BAB 3 ANALISIS DAN PERANCANGAN

Pada bab ini akan dibahas mengenai analisis dan perancangan pada sistem yang dibangun yakni penerapan algoritma *Breadth First Search* dan *Depth First Search* pada permainan Onet Fruit.

3.1 Analisis Masalah

Analisis masalah merupakan proses identifikasi serta evaluasi mengenai masalah-masalah yang timbul dari pengembangan sistem, baik mengenai aturan serta teknik-teknik yang digunakan dalam pengembangan sistem .

Adapun aturan-aturan yang harus dipenuhi dalam menjalankan game onet *fruit* ini adalah sebagai berikut:

1. Permainan ini hanya dimainkan oleh satu orang *(single player).*

2. Papan yang digunakan dalam permainan ini terdiri dari 3 level yakni:

a. level 1, papan berukuran 5x8 kotak, yang berisi 20 gambar berpasangan

b. level 2, papan berukuran 7x12 kotak, yang berisi 42 gambar berpasangan

c. level 3, papan berukuran 9x16 kotak, yang berisi 72 gambar berpasangan

Pemain hanya dapat memainkan level 3 jika telah berhasil melewati level 1 dan 2.

3. Permainan dimulai dengan papan terisi gambar penuh yang tersusun secara

acak, dan dikatakan selesai jika pemain berhasil menghabiskan gambar.

4. Pemain harus memilih dua gambar yang sama dengan syarat kedua gambar itu harus dihubungkan maksimal 3 garis maya tanpa terhalang gambar-gambar yang lain. Jika dalam papan onet tersebut sudah tidak ada lagi gambar yang dapat dipasangkan, maka sistem akan mengacak secara otomatis. Setiap gambar yang telah dicocokan oleh pemain akan hilang.

5. Setiap level diberi waktu 6 menit (360 detik) untuk mencocokan gambar.

6. Setiap gambar yang berhasil dipasangkan akan diberi nilai 20 point.

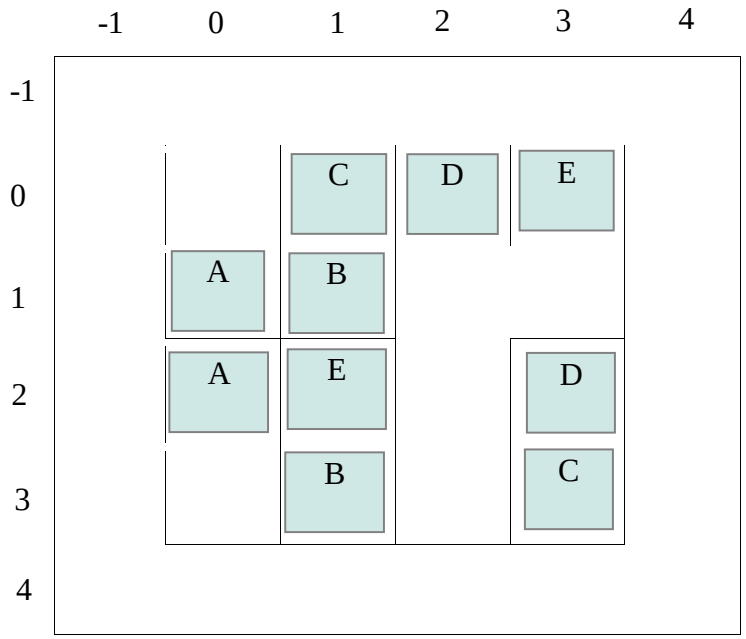
Adapun teknik yang dilakukan dalam menyelesaikan *game* onet *fruit* ini adalah dengan menggunakan algoritma BFS dan DFS. Algoritma BFS dan DFS ini akan diterapkan dalam pencarian jalan yang diasumsikan dengan titik awal (keadaan pertama) dan titik tujuan (keadaan akhir). Selain mencari jalan dari titik awal menuju titik tujuan, algoritma BFS atau DFS ini juga digunakan untuk memeriksa apakah masih ada titik yang dapat dijadikan solusi untuk menyelesaikan permainannya, jika sudah tidak ada titik yang dapat dilalui dengan 3 garis maya, maka sistem akan mengacak ulang datanya secara otomatis.

3.2 Analisis Kasus

Algoritma BFS dan DFS ini akan diterapkan pada analisis kasus pencarian jalan untuk mencari jarak atau banyaknya garis menuju tujuan, banyaknya sel menuju tujuan serta node yang diperiksa dari titik awal menuju titik tujuan pada

permainan onet *fruit*.

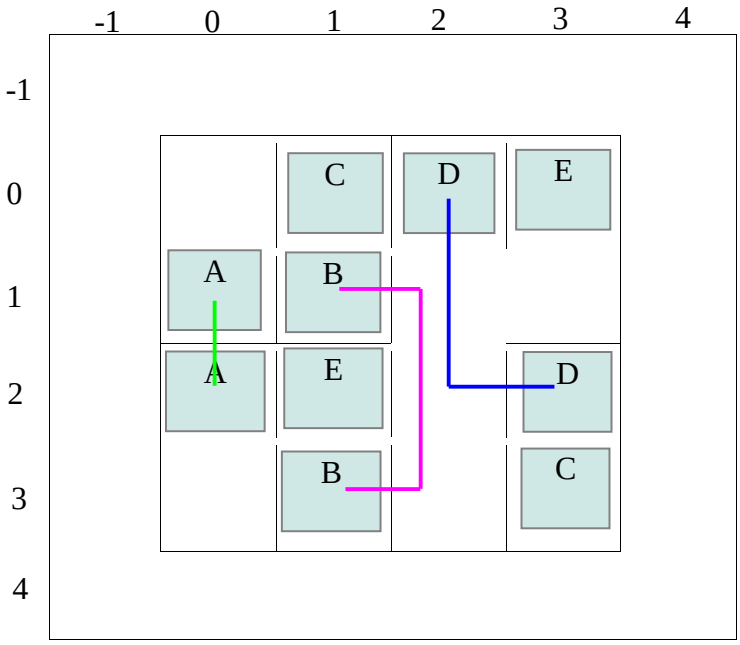
Setiap sel yang dikunjungi akan disimpan pada suatu list. List ini akan digunakan untuk melakukan pengecekan apakah kita sudah pernah melewati sel tersebut atau belum, agar kita tidak melewati sel yang sama berkali-kali. Dengan menerapkan strategi ini, kita dapat mencapai titik tujuan lebih cepat.



Gambar 3.1 Contoh permainan onet *fruit* 6X6

Pada gambar 3.1, adalah contoh permainan onet *fruit* 6x6 dimana kotak yang berisi huruf merupakan sel yang belum terpecahkan, sedangkan kotak kosong merupakan ruang kosong yang dapat dilalui untuk pencarian jalan. Cara kerja dalam pencarian jalan pada permainan onet ini yakni setiap sel akan memeriksa ke arah kiri, atas, kanan dan bawahnya. Jika arah yang dilaluinya terdapat gambar atau bukan ruang kosong maka pencarian ke arah tersebut di hentikan dan dilanjutkan pencarian ke arah lainnya. Jika arah yang dilalui untuk

menuju titik tujuan melebihi 3 garis maka pencarian pun akan dihentikan juga. Dengan cara tersebut, maka terdapat 3 solusi yang ditemukan pada gambar 3.1 di atas yakni melalui 1 garis maya pada huruf A, 2 garis maya pada huruf D dan 3 garis maya pada huruf B. Solusi pencarian jalan dapat terlihat pada gambar 3.2.



Gambar 3.2 Solusi pencarian jalan permainan onet *fruit* 6X6

3.3 Analisis Pemecahan Masalah pada Permainan Onet *Fruit*

Terdapat beberapa hal yang perlu didefinisikan terlebih dahulu dalam kasus *game* onet *fruit*. Adapun istilah- istilah yang akan dibahas yaitu :

1. Sel merupakan informasi posisi x dan posisi y pada sebuah papan.

2. Node atau simpul memiliki informasi mengenai posisi x, posisi y, jumlah cell, jumlah garis, parent dari node dan arah yang dilalui (*left*, *up*, *right*, *down*).

3. Suksesor, *child* dari node yang dituju.

3.3.1 Analisis Pemecahan Masalah dengan Algoritma *Breadth First Search*

Langkah-langkah yang dilakukan pada permainan onet *fruit* menggunakan algoritma BFS :

1. Tentukan titik awal dan titik tujuan

2. Periksa nilai titik awal dan titik tujuan, jika berbeda proses selesai.

3. Buatlah node dari titik awal dan *set parent*nya dengan nilai null lalu masukan ke dalam *linked list*

4. Periksa apakah node tersebut sama dengan titik tujuan, jika ya maka proses selesai dengan mengembalikan nilai node tersebut.

5. Periksa node tersebut apakah *parent*nya tidak sama dengan null dan tidak sama dengan ruang kosong (tidak bisa dilalui), jika benar maka tahap berikutnya tidak di proses dan dilanjut ke node berikutnya.

6. Periksa apakah node.x > -1atau node.x < lebar papar atau node.y > -1 atau node.y < tinggi papan, jika ya masuk tahap berikutnya, jika tidak dilanjut ke node berikutnya.

