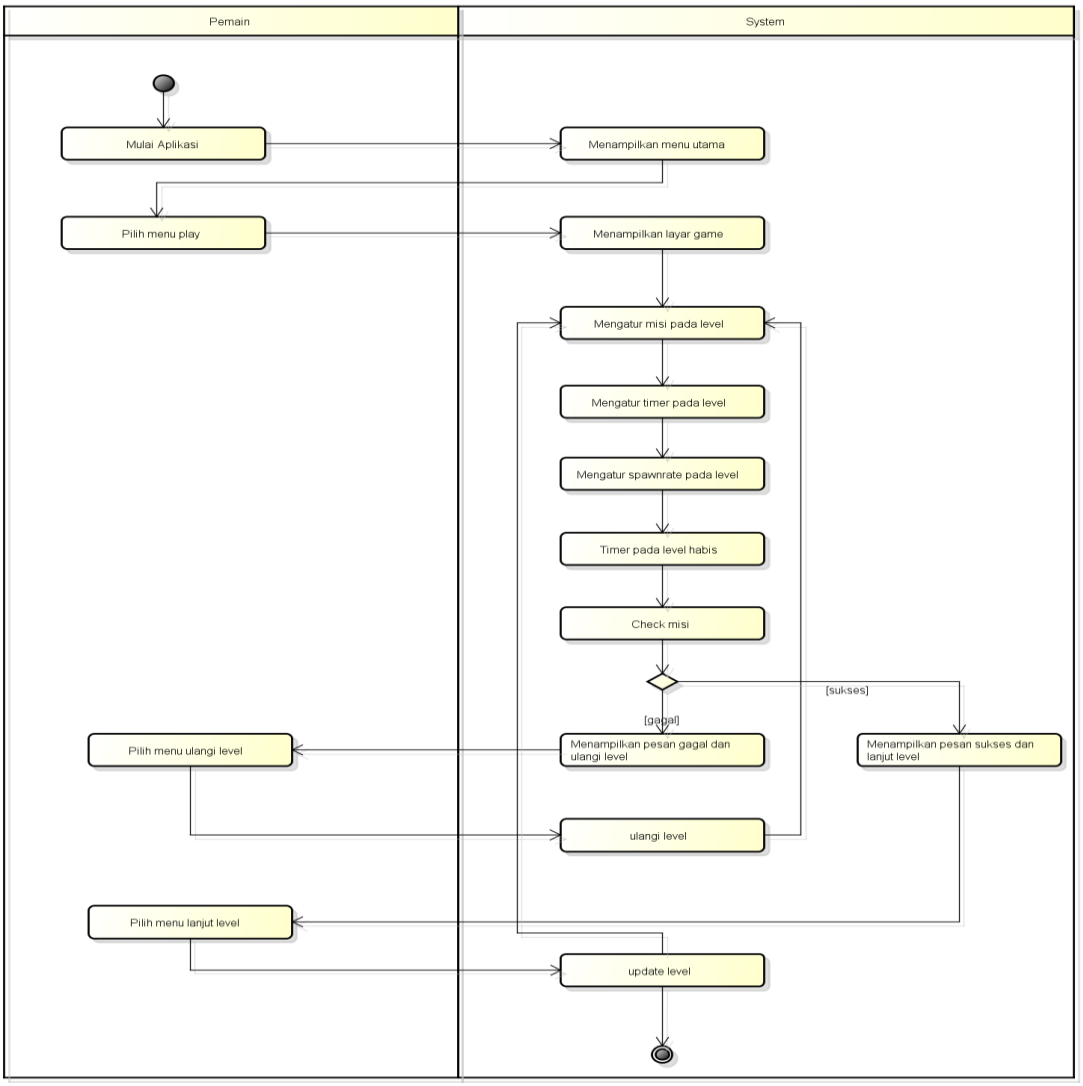
Aksi Aktor	Reaksi Sistem
Skenario *N*ormal	
	1. Menampilkan layar *game*
	2. Menampilkan lampu lalu lintas ke 1 status kuning
3. Pilih lampu lalu lintas ke 1-4	
	4. Menampilkan lampu lalu lintas yang dipilih dengan status kuning

3.9.5 *Activity Diagram*

Activity diagram menggambarkan workflow (aliran kerja) atau aktivitas dari sebuah sistem yang ada pada perangkat lunak [2]. Berikut ini adalah beberapa *activity diagram* yang terdapat pada *game* yang akan dibangun :

1. *Activity Diagram* Mulai *Game*

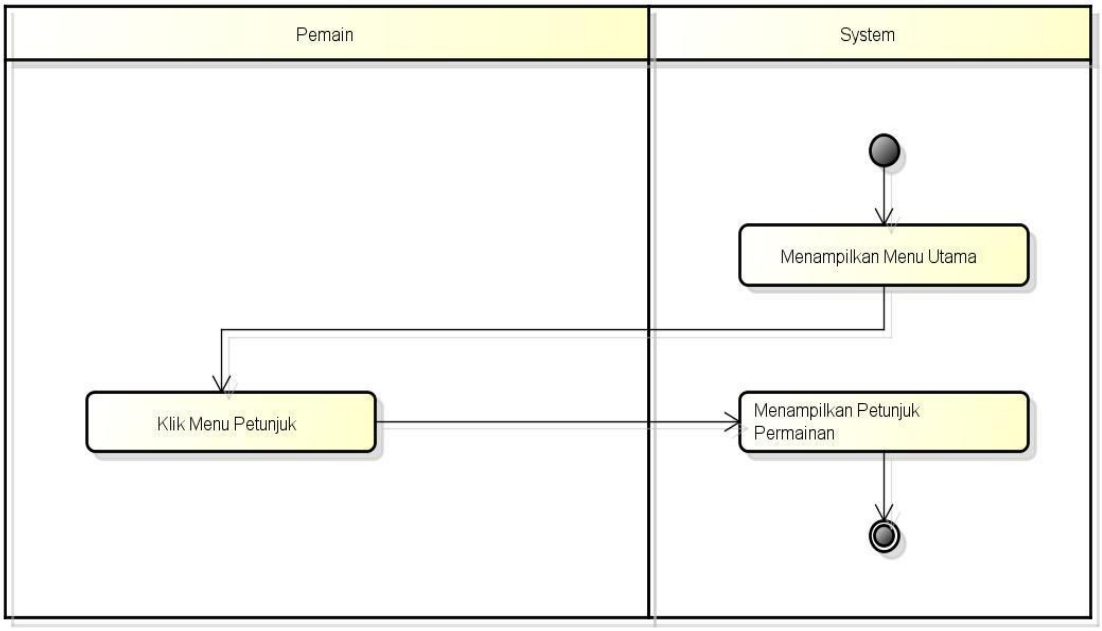
Proses ini terjadi saat pemain memilih *Menu* play pada *Menu* utama. Selanjutnya, sistem akan masuk ke *scene game*. Berikut adalah *activity diagram* mulai *game* yang terdapat pada gambar 3.10.



Gambar 3.11 *Activity Diagram* Mulai *Game*

2. *Activity Diagram* Lihat Petunjuk

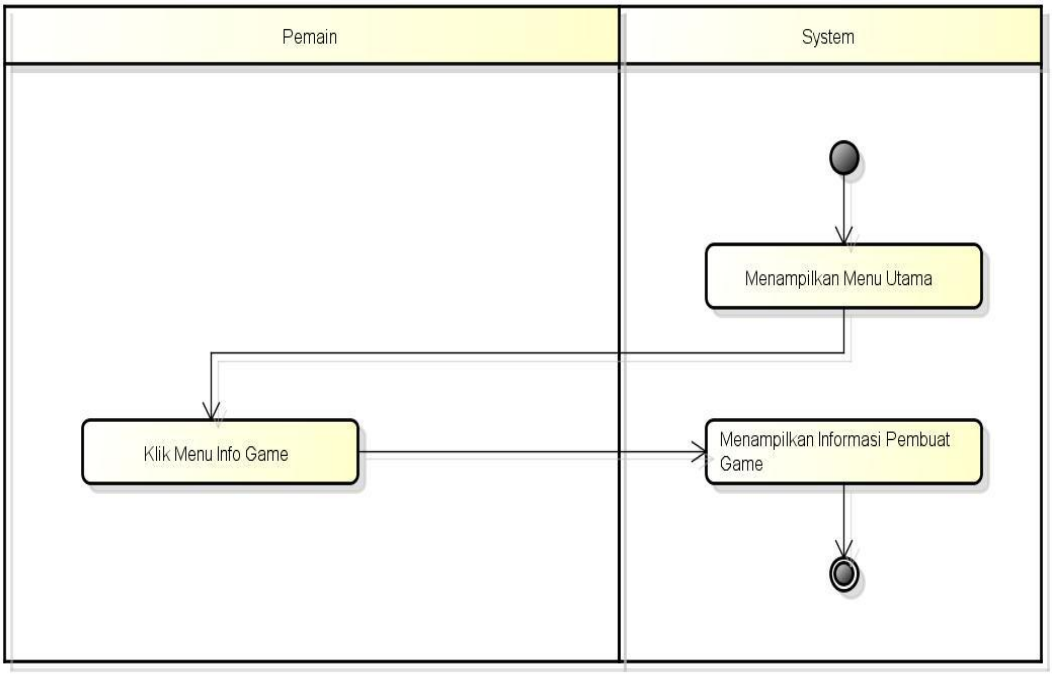
Proses ini terjadi saat pemain memilih *Menu* petunjuk pada *Menu* utama. Selanjutnya, sistem akan menampilkan informasi petunjuk permainan. Berikut adalah *activity diagram* lihat petunjuk yang terdapat pada gambar 3.11.



