BAB 3
ANALISIS DAN PERANCANGAN SISTEM

3.1 Analisis Sistem

Analisis Sistem adalah Penguraian dari suatu sistem informasi yang utuh ke dalam bagian - bagian komponennya dengan maksud untuk mengidentifikasi dan mengevaluasi permasalahan, kesempatan, hambatan yang terjadi dan kebutuhan yang diharapkan sehingga dapat diusulkan perbaikan [37].

Pada analisis sistem akan mencakup analisis masalah, analisis *game* sejenis, analisis *game* yang akan dikembangkan, analisis algoritma, analisis kebutuhan non-fungsional, analisis kebutuhan fungsional, serta perancangan sistem untuk membuat *game endemic Indonesia island*.

3.1.1 Analisis Masalah

Analisis masalah adalah suatu gambaran masalah yang diangkat dalam penulisan skripsi tentang latar belakang masalah yang diangkat untuk menjabarkan masalah yang terjadi atau alasan harus dibangunnya *game* ini.

Pada hutan pulau Jawa terus terancam *deforestasi*, berikut ini *deforestasi* yang terjadi di pulau Jawa [17].

1. Perambaan untuk ladang, pemukiman atau lokasi pabrik.
2. Kerusakan hutan akibat penebangan liar.

Sumatera adalah satu-satunya pulau di bumi ini dimana gajah, harimau, badak dan orangutan bisa ditemukan bersamaan. Kempat satwa liar ini kini di ambang bahaya akibat *deforestasi*. Berikut ini adalah akibat adanya *deforestasi* hutan pada pulau Sumatera[19].

A. Dalam penelitian antara tahun 1950an hingga 1960an, penyebab utama terjadinya *deforestasi* yaitu :

   1. Ekspansi pertanian untuk areal persawahan.
   2. Penebangan hutan skala kecil untuk ditanami kopi dan karet.

B. Di era 1970-an hingga 1990-an, penyebab utama terjadinya *deforestasi* yaitu :
   1. Operasi perusahaan kayu skala besar dan hutan tanaman industri menjadi faktor yang dominan.
   2. Program transmigrasi yang didorong pemerintah.
   3. Kebakaran hutan antara tahun 1982 hingga 1983 menjadi faktor sekunder.

C. Setelah era 90-an, penyebab utama terjadinya deforestasi yaitu :
   1. Perkebunan sawit dan *pulp* and *paper* menjadi ancaman utama *deforestasi*.
   2. Penebangan liar menjadi penyebab utama *degradasi* hutan.

Kalimantan merupakan daerah dengan kawasan hutan yang terbanyak. Faktor penyebab terjadinya *deforestasi* hutan di pulau Kalimantan adalah sebagai berikut[20]:

1. Pertumbuhan laju penduduk yaitu tingkat laju pertumbuhan penduduk di Kalimantan terus meningkat sehingga menyebabkan kerusakan hutan semakin meningkat.
2. Perluasan hutan yaitu perluasan hutan untuk pembukaan lahan pertanian dan perkebunan.
3. Perburuan dan pembunuhan satwa liar.

Pada pulau Sumatera, Jawa dan Kalimantan dari pemaparan masalah diatas ketiga pulau tersebut memiliki permasalahan yang sama pada kerusakan ekosistem hutan sehingga dampaknya mengancam kehidupan masyarakat serta kelestarian satwa endemik terancam punah. Untuk itu dengan pengenalan hewan endemik bisa memberikan pengetahuan tentang hewan endemik sehingga suatu saat bisa melestarikannya.

Dalam pembahasan penelitian *game* sejenis yaitu *shape memory game*, *gems swap dan pac-man* yang menjadi acuan dasar dalam pembuatan *game Endemic Indonesia Island*. Tujuan dari pembuatan *game* ini adalah untuk memperkenalkan jenis – jenis hewan endemik yang berada di pulau – pulau Indonesia terutama sebagai contoh pulau Sumatera, Jawa dan Kalimantan.

3.1.2 Analisis *Game* Sejenis

Pada tahap ini akan dilakukan analisis terhadap *game* yang sejenis, Analisis *game* sejenis dilakukan dengan metode observasi terhadap *game-game* sejenis.

1. ***Game "Shape Memory Game"***

   Berikut ini tampilan jendela dari permainan "*Shape Memory Game"* dapat dilihat pada gambar 3.1.



Gambar 3.1 Jendela Tampilan *Shape Memory Game*

Gambar 3.1 menampilkan jendela *game Shape Memory Game*. *Shape Memory Game* adalah sebuah *game flash online* yang dibuat oleh *thekidzpage.com* untuk dimainkan secara *online* [12].

A. Gameplay

Cara bermain *game* ini yaitu dengan mencari bentuk *shape* yang sama dengan bentuk pertama yang telah diklik oleh pemain hingga semuanya berpasangan sesuai dengan bentuk yang sama, setelah ditemukan yang

sama maka gambar itupun menghilang dari *game* sehingga menyisakan gambar yang belum berpasangan. Bentuk *game* ini adalah permainan *match 2 pair game* yang dibuat secara sederhana untuk dimainkan oleh anak-anak maupun dewasa.

2. ***Game "Gems Swap"***

Berikut ini tampilan jendela dari permainan "*Gems Swap"* dapat dilihat pada gambar 3.2.



Gambar 3.2 Jendela Tampilan *Gems Swap*

Gambar 3.2 menampilkan jendela *game Gems Swap*. *Gems Swap* adalah *game flash online* yang dibuat oleh *www.mochigames.com* untuk dimainkan secara *online*. *Gems Swap* adalah sebuah permainan yang mengasah logika, ketelitian, skill taktik dan strategi untuk penggunanya yang bisa dimainkan oleh anak-anak maupun dewasa [13].

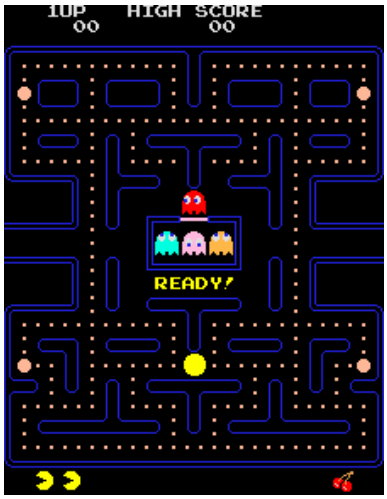
A. ***Gameplay***

Didalam Permainan *Gems Swap* ada 6 buah permata utama dengan 6 bentuk dan warna yang berbeda. Didalam *Gems Swap* pemain harus berusaha mensejajarkan permata-permata sejenis untuk mendapatkan poin. Permainan bisa dilakukan dengan menukar secara vertikal maupun

horizontal satu permata, untuk membuat sebuah baris atau kolom berisi minimal tiga, untuk mendapatkan poin. Permainan ini berjenis *Match three pair games.*

3. ***Game* "*Pac-Man*"**

Berikut ini tampilan jendela dari permainan "*Pac-Man"* dapat dilihat pada gambar 3.3.



Gambar 3.3 Jendela Tampilan *Game Pac-Man*

Gambar 3.3 menampilkan jendela *game Pac-Man*. *Pac-Man* adalah *game arcade* yang dikembangkan *Namco* dan diterbitkan *Midway*, dirilis di Jepang pada 22 Mei 1980. Awalnya dirilis di *arcade* game saja, *Pac-Man* yang masih populer hingga kini telah dirilis pula dalam *platform* lainnya seperti *Game Boy* dan SNES. Perancang permainan ini adalah *Toru Iwatani*, yang merupakan karyawan *Namco*[14].

A. G*ameplay*

Pemain harus mengontrol tokoh berwarna kuning bernama *Pac-Man* dan membawanya mengelilingi lorong berlika-liku sambil "memakan" titik-titik kecil dan benda-benda khusus lainnya. Pada saat yang sama, terdapat empat "hantu" yang berkeliling di lorong tersebut yang bertugas menangkap *Pac-Man*. Sang pemain dapat menyelesaikan satu *level* (tingkat) jika berhasil memakan seluruh titik dan benda khusus.

Berikut ini perbandingan antara *game shape memory game, gems swap, pac-man* dan *game* yang akan dibangun akan dijelaskan pada tabel 3.1:

Tabel 3.1 Perbandingan *Game* Sejenis

No	Perbandingan	*Shape Memory Game*	*Pac-Man*	*Gems Swap*	*Endemic Indonesia Island*
1.	Episode	Tidak menggunakan episode.	Tidak menggunakan episode.	Tidak menggunakan episode.	3 episode.
2.	*Level*	Tidak ada tingkatan *level.*	Ada tingkatan *level.*	Ada tingkatan *level.*	Ada tingkatan *level.*

3.1.3 Analisis Game Endemic Indonesia Island

Analisi *game Endemic Indonesia Island* merupakan cakupan analisis dari *game* yang akan dibangun, diantaranya adalah sebagai berikut :

3.1.3.1 Deskripsi *Game Endemic Indonesia Island*

Game yang dibangun ini adalah *game single player* sehingga permainan berdiri sendiri yang berbentuk 2 dimensi berbasis *desktop* dengan *genre casual game. Game* ini berkonsep *puzzle match pair game* sebagai bentuk dasar gamenya dengan tujuan untuk memperkenalkan jenis – jenis hewan endemik yang ada di beberapa pulau Indonesia yang diharapkan dapat melatih logika, ketelitian, skill taktik dan strategi.

Beberapa fitur yang ditawarkan dalam *game* ini :

1. *Single Player.*
2. *Puzzle memory* episode Sumatera
3. *Match three pair game* pada episode Jawa.

4. *Maze game* pada episode Kalimantan.
5. *Game* dibuat dengan grafis 2D.
6. Karakter mengambil dari hewan endemik Indonesia.
7. Hewan endemik terdiri dari tiga pulau yang ada di Indonesia yaitu Sumatera, Jawa dan Kalimantan.

3.1.3.4 Story Line Game Endemic Indonesia Island

Story line merupakan jalan cerita sebuah *game* yang akan dibangun sesuai dengan topik atau judul yang dibuat oleh *developer game* [24].

Endemic Indonesia Island adalah *game* yang memperkenalkan jenis dan bentuk hewan endemik Indonesia dengan menggunakan gambar hewan endemik yang berupa *game puzzle* yaitu *match pair game*. *Game* ini terdiri dari 3 episode *game puzzle* berdasarkan hewan endemik yang hidup di pulau yang ada di Indonesia yaitu Sumatera, Jawa dan Kalimantan.

3.1.3.3 Pelevelan

Dalam *game* ini level yang diberikan yaitu dengan episode karena *game* ini berdasarkan episode hewan endemik pada pulau yang ada di Indonesia yang disesuaikan dengan jumlah hewan endemiknya pada setiap pulau. Setiap episode berbeda bentuk *gameplay* yang akan dimainkan. Berikut ini adalah *gameplay* yang akan dimainkan berdasarkan episodenya :

1. Episode *Game* Sumatera.

   Game berbentuk *puzzle memory game* yang akan dimainkan oleh *player* dengan jumlah 8 pasang hewan endemik yang harus di pasangkan dengan bentuk yang samanya.
2. Episode *Game* Jawa.

