BAB 3
ANALISIS DAN PERANCANGAN SISTEM

3.1. Analisis

Pada bagian analisis sistem ini, akan dilakukan berbagai macam analisis yang berhubungan dengan *game* yang akan dibangun, mulai dari analisis terhadap *game* sejenis sampai dengan analisis berbagai kebutuhan baik fungsional maupun non-fungsional.

3.1.1. Analisis *Game* Sejenis

Pada analisis *game* sejenis, akan dilakukan observasi terhadap *game* yang memiliki kesamaan *gameplay* dan *genre* dengan *game* yang akan dibangun.

3.1.1.1. *Game Grand theft Auto*

Game yang pertama yaitu *game* yang bernama *Grand theft auto* yang merupakan *game* konsol dari *Gameboy Advance Color*.



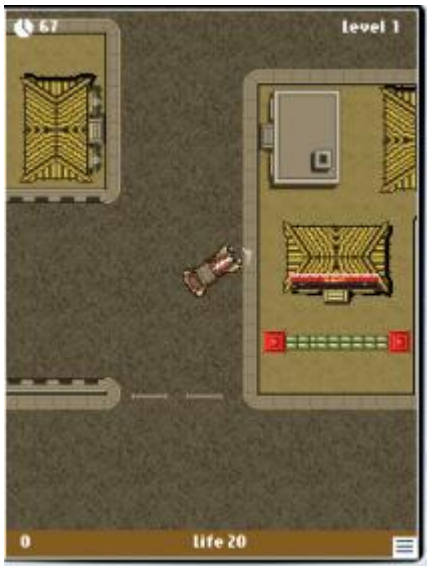
Gambar 3.1 *Grand theft auto (gameboy color)*

Game ini menawarkan sebuah permainan *game* ber-*genre advanture*. Pada *game* tersebut tampak atas dengan lokasi seperti jalan raya. Karakter utama pada

game ini adalah seseorang yang sedang melakukan misi-misi yang harus dikerjakan. Pada *game* ini pemain hanya menyusuri jalan menuju tempat yang dituju. Tidak ada musuh dan halangan pada *game* ini

3.1.1.2. *Mafia Driver Chinatown*

Game yang kedua adalah *game* yang bernama *Mafia Driver Chinatown* yang merupakan *game handheld* J2ME.



Gambar 3.2 *Mafia Driver Chinatown*

Game ini menawarkan sebuah permainan *game* ber-*genre advanture*. *Game* ini tidak jauh berbeda dengan *game* yang pertama, dimana pemain menyusuri jalan menuju tempat yang dituju. Tidak ada musuh dan halangan pada *game* ini.

3.1.2. Analisis Masalah

Berdasarkan analisis yang dilakukan pada *game Grand theft auto gameboy color* dan *Mafia driver in Chinatown* didapat kesimpulan bahwa pada *game* tersebut mempunyai kekurangan, diantaranya ketika pemain mencari tempat tujuan, tidak ada halangan atau orang yang mengejar. Solusi untuk memperbaiki kekurangan tersebut adalah :

1. Menambah halangan pada permainan, ketika ditambah halangan kemungkinan akan terjadi tubrukan (*collision* detection) antara karakter utama dan halangan. Dalam analsis ini *collision detection* yang digunakan adalah rectangle *collision*

2. Menambah musuh pada permainan, agar musuh menjadi lebih pintar maka akan diterapkan kecerdasan buatan. Kecerdasan buatan yang diterapkan untuk musuh bisa mengejar pemain salah satunya adalah dengan menerapkan algoritma A*. Algoritma A* (A *star*) merupakan algoritma yang terbaik dalam pencarian rute terpendek, jadi sangat cocok bila diterapkan ke dalam sebuah *game* yang akan dibuat.

Tabel 3.1 Analisis *game* sejenis

Analisis *Game*	GTA	China Town
Device *game*	Konsol (*Game* Boy Advance)	J2ME
Pemain	Single Player	Single Player
*Game*play	a. Seorang gangster yang harus menyelesaikan tugas-tugasnya b. Mengelilingi kota, menyusuri jalan sampai menemukan spot atau tempat yang dituju untuk mencari misi selanjutnya. c. Melawan gangster-gangster lain yaitu lawan dari pemain.	Mengelilingi kota , menyusuri jalan sampai menemukan tempat yang dituju.
Basis	*Game*Time : Menggunakan waktu yang ersedia dalam permainan.	*Game*Time : Menggunakan waktu yang ersedia dalam permainan.
Fitur	a. Highscore b. Bisa memakai senjata c. Menaiki kendaraan	Highscore

Kelebihan	Misi yang tidak terlalu sulit karena hanya mengikuti perintah dari misi.	a. *Game*play yang mudah untuk dimainkan b. Tidak terlalu memenuhi memory telepon.
Kekurangan	a. Grafik *game* yang kurang jelas b. Control *game* yang tidak terlalu mudah	a. Menggerakkan pemain sangat sulit b. Permainan monoton karena tidak ada lawan yang mengejar c. Hanya mencari spot sebagai selesainya misi tiap level.

3.2. Analisis Pembangunan Aplikasi *Game Crazy* Becak

Pada bagian ini akan dijelaskan mengenai analisis-analisis *game* yang akan dibuat dari pengenalan *game* sampai dengan metode dan algoritma yang akan digunakan.

3.2.1. Pengenalan *Game*

Nama Pada Aplikasi *game* ini adalah *Crazy* Becak. *Game* ini merupakan *game* ber-*genre Maze-Adventure*. Pemberian nama pada *game* ini dilihat dari permainannya karena pemain harus menghindar dari halangan-halangan dalam perjalanannya sampai tujuan. *Game* ini merupakan *game* 2D, dan dimainkan dengan menggunakan *handphone* berbasis java. Karena dimainkan dengan *handphone* maka permainan ini merupakan *single player.*

3.2.2. Alur Permainan

Alur permainan dalam *game Crazy* Becak adalah sebagai berikut:

1. Becak bergerak menggunakan keypad handphone.
2. Pemain harus mencari spot, atau tempat yang harus dituju.
3. Bila pemain mengenai halangan dan mobil maka becak akan terjadi tabrakan dan mengurangi *lifebar*.

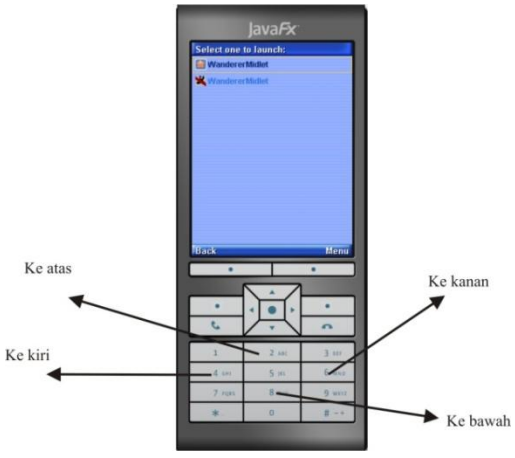
4. Bila *lifebar* habis maka *game* over dan permainan kembali ke awal.
5. Jika pemain sampai ke tujuan per *level* maka skor akan ditentukan dari sisa waktu dari pemain dikalikan dengan 100.
6. Di dalam permainan ada wilayah dimana becak tidak boleh melewatinya.
7. Bila becak melewatinya maka satpol yang menjaga akan mengejar becak, tetapi apabila becak kembali ke wilayah yang boleh dilewati oleh becak maka satpol berhenti untuk mengejar.
8. Bila Becak terkejar dan terjadi *collision* dengan satpol maka permainan berakhir dan harus mengulangi dari level yang sedang dimainkan.

3.2.3. Misi Permainan

Misi dari permainan ini adalah melakukan perjalanan dari tempat awal permainan menuju tempat yang akan dituju dengan tanda tujuan. Selain itu pemain juga harus mengumpulkan koin yang disediakan.

3.2.4. *Game*play

Pemain dapat memainkan *game* ini dengan menggunakan keypad pada *handphone* yang sudah mendukung java. Untuk menghindari dari halangan pemain bisa menekan angka 2 untuk keatas, 4 untuk ke kiri, 6 untuk mengarahkan ke kanan, dan 8 untuk mengarahkan karakter utama ke bawah.



Gambar 3.3 *Gameplay game Crazy* Becak

3.2.5. Sasaran user dan Kegunaan aplikasi permainan

Adapun sasaran *user* atau dalam hal ini pemain yang memainkan aplikasi permainan *Crazy Becak* anak-anak mulai dari umur 8 tahun, alasannya adalah pada aplikasi permainan ini para pemain tidak terlalu dipusingkan dengan *gameplay*-nya.

Kegunaan dari aplikasi permainan *Crazy Becak* ini selain menjadi sarana hiburan bagi para pemainnya juga dapat melatih kecepatan otak dalam berpikir. Pengambilan keputusan pada *game* ini akan melatih otak pemain untuk terampil dan cepat dalam mengambil tindakan.

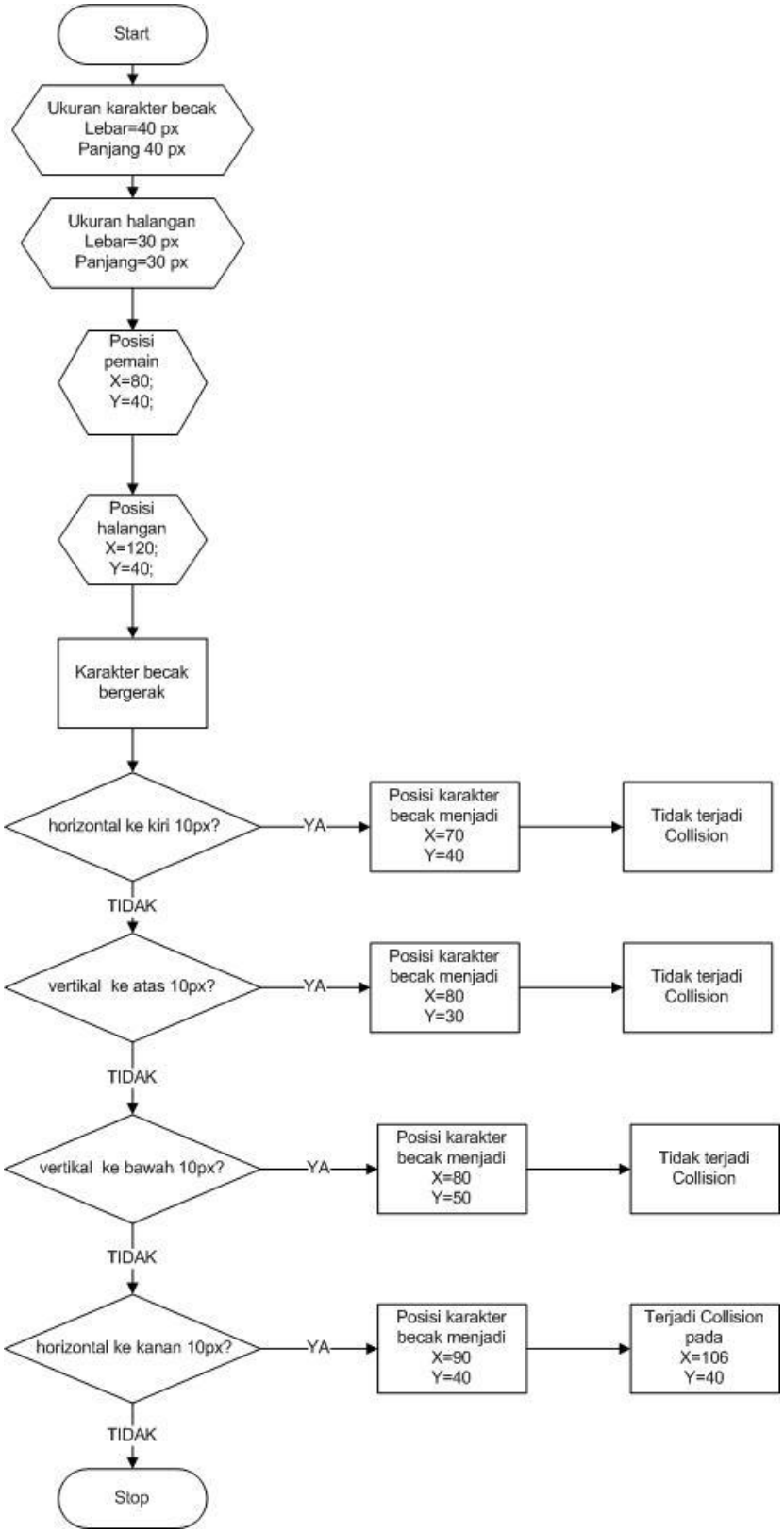
Aplikasi *game* ini dimainkan dengan *mobile phone* yang sudah mendukung java (j2me).

3.2.6. Analisis Metode dan Algoritma

Analisis metode dan Algoritma yang akan dijelaskan adalah metode *collision* detection dan algoritma A*.

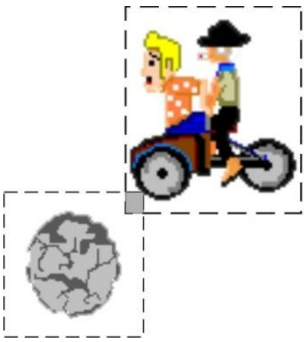
3.2.6.1. *Collision* Detection

Metode *Collision Detection* adalah Pendeteksian proses tabrakan antara dua objek atau lebih. *Collision Detection* sendiri terjadi ketika pemain menabrak halangan, sehingga terjadi proses tabrakan antara objek halangan dan karakter utama permainan ini. Ketika tabrakan terjadi akan terpicu sebuah event yaitu berkurangnya *lifebar*. *Flowchart Collision detection* bisa dilihat pada gambar 3.3.



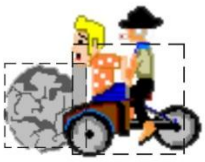
Gambar 3.4 *Flowchart Collision* Detection

Teknik *collision detection* yang digunakan pada aplikasi *game Crazy* becak ini adalah teknik *rectangle collision.* Secara default dengan menggunakan teknik *rectangle collision.* maka objek *sprite* akan dianggap saling bertabrakan apabila kotak *shape image* bersinggungan tanpa memperdulikan *pixel* gambar yang tertabrak.



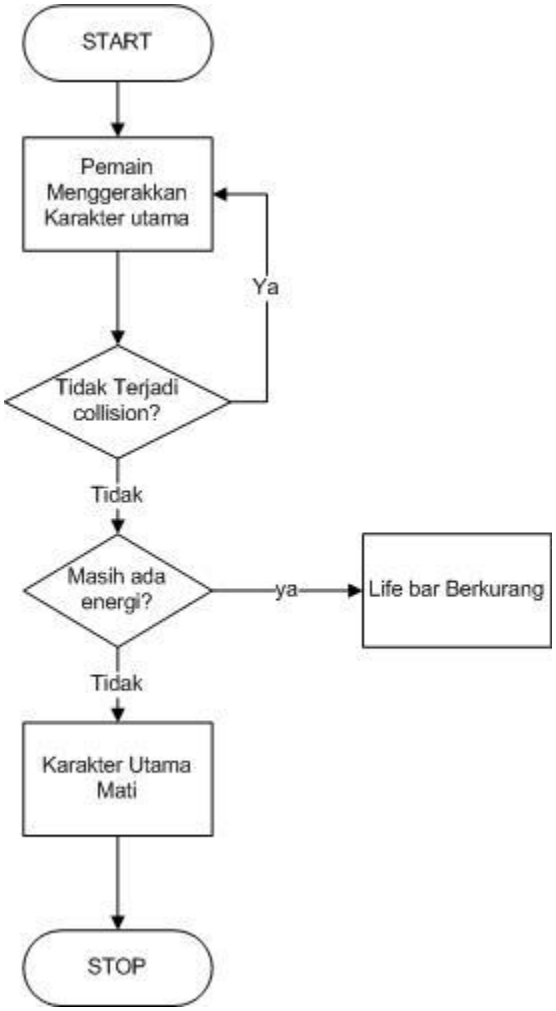
Gambar 3.5 rectangle *collision* detection secara default.

Tetapi hal ini menyebabkan ketidaksempurnaan fungsi *collision* detection. Maka dari itu di J2ME ada fungsi atau method yang bisa menentukan area piksel mana terjadinya *collision*.



Gambar 3.6 rectangle *collision* dengan piksel yang diinginkan

. Berikut adalah *Flowchart* diagram ketika pemain menabrak halangan dan musuh yang disini adalah satpol.



Gambar 3.7 *Flowchart* ketika Becak menabrak halangan



Gambar 3.8 *Flowchart* ketika Becak mengenai satpol

3.2.6.2. Analisis Algoritma

AI merupakan faktor terpenting dalam meningkatkan *game*play pada semua jenis permainan. Hal penting dalam menentukan keberhasilan suatu aplikasi *game* yang menerapkan AI adalah kesuksesan dalam pencarian dan pelacakan. Pada dasarnya ada dua teknik pencarian dan pelacakan yang digunakan, yaitu pencarian buta (*blind search*) dan pencarian terbimbing (*heuristic search*). Dalam pencarian terbimbing ada beberapa algoritma A*. Algoritma ini dapat menyelesaikan permasalahan secara optimal untuk menemukan jalan terpendek menuju suatu tujuan.

1. Analisis Penggunaan algoritma A* (A star)

Pada *game Crazy* Becak arena permainan berupa kumpulan tile-tile yang terdiri dari orde X x Y. Tiap tile memiliki koordinat X dan Y atau berbentuk seperti array dua dimensi.

	0	1	2
0	(0,0)	(1,0)	(2,0)
1	(0,1)	(1,1)	(2,1)
2	(0,2)	(1,2)	(2,2)

Gambar 3.9 tile map orde X x Y

Algoritma A* akan diterapkan pada objek musuh dalam melakukan pencarian karakter pemain dalam arena permainan. Adapun istilah-istilah yang akan dibahas yaitu open list, Closed list, nilai f,g dan h.

1. OPEN LIST adalah list yang menyimpan kemungkinan path yang akan diperiksa. OPEN LIST dibuat terurut berdasarkan nilai f. OPEN LIST digunakan untuk menentukan secara selektif (berdasarkan nilai f) jalan yang dikira lebih dekat menuju pada path tujuan. OPEN berisi simpul-simpul yang masih memiliki peluang untuk terpilih sebagai simpul terbaik.

2. CLOSED LIST adalah list untuk menyimpan simpul-simpul yang sudah pernah dibangkitkan dan sudah terpilih sebagai simpul terbaik atau list yang menyimpan jalan yang sudah diperiksa dari open list. Artinya, CLOSED berisi simpul-simpul yang tidak mungkin terpilih sebagai simpul terbaik (peluang untuk terpilih sudah tertutup). Kedua list (OPEN LIST dan CLOSED LIST) ini bertujuan juga untuk menghindari penulusuran berkali-kali jalan (rute) yang memang sudah diidentifikasi agar tidak masuk kembali ke dalam OPEN LIST.

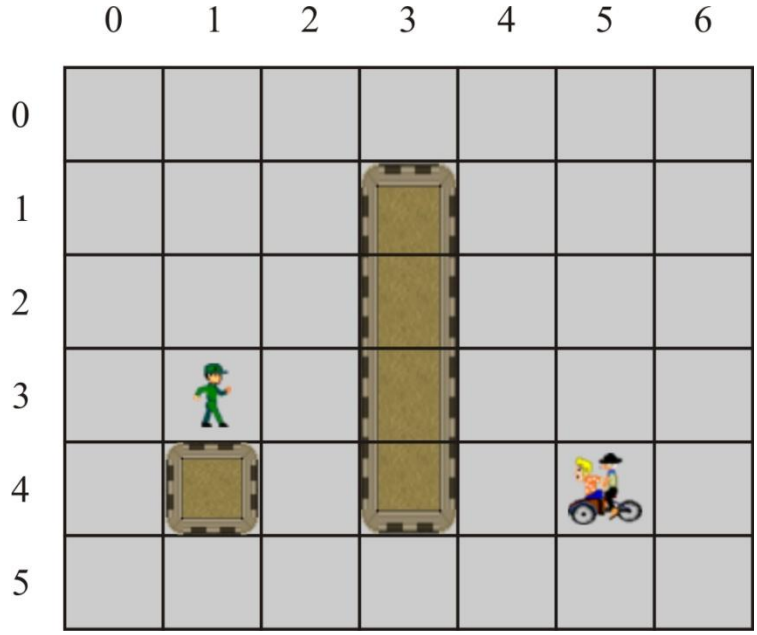
3. Nilai F adalah cost perkiraan suatu path yang diidentifikasikan. Nilai F merupakan hasil dari f(n).

