BAB 3
ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Sistem

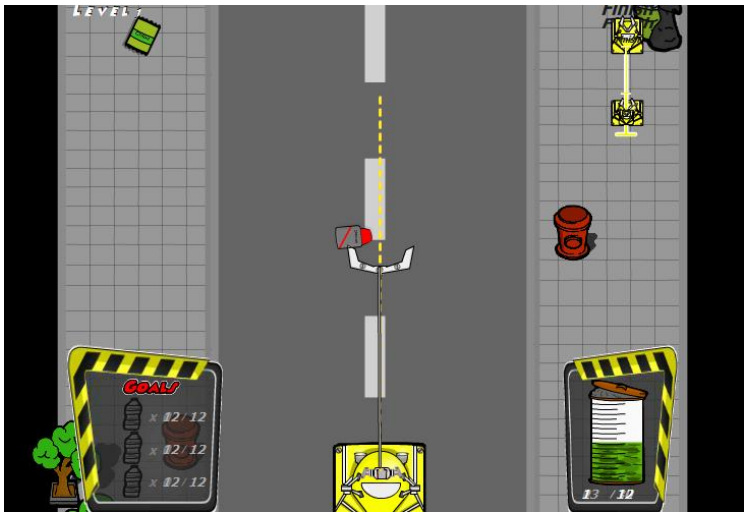
Analisis sistem merupakan proses penguraian konsep ke dalam bagian-bagian yang lebih sederhana. Pada analisis sistem akan mencakup berbagai macam analisis yang berhubungan dengan *game* yang akan dibangun seperti analisis masalah, *storyline, gameplay,* analisis kecerdasan buatan yang dipakai, analisis kebutuhan fungsional dan 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. Tujuan dari analisis ini yaitu mendapatkan konsep permainan dan perbandingan dari *game* yang telah ada.

3.1.1.1 *Game Garbage Grab*

Game Garbage Grab merupakan *game* dengan jenis *casual games*, terdapat di sebuah website *flash game* portal yang menggunakan *platform flash* (swf). *Game* ini mempunyai *gameplay* mengumpulkan sampah dengan alat pengait pada truknya. Karakter utama dalam *game* ini adalah truk sampah dan grafis yang digunakan adalah 2 dimensi. Berikut adalah tampilan dari *game garbage grab* yang dapat dilihat pada Gambar 3.1.



Gambar 3.1 Tampilan permainan *game garbage grab*

1. **Gameplay**

Pemain akan mengendalikan pengait truk sampah, truk ini bergerak tetap ke depan tanpa bisa mundur ataupun belok. Tujuan pada *game* ini adalah mengumpulkan sampah sesuai dengan target. Cara pengambilan sampah yaitu dengan alat pengait pada truk sampah yang berotasi ke kiri dan kanan. Untuk itu dibutuhkan keakuratan dan waktu yang tepat agar pengait dapat meraih sampah. Setiap sampah yang diambil akan menghasilkan 20 poin. Pada level 1 tujuannya hanya mengumpulkan 10 buah sampah, setelah target sampah terpenuhi dan truk sudah mencapai akhir dari jalan maka pemain memenangkan level 1 dan bisa melanjutkan ke level 2. Pada level 1 ini tidak ada rintangan. Di level 2 target sampah yang dikumpulkan bertambah dan beragam, pemain harus mengumpulkan sampah sebanyak 30 dan mengambil 2 jenis sampah tertentu sesuai dengan jumlah target dan terdapat rintangan serta item khusus. Rintangan pada yang ada di level 2 ini yaitu item yang apabila diambil akan menyebabkan kerusakan pada pengait sampah sehingga sulit untuk meraih sampah. Sedangkan item khusus yang lain apabila diambil akan menambah keakurasian dalam meraih sampah. Jika pemain tidak dapat memenuhi target maka pemain tidak dapat memainkan level selanjutnya dan harus mengulangi dari awal permainan level pertama. Pada level selanjutnya target yang harus dicapai semakin banyak dan beragam.

2. Skenario *Game*

Skenario dalam *game garbage grab* dapat dilihat alur permainannya sebagai berikut:

1. Truk bergerak otomatis ke depan dengan alat pengait yang berotasi ke kiri dan kanan.
2. Sampah berada di badan jalan dan sekitaran jalan.
3. Ambil sampah dengan alat pengait pada truk.
4. Klik pada layar ketika alat pengait sudah dalam arah yang tepat.
5. Sampah akan terambil apabila arah alat pengait tepat pada posisi sampah.
6. Sampah tidak akan terambil apabila arah alat pengait tidak tepat pada posisi sampah.
7. Kumpulkan sampah sesuai dengan jumlah target.
8. Permainan berakhir ketika truk sudah mencapai akhir jalan.
9. Apabila jumlah sampah yang didapat minimal sesuai dengan target maka permainan berhasil diselesaikan dan dapat melanjutkan ke level selanjutnya.
10. Apabila jumlah sampah yang didapat kurang dari target maka permainan gagal dan harus mengulangi dari level pertama.

3. Kekurangan Pada *Garbage Grab*

Pada *game Garbage Grab* ini terdapat beberapa kekurangan, yaitu diantaranya sebagai berikut:

1. Truk tidak dapat rusak atau hancur.
2. Tidak ada rintangan saat truk melaju.
3. Sampah tidak dibedakan menjadi organik atau anorganik.
4. Pada level 2 dan selanjutnya permainan menjadi terlalu sulit untuk diselesaikan karena jarak antara awal permainan sampai akhir terlalu dekat dengan target jumlah sampah yang banyak.
5. *Gameplay* hanya mengumpulkan sampah.

3.1.2 Analisis *Game* Yang Dibangun

Pada tahapan ini akan mendeskripsikan *game* yang akan dibangun. Pada bagian ini terdiri dari *storyline, gameplay,* jenis permainan, tingkat kesulitan, dan analisis *scoring.*

Tabel 3.1 Perbandingan *Game* Sejenis Dengan *Game* Yang Dibangun

	Game Garbage Grab	***Game* Yang Dibangun**
Genre	Casual	Casual
Desain Grafis	2D	2D
Karakter Utama	Truk	Truk
Pergerakan Karakter	Pergerakan karakter otomatis lurus ke depan, tanpa bisa berbelok atau mundur.	Pergerakan karakter digerakan oleh pemain. Bisa ke depan, belakang, belok kiri dan kanan.
Rintangan	Pengait sampah yang terus berotasi ke kiri dan kanan	Mobil berdatangan dari arah berlawanan. Waktu yang disediakan terbatas.
Target	Mengumpulkan sampah dan jenis sampah tertentu sesuai target.	Mengumpulkan sampah, mendaur ulang sampah menjadi produk.
Gameplay	Untuk menyelesaikan permainan pemain harus mengumpulkan jumlah sampah sesuai target. Cara mendapatkan sampah menggunakan pengait yang berotasi ke kiri dan kanan. Untuk itu dibutuhkan keakuratan dalam meraih sampah.	Untuk menyelesaikan permainan pemain harus mengumpulkan jenis sampah tertentu sesuai target. Kemudian sampah tersebut di daur ulang menjadi sejumlah produk.
Basis	Web	Desktop

3.1.2.1 *Gameplay*

Berikut adalah *gameplay* pada *game* yang akan dibangun:

1. Permainan pertama adalah mengumpulkan sampah yang ada di jalanan.
2. Pemain mengendalikan sebuah truk sampah yang dapat bergerak ke depan, kiri dan kanan.
3. Dapatkan sampah yang berada di jalanan.
4. Terdapat berbagai macam sampah yang terdiri dari jenis sampah organik dan non-organik.

5. Poin akan bertambah jika mengklik sampah sesuai jenisnya dan poin berkurang jika salah.
6. Waktu yang disediakan terbatas. Terdapat rintangan yaitu mobil yang datang dari arah berlawanan, jika menabrak maka permainan gagal.
7. Pemain menyelesaikan permainan jika sudah mengumpulkan poin sesuai target sebelum waktu habis.
8. Untuk menambahkan poin tambahan daur ulang sampah yang didapat dan jadikan produk
9. Terdapat beberapa produk yang dapat dibuat. Pilih produk yang akan dibuat, jika bahan yang dibutuhkan mencukupi maka akan muncul produk yang dibuat.
10. Jika target tidak tercapai maka permainan gagal dan harus mengulangi dari permainan pertama.

3.1.2.2 Jenis Permainan

Dalam *game* yang akan dibangun terdapat 2 jenis permainan dalam setiap levelnya, yaitu:

1. Mengumpulkan sampah

   Truk mengambil sampah yang berada di jalanan dan harus dengan benar membedakan jenis sampah untuk mendapatkan poin, waktu yang diberikan terbatas.
2. Mendaur ulang

   Sampah yang telah dikumpulkan harus di daur ulang menjadi produk. Permainan mendaur ulang adalah permainan tambahan untuk menambahkan poin dan dilakukan bersamaan dengan permainan mengumpulkan sampah.

3.1.2.3 Tingkat Kesulitan

Dalam *game* ini terdapat 2 level permainan yang harus diselesaikan, setiap level memiliki permainan dan kesulitan yang berbeda.

1. Level 1

   Truk harus mendapatkan sampah. Jenis sampah ada 3, target poin yang harus didapat adalah 400 dengan waktu yang diberikan 90 detik.

2. Level 2

   Jenis sampah bertambah menjadi 6 jenis sampah dan terdapat rintangan yaitu mobil yang berdatangan dari arah berlawanan. Poin yang harus didapat adalah 600.

3. Level 3

   Jenis sampah semakin beragam, pada level ini terdapat 9 jenis sampah. Mobil yang berdatangan semakin banyak. Poin yang harus didapat adalah 800.

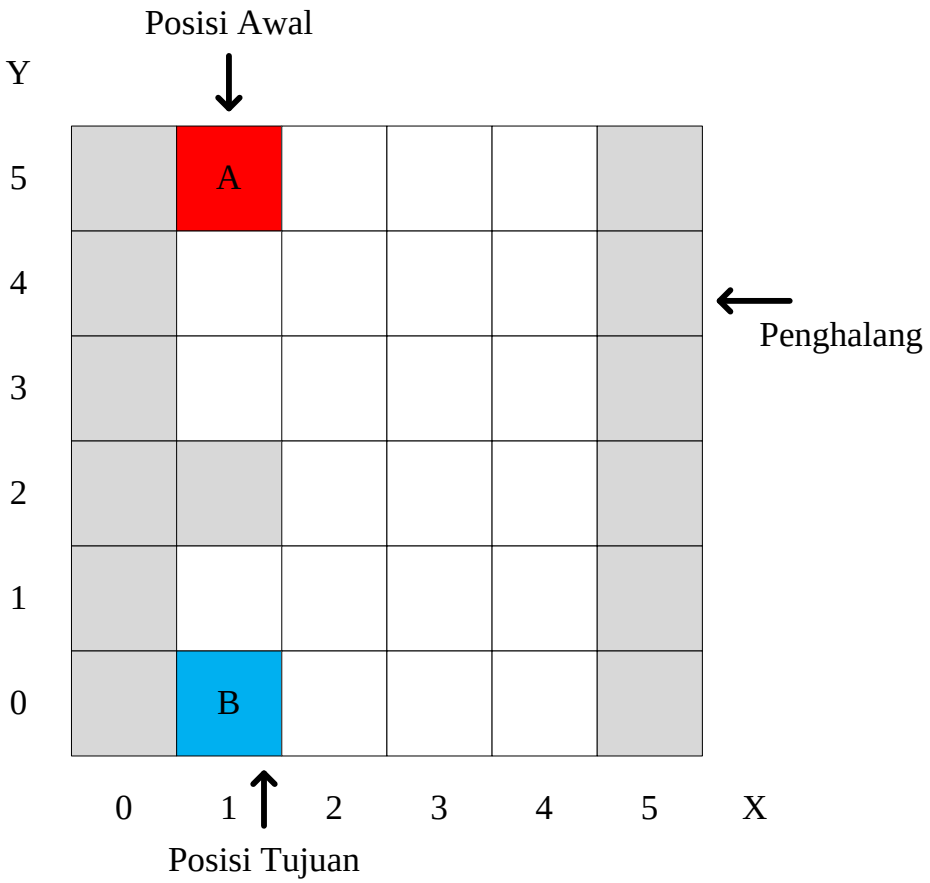
3.1.2.4 Analisis *Scoring*

Pada *game* yang akan dibangun terdapat poin yang diberikan setiap menyelesaikan permainan. Poin akan hilang atau kembali ke keadaan awal jika pemain tidak menyelesaikan misi dalam setiap levelnya. Jumlah poin yang didapat dari setiap misi yang dilakukannya sebagai berikut:

a. Mengklik sampah sesuai jenisnya = +15
b. Mengklik sampah tidak sesuai jenisnya = -10
c. Tas = +100
d. Pupuk = +80
e. Lampu Meja = +90

3.1.3 Analisis Algoritma A* (A *Star*)

Algoritma A* akan diterapkan pada NPC mobil dalam melakukan penentuan jalur terpendek atau *pathfinding* menuju target. Cara kerja algoritma A* dalam *game* yang dibangun akan dijelaskan dengan ordo 6 x 6. Berikut adalah ilustrasi A* yang dapat dilihat pada Gambar 3.2.



Gambar 3.2 Cara Kerja Algoritma A*

Pada Gambar 3.2 terdapat beberapa kotak yang memiliki koordinat (1,5) sebagai posisi awal dan (1,0) sebagai posisi tujuan. Kotak atau *node* yang masing-masing telah memiliki koordinat merupakan *node* yang digunakan untuk melakukan pencarian jalur terpendek. Terdapat beberapa *node* yang memiliki warna berbeda, berikut penjelasan dari masing-masing *node* tersebut:

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

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

1. Nilai F, nilai yang diperoleh dari penjumlahan nilai G dan H, jumlah nilai tiap simpul dalam jalur terpendek dari *starting point* ke *current node,* dan

H, jumlah nilai perkiraan dari sebuah simpul ke *target point.*

2. Nilai G, nilai atau biaya yang dikeluarkan dari *node* awal atau *node* A ke *node* yang ada disekitarnya.
3. Nilai H, nilai *heuristic* yang digunakan untuk menghitung perkiraan biaya yang dikeluarkan dari *node* awal atau *node* A menuju *node* tujuan atau *node* B.