Gambar 3.12 *Activity Diagram* Lihat Petunjuk

3. *Activity Diagram* Lihat Info *Game*

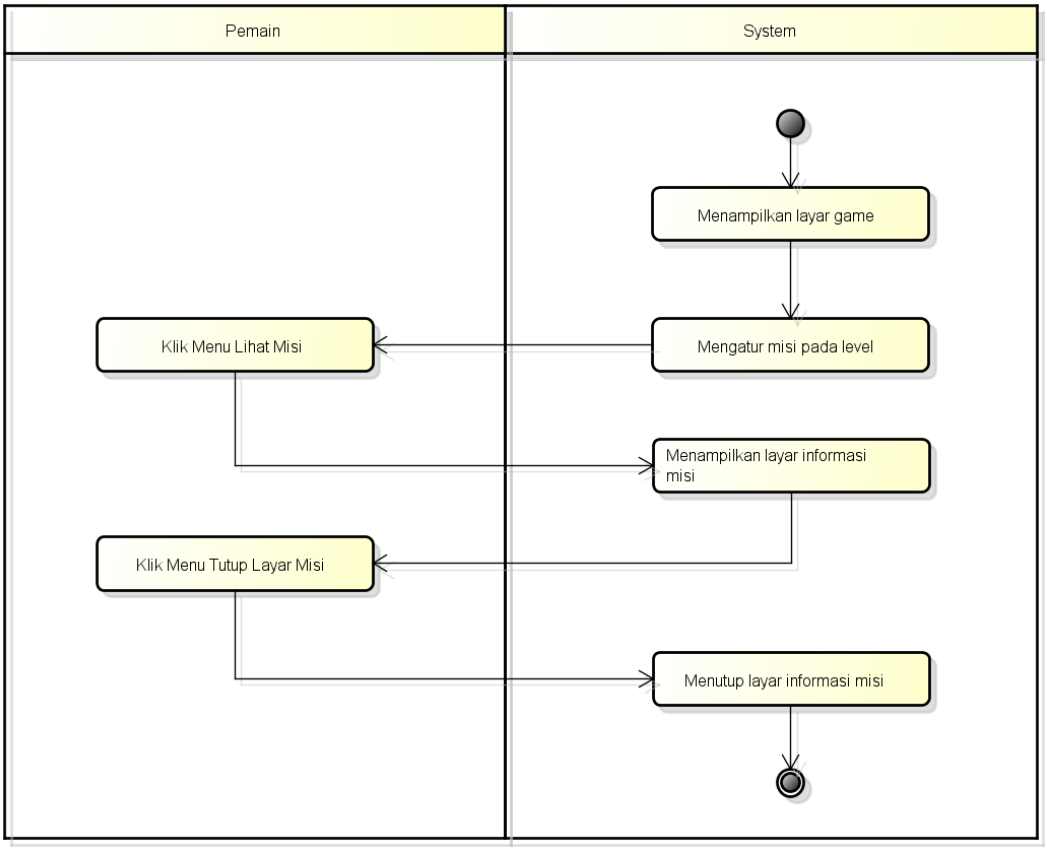
Proses ini terjadi saat pemain memilih *Menu* info *game* pada *Menu* utama. Selanjutnya, sistem akan menampilkan informasi pembuat *game*. Berikut adalah *activity diagram* lihat info *game* yang terdapat pada gambar 3.12.



Gambar 3.13 *Activity Diagram* Proses Lihat Info *Game*

4. *Activity Diagram* Lihat Misi

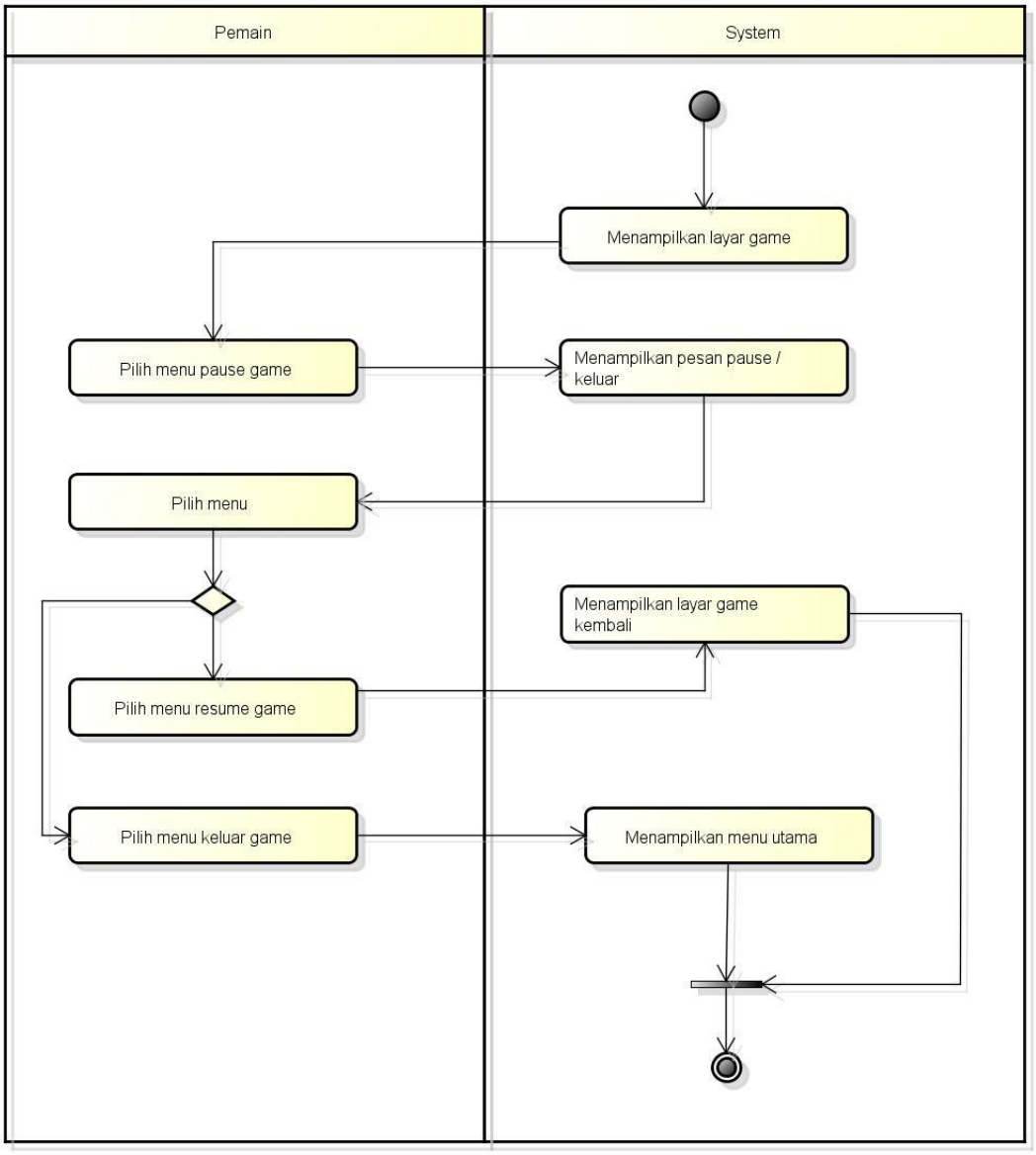
Proses ini terjadi setelah sistem menampilkan *scene game* dan pemain memilih *Menu* lihat misi. Selanjutnya, sistem akan menampilkan informasi misi. Berikut adalah *activity diagram* lihat lihat isi yang terdapat pada gambar 3.14.