4. Nilai G hasil dari fungsi g(n), adalah banyaknya langkah yang diperlukan untuk menuju ke path sekarang.

5. Setiap simpul harus memiliki informasi h(n), yaitu hasil estimasi harga simpul tersebut dihitung dari simpul tujuan yang hasilnya menjadi nilai H.

2. Ilustrasi perhitungan Algoritma A*

Arena permainan pada *game* Crazy Becak terdiri dari orde X x Y. Dalam analisis ini akan dijelaskan dengan ordo 7 x 6. Simpul awal atau posisi musuh dicontohkan berada pada koordinat (1,3), sedangkan simpul tujuan atau posisi karakter pemain berada pada koordinat (5,4) di arena permainan.



Gambar 3.10 Contoh kondisi awal.

Langkah selanjutnya akan dilakukan pemeriksaan terhadap simpul-simpul yang bertetangga yang memiliki nilai cost / biay terkecil sebagai best node sebagai rute yang dipilih musuh, dengan melakukan langkah perhitungan sebagai berikut.

1. Langkah Pertama

a. Koordinat n(2,3)

Nilai nx=2 Nilai ny=3

Simpul Tujuan (5,4) sehingga goal.x=5, goal.y=4

g(2,3)=1

h(2,3) = (abs(nx-goal.x)+abs(ny-goal.y))

h(2,3)=(abs(2-5)+abs(3-4))

h(2,3)=abs(-3)+abs(-1)

h(2,3)=3+1

h(2,3)=4

f(2,3)=g(2,3)+h(2,3)

f(2,3)=1+4

f(2,3)=5

b. Koordinat (1,2)

Nilai nx=1 Nilai ny=2

Simpul Tujuan (5,4) sehingga goal.x=5, goal.y=4

g(1,2) = 1

h(1,2) = (abs(nx-goal.x)+abs(ny-goal.y))

h(1,2) = (abs(1-5)+abs(2-4))

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

h(1,2) = 4+2

h(1,2) = 6

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

f(1,2) = 1+6

f(1,2) = 7

c. Koordinat(0,3)

Nilai nx=0 Nilai ny=3

Simpul Tujuan (5,4) sehingga goal.x=5, goal.y=4

g(0,3)=1

h(0,3) = (abs(nx-goal.x)+abs(ny-goal.y))

h(0,3)=(abs(0-5)+abs(3-4))

h(0,3)=abs(-5)+abs(-1)

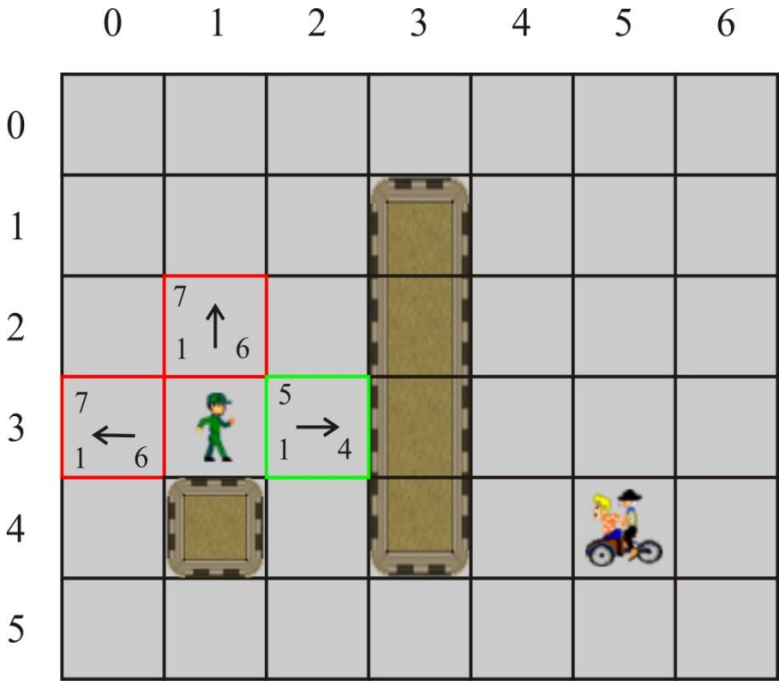
h(0,3)=5+1

h(0,3)=6

f(0,3)=g(0,3)+h(0,3)

f(0,3)=1+6

f(0,3)=7



Gambar 3.11 Langkah pertama pencarian best Node

Keterangan:

f(n) : berada di dalam simpul di tepi kiri atas

g(n) : berada didalam simpul di tepi kiri bawah

h(n) : berada di dalam simpul di tepi kanan bawah

Dari hasil perhitungan diatas seperti terlihat pada gambar 3. Tiga simpul yang mungkin menjadi bestnode. Dari ketiga simpul tersebut yang mungkin menjadi bestnode maka dipilihlah simpul (2,3) sebagai bestnode karena memiliki biaya terkecil yaitu 5.

2. Langkah kedua

a. Koordinat (2,2)

Nilai nx=2 Nilai ny=2

Simpul Tujuan (5,4) sehingga goal.x=5, goal.y=4

g(2,2)=2

h(2,2) = (abs(nx-goal.x)+abs(ny-goal.y))

h(2,2)=(abs(2-5)+abs(2-4))

h(2,2)=abs(-3)+abs(-2)

h(2,2)=3+2

h(2,2)=5

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

f(2,2)=1+6

f(2,2)=7

b. Koordinat (2,4)

Nilai nx=2 Nilai ny=4

Simpul Tujuan (5,4) sehingga goal.x=5, goal.y=4

g(2,4)=2

h(2,4) = (abs(nx-goal.x)+abs(ny-goal.y))

h(2,4)=(abs(2-5)+abs(4-4))

h(2,4)=abs(-3)+abs(0)

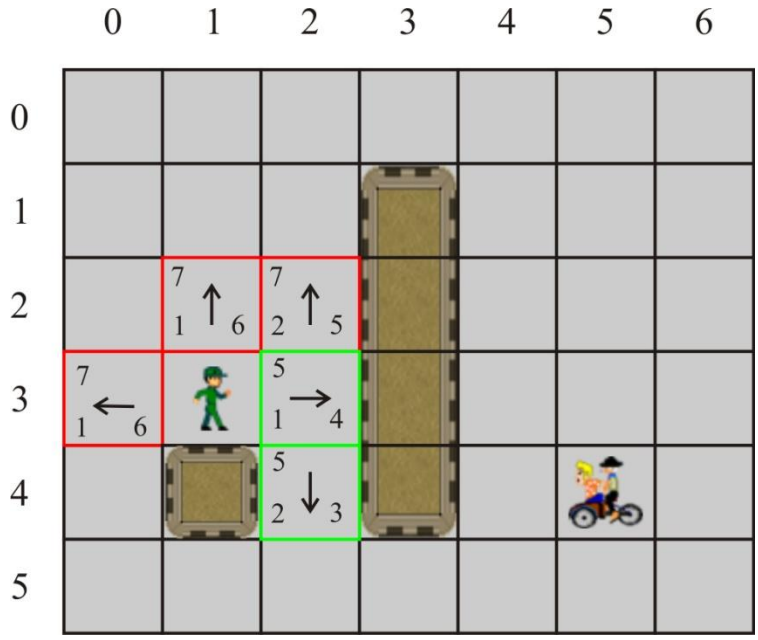
h(2,4)=3+0

h(2,4)=3

f(2,4)=g(2,4)+h(2,4)

f(2,4)=2+3

f(2,4)=5



Gambar 3.12 Langkah kedua pencarian best Node

Setelah mendapatkan bestNode dilangkah pertama, dilakukan penelusuran selanjutnya untuk mendapatkan bestnode selanjutnya dengan melakukan penelusuran simpul-simpul yang memiliki nilai cost terkecil. Simpul yang dihitung selanjutnya adalah koordinat (2,2) dan (2,4), koordinat (3,3) tidak dihitung atau diabaikan karena terdapat penghalang yang tidak bisa dilewati musuh. Hasilnya adalah koordinat (2,4) mempunyai cost terkecil yaitu 5, maka koordinat (2,4) dipilih sebagai bestnode selanjutnya.

3. Langkah ketiga

   a. Koordinat (2,5)

   Nilai nx=2 Nilai ny=5

   Simpul Tujuan (5,4) sehingga goal.x=5, goal.y=4

   g(2,5)=3

   h(2,5) = (abs(nx-goal.x)+abs(ny-goal.y))

   h(2,5)=(abs(2-5)+abs(5-4))

h(2,5)=abs(-3)+abs(1)

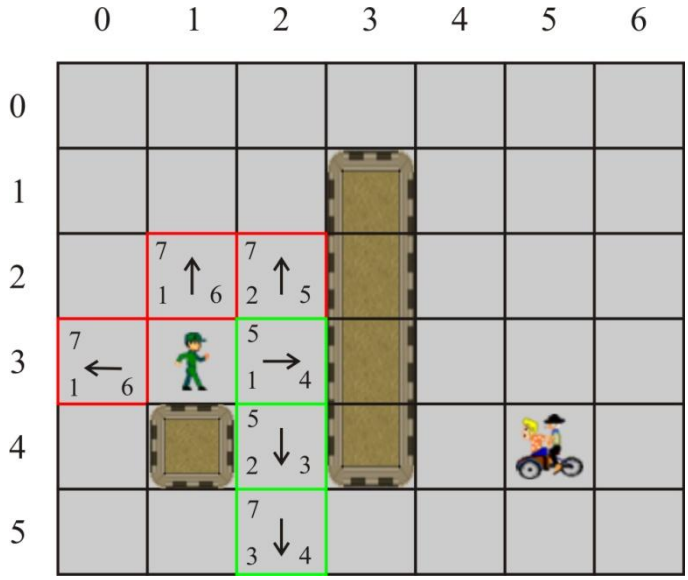
h(2,5)=3+1

h(2,5)=4

f(2,5)=g(2,5)+h(2,5)

f(2,5)=3+4

f(2,5)=7



Gambar 3.13 Langkah ketiga pencarian best Node

Setelah mendapatkan bestNode dilangkah kedua, dilakukan penelusuran selanjutnya untuk mendapatkan bestnode selanjutnya dengan melakukan penelusuran simpul-simpul yang memiliki nilai cost terkecil. Simpul yang dihitung selanjutnya adalah koordinat (2,5), koordinat (1,4) dan (3,4) tidak dihitung atau diabaikan karena terdapat penghalang yang tidak bisa dilewati musuh. Hasilnya adalah koordinat (2,5) mempunyai nilai cost 7, maka koordinat (2,5) dipilih sebagai bestnode selanjutnya.

4. Langkah keempat.

   a. Koordinat (3,5)

Nilai nx=3 Nilai ny=5

Simpul Tujuan (5,4) sehingga goal.x=5, goal.y=4

g(3,5)=4

h(3,5) = (abs(nx-goal.x)+abs(ny-goal.y))

h(3,5)=(abs(3-5)+abs(5-4))

h(3,5)=abs(-2)+abs(1)

h(3,5)=2+1

h(3,5)=3

f(3,5)=g(2,5)+h(2,5)

f(3,5)=4+3

f(3,5)=7

b. Koordinat (1,5)

Nilai nx=1 Nilai ny=5

Simpul Tujuan (5,4) sehingga goal.x=5, goal.y=4

g(1,5)=4

h(1,5) = (abs(nx-goal.x)+abs(ny-goal.y))

h(1,5)=(abs(1-5)+abs(5-4))

h(1,5)=abs(-4)+abs(1)

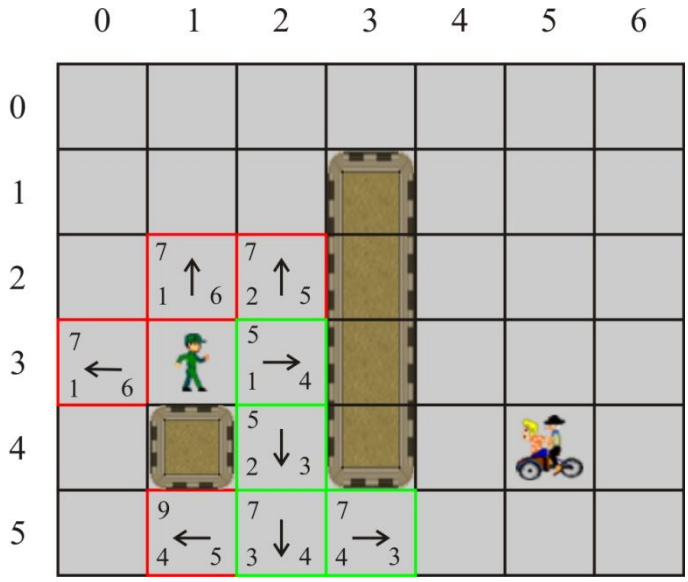
h(1,5)=4+1

h(1,5)=5

f(1,5)=g(2,5)+h(2,5)

f(1,5)=4+5

f(1,5)=9



Gambar 3.14 Langkah keempat pencarian best Node

Setelah mendapatkan bestNode dilangkah ketiga, dilakukan penelusuran selanjutnya untuk mendapatkan bestnode selanjutnya dengan melakukan penelusuran simpul-simpul yang memiliki nilai cost terkecil. Simpul yang dihitung selanjutnya adalah koordinat (3,5) dan koordinat (1,5). Hasilnya adalah koordinat (3,5) mempunyai cost terkecil yaitu 7, maka koordinat (3,5) dipilih sebagai bestnode selanjutnya.

5. Langkah kelima

a. Koordinat (4,5)

Nilai nx=4 Nilai ny=5

Simpul Tujuan (5,4) sehingga goal.x=5, goal.y=4

g(4,5)=5

h(4,5) = (abs(nx-goal.x)+abs(ny-goal.y))

h(4,5)=(abs(4-5)+abs(5-4))

h(4,5)=abs(-1)+abs(1)

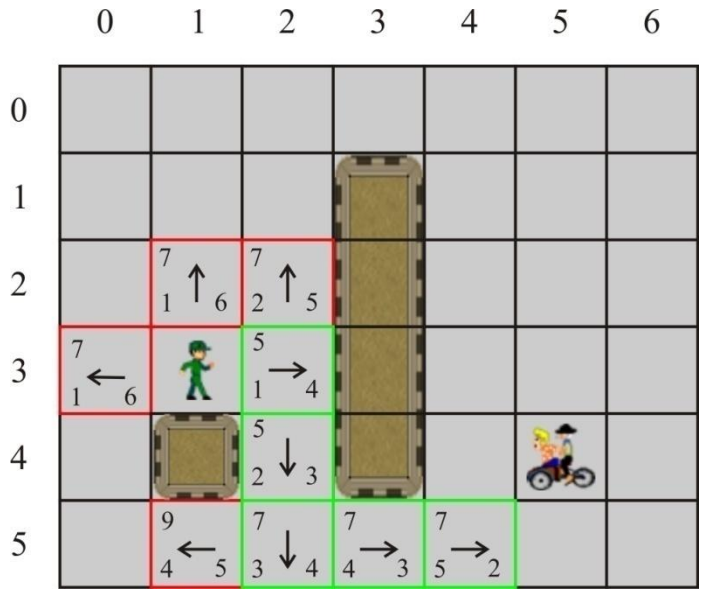
h(4,5)=1+1

h(4,5)=2

f(4,5)=g(4,5)+h(4,5)

f(4,5)=5+2

f(4,5)=7



Gambar 3.15 Langkah kelima pencarian best Node

Setelah mendapatkan bestNode dilangkah keempat, dilakukan penelusuran selanjutnya untuk mendapatkan bestnode selanjutnya dengan melakukan penelusuran simpul-simpul yang memiliki nilai cost terkecil. Simpul yang dihitung selanjutnya adalah koordinat (4,5), koordinat (3,4) tidak dihitung atau diabaikan karena terdapat penghalang yang tidak bisa dilewati musuh. Hasilnya adalah koordinat (4,5) mempunyai nilai cost 7, maka koordinat (4,5) dipilih sebagai bestnode selanjutnya.

6. Langkah keenam

a. Koordinat (5,5)

Nilai nx=5 Nilai ny=5

Simpul Tujuan (5,4) sehingga goal.x=5, goal.y=4

g(5,5)=6

h(5,5) = (abs(nx-goal.x)+abs(ny-goal.y))

h(5,5)=(abs(5-5)+abs(5-4))

h(5,5)=abs(0)+abs(1)

h(5,5)=0+1

h(5,5)=1

f(5,5)=g(5,5)+h(5,5)

f(5,5)=6+1

f(5,5)=7

b. Koordinat (4,4)

Nilai nx=4 Nilai ny=4

Simpul Tujuan (5,4) sehingga goal.x=5, goal.y=4

g(4,4)=6

h(4,4) = (abs(nx-goal.x)+abs(ny-goal.y))

h(4,4)=(abs(4-5)+abs(4-4))

h(4,4)=abs(-1)+abs(0)

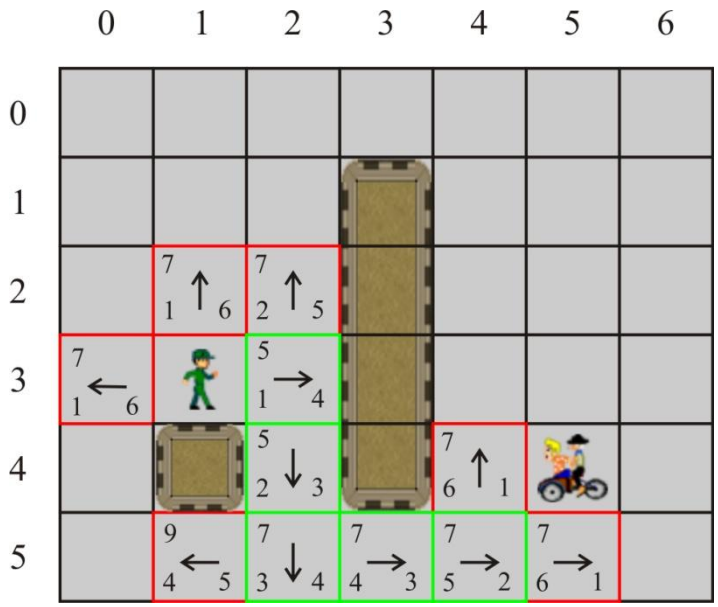
h(4,4)=1+0

h(4,4)=1

f(4,4)=g(4,4)+h(4,4)

f(4,4)=6+1

f(4,4)=7



Gambar 3.16 Langkah keenam pencarian best Node

Setelah mendapatkan bestNode dilangkah kelima, dilakukan penelusuran selanjutnya untuk mendapatkan bestnode selanjutnya dengan melakukan penelusuran simpul-simpul yang memiliki nilai cost terkecil. Simpul yang dihitung selanjutnya adalah koordinat (5,5) dan (4,4). Hasilnya adalah koordinat (5,5) dan (4,4) mempunyai nilai cost yang sama yaitu 7, maka akan dihitung untuk langkah selanjutnya dengan menghitung koordinat (5,4) dan (6,5).