Perhitungan nilai F didapat dari persamaan F = G + H. Nilai G adalah Nilai dari *node* ke *node* yang akan dihitung secara *vertical* atau *horizontal*, nilai G akan digunakan dengan nilai 10. Sedangkan untuk nilai dari *node* ke *node* yang akan dihitung secara *diagonal* mengunakan metode *Euclidean* dengan perhitungan nilai *node* ke *node* yang akan dihitung dikali √2 yang berarti 10x√2 = 14.14 Untuk mempermudah perhitungan 14.14 dibulatkan menjadi 14. Dan untuk pehitungan nilai H digunakan fungsi *heuristic,* metode yang digunakan adalah metode *Manhattan* dimana perhitungan jumlah *node* hanya yang bergerak secara *vertical* dan *horizontal* menuju tujuan serta mengabaikan penghalang, yang kemudian nilainya dikalikan nilai node atau dirumuskan dengan:

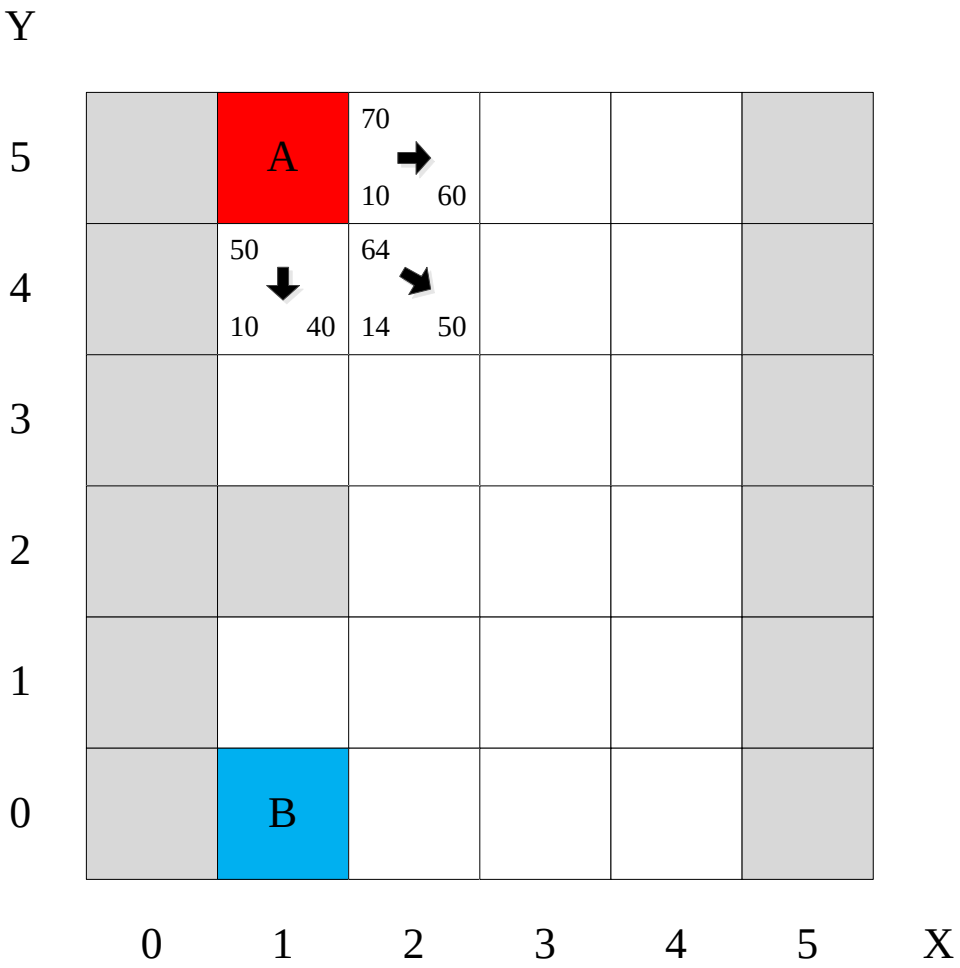
$$H = 10*(abs(currentX-targetX) + abs(currentY-targetY)) \quad (3.1)$$

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

1. Langkah pertama adalah memasukkan *node* A kedalam *open list*.
2. Ambil *node* A sebagai *current node.*
3. Periksa apakah *current node* sama dengan *node* akhir atau *current node* yang sudah didapatkan merupakan *node* akhir. Jika ya maka jalan sudah ditemukan, jika tidak maka hapus *node* awal dari *open list* dan tambahkan ke dalam *closed list*.
4. Periksa semua *neighbor node* atau *node* yang berdekatan dengan *node* awal apakah *walkable* atau tidak, abaikan *node* penghalang lalu tambahkan

semua *node* tersebut ke dalam *open list*. Dan untuk setiap *node* yang baru ditambahkan, simpan *node* awal sebagai *parent* yang nantinya akan digunakan untuk menelusuri jalan .

Berikut ini perhitungan pada langkah pertama dengan *node* yang berada pada koordinat (1,5).



Gambar 3.3 Ilustrasi Langkah Pertama

Pada langkah pertama, *node* (1,5) dimasukkan ke dalam *open list* dan selanjutnya memeriksa *neighbor node* atau *node* yang berdekatan dengan *node* (1,5) yaitu *node* (2,5), (2,4), (1,4). Setelah diperiksa *node* tersebut dimasukkan ke dalam *open list*. Hapus *node* (1,5) dari *open list* dan masukkan *node* (1,5) ke dalam *closed list*. Lalu dari *node-node* yang berada di dalam *open list* pilih *node*

dengan nilai F yang paling rendah, yaitu *node* (1,4) dengan nilai F = 50. Untuk memperjelas perhitungan maka akan dijelaskan langkah – langkah sebagai berikut.

1. Langkah pertama dengan menghitung *node* (2,5)

   Nilai n.x = 2

   Nilai n.y = 5

   Node tujuan (1,0) sehingga targetX = 1 dan targetY = 0

   G(2,5) = 10 (nilai node ke node yang akan dihitung)

   H(2,5) = 10*(abs(currentX - targetX) + abs(currentY - targetY))

   = 10*(abs(2 - 1) + abs(5 - 0))

   = 10*(abs(1) + abs(5))

   = 10*(1 + 5)

   = 10*6

   = 60

   F(2,5) = 10 + 60

   = 70

2. Langkah kedua dengan menghitung *node* (2,4)

   Nilai n.x = 2

   Nilai n.y = 4

   Node tujuan (1,0) sehingga targetX = 1 dan targetY = 0

   G(2,4) = 10*√2 = 14.14

   = 14 (dibulatkan)

   H(2,4) = 10*(abs(currentX - targetX) + abs(currentY - targetY))

   = 10*(abs(2 - 1) + abs(4 - 0))

   = 10*(abs(1) + abs(4))

   = 10*(1 + 4)

   = 10*5

   = 50

   F(2,4) = 14 + 50

   = 64

3. Langkah ketiga dengan menghitung *node* (1,4)

   Nilai n.x = 1

   Nilai n.y = 4

   Node tujuan (1,0) sehingga targetX = 1 dan targetY = 0

   G(1,4) = 10

   H(1,4) = 10*(abs(currentX - targetX) + abs(currentY - targetY))

   = 10*(abs(1 - 1) + abs(4 - 0))

   = 10*(abs(0) + abs(4))

   = 10*(0 + 4)

   = 10*4

   = 40

   F(1,4) = 10 + 40

   = 50

Langkah pertama menghasilkan *node-node* di dalam *open list* dan *closed list* dijelaskan pada tabel berikut.

Tabel 3.2 Perhitungan Langkah Pertama

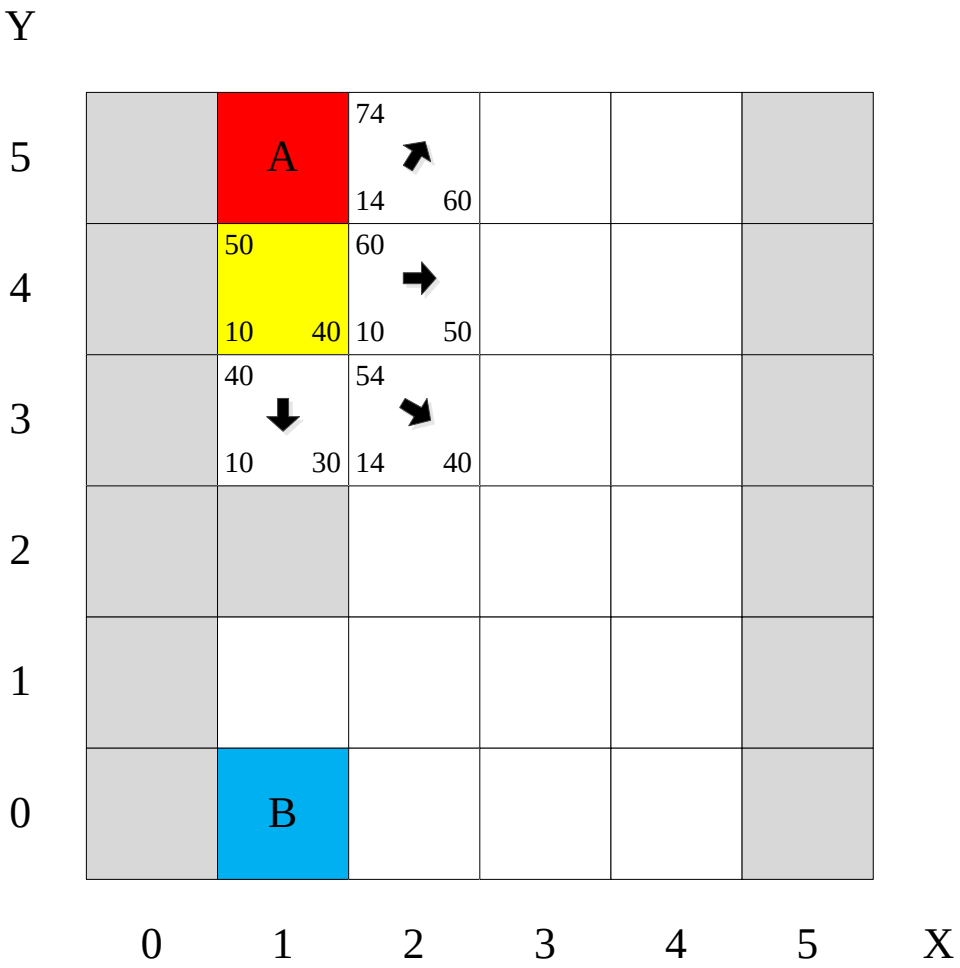
Open List	(2,5), (2,4), (1,4)
Closed list	(1,5)
Parent	(1,5)
Current node	(1,4)

Proses ini dilakukan berulang kali, dimulai dari *open list* dan menemukan *node* dengan nilai F paling rendah. Langkahnya adalah sebagai berikut:

1. Hapus *current node* tersebut dari *open list* dan masukkan ke dalam *closed list.*
2. Periksa semua *node* yang berdekatan dengan *current node*, abaikan *node* yang *unwalkable* atau *node* dengan penghalang. Jika tidak ada pada *open list*, tambahkan ke dalam *open list*. Dan tambahkan *parent* untuk *node* tersebut.
3. Jika *node* sudah ada pada *open list*, periksa apakah ini jalan terbaik dari *node* awal, dengan kata lain periksa apakah nilai G-nya lebih rendah

apabila menggunakan *node* ini, jika lebih rendah maka hitung ulang nilai F dan G serta merubah arah *pointer*, jika tidak maka jangan lakukan apa-apa.

Selanjutnya perhitungan pada langkah kedua dengan *node* berada pada koordinat (1,4).



Gambar 3.4 Ilustrasi Langkah Kedua

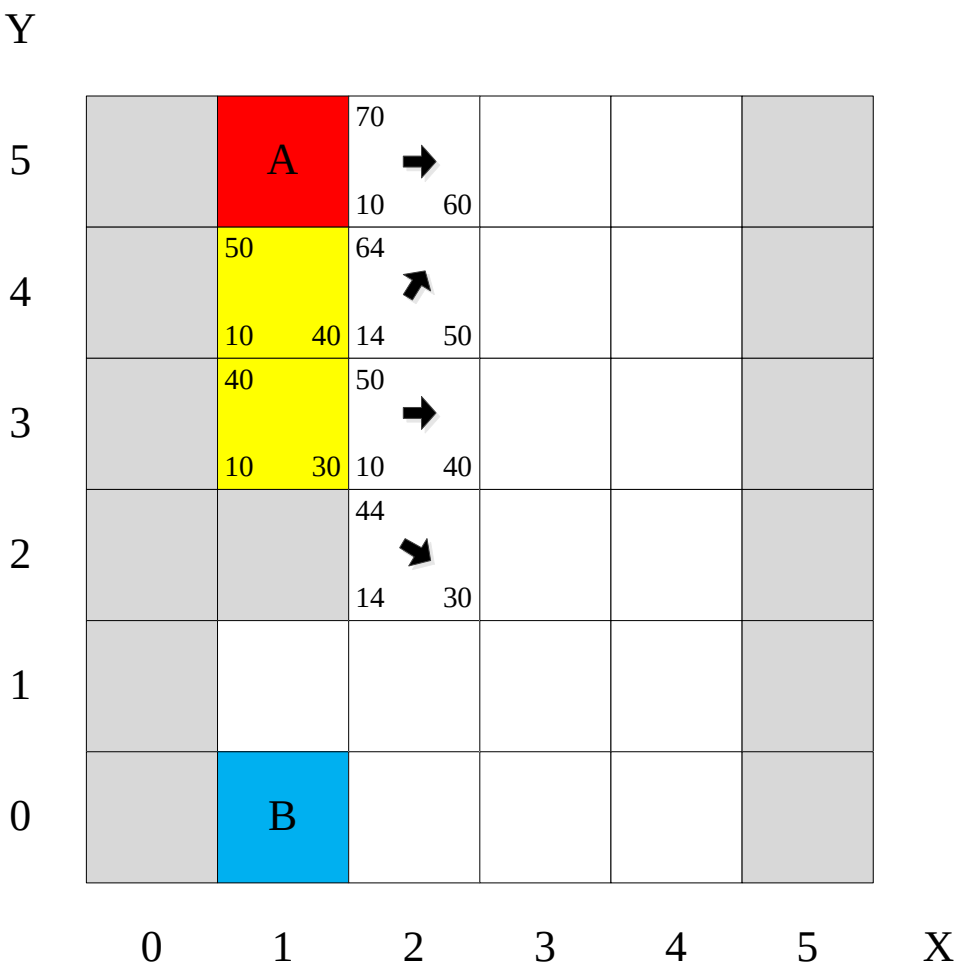
Pada langkah kedua terdapat 2 *node* yang masuk ke dalam *open list* yaitu *node* dengan koordinat (1,3), (2,3), (2,4) dan (2,5) sedangkan untuk *node* (1,5) dan (1,4) merupakan *node closed list*. Dari langlah kedua, *node* yang masuk ke dalam *open list* dengan nilai F paling rendah yaitu *node* (1,3) dengan nilai F = 40.

Langkah kedua menghasilkan *node-node* di dalam *open list* dan *closed list* dijelaskan pada tabel berikut.

Tabel 3.3 Perhitungan Langkah Kedua

Open List	(1,3), (2,3), (2,4), (2,5)
Closed list	(1,5), (1,4)
Parent	(1,4)
Current node	(1,3)

Selanjutnya perhitungan pada langkah ketiga dengan *node* berada pada koordinat (2,2).



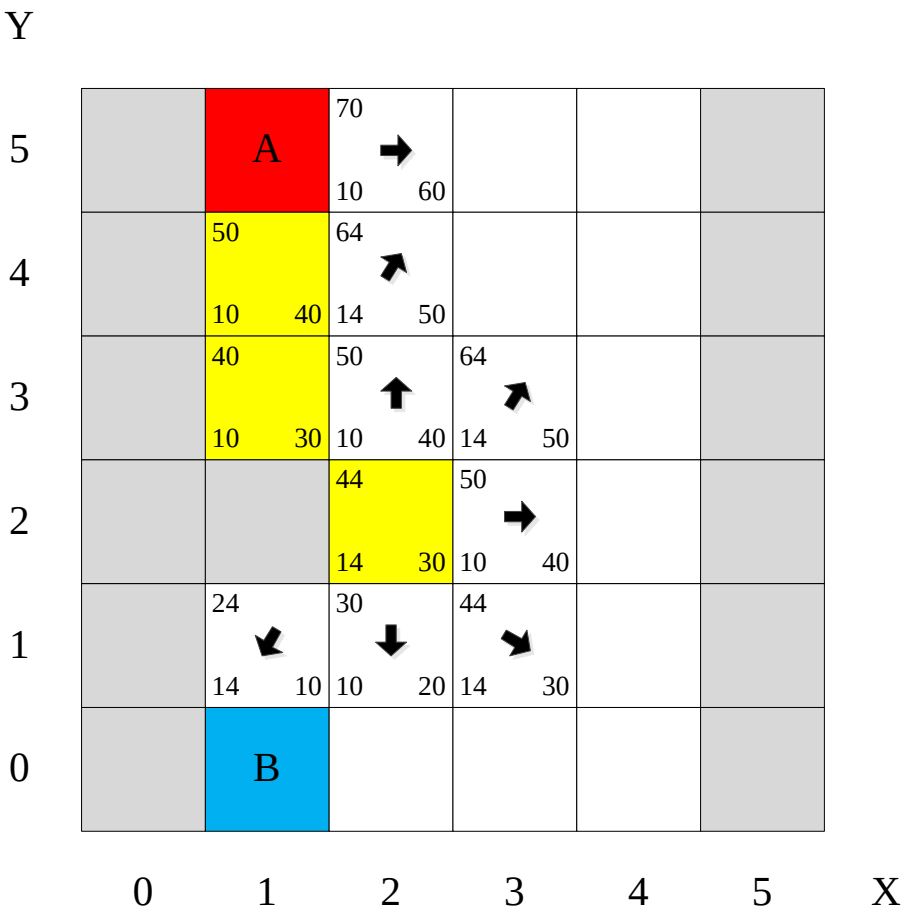
Gambar 3.5 Ilustrasi Langkah Ketiga

Langkah ketiga menghasilkan *node-node* di dalam *open list* dan *closed list* dijelaskan pada tabel berikut.