   Game berbentuk *match three pair game* yang akan dimainkan oleh *player* dengan jumlah 6 hewan endemik yang harus di pasangkan dengan bentuk yang sama dengan minimal 3 bentuk yang sama.

3. Episode *Game* Kalimantan.

   Game berbentuk *maze game* yang akan dimainkan oleh *player* dengan jumlah 1 pemburu sebagai *enemy* atau musuh dan 1 hewan endemik sebagai *player*, *player* harus menghindar dari kejaran pemburu agar tidak tertangkap.

3.1.3.4 *Gameplay Game Endemic Indonesia Island*

Gameplay adalah aturan detil yang mengatur bagaimana seorang pemain memainkan sebuah permainan, berinteraksi dengan permainan, dan mencapai kemenangan atau mengakhiri sebuah permainan [24].

Pada *game endemic Indonesia Island* pemain harus menyelesaikan setiap *Episode* permainan dalam peta Indonesia .

Berikut adalah *gameplay* pada *game Endemic Indonesia Island*.

1. Episode Sumatera

   Pada episode ini pemain harus mencocokan dua jenis gambar yang sama yaitu gambar hewan endemik pulau Sumatera. Jumlah hewan endemik yang harus dipasangkan adalah 8 pasang gambar hewan endemik dan pemain harus menyelesaikan tepat sebelum waktunya yaitu 30 detik untuk menyelesaikan permainan ini.

2. Episode Jawa

   Pada episode ini pemain harus berusaha mensejajarkan gambar hewan – hewan endemik sejenis untuk mendapatkan poin atau *score*. Permainan bisa dilakukan dengan menukar secara vertikal maupun *horisontal* satu gambar hewan yang ingin disamakan dan yang berbeda, untuk membuat sebuah baris atau kolom berisi minimal tiga, untuk mendapatkan poin. Permainan ini berjenis *match three pair games* atau mencocokan minimal 3 gambar yang sama baik secara *vertical* maupun *horizontal*, pemain diberikan waktu 60 detik untuk menyelesaikan *game* ini.

3. Episode Kalimantan

   Pada episode ini pemain harus mengontrol tokoh “hewan endemik” dan membawanya mengelilingi lorong berlika-liku sambil memakan koin-koin. Pada saat yang sama, terdapat satu “pemburu” yang berkeliling di

lorong tersebut yang bertugas menangkap"hewan endemik". Pemain dapat menyelesaikan satu *episode* ini jika berhasil memakan seluruh titik dan tidak tertangkap oleh para "pemburu" juga menyelesaikan tepat pada waktunya sebelum 60 detik. Permainan ini berjenis *maze game*.

3.0.3.5 *Scoring Game Endemic Indonesia Island*

Scoring adalah evaluasi kinerja dengan menetapkan *grade* atau nilai yang akan didapatkan dalam memainkan permainan ini [22].

Pada *game endemic Indonesia Island* pemain harus menyelesaikan setiap episode permainan dalam episode pulau yang ada di Indonesia .

Berikut adalah *scoring* pada *game Endemic Indonesia Island*.

1. Episode Sumatera

Pada episode ini pemain harus mencocokan dua jenis gambar yang sama yaitu gambar hewan endemik pulau Sumatera. Pemain akan mendapatkan skor 100 poin untuk setiap gambar yang telah dicocokan, dan diberikan waktu menyelesaikan permainan dalam waktu 30 detik. Jika waktu habis maka pemain mendapatkan skor kecil dan jika menyelesaikannya tepat sebelum waktunya habis pemain akan mendapatkan nilai tinggi.

2. Episode Jawa

Pada episode ini pemain harus berusaha mensejajarkan gambar hewan – hewan endemik sejenis untuk mendapatkan poin atau *score*. Permainan bisa dilakukan dengan menukar secara vertikal maupun *horisontal* satu gambar hewan yang ingin disamakan dan yang berbeda, untuk membuat sebuah baris atau kolom berisi minimal tiga, untuk mendapatkan poin. Permainan ini berjenis *match three pair games* atau mencocokan minimal 3 gambar yang sama baik secara vertical maupun horisontal. pemain akan mendapatkan skor 100 poin setiap mencocokan yang benar. Permainan akan berakhir ketika 60 detik, waktu ini diberikan untuk pemain memainkan permainan ini.

3. Episode Kalimantan

Pada episode ini pemain harus mengontrol tokoh “hewan endemik” dan membawanya mengelilingi lorong berlika-liku sambil memakan titik-titik kecil. Pada saat yang sama, terdapat satu “pemburu” yang berkeliling di lorong tersebut yang bertugas menangkap”hewan endemik”. Pemain dapat menyelesaikan satu *episode* ini, jika berhasil mendapatkan seluruh koin – koin tersebut dan tidak tertangkap oleh para “pemburu”. Koin – koin ini bernilai 100 poin. Jika hewan endemik berhasil tertangkap oleh pemburu maka permainan akan dimulai dari awal kembali dan kesempatan/nyawa pemain akan berkurang 1, kesempatan/nyawa diberikan 3 kali untuk menyelesaikan permainan ini, jika tidak selesai dan kesempatan habis maka permainan berakhir. Dalam memainkan permainan ini pemain diberikan waktu 60 detik dalam memainkannya.

3.2 Analisis Algoritma

Analisis algoritma merupakan analisa urutan langkah yang tepat dan pasti dalam memecahkan suatu masalah secara logis yang akan diterapkan sehingga memberikan solusi terbaik dalam pemecahannya.

3.2.1 Analisis Algoritma *Memory Game* Pada Episode *Game* Sumatera

Berikut ini adalah langkah pembuatan logika algoritma dari *memory game* :

Langkah 1 :
Inisialisasi kartu dan buat array. Untuk setiap i dalam range [0 ... n-1],
set Kartu [i] = k.
Langkah 2 (opsional) :
Buat random number generator atau rumus matematika penghitungan angka random dari array.
Langkah 3: Set **i = 0**.
Langkah 4: Set **i** dan **j** = random_number * n.
Langkah 5: Tukar nilai kartu [i] dan kartu [j].
Langkah 6: Set **i**= **i** + 1 Jika **i** kurang dari n, lanjutkan ke langkah 4.

Berikut ini adalah contoh penerapan algoritma dalam *ActionScript 3.0.*

```
Acak Kartu();

Kartu = new Array("kartu1","kartu1",...,"kartu8","kartu8");

for (var i=0; i>n; i++)
{
 var i = Math.floor(Math.random()*n);

 var j = Math.floor(Math.random()*n);

 var k = Kartu[i];

 Kartu[i] = Kartu[j];

 Kartu[j] = k;
}
```

Gambar 3.3 Logika Algoritma *Memory Game*

3.2.2 Analisis Algoritma *Match Three Pair Game* Pada Episode *Game* Jawa

Berikut ini adalah langkah pembuatan algoritma dari *match three pair game* pada *ActionScript 3.0* :

1. Membuat *random board*

   Pada langkah ini, dibuat sebuah matrik *board* 8 x 8 dengan susunan acak dari tujuh item yang berbeda dibuat untuk memulai permainan[38].

```
public function MatchThree() {
    // membuat grid array
    grid = new Array();
    for(var gridrows:int=0;gridrows<8;gridrows++) {
    grid.push(new Array());
                        }
    setUpGrid();
    isDropping = false;
    isSwapping = false;
    gameScore = 0;
    addEventListener(Event.ENTER_FRAME,movePieces);
}
```

Gambar Error! No text of specified style in document.**.4 Algoritma Matrik 8x8 *random board***

2. Memeriksa kesamaan atau kecocokan *(matching)*

   Ada beberapa pembatasan awal pada papan (*board*). Pembatasan pertama adalah bahwa tidak ada tiga potongan dalam baris yang sudah dicocokan oleh papan (*board*). Ini berarti pemain harus menemukan pencocokan pertama sendiri [38].

```
public function setUpGrid() {
 // loop sampai awal grid valid
while (true) {
 // Membuat sprite
    gameSprite = new Sprite();

 // tambahkan 64 potongan acak
    for(var col:int=0;col<8;col++)
    for(var row:int=0;row<8;row++) {

       addPiece(col,row);
    }
      }
 // Coba lagi jika terdapat kecocokan
    if (lookForMatches().length != 0) continue;

 // Coba lagi jika tidak ada kemungkinan kecocokan

    if (lookForPossibles() == false) continue;
 // Tidak ada kecocokan, tapi kemungkinan ada : ketika board atau kecocokan ditemukan benar
    break;
}
                        // tambahkan sprite
                        addChild(gameSprite);
}
```

Gambar Error! No text of specified style in document..5 Algoritma Mengecek Kecocokan *Piece.*

3. Memeriksa langkah berikutnya (*moves)*

   Pembatasan kedua adalah bahwa harus ada setidaknya satu langkah yang valid. Itu berarti pemain harus mampu untuk menukar (*swap)* dua potong dan membuat pencocokan [38].

```
// pemain mengklik pada piece
		public function clickPiece(event:MouseEvent) {
			var piece:Piece = Piece(event.currentTarget);

			// Potongan Pertama yang dipilih
			if (firstPiece == null) {
				piece.select.visible = true;
				firstPiece = piece;

			// Mengklik potongan pertama kembali
			} else if (firstPiece == piece) {
				piece.select.visible = false;
				firstPiece = null;

			// menklik potongan yang kedua
			} else {
				firstPiece.select.visible = false;

				// Baris yang sama, satu kolom selesai
				if ((firstPiece.row == piece.row) && (Math.abs(firstPiece.col-piece.col) == 1)) {
					makeSwap(firstPiece,piece);
					firstPiece = null;

				// Kolom yang sama, satu baris selesai
				} else if ((firstPiece.col == piece.col) && (Math.abs(firstPiece.row-piece.row) == 1)) {
					makeSwap(firstPiece,piece);
					firstPiece = null;

				// Kesalahan move, menempatkan kembali piece atau potongan
				} else {
					firstPiece = piece;
					firstPiece.select.visible = true;
				}
			}
		}
```

Gambar 3.6 Algoritma Menukar dan Mencocokan

4. Pemain memilih dua potongan

   Pada langkah ini, potongan-potongan harus berdekatan satu sama lain secara horizontal atau vertikal, dan *swap* harus menghasilkan kecocokan [38].

```
// swap 2 potongan
		public function swapPieces(piece1,piece2:Piece) {
			// swap nilai baris dan kolom
			var tempCol:uint = piece1.col;
			var tempRow:uint = piece1.row;
			piece1.col = piece2.col;
			piece1.row = piece2.row;
			piece2.col = tempCol;
			piece2.row = tempRow;

			// swap posisi grid
			grid[piece1.col][piece1.row] = piece1;
			grid[piece2.col][piece2.row] = piece2;

		}
```

Gambar 3.7 Algoritma Memilih Dua Potongan

5. Memeriksa apakah potongan sudah tertukar (*swapped*)

   Pada langkah ini , papan (*board*) menampilkan animasi yang menunjukkan dua potong pindah ke tempat satu sama lain atau saling bertukar [38].

```
// start animasi swap 2 potongan
                public function makeSwap(piece1,piece2:Piece) {
                        swapPieces(piece1,piece2);

                        // mengecek untuk melihat apakah langkah itu berbuah
                        if (lookForMatches().length == 0) {
                                swapPieces(piece1,piece2);
                        } else {
                                isSwapping = true;
                        }
                }
```