7. Langkah ketujuh

a. Koordinat (5,4)

Nilai nx=5 Nilai ny=4

Simpul Tujuan (5,4) sehingga goal.x=5, goal.y=4

g(5,4)=7

h(5,4) = (abs(nx-goal.x)+abs(ny-goal.y))

h(5,4)=(abs(5-5)+abs(4-4))

h(5,4)=abs(0)+abs(0)

h(5,4)=0+0

h(5,4)=0

f(5,4)=g(5,4)+h(5,4)

f(5,4)=7+0

f(5,4)=7

b. Koordinat (6,5)

Nilai nx=6 Nilai ny=5

Simpul Tujuan (5,4) sehingga goal.x=5, goal.y=4

g(6,5)=7

h(6,5) = (abs(nx-goal.x)+abs(ny-goal.y))

h(6,5)=(abs(6-5)+abs(5-4))

h(6,5)=abs(1)+abs(1)

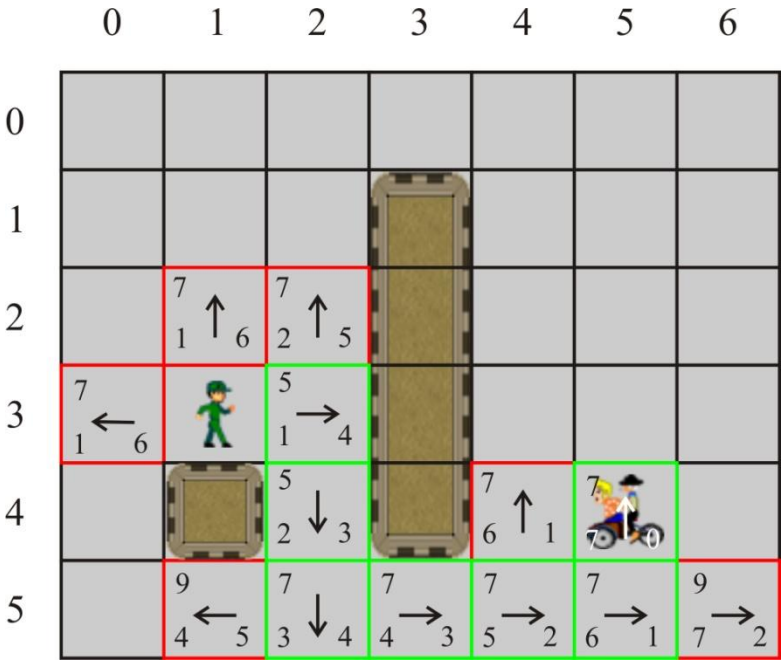
h(6,5)=1+1

h(6,5)=2

f(6,5)=g(6,5)+h(6,5)

f(6,5)=7+2

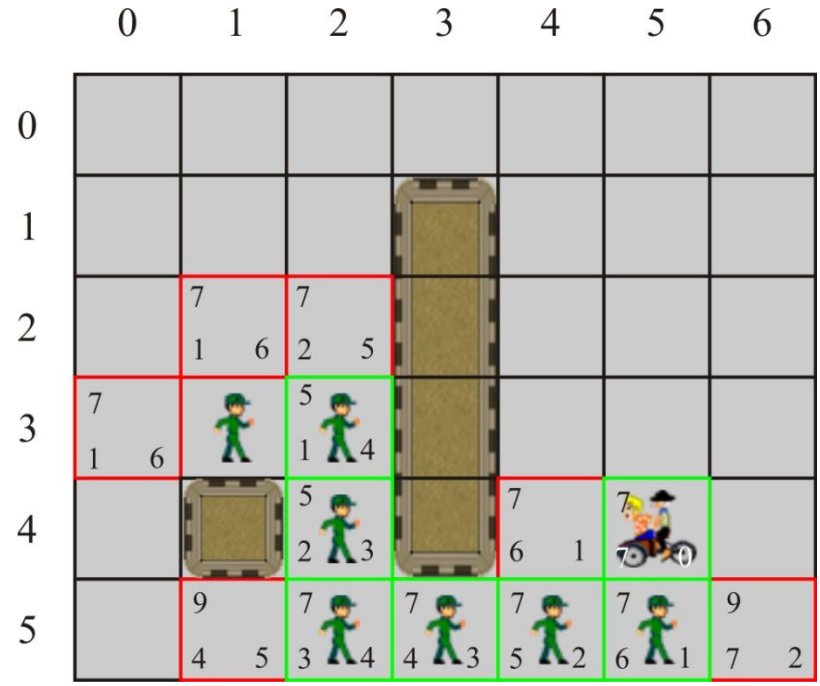
f(6,5)=9



Gambar 3.17 Langkah ketujuh pencarian best Node

Setelah mendapatkan bestNode dilangkah keenam, dilakukan penelusuran selanjutnya untuk mendapatkan bestnode selanjutnya dengan melakukan penelusuran simpul-simpul yang memiliki nilai cost terkecil. Simpul yang dihitung selanjutnya adalah koordinat (6,5) dan (5,4). Hasilnya adalah koordinat (5,4) mempunyai cost paling kecil yaitu 7, maka koordinat (5,4) dipilih sebagai bestnode selanjutnya.

Dari perhitunganyang telah dilakukan dipilihlah biaya / cost terkecil pada setiap langkahnya sehingga akan menghasilkan jalur terpendek yang akan dilalui musuh untuk melakukan pencarian posisi pemain yang terlihat pada gambar 3.16.



Gambar 3.18 Hasil pencarian jalan menggunakan algoritma A*

Pada *game Crazy* becak yang akan dibangun algoritma A* dijalankan di setiap *level*. Algoritma ini akan berjalan apabila becak melewati jalan yang tidak boleh dilewati oleh becak. Ketika becak melewati jalan tersebut maka satpol mulai mengejar becak tersebut.

3.3. Analisis dan Kebutuhan Non-Fungsional

Analisis Kebutuhan non-Fungsional terbagi menjadi dua yaitu analisis kebutuhan perangkat lunak dan analisis kebutuhan perangkat keras.

3.3.1. Analisis Kebutuhan Perangkat lunak

Analisis kebutuhan perangkat lunak yaitu untuk pengguna dan pengembang.

3.3.1.1. Pengembang

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 *game crazy* becak ini adalah sebagai berikut :

1. Sistem Operasi Windows 7 Professional
2. Java Developmen Kit 6.24
3. Menggunakan Netbean IDE versi 6.9.1

3.3.1.2. Pengguna

Adapun spesifikasi perangkat lunak yang digunakan untuk menjalankan aplikasi permainan *Crazy* becak adalah perangkat device mobile yang sudah berbasis Java.

3.3.2. Analisis Kebutuhan Perangkat Keras

Analisis Kebutuhan perangkat kerang terbagi menjadi dua yaitu untuk pengguna dan pengembang.

3.3.2.1. Pengembang

Komputer terdiri dari perangkat keras dan perangkat lunak yang saling berinteraksi. Perangkat lunak memberikan instruksi-instruksi kepada perangkat keras untuk melakukan tugas tertentu. Adapun perangkat keras yang digunakan dalam membangun aplikasi *game Crazy* Becak adalah sebagai berikut :

1. Processor intel core 2 duo
2. RAM 1 GB
3. Hardisk 120 GB
4. Monitor LCD 1200x800

3.3.2.2. Pengguna

Mobile handphone yang digunakan minimal harus mempunyai spesifikasi perangkat keras sebagai berikut:

1. CLDC 1.1

2. MIDP 2.0

3. Ukuran layar 240x320 pixel

3.3.3. Analisis Pengguna

Analisis pengguna digunakan untuk mengetahui spesifikasi pengguna untuk dapat memainkan aplikasi permainan *Crazy* Becak.

Tabel 3.2 Analisis Pengguna

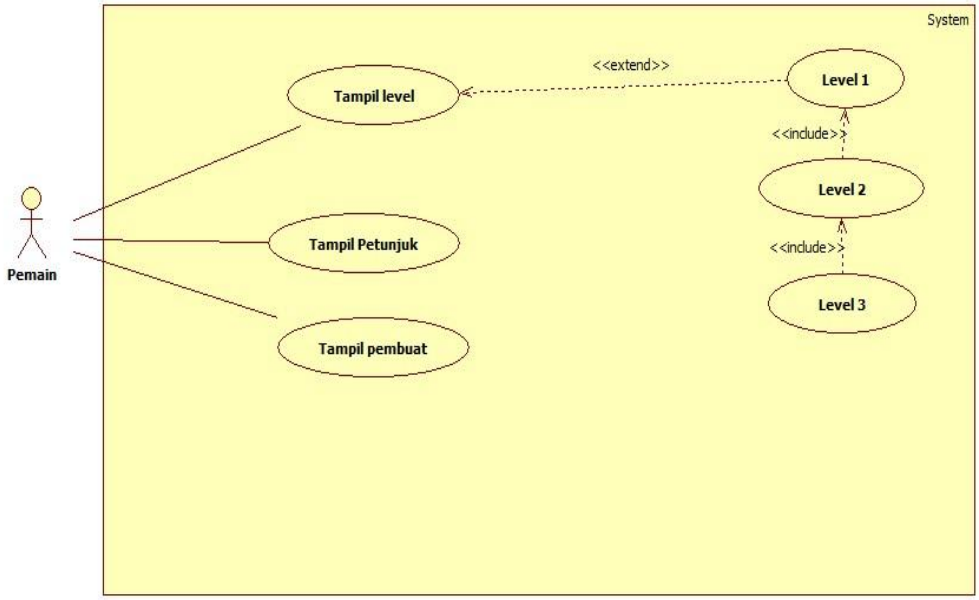
Peran	Pemain
Usia	8 tahun keatas
Tingkat keterampilan	Minimal bisa menggunakan handphone berbasis java

3.4. Analisis dan Kebutuhan Fungsional

Kebutuhan Fungsional yang akan digunakan adalah diagram *usecase*, *activity*, *class* dan *sequence.*

3.4.1. Diagram *Usecase*

Secara umum, interaksi antara pengguna yaitu pemain dengan aplikasi *game Crazy becak* digambarkan pada diagram *Use Case* adalah sebagai berikut :



Gambar 3.19 Diagram *Use Case Crazy* Becak

3.4.1.1. Definisi *Usecase*

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

Tabel 3.3 Definisi *Use case*

No.	*Use Case*	Deskripsi
1	Tampil *Level*	Fungsional untuk memilih *level* permainan
2	Tampil Petunjuk	Fungsional untuk melihat pentunjuk cara bermain aplikasi *game* tersebut
3	Tampil Tentang	Fungsional untuk melihat informasi pembuat aplikasi
4	*Level* 1	Tingkatan *Level* 1
5	*Level* 2	Tingkatan *Level* 2
6	*Level* 3	Tingkatan *Level* 3

3.4.1.2. Skenario *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*.

Nama *Usecase* : Tampil *Level*

Skenario :

Tabel 3.4 Scenario *Use Case* Tampil *Level*

Identifikasi	
Nama	Tampil *Level*
Tujuan	Masuk ke arena permainan sesuai *level*
Aktor	Pemain
Skenario Utama	

Kondisi Awal	Memilih mulai permainan	
Aksi Aktor		**Reaksi Sistem**
Pilih Mulai permainan.		
		Aplikasi merespon dengan cara memanggil dan menampilkan pilihan *level.*
Memilih Level.		
		Menampilkan arena permainan sesuai level yang dipilih.
Kondisi Akhir	Aplikasi Menampilkan arena permainan sesuai dengan level yang dipilih.	

Nama *Usecase* : Tampil Petunjuk

Skenario :

Tabel 3.5 Scenario *Use Case* Tampil Petunjuk

Identifikasi		
Nama	Tampil Petunjuk	
Tujuan	Melihat informasi mengenai *game*play dan tips bermain	
Aktor	Pemain	
Skenario Utama		
Kondisi Awal	Menu Utama ditampilkan	
Aksi Aktor		**Reaksi Sistem**
		Menampilkan menu utama
Pilih petunjuk		
		Aplikasi merespon dengan cara

	memanggil dan melihat kelas Petunjuk
Kondisi Akhir	Aplikasi menampilkan informasi petunjuk aplikasi *game*

Nama *Usecase* : Tampil Tentang

Skenario :

Tabel 3.6 Scenario *Use Case* Tampil Tentang

Identifikasi	
Nama	Tampil Tentang
Tujuan	Melihat informasi mengenai pembuat *game Crazy* Becak
Aktor	Pemain
Skenario Utama	
Kondisi Awal	Menu Utama ditampilkan
Aksi Aktor	**Reaksi Sistem**
	Menampilkan menu utama
Pilih dan pijit tentang	
	Aplikasi merespon dengan cara memanggil dan melihat kelas Tentang
Kondisi Akhir	Aplikasi Menampilkan informasi Pembuat aplikasi *game*.

Nama *Usecase* : *Level* 1

Skenario :

Tabel 3.7 Scenario *Use Case Level* 1

Identifikasi		
Nama	*Level* 1	
Tujuan	Menampilkan info permainan *level* 1	
Aktor	Pemain	
Skenario Utama		
Kondisi Awal	Menampilkan arena permainan *Level* 1	
Aksi Aktor		**Reaksi Sistem**
		Aplikasi menampilkan area permainan level 1
		Menginisialisasi Pemain, Map, Musuh, dan Halangan
Menggerakkan Karakter utama		
		Karakter utama bergerak
		Karakter utama menabrak halangan
		Apabila energi habis maka permainan akan kembali ke awal permainan tapi apabila belum habis maka permainan dilanjutkan
		Memeriksa waktu, jika waktu habis maka permainan akan kembali ke awal permainan bila belum habis permainan dilanjutkan
		Misi belum selesai, permainan

	dilanjutkan.
	Misi selesai
	Menampilkan info permainan
Kondisi Akhir	Aplikasi masuk ke arena dalam pilihan *level* 1 untuk menyelesaikan misi sehingga bisa menampilkan info dari permainan.

Nama *Usecase* : *Level* 2

Skenario :

Tabel 3.8 Scenario *Use Case Level* 2

Identifikasi	
Nama	*Level* 2
Tujuan	Menampilkan info permainan *level* 2
Aktor	Pemain
Skenario Utama	
Kondisi Awal	Menampilkan arena permainan *Level* 2
Aksi Aktor	**Reaksi Sistem**
	Aplikasi menampilkan arena permainan *level* 2
	Menginisialisasi Pemain, Map, Musuh, dan Halangan
Menggerakkan Karakter utama	
	Karakter utama bergerak
	Karakter utama menabrak halangan
	Apabila energi habis maka permainan

	akan kembali ke awal permainan tapi apabila belum habis maka permainan dilanjutkan
	Memeriksa waktu, jika waktu habis maka permainan akan kembali ke awal permainan bila belum habis permainan dilanjutkan
	Misi belum selesai, permainan dilanjutkan.
	Misi selesai
	Menampilkan info permainan
Kondisi Akhir	Aplikasi masuk ke arena dalam pilihan *level* 2 untuk menyelesaikan misi sehingga bisa menampilkan info terakhir dari permainan.

Nama *Usecase* : *Level* 3

Skenario :

Tabel 3.9 Scenario *Use Case Level* 3

Identifikasi	
Nama	*Level* 3
Tujuan	Menampilkan info permainan *level* 3
Aktor	Pemain
Skenario Utama	
Kondisi Awal	Menampilkan arena permainan *Level* 3
Aksi Aktor	**Reaksi Sistem**

	Aplikasi menampilkan arena permainan *level* 3
	Menginisialisasi Pemain, Map, Musuh, dan Halangan
Menggerakkan Karakter utama	
	Karakter utama bergerak
	Karakter utama menabrak halangan
	Apabila energi habis maka permainan akan kembali ke awal permainan tapi apabila belum habis maka permainan dilanjutkan
	Memeriksa waktu, jika waktu habis maka permainan akan kembali ke awal permainan bila belum habis permainan dilanjutkan
	Misi belum selesai, permainan dilanjutkan.
	Misi selesai
	Menampilkan info permainan
Kondisi Akhir	Aplikasi masuk ke arena dalam pilihan *level* 3 untuk menyelesaikan misi sehingga bisa menampilkan info terakhir dari permainan.

3.4.2. Diagram *Activity*

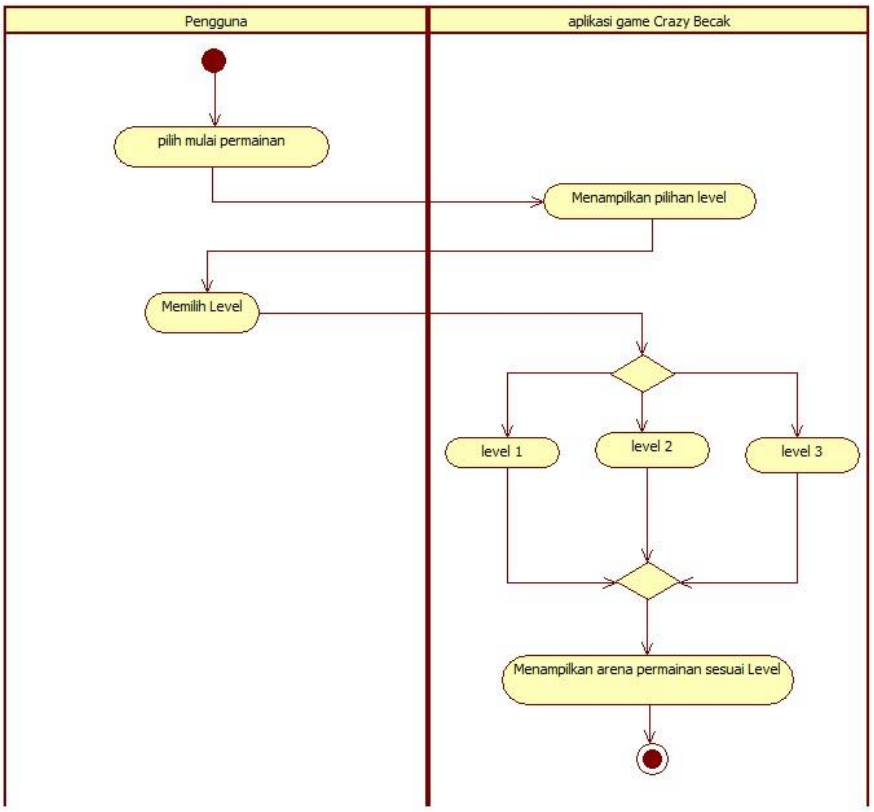
Diagram ini berisi penjelasan secara lengkap mengenai urutan aktivitas yang terjadi antara pemain dengan aplikasi permainan *Crazy* Becak.

3.4.2.1. *Activity* Diagram Tampil *Level*

Adapun penjelasan tentang Tampilan *level* yang terdapat pada aplikasi permainan *Crazy* Becak adalah sebagai berikut :

Tabel 3.10 Penjelasan Tampil *Level*

Deskripsi	Proses ini terjadi saat pemain memilih menu mulai permainan dari menu utama aplikasi permainan *Crazy* Becak.
Normal Flow	1. Pemain memilih Mulai permainan. 2. Pemain memilih *level* yang akan dimainkan.