Tabel 3.4 Perhitungan Langkah Ketiga

Open List	(2,4), (2,3), (2,2)
Closed list	(1,5), (1,4), (1,3)
Parent	(1,3)
Current node	(2,2)

Selanjutnya perhitungan pada langkah keempat dengan *current node* berada pada *node* (2,1).



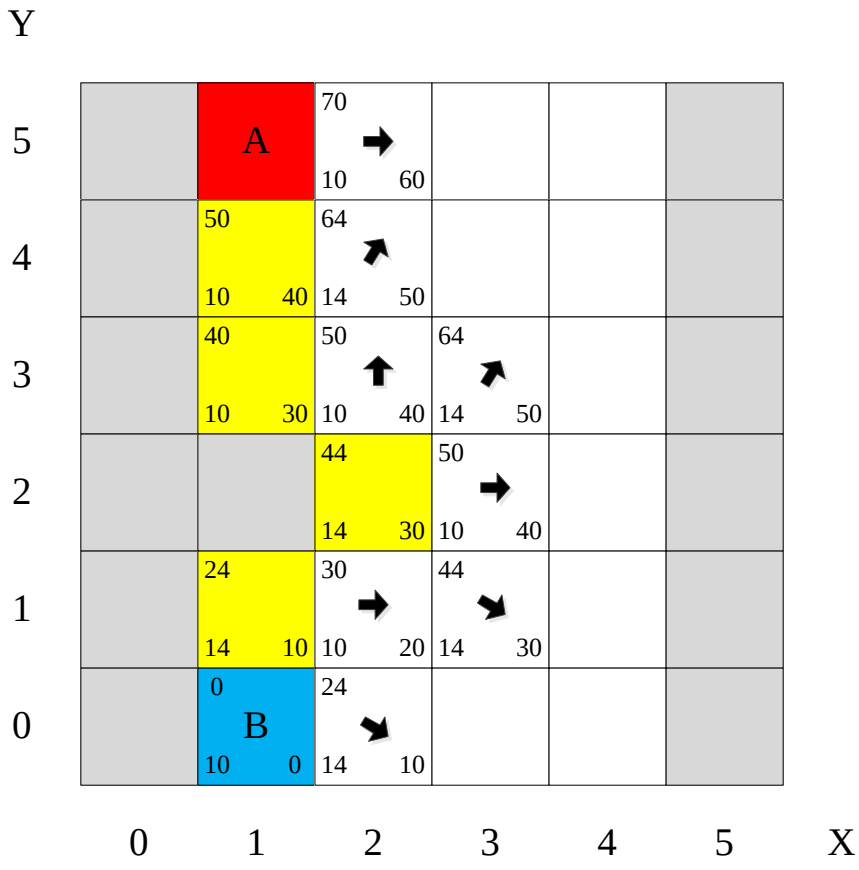
Gambar 3.6 Ilustrasi Langkah Keempat

Langkah keempat menghasilkan *node-node* di dalam *open list* dan *closed list* dijelaskan pada berikut.

Tabel 3.5 Perhitungan Langkah Keempat

Open List	(2,3), (3,3), (3,2), (3,1), (2,1), (1,1)
Closed list	(1,5), (1,4), (1,3), (2,2)
Parent	(2,2)
Current node	(1,1)

Penentuan jalan terbaik dengan melakukan pemeriksaan dimulai dari *node* awal atau *node* A (1,5) yang ditandai dengan warna merah sampai *target point* atau *node* B (1,0). Dengan mengikuti arah *pointer* yang berarti akan membawa kembali ke *node* awal, maka akan didapatkan jalan terbaiknya, untuk lebih jelasnya dapat dilihat pada gambar berikut.



Gambar 3.7 Ilustrasi Langkah Terakhir

Langkah terakhir menghasilkan *node-node* di dalam *open list* dan *closed list* dijelaskan pada tabel berikut.

Tabel 3.6 Perhitungan Langkah Terakhir

Open List	(2,1), (2,0), (1,0)
Closed list	(1,5), (1,4), (1,3), (2,2), (1,1)
Parent	(1,1)
Current node	(1,0)

3.1.3.1 *Pseudocode*

Berikut adalah *pseudocode* algoritma A*

```
Algoritma A*
{Algoritma untuk mencari jalur terpendek}

Deklarasi
openList          : startNode
closedList        : array kosong
goalNode, startNode   : real
f, g, h   : real

Algoritma:
Masukkan startNode ke dalam openList
while openList ≠ kosong do
    bestNode ← f terkecil dalam openList
    keluarkan bestNode dari openList
    for each bestNode
        if bestNode = goalNode then
            selesai
        endif
        g ← jarak antara startNode ke node sekitarnya
        h ← jarak dari bestNode ke goalNode
        f ← g + h
        if node baru di openList memiliki nilai f lebih kecil dari bestNode
        then
```

simpan node baru sebagai bestNode baru

endif

endfor

keluarkan bestNode lama dari openList dan masukkan ke closedList

endwhile

3.1.4 Analisis Kebutuhan Non-Fungsional

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

3.1.4.1 Analisis Kebutuhan Perangkat Lunak

Perangkat lunak yang digunakan dalam sebuah sistem merupakan perintah-perintah yang diberikan kepada perangkat keras agar bisa saling berinteraksi diantara keduanya.

1. **Pengembang**

Spesifikasi perangkat lunak yang digunakan pengembang untuk membangun *game* ini dapat dilihat pada Tabel 3.7.

Tabel 3.7 Spesifikasi Perangkat Lunak Pengembang

Nama Perangkat Lunak	Spesifikasi
Sistem Operasi	Microsoft Windows 7 SP1
Tools	- Unity 3D 4.3 - Adobe Photoshop CS6 - DirectX 11

2. Pengguna

Spesifikasi minimum perangkat lunak yang dibutuhkan pengguna untuk memainkan *game* ini dapat dilihat pada Tabel 3.8.

Tabel 3.8 Spesifikasi Minimum Perangkat Lunak Pengguna

Nama Perangkat Lunak	Spesifikasi
Sistem Operasi	Microsoft Windows XP SP2
Tools	- Unity Web Player - DirectX 9

3.1.4.2 Analisis Kebutuhan Perangkat Keras

Komputer terdiri dari perangkat keras dan perangkat lunak yang saling berinteraksi. Perangkat lunak memberikan instruksi-instruksi kepada perangkat keras untuk melakukan suatu tugas tertentu, sehingga dapat menjalankan suatu sistem di dalamnya.

1. Pengembang

Spesifikasi perangkat keras yang digunakan pengembang untuk membangun *game* ini dapat dilihat pada Tabel 3.9.

Tabel 3.9 Spesifikasi Perangkat Keras Pengembang

Nama Perangkat	Spesifikasi
Processor	Intel Core 2 Duo 2.20 GHz
RAM	4 GB
VGA Card	250 MB
Harddisk	500 GB
Monitor	Resolusi 1366 x 768
Mouse dan *keyboard*	Standar

2. Pengguna

Berdasarkan *system requirement* yang ada pada website Unity3D spesifikasi minimum perangkat keras bagi pengguna untuk dapat memainkan *game* ini dengan baik dapat dilihat pada Tabel 3.10.

Tabel 3.10 Spesifikasi Minimum Perangkat Keras Pengguna

Nama Perangkat	Spesifikasi
Processor	1.80 GHz
RAM	2 GB
VGA Card	250 MB
Harddisk	Tersedia 100 MB
Monitor	Resolusi 800 x 600
Mouse dan *keyboard*	Standar

Dengan spesifikasi yang lebih rendah maka pengguna tidak akan memainkan aplikasi *game* dengan maksimal seperti mengalami ketersendatan atau gambar yang kabur.

3.1.4.3 Analisis Pengguna

Pada analisis pengguna ini akan mencakup analisis beberapa parameter terhadap calon pengguna dari aplikasi.

1. ***User Knowledge and Experience* dari target pengguna yang akan menggunakan *game***

Game yang akan dibangun ini bisa digunakan oleh kalangan apapun, tetapi pengetahuan dan pengalaman akan memudahkan pengguna dalam penggunaannya. Berikut ini klasifikasi *knowledge and experience* dari aplikasi:

Tabel 3.11 Analisis klasifikasi *knowledge and experience*

Educational Level	*Game* ini bisa digunakan oleh berbagai kalangan seperti pelajar, mahasiswa hingga masyarakat awam
Reading Level	*Game* ini bisa digunakan oleh berbagai level pendidikan dengan *reading* level rendah
Typing Skills	*Game* ini tidak memerlukan *typing skills* tinggi
Computer Literacy	*Game* ini bisa digunakan oleh pengguna yang memiliki kemampuan komputer yang rendah
Task Experience	*Game* ini bisa digunakan oleh dengan pengalaman *game* yang rendah
System Experience	*Game* ini bisa digunakan oleh dengan pengalaman komputer dan *game* yang rendah
Application Experience	*Game* ini bisa digunakan dalam semua sistem operasi Windows minimal XP

Native Language	*Game* ini menggunakan bahasa Indonesia
Use of Other System	*Game* ini bisa dijalankan tanpa perlu install aplikasi lain

2. *User Physical Characteristic*

Keadaan fisik seseorang mungkin akan berpengaruh pada penggunaan aplikasi *game* ini. Ada hal-hal yang harus diperhatikan juga terhadap pengguna dari karakteristik fisiknya untuk dapat menggunakan aplikasi ini yaitu, *color blind, handedness,* dan *gender*.

Tabel 3.12 Analisis *Users Physical Characteristic*

Age	9 tahun ke atas
Gender	Pria dan wanita
Handedness	Kanan dan kiri
Disabilities	Pengguna yang buta tidak akan bisa menggunakan *game* ini, untuk yang penglihatannya kurang dan tuli masih dapat memainkan tetapi tidak akan optimal.

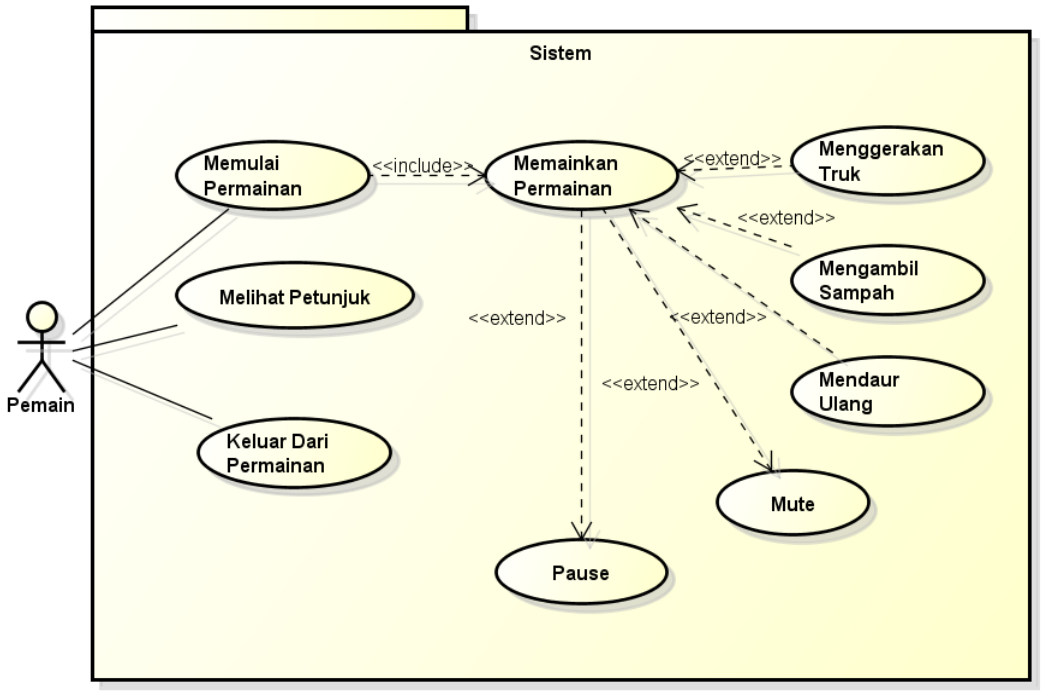
Target pengguna yaitu usia 9 tahun ke atas karena pada usia tersebut lebih banyak terlibat dalam *games with rules* dimana kegiatan anak dikendalikan oleh peraturan permainan.

3.1.5 Analisis Kebutuhan Fungsional

Analisis kebutuhan fungsional menggambarkan proses kegiatan yang akan diterapkan dalam sebuah sistem dan menjelaskan kebutuhan yang diperlukan sistem agar sistem dapat berjalan dengan baik sesuai kebutuhan. Analisis kebutuhan fungsional dimodelkan dengan menggunakan UML *(Unified Modeling Language).*

3.1.5.1 *Use Case Diagram*

Diagram *use case* menggambarkan hubungan antara aktor dan sistem yang dibangun. Perancangan proses-proses yang terdapat pada *game* yang akan dibangun akan digambarkan dengan diagram *use case* pada Gambar 3.8.



Gambar 3.8 *Usecase Diagram*

1. Definisi Aktor

Definisi aktor berfungsi untuk menjelaskan aktor yang terdapat pada diagram *use case*. Definisi aktor diterangkan pada Tabel 3.13.

Tabel 3.13 Definisi Aktor

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

2. Definisi *Use Case*

Definisi *use case* berfungsi untuk menjelaskan fungsi *use case* yang terdapat pada diagram *use case*. Definisi *use case* diterangkan pada Tabel 3.14.

Tabel 3.14 Definisi *Use Case*

No.	*Use Case*	Deskripsi
1.	Memulai permainan	Proses untuk memulai permainan atau *game*.
2.	Melihat petunjuk	Proses untuk menampilkan petunjuk atau panduan cara bermain.
3.	Keluar dari permainan	Proses keluar dari aplikasi permainan.
4.	Memainkan permainan	Proses memainkan permainan
5.	Menggerakkan truk	Proses menggerakkan truk dengan menggunakan

		tombol *keyboard*.
6.	Mengambil sampah	Proses mengambil sampah.
7.	Mendaur ulang	Proses mendaur ulang sampah menjadi produk.
8.	*Pause*	Proses untuk berhenti sementara dari permainan dan melanjutkan kembali permainan
9.	*Mute*	Proses untuk melakukan *mute* atau *unmute* suara pada permainan.