Gambar 3.14 *Activity Diagram* Lihat Misi

5. *Activity Diagram* Pengaturan Pause *Game*

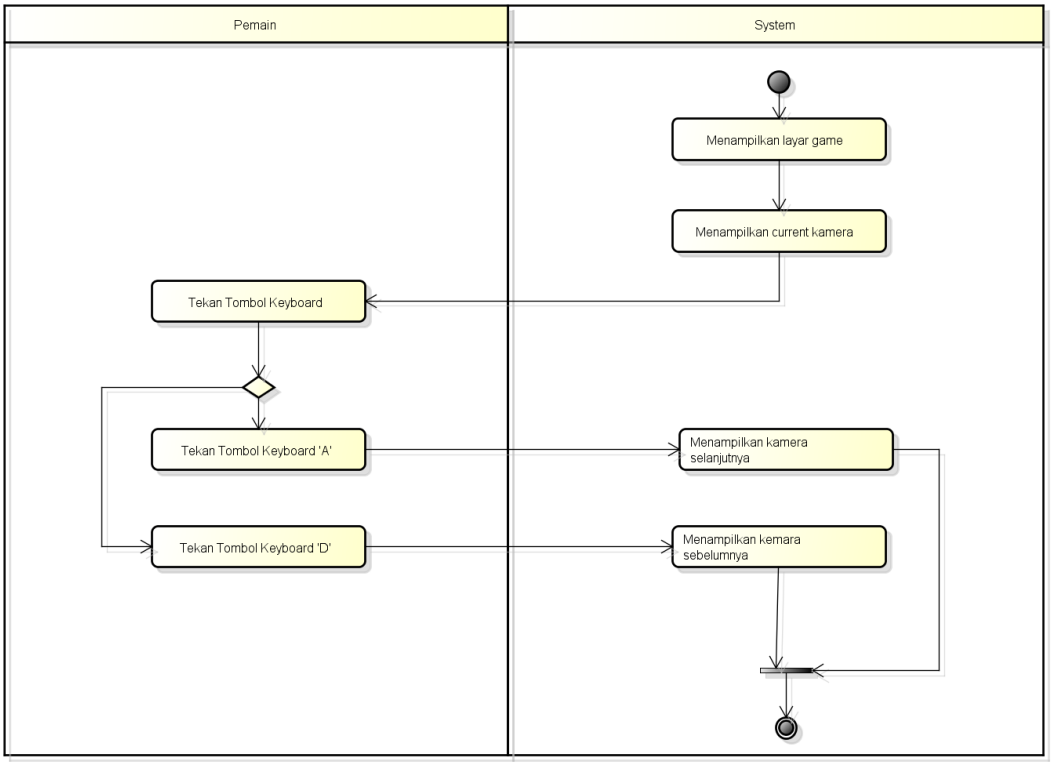
Proses ini terjadi setelah sistem menampilkan *scene game* dan pemain memilih *Menu Pause game*. Sistem akan melakukan *Pause game* dan menampilkan *no*tifikasi *Pause game*. Berikut adalah *activity diagram* pengaturan *Pause game* yang terdapat pada gambar 3.15.



Gambar 3.15 *Activity Diagram* Pengaturan Pause *Game*

6. *Activity Diagram* Pengaturan Kamera

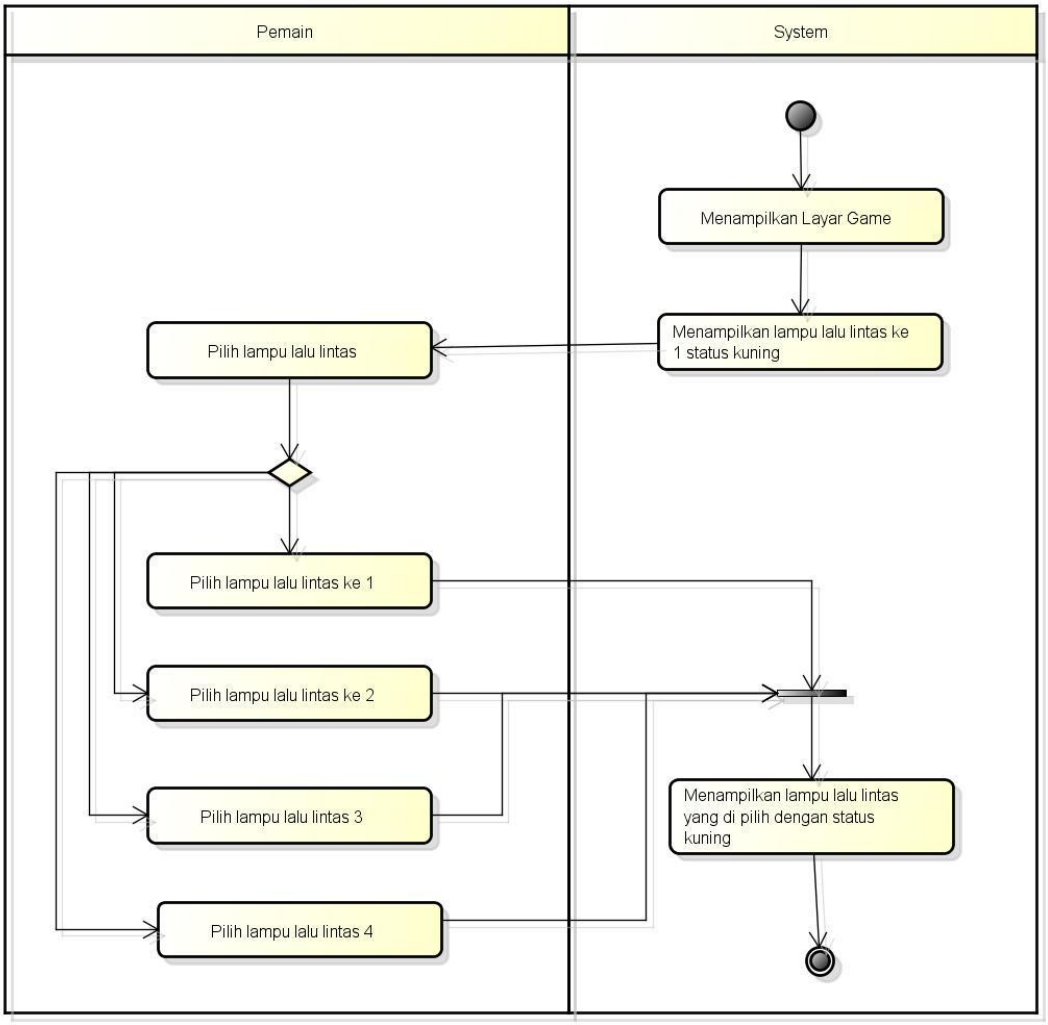
Proses ini terjadi setelah sistem menampilkan *scene game* dan pemain menekan tombol keyboard ‘a’ atau ‘d’. Sistem akan melakukan pindah focus kamera.Berikut adalah *activity diagram* pengaturan kamera yang terdapat pada gambar 3.18.



Gambar 3.16 *Activity Diagram* Pengaturan Kamera

7. *Activity Diagram* Pengaturan Lampu Lalu Lintas

Proses ini terjadi setelah sistem menampilkan *scene game* dan melakukan klik pada *game object* lampu lalu lintas. Sistem akan merubah status lampu lalu lintas menjadi status (kuning). Berikut adalah *activity diagram* pengaturan lampu lalu lintas yang terdapat pada gambar 3.19.