Gambar 3.8 Algoritma Mengecek *Swap*

6. Mencari potongan yang sama atau cocok

   Pada langkah ini, setelah penukaran (*swap)* dilakukan, papan (*board*) harus mencari kecocokan baru tiga berturut-turut atau lebih. Jika tidak ada yang cocok, swap perlu berbalik kembali [38].

```
//return array ketika semua kecocokan ditemukan
                public function lookForMatches():Array {
                        var matchList:Array = new Array();
                        // Pencarian untuk kecocokan secara horisontal
                        for (var row:int=0;row<8;row++) {
                                for(var col:int=0;col<6;col++) {
                                        var match:Array = getMatchHoriz(col,row);
                                        if (match.length > 2) {
                                                matchList.push(match);
                                                col += match.length-1;
                                        }
                                }
                        }
                        // Pencarian untuk kecocokan secara vertikal
                        for(col=0;col<8;col++) {
                                for (row=0;row<6;row++) {
                                        match = getMatchVert(col,row);
                                        if (match.length > 2) {
                                                matchList.push(match);
                                                row += match.length-1;
                                        }

                                }
                        }
                        return matchList;
                }
```

Gambar 3.9 Algoritma Mencari Potongan Yang Sama

7. Penambahan *point* atau *score*

   Pada langkah ini, jika kecocokan ditemukan, poin atau skor dapat diberikan [38].

```
for(var j:int=0;j<matches[i].length;j++) {
                                        if (gameSprite.contains(matches[i][j])) {
                                                var pb = new PointBurst(this,numPoints,matches[i][j].x,matches[i][j].y);
                                                addScore(numPoints);
                                                gameSprite.removeChild(matches[i][j]);
                                                grid[matches[i][j].col][matches[i][j].row] = null;
                                                affectAbove(matches[i][j]);
                                        }
                                }
```

Gambar 3.10 Algoritma Menambahkan *Point* atau *Score*

8. Menghapus potongan yang sudah cocok

   Pada langkah ini potongan-potongan yang terlibat dalam pencocokan atau sudah sesuai, harus dihapus dari papan (*board*) [38].

```
// mendapat kecocokan dan menghapusnya, maka poin ditambahkan
                public function findAndRemoveMatches() {
                        // Mendapatkan list kecocokan
                        var matches:Array = lookForMatches();
                        for(var i:int=0;i<matches.length;i++) {
                                var numPoints:Number = (matches[i].length-1)*50;
                                for(var j:int=0;j<matches[i].length;j++) {
                                        if (gameSprite.contains(matches[i][j])) {
                                                var pb = new PointBurst(this,numPoints,matches[i][j].x,matches[i][j].y);
                                                addScore(numPoints);
                                                gameSprite.removeChild(matches[i][j]);
                                                grid[matches[i][j].col][matches[i][j].row] = null;
                                                affectAbove(matches[i][j]);
                                        }
                                }
                        }
```

Gambar 3.11 Algoritma Menghapus Potongan

9. Men-*drop down* potongan

   Potongan-potongan dari atas harus drop-down untuk mengisi ruang yang sudah terhapus [38].

```
// Jika ada piece tidak ada pada kolom, tambahkan 1 dan drop piece itu
public function addNewPieces() {
        for(var col:int=0;col<8;col++) {
                var missingPieces:int = 0;
                for(var row:int=7;row>=0;row--) {
                        if (grid[col][row] == null) {
                                var newPiece:Piece = addPiece(col,row);
                                newPiece.y = offsetY-spacing-spacing*missingPieces++;
                                isDropping = true;
                        }
                }
        }
}
```

Gambar 3.12 Algoritma *Men-Drop Down* Potongan

10. Menambahkan potongan baru (*add new*)

    Pada langkah ini, potongan baru perlu *drop down* dari atas papan (*board*) untuk mengisi ruang kosong [38].

```
// Membuat random piece, tambahkan  ke sprite dan grid
                public function addPiece(col,row:int):Piece {
                        var newPiece:Piece = new Piece();
                        newPiece.x = col*spacing+offsetX;
                        newPiece.y = row*spacing+offsetY;
                        newPiece.col = col;
                        newPiece.row = row;
                        newPiece.type = Math.ceil(Math.random()*6);
                        newPiece.gotoAndStop(newPiece.type);
                        newPiece.select.visible = false;
                        gameSprite.addChild(newPiece);
                        grid[col][row] = newPiece;
                        newPiece.addEventListener(MouseEvent.CLICK,clickPiece);
                        return newPiece;
                }
```

Gambar 3.13 Algoritma Menambahkan Potongan Baru

11. Mencari potongan yang cocok lagi

    Pada langkah ini, setelah semua potongan telah tertukar dan potongan yang baru telah mengisi celah kekosongan potongan yang cocok, maka pencarian lain untuk pencocokan diperlukan. Kembali ke langkah 6 [38].

12. Memeriksa apakah potongan tidak ada yang cocok lagi

    Pada langkah ini, sebelum memberikan kontrol kembali ke pemain, pengecekan dilakukan untuk melihat apakah ada langkah yang mungkin untuk di lakukan. Jika tidak, permainan berakhir [38].

```
// melihat kembali jika langkah kemungkinan ada di board
		public function lookForPossibles() {
			for(var col:int=0;col<8;col++) {
				for(var  row:int=0;row<8;row++) {

					// horizontal possible, 2+1
					if (matchPattern(col, row, [[1,0]], [[-2,0],[-1,-1],[-1,1],[2,-1],[2,1],[3,0]])) {
						return true;
					}
					// horizontal possible, middle
					if (matchPattern(col, row, [[2,0]], [[1,-1],[1,1]])) {
						return true;
					}
					// vertical possible, 2+1
					if (matchPattern(col, row, [[0,1]], [[0,-2],[-1,-1],[1,-1],[-1,2],[1,2],[0,3]])) {
						return true;
					}
					// vertical possible, middle
					if (matchPattern(col, row, [[0,2]], [[-1,1],[1,1]])) {
						return true;
					}
				}
			}
			// Kemungkinan untuk melangkah tidak ditemukan
			return false;
		}
```

Gambar 3.14 Algoritma Pengecekan Tidak Cocok Lagi.

3.2.3 Analisis Algoritma A* Pada Episode Game Kalimantan

Algoritma A* (*A Star*) di dalam game *Endemic Indonesia Island* akan diterapkan pada NPC (*non-playable character)* untuk pencarian aktor . Cara kerja algoritma A* di dalam game *Endemic Indonesia Island* lebih jelasnya adalah sebagai berikut :

1. Analisis Penggunaan Algoritma A *

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

	0	1	2
0	(0,0)	(1,0)	(2,0)
1	(0,1)	(1,1)	(2,1)
2	(0,3)	(1,2)	(2,2)

Gambar 3.15 *Tile Map* Ordo X x Y.

Algoritma A* akan diterapkan pada objek hewan endemik dalam melakukan pencarian karakter pemain dalam arena permainan. Adapun istilah-istilah yang akan dibahas yaitu *open list*, *Closed list*, nilai f,g dan h [29].

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

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

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

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

E. Setiap simpul harus memiliki informasi h(n), yaitu hasil estimasi harga simpul tersebut dihitung dari simpul tujuan yang hasilnya menjadi nilai H. Berikut ini adalah bentuk perhitungan pencarian nilai H dalam *Manhattan Distance*.

```
H = Math.abs(start.x-destination.x) + Math.abs(start.y-destination.y));
```

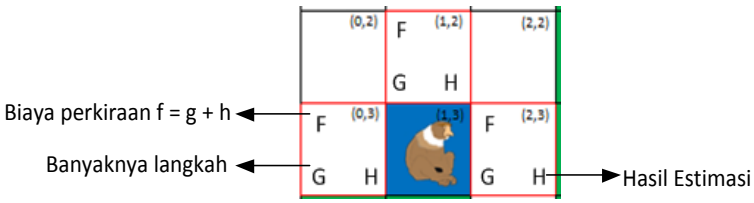
F. Langkah – langkah deskripsi algoritma A*[35].

1. Tambahkan node awal ke dalam *open list*.
2. Ulangi langkah berikut:

a. Cari *node* F terendah pada *open list*. Ini adalah node saat ini/*current node*.
b. Jika *node* saat ini adalah *node* akhir, Anda sudah selesai. Lanjutkan ke langkah 4.
c. Periksa setiap *node* sekitarnya (dalam kotak persegi panjang, akan ada delapan *node*). untuk masing-masing simpul sekitarnya:
    i. Jika tidak *walkable* atau jika termasuk pada *closed list*, abaikan saja dan lanjutkan dengan simpul sekitarnya berikutnya.
    ii. Hitung biaya.
    iii. Mengatur node saat ini sebagai induknya (*parent*).
    iv. Menambahkannya ke *open list*.
d. Stop ketika anda :
    i. Tambahkan node saat ini ke *closed list*. Dalam hal ini jalan atau *path* telah ditemukan.
    ii. Gagal untuk menemukan target persegi, dan *open list* kosong. Dalam hal ini, tidak ada jalan.

3. Ulangi langkah 2 dengan *open list* diperbarui.
4. Anda telah menemukan simpul akhir. Buat *path list* dan menambahkan *node* akhir untuk itu.
5. Tambahkan induk (*parent*) dari node akhir ke *path list*.
6. Tambahkan induk (*parent*) dari simpul tersebut ke *path list*. Ulangi sampai Anda mencapai n*ode* awal. *path list* sekarang memegang daftar *node* yang membentuk jalur terbaik dari awal sampai akhir.

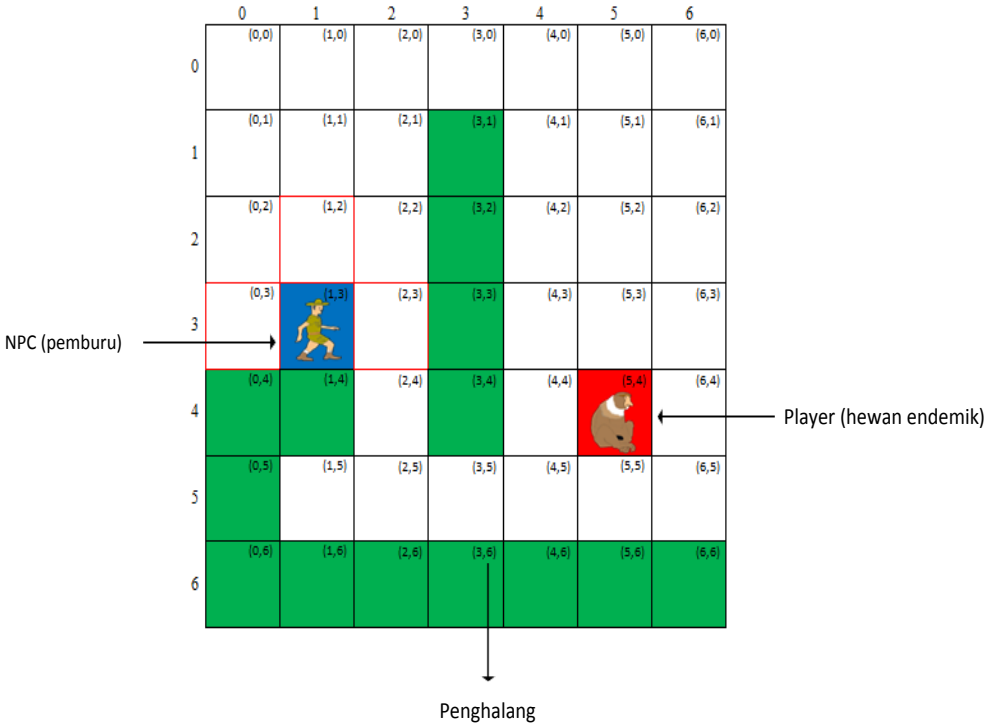
G. Berikut ini adalah ilustrasi bentuk perhitungan pencarian jalur A* dapat dilihat pada gambar 3.16.