3. Skenario *Use Case*

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

Nama *use case*: Memulai permainan

Skenario:

Tabel 3.15 Skenario *Use Case* Memulai Permainan

Aksi Aktor	Reaksi Sistem
Skenario Normal	
1. Memilih menu mulai	
	2. Menampilkan misi permainan
2. Memilih menu main	
	3. Menampilkan layar permainan

Nama *use case*: Melihat petunjuk

Skenario:

Tabel 3.16 Skenario *Use Case* Melihat Petunjuk

Aksi Aktor	Reaksi Sistem
Skenario Normal	
1. Memilih menu petunjuk	
	2. Menampilkan petunjuk bermain
3. Menutup tampilan petunjuk	
	4. Menampilkan menu utama

Nama *use case*: Keluar dari permainan

Skenario:

Tabel 3.17 Skenario *Use Case* Keluar Dari Permainan

Aksi Aktor	Reaksi Sistem
Skenario Normal	
1. Memilih menu keluar	
	2. Menampilkan pesan keluar dari aplikasi *game*
3. Memilih menu "ya"	
	4. Menutup aplikasi *game*
Skenario Alternatif	
1. Memilih menu keluar	
	2. Menampilkan pesan keluar dari aplikasi *game*
3. Memilih menu "batal"	
	4. Menampilkan menu utama

Nama *use case*: Memainkan permainan

Skenario:

Tabel 3.18 Skenario *Use Case* Memainkan Permainan

Aksi Aktor	Reaksi Sistem
Skenario Normal	
	1. Menampilkan misi permainan
2. Memilih menu main	
	3. Menampilkan layar permainan
4. Memainkan permainan	
	5. Mengatur waktu
	6. Cek misi
	7. Menampilkan pesan misi berhasil jika misi terpenuhi
8. Pilih menu lanjutkan	
	9. Masuk ke level selanjutnya
	10. Menampilkan pesan misi gagal jika misi tidak terpenuhi
11. Mengulang permainan	

Nama *use case*: Menggerakan truk

Skenario:

Tabel 3.19 Skenario *Use Case* Menggerakan Truk

Aksi Aktor	Reaksi Sistem
Skenario Normal	
1. Menekan tombol *keyboard* A atau ←	
	2. Truk belok ke kiri
3. Menekan tombol *keyboard* D atau →	
	4. Truk belok ke kanan
5. Menekan tombol *keyboard* W atau ↑	
	6. Truk bergerak ke depan

Nama *use case*: Mengambil sampah

Skenario:

Tabel 3.20 Skenario *Use Case* Mengambil Sampah

Aksi Aktor	Reaksi Sistem
Skenario Normal	
1. Menggerakan truk ke arah sampah	
	2. Klik kiri atau kanan
	3. Poin bertambah jika mengklik dengan benar
	4. Poin berkurang jika salah mengklik
	5. Sampah diambil
Skenario Alternatif	
1. Menggerakan truk ke arah sampah	
	2. Tidak mendapatkan sampah
3. Menggerakan kembali truk ke arah sampah	

Nama *use case*: Mendaur ulang

Skenario:

Tabel 3.21 Skenario *Use Case* Mendaur Ulang

Aksi Aktor	Reaksi Sistem
Skenario Normal	
1. Memilih menu *recycle*	
	2. Menampilkan *pop-up* mendaur ulang
3. Memilih produk	
	4. Bahan yang dibutuhkan mencukupi
	5. Menampilkan produk yang dibuat
	6. Poin bertambah
	7. Jumlah sampah berkurang
Skenario Alternatif	
1. Memilih menu *recycle*	
	2. Menampilkan *pop-up* mendaur ulang
3. Memilih produk	
	4. Bahan yang dibutuhkan tidak mencukupi
	5. Tidak dapat membuat produk
6. Memilih produk lain	
7. Kembali ke permainan	

Nama use case: Pause

Skenario:

Tabel 3.22 Skenario *Use Case Pause*

Aksi Aktor	Reaksi Sistem
Skenario Normal	
	1. Menampilkan layar permainan
2. Memilih menu *pause*	
	3. Permainan berhenti
	4. Menampilkan menu *resume*
	5. Menampilkan menu keluar
6. Memilih menu *resume*	
	6. Melanjutkan permainan
7. Memilih menu keluar	
	8. Keluar dari permainan menuju menu utama

Nama *use case*: *Mute*

Skenario:

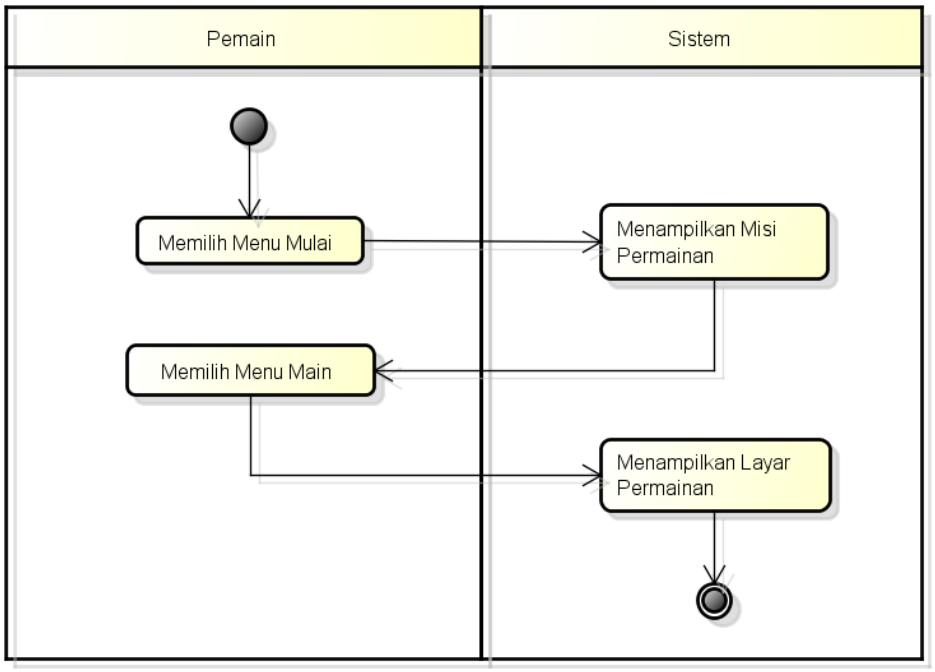
Tabel 3.23 Skenario *Use Case Mute*

Aksi Aktor	Reaksi Sistem
Skenario Normal	
	1. Menampilkan scene permainan
2. Memilih menu *mute*	
	3. Mematikan suara permainan
	4. Menampilkan menu *unmute*
5. Memilih menu *unmute*	
	6. Menghidupkan suara permainan

3.1.5.2 *Activity Diagram*

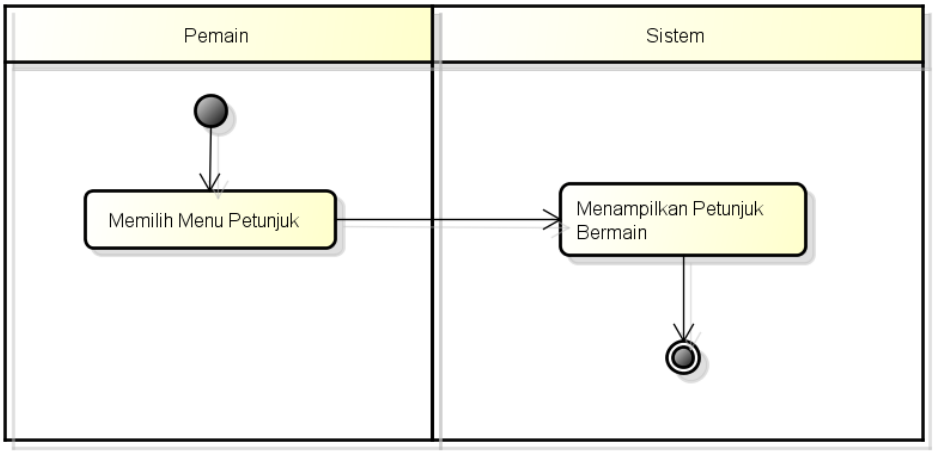
Diagram *activity* menggambarkan proses-proses yang terjadi mulai aktivitas dimulai sampai aktivitas berhenti pada sistem yang sedang dirancang. Bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir.

1. ***Activity Diagram* Memulai Permainan**



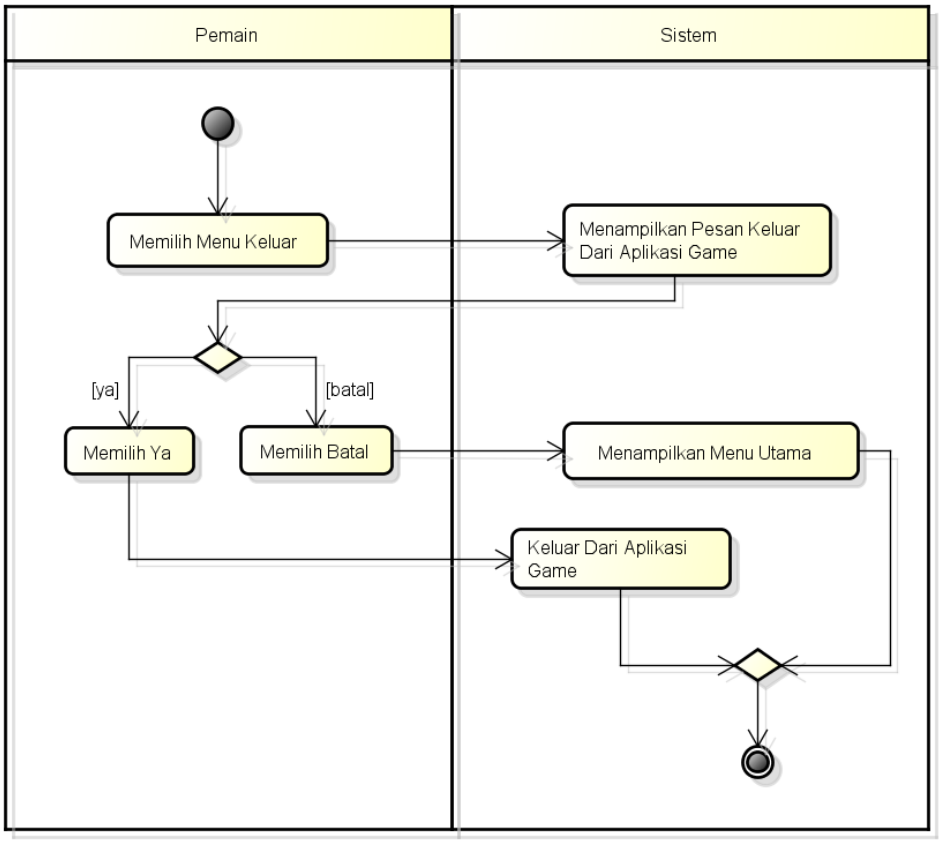
Gambar 3.9 *Activity Diagram* Memulai Permainan

2. *Activity Diagram* Melihat Petunjuk



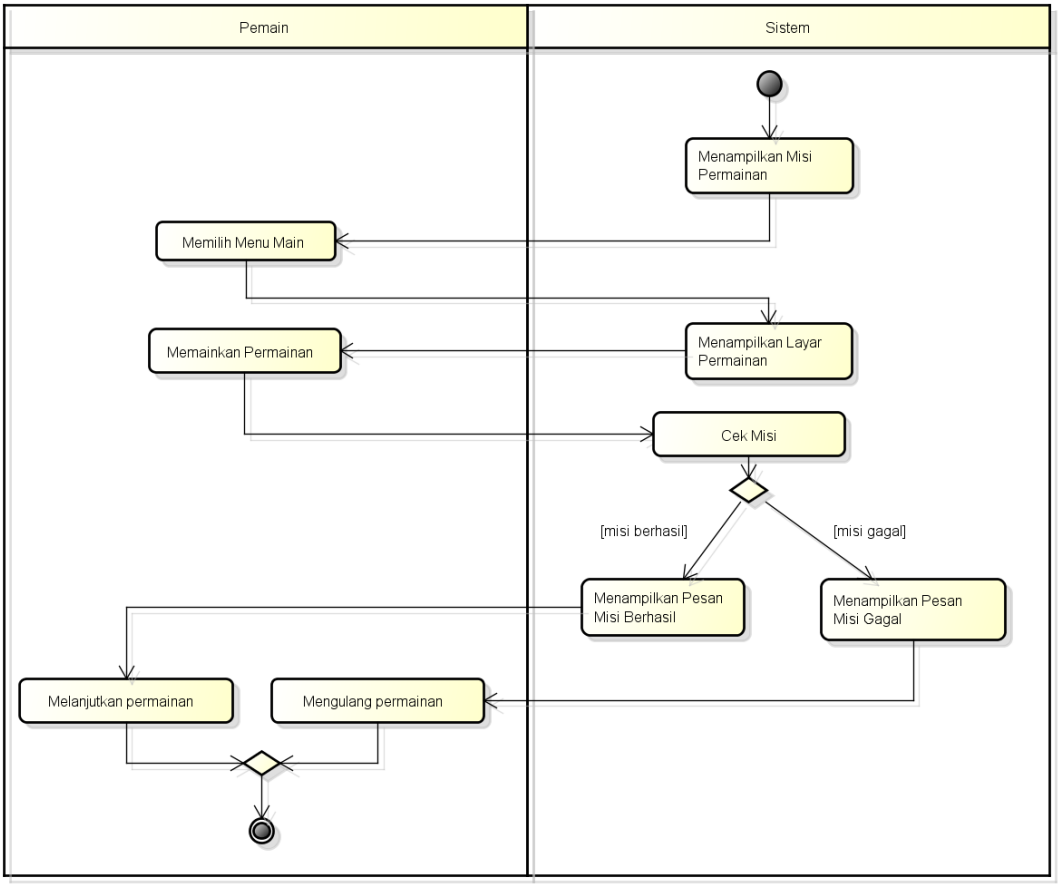
Gambar 3.10 *Activity Diagram* Melihat Petunjuk

3. *Activity Diagram* Keluar Dari Permainan



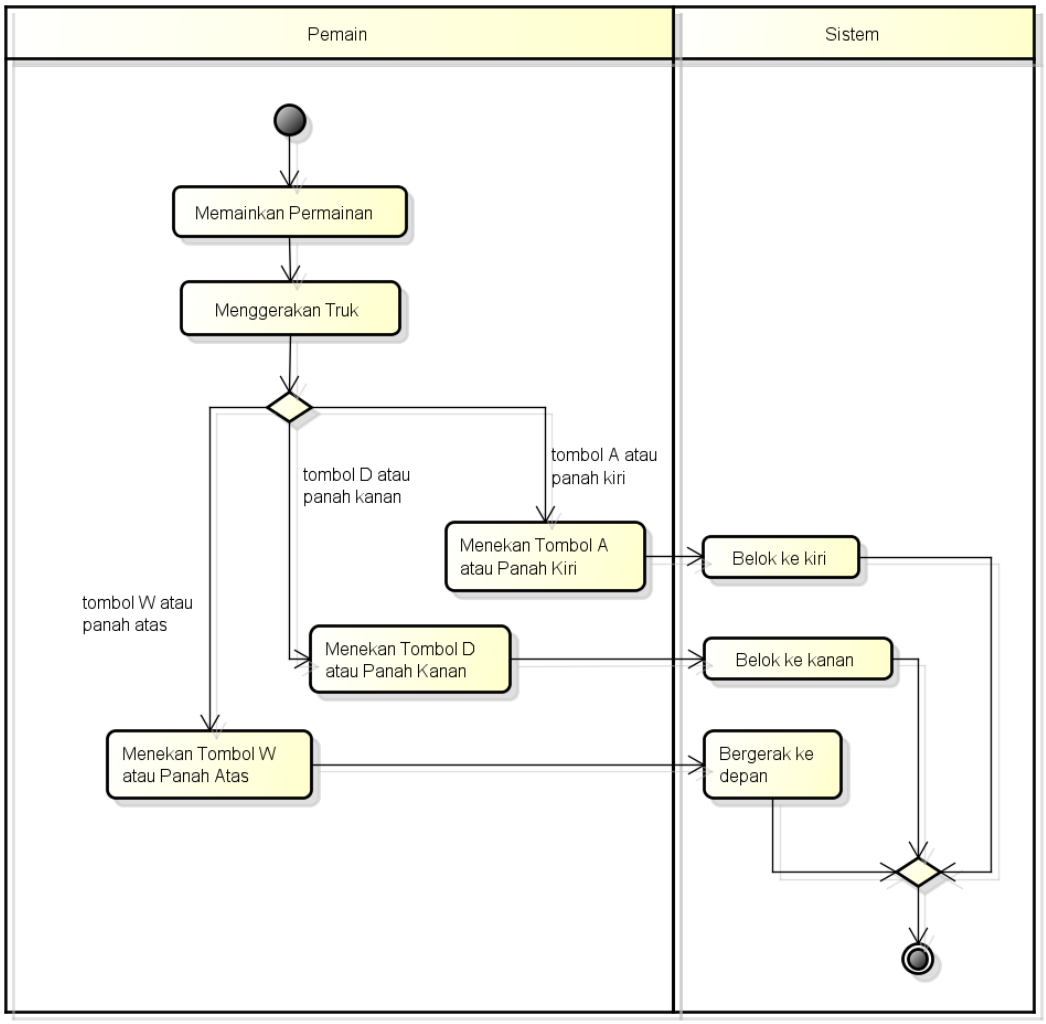
Gambar 3.11 *Activity Diagram* Keluar Dari Permainan