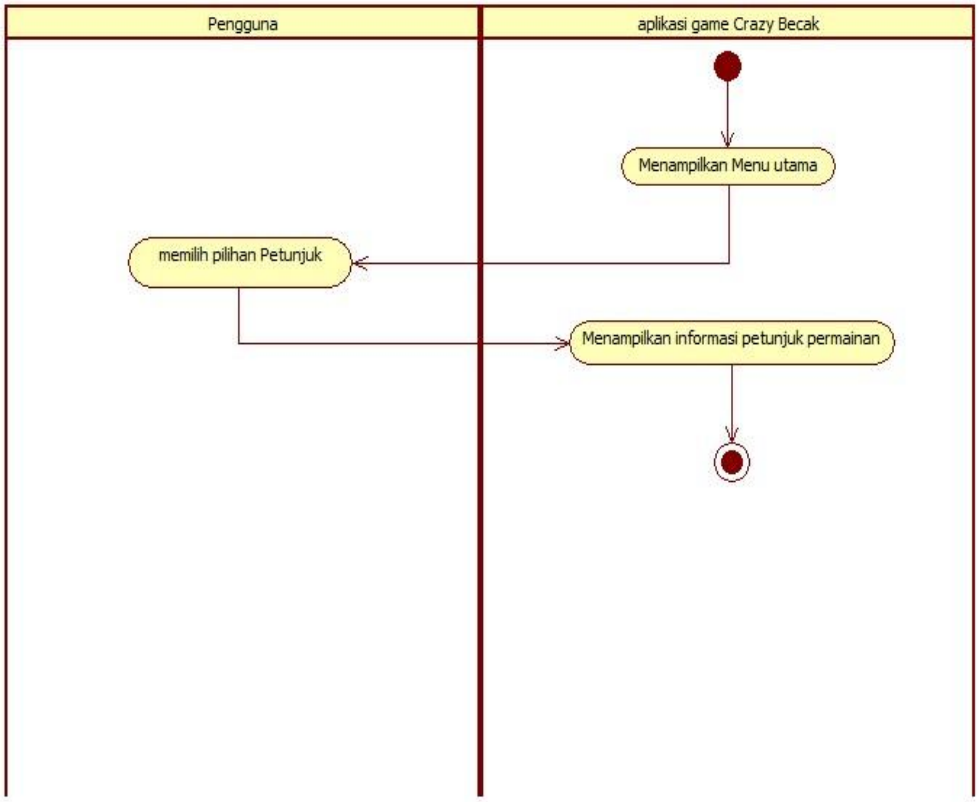
Gambar 3.20 Diagram *Activity* tampil *level Crazy* Becak

3.4.2.2. *Activity Diagram Tampil Petunjuk*

Adapun penjelasan tentang *view* petunjuk yang terdapat pada aplikasi permainan *Crazy* Becak adalah sebagai berikut :

Tabel 3.11 Penjelasan Tampil petunjuk

Deskripsi	Proses ini terjadi saat pemain memilih menu petunjuk dari menu utama aplikasi permainan *Crazy* Becak.
Normal Flow	1. Aplikasi menampilkan menu Utama. 2. Pemain memilih pilihan Petunjuk 3. Menampilkan informasi petunjuk dalam memainkan *game*.



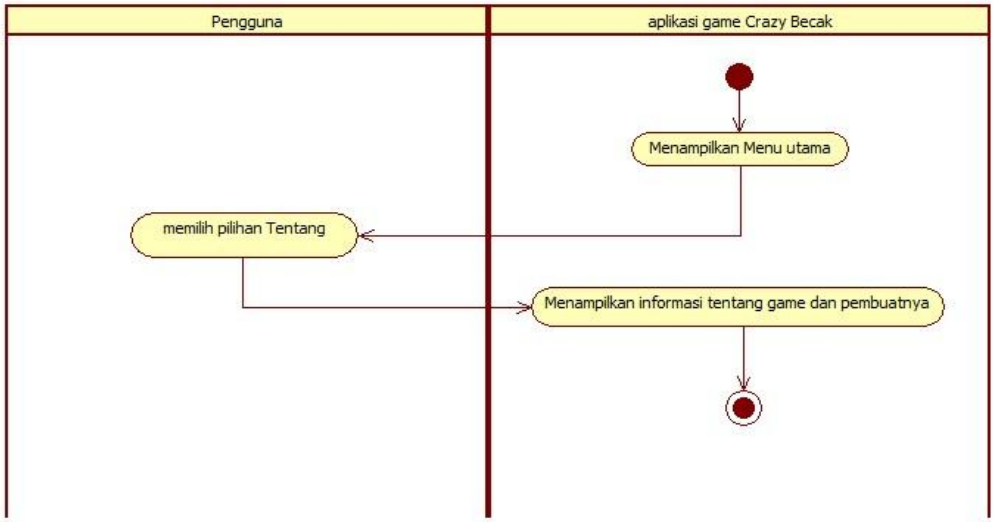
Gambar 3.21 Diagram *Activity* Tampil Petunjuk *Crazy* Becak

3.4.2.3. Activity Diagram Tampil Tentang

Adapun penjelasan tentang *view* Tentang yang terdapat pada aplikasi permainan *Crazy* Becak adalah sebagai berikut :

Tabel 3.12 Penjelasan Tampil tentang

Deskripsi	Proses ini terjadi saat pemain memilih menu Tentang dari menu utama aplikasi permainan *Crazy* Becak.
Normal Flow	1. Aplikasi menampilkan menu utama. 2. Pemain memilih pilihan Lihat Tentang. 3. Menampilkan informasi tentang *game* dan pembuat *game*.



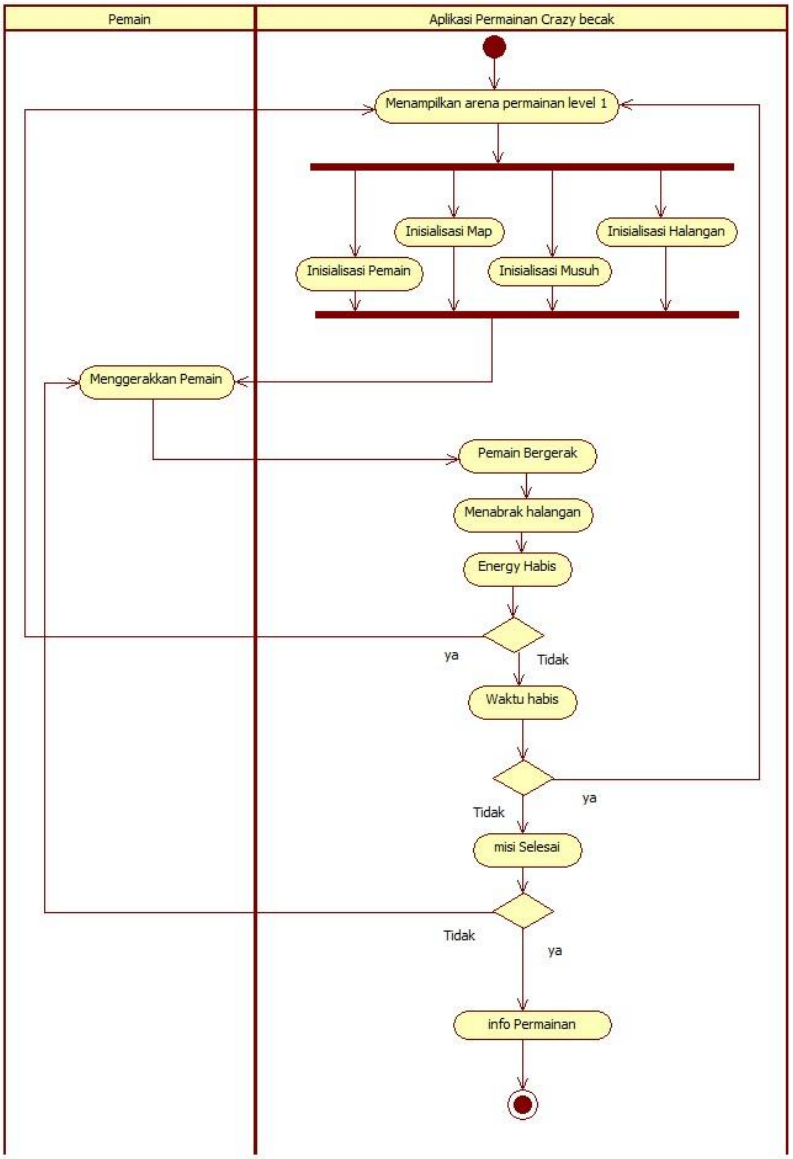
Gambar 3.22 Diagram Tampilan Tentang *Crazy* Becak

3.4.2.4. Activity Diagram Level 1

Adapun penjelasan tentang *level* 1 yang terdapat pada aplikasi permainan *Crazy* Becak adalah sebagai berikut :

Tabel 3.13 Penjelasan *Activity* Diagram*Level* 1

Deskripsi	Proses ini terjadi saat pemain memilih *level* 1 pada tampilan pilihan level., apabila misi tidak terselesaikan pada *level* 1 maka pemain kembali untuk memulai permainan *level* 1 lagi.
Normal Flow	1. Menampilkan arena permainan level 1 2. Aplikasi menginisialisasi entitas permainan 3. Pemain memulai permainan pada level 1



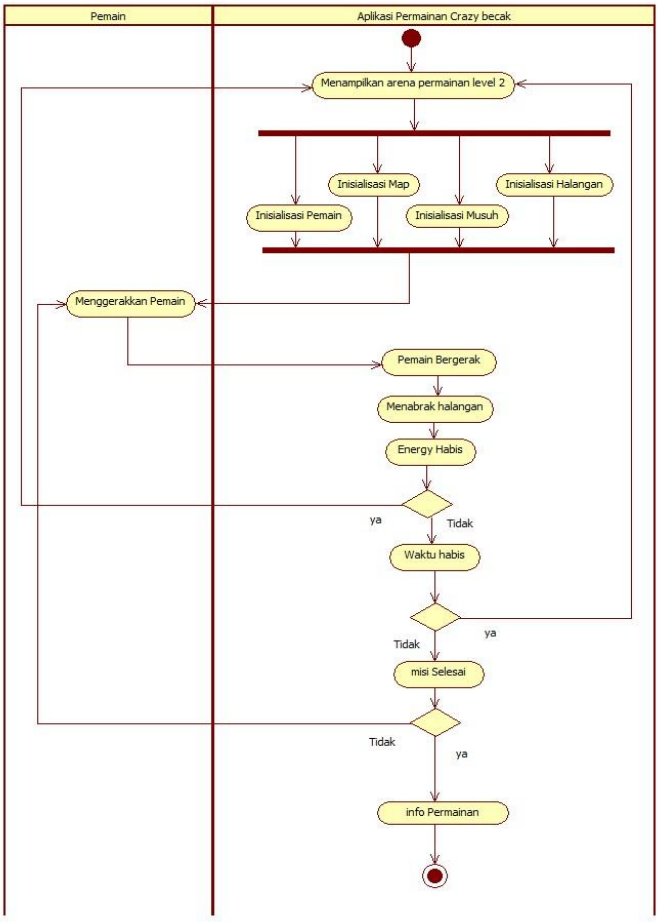
Gambar 3.23 Diagram *Activity Level* 1

3.4.2.3. Activity *Diagram* Level *2*

Adapun penjelasan tentang keluar yang terdapat pada aplikasi permainan *Crazy* Becak adalah sebagai berikut :

Tabel 3.14 Penjelasan *Activity Diagram Level* 2

Deskripsi	Proses ini terjadi saat pemain memilih menu tampilan pilihan *level* pada aplikasi permainan *Crazy* Becak., dan pemain sudah menyelesaikan misi *level* 1. apabila misi tidak terselesaikan pada *level* 2 maka pemain kembali ke menu tampilan *level* untuk memulai permainan *level* 2 lagi.
Normal Flow	1. Menampilkan arena permainan level 2 2. Aplikasi menginisialisasi entitas permainan 3. Pemain memulai permainan pada level 2



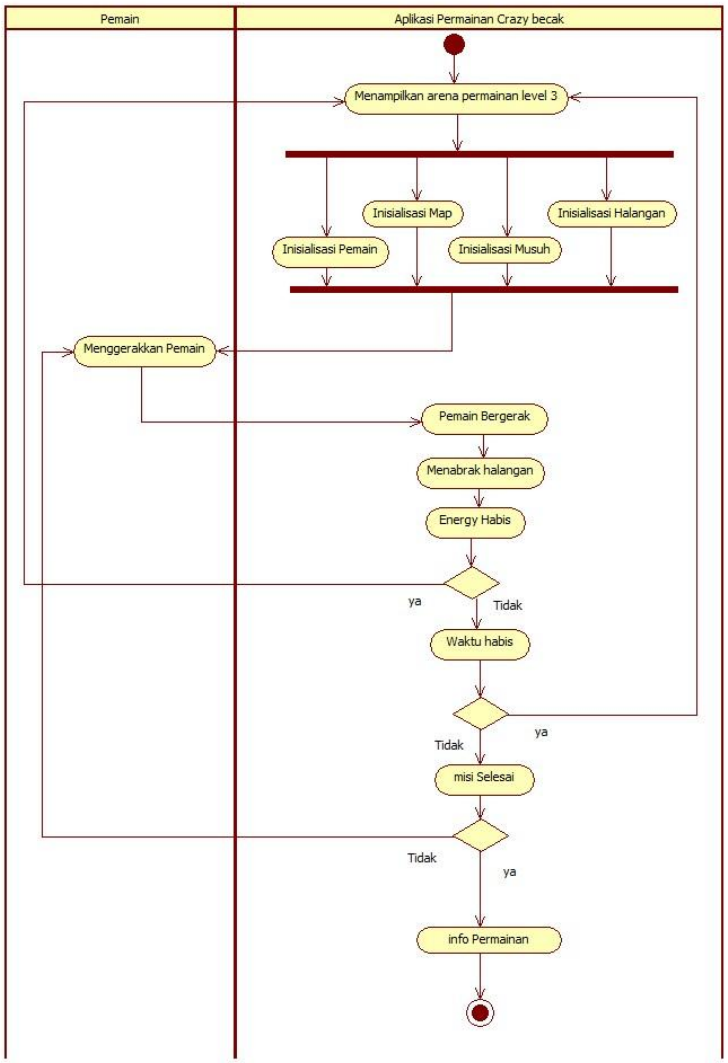
Gambar 3.24 Diagram *Activity Level* 2 dari aplikasi *game Crazy* Becak

3.4.2.3. Activity *Diagram* Level *3*

Adapun penjelasan tentang keluar yang terdapat pada aplikasi permainan *Crazy* Becak adalah sebagai berikut :

Tabel 3.15 Penjelasan *Activity* Diagram *Level* 3

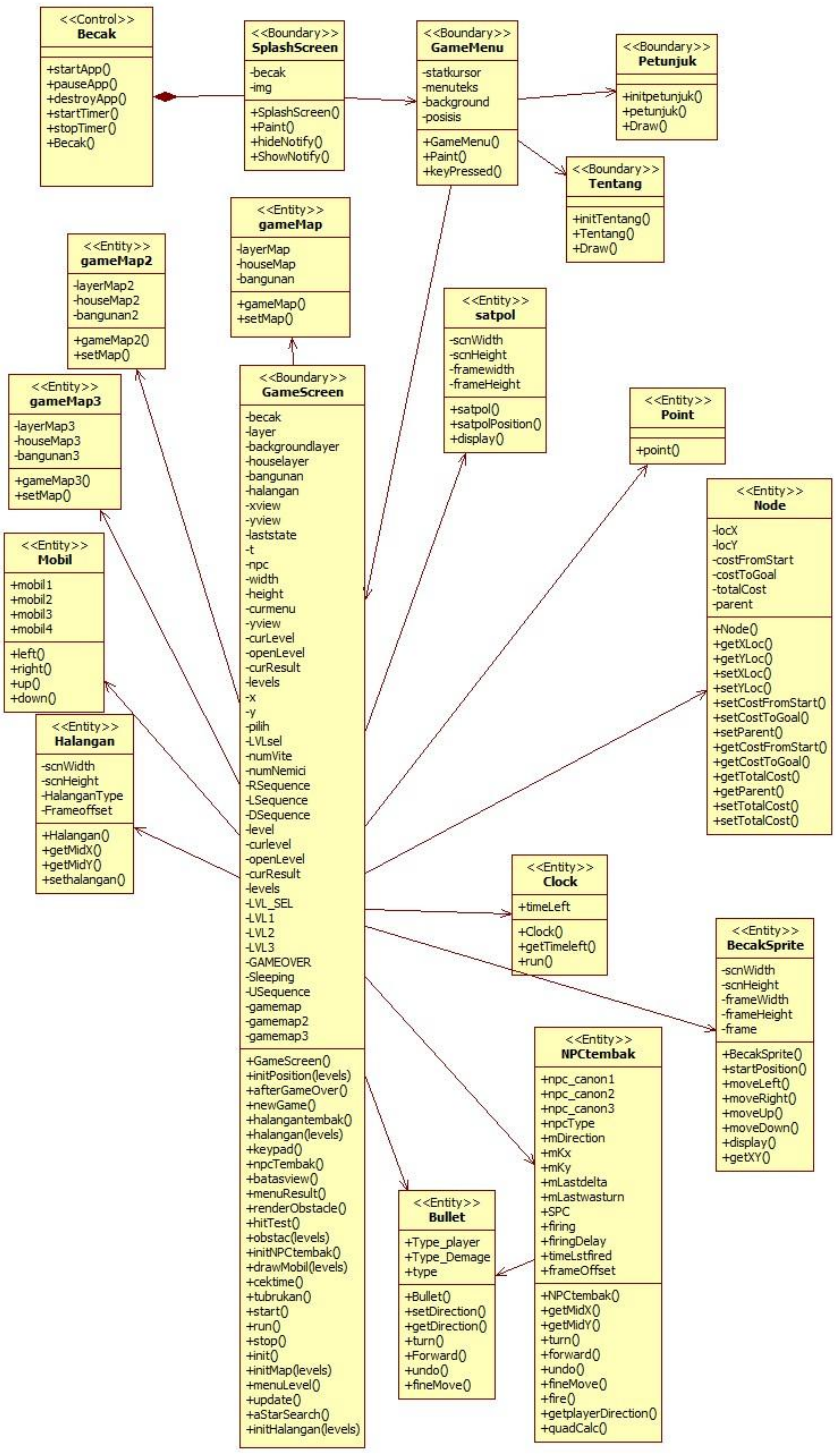
Deskripsi	Proses ini terjadi saat pemain memilih menu tampilan pilihan *level* pada aplikasi permainan *Crazy* Becak., dan pemain sudah menyelesaikan misi *level* 2. apabila misi tidak terselesaikan pada *level* 3 maka pemain kembali ke menu tampilan *level* untuk memulai permainan *level* 3 lagi.
Normal Flow	1. Menampilkan arena permainan level 2 2. Aplikasi menginisialisasi entitas permainan 3. Pemain memulai permainan pada level 2



Gambar 3.25 Diagram *Activity Level* 3 dari aplikasi *game Crazy* Becak

3.4.3. *Class* Diagram

Class Diagram pada aplikasi *game Crazy* Becak adalah sebagai berikut :



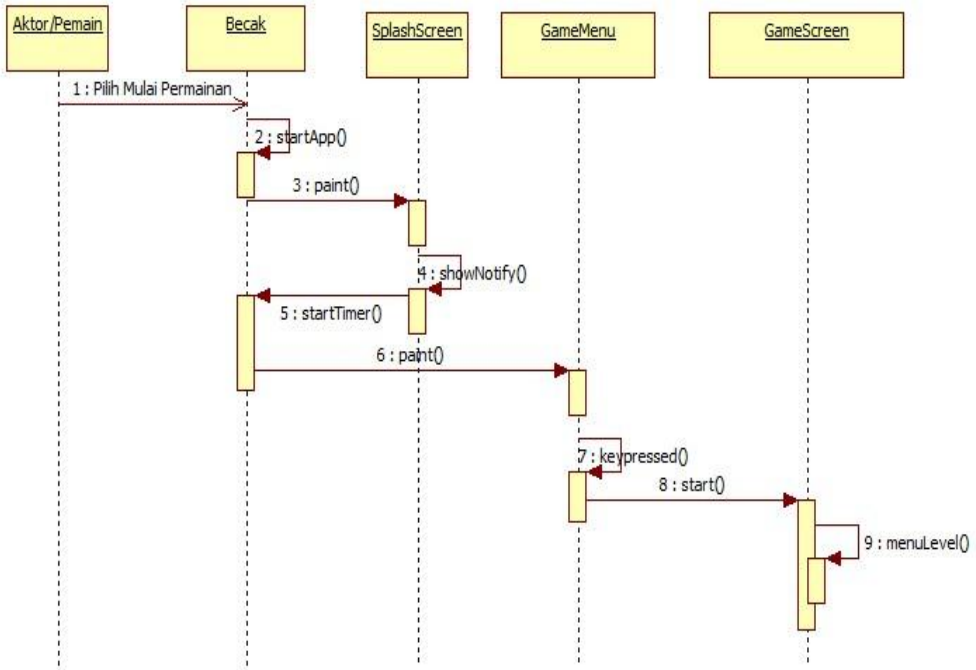
Gambar 3.26 Diagram *Class game Crazy* Becak

3.4.4. Diagram *Sequence*

Diagram *Sequence* mengambarkan apa yang dilakukan objek pada *use case* dengan mendeskripsikan waktu hidup dan message yang dikirimkan dan diterima antarobjek.

