BAB 3
ANALISIS DAN PERANCANGAN

3.1 Analisis Sistem

Analisis sistem mendefinisikan analisis masalah kebutuhan fungsional dan non-fungsional.

3.1.1 Analisis Masalah

Berdasarkan latar belakang masalah maka masalah yang dihadapi adalah sebagai berikut :

1. Kurangnya pengetahuan anak-anak di desa Anjun tentang penyakit chikungunya.
2. Kurangnya pengetahuan tentang langkah-langkah pencegahan penyakit chikungunya.

Selama ini informasi yang diberikan oleh kepala desa dan klinik yang ada di lingkungan desa hanya berupa poster yang tersedia di desa dan klinik setempat. solusi tersebut dirasa belum optimal, karena tidak ada penyuluhan khusus mengenai penyakit chikungunya sehingga tidak sampai kepada masyarakat. Perlu adanya inovasi baru untuk memberikan informasi tentang chikungunya dilihat betapa pentingnya informasi tersebut untuk mencegah lebih banyak lagi yang terkena penyakit chikungunya.

Oleh karena itu solusi yang diberikan adalah media pembelajaran yang efektif dan efisien berupa aplikasi *game* Save Family From Chikungunya.

3.1.2 Analisis Game Sejenis

Sebelum merancang *game* yang akan dibuat, terlebih dahulu dilakukan analisis pada *game* sejenis. *Game* sejenis yang bertemakan *Mosquito* adalah *Save Family From Malaria.* Analisis *game* sejenis dilakukan untuk memberikan gambaran sistem agar lebih mudah dalam pembangunan dan sebagai pembanding untuk kesempurnaan *game* yang akan dicapai.

3.1.2.1 *Save Family From Malaria*

Save Family From Malaria adalah *game* bertemakan malaria yang dibangun menggunakan flash. *Game* ini bergenre edukasi. *Story board* dari *game* tersebut dapat dilihat pada Gambar 3.1



Gambar 3.1 Tampilan Menu Utama



Gambar 3.2 Tampilan Permainan

a. Storyline

Game edukasi "*Save Family From Malaria*" menceritakan keteladanan seorang anak SD bernama Bambang. Keluarganya terkena penyakit malaria, ibu dan satu adiknya dilarikan ke rumah sakit karena demam yang sangat tinggi.

Sementara itu, Bambang diberitahu oleh ayah dan dokter tentang penyakit yang diderita ibu dan adiknya tersebut. Penyebab penyakit tersebut yaitu disebabkan gigitan nyamuk malaria (*Anopheles*). Setelah itu, Bambang membaca pamphlet yang ditujukan dokter, kemudian berimajinasi tentang apa yang ada pada pamphlet tersebut seperti bahaya malaria, dan pencegahan menggunakan kelambu. Program 3M, dan DDT.

Dalam imajinasinya dia berjalan menyusuri tempat sarang nyamuk dan memberantas mereka dengan menyemprotkan obat nyamuk. Dia juga melakukan apa yang ada pada program 3M (menguras, menutup, mengubur), memasang kelambu dan diakhir cerita melawan ratu nyamuk yang memiliki parasit paling berbahaya dan mematikan menggunakan semprotan DDT.

b. Misi

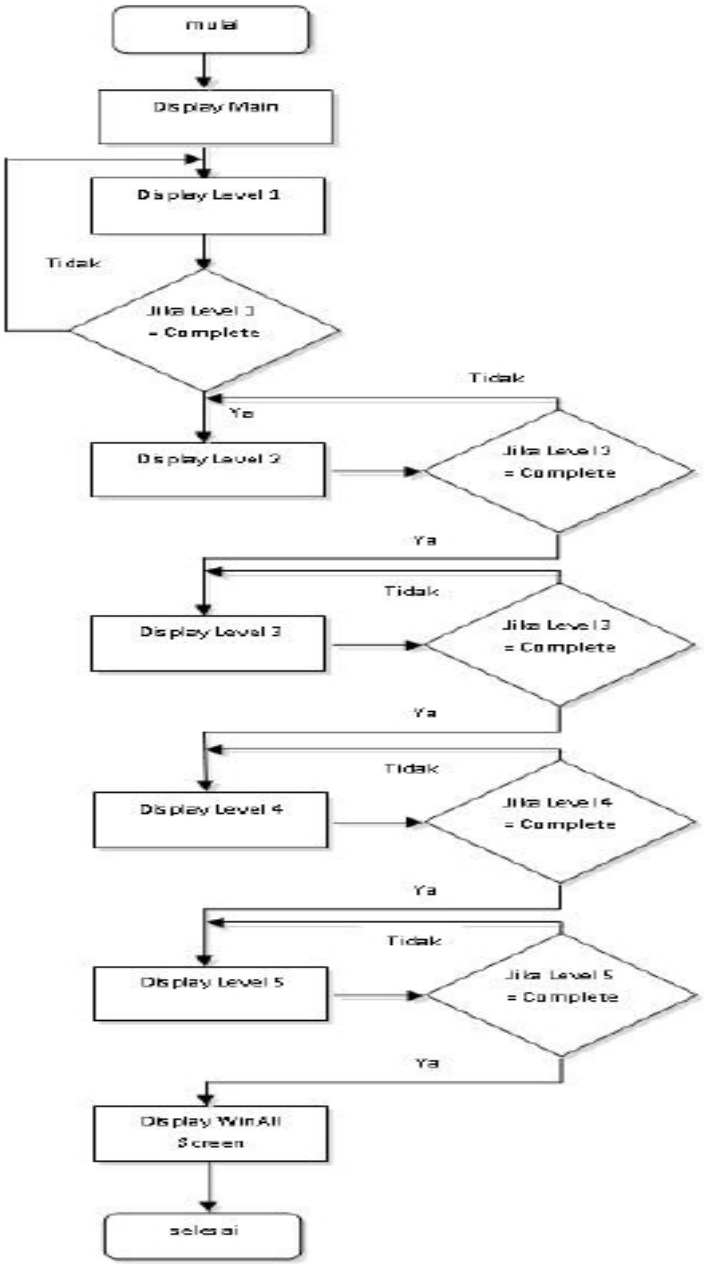
1. Pemain mengendalikan Bambang dan mencoba untuk membrantas nyamuk malaria menggunakan semprotan, memasang kelambu di tempat tidur dan menuju pintu hutan.
2. Pemain mengendalikan Bambang dan mencoba untuk membrantas nyamuk, mengambil sikat pembersih, dan menguras tempat air.
3. Pemain mengendalikan Bambang dan mencoba untuk membrantas nyamuk, mengambil penutup, dan menutup tempat air
4. Pemain mengendalikan Bambang dan mencoba untuk membrantas nyamuk malaria, mengambil, meletakan, dan mengubur wadah kosong.
5. Pemain mengendalikan Bambang dan mencoba untuk membrantas perkembanganbiakan nyamuk malaria dan Ratu dengan menyemprotkan obat nyamuk dan DDT.

c. *GamePlay*

Tugas utama pemain dalam memainkan *game* ini yaitu melalui level-level secara bertahap. Setiap bagian level menyediakan bentuk permainan dan *stage* yang berbeda-beda yang harus di selesaikan oleh pemain.

Pemain memulai permainan dengan level 1, dimana nantinya pemain harus menyelesaikan level tersebut terlebih dahulu untuk kemudian melanjutkan

permainan ke level selanjutnya. Begitu juga dengan level ke 2, 3, 4 dan 5. Hal ini bertujuan untuk memberikan penjelasan detail tentang kronologi dari cerita game dan urutan penanggulan malaria. Untuk lebih jelas dapat dilihat pada *flowchart gameplay* dibawah ini.



Gambar 3. 3 ***Flowchart Gameplay***

Table 3.1 Elemen Yang Terdapat Pada *Game* Sejenis

Elemen	Save Family From Chikungunya
Karakter	Seorang anak-anak
Rules	Memenuhi *rules game edukasi* dengan kesederhanaan aturan dalam permainan
Nyawa	Ada
Score	Ada
Level	5 *Level*
Bos	Ada
Penyajian Materi	Informasi tentang malaria dan pencegahan
Victory Conditions	Ada
Perspective	*Side Scrolling*
Kelebihan	Simulasi 3M
Kekurangan	Penyampaian informasi tentang pencegahan malaria kurang detail, dan terdapat dalam menu utama sehingga tidak semua user akan membaca informasi tersebut.

3.1.3 Analisis Game yang Akan Dikembangkan

Analisis *game* yang akan dikembangkan adalah meliputi Story Line dan *Game* Play *game* Save Family From Chikungunya.

3.1.3.1 Story Line

Game *Save Family From Chikungunya* ini menceritakan tentang anak yang bernama Ibro tinggal di suatu desa, desa tersebut sedang terjangkit wabah penyakit Chikungunya. Ibro melihat adiknya yang menderita penyakit tersebut, dia sangat sedih. Maka dari itu dia bertekad untuk memberantas penyebaran nyamuk yang membawa virus penyakit Chikungunya. Dalam petualangannya dia harus melakukan 3M diantaranya mengubur sampah-sampah dan menutup genangan-genangan air di sekitar lingkungannya, yang dapat menjadi tempat berkembang biaknya nyamuk. Di dalam petualangannya Ibro harus menyelesaikan misi dengan waktu yang sudah ditentukan, Ibro semakin sulit untuk menyelesaikan misinya, Karena di setiap *level* dia akan selalu diserang oleh nyamuk. Ibro akan kalah ketika dia tidak bisa menyelesaikan misi tepat waktu, dan Ibro akan kalah ketika nyawa yang dia punya sudah habis. Nyawa Ibro akan berkurang jika Ibro terkena gigitan nyamuk. Pada setiap *level* Ibro juga akan diberikan senjata berupa semprotan.

3.1.3.2 *Gameplay*

Tugas utama *User* (*gamer)* dalam memainkan aplikasi ini yaitu melalui *level* 1 sampai *level* 3. Pada menu utama akan menampilkan terdapat pilihan bermain, cara bermain, pembuat, dan keluar. Jika *User* memilih menu "bermain" maka akan muncul animasi yang menceritakan awal mula Ibro melakukan petualangan, dan memulai permainan level 1, pada setiap level akan ada pemberitahuan tentang langkah-langkah pencegahan penyakit chikungunya beserta misi yang harus dilakukan oleh user.

Pada *level* 1 Ibro akan memulai petualangan di lingkungan sekitar rumahny misi pada *level* 1 dia harus mengumpulkan 5 buah bungkusan sampah dan menguburnya dalam waktu 2 menit. Pada saat Ibro mencari bungkusan sampah yang akan dikumpulkan tersebut Ibro akan bertemu dengan nyamuk-nyamuk, Ibro bisa menghindar ataau membunuh nyamuk tersebut dengan senjata semprotan yang sudah disediakan. Ibro akan menang jika dia bisa mengumpulkan semua bungkusan sampah tersebut dan berhasil menguburnya tepat waktu. Ibro akan kalah jika Ibro tidak berhasil mengumpulkan dan mengubur semua bungkusan sampah pada tepat waktu, dan Ibro juga akan kalah jika Ibro diserang oleh nyamuk sehingga nyawa yang dia miliki telah habis.

Pada *level* 2 sama dengan *level* 1, hanya di *level* 2 ini Ibro memulai petualangannya di rawa-rawa, dan misi pada *level* ini adalah Ibro harus menutup genangan-genangan air yang terdapat di dalam rawa tersebut.

Pada *level* 3 Ibro akan memulai petualangan dari rawa dan misinya yaitu Ibro harus menutup genangan air, harus mengumpulkan sampah, dan Ibro juga harus mendapatkan alat untuk membersihkan bak, sedangkan waktu yang diberikan tidak banyak. Serangan dari nyamuk akan lebih sering dan banyak, ketika semua misi sudah diselesaikan Ibro akan bertemu dengan nyamuk raksasa (Raja nyamuk). Ibro harus bisa mengalahkan raja nyamuk tersebut dengan senjata yang dia miliki, dan permainan selesai. Pada *level* ini Ibro akan menang jika Ibro mampu mengalahkan semua nyamuk dan raja nyamuk. Ibro akan kalah jika Ibro tidak mampu mengalahkan Raja nyamuknya.

Dalam menu utama juga ada pilihan “Cara bermain” yaitu untuk melihat cara bermain aplikasi ini dan pilihan lainnya adalah pilihan “Pembuat”, sebagai keterangan yang menyediakan informasi mengenai pembuat game ini.

3.1.3.3 Analisis Tingkat Kesulitan

Tingkat kesulitan pada *game* ini adalah sebagai berikut :

1. Misi atau tugas

Misi yang diberikan akan berbeda-beda sesuai level yang dimainkan. Misi yang diberikan sesuai dengan langkah-langkah yang harus dilakukan untuk mencegah penyebaran nyamuk *Aedes aegypti* yang membawa virus chikungunya **[4].** Misi-misi yang akan diberikan pada tiap level adalah:

a. Level 1

Misi yang diberikan di level ini adalah Pencegahan dengan cara mengubur sampah-sampah.

b. Level 2

Misi yang diberikan di level ini adalah Pencegahan dengan cara menutup genangan air.

c. Level 3

Misi yang diberikan di level ini adalah Pencegahan dengan cara 3M, yaitu: Mengubur sampah, menutup genangan air, dan menguras bak.

2. Musuh

Pada setiap level Ibro akan bertemu dengan nyamuk yang terbang ke kanan dan ke kiri. Semakin tinggi level, maka jumlah musuh akan bertambah sehingga permainan menjadi lebih sulit. Jenis musuh yang terdapat dalm game ini adalah:

a. Nyamuk

b. Raja nyamuk (King Mosquito)

Kemunculan nyamuk-nyamuk tersebut pada tiap level bisa dilihat pad tabel 3.2

Table 3. 2 Jumlah Musuh Tiap *Level*

No	***Level 1***	***Level 2***	***Level 3***
Jumlah Musuh	10	13	15
Jumlah Raja Musuh	-	-	**1**

3.1.3.4 Analisis Algoritma A* (A Star)

Pada *game* Edukasi 2D "Save Family From Chikungunya", algoritma A* diterapkan pada raja musuh di level 3 dengan beberapa *node* pada peta dengan ordo X dan Y. Struktur data yang digunakan adalah *array* dua dimensi dengan menentukan bobot sesuai koordinat setiap *node*. Berikut ini adalah contoh kasus algoritma A* pada *game* Save Family From Chikungunya dapat dilihat pada gambar 3.4

X / Y	1	2	3	4	5	6	7	8	9	10
1				Enemy						
2										
3										
4										
5	Player									

Gambar 3.4 Contoh Kasus Algoritma Pada *Game Save Family From Chikungunya*

Tile atau *node* tujuan atau pada kasus ini disebut *node* pemain direpresentasikan dengan warna biru dan berada di koordinat (1,5) sedangkan *node* awal atau pada kasus ini disebut musuh direpresentasikan dengan warna merah dan berada di *node* (4,1), sedangkan untuk node yang berwarna abu-abu merupakan penghalang pada *game.* Musuh dapat bergerak ke semua *node* yang berwarna putih tetapi musuh tidak dapat bergerak ke arah *node* yang berwarna abu-abu. Sedangkan untuk penentuan nilai heuristik dapat dilihat di gambar 3.5

X / Y	1	2	3	4	5	6	7	8	9	10
1	H=40	H=50	H=60	Enemy	H=80	H=90	H=100	H=110	H=120	H=130
2	H=30	H=40	H=50			H=80	H=90			
3	H=20	H=30	H=40	H=50	H=60	H=70	H=80	H=90	H=100	H=110
4	H=10	H=20	H=30						H=90	H=100
5	Player	H=10	H=20	H=30	H=40	H=50	H=60	H=70	H=80	H=90