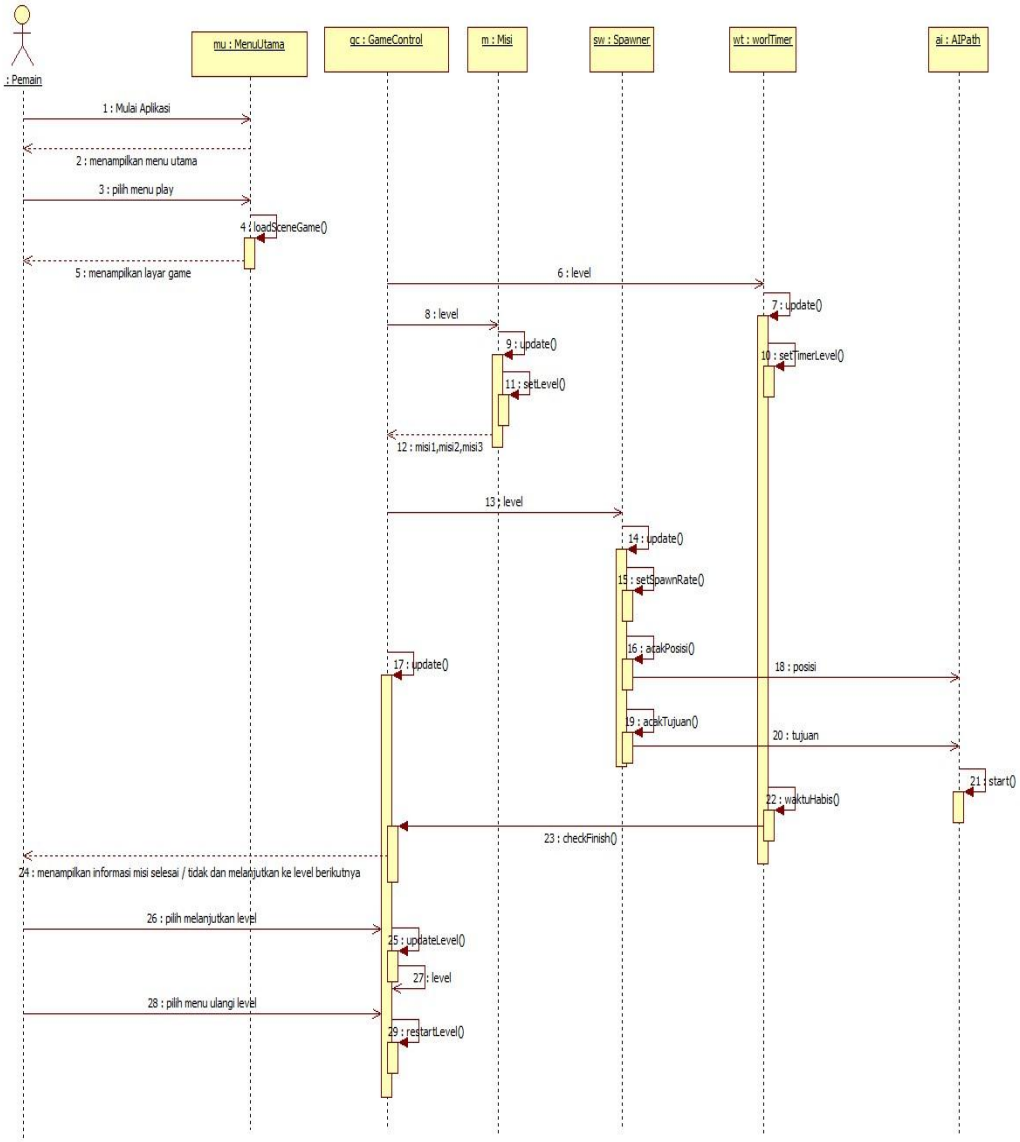


Gambar 3.17 *Activity Diagram* Pengaturan Lampu Lalu lintas

3.9.6 *Sequence Diagram*

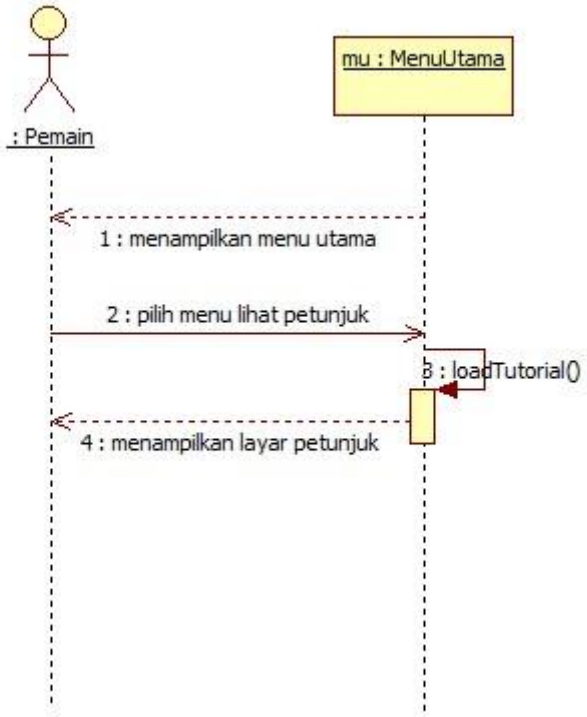
Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup dan *message* yang dikirimkan dan diterima antar objek [2]. Berikut adalah beberapa *diagram sequence* dari *game* yang akan dibangun :

1. *Sequence Diagram* Mulai *Game*



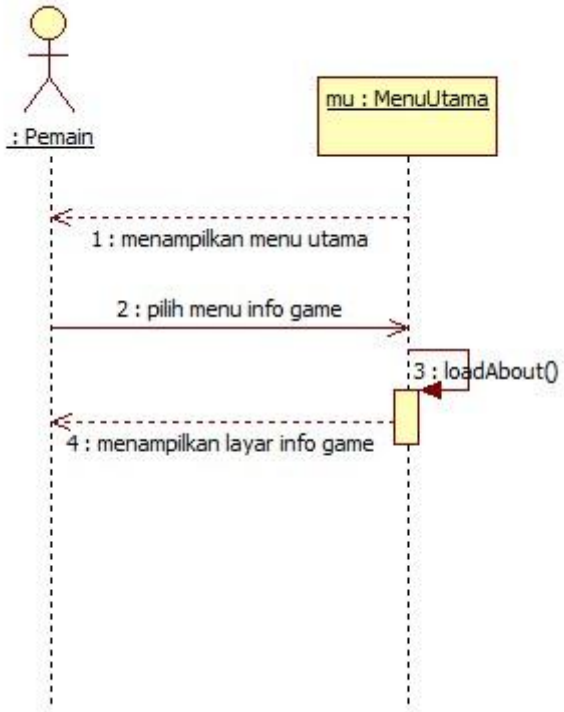
Gambar 3.18 *Sequence Diagram* Mulai *Game*

2. *Sequence Diagram* Lihat Petunjuk



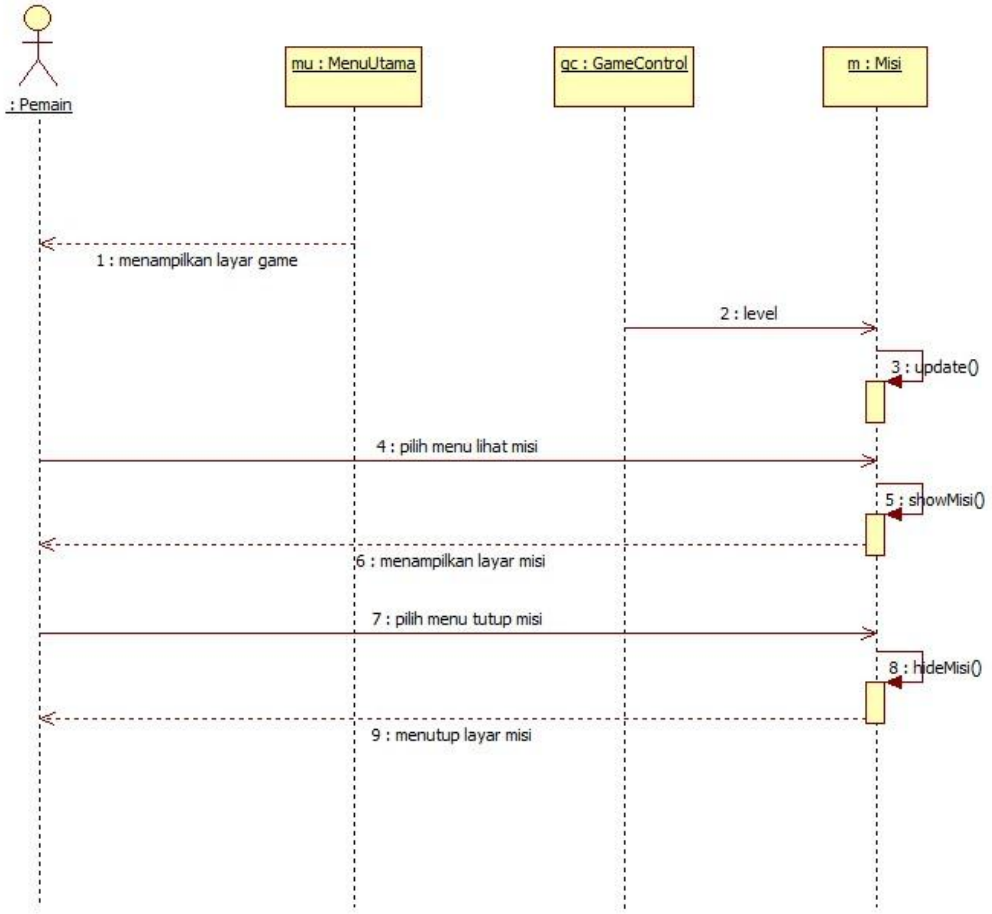
Gambar 3.19 *Sequence Diagram* Lihat Petunjuk