7. Periksa apakah jumlah garis pada *parent* tersebut sudah 3 garis, Jika sudah maka dilanjut ke arah node berikutnya.

8. Bangkitkan semua suksesor pada node tersebut dengan cara mengunjungi setiap arah (*left, up, right, down*)

Left : (node.x – 1, node.y);

Up : (node.x, node.y – 1);

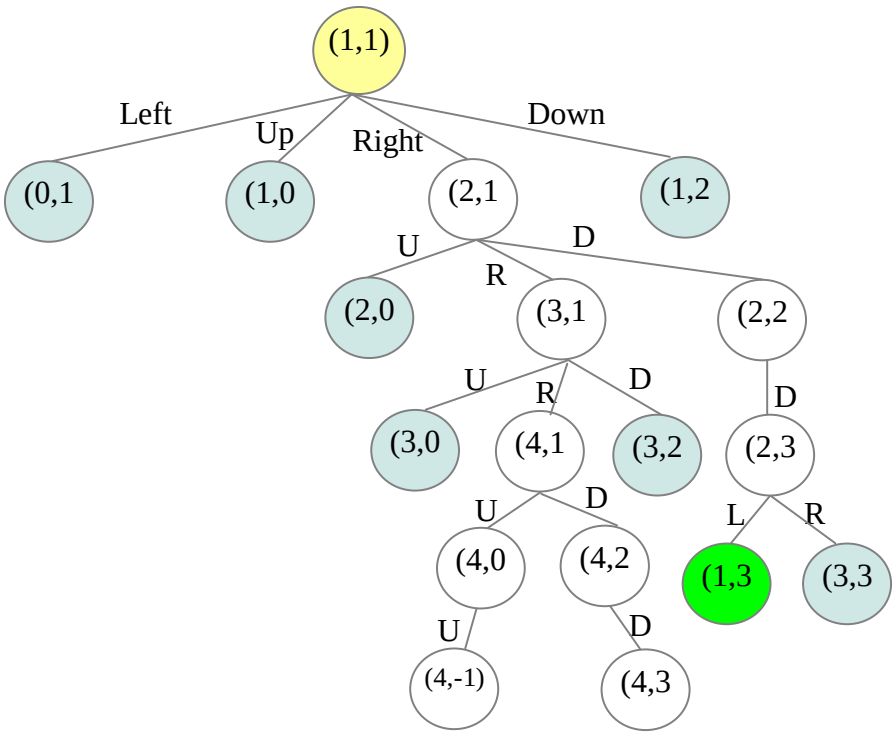
Right : (node.x + 1, node.y);

Down : (node.x, node.y + 1);

9. Masukan semua suksesor / *child* tersebut kedalam *linked list* dalam bentuk node dimana parentnya adalah node sebelumnya, dengan syarat node tersebut belum pernah dilalui.

10. Ulangi langkah 4 – 9.

Berdasarkan papan onet *fruit* yang terdapat pada analisis kasus, maka pemecahan masalah dengan algoritma BFS untuk pencarian jalan pada huruf B dengan titik awal (1,1) dan titik tujuan (1,3) dalam pohon pencarian adalah sebagai berikut:



Gambar 3.3 Pohon pencarian jalan pada huruf B menggunakan algoritma BFS pada permainan onet *fruit* 6x6

Pada gambar 3.3 pohon pencarian jalan, node yang berwarna kuning merupakan titik awal, sedangkan node berwarna biru merupakan node yang tidak dapat dilalui atau node tersebut dalam keadaan CLOSED, node yang berwarna putih merupakan node yang dapat dilalui (ruang kosong) atau node yang berada di OPEN, dan node berwarna hijau merupakan titik tujuan. Jika node sudah pada kondisi CLOSED maka diabaikan. Berdasarkan gambar 3.3 tersebut maka langkah atau jalan yang dilalui dari titik awal (1,1) menuju titik tujuan (1,3) adalah (1,1) – (2,1) – (2,2) – (2,3) – (1,3). Untuk kondisi pada gambar 3.3 dapat dijelaskan langkahnya sebagai berikut:

a. Langkah pertama, karena hanya terdapat satu node yaitu titik awal (1,1) maka maka titik awal terpilih sebagai BestNode dan dipindahkan ke CLOSED.

Kemudian dibangkitkan semua suksesor (1,1) yang berada di posisi *left, up, right, down* yaitu node (0,1), (1,0), (2,1) dan (1,2). Karena ke empat suksesor tidak ada di OPEN maupun CLOSED, maka ke empatnya dimasukkan ke OPEN . Langkah pertama ini menghasilkan OPEN = [(0,1), (1,0), (2,1),(1,2)] dan CLOSED = [A(1,1)].

b. Langkah ke dua, semua suksesor (1,1) kemudian di cek nilainya. Jika menghasilkan nilai -1 maka node tersebut dapat dilalui (ruang kosong) dan statusnya di OPEN. Jika memiliki nilai (bukan ruang kosong) maka statusnya di CLOSED. Pada suksesor (1,1), node (0,1) memiliki nilai maka dipindahkan ke CLOSED, pengecekan berlanjut ke node (1,0) dan hasilnya memiliki nilai juga maka dipindahkan ke CLOSED. Lalu node (2,1) ternyata menghasilkan nilai -1 atau ruang kosong, maka node (2,1) dipindahkan ke CLOSED lalu dibangkitkan dan di OPEN semua suksesor (2,1) yaitu node (2,0), (3,1) dan (2,2). Adapun node (1,1) tidak di OPEN karena telah dilalui sebelumnya. Dan node (1,2) dari suksesor (1,1) menghasilkan nilai (bukan ruang kosong) maka di CLOSED.

c. Ikuti langkah (b) sampai menemukan titik tujuan. Jika node yang dikunjungi sama dengan node tujuan maka proses berenti dan node yang berada dibelakangnya tidak perlu di cek. Hasil pencarian jalan bisa ditelusuri balik dari titik tujuan menuju titik awal karena setiap simpul hanya memiliki satu parent.

Simpul-simpul yang diperiksa tiap langkah dapat dilihat pada tabel 3.1.

Tabel 3.1 Simpul / titik pencarian menggunakan BFS

OPEN	CLOSED
[(0,1),(1,0),(2,1),(1,2)]	[(1,1)]
[(1,0),(2,1),(1,2)]	[(1,1),(0,1)]
[(2,1),(1,2)]	[(1,1),(0,1),(1,0)]
[(1,2),(2,0), (3,1),(2,2)]	[(1,1),(0,1),(1,0),(2,1)]
[(2,0), (3,1),(2,2)]	[(1,1),(0,1),(1,0),(2,1),(1,2)]
[(3,1),(2,2)]	[(1,1),(0,1),(1,0),(2,1),(1,2),(2,0)]
[(2,2),(3,0),(4,1),(3,2)]	[(1,1),(0,1),(1,0),(2,1),(1,2),(2,0),(3,1)]
[(3,0),(4,1),(3,2),(2,3)]	[(1,1),(0,1),(1,0),(2,1),(1,2),(2,0),(3,1),(2,2)]
[(4,1),(3,2),(2,3)]	[(1,1),(0,1),(1,0),(2,1),(1,2),(2,0),(3,1),(2,2),(3,0)]
[(3,2),(2,3),(4,0),(4,2)]	[(1,1),(0,1),(1,0),(2,1),(1,2),(2,0),(3,1),(2,2),(3,0),(4,1)]
[(2,3),(4,0),(4,2)]	[(1,1),(0,1),(1,0),(2,1),(1,2),(2,0),(3,1),(2,2),(3,0),(4,1),(3,2)]
[(4,0),(4,2),(1,3),(3,3)]	[(1,1),(0,1),(1,0),(2,1),(1,2),(2,0),(3,1),(2,2),(3,0),(4,1),(3,2),(2,3)]
[(4,2),(1,3),(3,3),(4,-1)]	[(1,1),(0,1),(1,0),(2,1),(1,2),(2,0),(3,1),(2,2),(3,0),(4,1),(3,2),(2,3),(4,0)]
[(1,3),(3,3),(4,-1),(4,3)]	[(1,1),(0,1),(1,0),(2,1),(1,2),(2,0),(3,1),(2,2),(3,0),(4,1),(3,2),(2,3),(4,0),(4,2)]

Berdasarkan simpul-simpul pada tabel 3.1, proses pencarian dari titik awal (1,1) menuju titik tujuan (1,3) menggunakan algoritma BFS ini terdapat 17 node yang telah dilewati.

3.3.2 Analisis Pemecahan Masalah dengan Algoritma *Depth First Search*

Langkah-langkah yang dilakukan pada permainan onet *fruit* menggunakan algoritma DFS :

1. Tentukan titik awal dan titik tujuan

2. Periksa nilai titik awal dan titik tujuan, jika berbeda proses selesai.

3. Buatlah node dari titik awal, dengan parent di set *null*.

4. Periksa apakah node tersebut sama dengan titik tujuan, jika ya proses selesai dengan mengembalikan nilai node tersebut..

5. Periksa node tersebut apakah *parent*nya tidak sama *null* dan tidak sama dengan ruang kosong (tidak bisa dilalui), jika benar maka tahap berikutnya tidak di proses dan dilanjut ke node berikutnya

6. Periksa apakah node.x > -1atau node.x < lebar papar atau node.y > -1 atau node.y < tinggi papan, jika ya masuk tahap berikutnya, jika tidak dilanjut ke node berikutnya.

7. Periksa apakah node.x dan node.y sudah pernah dikunjungi atau belum, jika sudah dilanjut ke arah berikutnya jika belum masuk ke tahap berikutnya.