Gambar 3.5 Penentuan Nilai Heuristik

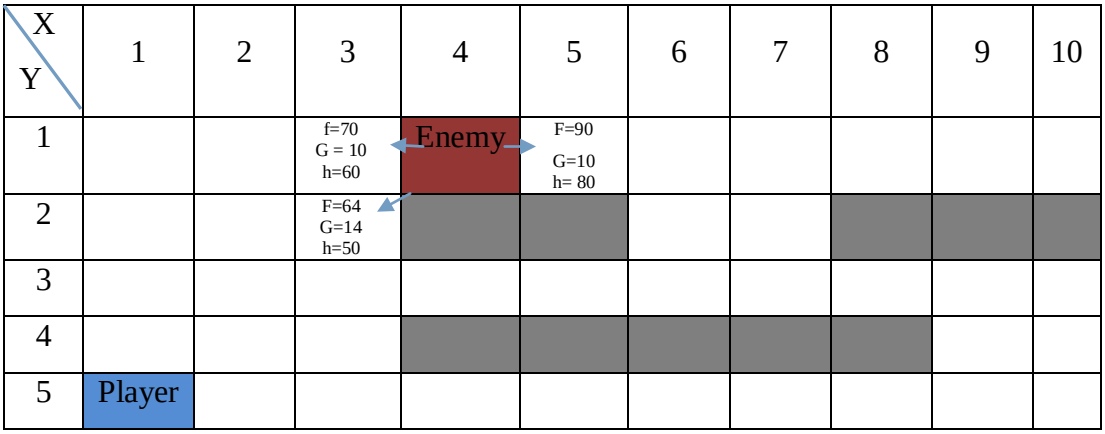
Nilai untuk pergerakan secara horizontal dan vertikal adalah 10, sedangkan untuk nilai pergerakan diagonal adalah 14. Untuk mendapatkan nilai heuristik menggunakan *diagonal distance* yang memungkinkan pergerakan ke 8 arah maka digunakan notasi matematika

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

Selanjutnya *node* musuh akan melakukan pencarian untuk menemukan jalur terpendek dengan cara memeriksa *node* disekitarnya. Langkah-langkah pencarian jalur terpendek adalah sebagai berikut:

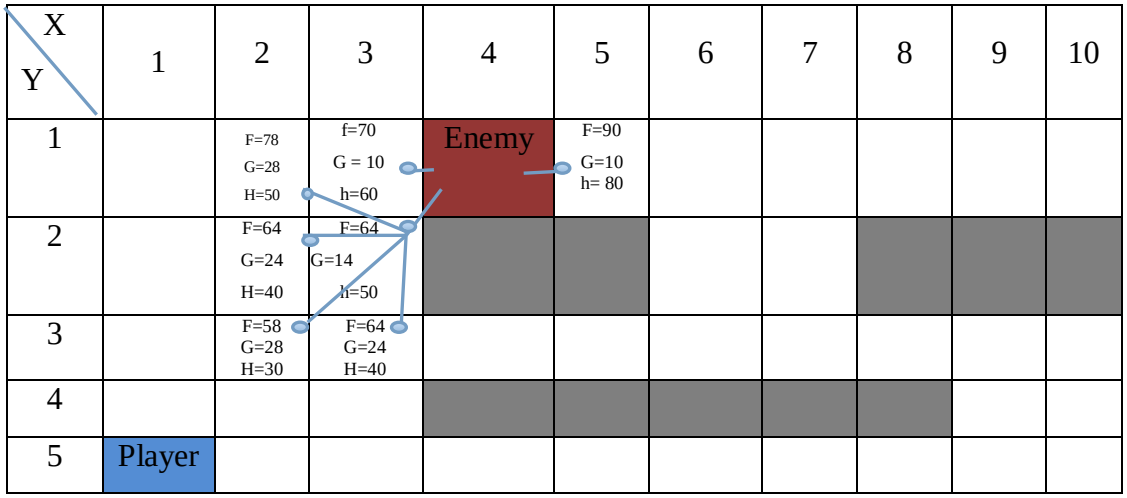
1. *Node* musuh dimasukkan ke dalam *open list.*
2. Periksa semua *node* yang berdekatan dengan *node* musuh apakah *node* tersebut merupakan penghalang atau tidak, abaikan *node* yang tidak bisa dilalui, lalu masukkan *node* tersebut ke dalam *open list.* Dan untuk setiap *node* baru yang ditambahkan ke dalam *open list*, simpan *node* lama sebagai "*parent"* yang akan digunakan untuk menelusuri jalan.
3. *Node* dengan nilai f terkecil akan terpilih sebagai *best node* dan selanjutnya pencarian akan dimulai dari *node* ini.
4. Hapus *node* musuh atau *node* sebelumnya dai *open list* dan masukkan ke dalam *closed list.*

Di bawah ini adalah gambar membangkitkan suksesor *node* musuh dapat dilihat di gambar 3.6



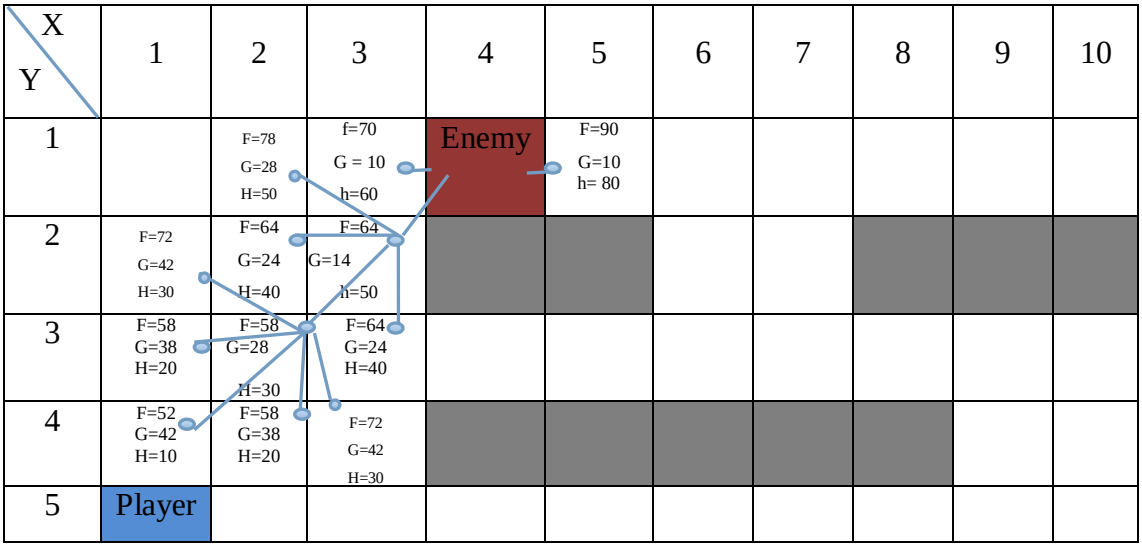
Gambar 3.6 Pembangkitan *Node* Suksesor Musuh

Langkah pertama, *node* musuh membangkitkan *node* di sekitarnya dan hasilnya membangkitkan 3 *node*, baik arah horizontal, vertical, maupun diagonal. *Node* pertama dengan koordinat (3,1) memiliki nilai g = 10, h = 60 sehingga f = g + h = 70. *Node* kedua dengan koordinat (3,2) memiliki nilai g = 14, h = 50 sehingga f = g + h = 64. *Node* ketiga dengan koordinat (5,1) memiliki nilai g = 10, h = 80 sehingga f = g + h = 90. Berikut ini adalah langkah pertama pencarian rute terpendek menggunakan akgoritma A* dapat dilihat di gambar 3.7



Gambar 3.7 Langkah Ke-1 Pencarian Jalur Terpenek Dengan Algoritma A*

Node dengan koordinat (3,2) memiliki biaya terkecil (nilai f=64) terpilih sebagai *BestNode* dan pencarian selanjutnya dimulaidari *node* ini. Pencarian dimulai dengan menelususri *node* pertama dengan koordinat (2,3) memiliki nilai g = 28, angka ini didapat berdasarkan g *node* sebelumnya ditambah 14, h = 30, sehingga f = g + h = 58. *Node* kedua dengan koordinat (2,1) memiliki nilai g = 28, angka ini didapat berdasarkan g *node* sebelumnya ditambah 14, h = 50, sehingga f = g + h = 78. *Node* ketiga dengan koordinat (2,2) memiliki nilai g = 24, angka ini didapat berdasarkan g *node* sebelumnya ditambah 10, h = 40, sehingga f = g + h = 64. *Node* keempat dengan koordinat (3,3) memiliki nilai g = 24, angka ini didapat berdasarkan g *node* sebelumnya ditambah 10, h = 40, sehingga f = g + h = 64. Berikut ini adalah langkah kedua pencarian rute terpendek menggunakan algoritma A* dapat dilihat di gambar 3.8



Gambar 3. 8 Langkah Ke-2 Pencarian Jalur Terpendek Dengan Algoritma A*

Node dengan arah diagonal kiri bawah memiliki biaya terkecil (f = 58) terpilih sebagai *BestNode* dan pencarian selanjutnya dimulai dari *node* ini. Pencarian dimulai dengan menelusuri *node* pertama dengan koordinat (1,3) memiliki nilai g = 38 angka ini didapat berdasarkan g *node* sebelumnya ditambah 10, h = 20, dan f = g + h = 58. *Node* kedua dengan koordinat (2,4) memiliki nilai g = 38 angka ini didapat berdasarkan g *node* sebelumnya ditambah 10, h = 20, dan f = g + h = 58. *Node* ketiga dengan koordinat (1,2) memiliki nilai g = 42 angka ini didapat berdasarkan g *node* sebelumnya ditambah 14, h = 30, dan f = g + h = 72. *Node* keempat dengan koordinat (1,4) memiliki nilai g = 42 angka ini didapat berdasarkan g *node* sebelumnya ditambah 14, h = 10, dan f = g + h = 52. *Node* kelima dengan koordinat (3,4) memiliki nilai g = 42 angka ini didapat berdasarkan g *node* sebelumnya ditambah 14, h = 30, dan f = g + h = 72. Berikut ini adalah langkah ketiga pencarian rute terpendek menggunakan algoritma A* dapat dilihat di gambar 3.9

X / Y	1	2	3	4	5	6	7	8	9	10
1		F=78 G=28 H=50	f=70 G = 10 h=60	Enemy	F=90 G=10 h= 80					
2	F=72 G=42 H=30	F=64 G=24 H=40	F=64 G=14 h=50							
3	F=58 G=38 H=20	F=58 G=28 H=30	F=64 G=24 H=40							
4	F=52 G=42 H=10	F=58 G=38 H=20	F=72 G=42 H=30							
5	Player	F=66 G=56 H=10								

Gambar 3.9 Langkah Ke-3 Pencarian Jalur Terpendek Dengan Algoritma A*

Node dengan arah diagonal kiri bawah memiliki biaya terkecil (f = 52) terpilih sebagai *BestNode* dan pencarian selanjutnya dimulai dari *node* ini. Pencarian dimulai dengan menelusuri *node* pertama dengan koordinat (2,5) memiliki nilai g = 56, angka ini didapat berdasarkan g *node* sebelumnya ditambah 14, h = 10, dan f = g + h = 66 *Node* kedua dengan koordinat (1,5) memiliki nilai g = 52 angka ini didapat berdasarkan g *node* sebelumnya ditambah 10, h = 0, dan f = g + h = 52. Berikut ini adalah langkah keempat pencarian rute terpendek menggunakan algoritma A* dapat dilihat di gambar 3.10

X / Y	1	2	3	4	5	6	7	8	9	10
1		F=78 G=28 H=50	f=70 G = 10 h=60	Enemy	F=90 G=10 h= 80					
2	F=72 G=42 H=30	F=64 G=24 H=40	F=64 G=14 h=50							
3	F=58 G=38 H=20	F=58 G=28 H=30	F=64 G=24 H=40							
4	F=52 G=42 H=10	F=58 G=38 H=20	F=72 G=42 H=30							

5	Player	F=66 G=56 H=10								

Gambar 3.10 Langkah Ke-4 Pencarian Jalur Terpendek Dengan Algoritma A*

Dari node berwarna merah (enemy) menuju node berwarna biru (player) maka di dapatkan jalur terpendek dengan menggunakan algoritma pencarian sebagai berikut:

1. Dari node merah (enemy) bergerak ke koordinat (3,2) dengan f = 64.
2. Dari node dengan nilai f=64 bergerak ke koordinat (2,3) dengan nilai f = 58.
3. Dari node dengan nilai f=58 bergerak ke koordinat (1,4) dengan nilai f=52.
4. Dari node dengan nilai f=52 bergerak ke koordinat (1,5) dan menemukan goal (*node* player).

3.1.4 Analisis Kebutuhan Non-Fungsional

Analisis kebutuhan non-fungsional untuk menetukan spesifikasi kebutuhan sistem. Sistem membutuhkan elemen dan komponen sebagai masukan, proses dan keluaran.

3.1.4.1 Analisis Kebutuhan Perangkat Keras

Kebutuhan perangkat keras yang dibutuhkan agar aplikasi *Game* "Save Family From Chikungunya" dapat digunakan sebagai berikut:

1. Processor minimal 2,6 GHZ
2. RAM : 1GB
3. VGA 512 MB
4. Harddisk 320 GB
5. Monitor 14'
6. Keyboard & Mouse
7. Speaker

3.1.4.2 Analisis Kebutuhan Perangkat Lunak

Perangkat lunak sangat dibutuhkan untuk membuat dan memainkan game edukasi ini. Permainan edukatif save family from Chikungunya ini dalam pembangunan menggunakan perangkat lunak sebgai berikut:

1. Adobe Flash CS3 ActionScript 3.0
2. Adobe Illustrator CS3
3. Sistem operasi Windows 7

Perangkat lunak yang dibutuhkan untuk mengoperasikan aplikasi game ini, yaitu:

1. Sistem operasi Windows
2. Adobe Flash Player 10.2

3.1.4.3 Analisis User

User pada penelitian ini adalah anak-anak desa Anjun yang membutuhkan media pembelajaran tentang penyakit chikungunya. Dalam hal ini dibutuhkan media pembelajaran yang efektif dan efisien untuk anak-anak di desa Anjun. Alasan dibutuhkannya media ini karena anak-anak desa Anjun membutuhkan media pembelajaran efektif dan efisien yang menerangkan tentang penyakit chikungunya dan langkah-langkah pencegahannya. Diharapkan dengan adanya sebuah media dapat membantu anak-anak lebih memahami tentang wabah penyakit chikungunya.

Untuk membuat permainan edukatif yang dapat digunakan oleh anak-anak desa Anjun, *game* Save Family From Chikungunya tidak memiliki ciri-ciri dan prinsip khusus.

Yakni berdasarkan *user knowledge & experience,* dan *user physical characteristic* seperti yang bisa dilihat pada tabel 3.3

Table 3.3 Analisis Pengguna

KNOWLEDGE AND EXPERIENCE	
Computer Literacy	Rendah, minimal bisa menggunakan komputer
Pengalaman Sistem	Sedang, minimal biasa menggunakan komputer
Pengalaman Aplikasi	Sedang, minimal pernah memainkan game
Penggunaan Sistem Lain	Jarang
Edukasi	Sekolah Dasar
Bahasa / Kultur	Bahasa Indonesia
PSYCHOLOGICAL CHARACTERISTICS	
Attitude	Positif

Motivasi	Tinggi
Stress Level	*No Stress*
PHYSICAL CHARACTERISTICS	
Umur	7-11 tahun
Jenis Kelamin	Laki – laki dan Perempuan
Handedness	Kanan, Kidal, dan Keduanya

3.1.5 Analisis Kebutuhan Fungsional

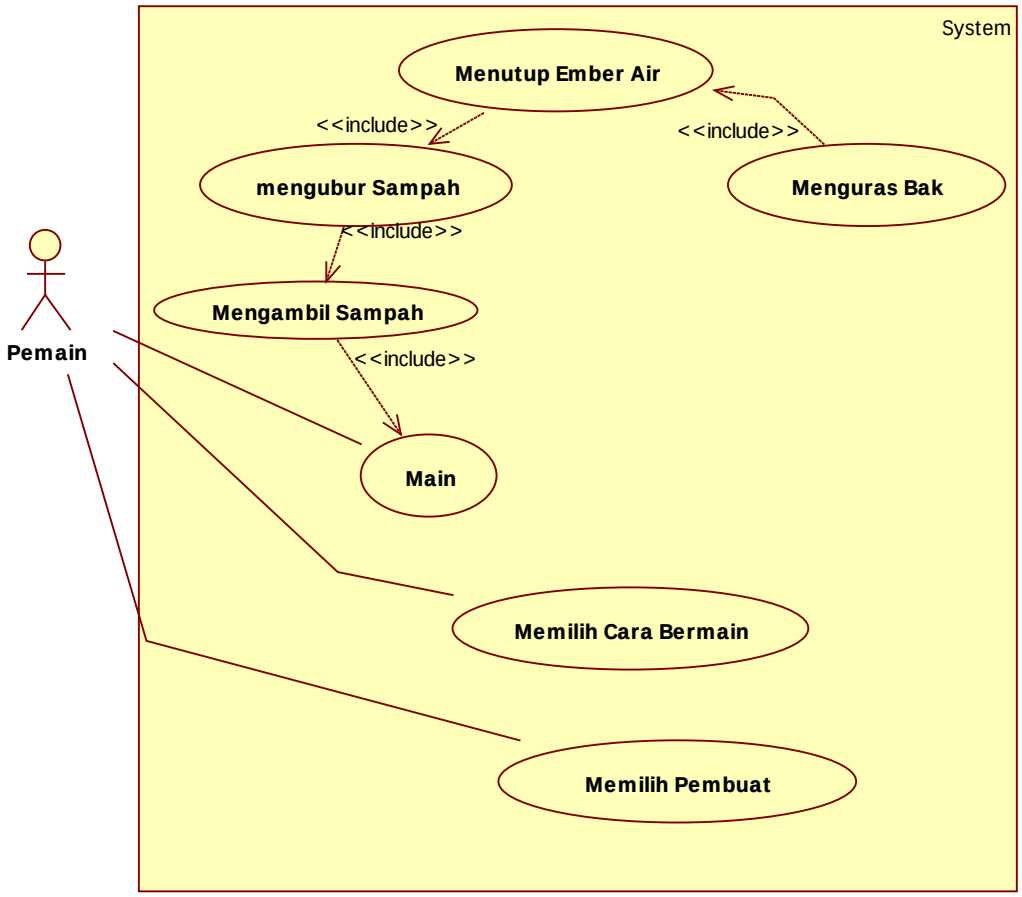
Analisis kebutuhan fungsional merupakan tindak lanjut dari analisis sistem. Tujuannya adalah untuk memberikan gambaran mengenai sistem yang akan dibuat. Tahap awal yang dilakukan dalam melakukan analisis berorientasi objek menggunakan UML (Unified Modeling Language). Dimana tahap-tahap perancangan yang dilakukan dalam membangun *Game* Chikungunya antara lain Use Case Diagram, Class Diagram, Sequence Diagram, Activity Diagram.