3. *Sequence Diagram* Lihat Info *Game*



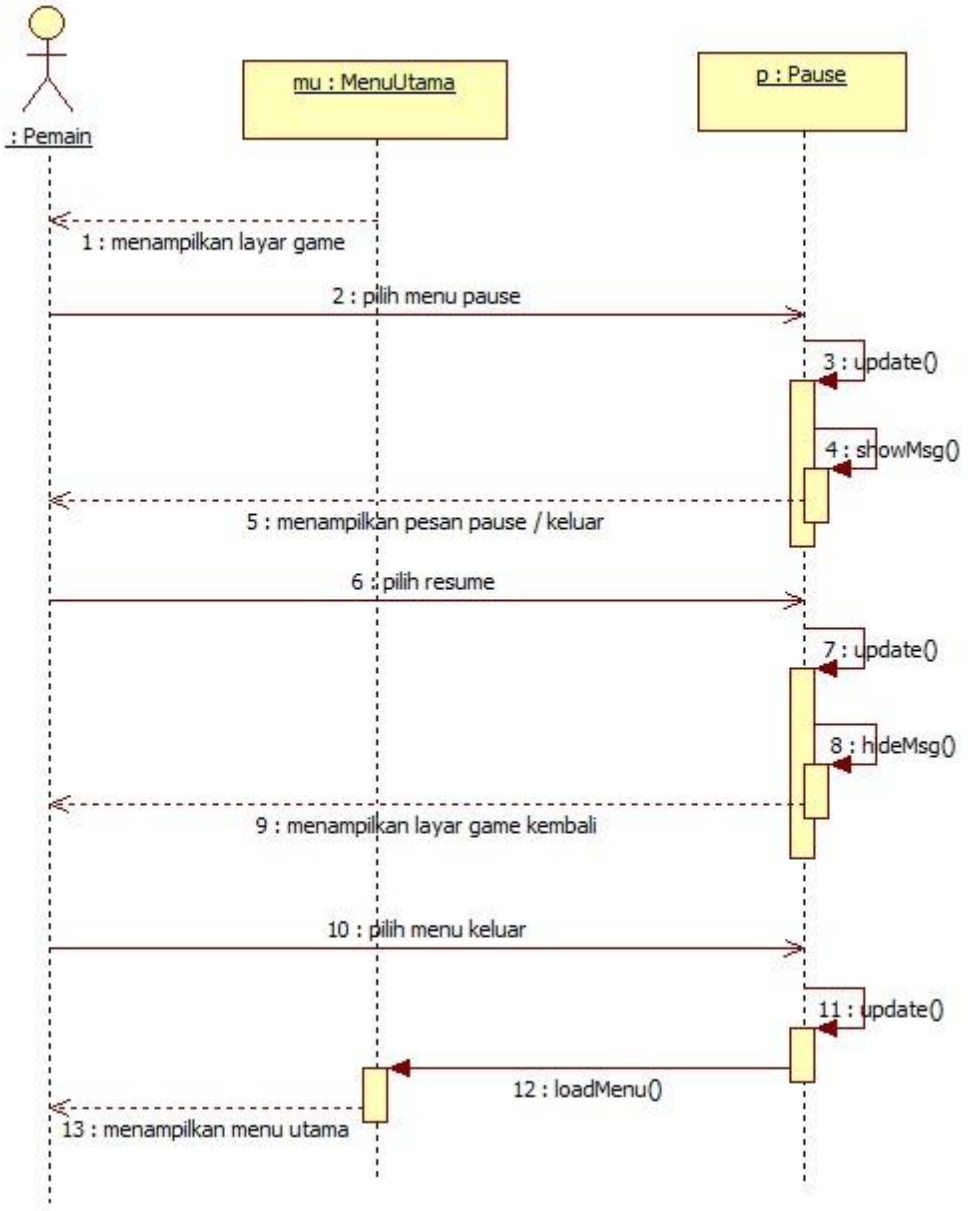
Gambar 3.20 *Sequence Diagram* Lihat Info *Game*

4. *Sequence Diagram* Lihat Misi



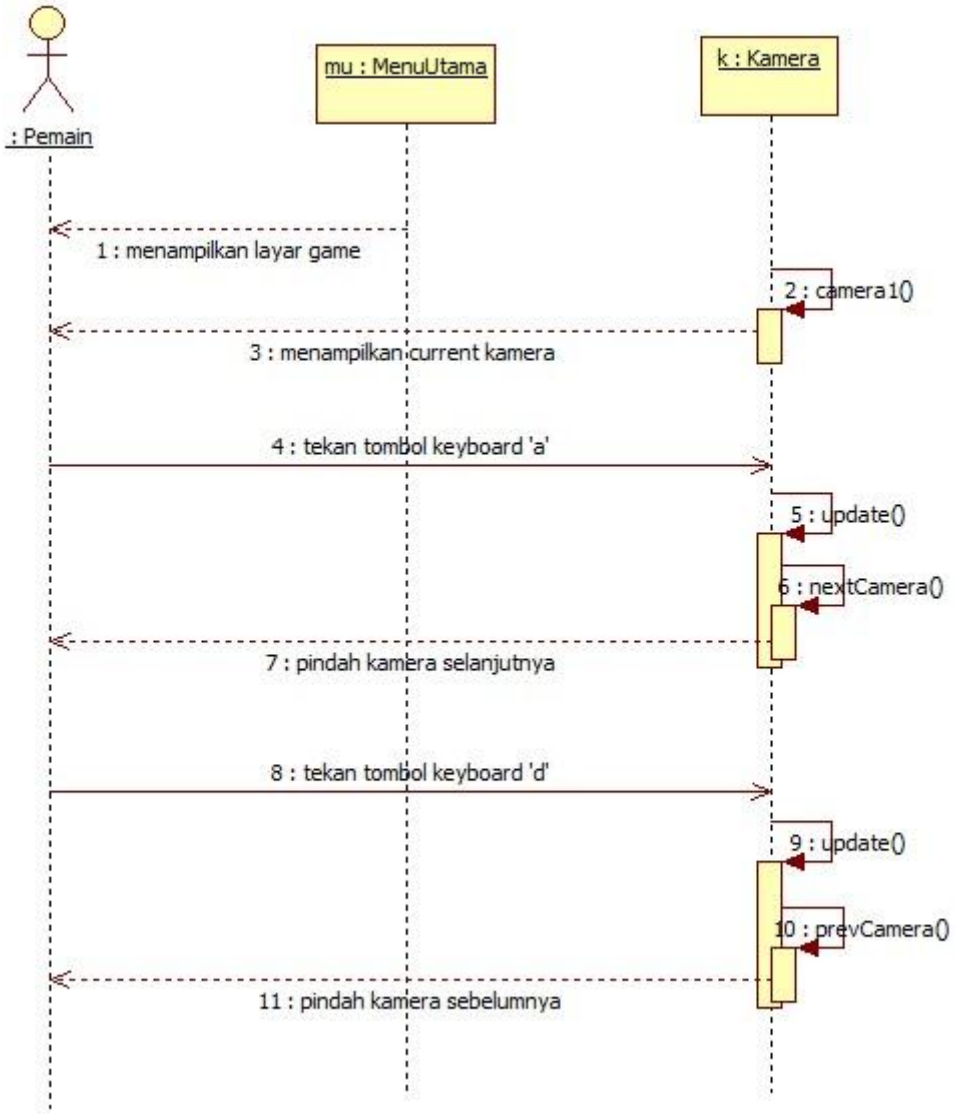
Gambar 3.21 *Sequence Diagram* Lihat Misi

5. *Sequence Diagram* Pengaturan *Pause Game*



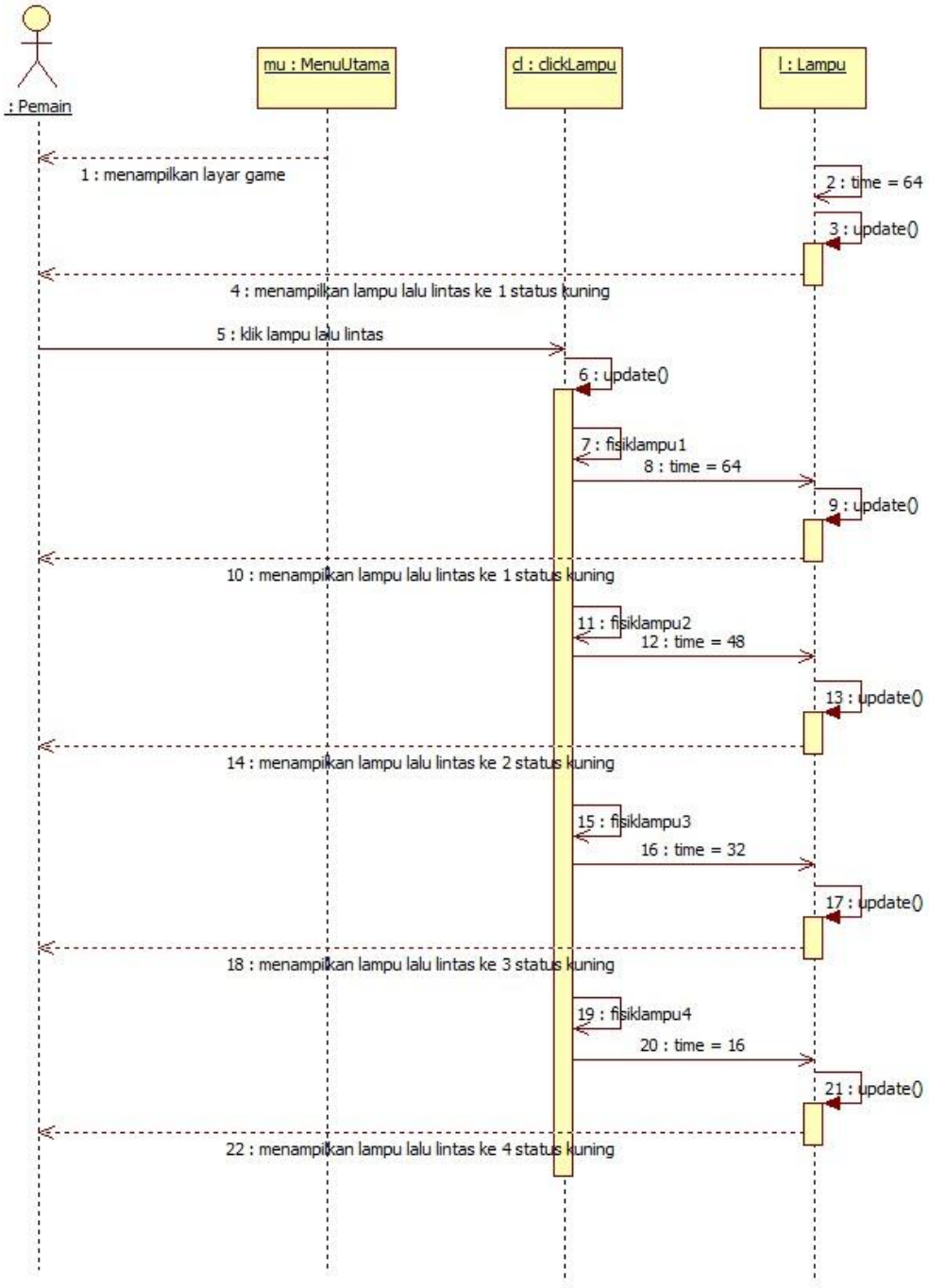
Gambar 3.22 *Sequence Diagram* Pause *Game*