Gambar 3.16 Ilustrasi Perhitungan Algoritma A*(*Star*)

1. Ilustrasi Pencarian Jalur Algoritma A* dengan *manhattan distance.*

   Manhattan distance adalah pencarian heuristik A* yang paling sederhana karena mengabaikan setiap gerakan diagonal. Heuristik ini hanya menambah jumlah baris dan kolom antara dua *node* yang disediakan untuk mencapai *node* tujuan. Berikut ini adalah kondisi awal pada pencarian jalur A* antara NPC dan *Player* dapat dilihat pada gambar 3.17.



Gambar 3.17 Kondisi Awal NPC dan *Player.*

Berikut ini adalah langkah – langkah pencarian jalur algoritma A*.

1. Langkah Pertama

   Masukkan posisi awal pemburu/NPC (1,3) dan posisi pemain/*player* (5,4). Kemudian cari titik koordinat terdekat berikutnya yang bisa di pakai sebagai *path* untuk menemukan *player*.

   a. Pada Koordinat n (2,3)

      Diketahui : Nilai nx = 2 dan Nilai ny = 3

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

      Perhitungan :

      g(2,3) = 1

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

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

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

      h(2,3) = 3+1

      h(2,3) = 4

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

      f(2,3) = 1+4

      f(2,3) = 5

   b. Pada Koordinat (1,2)

      Diketahui : Nilai nx = 1 dan Nilai ny = 2

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

      Perhitungan :

      g(1,2) = 1

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

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

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

      h(1,2) = 4+2

      h(1,2) = 6

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

      f(1,2) = 1+6

      f(1,2) = 7

c. Pada Koordinat (0,3)

Diketahui : Nilai nx = 0 dan Nilai ny = 3

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

Perhitungan :

g(0,3)=1

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

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

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

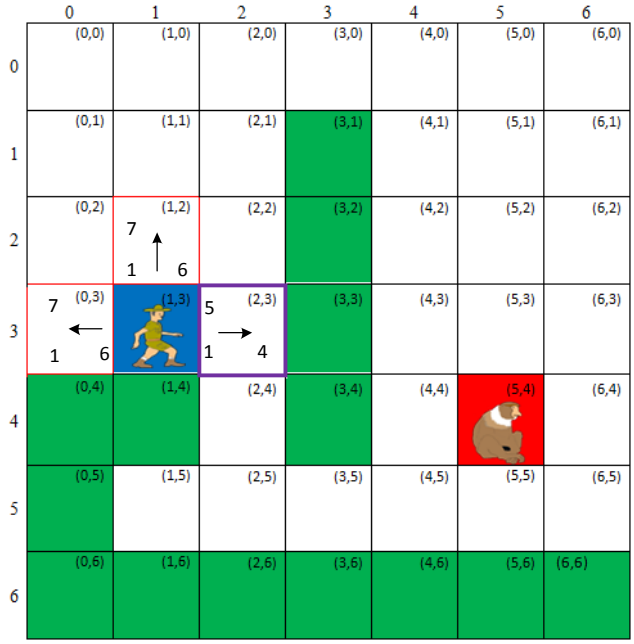
h(0,3)=5+1

h(0,3)=6

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

f(0,3)=1+6

f(0,3)=7



Gambar Error! No text of specified style in document.**.18 Langkah Pertama Pencarian bestnode.**

Dari hasil perhitungan diatas seperti terlihat pada gambar 3.18. Tiga simpul yang mungkin menjadi *bestnode*. Dari ketiga simpul tersebut

yang mungkin menjadi *bestnode* maka dipilihlah simpul (2,3) sebagai *bestnode* karena memiliki *cost* terkecil yaitu 5.

2. Langkah Kedua

Masukkan koordinat (2,2) dan (2,4) untuk menelusuri simpul berikutnya.

a. Pada Koordinat (2,2)

Diketahui : Nilai nx = 2 dan Nilai ny = 2

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

Perhitungan :

g(2,2) = 2

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

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

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

h(2,2) = 3+2

h(2,2) = 5

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

f(2,2) = 1+6

f(2,2) = 7

b. Pada Koordinat (2,4)

Diketahui : Nilai nx = 2 dan Nilai ny = 4

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

Perhitungan :

g(2,4)=2

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

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

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

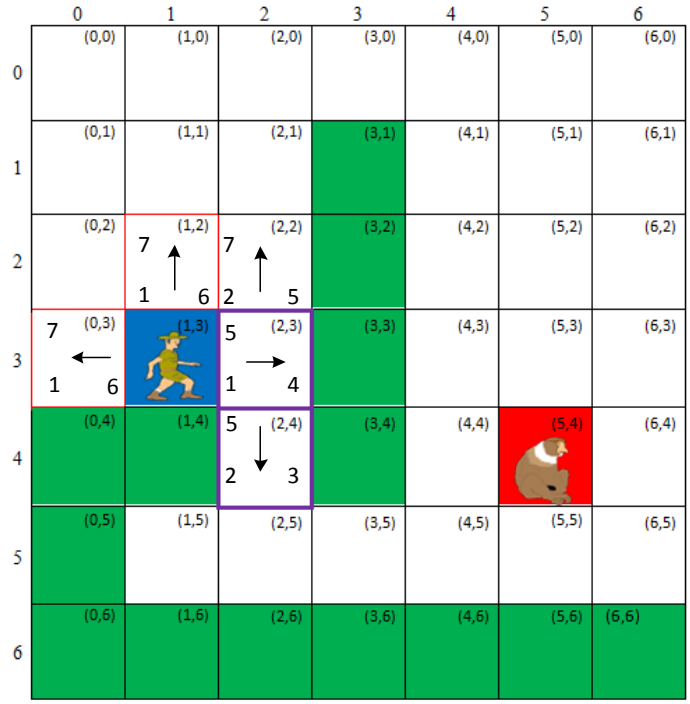
h(2,4)=3+0

h(2,4)=3

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

f(2,4)=2+3

f(2,4)=5



Gambar Error! No text of specified style in document.**.19 Langkah Kedua Pencarian bestnode.**

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

3. Langkah ketiga

   Masukkan koordinat (2,5) untuk menelusuri simpul berikutnya, cek berapa biaya terkecil yang dihasilkannya, sudah memenuhi syarat atau tidak.

   a. Pada Koordinat (2,5)

      Diketahui : Nilai nx = 2 dan Nilai ny =5

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

      Perhitungan :

g(2,5)=3

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

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

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

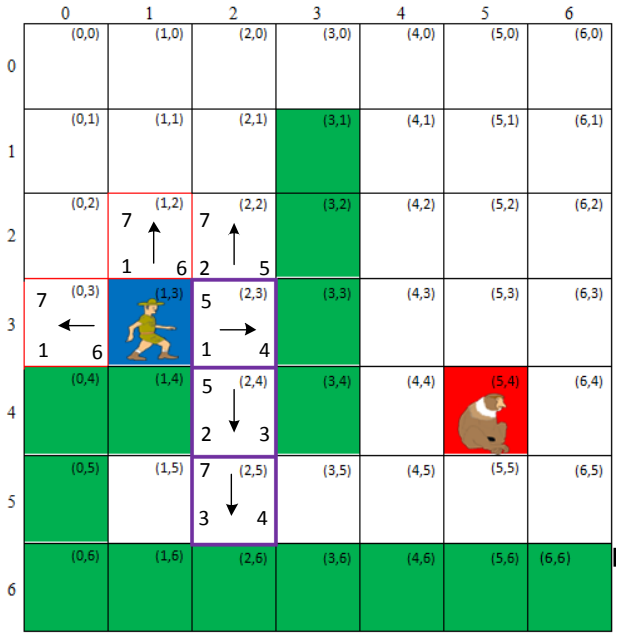
h(2,5)=3+1

h(2,5)=4

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

f(2,5)=3+4

f(2,5)=7



Gambar 3.20 Langkah Ketiga Pencarian bestnode.

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

4. Langkah Keempat

Masukkan Koordinat (3,5) dan (1,5) untuk menelusuri simpul berikutnya bandingkan koordinat tersebut mana yang memiliki *cost* terkecil.

a. Pada Koordinat (3,5)

Diketahui : Nilai nx = 3 dan Nilai ny =5

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

Perhitungan :

g(3,5)=4

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

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

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

h(3,5)=2+1

h(3,5)=3

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

f(3,5)=4+3

f(3,5)=7

b. Pada Koordinat (1,5)

Diketahui : Nilai nx = 1 dan Nilai ny =5

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

Perhitungan :

g(1,5)=4

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

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

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

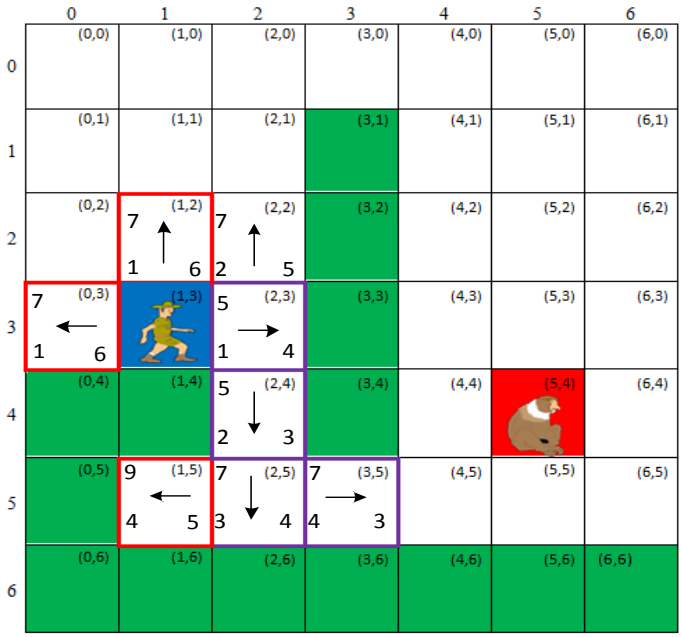
h(1,5)=4+1

h(1,5)=5

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

f(1,5)=4+5

f(1,5)=9



Gambar 3.21 Langkah Keempat Pencarian bestnode

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

5. Langkah Kelima

Masukkan koordinat (4,5) dan (3,4) sebagai simpul berikutnya, cek apakah memberikan *cost* terkecil atau tidak.

a. Pada Koordinat (4,5)