3.1.5.1 Identifikasi Aktor

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

3.1.5.2 Use Case Diagram

Berikut ini adalah perancangan proses-proses yang terdapat pada *game* Save Family From Chikungunya :



Gambar 3.11 Use Case Diagram *Save Family From Chikungunya*

1. Definisi Aktor

Definisi aktor merupakan penjelasan dari apa yang dilakukan oleh aktor-aktor yang terlibat dalam perangkat lunak yang sedang dirancang. Berikut deskripsi dari aktor terlibat:

Table 3.4 Definisi Aktor

No	Aktor	Deskripsi
1	*Pemain*	Orang yang akan memainkan permainan

2. Definisi Use Case

Definisi *Use Case* merupakan penjelasan fungsi *Use Case* yang terdapat pada *Use Case Diagram.* Berikut deskripsi dari *Use Case* yang ada:

Table 3.5 Definisi *Use Case*

No	*Use Case*	Deskripsi
1	Main	Fungsi untuk memainkan game
2	Memilih Cara Bermain	Fungsi untuk memilih cara bermain
3	Memilih Pembuat	Fungsi untuk memilih info pembuat
7	Mengambil Sampah	Fungsi untuk memainkan *level* 1 mengambil sampah
8	Mengubur Sampah	Fungsi untuk memainkan *level* 1 mengubur sampah
9	Menutup Ember Air	Fungsi untuk memainkan *level* 2 menutup ember air
10	Membersihkan Bak	Fungsi untuk memainkan *level* 3 membersihkan bak

3. Skenario Use Case

Skenario *Use Case* merupakan penjelasan apa saja yang dilakukan pada sebuah *Use Case.*

1. Main

Skenario *use case* ini menjelaskan interaksi antara *actor* yaitu pemain, dengan *use case* main yang dijelaskan pada table 3.6 di bawah ini.

Table 3.6 Skenario *Use Case Main*

Identifikasi	
Nama	Main
Tujuan	Untuk dapat memulai permainan
Deskripsi	Fungsi untuk memainkan game
Aktor	*Pemain*
Skenario Utama	
Kondisi awal	Menu utama ditampilkan
Aksi Aktor	**Reaksi Sistem**
1. Pemain memilih menu main	
	2. Menampilkan ilustrasi cerita part 1
3. Pemain menonton ilustrasi cerita hingga selesai	
	4. Menampilkan informasi pencegahan dan misi permainan
5. Pemain memilih tombol main	
	6. Menampilkan permainan
Skenario Aternatif	
1.Pemain memilih tombol selanjutnya	
	2. Menampilkan informasi pencegahan dan misi permainan
3. Pemain Memilih tombol main	
	4. Menampilkan permainan
Kondisi Akhir	Menampilkan permainan

2. Memilih Cara Bermain

Skenario *use case* ini menjelaskan interaksi antara *actor* yaitu pemain, dengan *use case* main yang dijelaskan pada table 3.7 di bawah ini.

Table 3.7 Skenario *Use Case* Memilih Cara Bermain

Identifikasi	
Nama	Memilih Cara Bermain
Tujuan	Untuk dapat melihat cara bermain
Deskripsi	Fungsi untuk memilih cara bermain
Aktor	Pemain
Skenario Utama	
Kondisi Awal	Menampilkan menu utama
Aksi Aktor	**Reaksi Sistem**
1. Pemain memilih menu Cara Bermain	
	2. Menampilkan info cara bermain
3. Pemain memilih tombol kembali	
Kondisi Akhir	Menampilkan menu utama

3. Memilih Pembuat

Skenario *use case* ini menjelaskan interaksi antara *actor* yaitu pemain, dengan *use case* main yang dijelaskan pada table 3.8 di bawah ini.

Table 3.8 Skenario *Use Case* Memilih Pembuat

Identifikasi	
Nama	Memilih Pembuat
Tujuan	Untuk dapat melihat info pembuat
Deskripsi	Fungsi untuk melihata pembuat
Aktor	*Pemain*
Skenario Utama	
Kondisi Awal	Menampilkan menu utama
Aksi Aktor	**Reaksi Sistem**
1. Memilih menu pembuat	
	2. Menampilkan info pembuat
3. Pemain memilih tombol kembali	
Kondisi Akhir	Menampilkan menu utama

4. Mengambil Sampah

Skenario *use case* ini menjelaskan interaksi antara *actor* yaitu pemain, dengan *use case* main yang dijelaskan pada table 3.9 di bawah ini.

Table 3.9 Skenario *Use Case* Mengambil Sampah

Identifikasi	
Nama	Mengambil Sampah
Tujuan	Untuk pencegahan penyebaran nyamuk dengan mengambil sampah
Deskripsi	Fungsi untuk mengambil sampah
Aktor	Pemain
Skenario Utama	
Kondisi Awal	Pemain berada pada permainan level 1
Aksi Aktor	**Reaksi Sistem**
	1. Menampilkan item sampah
2. Menghampiri item sampah	
	3. Sampah menghilang
Kondisi Akhir	Sampah sudah habis

5. Mengubur Sampah

Skenario *use case* ini menjelaskan interaksi antara *actor* yaitu pemain, dengan *use case* main yang dijelaskan pada table 3.10 di bawah ini.

Table 3.10 Skenario *Use Case* Mengubur Sampah

Identifikasi	
Nama	Mengubur Sampah
Tujuan	Untuk pencegahan penyebaran nyamuk dengan mengubur sampah
Deskripsi	Fungsi untuk mengubur sampah
Aktor	Pemain
Skenario Utama	
Kondisi Awal	Sampah sudah habis
Aksi Aktor	**Reaksi Sistem**
1. Menghampiri lubang	
	2. Jika sampah=5
	3. Menampilkan lubang terbuka
4. Masukan sampah pada lubang	
5. Menutup lubang	
	6. Menampilkan lubang tertutup
Kondisi Akhir	Memainkan *game* pencegahan dengan mengubur sampah

6. Menutup Ember Air

Skenario *use case* ini menjelaskan interaksi antara *actor* yaitu pemain, dengan *use case* main yang dijelaskan pada table 3.11 di bawah ini.

Table 3.11 Skenario *Use Case* Menutup Ember Air

Identifikasi	
Nama	Menutup Ember Air
Tujuan	Untuk pencegahan penyebaran nyamuk dengan menutup ember air

Deskripsi	Fungsi untuk menutup ember air
Aktor	Pemain
Skenario Utama	
Kondisi Awal	Pemain berada pada permainan level 2
Aksi Aktor	**Reaksi Sistem**
	1. Menampilkan item ember air
2. Menghampiri item ember air	
	3. Menampilkan ember air tertutup
Kondisi Akhir	Memainkan *game* pencegahan dengan menutup ember air

7. Menguras Bak

Skenario *use case* ini menjelaskan interaksi antara *actor* yaitu pemain, dengan *use case* main yang dijelaskan pada table 3.12 di bawah ini.

Table 3.12 Skenario *Use Case* Menguraa Bak

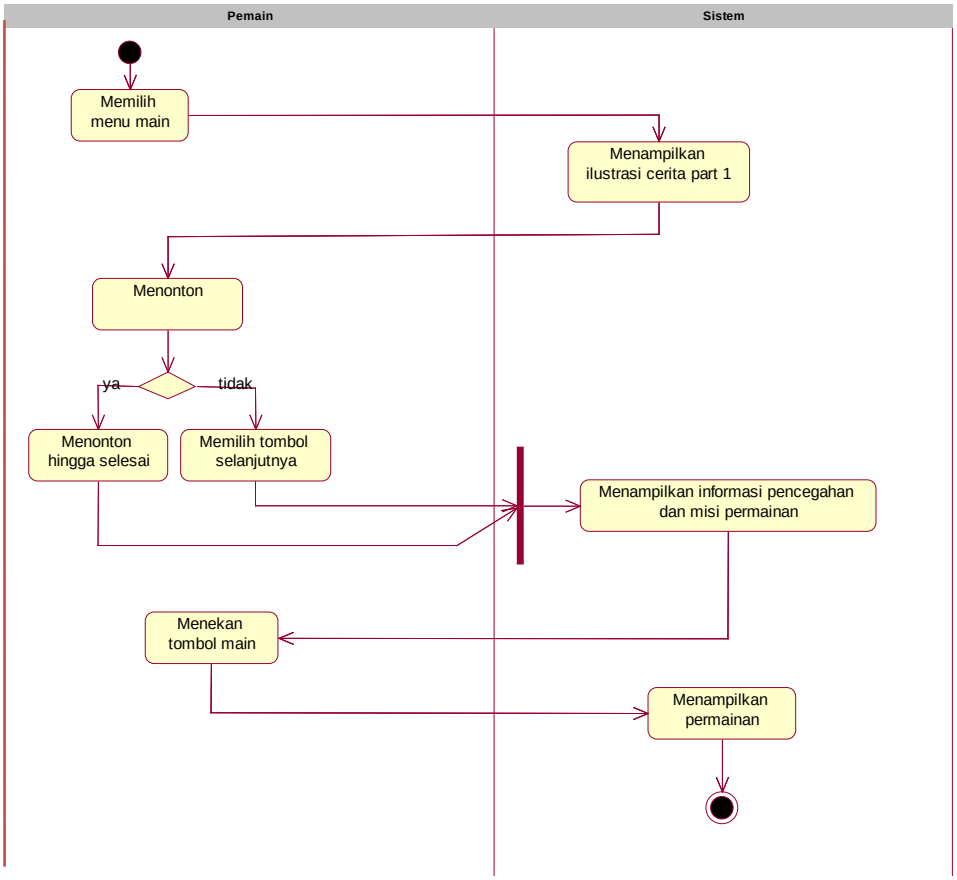
Identifikasi	
Nama	Menguras Bak
Tujuan	Untuk pencegahan penyebaran nyamuk dengan menguras bak
Deskripsi	Fungsi untuk menguras bak
Aktor	Pemain
Skenario Utama	
Kondisi Awal	Pemain berada pada permainan level 3
Aksi Aktor	**Reaksi Sistem**
	1. Menampilkan item-item pada permainan level 3
2. Menghampiri item-item pada permainan level 3	
	3. Menampilkan item pintu
4. Membuka pintu	
	5. Memunculkan raja nyamuk
6. Bertemu dengan raja nyamuk	
	7. Menggerakan raja nyamuk
8. Menembakkan semprotan	
	9. Darah raja=0
	10. Raja mati
	11. Menampilkan karakter sedang menguras bak
12. Memilih tombol selanjutnya	
	13. Menampilkan semua informasi tentang penyakit chikungunya
Kondisi Akhir	Memainkan *game* pencegahan dengan menguras bak

3.1.5.3 Activity Diagram

Diagram aktivitas atau *activity diagram* menggambarkan aliran kerja (*work flow*) atau aktivitas dari sebuah sistem atau proses bisnis.

1. Main

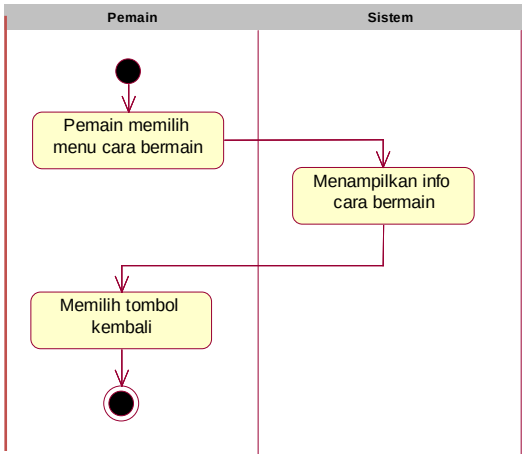
Activity diagram untuk use case Main



Gambar 3.12 Activity Diagram Main

2. Memilih Cara Bermain

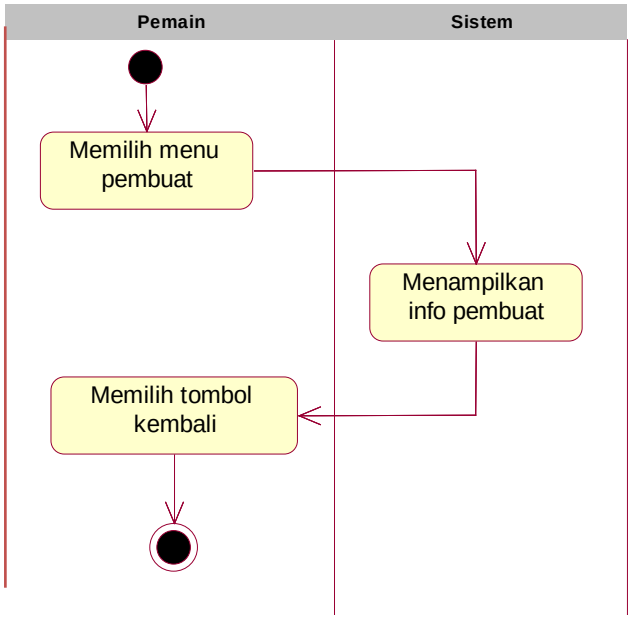
Activity diagram untuk use case Memilih Cara Bermain



Gambar 3.13 Activity Diagram Memilih Cara Bermain

3. Memilih Pembuat

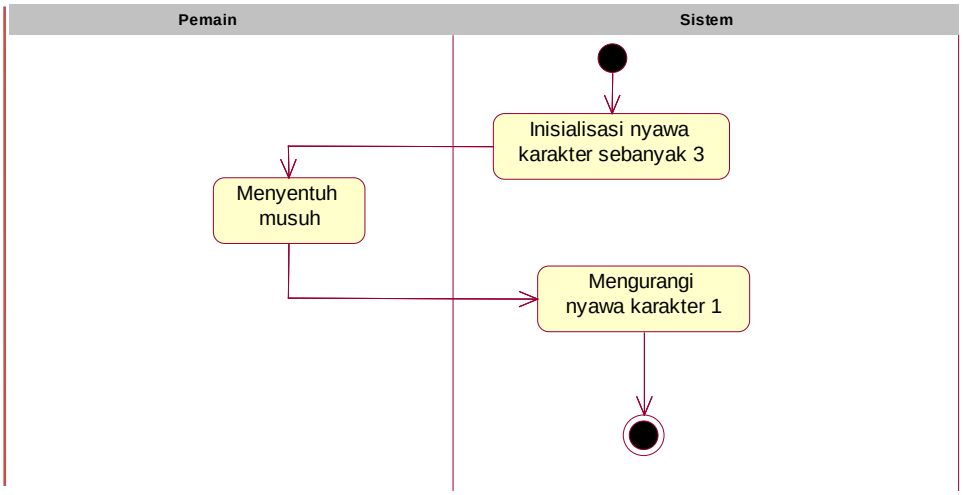
Activity Diagram untuk use case Memilih Pembuat