4. *Activity Diagram* Memainkan Permainan



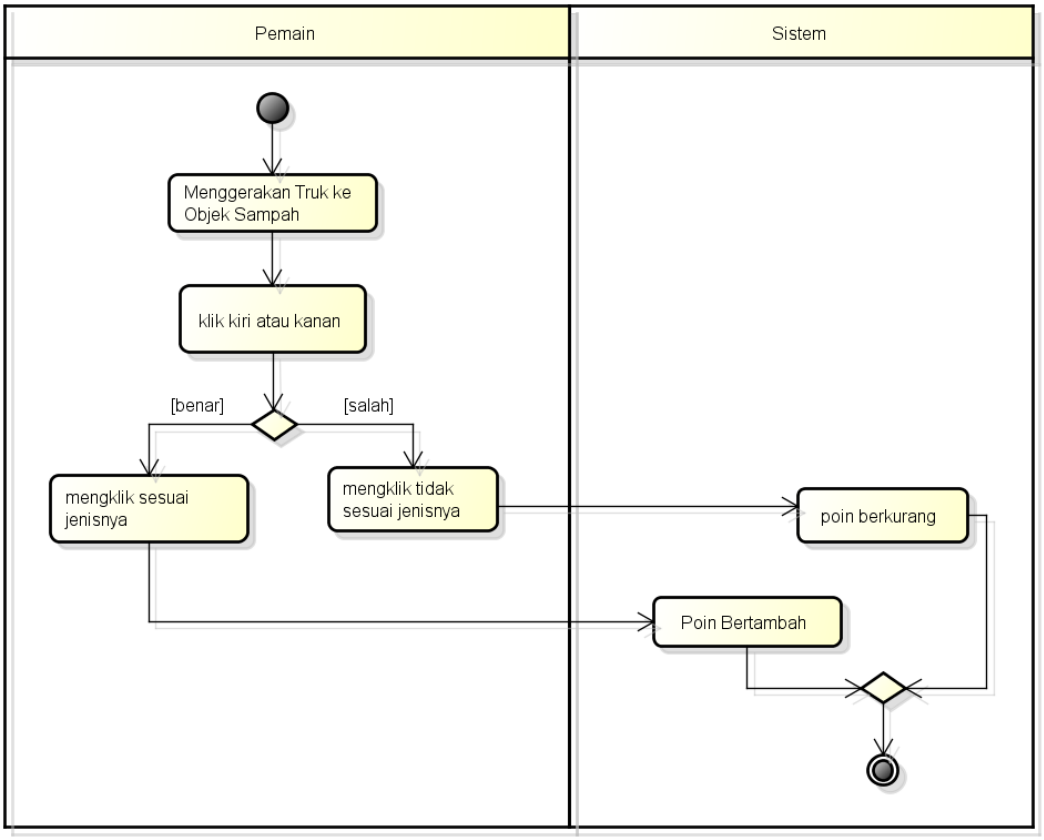
Gambar 3.12 *Activity Diagram* Memainkan Permainan

5. *Activity Diagram* Menggerakan Truk



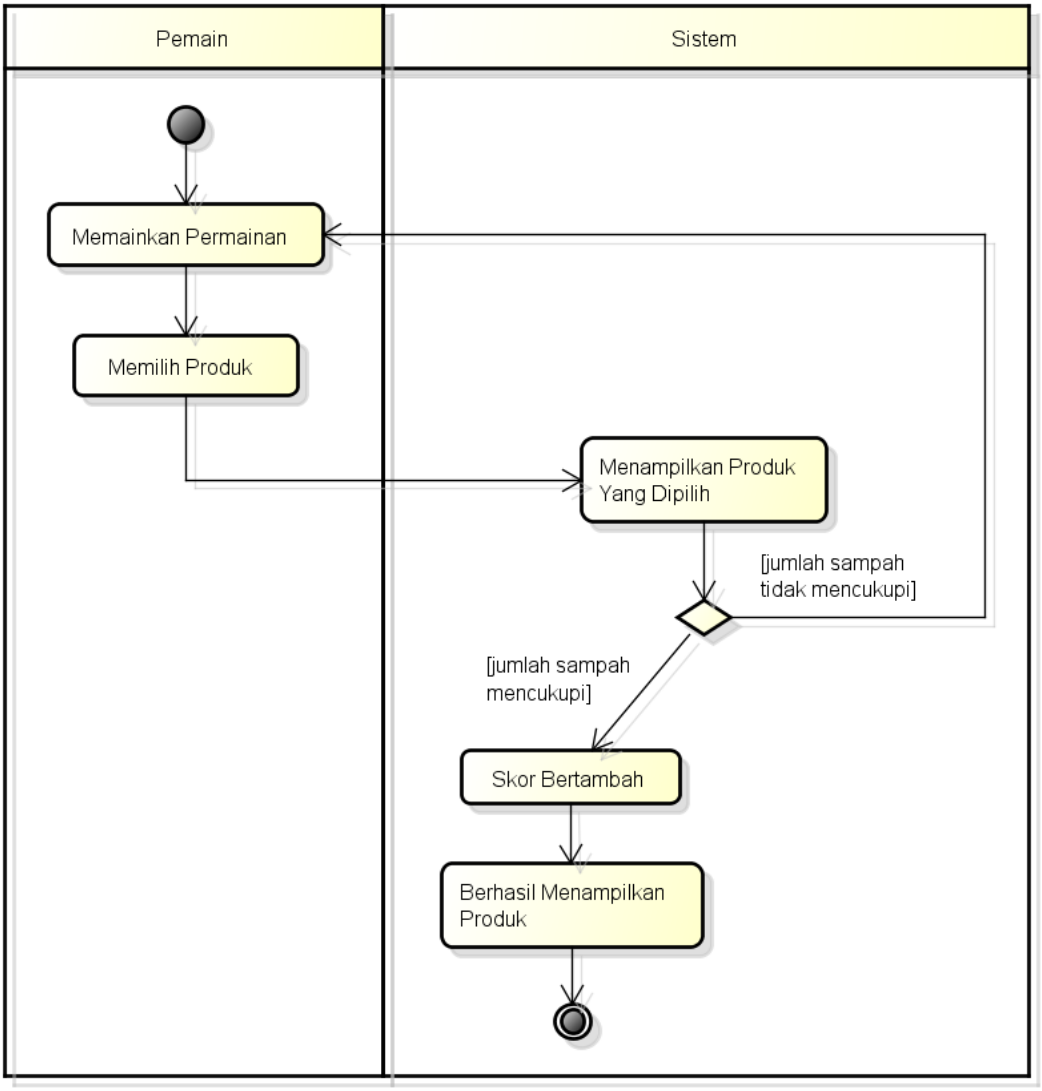
Gambar 3.13 *Activity Diagram* Menggerakan Truk

6. *Activity Diagram* Mengambil Sampah



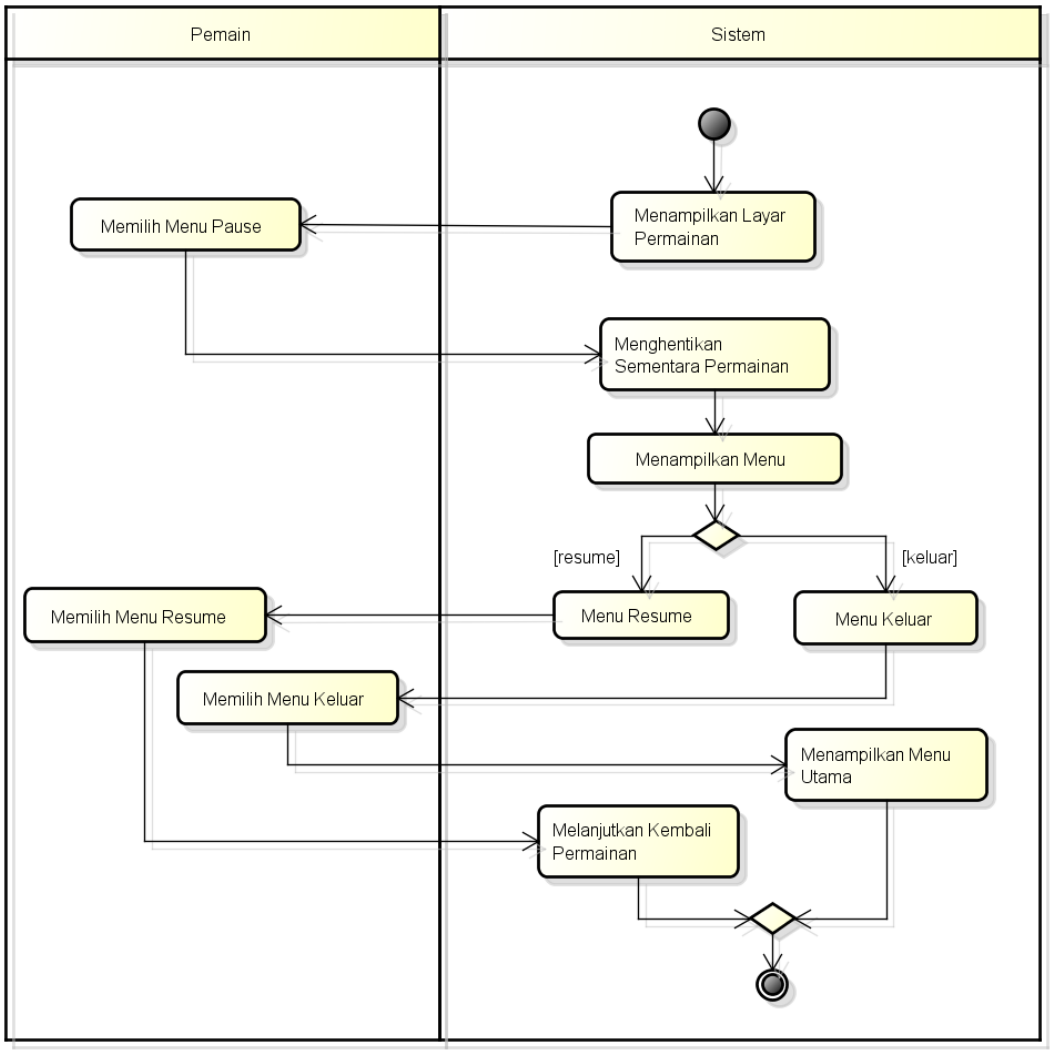
Gambar 3.14 *Activity Diagram* Mengambil Sampah

7. *Activity Diagram* Mendaur Ulang



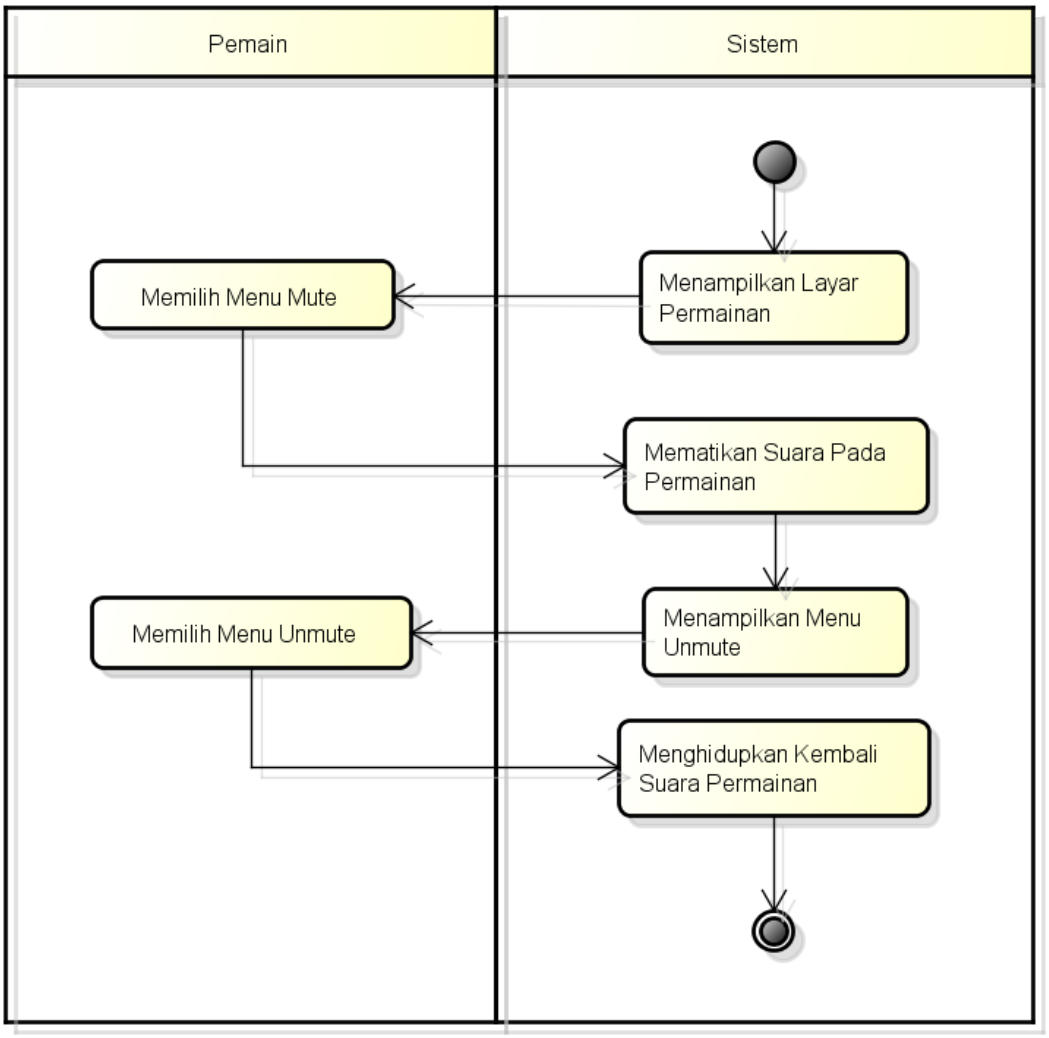
Gambar 3.15 *Activity Diagram* Mendaur Ulang