Diketahui : Nilai nx = 4 dan Nilai ny =5

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

Perhitungan :

g(4,5)=5

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

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

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

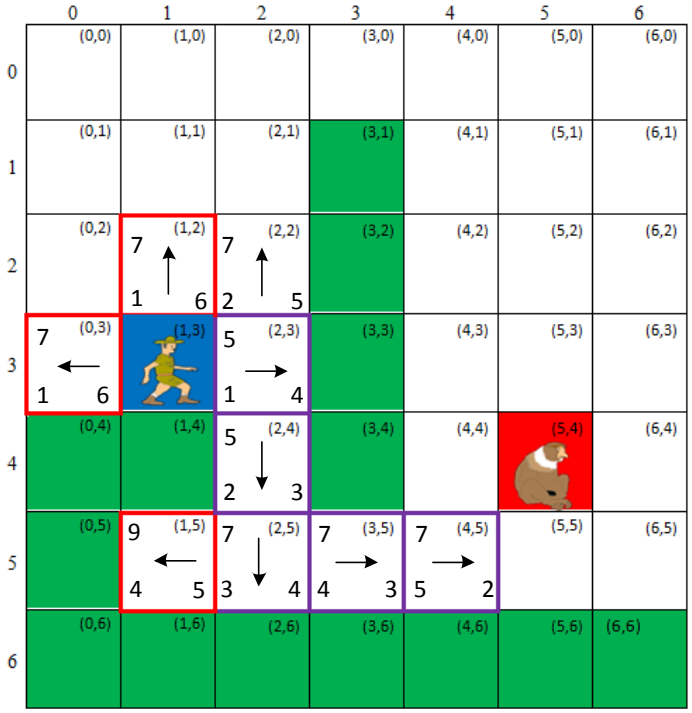
h(4,5)=1+1

h(4,5)=2

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

f(4,5)=5+2

f(4,5)=7



Gambar 3.22 Langkah Kelima Pencarian bestnode

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

6. Langkah Keenam

Masukkan koordinat (5,5) sebagai simpul berikutnya cek apakah memiliki *cost* terkecil atau tidak bandingkan dengan koordinat (4,4).

a. Pada Koordinat (5,5)

Diketahui : Nilai nx = 5 dan Nilai ny = 5

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

Perhitungan :

g(5,5)=6

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

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

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

h(5,5)=0+1

h(5,5)=1

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

f(5,5)=6+1

f(5,5)=7

b. Pada Koordinat (4,4)

Diketahui : Nilai nx = 4 dan Nilai ny = 4

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

Perhitungan :

g(4,4)=6

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

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

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

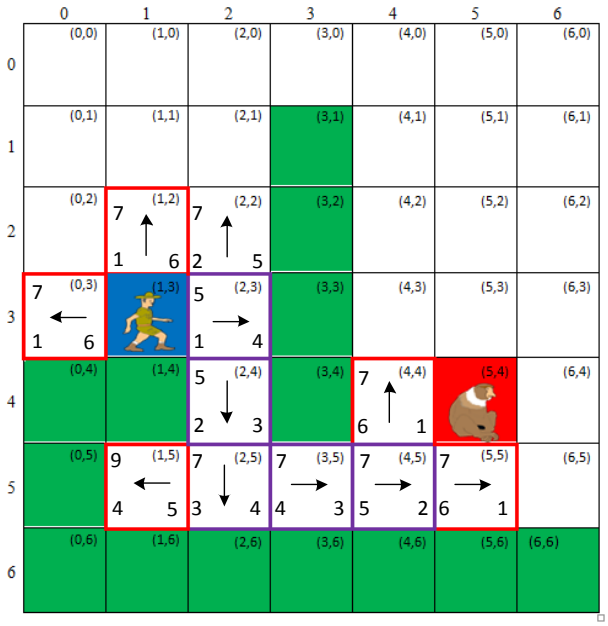
h(4,4)=1+0

h(4,4)=1

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

f(4,4)=6+1

f(4,4)=7



Gambar 3.23 Langkah Keenam Pencarian bestNode

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

7. Langkah Ketujuh

Masukkan koordinat (5,4) dan (6,5) sebagai simpul berikutnya apakah sudah sesuai dengan simpul tujuan atau tidak cek *cost* terkecilnya.

a. Pada Koordinat (5,4)

Diketahui : Nilai nx = 5 dan Nilai ny = 4

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

Perhitungan :

g(5,4)=7

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

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

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

h(5,4)=0+0

h(5,4)=0

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

f(5,4)=7+0

f(5,4)=7

b. Pada Koordinat (6,5)

Diketahui : Nilai nx = 6 dan Nilai ny = 5

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

Perhitungan :

g(6,5)=7

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

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

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

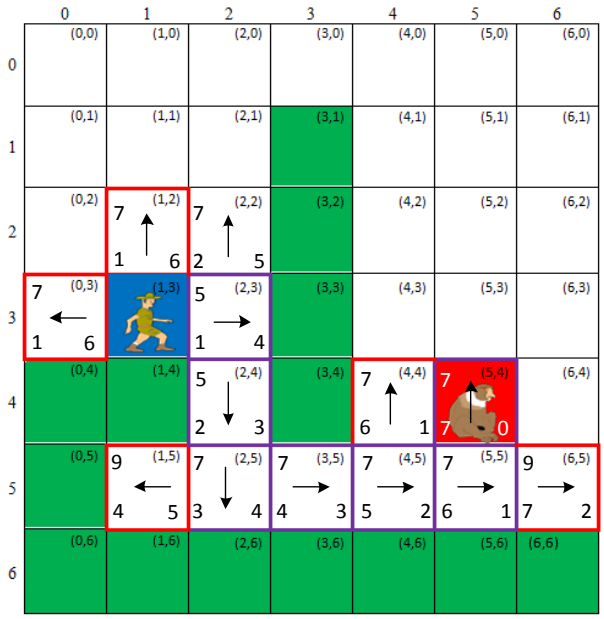
h(6,5)=1+1

h(6,5)=2

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

f(6,5)=7+2

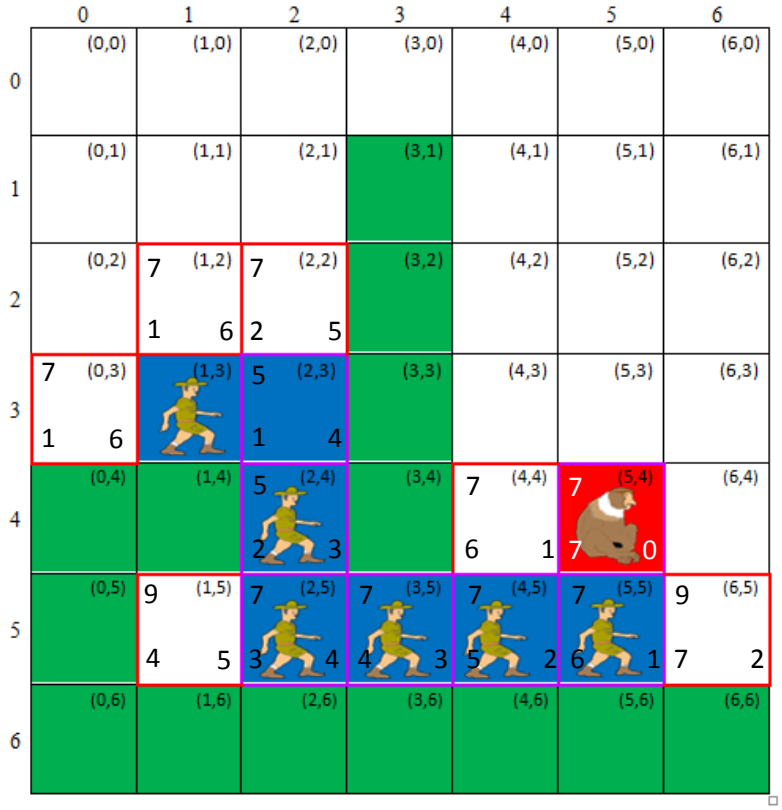
f(6,5)=9



Gambar 3.24 Langkah Ketujuh Pencarian bestNode.

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

Dari perhitunganyang telah dilakukan dipilihlah biaya / *cost* terkecil pada setiap langkahnya sehingga akan menghasilkan jarak terpendek yang akan dilalui hewan endemik untuk melakukan pencarian posisi pemain. Berikut ini dapat dilihat pada gambar 3.25.



Gambar 3.25 Hasil Pencarian Jarak Terpendek Menggunakan Algoritma A*

Dari gambar diatas maka di dapat jarak terpendek dari langkah – langkah diatas dengan mendapatkan posisi (5,4) sebagai posisi *player*/pemain.

Pada *game Endemic Indonesia Island* yang akan dibangun, algoritma A* dijalankan pada episode Kalimantan. Algoritma ini akan berjalan ketika pemburu mencari hewan endemik untuk ditangkap. Ketika hewan endemik itu tertangkap maka *player* harus mulai dari titik awal lagi untuk mendapatkan skor dan menjadikan kesempatan hidup *player* menjadi berkurang 1 setiap tertangkap oleh pemburu.

3.3 Analisis Kebutuhan Non-Fungsional

Kebutuhan non-fungsional adalah batasan layanan atau fungsi yang ditawarkan sistem seperti batasan waktu, batasan pengembangan proses, standarisasi dan lain – lain [39].

Pada analisis kebutuhan sistem non-fungsional ini dijelaskan analisis kebutuhan perangkat lunak, analisis kebutuhan perangkat keras dan analisis pengguna.

3.3.1 Analisis Kebutuhan Perangkat Keras (*Developer*)

Perangkat keras yang dipergunakan dalam pembuatan aplikasi yang akan dibangun menggunakan spesifikasi sebagai berikut :

Tabel 3.2 Analisis Kebutuhan Perangkat Keras (*Developer*)

Nama Perangkat Keras	Spesifikasi
Processor	Intel Core i3-380M, 2.53 Ghz.
Memory	DDR3 4 GB
HDD	500 GB
Display	Resolusi 1360 x 768 VRAM 1 GB
Mouse	1 Unit
Keyboard	1 Unit

3.3.2 Analisis Kebutuhan Perangkat Lunak (*Developer*)

Perangkat lunak yang dipergunakan dalam pembuatan aplikasi yang akan dibangun menggunakan *software* dan *tools* sebagai berikut :

Tabel 3.3 Analisis Kebutuhan Perangkat Lunak (Developer)

Nama Perangkat Lunak	Jenis
Sistem Operasi	*Windows 7*
Software* Pembangun *Game	*Adobe Flash CS5 Actionscript 3.0*
Software Pembangun Game	*Stencyl 3.1 Beta*
***Tools* Pembuat Karakter**	*Adobe Photoshop CS5*
Software* Pendukung *Game	*Adobe Flash Player 10*

3.3.3 Analisis Kebutuhan Perangkat Keras (*User*)

Perangkat keras yang dipergunakan dalam implementasi aplikasi menggunakan spesifikasi minimum sebagai berikut :

Tabel 3.4 Analisis Kebutuhan Perangkat Keras (*User*)

Nama Perangkat Keras	Spesifikasi
Processor	CPU 1.0 Ghz
Memory	DDR1 256 MB
HDD	10 MB Free Harddisk
Display	Display Resolusi 800 x 600
Mouse	1 Unit
Keyboard	1 Unit

3.3.4 Analisis Kebutuhan Perangkat Lunak (*User*)

Perangkat lunak yang dipergunakan dalam implementasi aplikasi menggunakan spesifikasi minimum sebagai berikut :

Tabel 3.5 Analisis Kebutuhan Perangkat Lunak (User)

Nama Perangkat Lunak	Jenis
Sistem Operasi	Windows XP

3.3.5 Analisis Pengguna (*User*)

Pada analisis pengguna (*user*) ini akan mencakup analisis beberapa parameter terhadap calon user dari aplikasi.