Gambar 3.14 Activity Diagram Memilih Menu Pembuat

4. Menghitung Nyawa

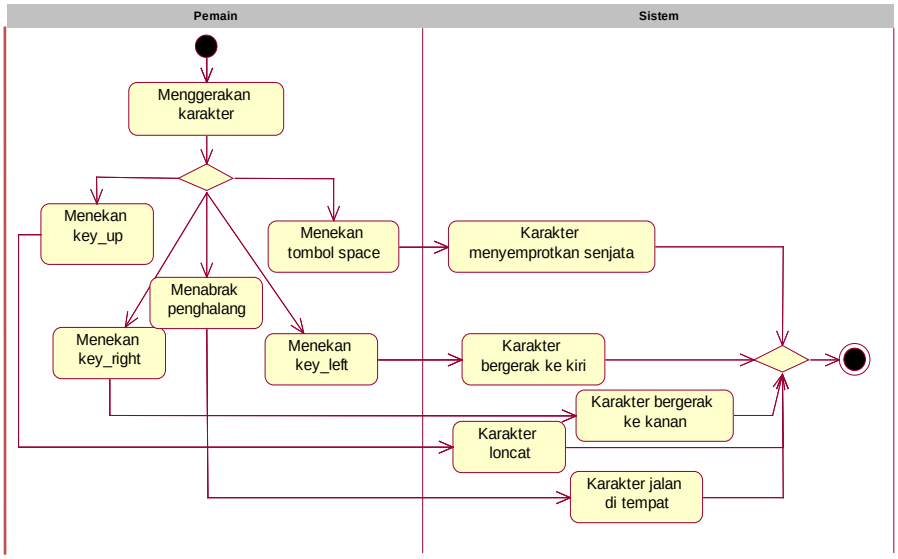
Activity diagram Menghitung Nyawa



Gambar 3.15 Activity Diagram Menghitung Nyawa

5. Pergerakan Karakter

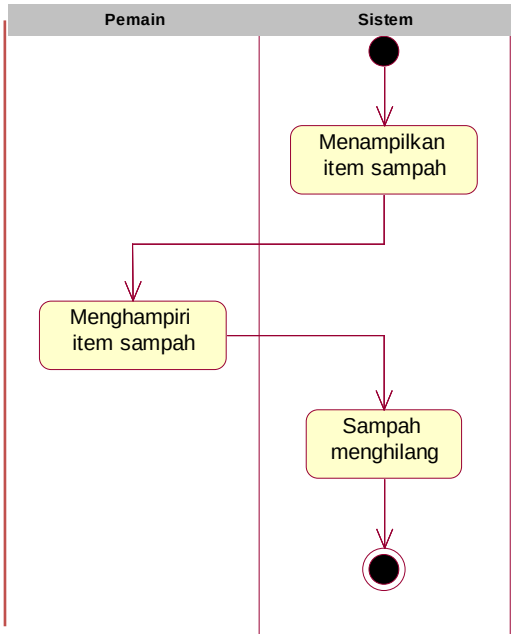
Activity diagram Pergerakan Karakter



Gambar 3.16 Activity Diagram Pergerakan Karakter

6. Mengambil Sampah

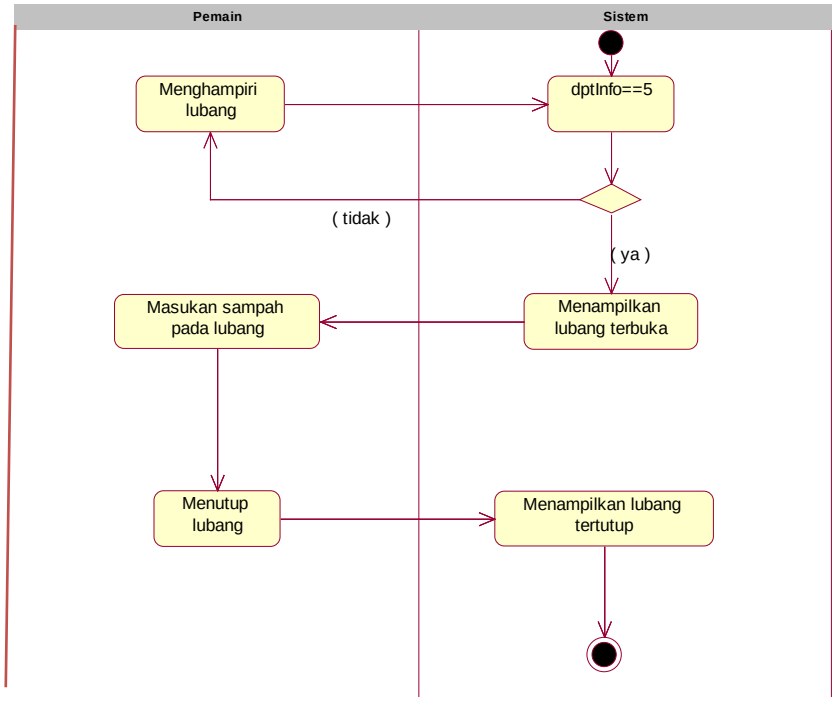
Activity diagram untuk use case Mengambil Sampah



Gambar 3.17 Activity Diagram Mengambil Sampah

7. Mengubur Sampah

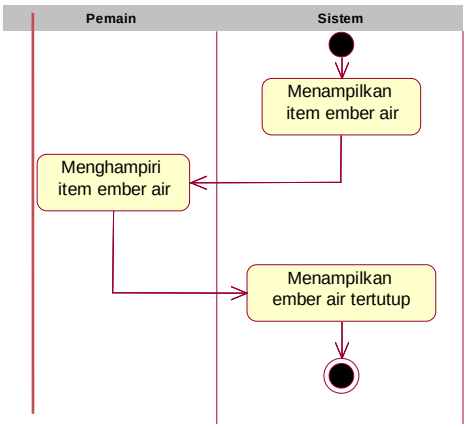
Activity diagram untuk use case Mengubur Sampah



Gambar 3.18 Activity Diagram Mengubur Sampah

8. Menutup Ember Air

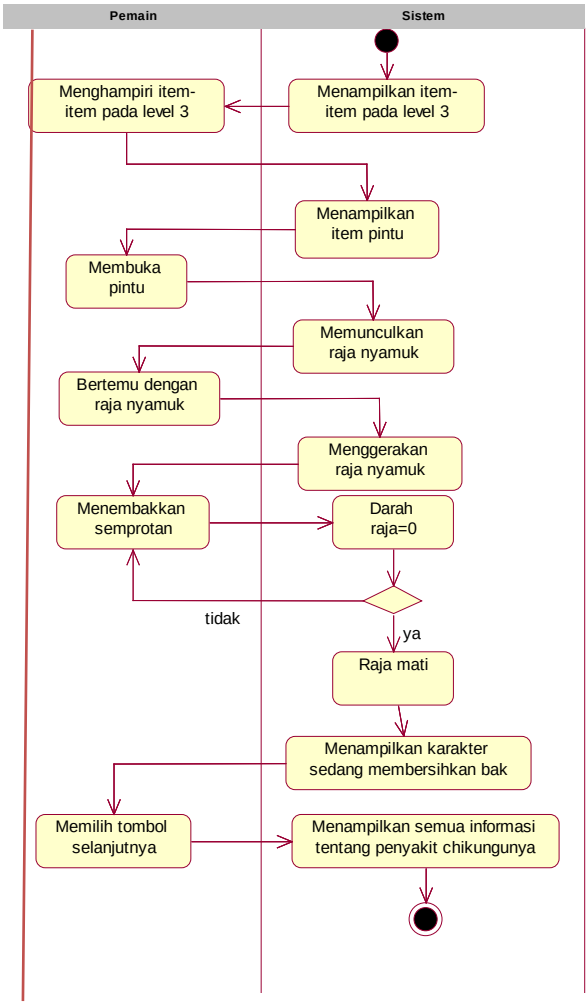
Activity diagram untuk use case Menutup Ember Air



Gambar 3.19 Activity Diagram Menutup Ember Air

9. Menguras Bak

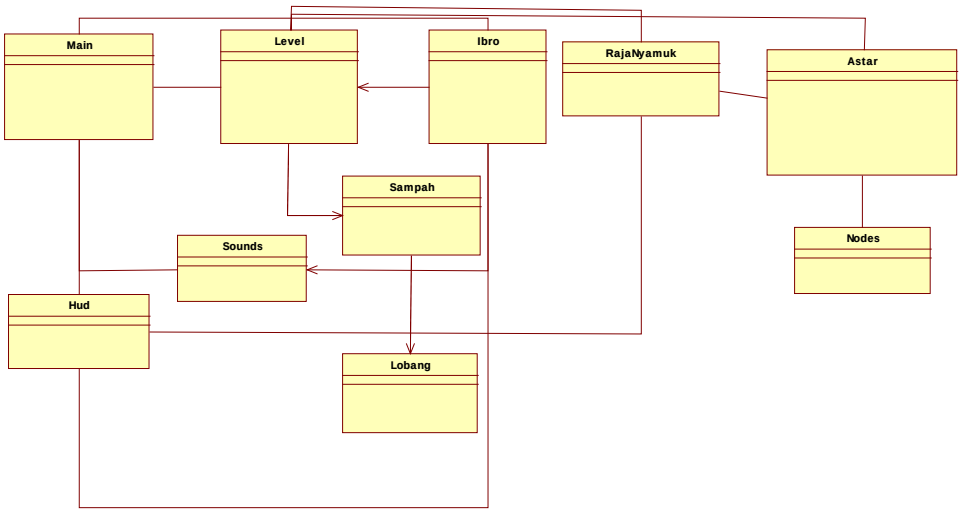
Activity diagram untuk use case Menguras Bak



Gambar 3.20Activity Diagram Menguras Bak

3.1.5.4 Class Diagram

Diagram kelas atau class diagram menggambarkan struktur system dari segi pendefinisian kelas-kelas yang dibuat untuk membangun system. Gambar 3.23 di bawah ini adalah perancangan struktur system yang terdapat pada *game* Save Family from Chikungunya, yang digambarkan dengan class diagram.



Gambar 3.21 Class Diagram Save family From Chikungunya

Method dan atribut pada class diagram *game Save Family From Chikungunya* dapat dilihat pada table 3.13

Table 3.13 Class Diagram

Nama Class	Atribut	Method
Main	music = new sounds (2)	main()
	ending = new sounds(13)	pauseGame(): Object
	sedih = new sounds(15)	bkel(): Object
	bro = new sounds(18)	resumeGame(): Object
	darah: int = 3	setButtonPlay(): Object
	Lvl	loadingS(): Object
	curLv: int = 0	setAnimated(a): Object
	takeObj: int = 0	pl(e: MouseEvent): Object
	takeObj2: int = 0	direct(): Object
	takeObj3: int = 0	loadingFinish(): Object
	Hd	setGame1(): Object
	score: int = 0	setgame2(): Object
	mainHolder: Sprite = new Sprite	setGame3(): Object
	press_left: Boolean = false	setgame2(): Object
	press_right: Boolean = false	setGame3(): Object
	press_up: Boolean = false	setgame4(): Object
	press_down: Boolean = false	tembak(event: event): Object
	press_space: Boolean = false	tembak2(event: Event): Object
	door: Boolean = false	wini(event: Event): Object
	min: int = 0	setButtonGo(event: Event): Object
	onPause: Boolean = false	setButtonGo2(event: Event): Object
		setButtonGo3(event: Event): Object
		setButtonPembuat(event: Event): Object
		setButtonKontrol(event: Event): Object
		bBack(event: Event): Object
		nx(event: Event): Object
		setButtonQuit(event: Event): Object
		setButtonGoScore(event: Event): Object
		setPageMenu(): Object

		setInstruc1(): Object
		setIntruc2(): Object
		setInstruc3(): Object
		gogo(event: Event): Object
		setGameOver(): Object
		setGameFinish(): Object
		setGameWin(): Object
		clrGame(): Object
		key_down(event: KeyboardEvent): void
		key_down(event: KeyboardEvent): void
		key_up(event: KeyboardEvent): void
		tambahScore(a): Object
		kurangDarah(): Object
		setObj(): Object
		setObj2(): Object
		setObj3(): Object
		setObjs(): Object
		setObjs2(): Object
		serObjs3(): Object
		kurangSpray(a): Object
		setTimes(a): Object
		sFocus(): Object
Level	lvlHolder: Sprite = new Sprite	Level(obj)
	enemyHolder: Sprite = new Sprite	build(): Object
	bgHolder: Sprite = new Sprite	initAstar()
	hudHolder: Sprite = new Sprite	hapus(a, b): Object
	blockHolder: Sprite = new Sprite	setCoor(a, X, Y): Object
	infoHolder: Sprite = new Sprite	addNyamuk(): Object
	alatHolder: Sprite = new Sprite	addBos(): Object
	objHolder: Sprite = new Sprite	jalanRaja(): Object
	astarMapHolder: Sprite = new Sprite	checker(a): Object
	bosHolder: Sprite = new Sprite	addSampah(): Object
	heroHolder: MovieClip = new MovieClip	addTong(): Object
	row: int = 0	convertToHHMMSS(seconds: Number): String:Object
	p: Point = newPoint(0,0)	detik(e: TimerEvent): void
	infos: int = 0	createLvl1(): void
	Hero	createLvl2(): void
	heroMv	createLvl3(): void
	Bos	createLvl4(): void
	Audio	setOutro(event: TimerEvent): Object
	theRoot	delAsset(): Object
	Bak	setEndLvl(): void
	point: Array = new Array	setDoneLvl(): void
	Timex	
	coorSampah: Array = new Array()	
	coorEnemy: Array = new Array()	
	coorObj: Array = new Array()	
	timer: Timer = new Timer (3000)	
	mhero: Ibro	

Hud	onpause: Boolean = false	Hud(a)
	Holder	lanjut(event: MouseEvent): Object
		keluar(event: MouseEvent): Object
		pa(event: MouseEvent): Object
		setDarah(a): Object
		setL(a): Object
		setObj(a): Object
		setObj2(a): Object
		setObj3(a): Object
		setScore(a): Object
		setSpray(a): Object
		setKoordinat(a, b): Object
Sounds	audio: Sound	Sounds(code)
	channel: SoundChannel	playLoop(): Void
	isPlay: Boolean	playIt(): Void
		onComplete(event: Event): void
		stopSound(): void
Sampah	take = new Sounds(50	Sampah(): Object
		ambilSampah(obj): Object
Lobang	take = new Sounds(5)	Lobang()
		ambilLobang(obj): Object
Ibro	selesai: Boolean = false	Ibro(lv): void
	theHolder	netral(event: TimerEvent): void
	sprayBottle = 100	cekB1()
	dptInfo: int = 0	cek()
	dptSampah: int = 0	cekKenaB(): Object
	dptobj: int = 0	cekKena()
	dptAlat: int = 0	jalanKeDepan()
	up: Boolean = false	jalanKeBelakang()
	hitDelay: Timer = new Timer(2000)	control(event: Event): void
	hitlonc: Timer = new Timer(1000)	kurangDarah(): Object
	soundSem = new Sounds(17)	
	soundKena = new Sounds(6)	
	leftBumpPoint: Point = new Point(-70,0)	
	rightBumpPoint = new Point(0,0)	
	upBumpPoint: Point = new Point(0,-20)	
	downBumpPoint: Point = new Point(0,125)	
	p: int = 0	
	stop: Boolean = false	
	shoot: Boolean = false	
	Player	
	scrollX = 0	
	scrollY = 0	
	xSpeed: Number = 0	
	ySpeed: Number = 0	
	speedConstant: Number = 2	
	frictionConstant: Number = 0.9	
	gravityConstant: Number = 1.8	
	jumpConstant: Number = -48	
	maxSpeedConstant: Number = 6	