8. *Activity Diagram Pause*



Gambar 3.16 ***Activity Diagram Pause***

9. *Activity Diagram Mute*

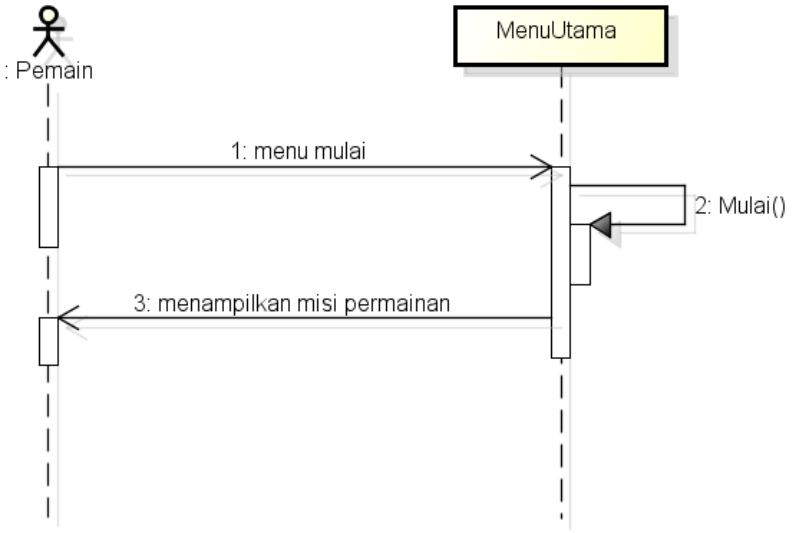


Gambar 3.17 *Activity Diagram Mute*

3.1.5.3 *Sequence Diagram*

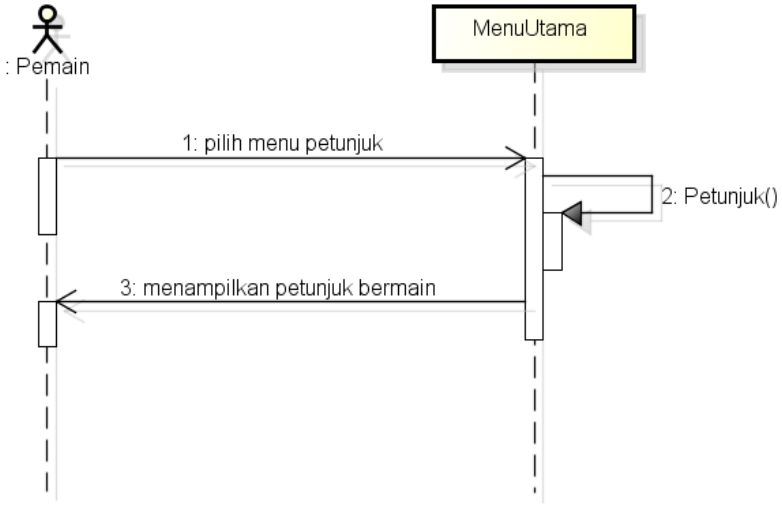
Sequence diagram merupakan penggambaran dari interaksi antar objek di dalam dan sekitar sistem (termasuk pengguna, *display*, dan sebagainya). *Sequence diagram* terdiri atas dimensi vertical (waktu) dan dimensi horizontal (objek-objek yang terkait).

1. ***Sequence Diagram*** **Memulai Permainan**



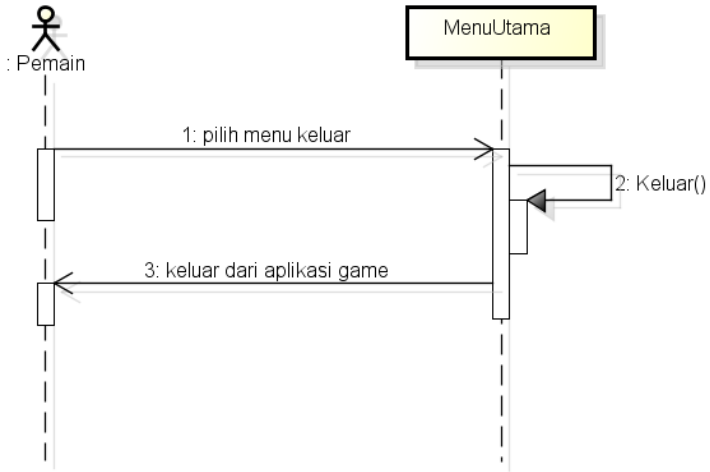
Gambar 3.18 ***Sequence Diagram*** **Memulai Permainan**

2. ***Sequence Diagram*** **Melihat Petunjuk**



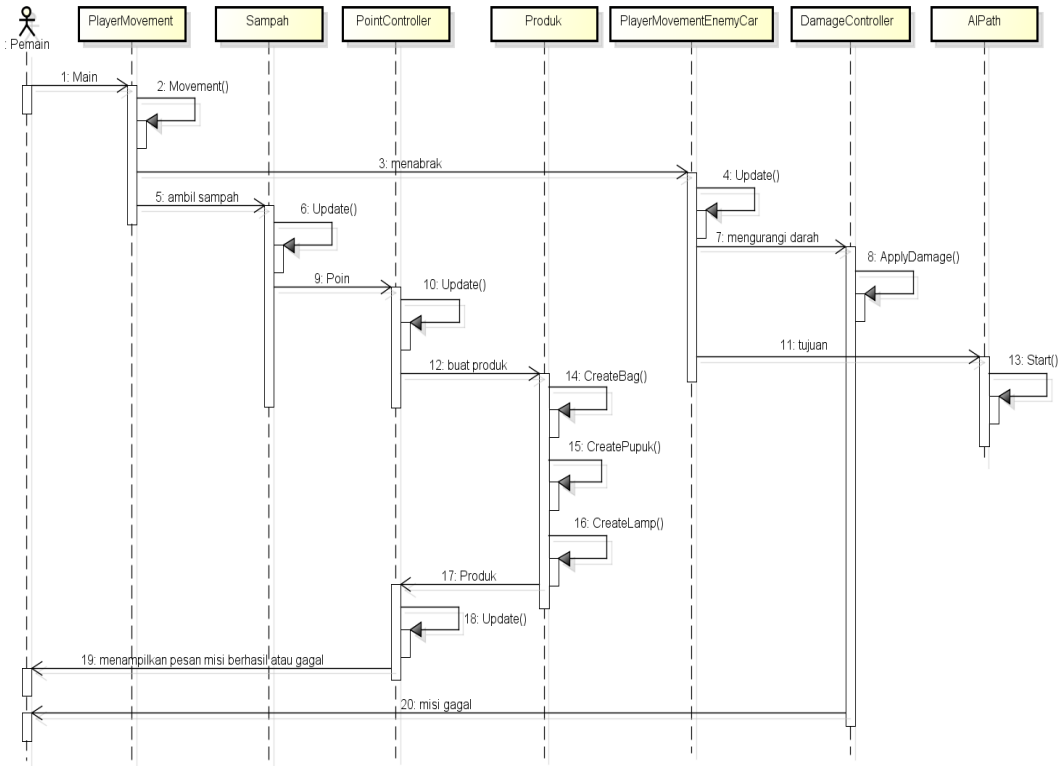
Gambar 3.19 ***Sequence Diagram*** **Melihat Petunjuk**

3. *Sequence Diagram* Keluar Dari Permainan



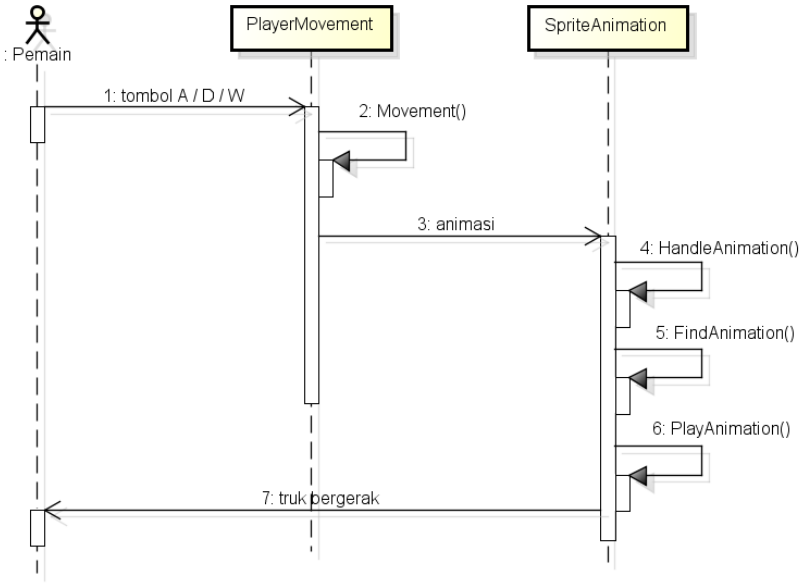
Gambar 3.20 *Sequence Diagram* Keluar Dari Permainan

4. *Sequence Diagram* Memainkan Permainan



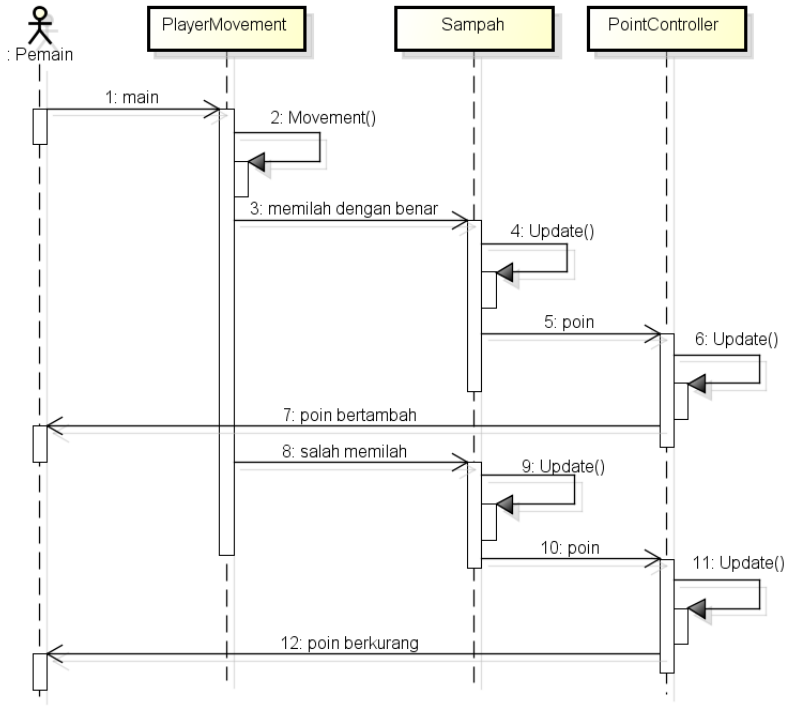
Gambar 3.21 *Sequence Diagram* Memainkan Permainan

5. *Sequence Diagram* Menggerakkan Truk



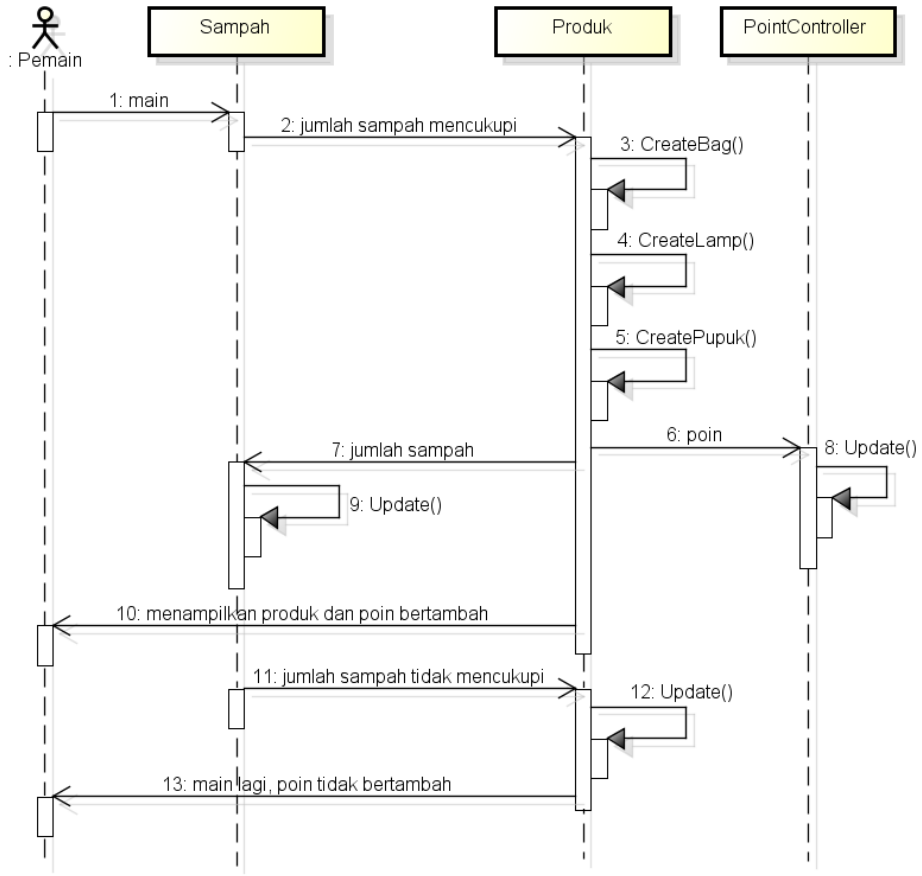
Gambar 3.22 *Sequence Diagram* Menggerakkan Truk

6. *Sequence Diagram* Mengambil Sampah



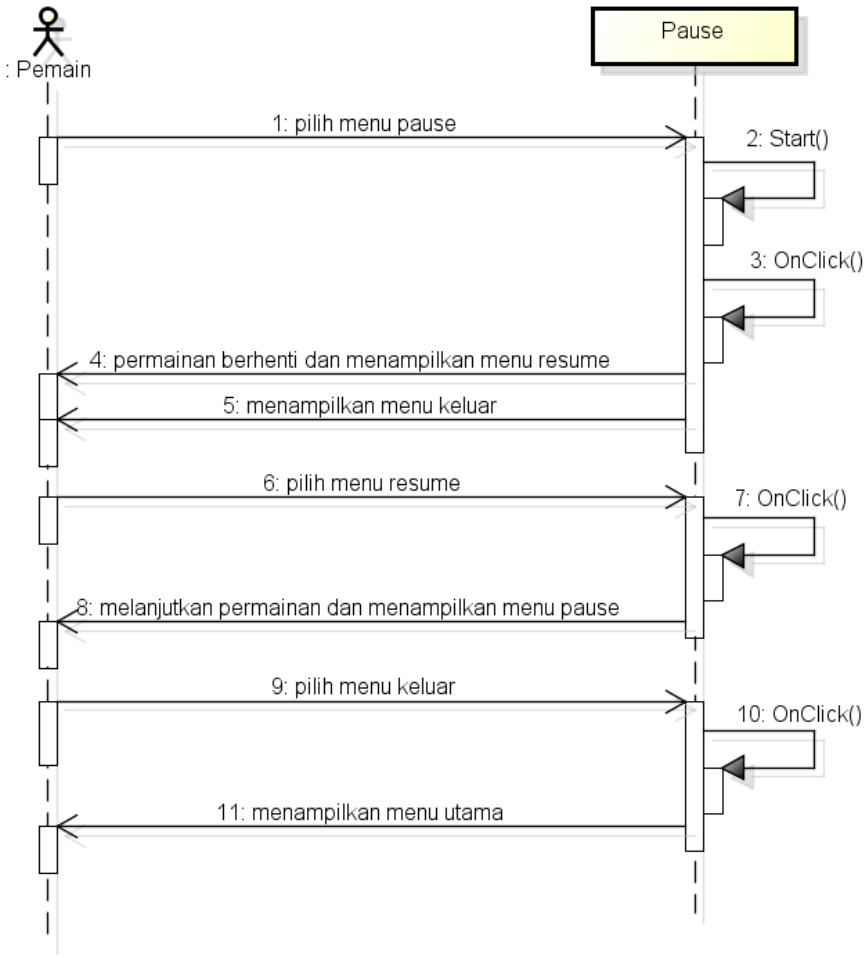
Gambar 3.23 *Sequence Diagram* Mengambil Sampah