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

   Game Endemic Indonesia Island ini bisa digunakan oleh anak umur 3 tahun keatas, tetapi pengetahuan dan pengalaman akan memudahkan *user* dalam permainannya. Terutama pengetahuan dan pengalaman dalam memainkan *casual game*.

Berikut ini klasifikasi *knowledge and experience* dari pemain aplikasi :

Tabel 3.6 *User Knowledge and Experience*

Education Level	***Reading Level***	***Typing Skills***
Pendidikan *user* minimal adalah pendidikan usia dini atau taman kanak - kanak pada umur 3 tahun keatas.	Kemampuan *user* bisa berbahasa Indonesia.	*User* tidak memerlukan *typing skills* yang tinggi.
Computer Literacy	***Task Experience***	***System Experience***
User atau pengguna yang memiliki kemampuan komputer yang *moderate* (menengah).	*User atau* pengguna sudah pernah menggunakan komputer dan bisa menggunakan komputer.	*User* bisa menggunakan fungsi-fungsi pada komputer dengan baik.
Application Experience	***Native Language***	***Use Of Other System***
User atau pengguna bisa menggunakan operasi *windows XP/Vista/7* untuk menjalankan *game*.	Bahasa yang digunakan, yakni bahasa Indonesia untuk dasar penggunaan bahasa dalam *game.*	*User* atau pengguna harus memiliki aplikasi pendukung yaitu *flash player* untuk menjalankan *game*.

2. ***Users Physical Characteristic***

   Keadaan fisik pengguna (*user*) mungkin akan berpengaruh pada penggunaan aplikasi game ini. Ada hal-hal yang harus diperhatikan terhadap pengguna (*user*) dari karakteristik fisiknya untuk dapat menggunakan aplikasi ini yaitu, umur, buta warna, jenis kelamin, dan Penggunaan tangan.

Tabel 3.7 *Users Physical Characteristic*

Umur	3 tahun keatas
Jenis Kelamin	Laki – laki atau Perempuan
Penggunaan Tangan	Kanan atau Kiri
Buta Warna	*User* yang tidak bisa membedakan warna yang satu dengan yang lainnya, maka tidak akan optimal dalam menggunakan aplikasi ini, karena pada *game Endemic Indonesia Island* terdapat warna-warna yang fungsinya untuk membedakan antara fungsional yang satu dengan fungsional yang lainnya.

3.4 Analisis Kebutuhan Fungsional

Kebutuhan fungsional adalah pernyataan layanan sistem yang harus disediakan, bagaimana sistem bereaksi pada *input* tertentu dan bagaimana perilaku sistem pada situasi tertentu. Sedangkan kebutuhan fungsional *user* merupakan pernyataan level tinggi dari apa yang seharusnya dilakukan sistem tetapi kebutuhan fungsional sistem menggambarkan layanan sistem secara detail [39].

3.4.1 Definisi Aktor (*Actor*)

Tahap pertama yang dilakukan dalam melakukan analisis berorientasi objek menggunakan UML (*Unified Modeling Language*) adalah menentukan aktor atau pengguna sistem. Aktor (*actor*) merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat[31].

Berikut ini adalah deskripsi definisi aktor pada *game endemic Indonesia island* dapat dilihat pada tabel 3.8.

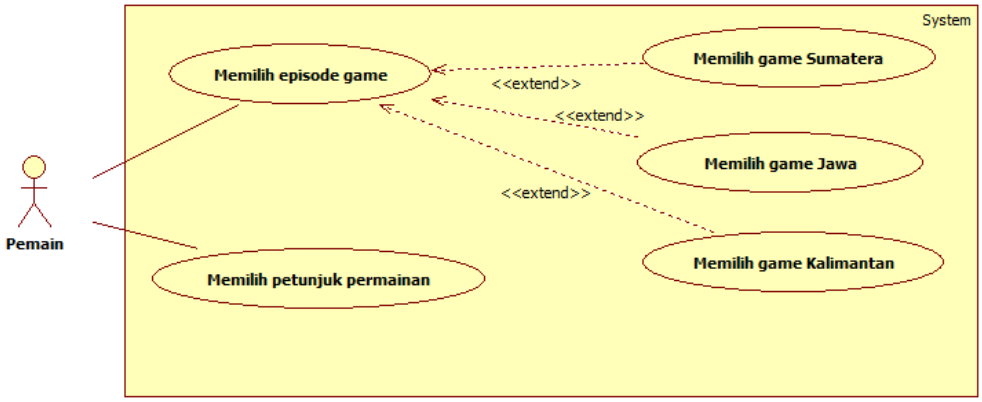
Tabel 3.8 Definisi Aktor (*Actor*)

No	Aktor	Deskripsi
1.	Pemain	Orang yang menggunakan permainan *endemic Indonesia island*.

3.4.2 *Use Case Diagram*

Use case diagram digunakan untuk memodelkan bisnis proses berdasarkan perspektif pengguna sistem. *Use case* diagram terdiri atas diagram untuk *use case* dan *actor*. *Use case* merepresentasikan operasi-operasi yang dilakukan oleh *actor* [31].

Berikut adalah deskripsi pendefinisian *use case diagram* pada *game endemic Indonesia island.*



Gambar 3 Error! No text of specified style in document.**..26 *Use Case Diagram Game Endemic Indonesia Island***

3.4.3 Definisi *Use Case Diagram*

Berikut ini deskripsi pendefinisian *use case diagram* pada *Game Endemic Indonesia Island.*

Tabel 3.9 Definisi *Use Case Diagram*

No	*Use Case*	Deskripsi
1.	Memilih episode *game*	Merupakan proses memilih episode *game* yang akan dimainkan.
2.	Memilih *game Sumatera*	Merupakan proses pemain memainkan *game Sumatera*.
3.	Memilih *game Jawa*	Merupakan proses pemain memainkan *game Jawa*.
4.	Memilih *game Kalimantan*	Merupakan proses pemain memainkan *game Kalimantan*.
5.	Memilih *petunjuk permainan*	Merupakan proses pemain memilih menu *petunjuk* untuk mengetahui cara bermain *game.*

3.4.4 *Use Case Scenario*

Use case scenario adalah sebuah dokumentasi terhadap kebutuhan fungsional dari sebuah sistem [31]. Berikut ini adalah *use case scenario* pada *game Endemic Indonesia Island*.

1. Nama *Use Case* : Memilih Episode *Game*.

   Skenario :

Tabel 3.10 Skenario Memilih Episode *Game*

Identifikasi	
Nomor	1
Nama	Memilih Episode *Game*
Deskripsi	Merupakan proses pemain memilih episode *game*

Aktor	Pemain.	
Kondisi Awal	Pemain belum masuk menu pilih episode *game*.	
Kondisi Akhir	Pemain sudah masuk menu pilih episode *game*.	
Skenario Awal	**Aksi Aktor**	**Reaksi Sistem**
Jika	1. Menekan tombol **Mulai** pada menu utama.	2. Menampilkan jendela menu pilih episode *game*.
	3. Menekan tombol **Kembali** pada menu pilih episode *game.*	4. Menampilkan menu utama *game*.
Skenario Alternatif	1. Tidak menekan tombol **Mulai** pada menu utama *game*.	2. Menampilkan menu utama *game*.

2. Nama *Use Case* : Memilih *Game* Sumatera.

   Skenario :

Tabel 3.11 Skenario Memilih *Game* Sumatera

Identifikasi	
Nomor	2
Nama	Memilih *Game* Sumatera
Deskripsi	Merupakan proses pemain memilih episode *game* Sumatera.
Aktor	Pemain.
Kondisi Awal	Pemain belum memainkan *game* Sumatera.

Kondisi Akhir	Pemain sudah memainkan *game* Sumatera.	
Skenario Awal	**Aksi Aktor**	**Reaksi Sistem**
Jika	1. Menekan tombol **Pulau Sumatera** pada episode Sumatera.	2. Menampilkan jendela *game Sumatera*.
	3. Menekan tombol **Kembali** pada menu *game.*	4. Menampilkan menu pilih episode *game.*
	5. Menekan tombol **Menu** pada menu *game*	6. Menampilkan menu utama *game*.
Skenario Alternatif	1. Tidak menekan tombol **Pulau Sumatera** pada pilih episode *game.*	2. Menampilkan menu pilih episode *game*.

3. Nama *Use Case* : Memilih *Game* Jawa.

   Skenario :

Tabel 3.12 Skenario Memilih *Game* Jawa

Identifikasi	
Nomor	3
Nama	Memilih *Game Jawa.*
Deskripsi	Merupakan proses pemain memilih episode *game Jawa*.
Aktor	Pemain.

Kondisi Awal	Pemain belum memainkan *game* Jawa.	
Kondisi Akhir	Pemain sudah memainkan *game* Jawa.	
Skenario Awal	**Aksi Aktor**	**Reaksi Sistem**
Jika	1. Menekan tombol **Pulau Jawa.**	2. Menampilkan jendela *game Jawa*.
	3. Menekan tombol **Kembali** pada menu *game*.	4. Menampilkan menu pilih episode *game*.
	5. Menekan tombol **Menu** pada menu *game* Jawa.	6. Menampilkan menu utama *game*.
Skenario Alternatif	1. Tidak menekan tombol **Pulau Jawa** pada menu pilih episode *game*.	2. Menampilkan menu pilih episode *game*.

4. Nama *Use Case* : Memilih *Game* Kalimantan.
   Skenario :

Tabel 3.13 Skenario Memilih Game Kalimantan

Identifikasi	
Nomor	4
Nama	Memilih *Game* Kalimantan
Deskripsi	Merupakan proses pemain memilih episode *game Kalimantan.*

Aktor	Pemain.	
Kondisi Awal	Pemain belum memainkan *game* Kalimantan.	
Kondisi Akhir	Pemain sudah memainkan *game* Kalimantan.	
Skenario Awal	Aksi Aktor	Reaksi Sistem
Jika	1. Menekan tombol **Pulau Kalimantan**.	2. Menampilkan jendela *game Kalimantan*.
	3. Menekan tombol **Kembali** pada menu *game*.	4. Menampilkan menu pilih episode *game*.
Skenario Alternatif	1. Tidak menekan tombol **Pulau Kalimantan** pada menu pilih episode *game*.	2. Menampilkan menu pilih episode *game*.

5. Nama *Use Case* : Memilih Petunjuk Permainan.

   Skenario :

Tabel 3.14 Memilih Petunjuk Permainan

Identifikasi	
Nomor	5
Nama	Memilih Petunjuk Permainan.
Deskripsi	Merupakan proses pemain memilih menu *petunjuk* untuk mengetahui cara bermain *game*.