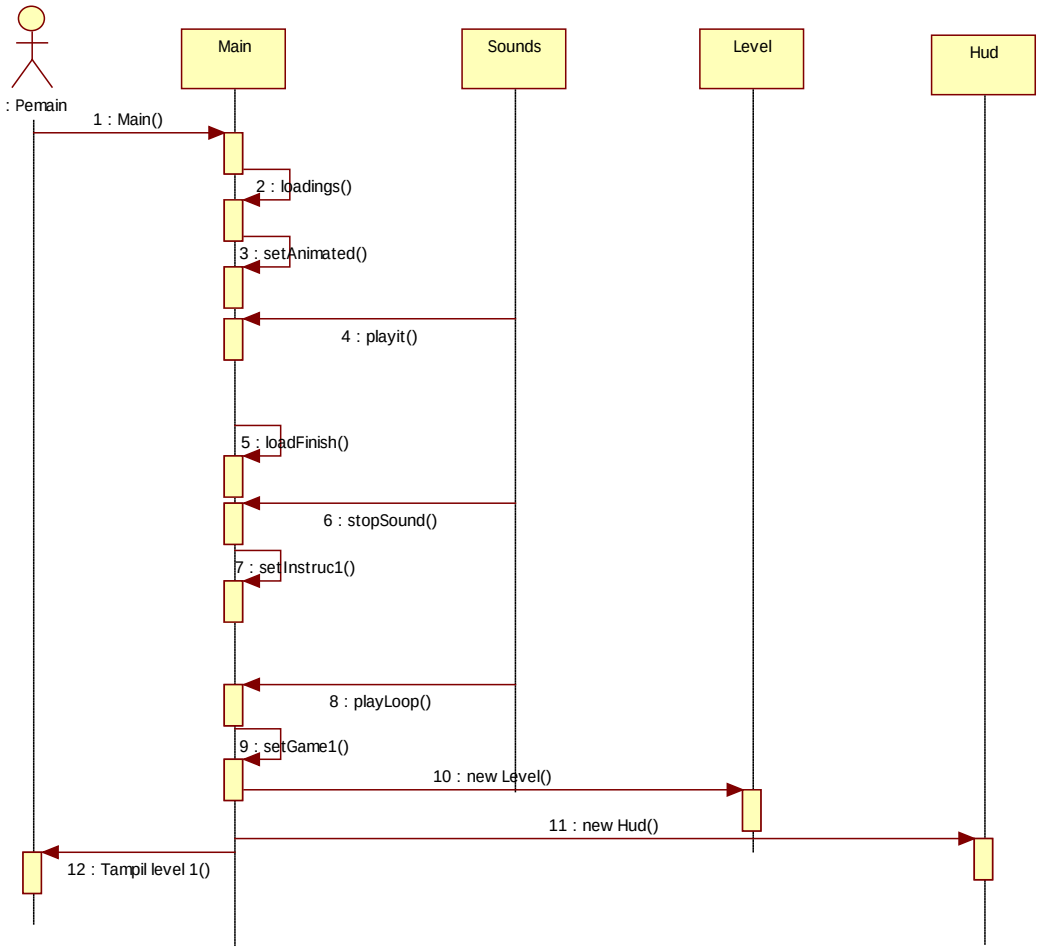
	doubleJumpReady: Boolean = false	
	upReleasedInAir: Boolean = false	
	leftBumping: Boolean = false	
	rightBumping: Boolean = false	
	upBumping: Boolean = false	
	downBumping: Boolean = false	
	S	
RajaNyamuk	loc: Array = newArray	RajaNyamuk(a)
	xSpeed: Number	getCurrentPos()
	tenagaOk = True	getAstarMap(a)
	ySpeed: Number	check(a)
	maxSpeed: Number	walkAstar()
	health: int = 1000	cariPlayer()
	tenaga: int = 1000	tenagaG(a): Object
	arah: String	daragG(): Object
	mati = false	kurangDarah(a): Object
	Damage	jalanAstar(X, Y)
	scan: Boolean = true	hapus()
	Lvl	
	Obj	
	Par	
	kena: Boolean = false	
	count: Number = 0	
	max2Speed: Number	
	timers: Timer = new Timer(1500)	
	lifeLength	
	Mc	
	sec: int = 0	
Astar	cur: int	Astar(par, i, c)
	des: int	getXy(): Object
	starts: int	setBobot(): Object
	path: Array = new Array	chk(1, xx, yy)
	len: int-38	getTetangga(i)
	opened: Array = new Array	heuristic(a, b)
	lvlArray: Array = new Array	pop(): Object
	varHolder: Array = new Array	heapop(array, cmp): Object
	Holder	backtrace(a): Object
	endN	pathFinder(): Object
	Pars	push(a): Object
	defaultCmp = 0	heappush(array, item, cmp): Object
		listUpdate(a): Object
		update(array, item, cmp): Object
		cmpX(a, b): Object
		si(array, startpos, pos, cmp): Object
		up(array, pos, cmp): Object
Nodes	tetangga: Array = new Array	Nodes(a: Number, xss, yss, wk)
	id: int	setTetangga(d): Object
	opened: Boolean = false	
	closes: Boolean = false	
	f: Number = 0	
	g: Number = 0	
	h: Number = 0	
	xs: Number = 0	

	ys: Number = 0	
	cost: Number = 1	
	par = null	
	walkable = true	

3.1.5.5 Sequence Diagram

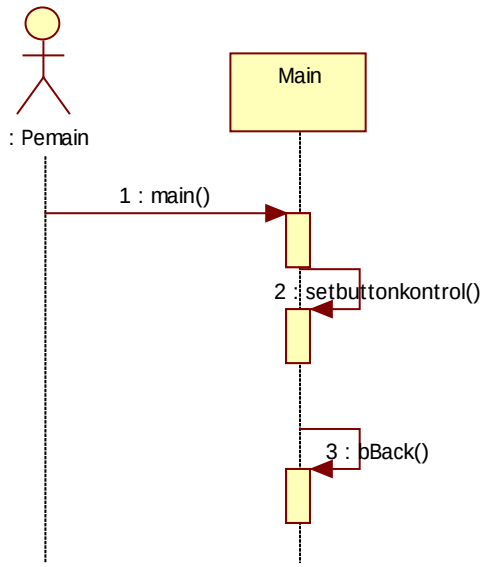
Sequence Diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek.

a. Main



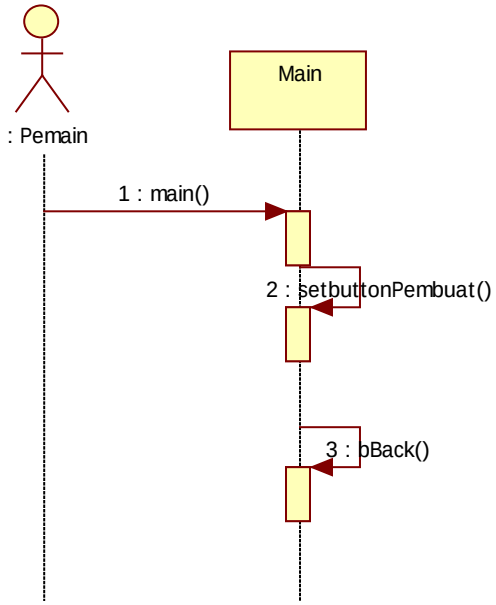
Gambar 3.22 Sequence Diagram Main

b. Memilih Cara Bermain



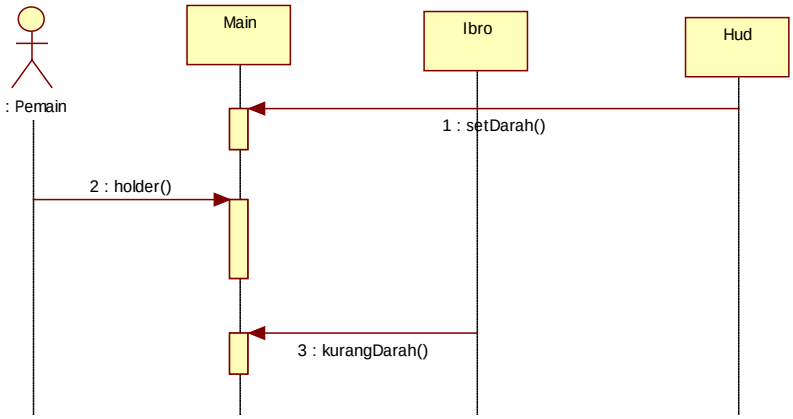
Gambar 3.23 Sequence Diagram Memilih Cara Bermain

c. Memilih Pembuat



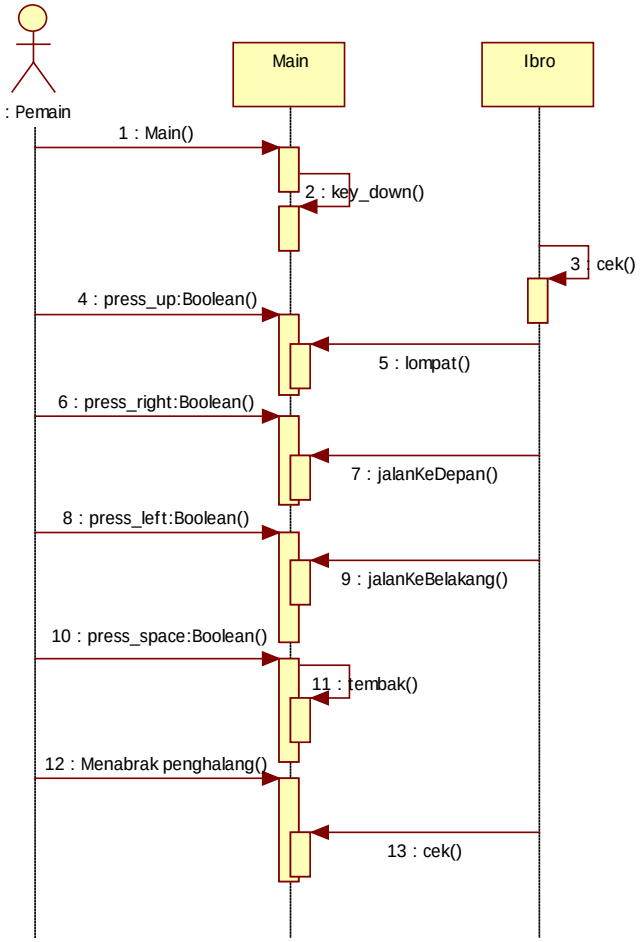
Gambar 3.24 Sequence Diagram Memilih Pembuat

d. Menghitung Nyawa



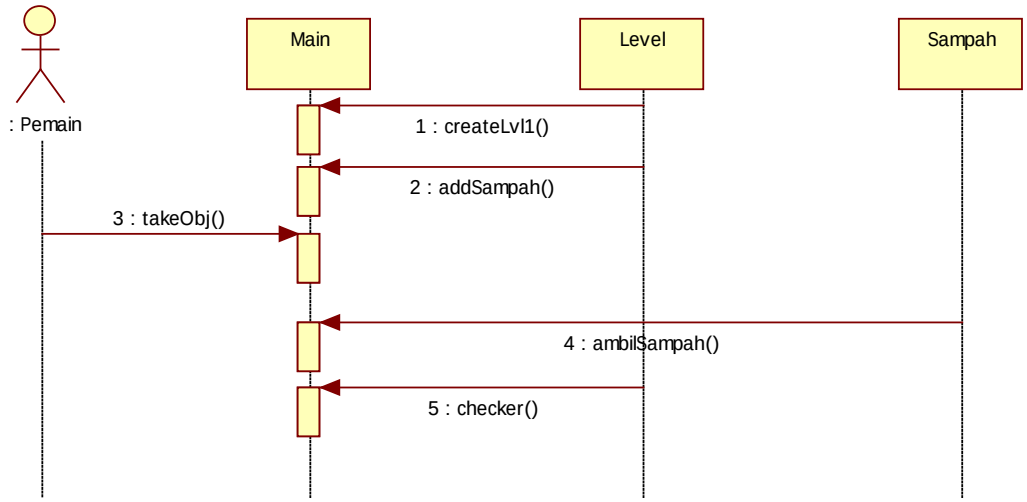
Gambar 3.25 Sequence Menghitung Nyawa

e. Pergerakan Karakter



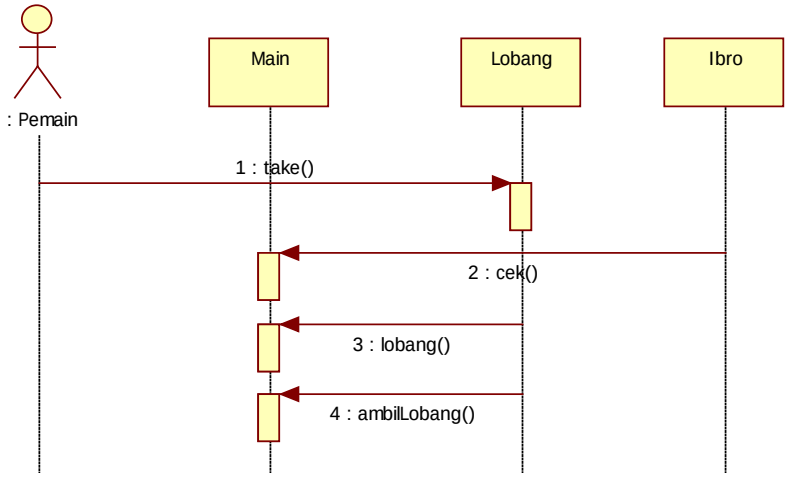
Gambar 3.26 Sequnce Diagram Pergerakan Karakter

f. Mengambil Sampah



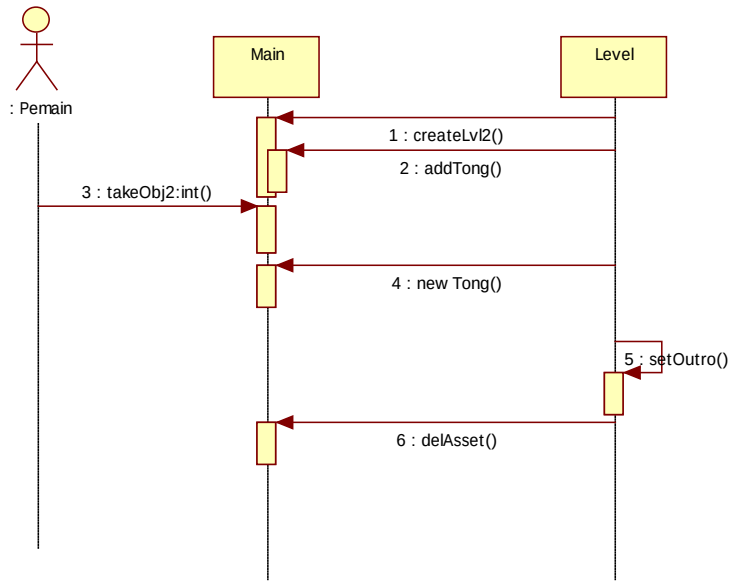
Gambar 3.27 Sequence Diagram Mengambil Sampah