7. *Sequence Diagram* Mendaur Ulang



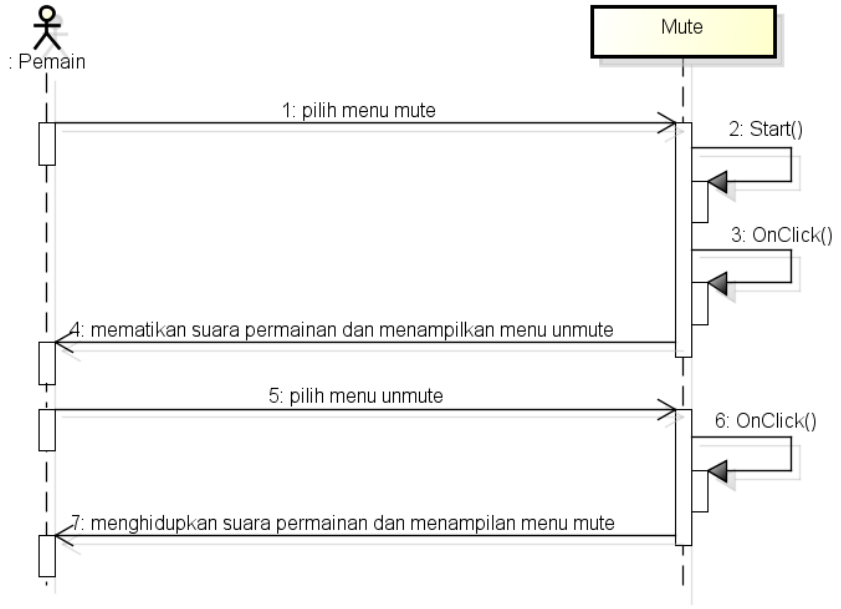
Gambar 3.24 *Sequence Diagram* Mendaur Ulang

8. *Sequence Diagram Pause*



Gambar 3.25 ***Sequence Diagram Pause***

9. *Sequence Diagram Mute*

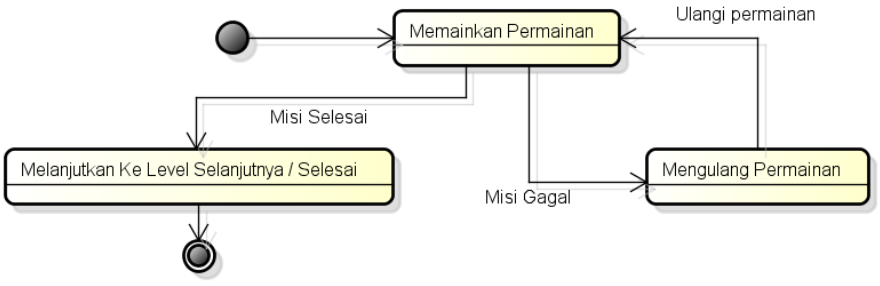


Gambar 3.26 *Sequence Diagram Mute*

3.1.5.4 *Statechart Diagram*

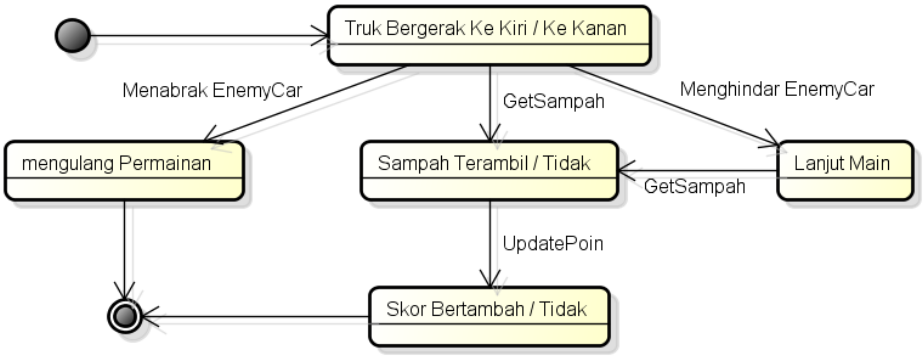
Statechart diagram digunakan untuk menggambarkan perubahan status atau transisi status dari sebuah mesin atau sistem atau objek. Perubahan tersebut digambarkan dalam suatu graf berarah.

1. *Statechart Diagram* Memainkan Permainan



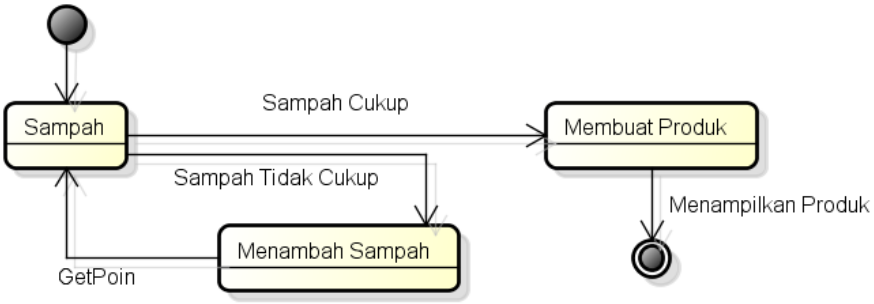
Gambar 3.27 *Statechart Diagram* Memainkan Permainan

2. *Statechart Diagram* Mengumpulkan Sampah



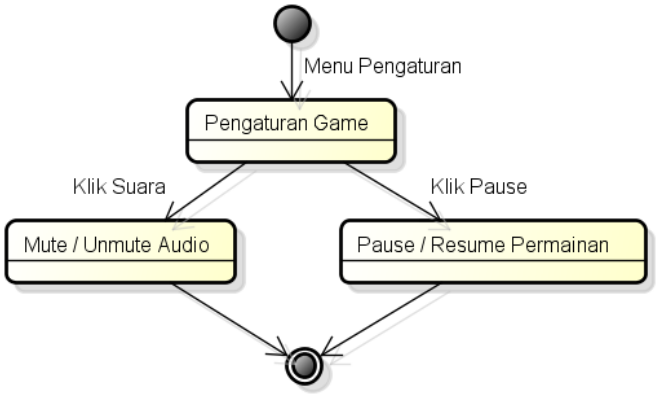
Gambar 3.28 *Statechart Diagram* Mengumpulkan Sampah

3. *Statechart Diagram* Mendaur Ulang



Gambar 3.29 *Statechart Diagram* Mendaur Ulang

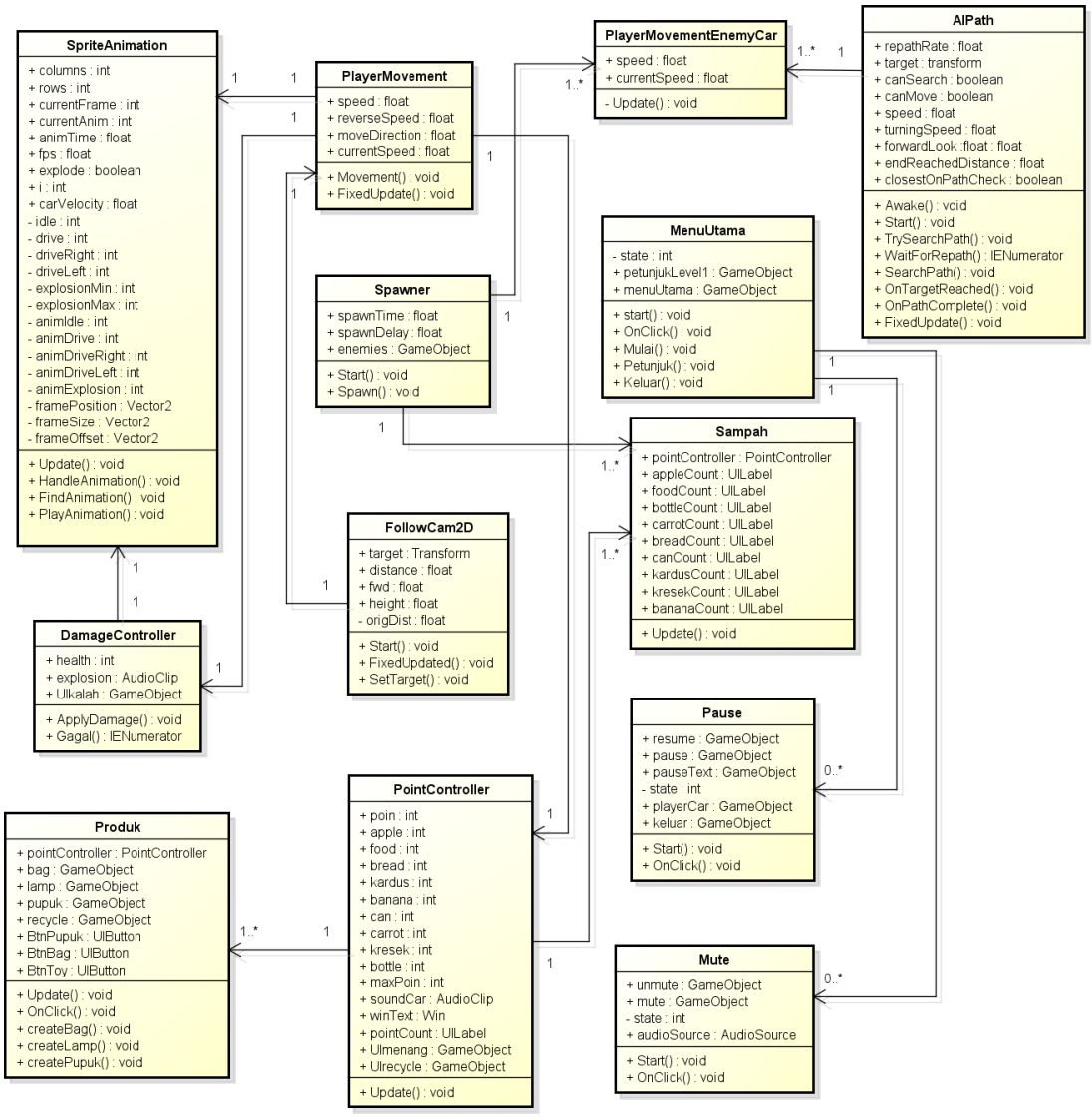
4. *Statechart Diagram* Pengaturan



Gambar 3.30 *Statechart Diagram* Pengaturan

3.1.5.5 *Class Diagram*

Class diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas-kelas yang ada pada struktur sistem harus dapat melakukan fungsi-fungsi sesuai dengan kebutuhan sistem sehingga pembuat perangkat lunak dapat membuat kelas-kelas di dalam program perangkat lunak sesuai dengan perancangan diagram kelas. *Class diagram* dari aplikasi *game* yang akan dibangun dapat dilihat pada Gambar 3.31.



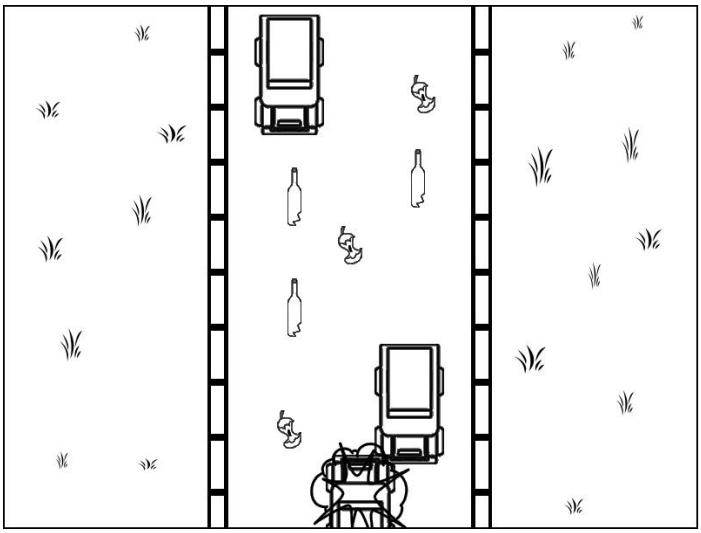
Gambar 3.31 *Class Diagram* Keseluruhan

3.2 Perancangan Sistem

Perancangan merupakan tahapan dari siklus pengembangan sistem yang didefinisikan dari kebutuhan-kebutuhan fungsional dan persiapan untuk rancang bangun implementasi yang menggambarkan bagaimana suatu sistem dibentuk, yang dapat berupa penggambaran, perancangan, dan pembuatan sketsa atau pengaturan dari beberapa elemen yang terpisah ke dalam satu kesatuan yang utuh dan berfungsi, juga menyangkut konfigurasi dari komponen-komponen perangkat keras dan perangkat lunak dari suatu sistem.

3.2.1 *Storyboard*

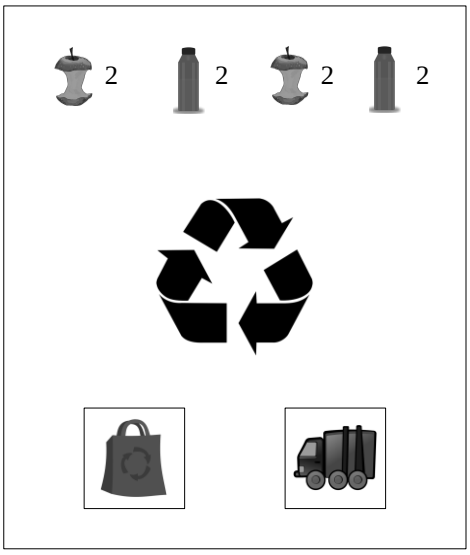
Storyboard adalah sketsa gambar yang disusun berurutan sesuai dengan naskah, dengan menggunakan *storyboard* dapat menyampaikan ide cerita kepada orang lain dengan lebih mudah karena dengan *storyboard* dapat mengimajinasikan khayalan yang sesuai dengan gambar-gambar yang ada sehingga dapat menghasilkan persepsi yang sama pada ide cerita yang dibuat. Berikut merupakan *storyboard* dari *game* yang akan dibangun:



Gambar 3.32 *Storyboard* permainan mengumpulkan sampah

Pada permainan mengumpulkan sampah ini pemain memainkan truk sampah sebagai karakternya kemudian mengumpulkan sampah-sampah yang ada di jalanan cukup dengan menempelkan truk dengan objek sampahnya. Setiap satu sampah yang diambil akan ditambahkan ke poin, tetapi terdapat tantangan dalam

mengumpulkan sampah tersebut. Mobil-mobil akan berdatangan dari arah yang berlawanan yang akan mengganggu truk dalam mengumpulkan sampah, apabila truk yang dikendalikan pemain menabrak mobil lain maka truk akan hancur dan permainan berakhir. Permainan dapat diselesaikan dengan mengumpulkan jumlah sampah sesuai target dengan waktu yang terbatas.



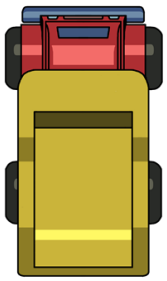
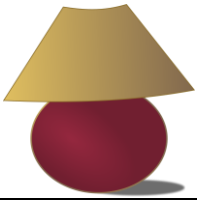
Gambar 3.33 *Storyboard* Mendaur Ulang

Di permainan mendaur ulang pemain akan mendaur ulang sampah yang telah didapat pada permainan mengumpulkan sampah. Sampah tersebut dijadikan suatu produk seperti tas, pupuk, dan lampu meja. Untuk membuat satu produk dibutuhkan kombinasi sampah dengan jumlah tertentu, misal sebuah tas dibutuhkan sampah plastik sebanyak 2 dan sampah kaleng minuman sebanyak 4. Apabila sampah tidak mencukupi maka tidak dapat membuat produk dan harus kembali ke permainan mengumpulkan sampah untuk memenuhi jumlah bahan yang dibutuhkan. Poin yang didapat disesuaikan dengan produk yang telah dibuat.