8. Periksa apakah jumlah garis pada *parent* tersebut sudah 3 garis, Jika sudah maka dilanjut ke arah node berikutnya.

9. Bangkitkan semua suksesor pada node tersebut dengan cara mengunjungi setiap arah (*left, up, right, down*).

Left : (node.x – 1, node.y);

Buat node dari titik *left* tersebut dan *set parent*nya dengan node sebelumnya. Jika arah *left* pada node tersebut masih bisa dikunjungi ulangi langkah 4-9, tetapi jika sudah tidak dapat dikunjungi maka ke arah selanjutnya yakni Up

Up : (node.x, node.y – 1);

Buat node dari titik *up* tersebut dan set *parent*nya dengan node sebelumnya. Jika arah *up* pada node tersebut masih bisa dikunjungi ulangi langkah 4-9, tetapi jika sudah tidak dapat dikunjungi maka ke arah selanjutnya yakni Right

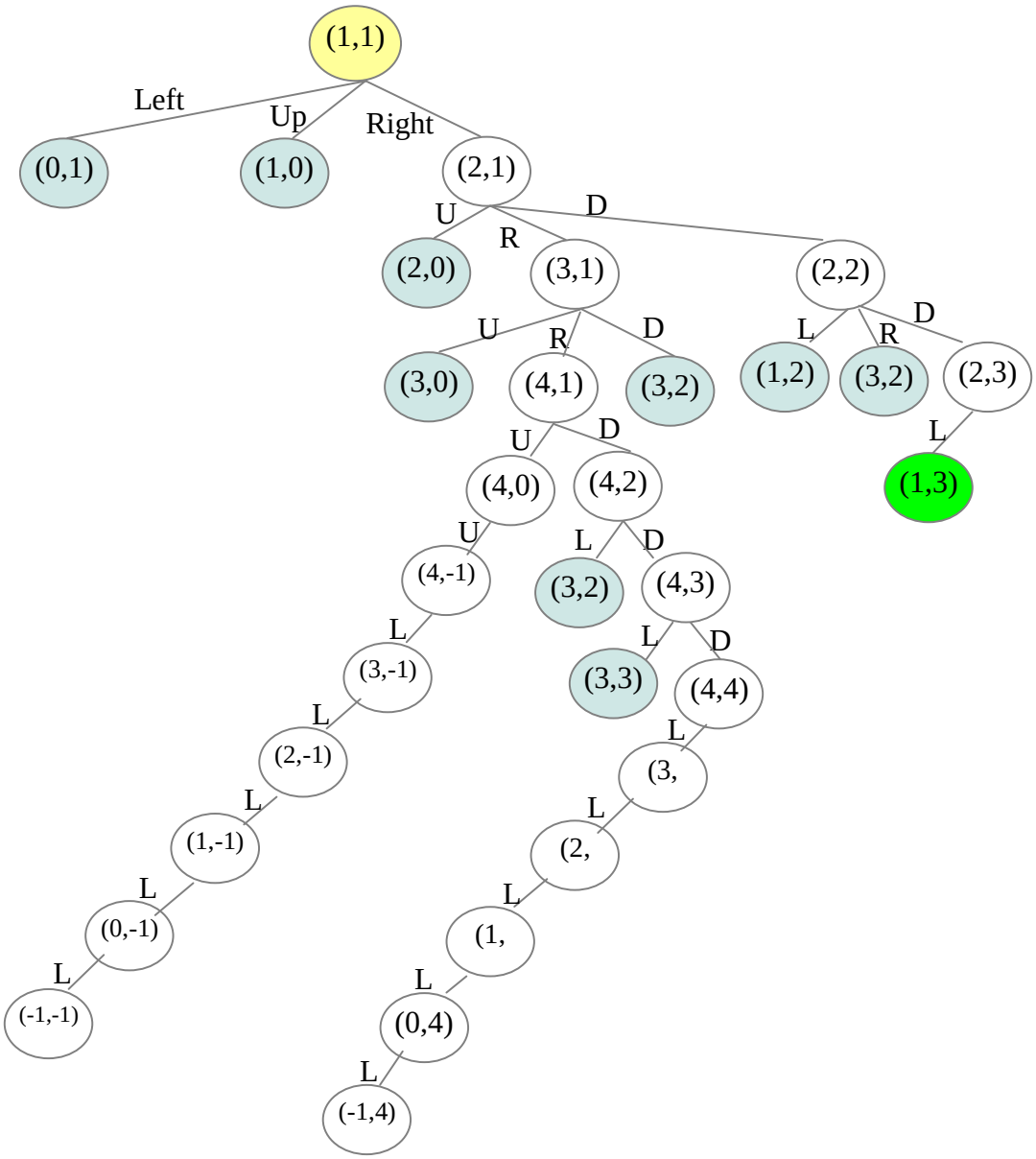
Right : (node.x + 1, node.y);

Buat node dari titik *right* tersebut dan set *parent*nya dengan node sebelumnya. Jika arah *right* pada node tersebut masih bisa dikunjungi ulangi langkah 4-9, tetapi jika sudah tidak dapat dikunjungi maka ke arah selanjutnya yakni Down

Down : (node.x, node.y + 1);

Buat node dari titik *down* tersebut dan set *parent*nya dengan node sebelumnya. Jika arah *down* pada node tersebut masih bisa dikunjungi ulangi langkah 4-9.

Berdasarkan papan onet *fruit* yang terdapat pada analisis kasus, maka pemecahan masalah dengan algoritma DFS untuk pencarian jalan pada huruf B ini yakni dengan mengunjungi setiap node yang dimulai dari titik awal (1,1) hingga mencapai titik tujuan (1,3).



Gambar 3.4 Pohon pencarian jalan pada huruf B menggunakan algoritma DFS pada permainan onet *fruit* 6x6

Pada gambar 3.4 pohon pencarian jalan, node yang berwarna kuning merupakan titik awal, sedangkan node berwarna biru merupakan node yang tidak dapat dilalui atau node tersebut dalam keadaan CLOSED, node yang berwarna putih merupakan node yang dapat dilalui (ruang kosong) atau node yang berada di OPEN, dan node berwarna hijau merupakan titik tujuan. Jika node sudah pada

kondisi CLOSED maka diabaikan.

Berdasarkan gambar 3.4 tersebut maka langkah atau jalan yang dilalui dari titik awal (1,1) menuju titik tujuan (1,3) menggunakan algortima DFS adalah (1,1) – (2,1) – (2,2) – (2,3) – (1,3). Dengan melewati 31 node, sebanyak 3 garis, dan 5 cell menuju titik tujuan. Untuk kondisi pada gambar 3.4 dapat dijelaskan langkahnya sebagai berikut:

a. Langkah pertama, karena hanya terdapat satu node yaitu titik awal (1,1) maka maka titik awal terpilih sebagai BestNode dan dipindahkan ke CLOSED.

b. Kemudian dibangkitkan semua suksesor node tersebut yang berada di posisi *left* sampai habis (bernilai *null*) atau nilai node sama dengan -1 (bukan ruang kosong). Kemudian dilanjut ke posisi Up, right, dan down.

c. Ulangi langkah point b dari node paling akhir menuju node awal sampai ditemukan titik tujuan.

d. Jika node yang dikunjungi sama dengan node tujuan maka proses berhenti. Hasil pencarian jalan bisa ditelusuri balik dari titik tujuan menuju titik awal karena setiap simpul hanya memiliki satu parent.

3.4 Analisis Kebutuhan Non Fungsional

Analisis kebutuhan non-fungsional merupakan proses indentifikasi dan evaluasi mengenai kebutuhan sistem maupun pengguna, sehingga diharapkan

game yang dibangun nanti dapat berjalan dengan baik.

Kebutuhan *hardware* dan *software* dalam membangun sistem *game* onet *fruit* ini adalah:

a. Spesifikasi *hardware* yang direkomendasikan adalah sebagai berikut:

- Procesor dengan kecepatan 1,6 GHz
- RAM DDR3 2 GB
- Harddisk 250 Gb
- Monitor / LCD
- Keyboard dan Mouse

b. Spesifikasi *software*

- Sistem operasi Linux / windows
- Eclipse sebagai IDE
- Java
- Android SDK dan ADT

Sedangkan kebutuhan dari segi pengguna yakni mempunyai *mobile* android yang *support* android versi 2.2 (Froyo) atau terinstal android emulator di

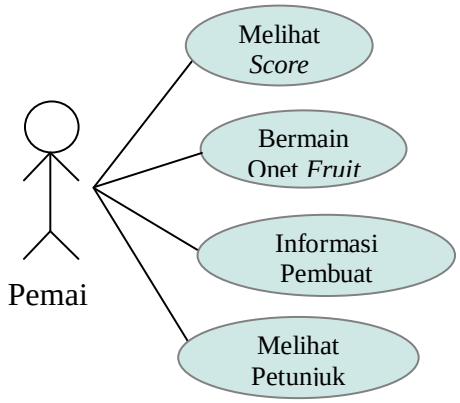
PC (*Personal Computer*) / Laptop.

3.5 Analisis Kebutuhan Fungsional

Analisis kebutuhan fungsional akan membahas perancangan sistem di dalam aplikasi. Perancangan sistem adalah penggambaran, perancangan dan pembuatan sketsa Pada tahap ini, *tools* / alat bantu yang digunakan untuk menganalisis dan merancang sistem adalah UML (*Unified Modelling Language*).

3.5.1 Use Case Diagram

Use case diagram digunakan untuk memodelkan proses bisnis dan merepresentasikan sebuah interaksi antara aktor dengan sistem. Aktor disini adalah pemain, yang mana pemain dapat melihat *score* dari permainan, bermain onet fruit, melihat informasi pembuat dan bagaimana cara bermain.



Gambar 3.5 Use Case Diagram pada permainan onet *fruit*

3.5.1.1 Skenario Use Case

Skenario setiap bagian pada *use case* menunjukkan proses apa yang terjadi pada setiap bagian didalam use case tersebut, dimana user memberikan perintah

pada setiap bagian dan respon apa yang diberikan oleh sistem kepada user setelah user memberikan perintah pada setiap bagian-bagian *use case*.