g. Mengubur Sampah



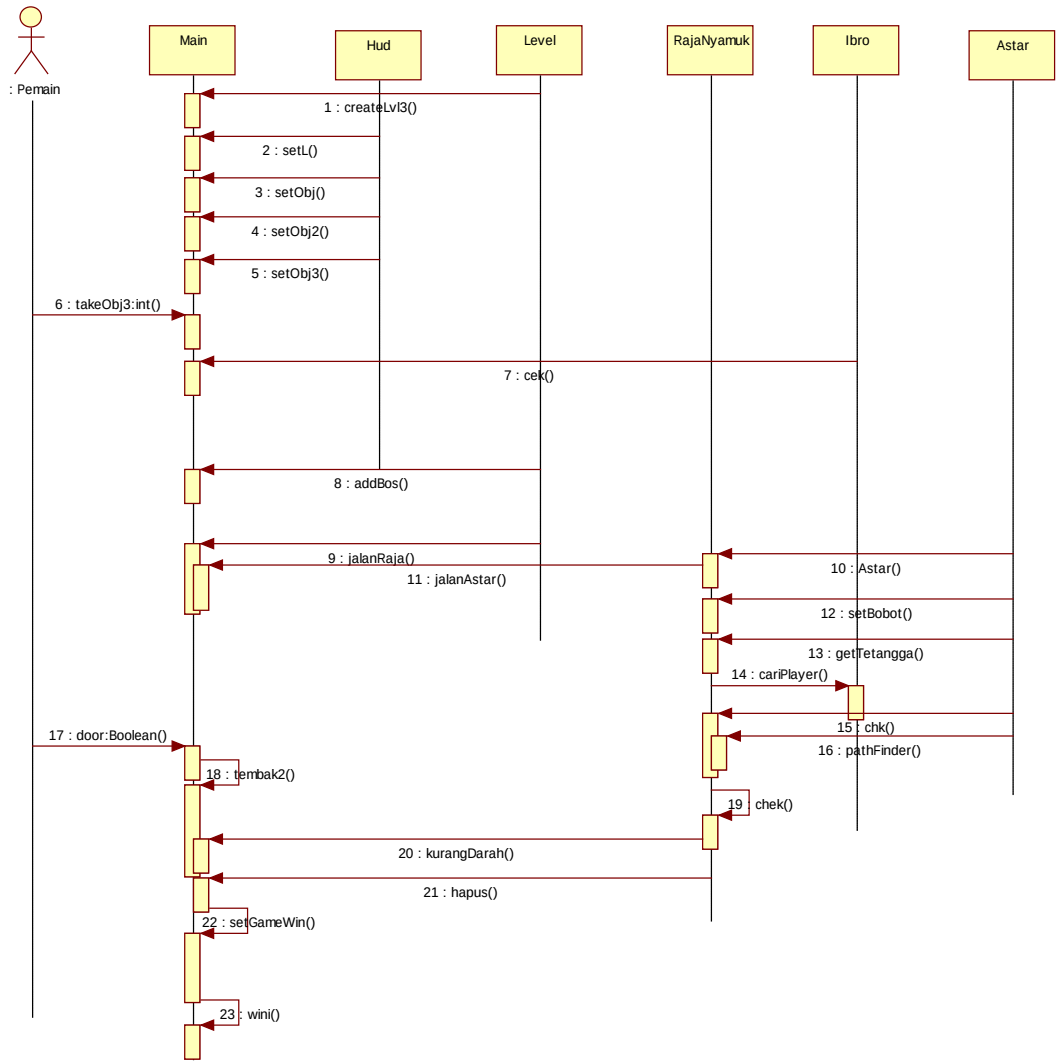
Gambar 3.28 Sequence Diagram Mengubur Sampah

h. Menutup Ember Air



Gambar 3.29 Sequence Menutup Ember Air

i. Menguras Bak



Gambar 3.30 Sequence Menguras Bak

3.2 Perancangan Sistem

3.2.1 Pengenalan Karakter

Pengenalan karakter merupakan pembahasan mengenai karakter-karakter yang terlibat dalam *game* Save Family From Chikungunya. Pengenalan karakter dapat dilihat pada Tabel 3.14 Di bawah ini.

Table 3.14 Pengenalan Karakter

No	Nama	Gambar	Keterangan
1	Ibro (Karakter Utama)		a. Karakter ini merupakan karakter yang akan digunakan oleh pengguna untuk memainkan permainan. b. Karakter ini digambarkan sebagai seorang laki-laki berusia 7 tahun.
2	Nyamuk (Karakter musuh)		a. Karakter musuh merupakan karakter yang terdapat setiap level, karakter ini harus dihindari atau dikalahkan oleh pemain apabila pemain ingin melanjutkan ke level selanjutnya. b. Karakter nyamuk yang banyak berkeliaran, menghalangi pemain dan dapat menggigit pemain.
3	King Mosquito (Karakter raja musuh		a. Karakter raja nyamuk merupakan karakter yang terdapat di level ke 3 dan paling kuat. b. karakter ini akan selalu mengejar dan menyerang pemain..
4	Sikat Pembersih		a. Konten ini harus diambil oleh pemain untuk membersihkan bak mandi. b. Konten ini ada pada misi level 3
5	Bak Mandi		a. Konten ini harus dikuras oleh pemain agar bisa bertemu dengan Raja nyamuk. Pemain menguras bak mandi dengan syarat mendapatkan alat pembersihnya terlebih dahulu berupa sikat pembersih. b. Misi ini hanya ada pada level 3

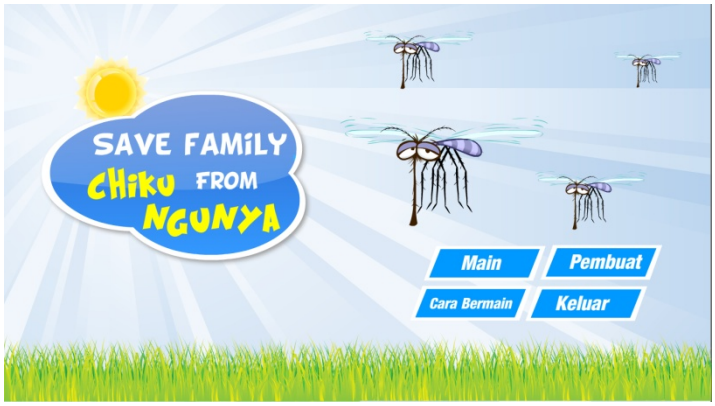
6	Bungkusan sampah		a. Konten yang harus di kumpulkan, dan dikubur. b. Jika pemain tidak mengubur semua bungkusan sampah dalam waktu yang diberikan maka dianggap kalah.
7	Tempat Ait		a. Konten ini harus ditutup oleh pemain terdapat. b. Jika semua tempat air tidak ditutup dalam waktu yang disediakan maka pemain dianggap kalah.

3.2.2 Story Board

Storyboard adalah sketsa gambar yang disusun berurutan sesuai dengan naskah, dengan storyboard kita dapat menyampaikan ide cerita kita kepada orang lain dengan lebih mudah, karena kita dapat menggiring khayalan seseorang mengikuti gambar-gambar yang tersaji, sehingga menghasilkan persepsi yang sama pad ide cerita kita.

Berikut ini adalah *storyboard* dari *Game* "Save Family From Chikungunya"

a. *Storyboard* Menu Utama



Gambar 3.31 Storyboard Menu Utama

b. *Storyboard* Ilustrasi Cerita



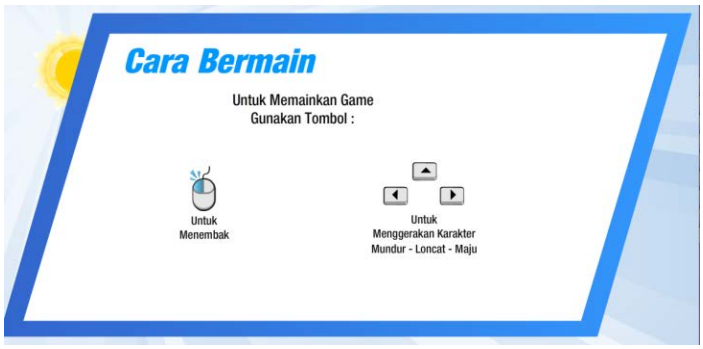
Gambar 3.32 Storyboard Tampilan Cerita

c. *Storyboard* Ilustrasi Cerita



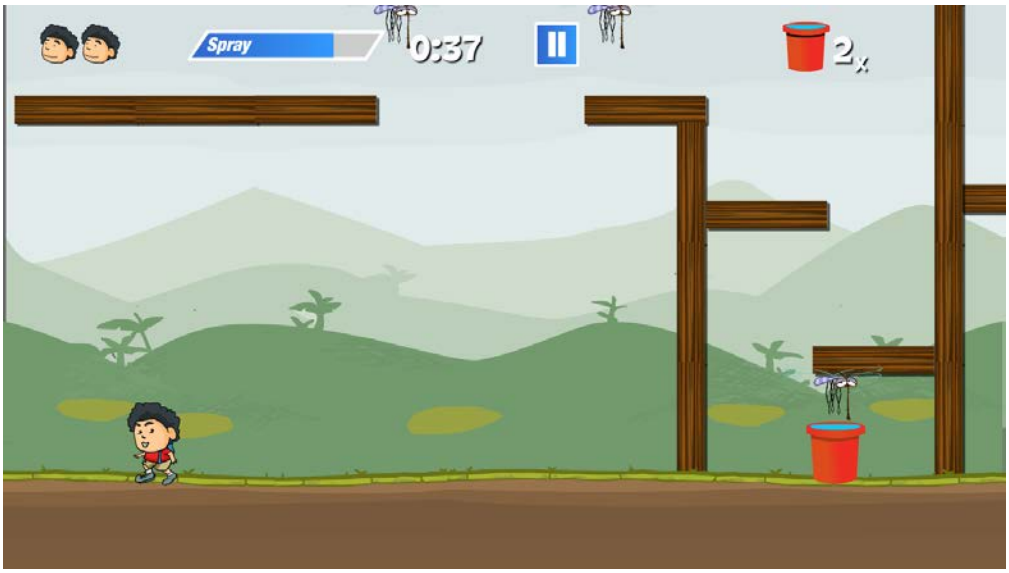
Gambar 3.33 Storyboard Tampilan Cerita

d. *Storyboard* Cara bermain



Gambar 3.34 Storyboard Cara Bermain

e. *Storyboard* Permainan



Gambar 3.35 Storyboard Permainan

f. *Storyboard* Misi Menimbun Sampah



Gambar 3.36 Storyboard Misi Menimbun Sampah

g. *Storyboard* Misi Menutup Ember Air



Gambar 3.37 Storyboard Misi Menutup Ember Air

h. *Storyboard* Misi Menguras Bak



Gambar 3.38 Storyboard Misi Menguras Bak

i. *Storyboard* Informasi Pencegahan dan Misi



Gambar 3.39 Storyboard Informasi Pencegahan dan Misi

g. *Storyboard* Tampilan Gagal



Gambar 3.40 Storyboard Tampilan Gagal

h. *Storyboard* Tampilan Berhasil



Gambar 3.41 Storyboard Tampilan Berhasil

3.2.3 Perancangan Struktur Menu

Perancangan menu dibuat dengan harapan agar pengguna dapat menggunakan *Game* dengan mudah tanpa kesulitan, sehingga memudahkan pemakai dalam memilih menu aplikasi yang sedang berjalan. Struktur menu dapat dilihat pada Gambar 3.28 Di bawah ini.



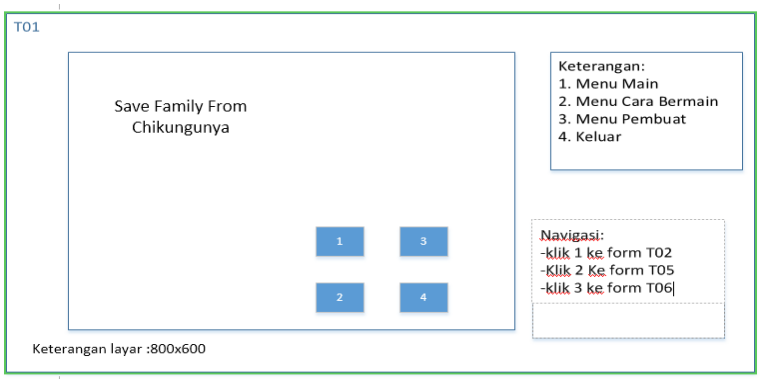
Gambar 3.42 Struktur Menu *Game Save Family From Chikungunya*

1. Main (Memulai Permainan) : submenu untuk memulai *game* Save Family From Chikungunya.
2. Cara bermain : submenu untuk melihat cara bermain.
3. Pembuat : submenu untuk melihat info pembuat
4. Keluar : submenu untuk keluar dari aplikasi *game Save Family From Chikungunya.*

3.2.4 Perancangan Antar Muka

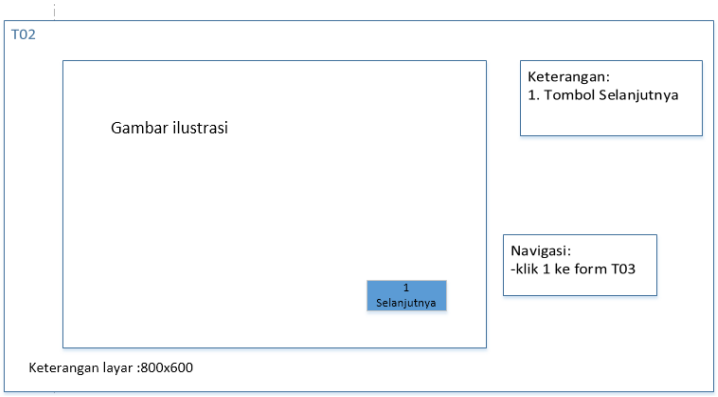
Antarmuka (*interface)* merupakan bentuk tampilan dari program yang tampil pada layar computer yang bertujuan memberikan gambaran tentang aplikasi yang akan dibangun, sehingga akan memudahkan mengimplementasikan aplikasi sesuai ukuran layar dan memudahkan dalam pembuatan aplikasi.

a. Rancangan Antarmuka Menu Utama



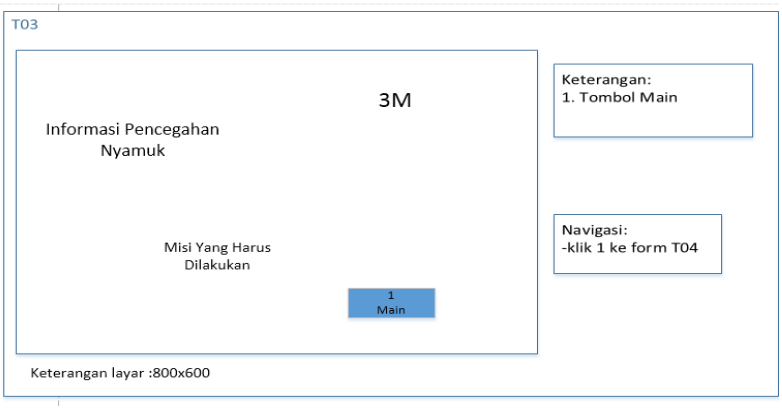
Gambar 3.43 Rancangan Antarmuka Menu Utama

b. Rancangan Antarmuka Menu ilustrasi cerita



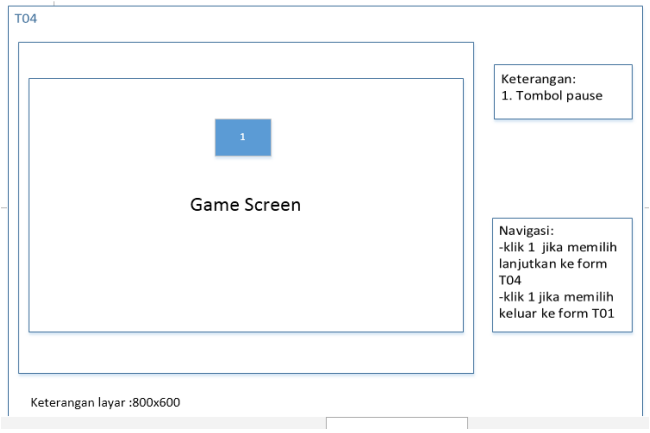
Gambar 3.44 Rancangan Antarmuka Ilustrasi Cerita

c. Rancangan Antarmuka Menu Informasi Pencegahan dan Misi



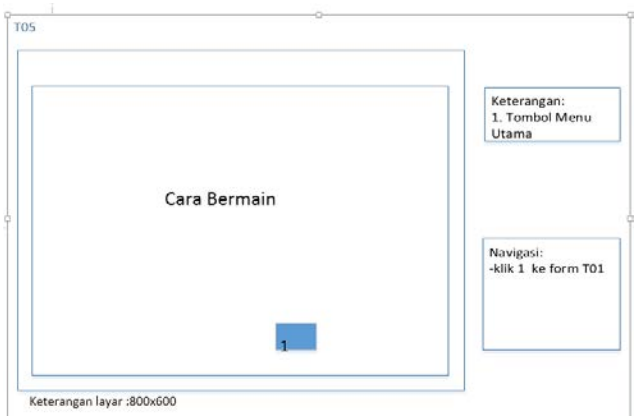
Gambar 3.45 Rancangan Antarmuka Menu Informasi dan Misi

d. Rancangan Antarmuka Menu Memainkan *Game*



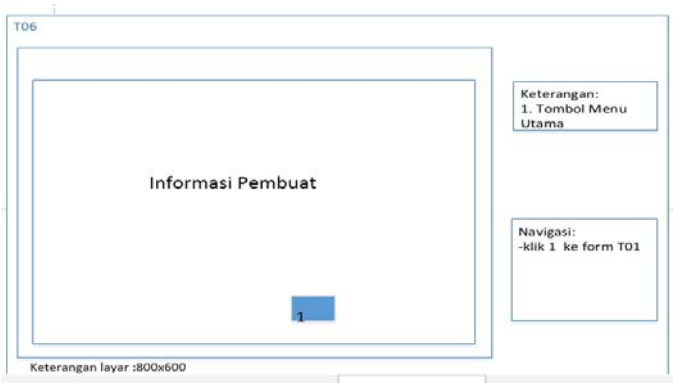
Gambar 3.46 Rancangan Antarmuka Menu Memainkan *Game*

e. Rancangan Antarmuka Menu Cara Bermain



Gambar 3.47 Rancangan Antarmuka Menu Cara Bermain

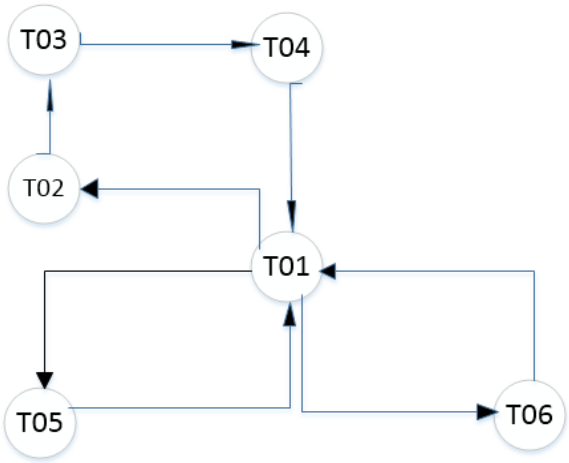
f. Rancangan Antarmuka Menu Pembuat



Gambar 3.48 Rancangan Antarmuka Menu Pembuat

3.2.5 Jaringan Semantik

Untuk mengetahui keterhubungan dari setiap tampilan yang dibuat dapat menggunakan jaringan semantic. Berikut adalah jaringan semantic pada game Save Family From Chikungunya**.**