6. *Sequence Diagram* Pengaturan Kamera



Gambar 3.23 *Sequence Diagram* Pengaturan Kamera

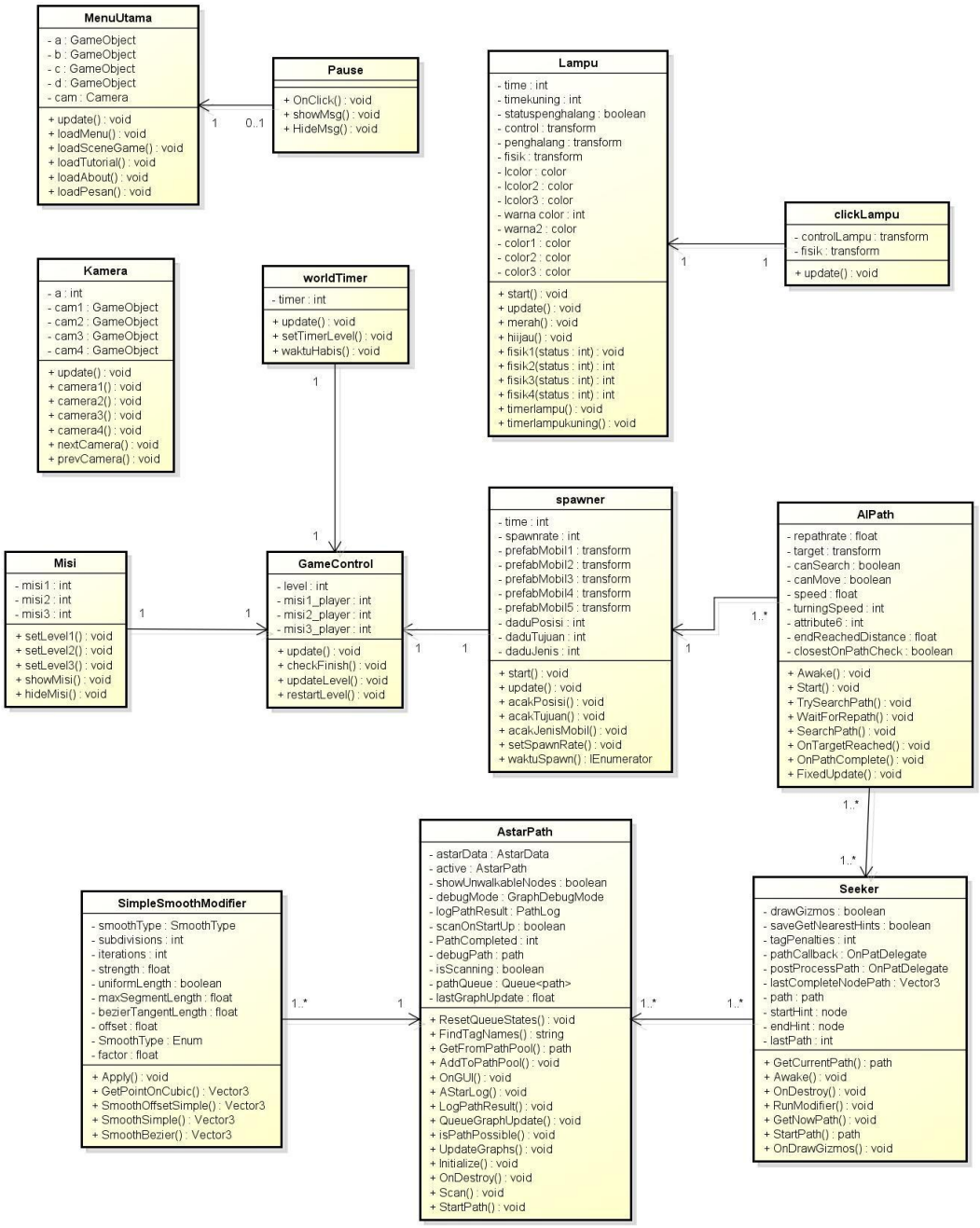
7. *Sequence Diagram* Pengaturan Lampu Lalu lintas



Gambar 3.24 *Sequence Diagram* Pengaturan Lampu Lalu lintas

3.9.7 *Class Diagram*

Class diagram digunakan untuk menampilkan kelas-kelas dan paket-paket di dalam sistem. *Class diagram* memberikan gambaran sistem secara statis dan relasi antar mereka. Beberapa *diagram* akan menampilkan subset dari kelas-kelas dan relasinya. Berikut adalah *class diagram* dari *game* yang akan dibangun pada Gambar 3.25.



Gambar 3.25 *Class Diagram*

3.10 Perancangan Sistem

Di dalam bagian ini akan dijelaskan perancangan komponen permainan, perancangan antarmuka, perancangan pesan dan jaringan semantik dari *game* yang akan dibangun.

3.10.1 Perancangan Komponen Permainan

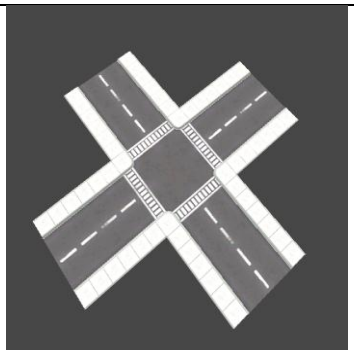
Di dalam bagian ini akan dijelaskan karakter atau entitas yang terdapat pada *game* dan *storyboard* dari *game* yang akan dibangun.

3.10.2 Perancangan Karakter

Di dalam bagian ini akan dijelaskan karakter atau entitas yang terdapat pada *game* yang akan dibangun seperti yang bisa dilihat pada Tabel 3.22.

Tabel 3.22 Penjelasan Proses Pengaturan

No	**Nama**	**Gambar**	**Keterangan**
1.	Mobil Hummer		Mobil pengguna jalan umum yang muncul dalam kota
2.	Mobil Hummer HX		Mobil pengguna jalan umum yang muncul dalam kota
3.	Mobil Biru		Mobil pengguna jalan umum yang muncul dalam kota

4.	Mobil Ambulance		Mobil ambulance yang muncul pada saat keadaan emergency
5.	Mobil Pemadam Kebakaran		Mobil pemadam kebakaran yang muncul saat keadaan emergency
6.	Lampu Lalu lintas		Lampu Lalu lintas yang dapat dikendalikan oleh pengguna dalam *game*
7.	Persimpangan Jalan		Persimpangan jalan tempat dimana terjadinya proses pengendalian lampu lalu lintas

3.10.3 *Storyboard*

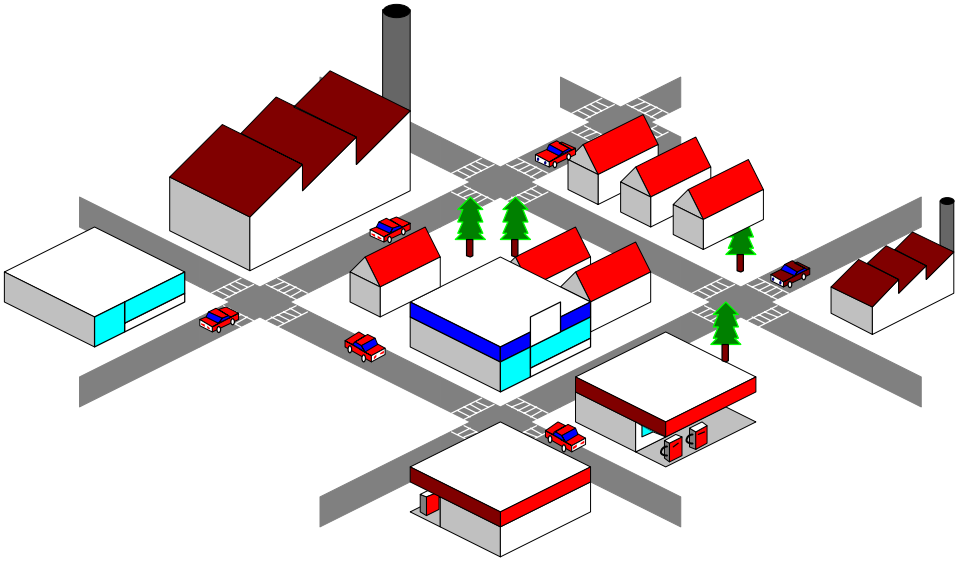
Storyboard merupakan sketsa gambar yang disusun berurutan sesuai dengan naskah yang ada, dengan *storyboard* dapat disampaikan ide cerita kepada orang lain dengan mudah.