Tabel 3.2 Skenario *Use Case* Pilihan *Score*

Identifikasi	
Nama	*Score*
Tujuan	Melihat informasi nilai yang pernah di peroleh
Deskripsi	Proses untuk melihat informasi nilai yang pernah di peroleh
Aktor	*Gamer*
Skenario Utama	
Kondisi awal	Pengguna berada di menu utama aplikasi *game* Onet *fruit*
Aksi Aktor	Reaksi Sistem
Memilih menu *score*	Aplikasi merespon dengan cara memanggil dan melihat kelas *score*
Kondisi Akhir	Aplikasi masuk ke dalam informasi *score* aplikasi game onet *fruit*

Tabel 3.3 Skenario *Use Case* Pilihan Bermain Onet *fruit*

Identifikasi	
Nama	Bermain Onet *Fruit*
Tujuan	Memulai permainan onet *fruit*
Deskripsi	Proses untuk memulai permainan onet *fruit*
Aktor	*Gamer*
Skenario Utama	
Kondisi awal	Pengguna berada di menu utama aplikasi *game* Onet *fruit*
Aksi Aktor	Reaksi Sistem
Memilih menu bermain onet *fruit*	Aplikasi merespon dengan cara memanggil dan melihat kelas permainan
Kondisi Akhir	Aplikasi masuk ke dalam sistem permainan onet *fruit*

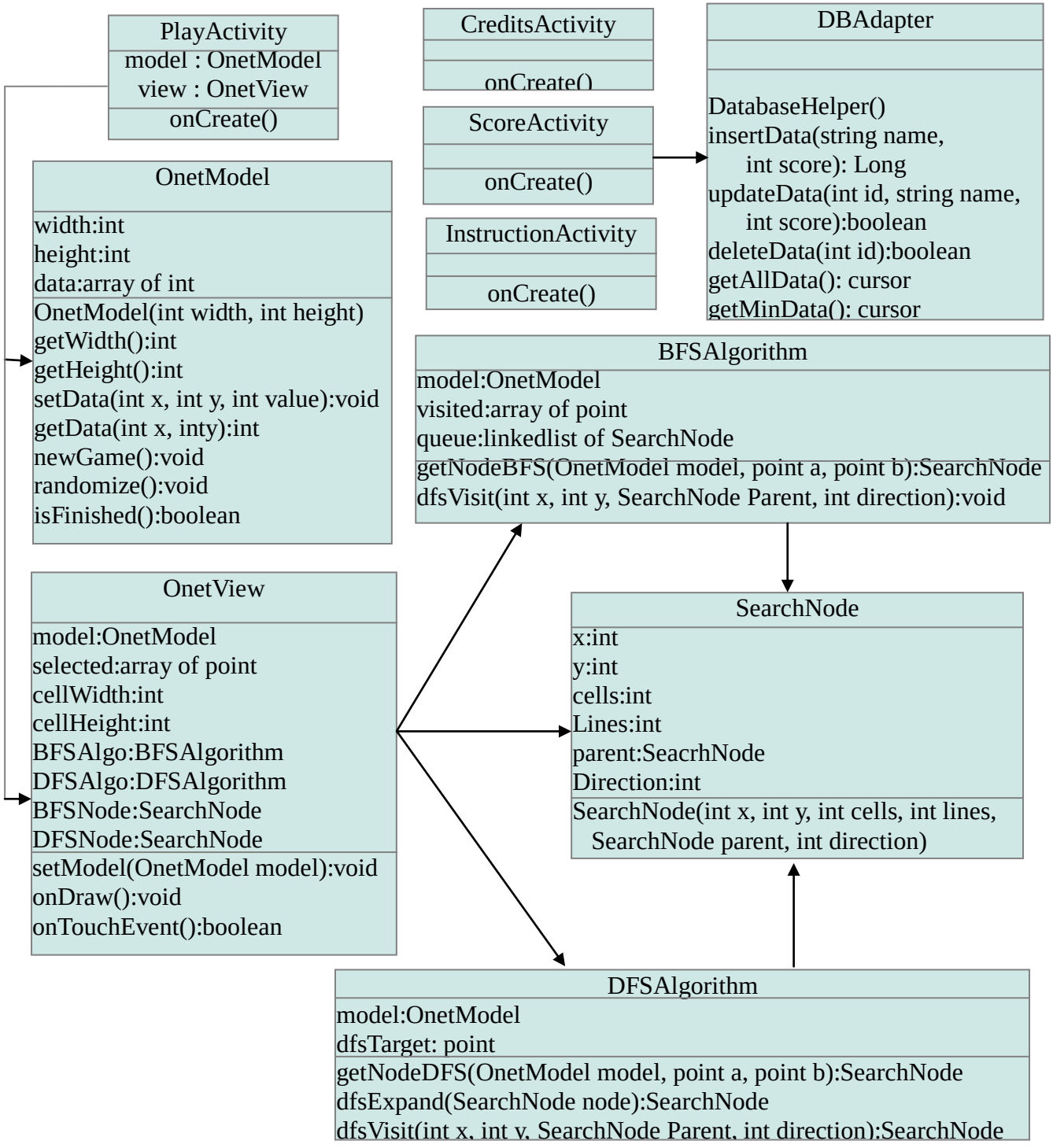
Tabel 3.4 Skenario *Use Case* Pilihan Pembuat

Identifikasi	
Nama	Pembuat
Tujuan	Melihat informasi mengenai pembuat aplikasi
Deskripsi	Proses untuk melihat informasi pembuat aplikasi
Aktor	*Gamer*
Skenario Utama	
Kondisi awal	Pengguna berada di menu utama aplikasi *game* Onet *fruit*
Aksi Aktor	Reaksi Sistem
Memilih menu pembuat	Aplikasi merespon dengan cara memanggil dan melihat kelas pembuat
Kondisi Akhir	Aplikasi masuk ke dalam informasi pembuat aplikasi game onet *fruit*

Tabel 3.5 Skenario *Use Case* Pilihan Petunjuk

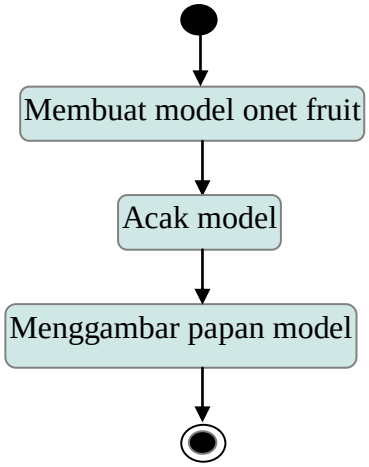
Identifikasi	
Nama	Petunjuk
Tujuan	Melihat informasi mengenai petunjuk aplikasi
Deskripsi	Proses untuk melihat informasi mengenai petunjuk aplikasi
Aktor	*Gamer*
Skenario Utama	
Kondisi awal	Pengguna berada di menu utama aplikasi *game* Onet *fruit*
Aksi Aktor	Reaksi Sistem
Memilih menu pembuat	Aplikasi merespon dengan cara memanggil dan melihat kelas petunjuk
Kondisi Akhir	Aplikasi masuk ke dalam informasi petunjuk / cara bermain pada aplikasi game onet *fruit*