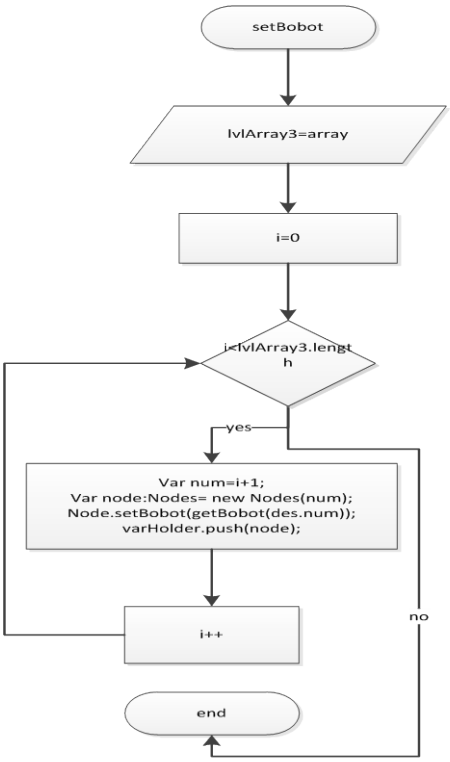
Gambar 3.49 Jaringan Semantik

3.2.6 Perancangan Metode

Perancangan metode merupakan suatu proses untuk menjelaskan proses maupun subproses yang terjadi pada sequence yang sudah dijelaskan. Maka dari itu, pada pembangunan *game* ini akan dijelaskan secara terurut yaitu metode yang digunakan sesuai dengan sequence yang telah dijelaskan untuk lebih dimengerti.

3.2.6.1 Perancangan Metode *A**: setBobot()

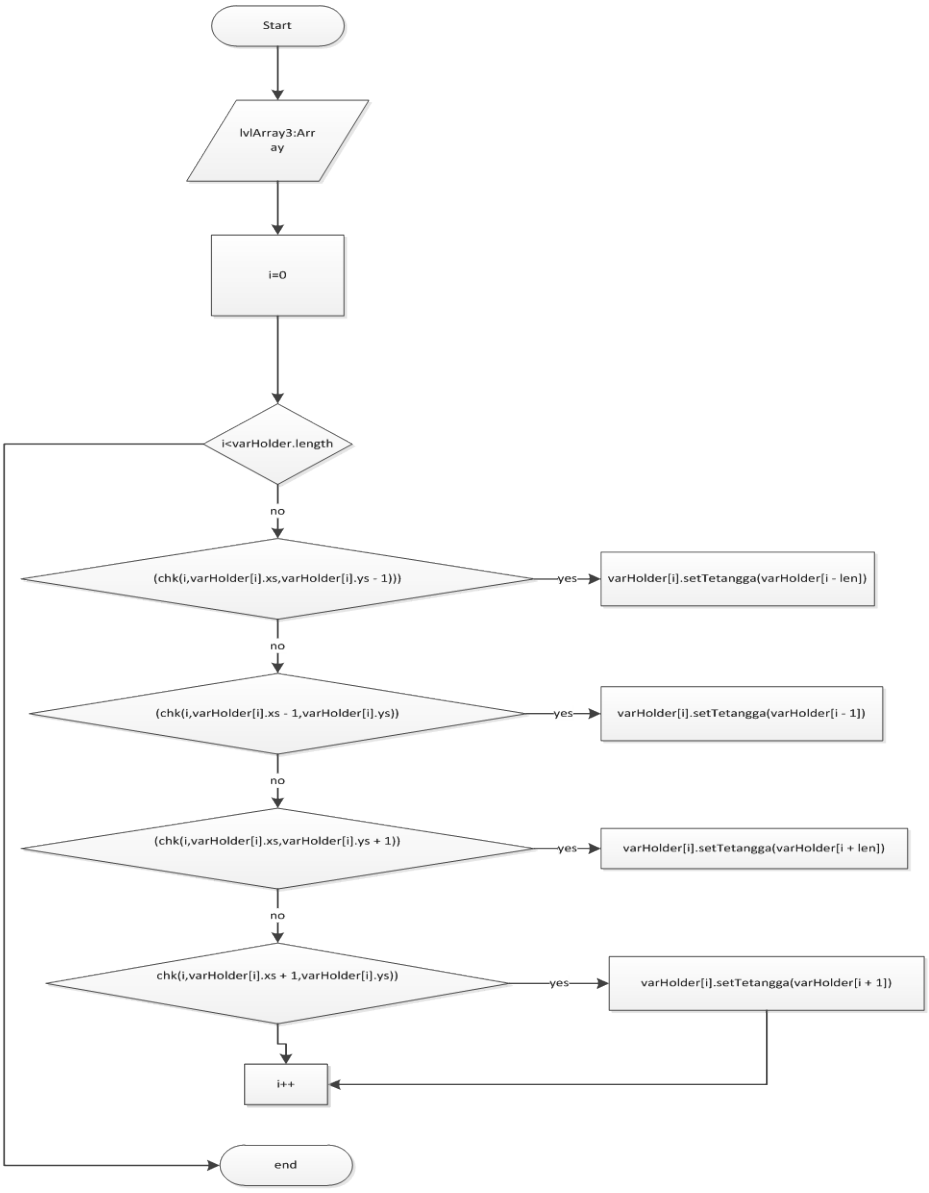
Metode ini dijalankan pada saat Level : createLvl4()dipanggil. Metode ini digunakan untuk menginisialisasi node-node dan memberikan bobot di setiap nodenya dengan perbandingan titik akhir yang akan diperiksa algoritma A*.



Gambar 3.50 Metode A*: setBobot()

3.2.6.2 Perancangan Metode *A**: getTetangga ()

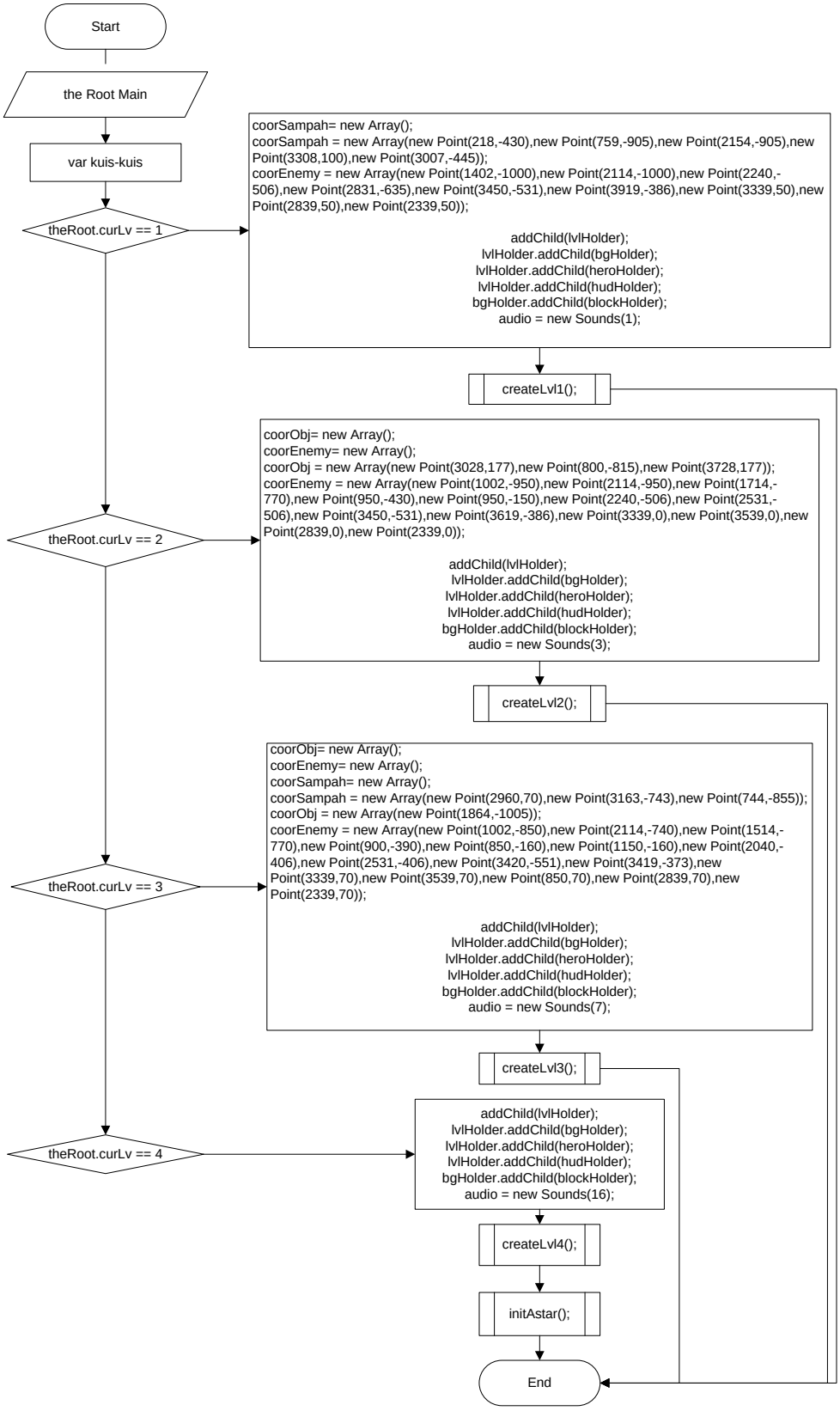
Metode ini dijalankan pada saat Level : createLvl4() dipanggil. Metode ini digunakan untuk menginisialisasi tetangga pada setiap node, sehingga node-node akan saling berhubungan.



Gambar 3.51 Metode A*: getTetangga()

3.2.6.3 Perancangan Metode Level : build()

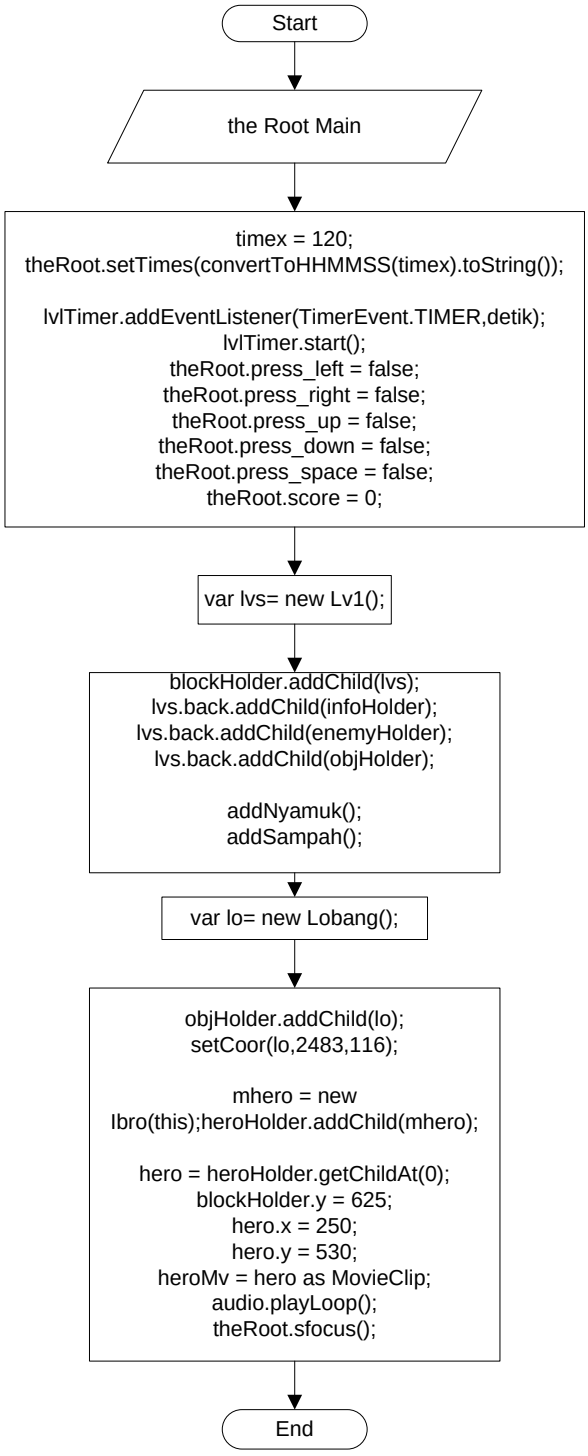
Metode ini digunakan ketika kedaan class Level di inisialisasi. Metode ini terpanggil pada constructor Level. Metode ini digunakan untuk menginisialisasi keadaan level yang akan ditampilkan



Gambar 3.52 Perancangan Method build()

3.2.6.4 Perancangan Metode *Level*: createLvl1 ()

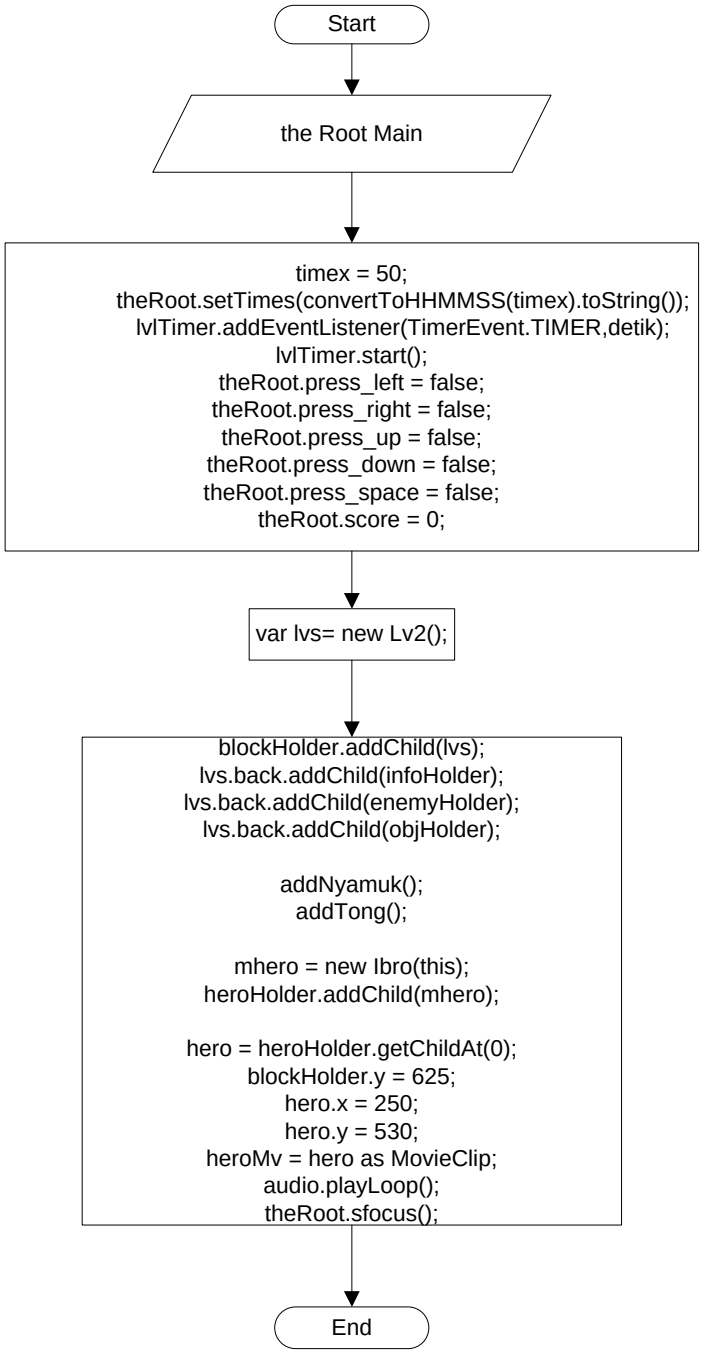
Metode ini digunakan untuk membuat level 1. Dengan acuan dari metode Level: build().