3.4.4.1. Diagram *sequence* Tampil *Level*

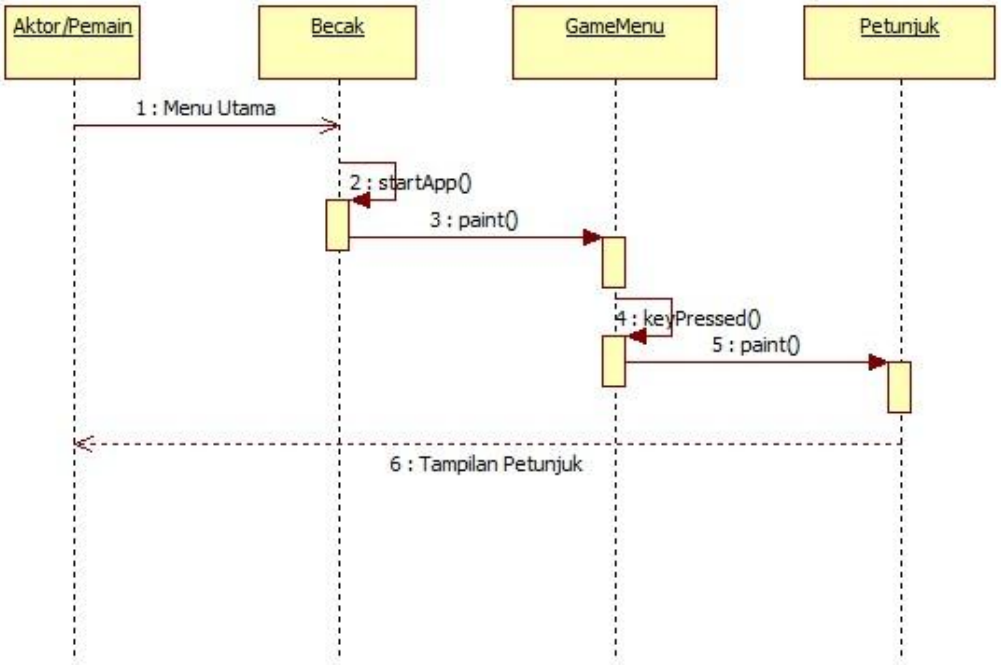
Diagram *sequence* tampil level terdapat pada gambar 3.27



Gambar 3.27 Diagram *sequence* tampil *level*

3.4.4.2. Diagram *sequence* Tampil Petunjuk

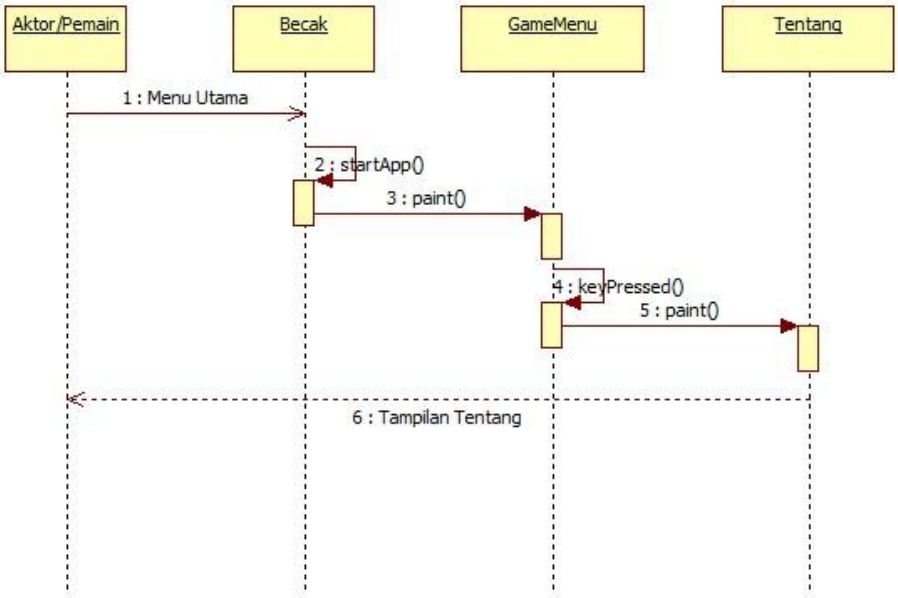
Diagram *sequence* tampil petunjuk terdapat pada gambar 3.28



Gambar 3.28 Diagram *sequence* tampil petunjuk

3.4.4.3. Diagram *sequence* Tampil Tentang

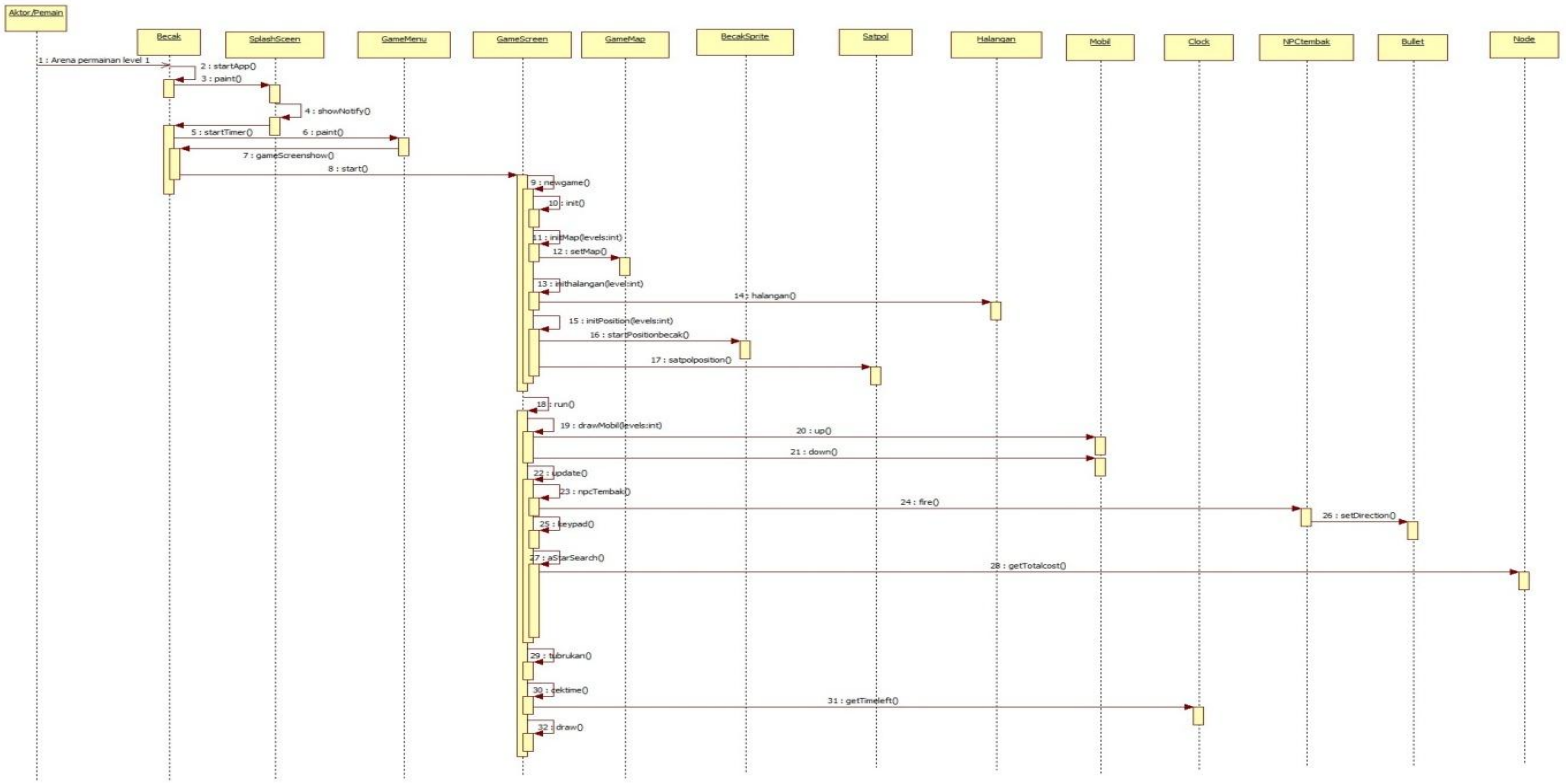
Diagram *Sequence* tampil tentang terdapat pada gambar 3.29



Gambar 3.29 Diagram *sequence* tampil tentang

3.4.4.4. Diagram *sequence level* 1

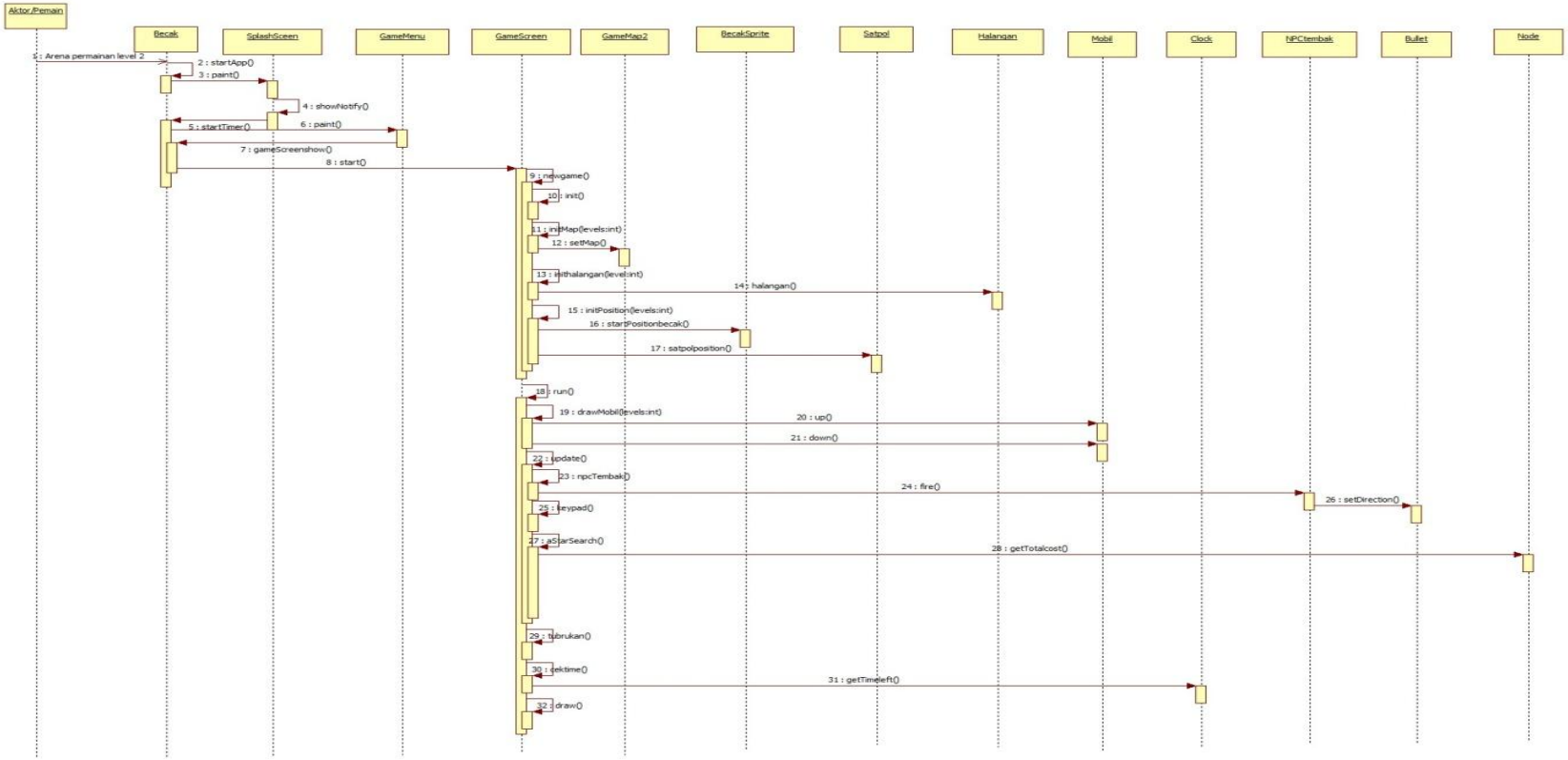
Diagram *Sequence* level 1 terdapat pada gambar 3.30



Gambar 3.30 Diagram *sequence level* 1.

3.4.4.5. Diagram *sequence level* 2

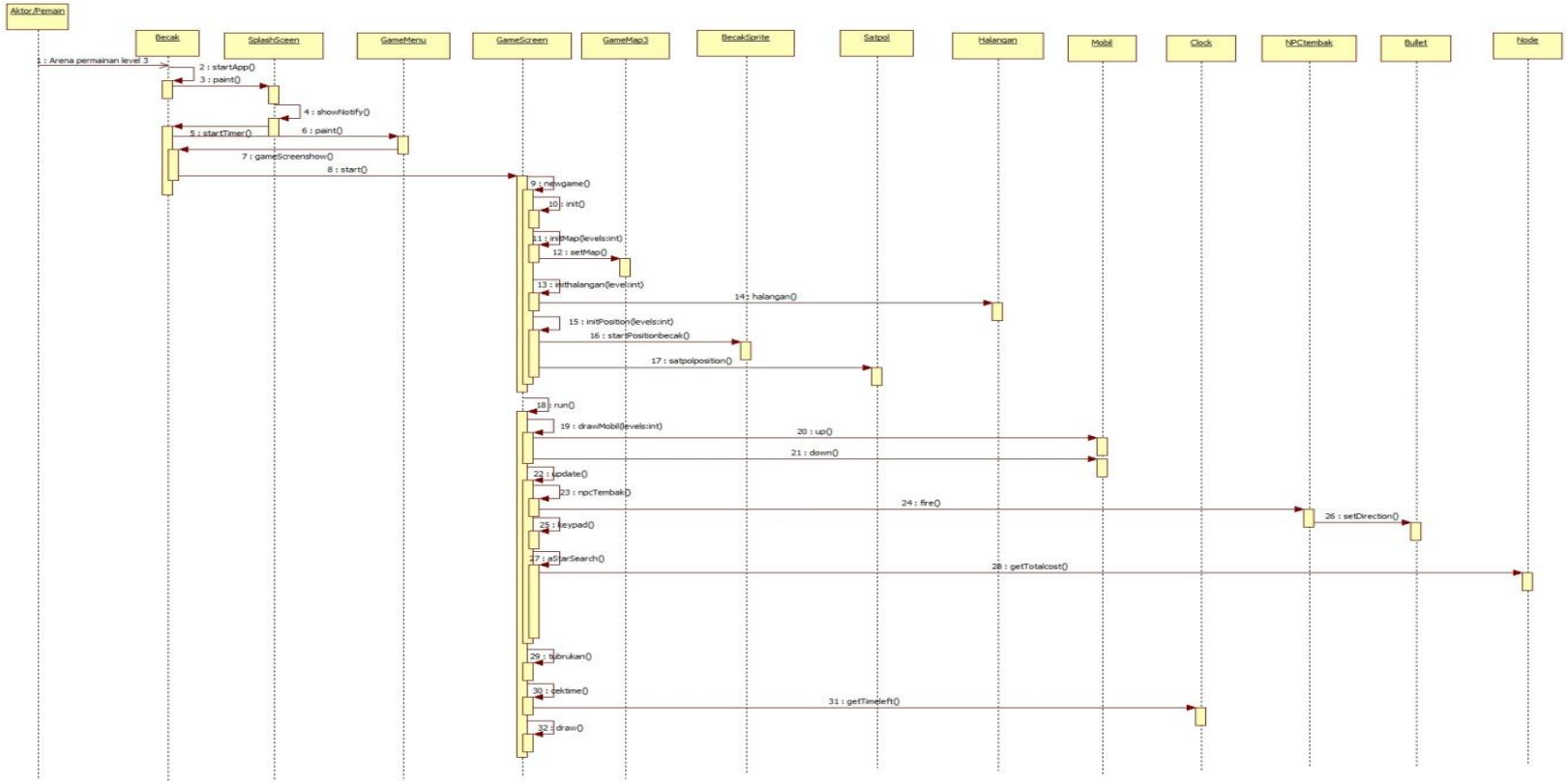
Diagram *Sequence* level 2 terdapat pada gambar 3.31



Gambar 3.31 Diagram *sequence level* 2

3.4.4.6. Diagram *sequence level* 3

Diagram *Sequence* level 3 terdapat pada gambar 3.32



Gambar 3.32 Diagram *sequence level* 3

3.5. Perancangan Perangkat Lunak

Perancangan perangkat lunak adalah kegiatan untuk merancang aplikasi yang akan kita bangun, tahapannya dimulai dari perancangan arsitektur menu, lalu berlanjut ke perancangan antar muka dan jaringan semantik.

3.5.1. Storyline *Game*

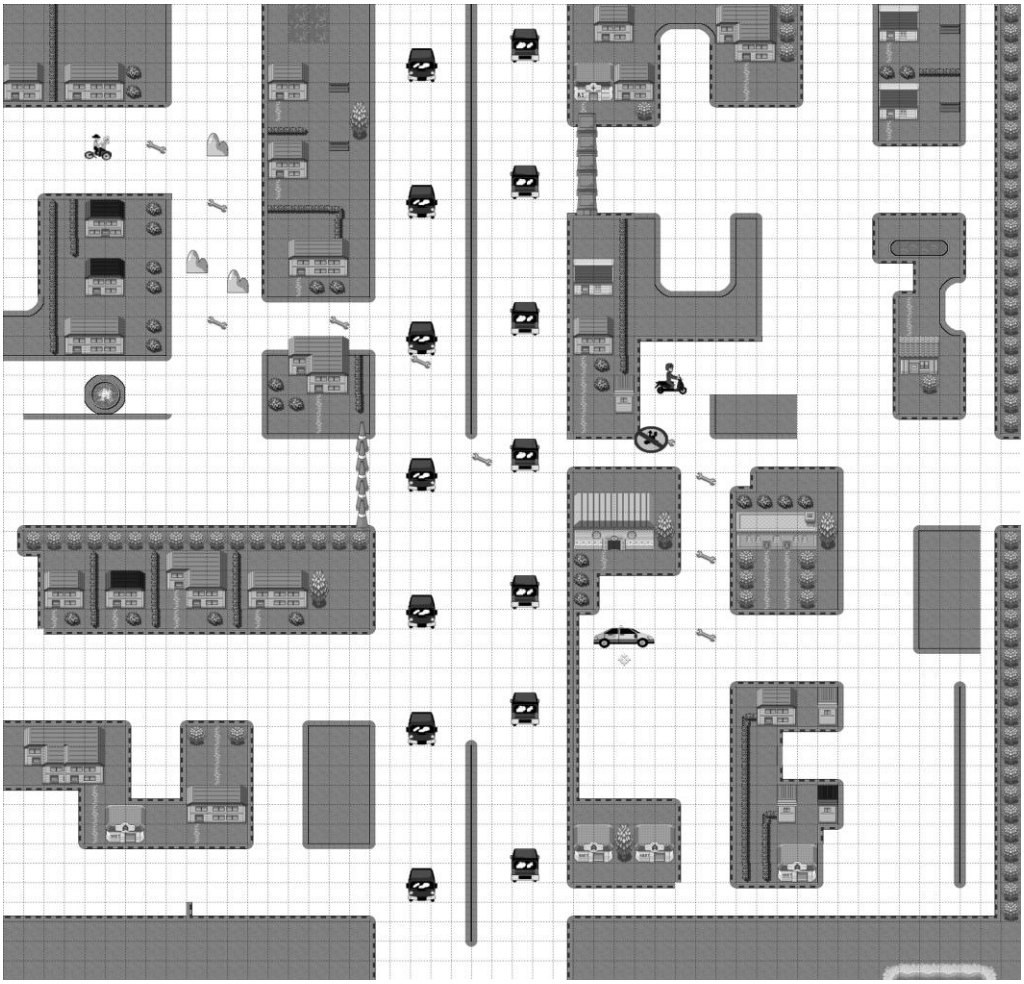
Suatu hari ada seorang bule atau orang luar negeri yang berkunjung ke Indonesia untuk tugas tertentu, tepatnya ada di daerah kota Bandung. Bule itu bernama Mr. Jhon, Ia harus mengikuti sebuah rapat di suatu tempat. Dia terlambat untuk menghadiri rapat tersebut, karena terburu-buru maka Mr. Jhon menggunakan taksi agar lebih cepat.