Pada game yang akan dibangun pemain akan ditugaskan untuk mengendalikan 4 lampu lalu lintas yang terdapat pada 4 persimpangan dalam

kota. Pada *game* yang akan dibangun juga terdapat 3 tahapan waktu yang harus dilewati oleh pemain. Berikut penjelasan tentang tantangan pada tiap tahapan waktu :

1. Waktu 00.00 - 05.00. Pada tahapan ini kecepatan waktu jeda kemunculan mobil adalah 4 detik untuk 1 mobil yang dimunculkan secara acak pada tiap persimpangan jalan, dan akan terdapat 5 keadaan emergensi yang muncul secara acak selama waktu berjalan. Selain itu pemain juga harus dapat memenuhi tantangan score sebanyak 1000.
2. Waktu 05.00 – 12.00. Pada tahapan ini kecepatan waktu jeda kemunculan mobil adalah 3 detik untuk 1 mobil yang dimunculkan secara acak pada tiap persimpangan jalan, dan akan terdapat 10 keadaan emergensi yang muncul secara acak selama waktu berjalan. Selain itu pemain juga harus dapat memenuhi tantangan score sebanyak 1800.
3. Waktu 12.00 – 18.00. Tahapan ini merupakan tahapan tersulit dalam *game* ini karena kecepatan waktu jeda kemunculan mobil adalah 2 detik untuk 1 mobil yang dimunculkan secara acak pada tiap persimpangan jalan, dan akan terdapat 15 keadaan emergensi yang muncul secara acak selama waktu berjalan. Selain itu pemain juga harus dapat memenuhi tantangan score yang telah ditentukan sebanyak 2500.

Berikut gambaran dari *storyboard* dari *game* yang akan dibangun, terdapat pada Gambar 3.26.

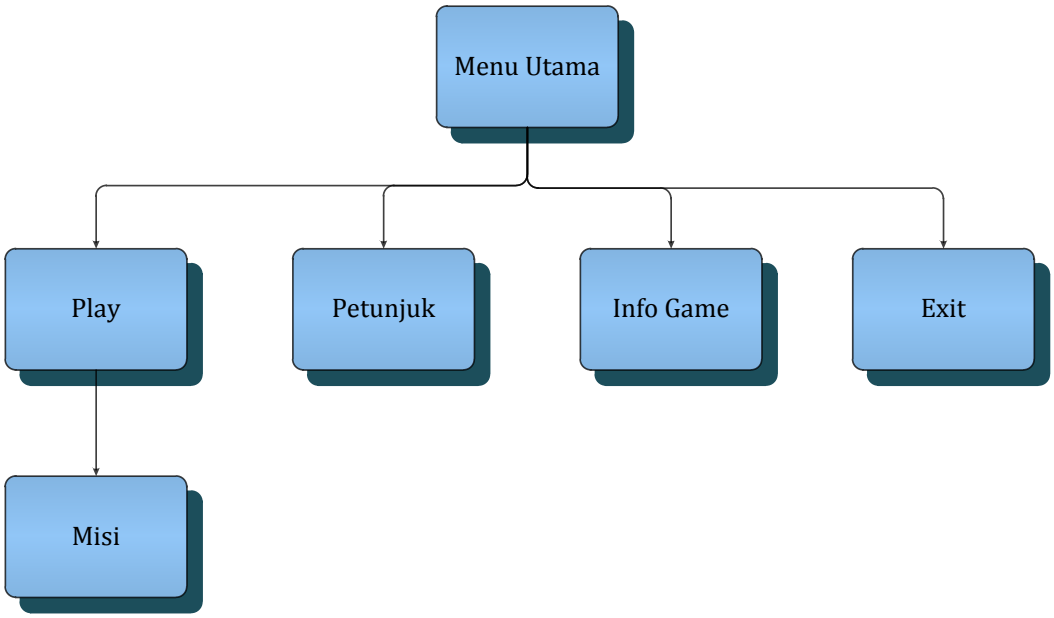


Gambar 3.26 *Storyboard Game*

3.11 Perancangan Struktur *Menu*

Untuk mempermudah dalam 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 nantinya akan diakses terlebih dahulu setelah program selesai dibuat.

Perancangan arsitektur *Menu* pada *game* yang akan dibangun dapat dilihat pada Gambar 3.27.



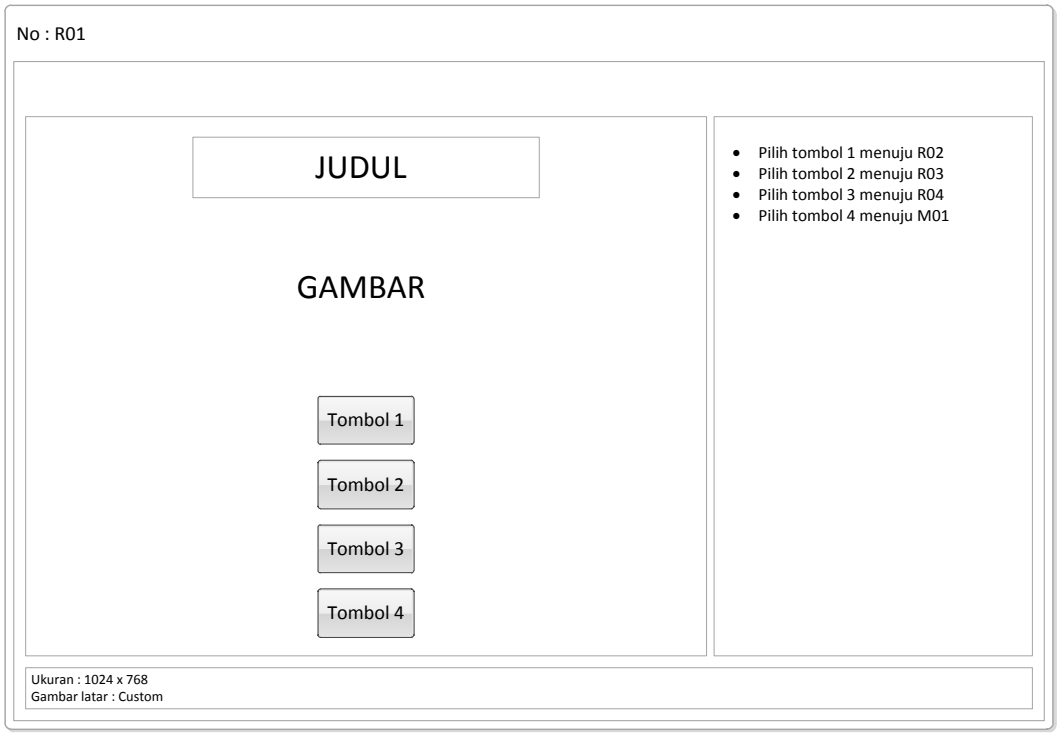
Gambar 3.27 Perancangan Struktur *Menu*

3.12 Perancangan Antar Muka

Perancangan antarmuka merupakan salah satu bagian penting dalam perancangan sishtem karena nantinya antarmuka tersebut akan menjadi fasilitas yang menjembatani interaksi manusia dengan sistem. Perancangan antarmuka pada *game* yang akan dibangun ini digambarkan pada Gambar 3.28 sampai denganGambar 3.32.

1. Perancangan Antarmuka *Menu* Utama

Perancangan antarmuka *Menu* utama pada *game* akan yang dibangun dapat dilihat pada Gambar 3.28.