Gambar 3.53 Metode *Level:createLvl1()*

3.2.6.5 Perancangan Metode *Level*: createLvl2 ()

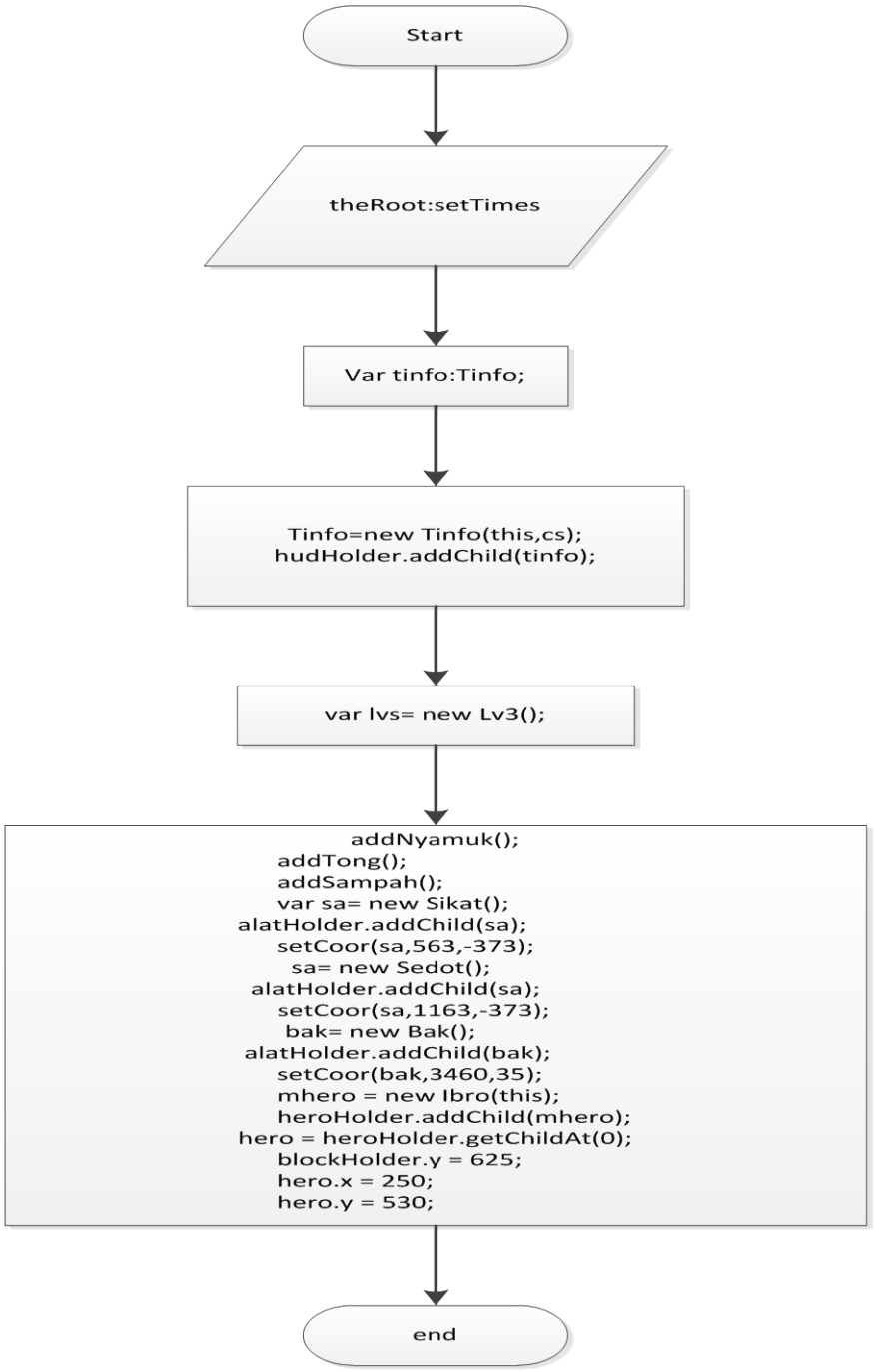
Metode ini digunakan untuk membuat level 2. Dengan acuan dari metode Level: build().



Gambar 3.54 Metode *Level: createLvl2()*

3.2.6.6 Perancangan Metode *Level*: createLvl3 ()

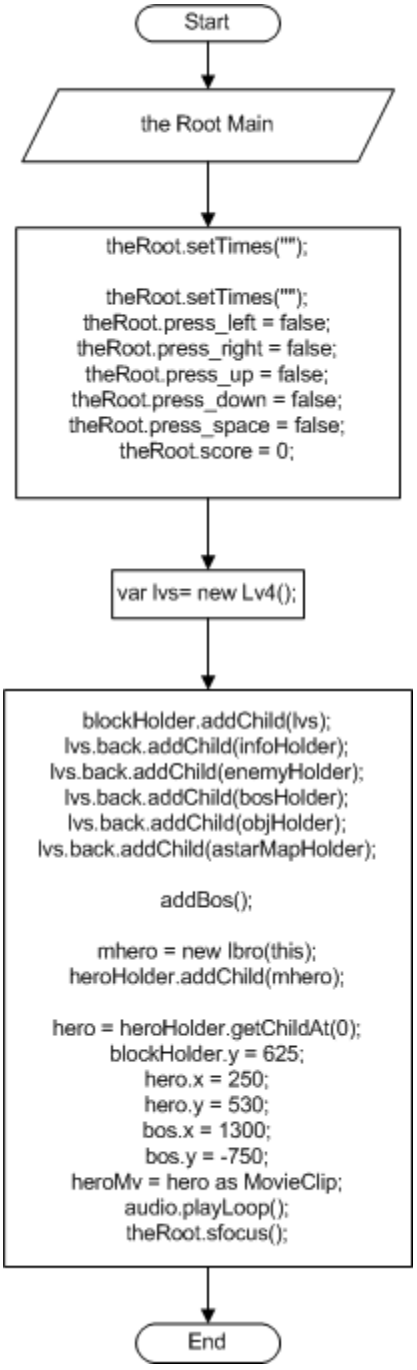
Metode ini digunakan untuk membuat level 3. Dengan acuan dari metode Level: build().



Gambar 3.55 Metode *Level: createLvl3()*

3.2.6.7 Perancangan Metode *Level*: createLvl4 ()

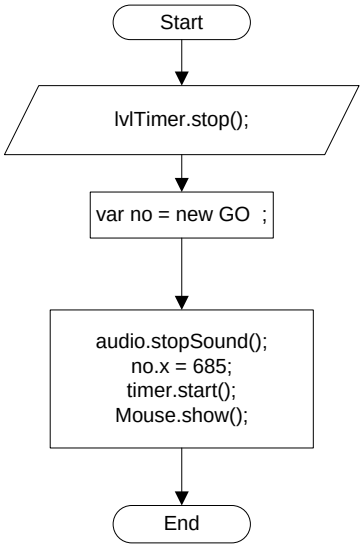
Metode ini digunakan untuk membuat level 4. Dengan acuan dari metode Level: build().



Gambar 3.56 Metode *Level: createlvl4()*

3.2.6.8 Perancangan Metode *Level :* setEndLvl()

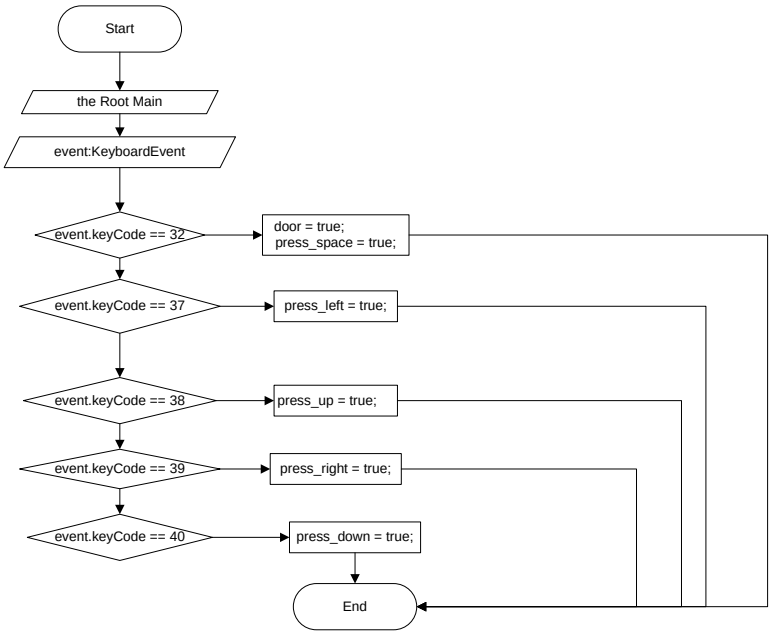
Metode ini digunakan ketika setiap level permainan selesai. Dengan acuan dari metode Level: build().



Gambar 3.57 Metode *Level: setEndLvl()*

3.2.6.9 Perancangan Metode *Main :* key_down(event:KeyboardEvent)

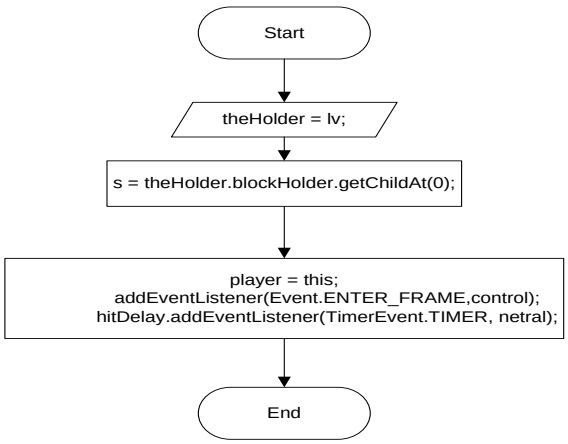
Metode ini digunakan untuk menggerakan karakter saat user menekan tombol pada *keyboard*.



Gambar 3.58 Metode *Main: key_down(event:KeyboardEvent)*

3.2.6.10 Perancangan Metode *Ibro:* Ibro(lv)

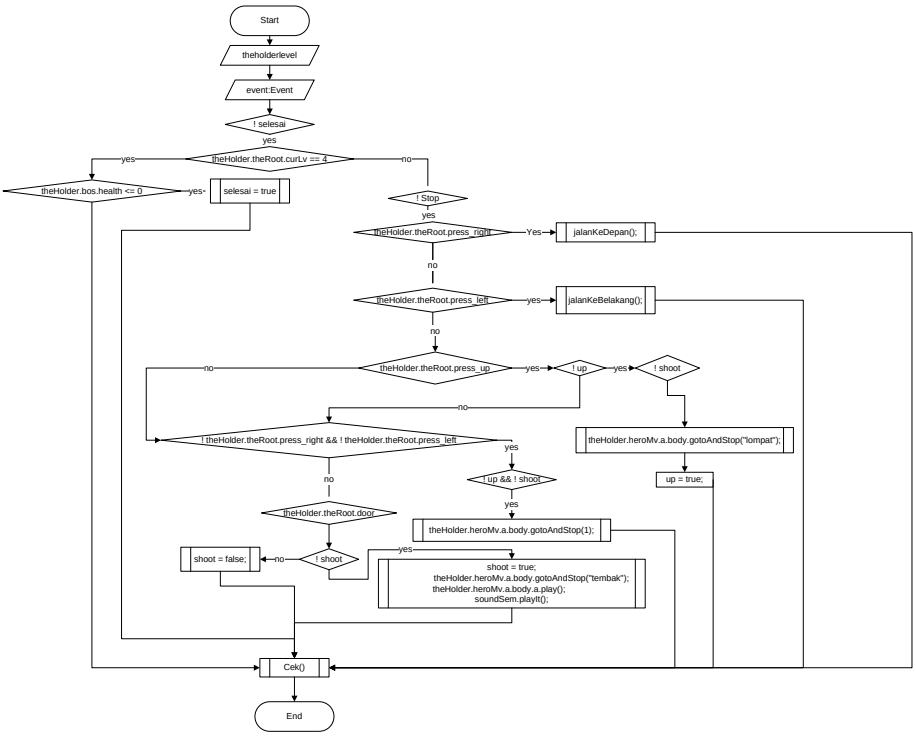
Metode ini digunakan saat memunculkan karakter ibro pada permainan. Dengan acuan dari metode Level: build().



Gambar 3.59 Metode *Ibro:Ibro(lv)*

3.2.6.11 Perancangan Metode *Ibro:* control(event:Event)

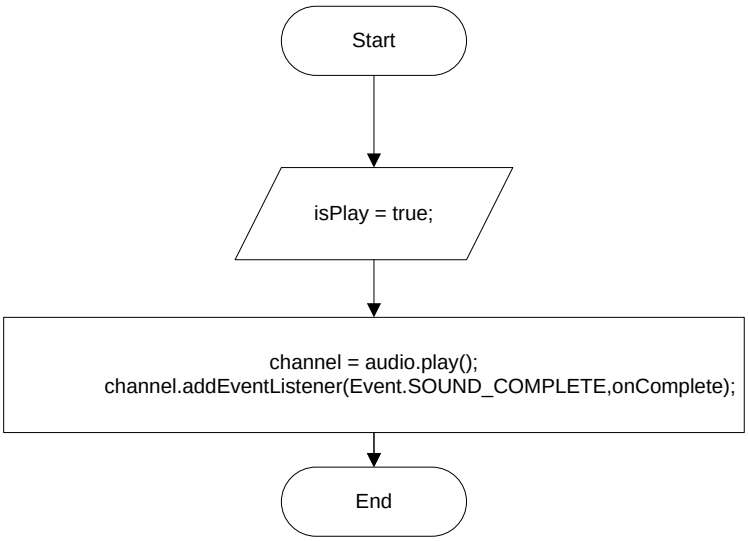
Metode ini dipanggil ketika sistem sudah menginisialisasi karakter Ibro dan sudah memberi akses kepada user atau pemain untuk menggerakan karakter. Metode ini dijelaskan kembali dari sebelumnya pada sequence Menggerakan karakter.



Gambar 3.60 Metode *Ibro: control(event:Event)*

3.2.6.12 Perancangan Metode Sounds:*playLoop()*

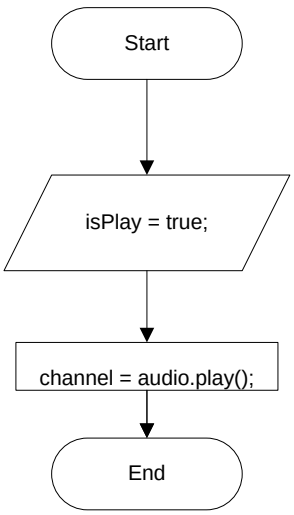
Metode ini dipanggil ketika sistem mulai menginisialisasi permainan, mulai dari animasi sampai stage demi stage selesai.



Gambar 3.61 Metode Sounds:playloop()

3.2.6.13 Perancangan Metode Sounds:*playIt()*

Metode ini dipanggil ketika sistem menampilkan ilustrasi cerita.



Gambar 3.62 Metode Sounda:playIt()