Dalam perjalanan ia merasa harus buru-buru untuk menghadiri rapat, tapi lama kemudian ia merasa tenang di dalam mobil, saking tenangnya ia tertidur didalam taksi tersebut. Sampailah Ia ke tempat rapat tersebut, dengan terburu-buru Ia turun dan membayar taksi tersebut. Tetapi apa yang terjadi, ia lupa tasnya masih di bagasi taksi yang dia naiki. Ia pun lemas sambil bingung apa yang harus dilakukan. Datanglah seorang tukang becak yang menanyakan keadaan dari bule tersebut. Tukang becak itu pun mau membantu, karena dia tau dimana biasanya taksi itu berpangkalan. Tapi bule itu ragu apakah tukang becak itu bisa membantunya dengan cepat.

Dimulailah *Game* yang sangat menarik untuk dimainkan, karena dengan waktu yang terbatas ia harus bisa sampai ke tempat dimana biasanya taksi tersebut berpangkalan.

3.5.2. Storyboard

Storyboard merupakan sketsa gambar yang disusun berurutan sesuai naskah yang ada, dan dengan *storyboard* kita dapat menyampaikan ide cerita kita kepada orang lain dengan mudah. Berikut merupakan *Storyboard* dari *game Crazy* Becak :

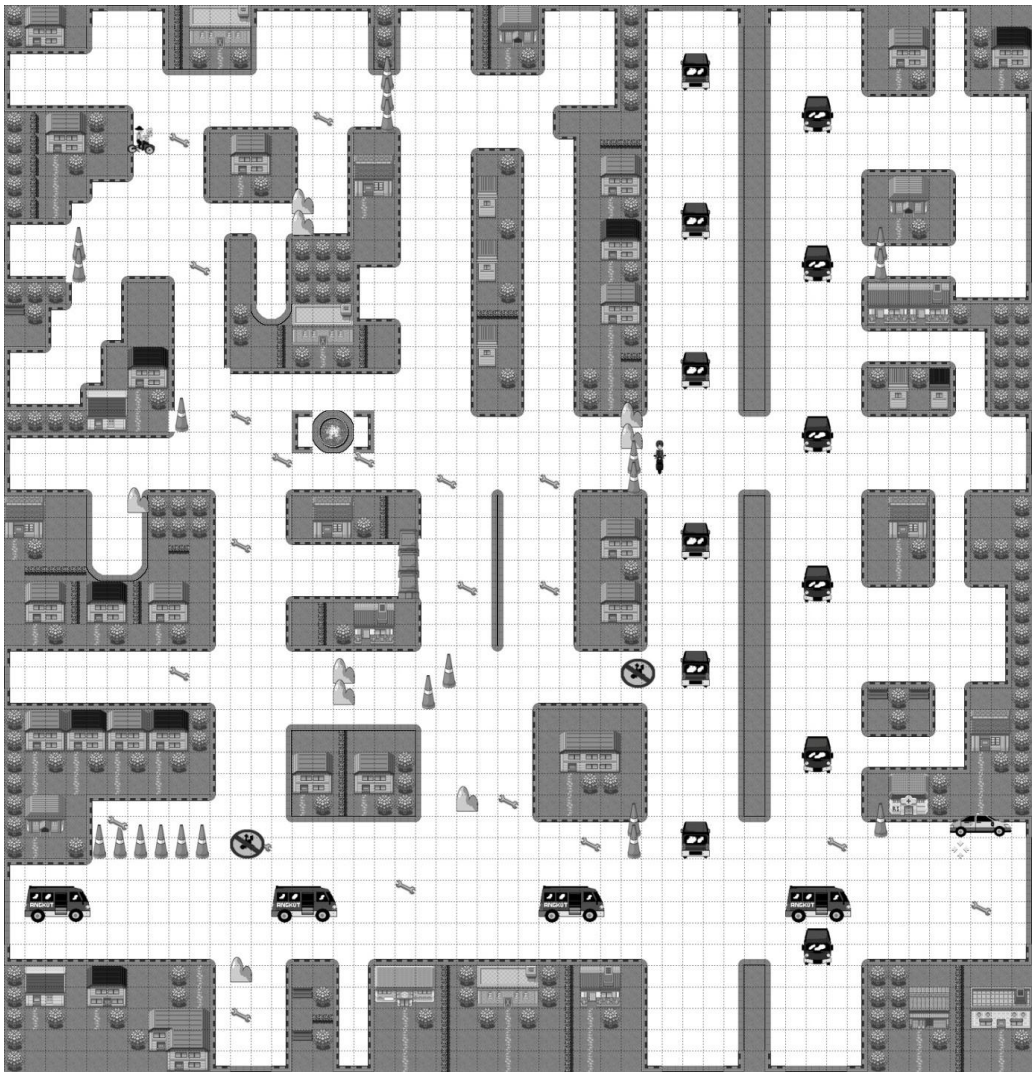


Gambar 3.33 *Storyboard level* 1

Pada *level* 1 ini bernama West Bandung Street, Tempat dari *level* ini ukurannya 1500x1500 Px dan halangan pada level ini ada 13 halangan dan ada 10 kunci. Setiap entitas *sprite* ditempatkan secara statis.

1. Posisi awal Karakter utama berada pada koordinat (85,180).
2. Posisi awal satpol berada pada koordinat (960,570).
3. Posisi Tujuan akhir yaitu dekat dengan objek taksi berada pada koordinat (900,990). Petunjuk untuk menuju tujuan akhir adalah dengan mengambil semua kunci.

4. Ketika Karakter utama berada pada posisi X lebih dari 930 maka pengejaran becak oleh satpol dimulai, ketika kembali ke posisi X kurang dari 930 maka pengejaran becak oleh satpol berhenti.



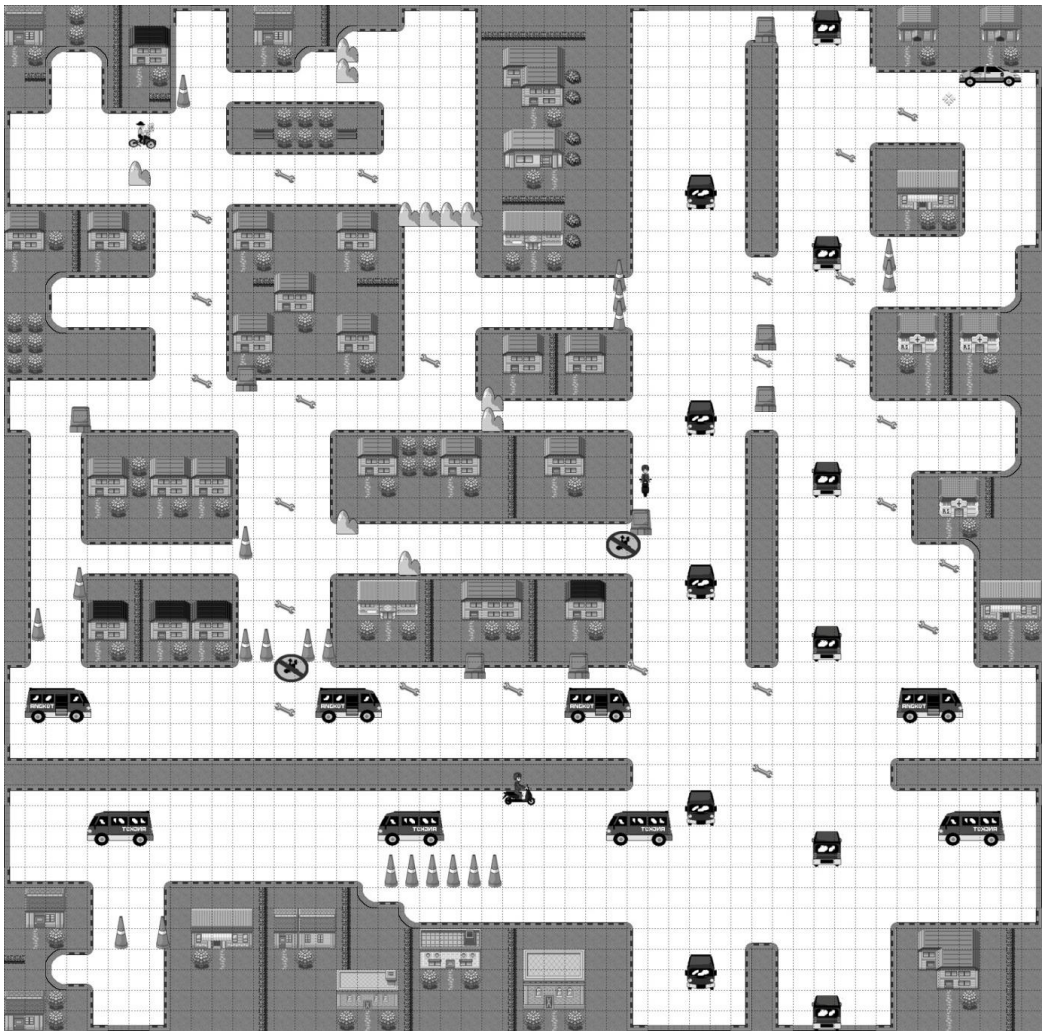
Gambar 3.34 *Storyboard Level* 2

Pada *level* 2 ini bernama *South* Bandung Street, Tempat dari *level* ini ukurannya 1500x1500 Px dan halangan pada level 2 ini ada 33 halangan dan ada 20 kunci . Setiap entitas *sprite* ditempatkan secara statis.

1. Posisi awal Karakter utama berada pada koordinat (180,180).
2. Posisi awal satpol berada pada koordinat (960,630).

3. Posisi Tujuan akhir yaitu dekat dengan objek taksi berada pada koordinat (1410,1170). Petunjuk untuk menuju tujuan akhir adalah dengan mengambil semua kunci.

4. Ketika Karakter utama berada pada posisi X lebih dari 930 maka pengejaran becak oleh satpol dimulai, ketika kembali ke posisi X kurang dari 930 maka pengejaran becak oleh satpol berhenti. Ketika Karakter utama berada pada posisi Y lebih dari 930 maka pengejaran becak oleh satpol dimulai, ketika kembali ke posisi Y kurang dari 930 maka pengejaran becak oleh satpol berhenti.



Gambar 3.35 *Storyboard Level* 3

Pada *level* 3 ini bernama *North* Bandung Street, Tempat dari *level* ini masih daerah ukurannya 1500x1500 Px dan halangan pada level 3 ini ada 40 halangan dan 25 kunci. Setiap entitas *sprite* ditempatkan secara statis yag artinya penempatan sudah ditetapkan. Unutk level 3 satpol yang mengejar ada 2 satpol.

1. Posisi awal Karakter utama berada pada koordinat (85,180).
2. Posisi awal satpol 1 berada pada koordinat (900,690), satpol 2 berada pada koordinat (720,1140)
3. Posisi Tujuan akhir yaitu dekat dengan objek taksi berada pada koordinat (1350,120). Petunjuk untuk menuju tujuan akhir adalah dengan mengambil semua kunci.
4. Ketika Karakter utama berada pada posisi X lebih dari 870 maka pengejaran becak oleh satpol dimulai, ketika kembali ke posisi X kurang dari 870 maka pengejaran becak oleh satpol berhenti. Ketika Karakter utama berada pada posisi Y lebih dari 930 maka pengejaran becak oleh satpol dimulai, ketika kembali ke posisi Y kurang dari 930 maka pengejaran becak oleh satpol berhenti.

Adapun bentuk algoritma ketika Algoritma A* aktif, kondisi tersebut bisa dilihat pada tabel 3.

Tabel 3.16 *Pseudocode* kondisi A* aktif

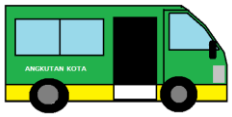
```
   Case Level OF
      1 : if posisibecakX >=930
           begin
                 Algoritma A* Aktif
            EndIf
      2 :  if posisibecakX >=930 or posisibecakY >=1170
          begin
               Algoritma A* Aktif
          EndIf
      3 :  if posisibecakX >=870 or posisibecakY >=930
          begin
               Algoritma A* Aktif
          EndIf
   EndCase
```

3.5.3. Entitas *Game*

Berikut adalah entitas karakter dan halangan yang ada pada *game Crazy* Becak ini :

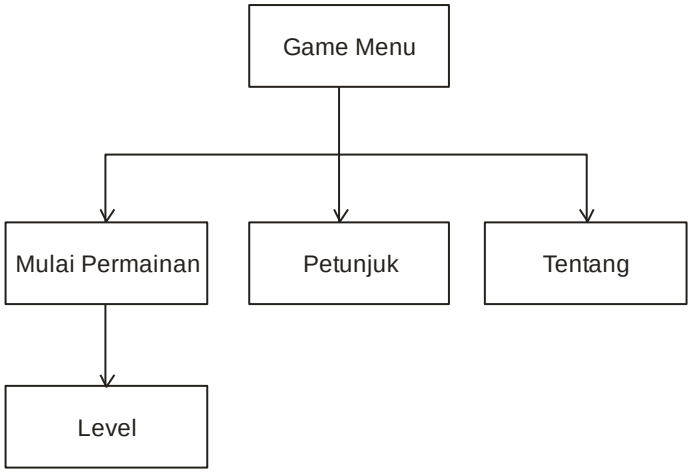
Tabel 3.17 Entitas karakter dan item.

No.	Nama	Gambar	Penjelasan
1	Mang Jana dan Mr. Jhon		Karakter ini merupakan karakter utama pada *game* ini, Karakter ini mempunyai *lifebar* untuk bisa melanjutkan permainan.
2	Satpol PP.		Karakter ini merupakan karakter satpol yang akan mengejar becak dalam *game* ini
3	*Wreck* atau Kunci		Item ini yang harus dikumpulkan oleh pemain.
4	Batu		Sebagai halangan pada setiap *level*, penghalang ini mengurangi *life* dari karakter utama bila menabraknya.
5	Pembatas jalan		mengurangi *life* dari karakter utama bila menabraknya dan memberhentikan jalannya becak.
6	Papan pembatas jalan		Penghalang ini mengurangi *life* dari karakter utama bila menabraknya

7	Mobil Angkot		Penghalang ini mengurangi *life* dari karakter utama bila menabraknya dan memberhentikan jalannya becak.

3.5.4. Perancangan Arsitektur Menu

Untuk mempermudah pembuatan 99emant diperlukan perancangan struktur menu program yang akan dibangun. Perancangan struktur menu program ini membantu dalam merancang bagian-bagian dari 99emant 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 99emant yang akan dibangun:



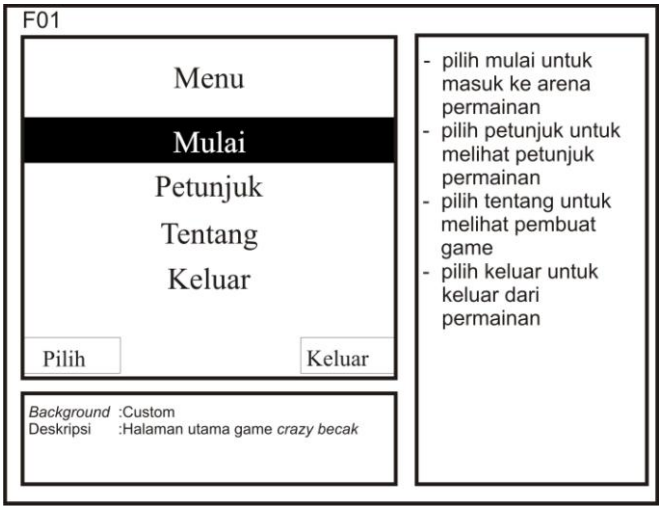
Gambar 3.36 Arsitektur Menu *game Crazy* becak

1. Mulai Permainan : merupakan submenu untuk memulai permainan baru
2. Lihat Petunjuk : merupakan submenu yang menampilkan informasi seputar petunjuk permainan seperti penekanan tombol tertentu pada *keypad* beserta reaksinya

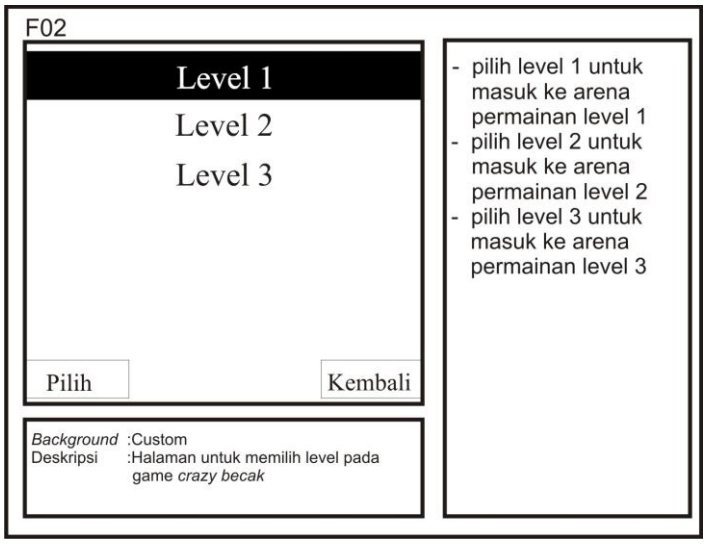
3. Lihat Tentang : merupakan submenu yang menampilkan informasi mengenai pengembang aplikasi beserta informasi lain seputar *game Crazy* Becak.
4. *Level* : merupakan submenu yang menampilkan pilihan *level* yang akan dimainkan.