3.5.2 Class Diagram

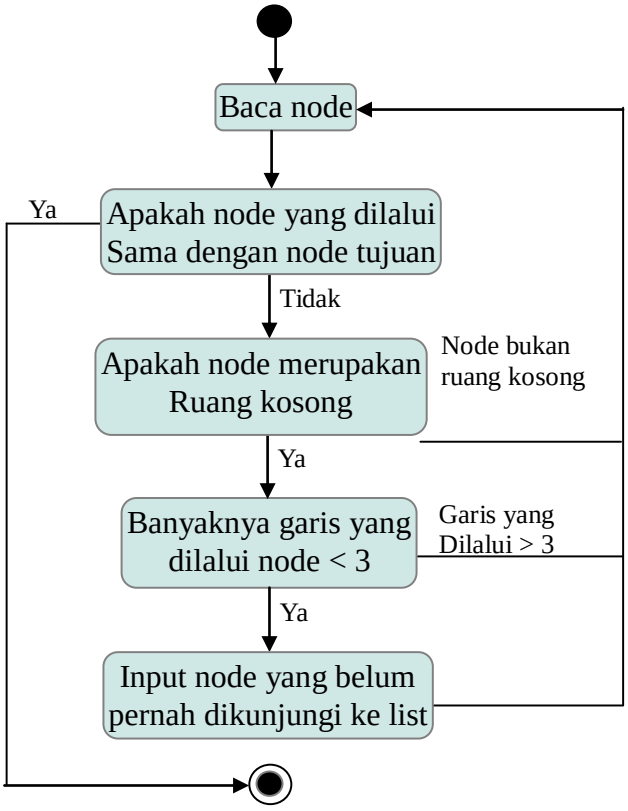


Gambar 3.6 *Class* Diagram pada Permainan Onet *Fruit*

3.5.3 Statechart Diagram

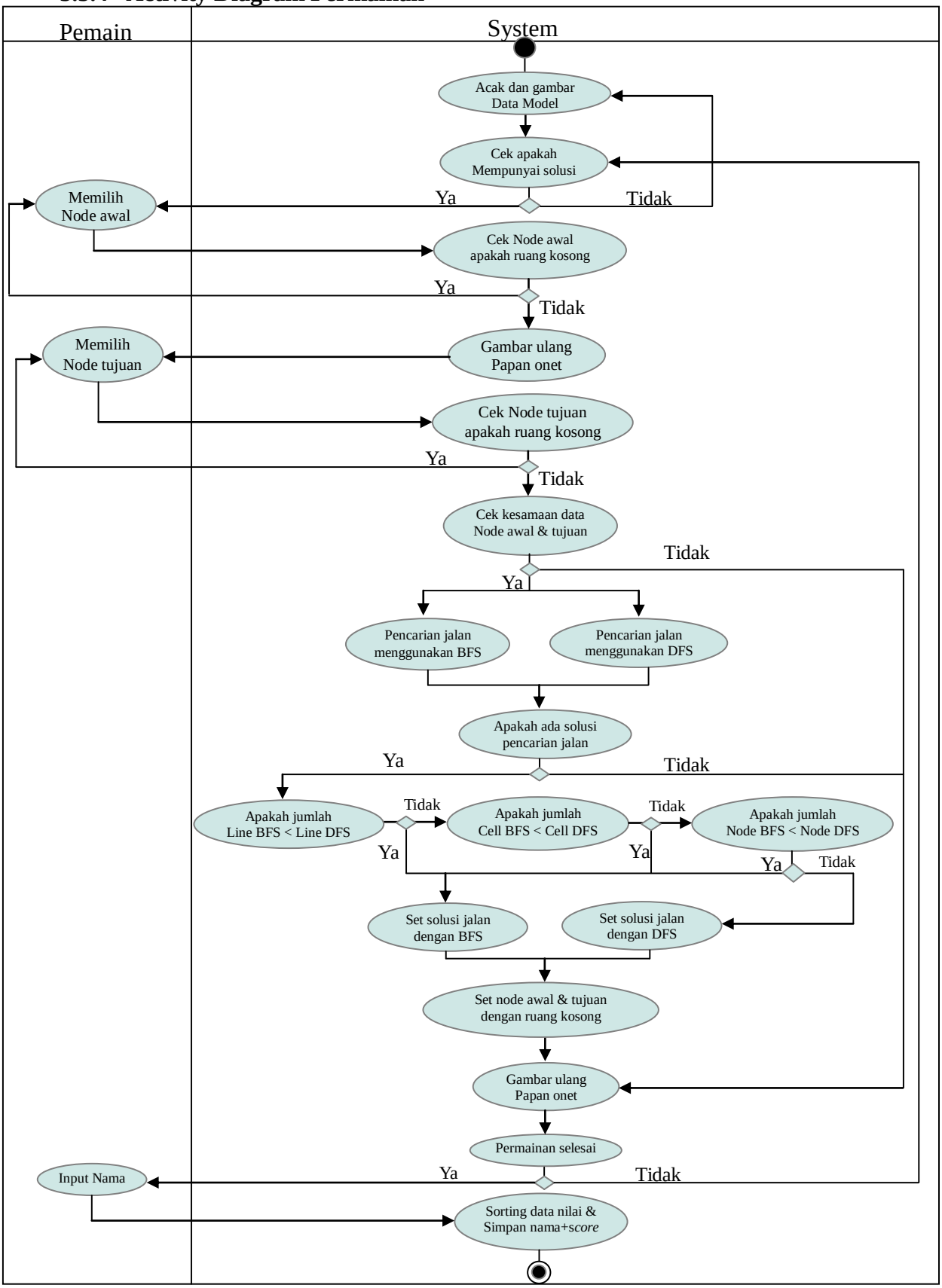


Gambar 3.7 *StateChart* Diagram pada awal permainan

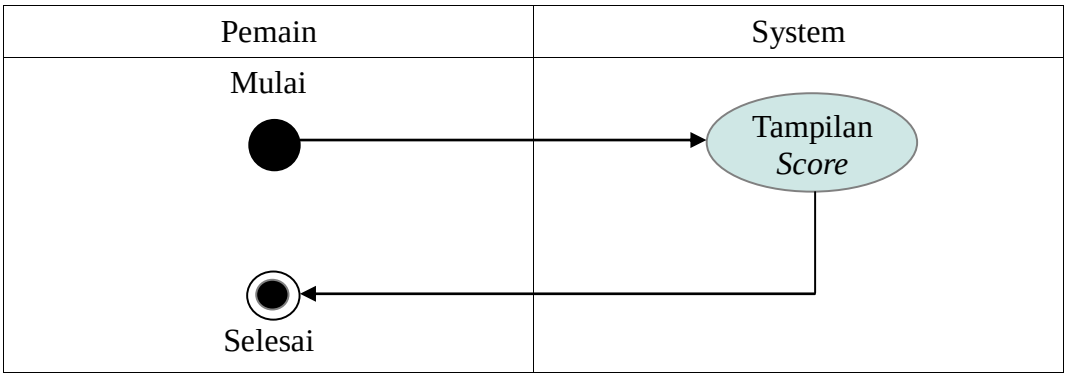


Gambar 3.8 *StateChart* Diagram pada proses pencarian jalan

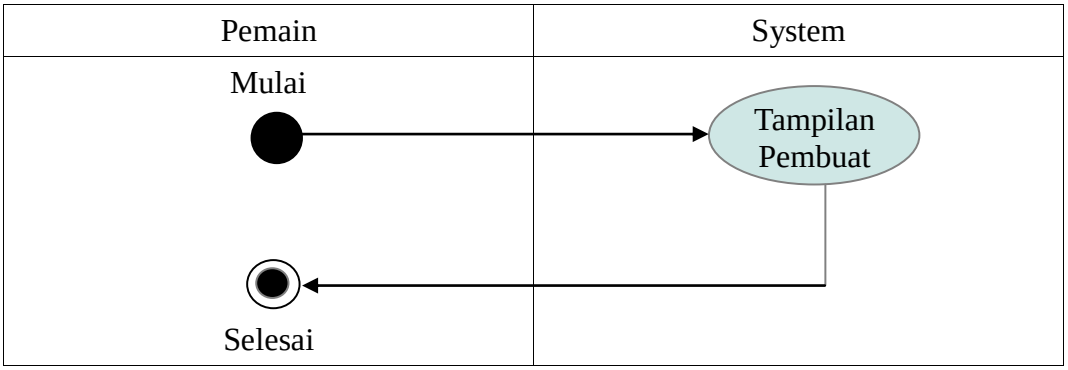
3.5.4 Activity Diagram Permainan



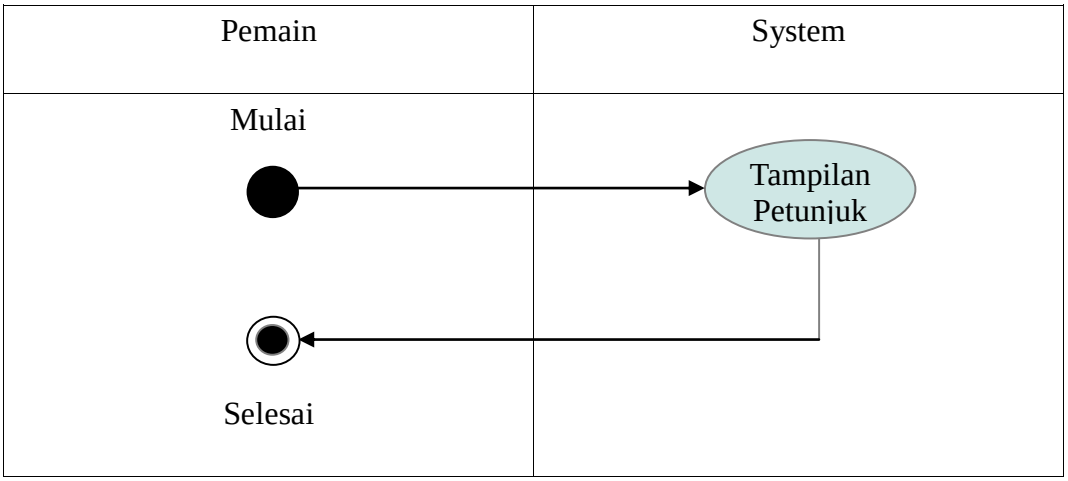
Gambar 3.9 *Activity Diagram* pada permainan onet *fruit*