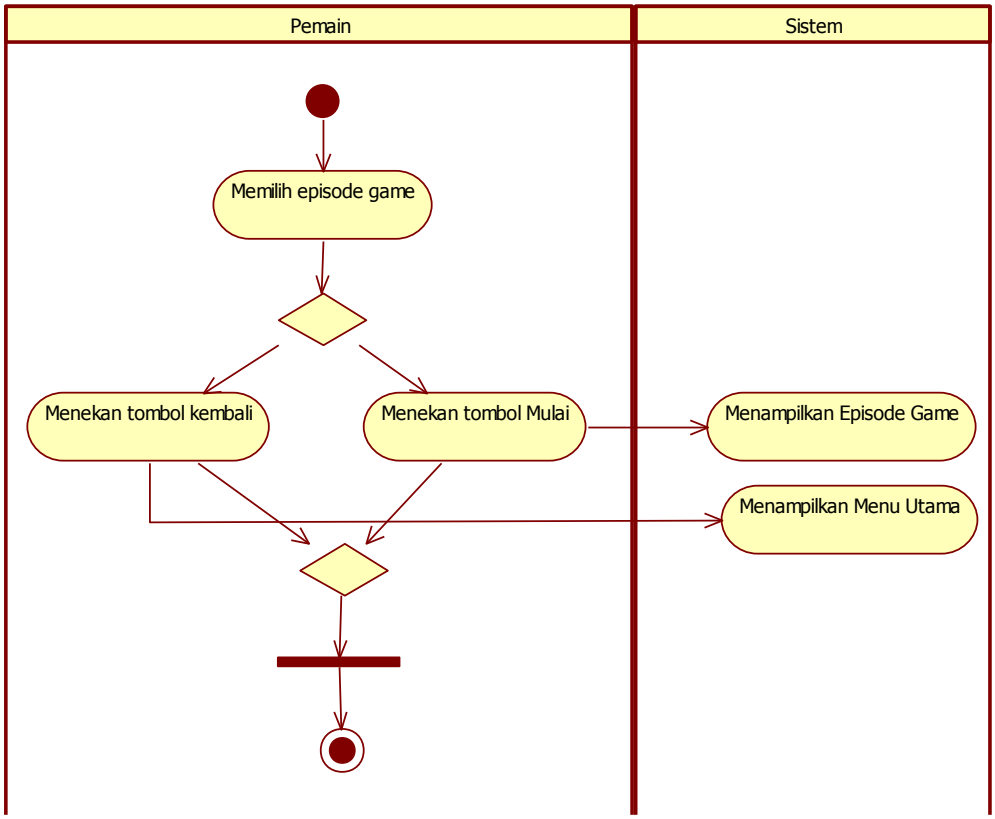
Aktor	Pemain.	
Kondisi Awal	Pemain belum masuk ke menu **Petunjuk**.	
Kondisi Akhir	Pemain sudah masuk ke menu **Petunjuk**.	
Skenario Awal	**Aksi Aktor**	**Reaksi Sistem**
Jika	1. Menekan tombol **Petunjuk**.	2. Menampilkan jendela cara bermain *game*.
	3. Menekan tombol **Kembali** pada menu**.**	4. Menampilkan menu utama *game*.
Skenario Alternatif	1. Tidak menekan tombol **Petunjuk.**	2. Menampilkan menu utama *game*.

3.4.5 *Activity Diagram*

Activity Diagram merupakan gambaran *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. *Activity diagram* menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, yaitu aktivitas yang dapat dilakukan oleh sistem [31].

1. ***Activity Diagram* Memilih Episode *Game***

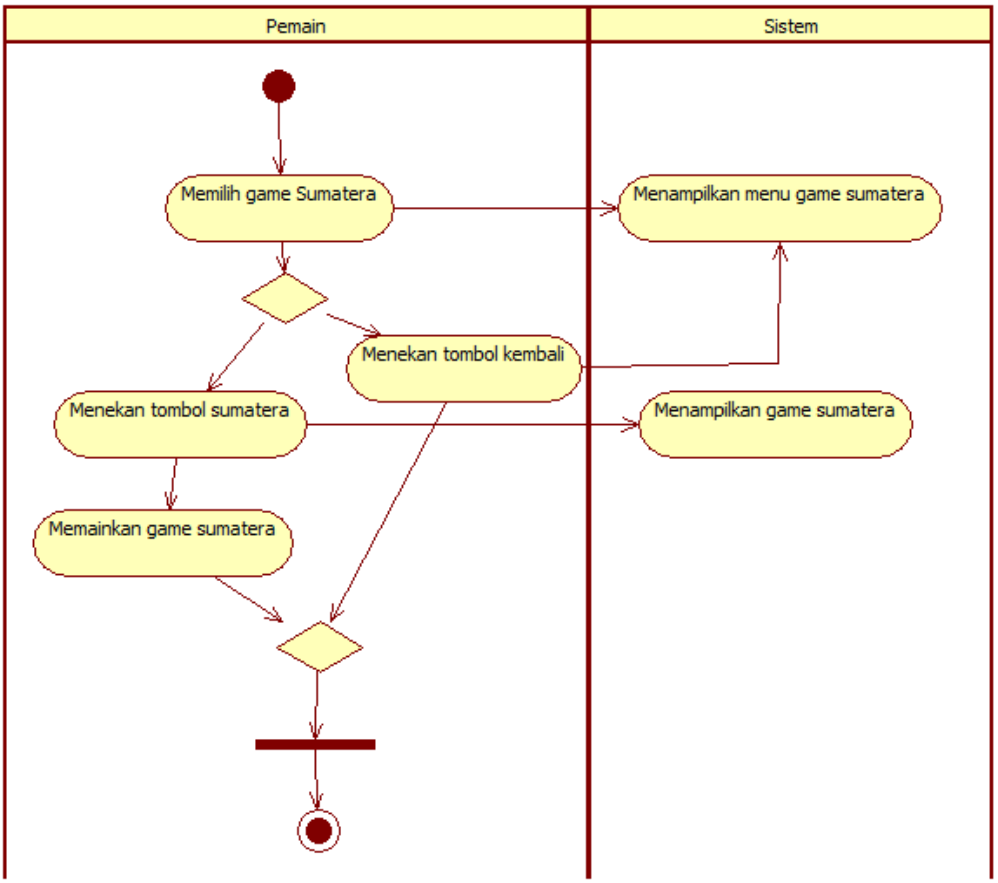
   Berikut ini adalah gambar *activity diagram* memilih episode *game* dapat dilihat pada gambar 3.27.



Gambar 3.27 *Activity Diagram* Memilih Episode *Game*

2. *Activity Diagram* Memilih *Game* Sumatera

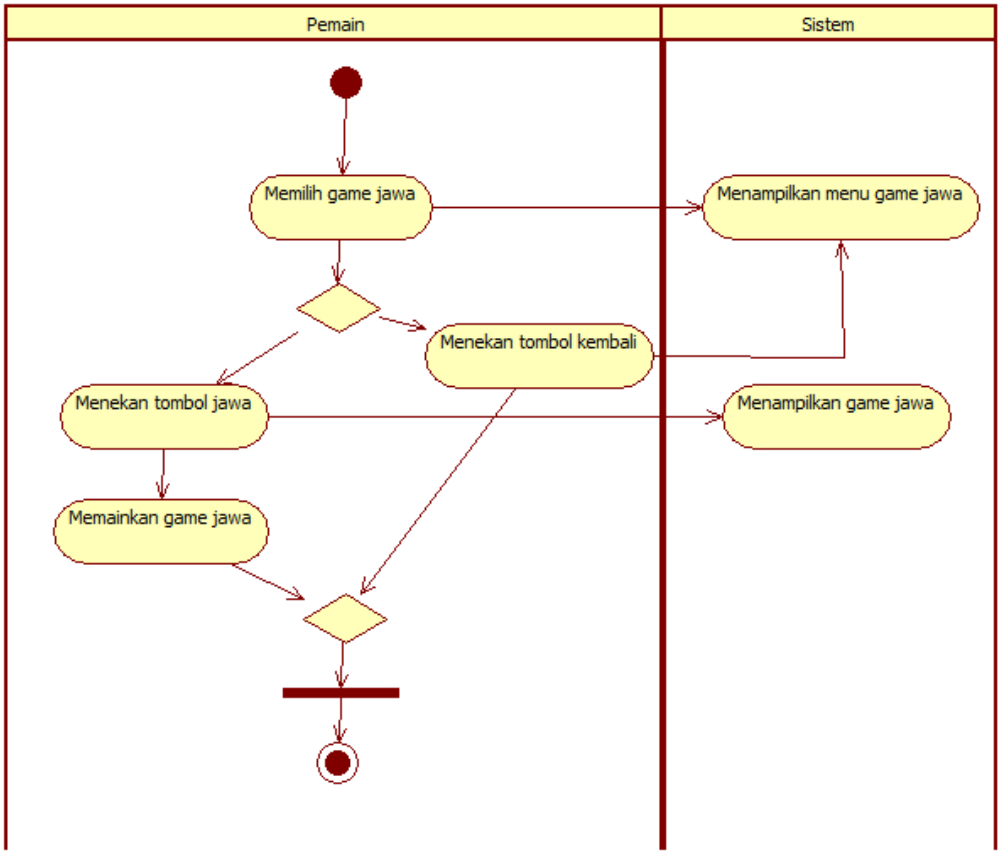
Berikut ini adalah gambar *activity diagram* memilih *game* Sumatera dapat dilihat pada gambar 3.28.



Gambar 3.28 *Activity Diagram* Memilih *Game* Sumatera

3. ***Activity Diagram* Memilih *Game* Jawa**

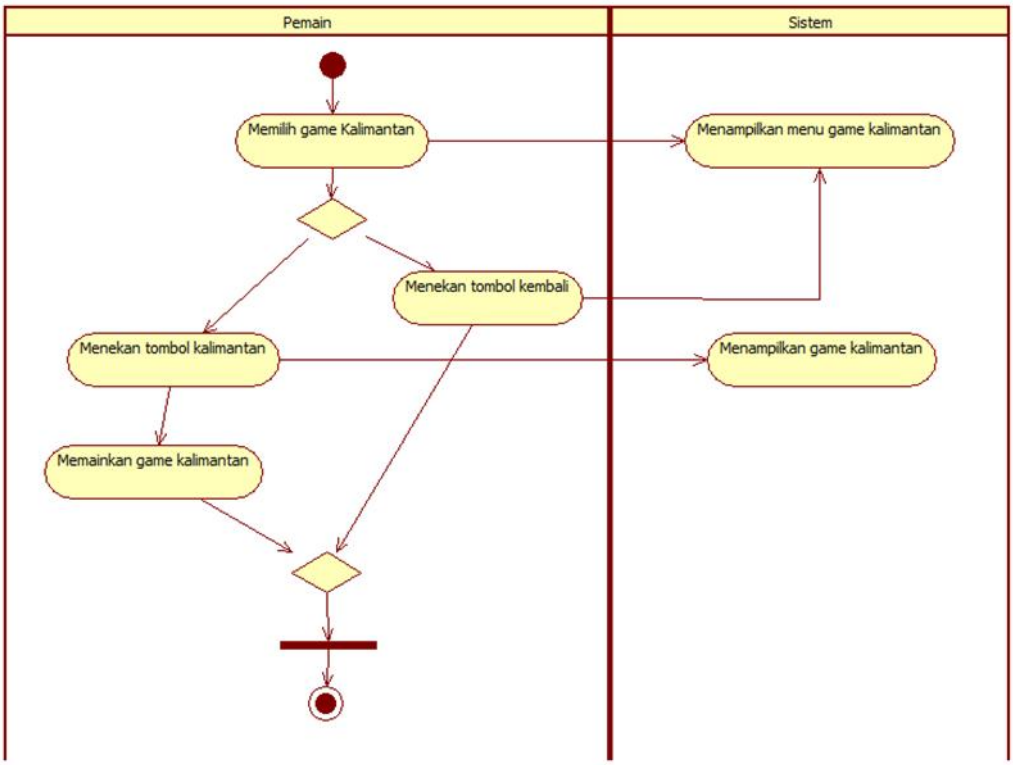
   Berikut ini adalah gambar *activity diagram* memulai *game Jawa* dapat dilihat pada gambar 3.29.