3.5.5. Perancangan Antar Muka

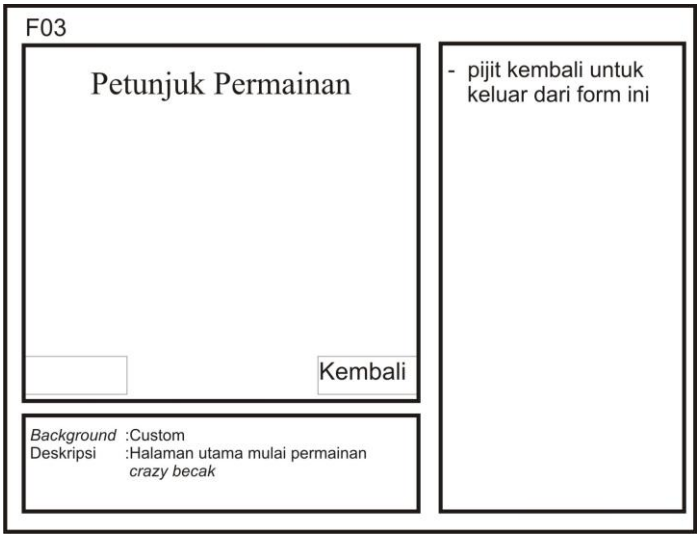
Perancangan antar muka pada *game Crazy* Becak adalah sebagai berikut :



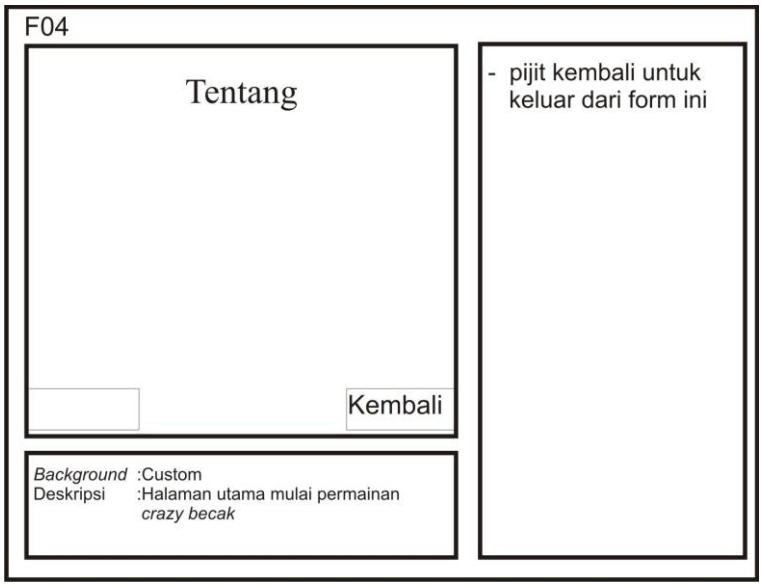
Gambar 3.37 Antar muka menu pada *game Crazy* becak



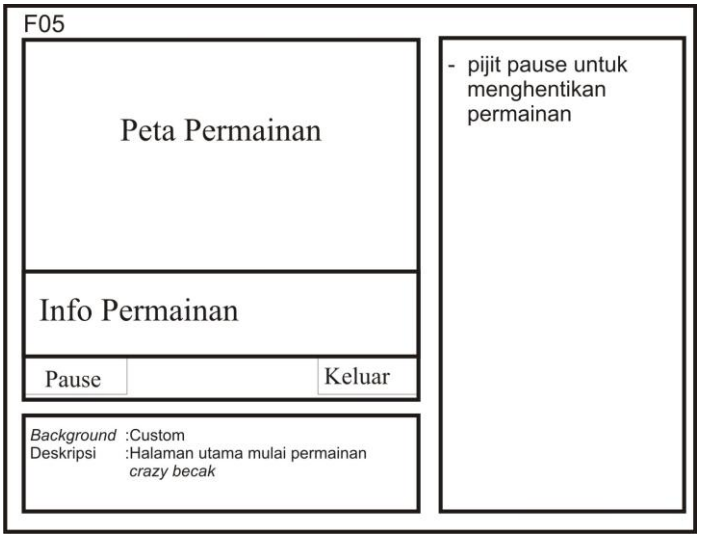
Gambar 3.38 Antar muka pilih *level* pada *game Crazy* becak



Gambar 3.39 Antar muka petunjuk permainan pada *game Crazy* becak



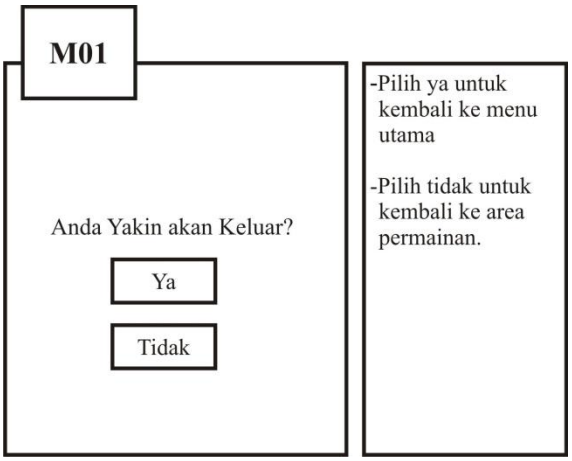
Gambar 3.40 Antar muka Tentang pada *game Crazy* becak



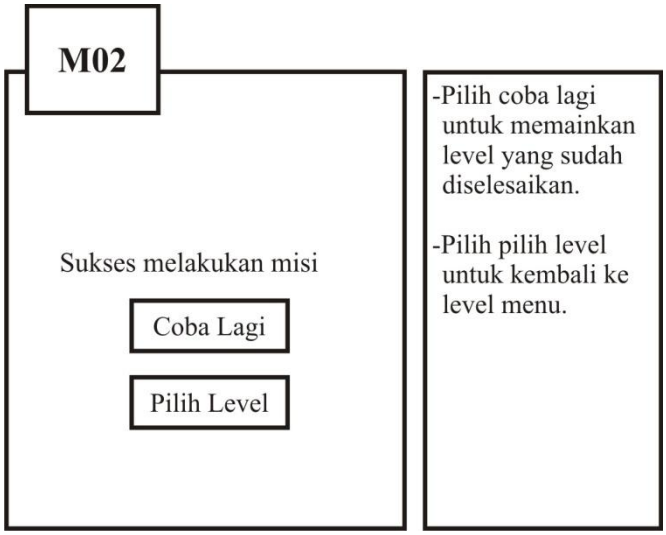
Gambar 3.41 Antar muka Permainan pada *game Crazy* becak

3.5.6. Perancangan Pesan

Pesan yang dirancang dalam konteks aplikasi yang dikembangkan merupakan pesan peringatan untuk keluar dari arena permainan untuk kembali ke manu utama dan setelah misi selesai apakah ingin bermain di *level* yang sama atau memilih *level*.



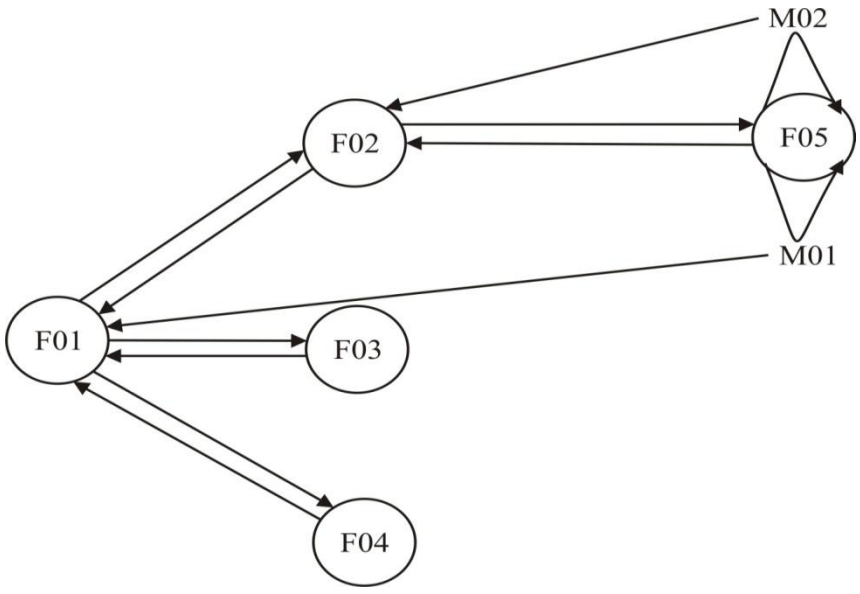
Gambar 3.42 Perancangan pesan peringatan untuk kembali ke menu utama.



Gambar 3.43 Perancangan pesan peringatan setelah berhasil melakukan misi.

3.5.7. Jaringan Semantik

Jaringan semantik adalah menggambarkan hubungan antar menu. Penggambaran jaringan semantik pada aplikasi ini adalah sebagai berikut:



Gambar 3.44 Jaringan semantik pada *game Crazy* becak

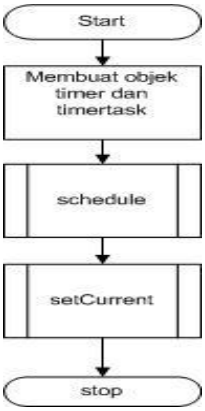
3.5.8. Perancangan Method

Perancangan komponen method merupakan perancangan yang dibuat setelah perancangan arsitektur menu, antarmuka, pesan, dan jaringan semantik. Perancangan

ini berfungsi untuk mendeskripsikan method-method yang berada di dalam aplikasi permainan *Crazy* Becak. Adapun method-method yang terdapat dalam aplikasi permainan *Crazy* Becak yang akan dibangun adalah sebagai berikut :

A. Method startTimer()

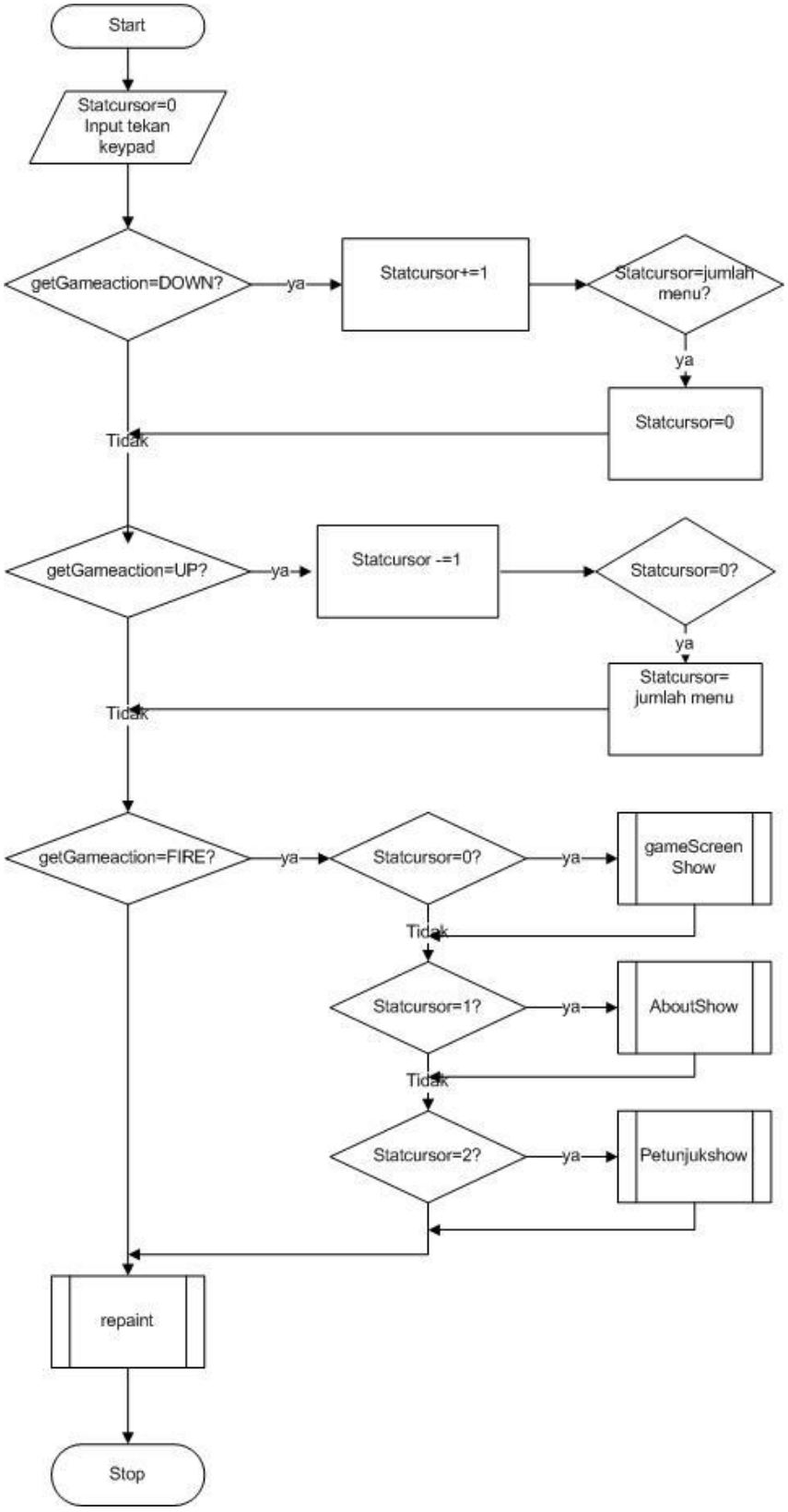
Method startTimer akan dipanggil ketika akan menampilkan menu dalam permainan.



Gambar 3.45 Flowchart Method startTimer()

B. Method keyPressed()

Method keyPressed akan dipanggil ketika akan pemain berinteraksi dengan tombol pada device untuk memilih menu pada menu utama.



Gambar 3.46 Flowchart Method keyPressed()

C. Method start()

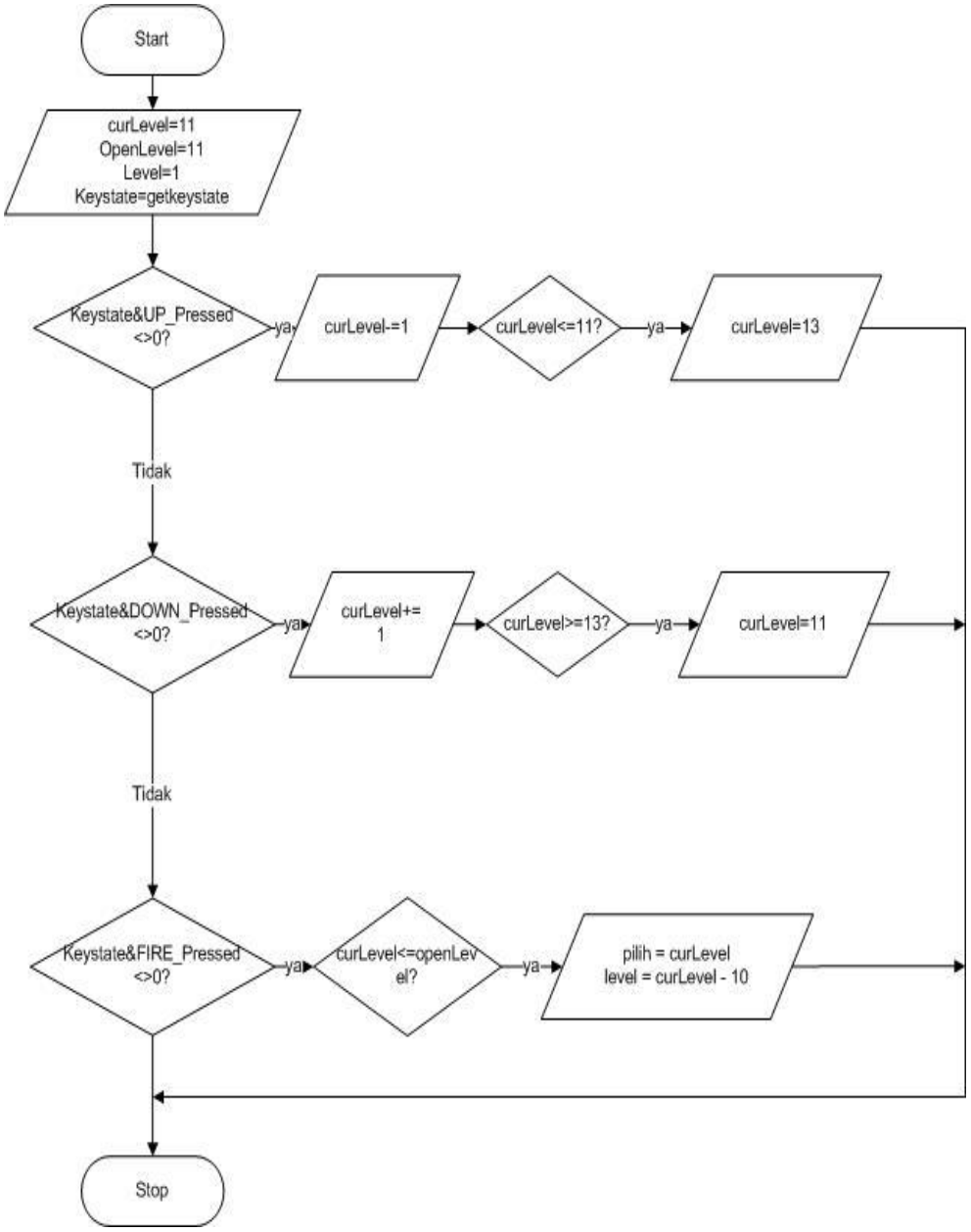
Method start akan dipanggil untuk menampilkan canvas *game*.



Gambar 3.47 Flowchart Method start()

D. Method Menu*Level*()

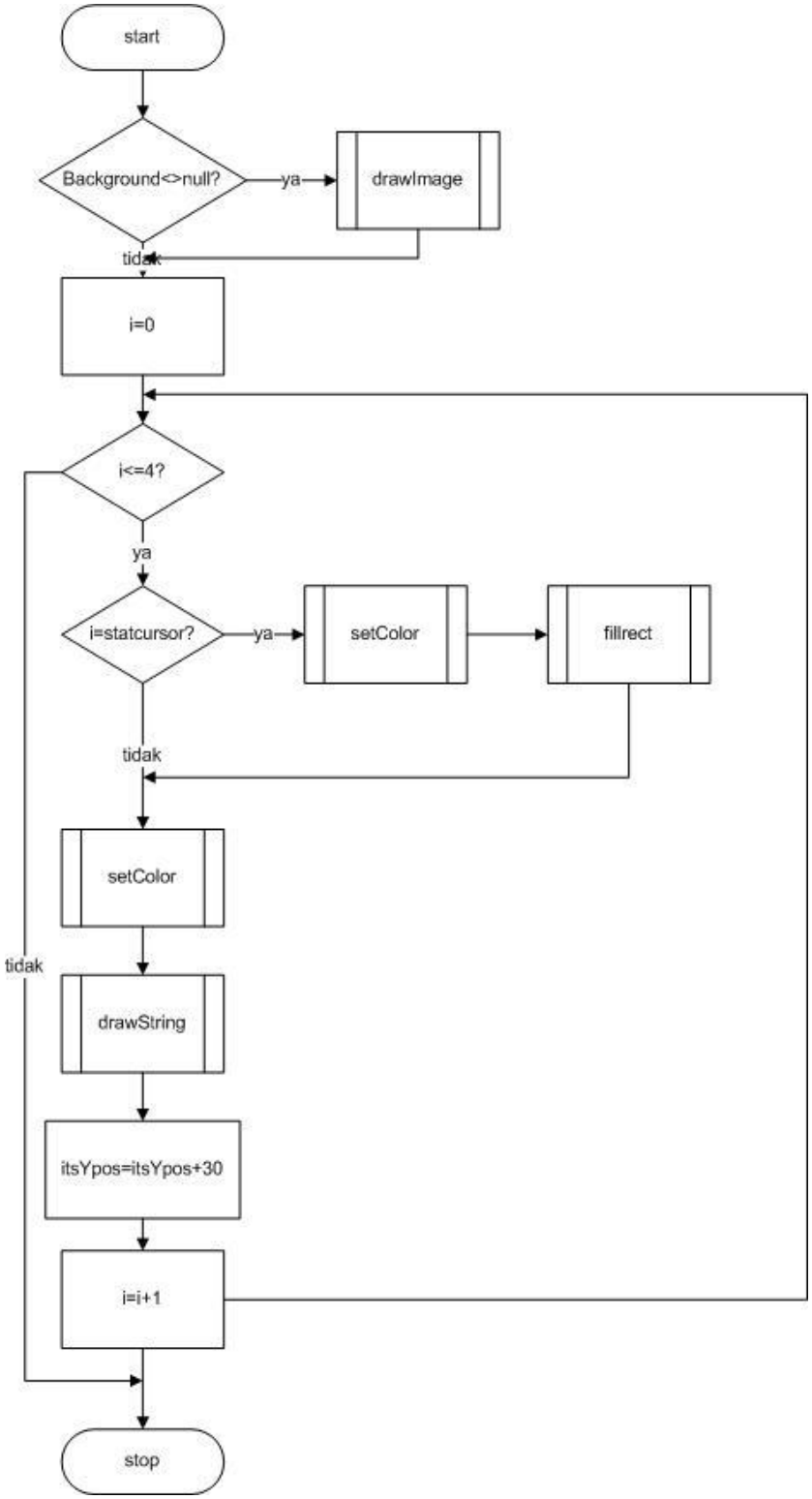
Method Menu*Level* akan dipanggil ketika akan menampilkan menu pilihan *level*.