3.2.2 Pengenalan Karakter Dan Item

Pengenalan karakter merupakan pembahasan mengenai karakter-karakter yang terlibat dalam *game* yang akan dibangun. Pengenalan karakter dapat dilihat pada Tabel 3.24.

Tabel 3.24 Pengenalan Karakter Dan Item

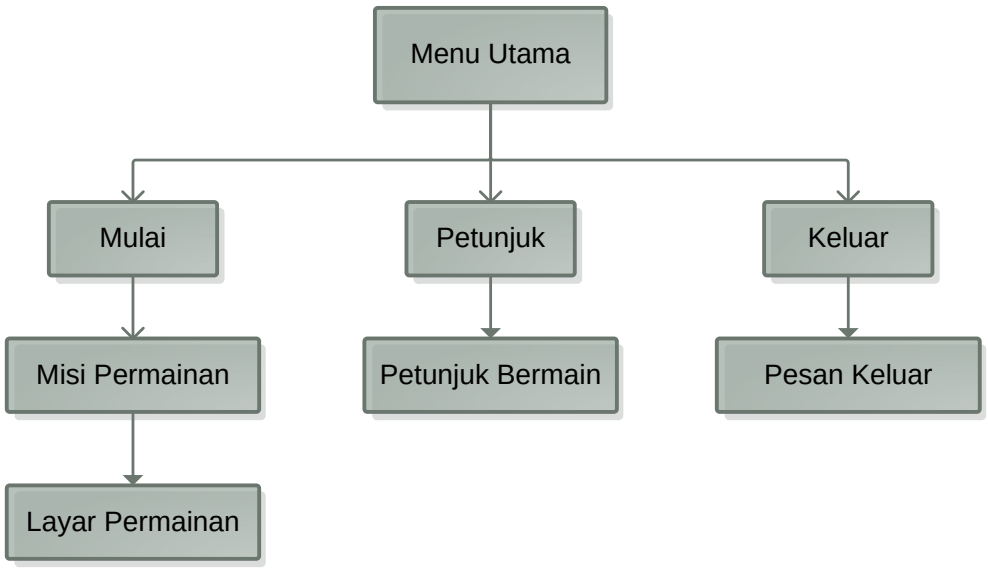
No.	Gambar	Nama	Keterangan
1		*Dump truck* / truk sampah	Karakter utama yang dimainkan oleh pemain.
2		Sampah	Objek atau item yang dikumpulkan oleh truk sampah
3		Sedan	NPC atau tantangan yang akan dihadapi oleh truk sampah. Datang dari arah berlawanan.
4	  	Tas, lampu meja, pupuk kompos	Produk atau item daur ulang

3.2.3 Perancangan Arsitektur

Perancangan arsitektur yang dibuat meliputi beberapa perancangan diantaranya struktur menu, perancangan antarmuka, dan jaringan semantik.

3.2.3.1 Perancangan Struktur Menu

Perancangan menu dibuat agar pengguna dapat menggunakan aplikasi *game* dengan mudah, sehingga pengguna dapat dengan mudah memilih menu aplikasi yang sedang berjalan. Struktur menu dapat dilihat pada Gambar 3.34.

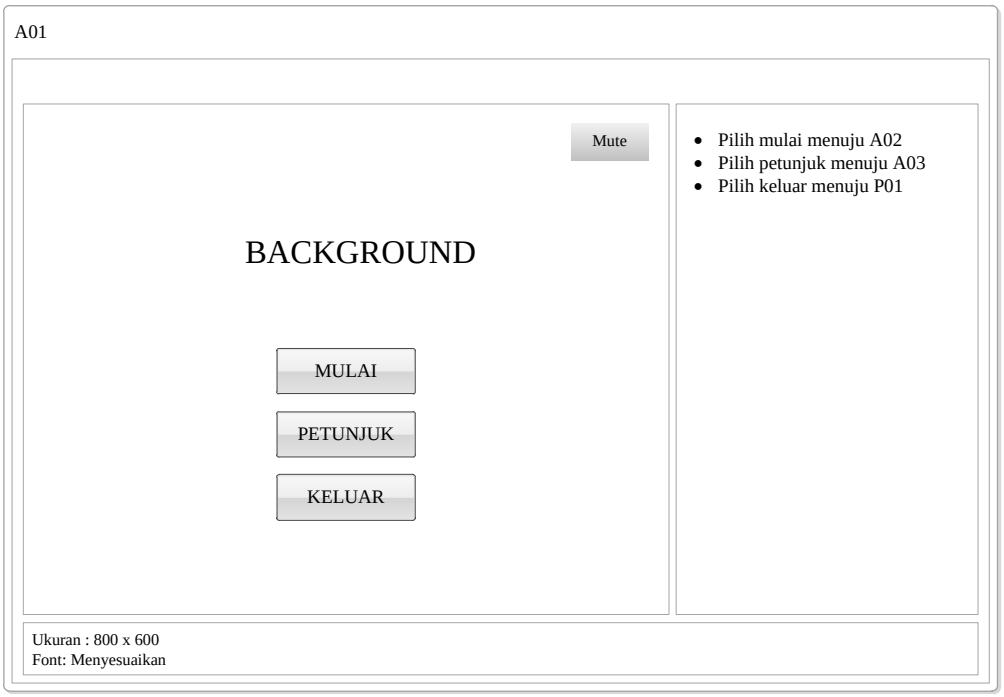


Gambar 3.34 Struktur Menu

3.2.3.2 Perancangan Antarmuka

Perancangan antarmuka dibutuhkan untuk mewakili keadaan sebenarnya dari aplikasi yang akan dibangun, berikut ini adalah perancangan antarmuka dari aplikasi yang akan dibangun.

1. Antarmuka Menu Utama



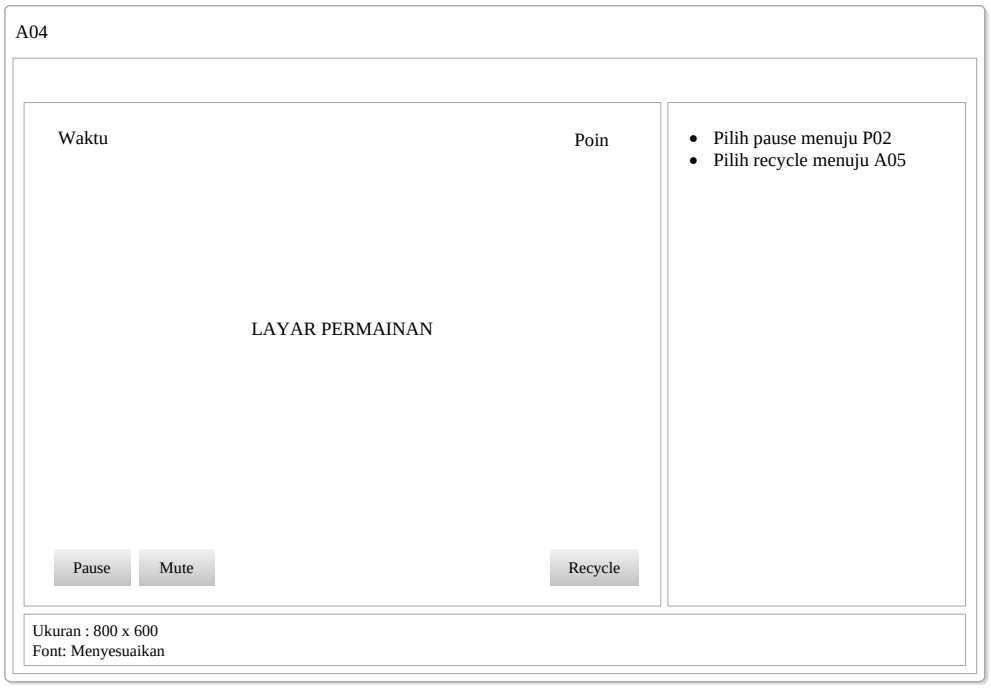
Gambar 3.35 Antarmuka Menu Utama

2. Antarmuka Menu Misi / Target Permainan



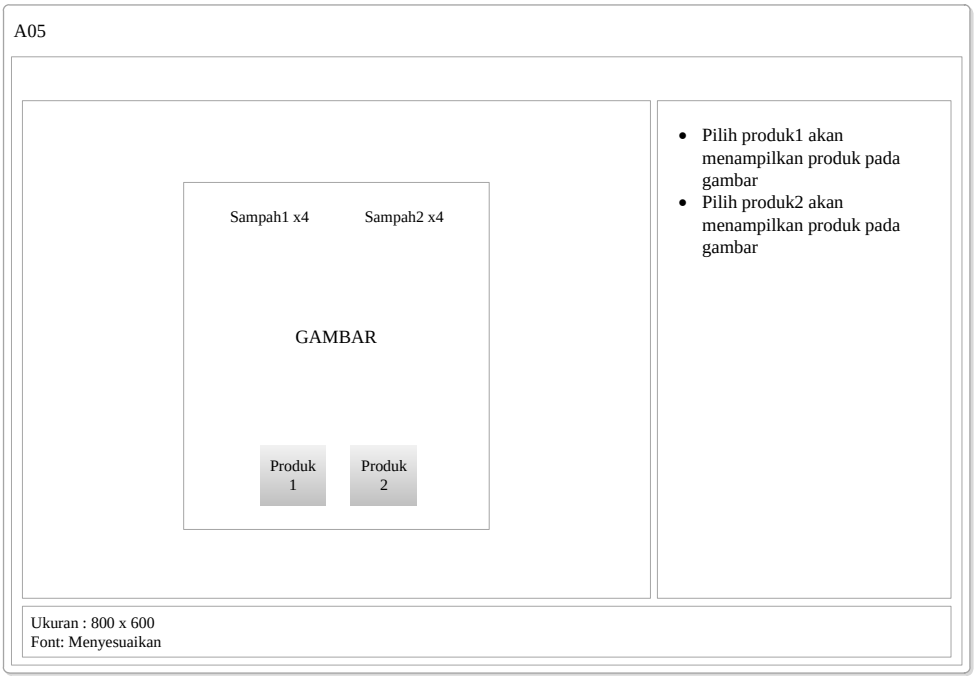
Gambar 3.36 Antarmuka Menu Mulai

3. Antarmuka Permainan



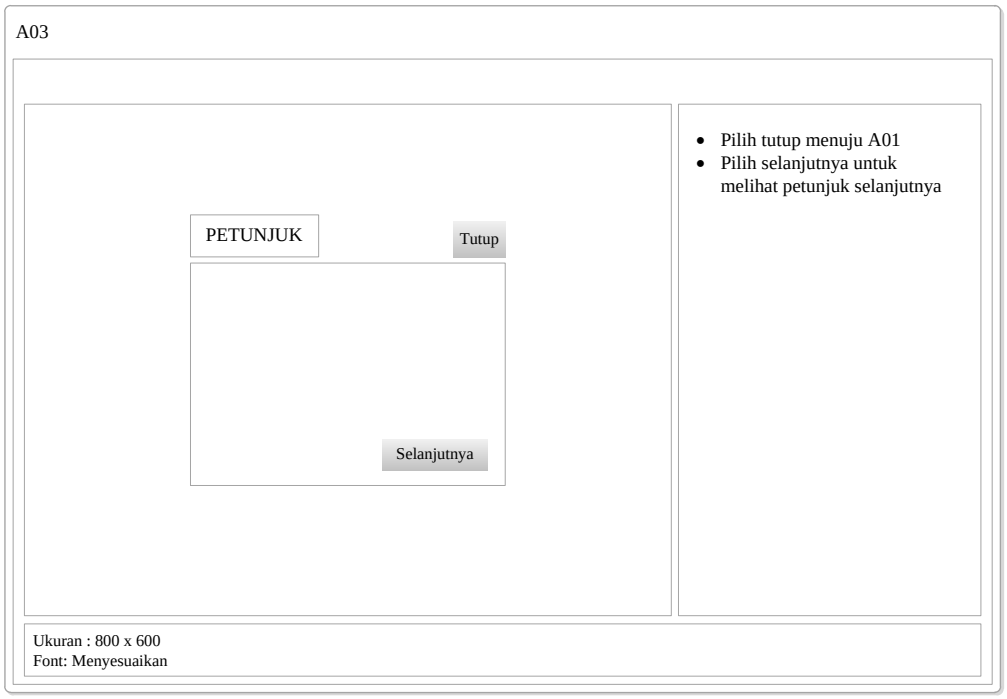
Gambar 3.37 Antarmuka Permainan Level 1

4. Antarmuka *Pop-Up* Permainan Mendaur Ulang



Gambar 3.38 Antarmuka *Pop-Up* Permainan Mendaur Ulang

5. Antarmuka Petunjuk



Gambar 3.39 Antarmuka Petunjuk Cara Bermain

6. Antarmuka Pengaturan *Pause*



Gambar 3.40 Antarmuka Pengaturan *Pause*

7. Antarmuka Permainan Gagal Diselesaikan



Gambar 3.41 Antarmuka Permainan Gagal Diselesaikan

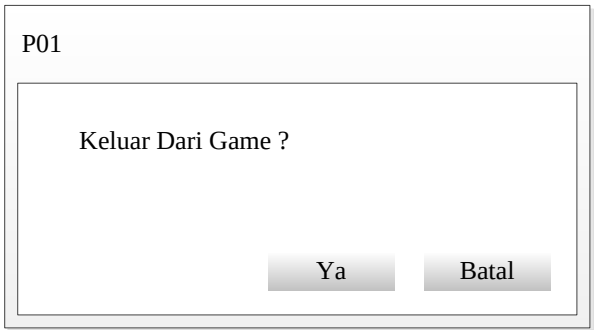
8. Antarmuka Permainan Berhasil Diselesaikan



Gambar 3.42 Antarmuka Permainan Berhasil Diselesaikan

3.2.3.3 Perancangan Pesan

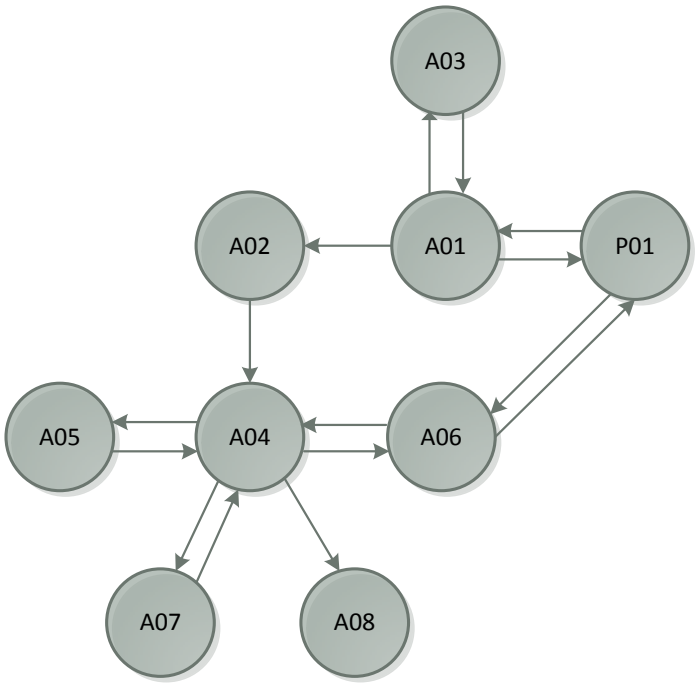
Perancangan pesan muncul sebagai konfirmasi kepada pengguna dalam mengoperasikan aplikasi.



Gambar 3.43 Perancangan Pesan P01

3.2.3.4 Jaringan Semantik

Jaringan semantik menggambarkan hubungan antar menu pada aplikasi *game* yang akan dibangun. Berikut adalah jaringan semantik pada *game* yang dibangun.



Gambar 3.44 Jaringan Semantik