Gambar 3.10 *Activity Diagram* pada proses *Score*

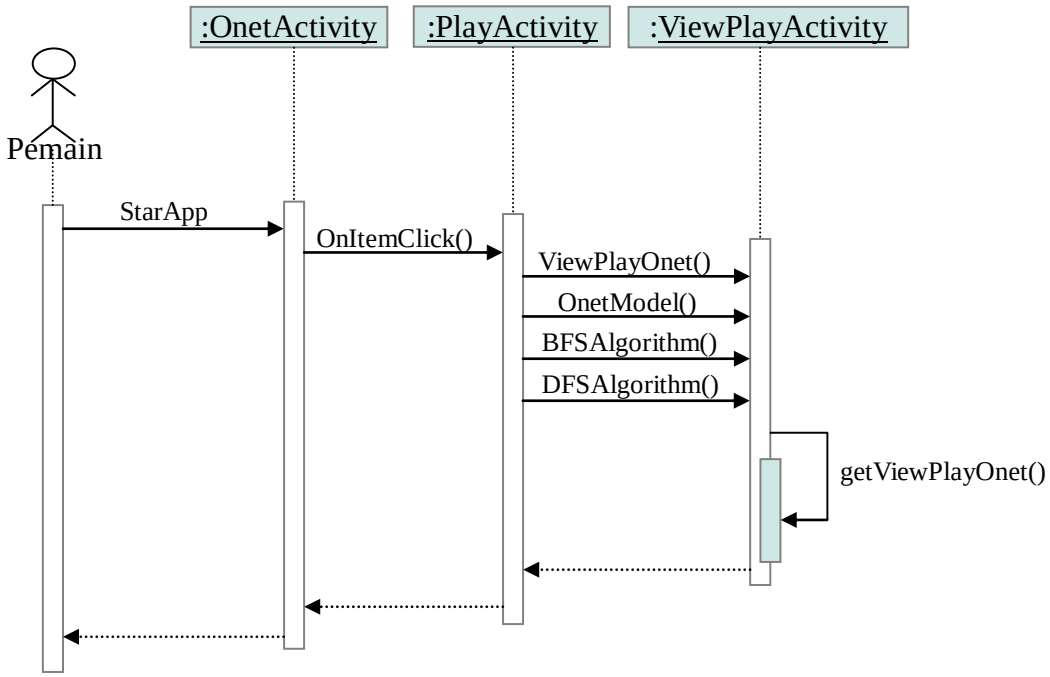


Gambar 3.11 *Activity Diagram* pada proses pembuat

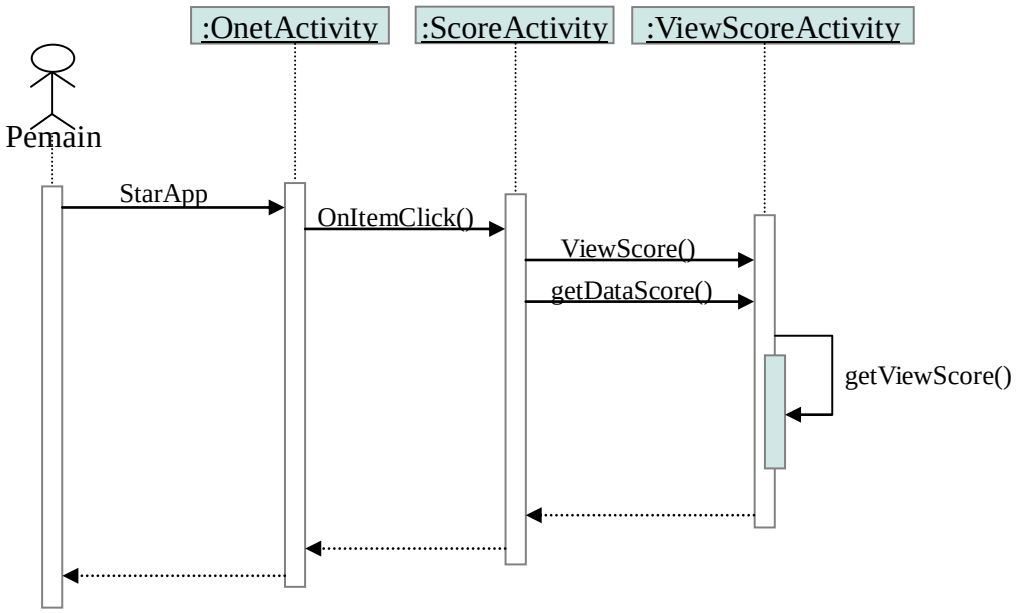


Gambar 3.12 *Activity Diagram* pada proses petunjuk

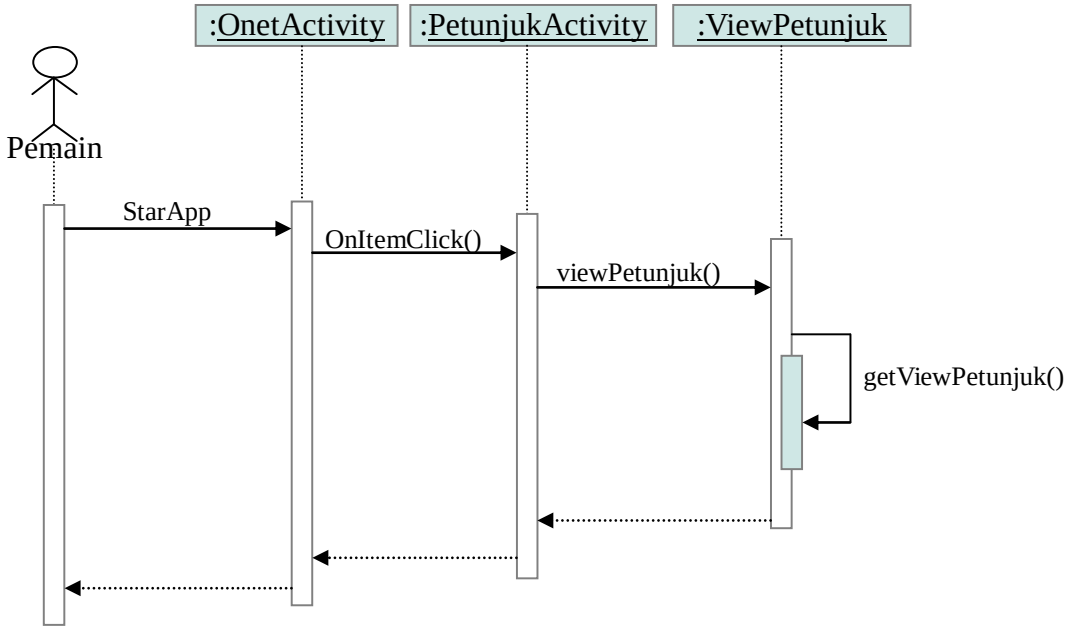
3.5.5 Sequence Diagram



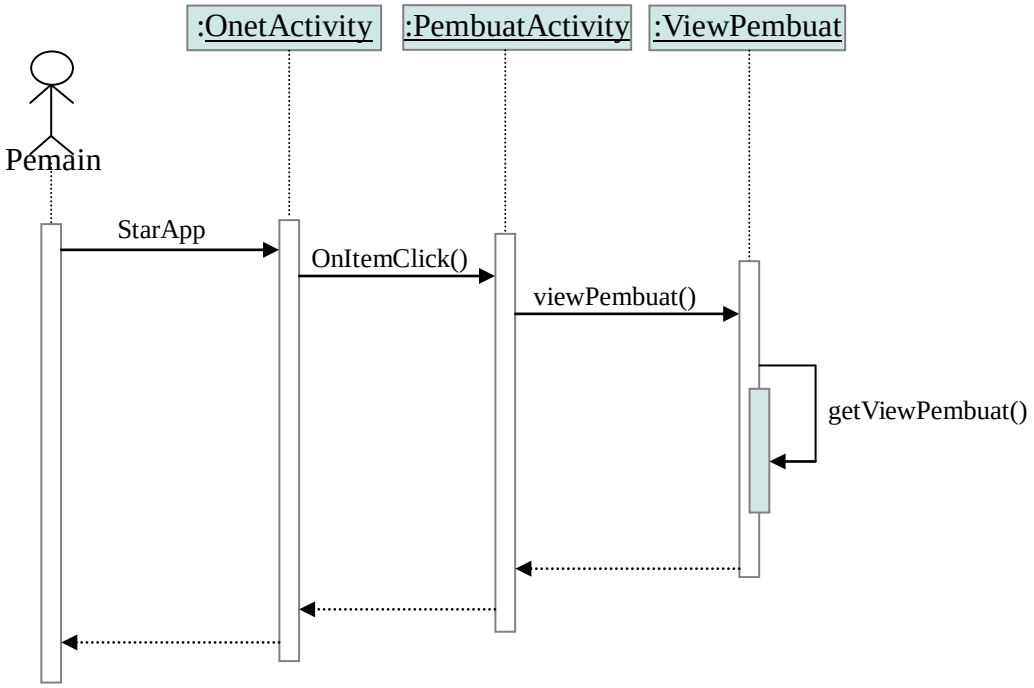
Gambar 3.13 *Sequence* Diagram Permainan



Gambar 3.14 *Sequence* Diagram *Score*

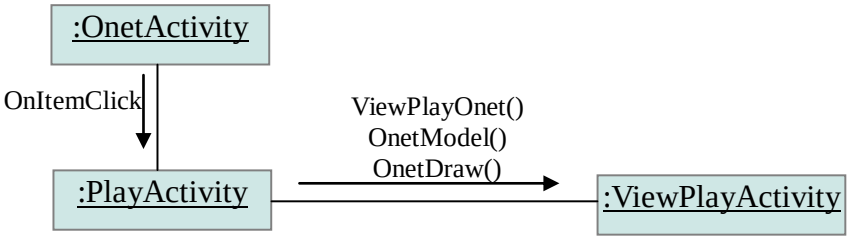


Gambar 3.15 Sequence Diagram Petunjuk

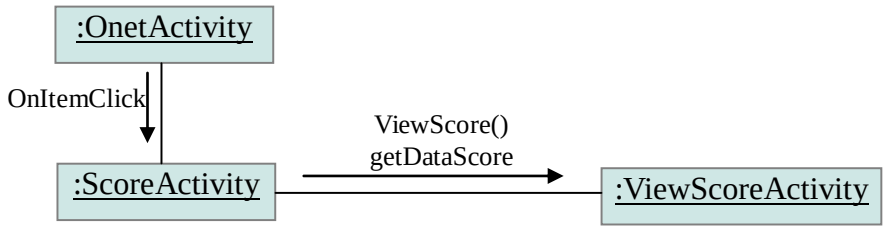


Gambar 3.16 Sequence Diagram Pembuat

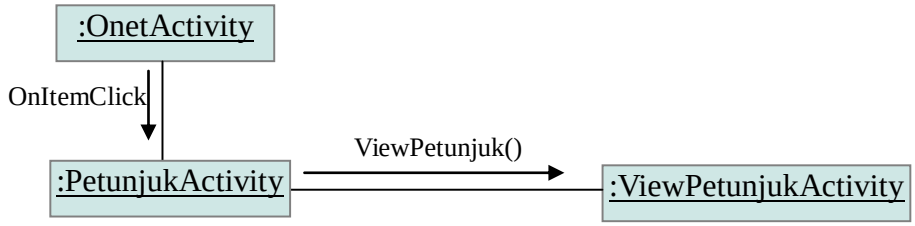
3.5.6 Collaboration Diagram



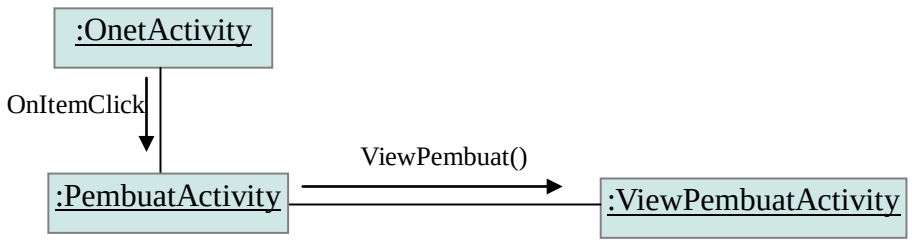
Gambar 3.17 *Collaboration Diagram* Permainan