Gambar 3.28 Perancangan Antarmuka *Menu* Utama

2. Perancangan Antarmuka Play

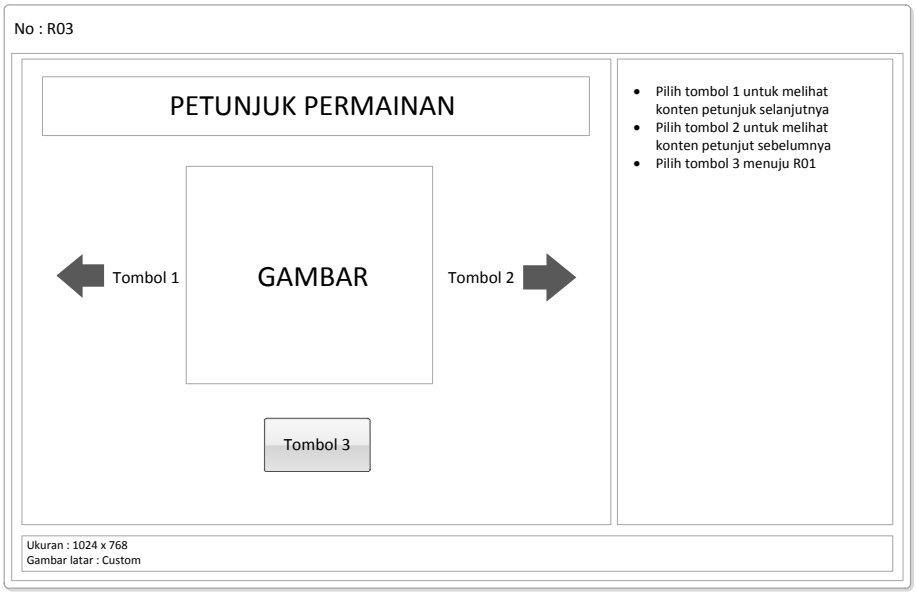
Perancangan antarmuka play pada *game* yang akan dibangun dapat dilihat pada Gambar 3.29.



Gambar 3.29 Perancangan Antarmuka Play

3. Perancangan Antarmuka *Menu* Petunjuk *Game*

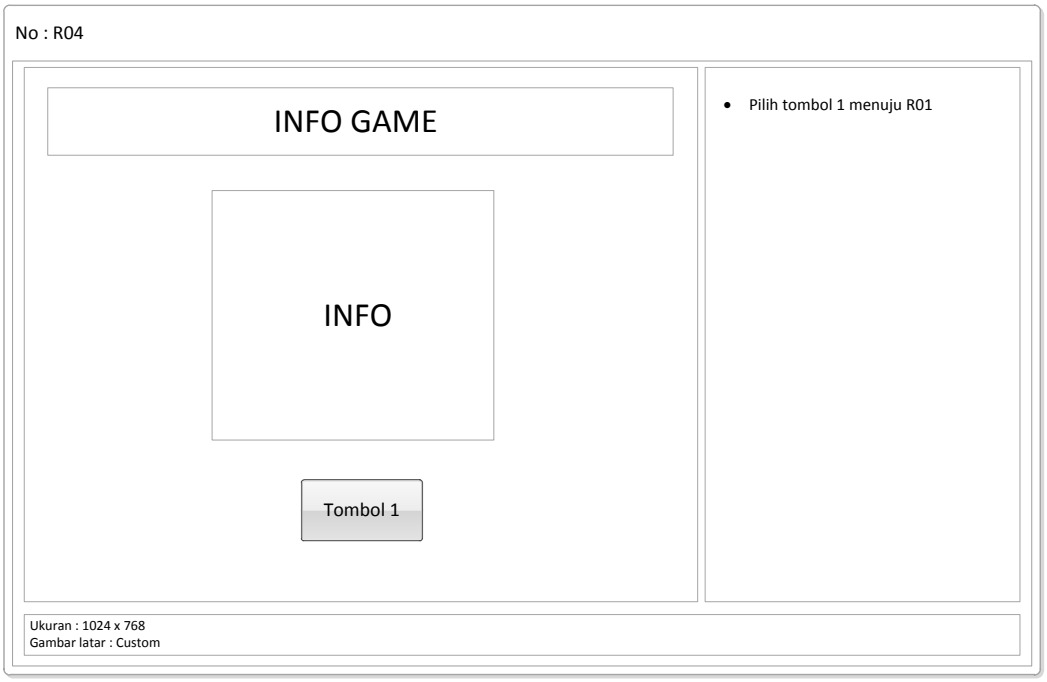
Perancangan antarmuka petunjuk pada *game* yang dibangun dapat dilihat pada Gambar 3.30.



Gambar 3.30 Perancangan Antarmuka Petunjuk

4. Perancangan Antar Muka *Menu* Info *Game*

Perancangan antarmuka petunjuk pada *game* yang dibangun dapat dilihat pada Gambar 3.31.



Gambar 3.31 Perancangan Antarmuka Info *Game*

5. Perancangan Antar Muka *Menu* Misi

Perancangan antarmuka *Menu* misi yang dibangun dapat dilihat pada Gambar 3.32.

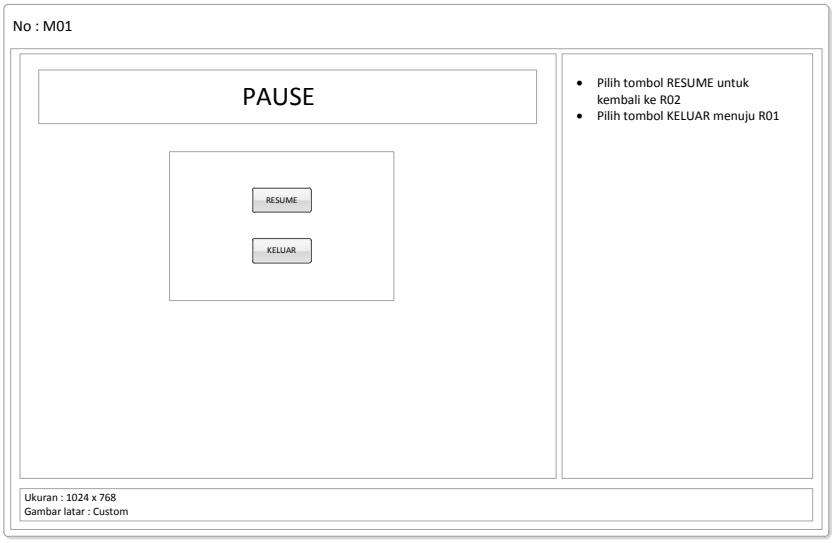


Gambar 3.32 Perancangan Antarmuka Info *Game*

3.13 Perancangan Pesan

Pada bagian ini akan dijelaskan perancangan pesan yang terdapat pada *game* yang akan dibangun yaitu perancangan pesan *Pause*, dan perancangan pesan keluar *game*.

1. Perancangan Pesan *Pause*



Gambar 3.33 Perancangan Pesan Pause

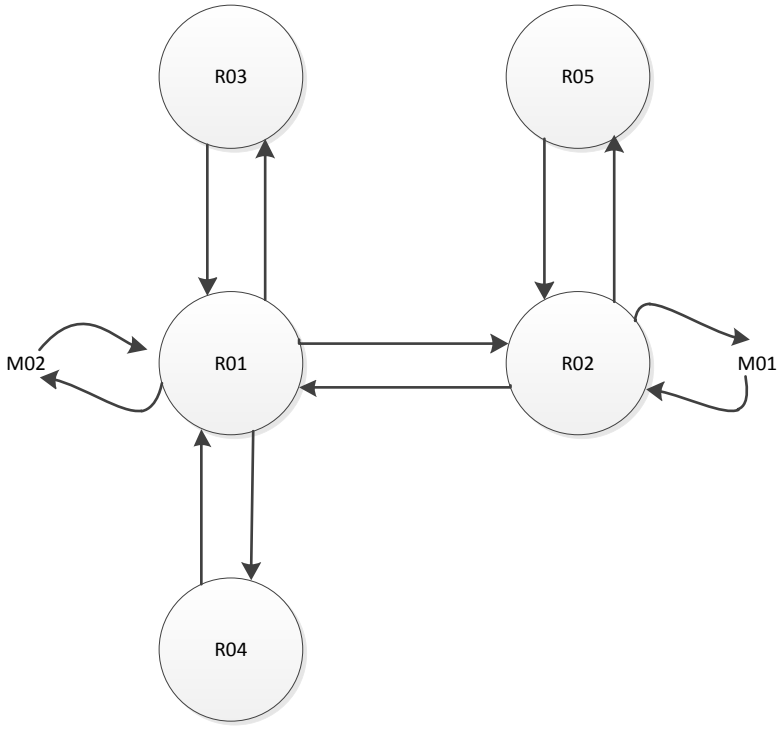
2. Perancangan Pesan Keluar *Game*



Gambar 3.34 Perancangan Pesan Keluar

3.14 Jaringan Semantik

Jaringan semantik adalah menggambarkan hubungan antar *Menu* pada aplikasi *game* yang akan dibangun. Berikut adalah jaringan semantik *game* yang akan dibangun terdapat pada Gambar 3.35.



Gambar 3.35 Jaringan Semantik