Gambar 3.48 Flowchart Method menuLevel()

E.Method paint()

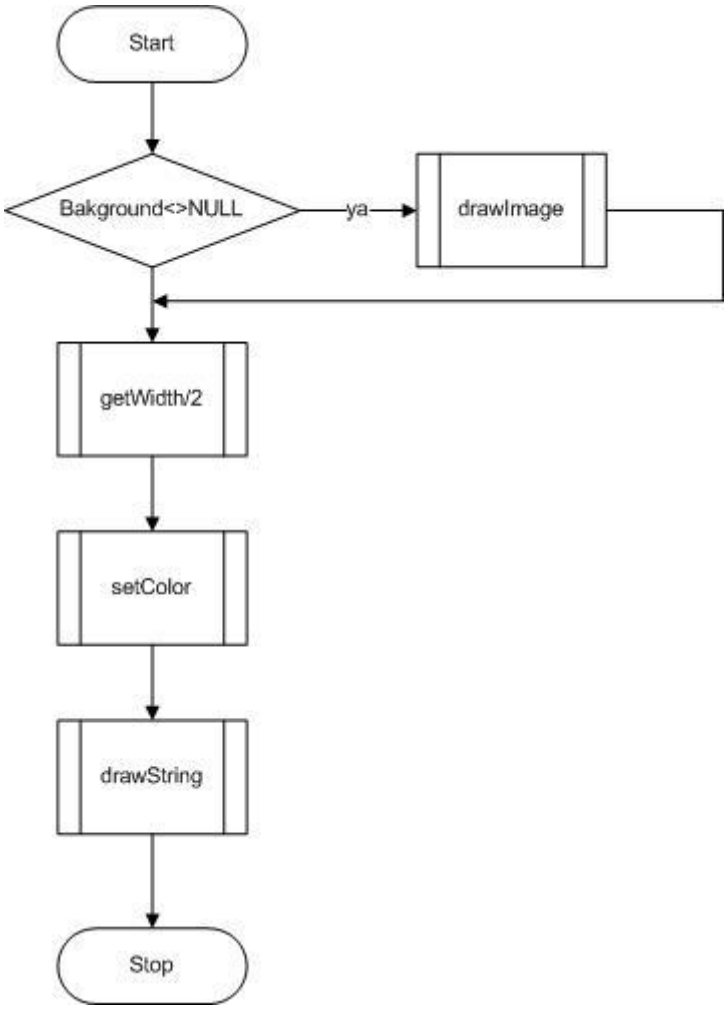
Method paint akan dipanggil untuk menampilkan menu utama dalam permainan.



Gambar 3.49 Flowchart Method paint() pada menu utama

F.Method paint()

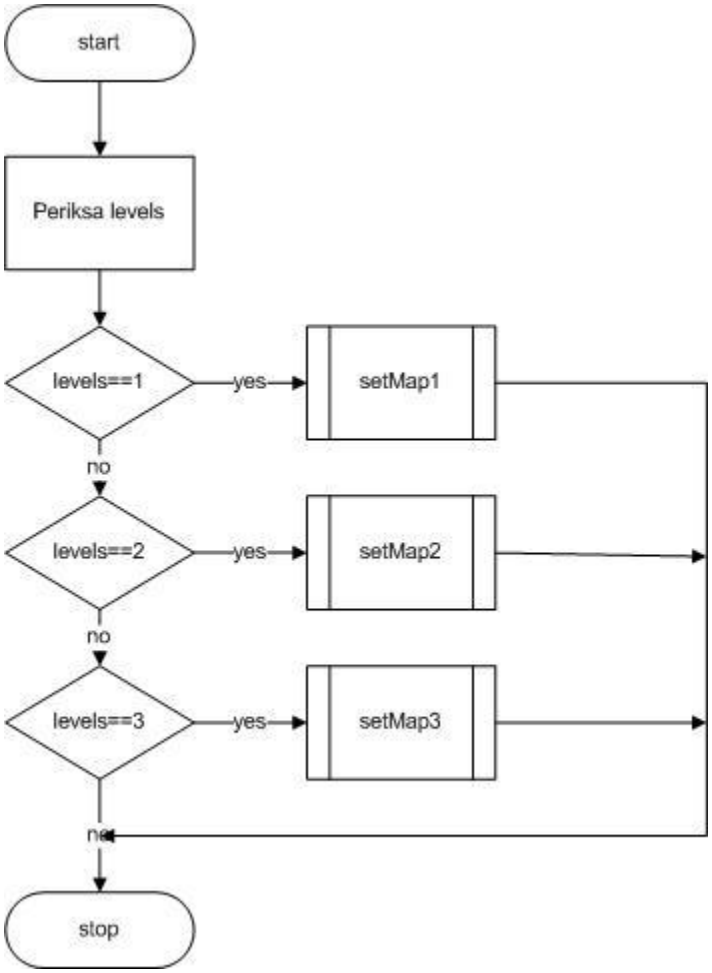
Method paint disini akan dipanggil untuk menampilkan tampilan tentang dalam permainan.



Gambar 3.50 Flowchart Method paint() pada kelas tentang

G. Method initMap()

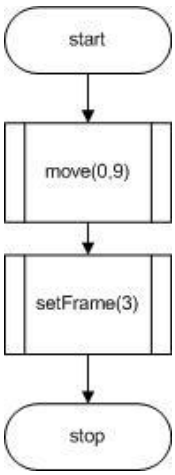
Method initMap akan dipanggil untuk Menginisialisasi Map *game* dalam permainan *Crazy* Becak.



Gambar 3.51 Flowchart Method initMap()

H. Method down()

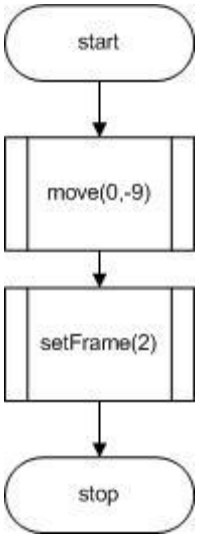
Method down akan dipanggil untuk ketika akan menggerakkan mobil ke bawah.



Gambar 3.52 Flowchart Method down()

I. Method up()

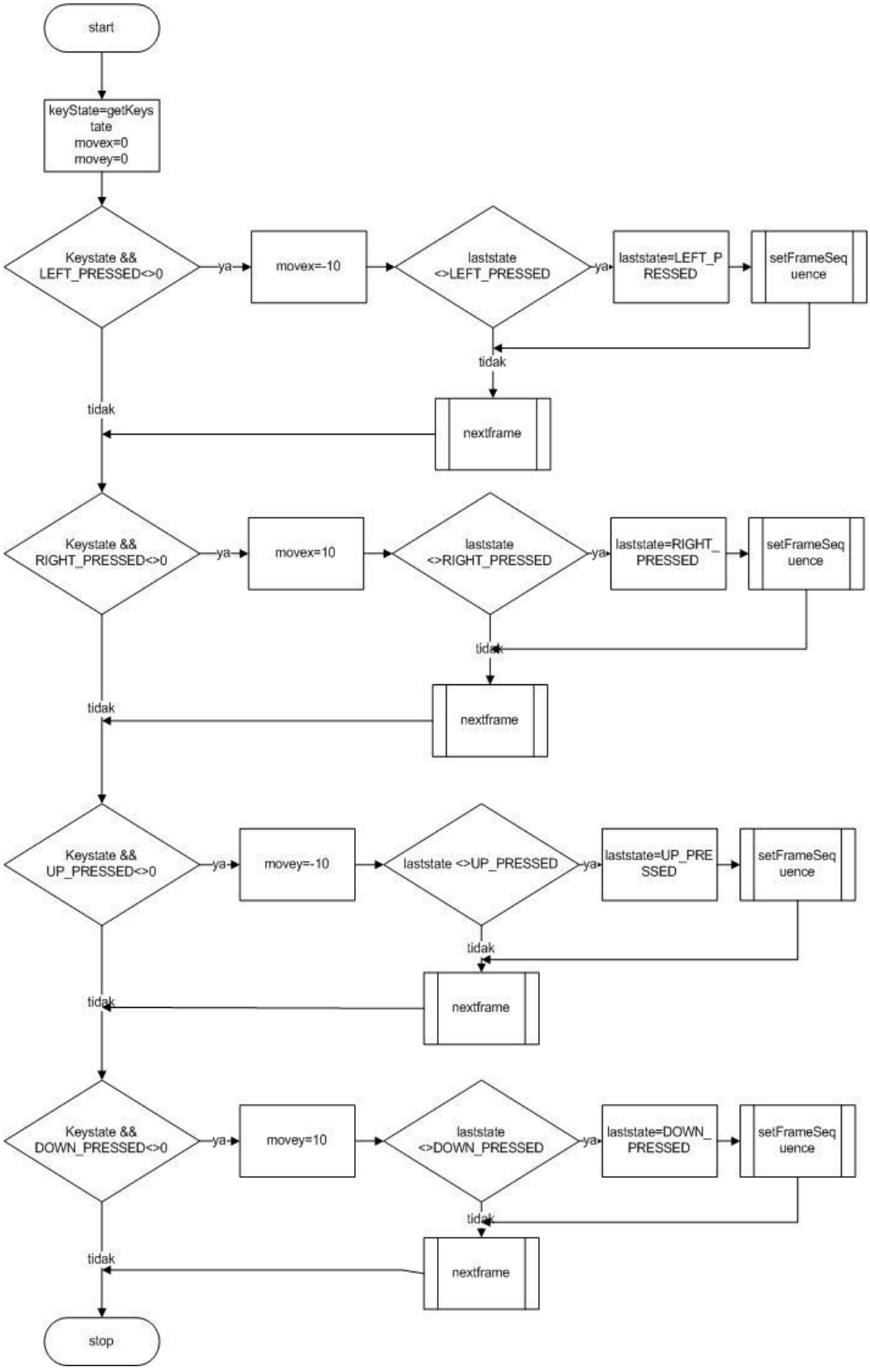
Method up akan dipanggil untuk menggerakkan mobill ke atas.



Gambar 3.53 Flowchart Method up()

J. Method keypad()

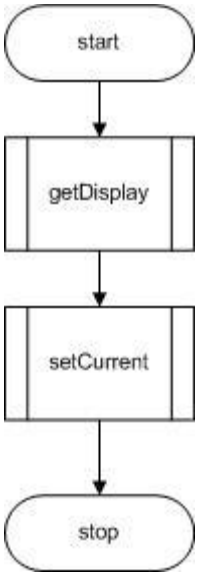
Method keypad akan dipanggil untuk menggerakkan pemain.



Gambar 3.54 Flowchart Method keypad()

K. Method startApp()

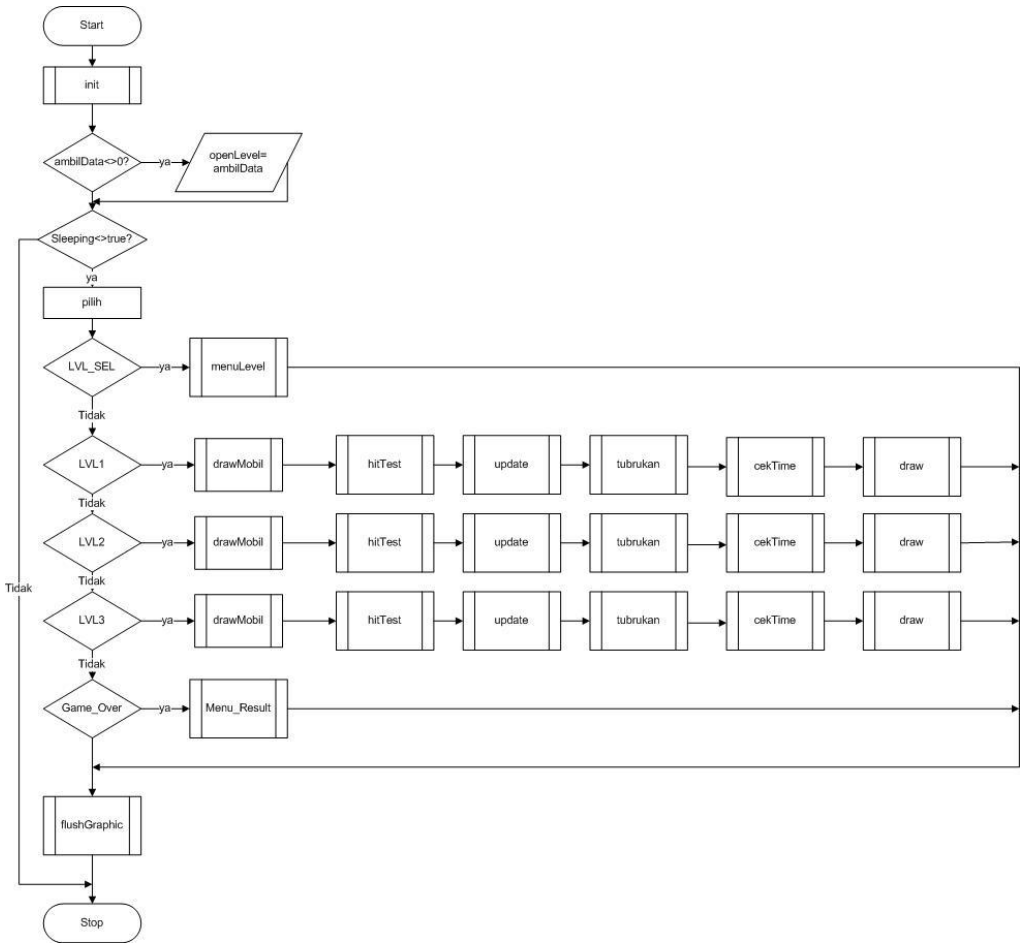
Method startApp akan dipanggil untuk memulai siklus aplikasi.



Gambar 3.55 Flowchart Method startApp()

L. Method run()

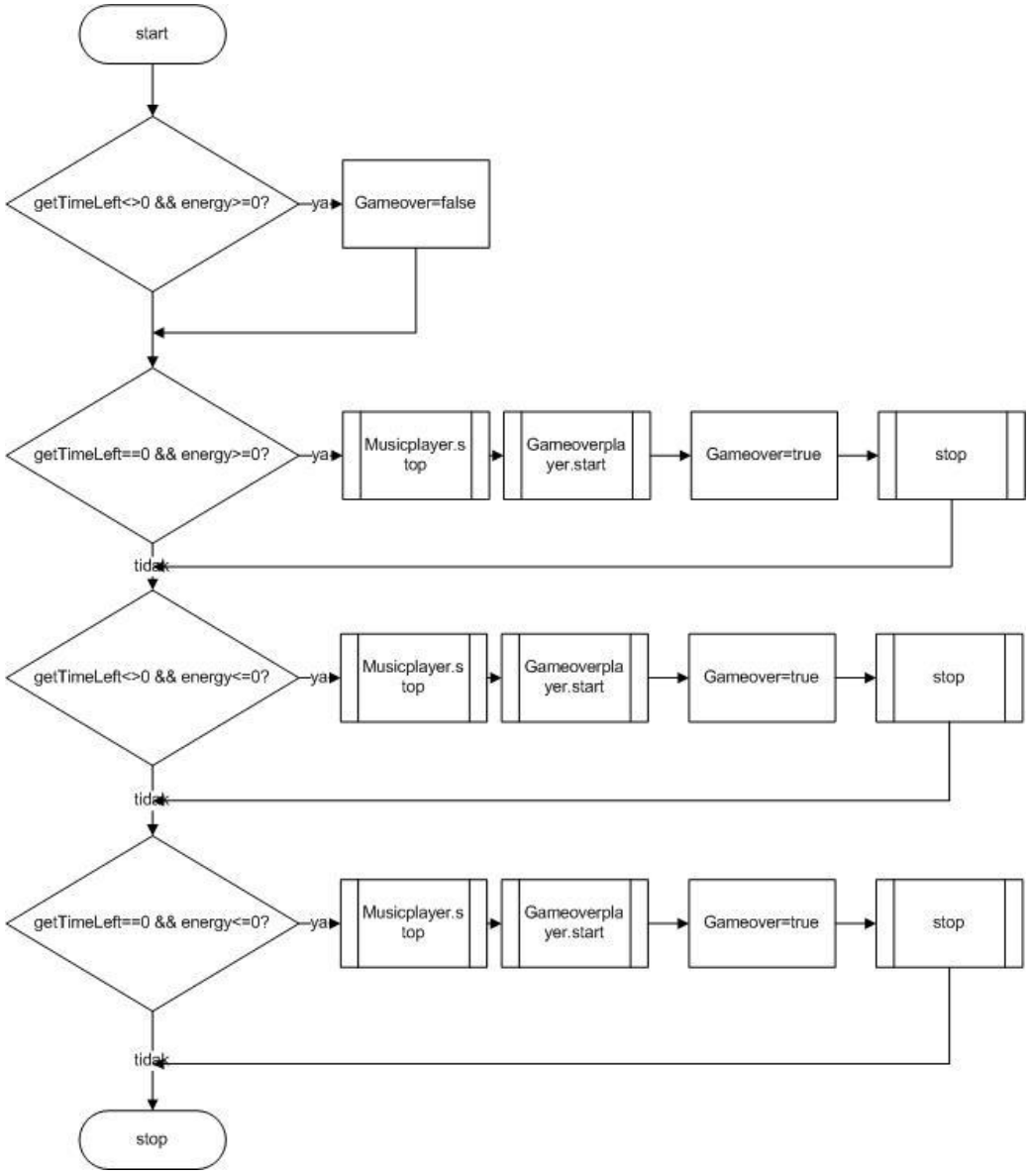
Method run akan dipanggil untuk menampilkan tampilan permainan yang juga merupakan looping dalam permainan *Crazy* Becak.



Gambar 3.56 Flowchart Method run()

J. Method cek*Time*()

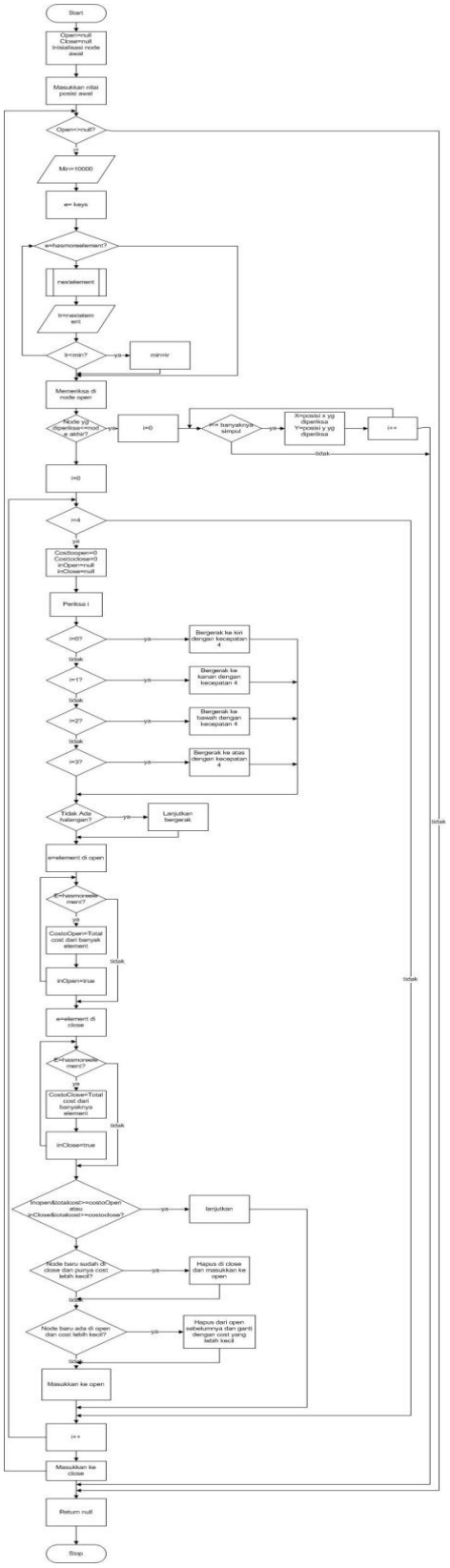
Method cekTime akan dipanggil untuk mengecek waktu permainan berakhir atau tidak.



Gambar 3.57 Flowchart Method cek*Time*()

K. Method aStarSearch()

Method ini dipanggil ketika karakter utama melewati jalan yang tidak boleh dilewati oleh becak.



Gambar 3.58 Flowchart Method algoritma()