Gambar 3.18 *Collaboration Diagram Score*

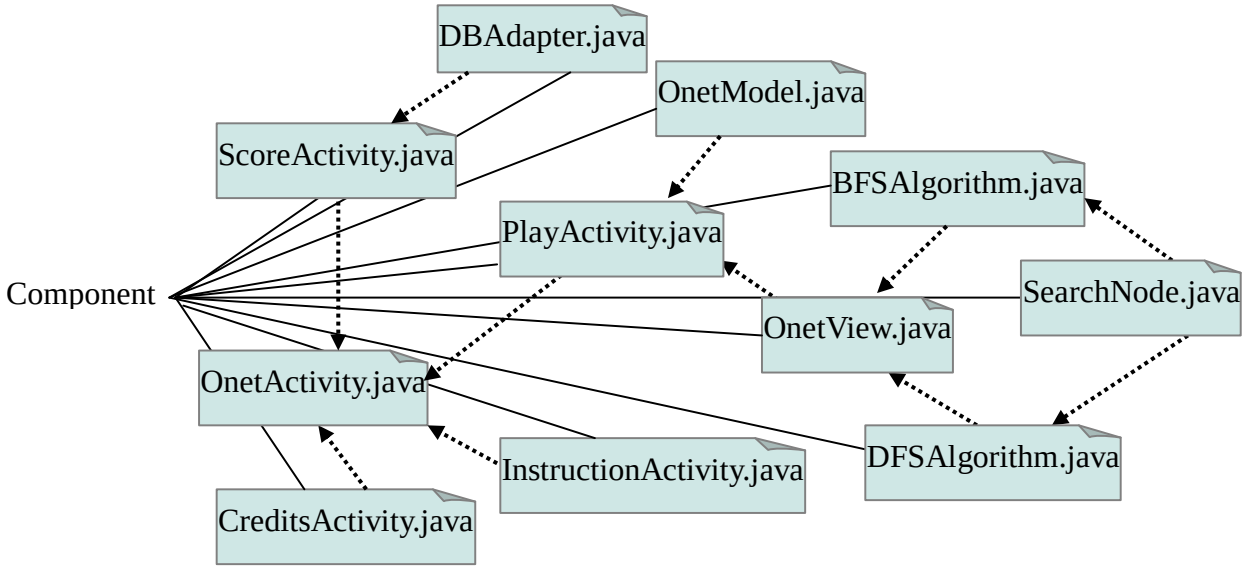


Gambar 3.19 *Collaboration Diagram* Petunjuk



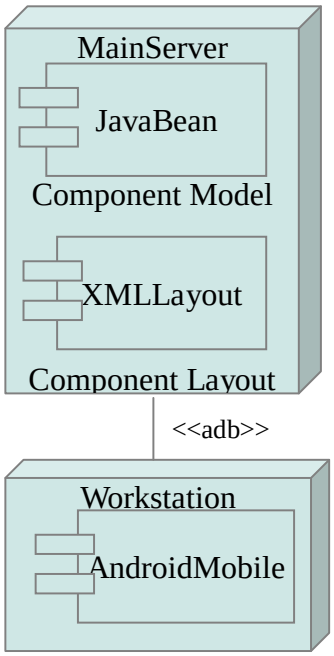
Gambar 3.20 *Collaboration Diagram* Pembuat

3.5.7 Component Diagram



Gambar 3.21 *Component Diagram – Source Code*

3.5.8 Deployment Diagram.



Gambar 3.22 *Deployment Diagram*

3.6 Perancangan Sistem

Perancangan sistem merupakan metodologi pengembangan suatu perangkat lunak yang dilakukan setelah melalui tahapan analisis.

3.6.1 Perancangan Komponen Permainan Onet *Fruit*

Perancangan komponen pada permainan onet terdiri dari pemain, titik awal dan titik tujuan yang akan dijelaskan sebagai berikut :

1. Pemain

Permainan ini akan dimainkan oleh satu orang. pemain dapat menempatkan titik awal dan, titik tujuan yang telah disediakan pada papan onet *fruit* sesuai dengan posisi yang diinginkan.

2. Titik awal dan titik tujuan

Titik awal merupakan titik pertama jalan yang akan dicari sedangkan titik tujuan merupakan titik akhir yang akan ditemukan.

3. Papan Onet *Fruit*

Papan yang digunakan pada permainan onet fruit ini akan berupa kotak-kotak dengan ordo X x Y . Terdiri dari kotak (grid) kecil dengan ordo minimal 6 x 10 dan maksimal 9 x 12. Papan ini merupakan tempat untuk memasangkan titik awal, dan titik tujuan serta untuk menampilkan jalan yang ditemukan. Hasil akhir *game* onet *fruit* ditandai oleh tampilnya jalur dan performansi yang dihasilkan

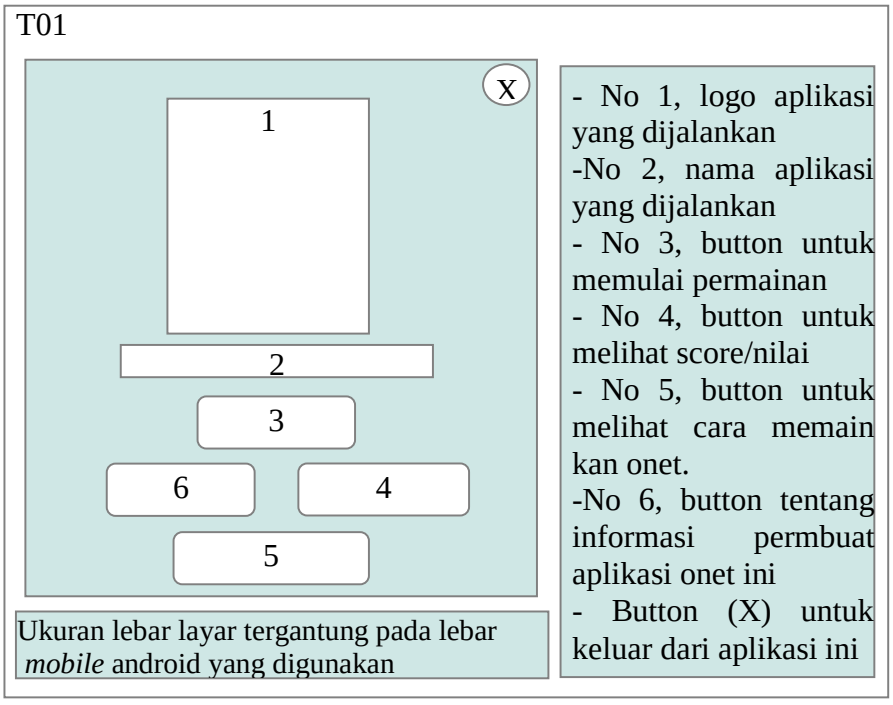
oleh pencarian BFS dan DFS.

3.6.2 Perancangan Antarmuka

Perancangan antarmuka bertujuan untuk memberikan gambaran tentang aplikasi yang akan dibangun, sehingga akan mempermudah dalam mengimple-mentasikan aplikasi serta akan memudahkan pembuatan aplikasi.

3.6.2.1 *Form* Pertama

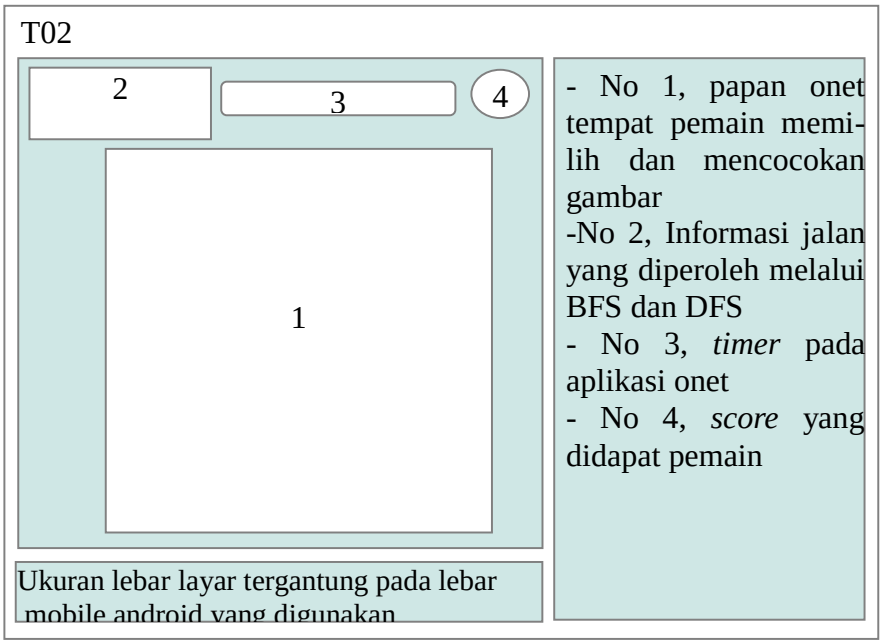
Form ini merupakan tampilan pertama saat aplikasi onet *fruit* dijalankan. Pada rancangan *form* ini terdapat tombol-tombol yang mempunyai fungsi tertentu yang berguna untuk menjalankan aplikasi. Rancangan *form* pertama dapat dilihat pada gambar berikut:



Gambar 3.23 Rancangan *Form* pertama

3.6.2.2 *Form* Permainan

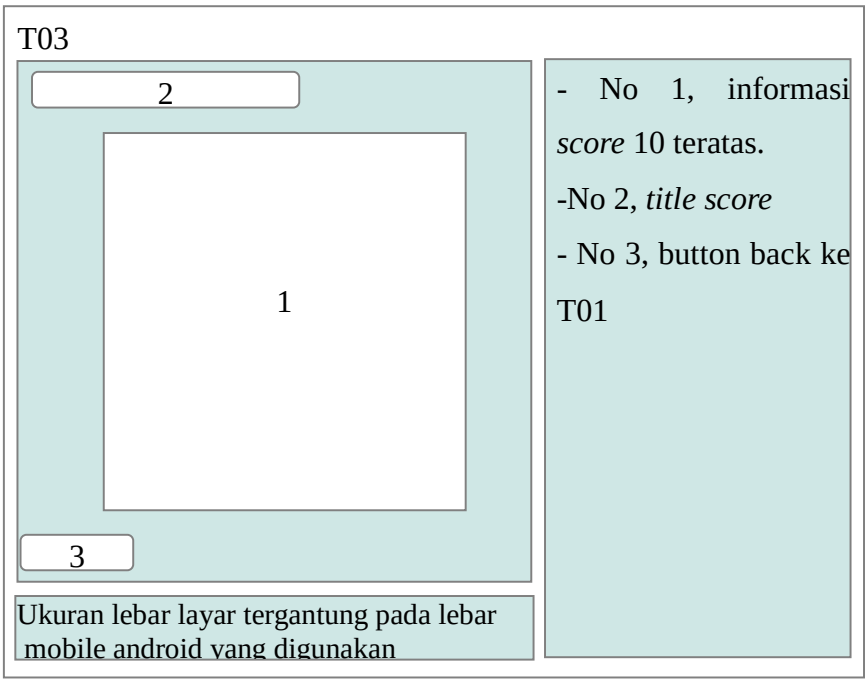
Form permainan merupakan rancangan form inti dari aplikasi onet *fruit*. Di *form* inilah pemain akan memainkan aplikasi onet *fruit*. Perancangan tampilan *form* ini dapat dilihat pada gambar berikut:



Gambar 3.24 Rancangan *Form* Permainan

3.6.2.3 Form Score

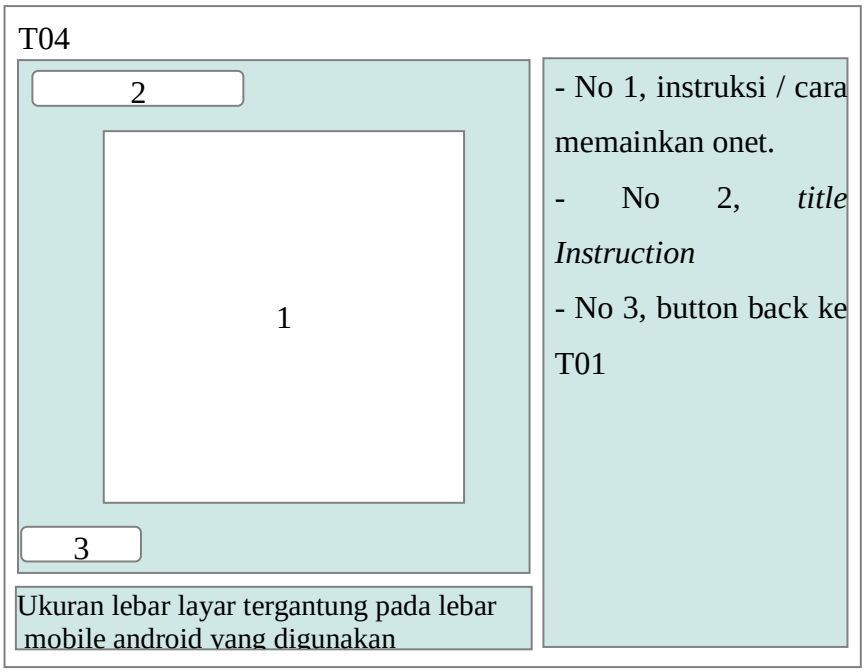
Form score merupakan informasi nilai yang didapat selama memainkan game onet *fruit* di *mobile*nya. Nilai yang ditampilkan hanya dibatasi 10 teratas saja. Perancangan *form score* dapat dilihat sebagai berikut:



Gambar 3.25 Rancangan *Form Score*

3.6.2.4 Form Instruction

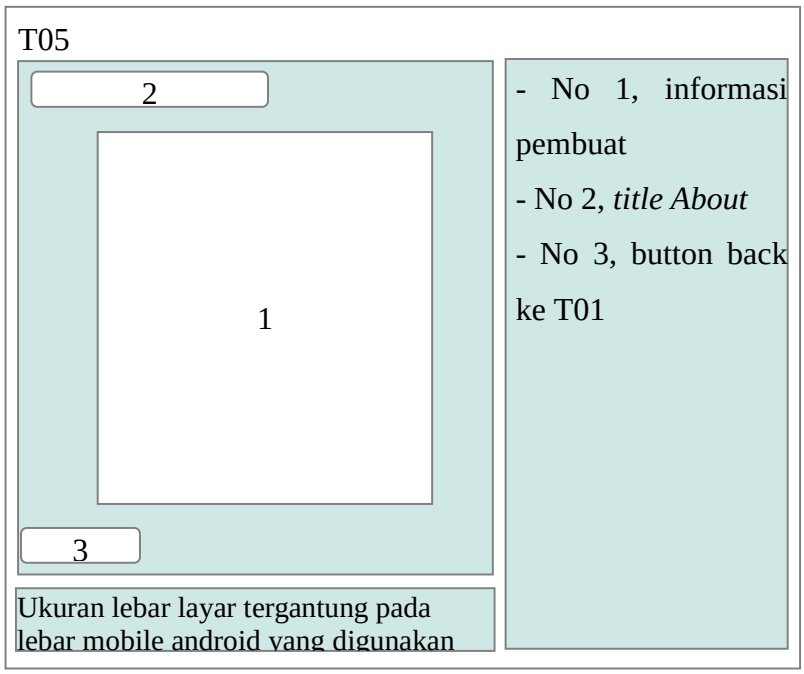
Form instruction merupakan informasi tentang bagaimana cara memainkan permainan onet *fruit*. Adapun rancangan *form* ini adalah sebagai berikut:



Gambar 3.26 Rancangan *Form Instruction*

3.6.2.5 Form About

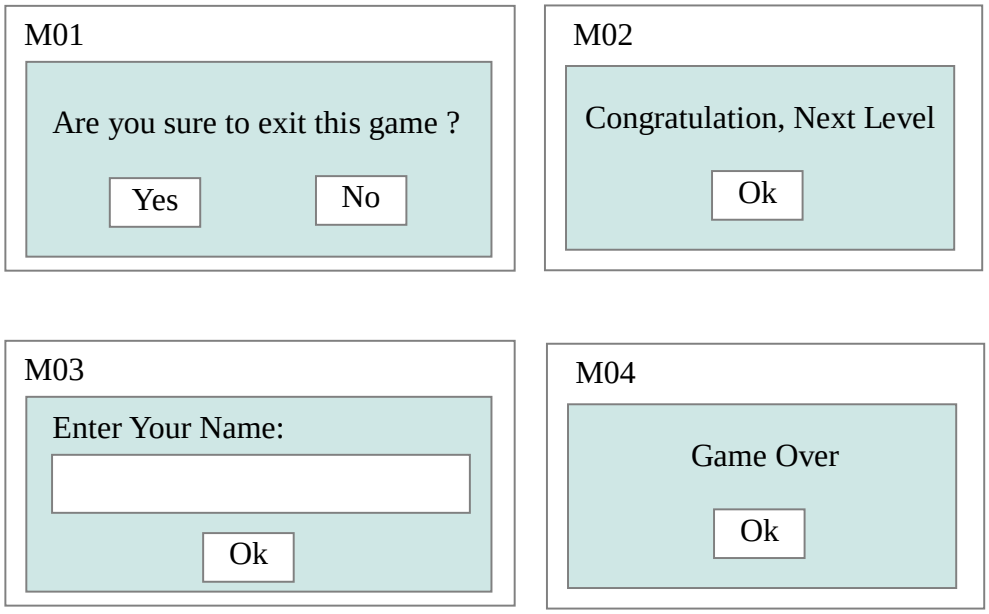
Form about merupakan informasi tentang pembuat. Adapun rancangan *form* ini adalah sebagai berikut:



Gambar 3.27 Rancangan *Form About*

3.6.2.6 Rancangan tampilan pesan

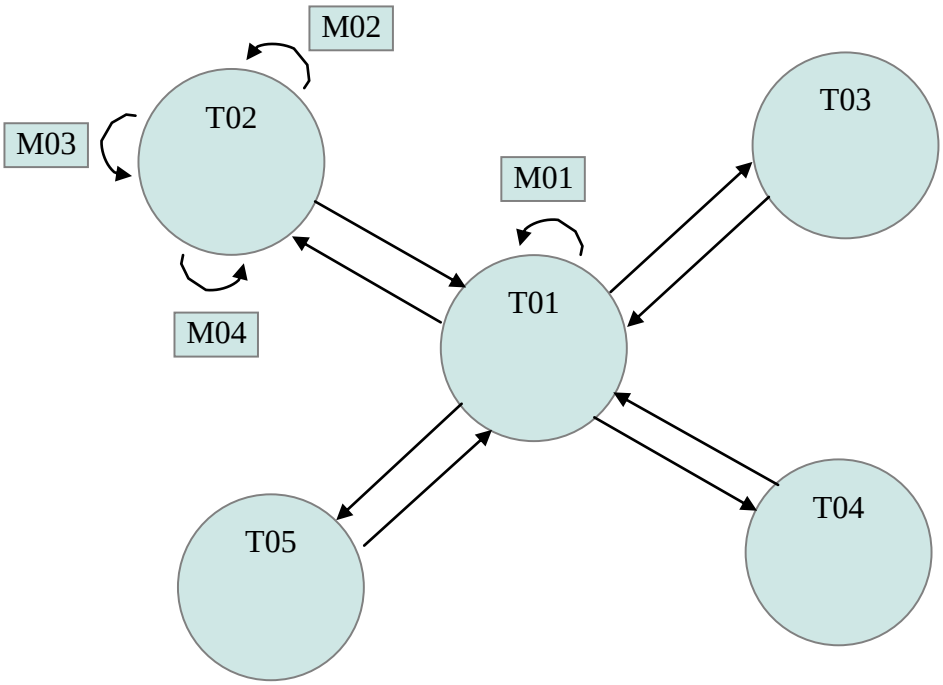
Perancangan Antar Muka ini adalah tampilan pesan yang muncul pada aplikasi ini seperti ditunjukan pada Gambar.



Gambar 3.28. Rancangan tampilan pesan

3.6.3 Jaringan Semantik

Perancangan jaringan semantik menggambarkan alur informasi dari setiap antarmuka aplikasi yang dijelaskan pada gambar berikut:



Gambar 3.29 Jaringan Semantik