Gambar 3.29 *Activity Diagram* Memilih *Game* Jawa

4. *Activity Diagram* Memilih *Game* Kalimantan

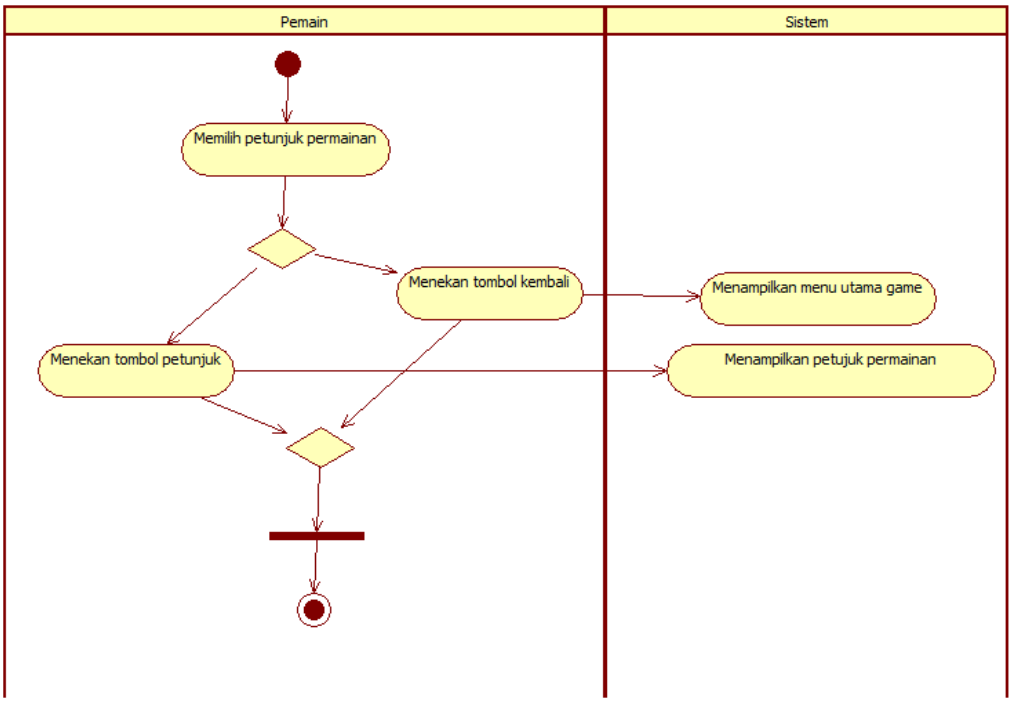
Berikut ini adalah gambar *activity diagram* memilih *game* Kalimantan dapat dilihat pada gambar 3.30.



Gambar 3.30 *Activity Diagram* Memilih *Game* Kalimantan

5. *Activity Diagram* Memilih Petunjuk Permainan

Berikut ini adalah gambar *activity diagram* memilih petunjuk permainan dapat dilihat pada gambar 3.31.



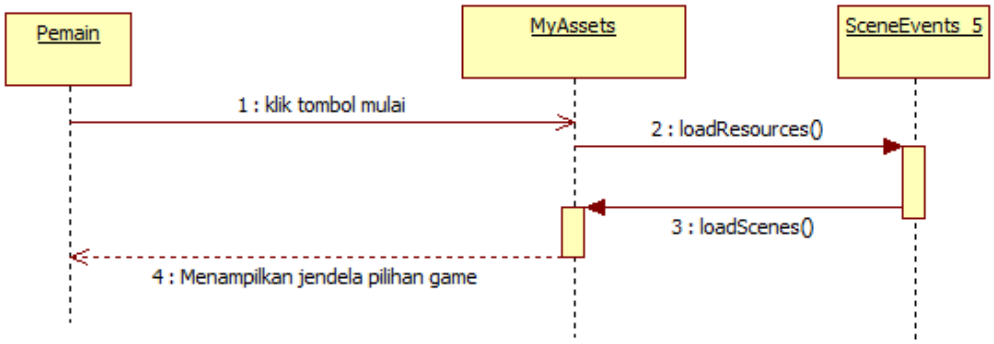
Gambar 3.31 *Activity Diagram* Memilih Petunjuk Permainan

3.4.6 Sequence Diagram

Sequence Diagram merupakan diagram yang menggambarkan aktivitas dan relasi kelas pada objek *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima objek. Banyak *sequence diagram* harus digambarkan sebanyak pendefinisian *use case.* Berikut ini adalah *sequence diagram* dari *game endemic Indonesia island*.

1. ***Sequence Diagram* Memilih Episode *Game*.**

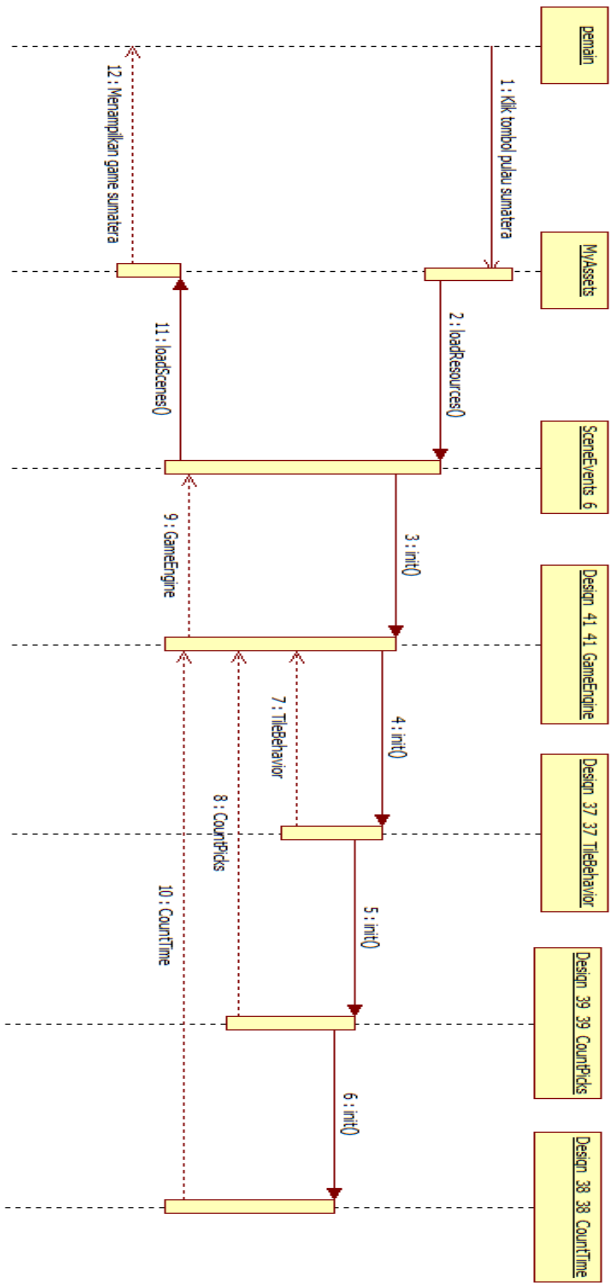
   Berikut ini adalah gambar *sequence diagram* memilih episode *game* dapat dilihat pada gambar 3.32.



Gambar 3.32 *Sequence Diagram* Memilih Episode *Game*

2. ***Sequence Diagram* Memilih *Game* Sumatera.**

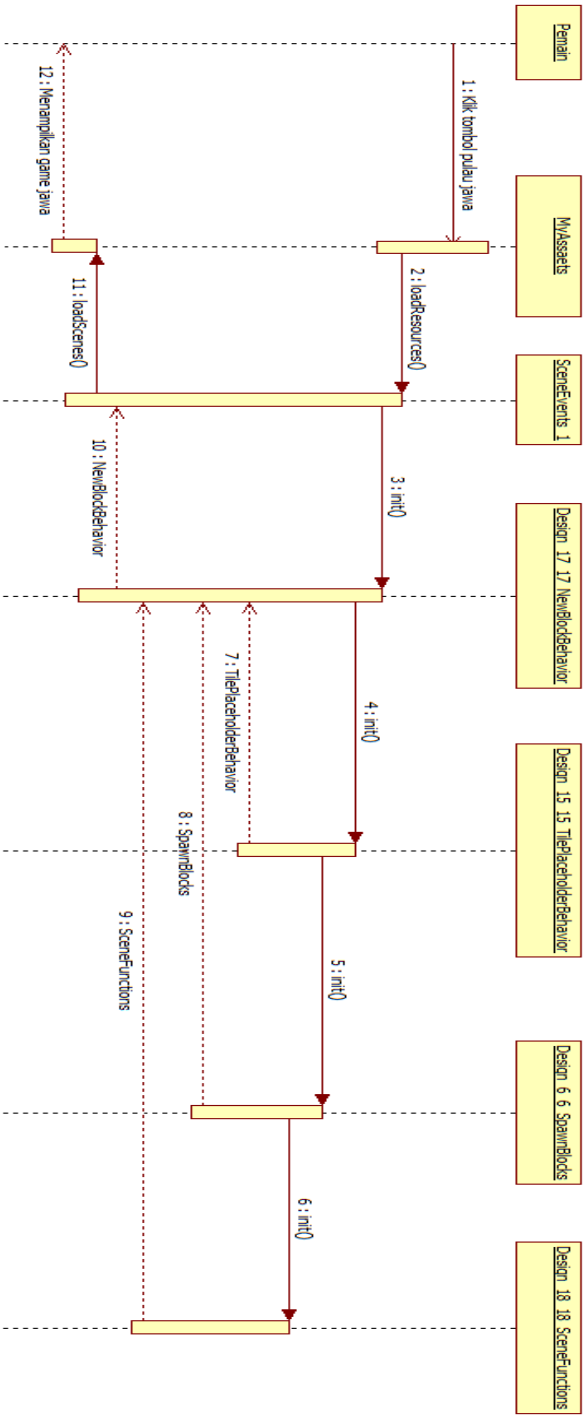
   Berikut ini adalah gambar *sequence diagram* memilih *game* sumatera dapat dilihat pada gambar 3.33.



Gambar 3.33 *Sequence Diagram* Memilih *Game* Sumatera

3. *Sequence Diagram* Memilih *Game* Jawa.

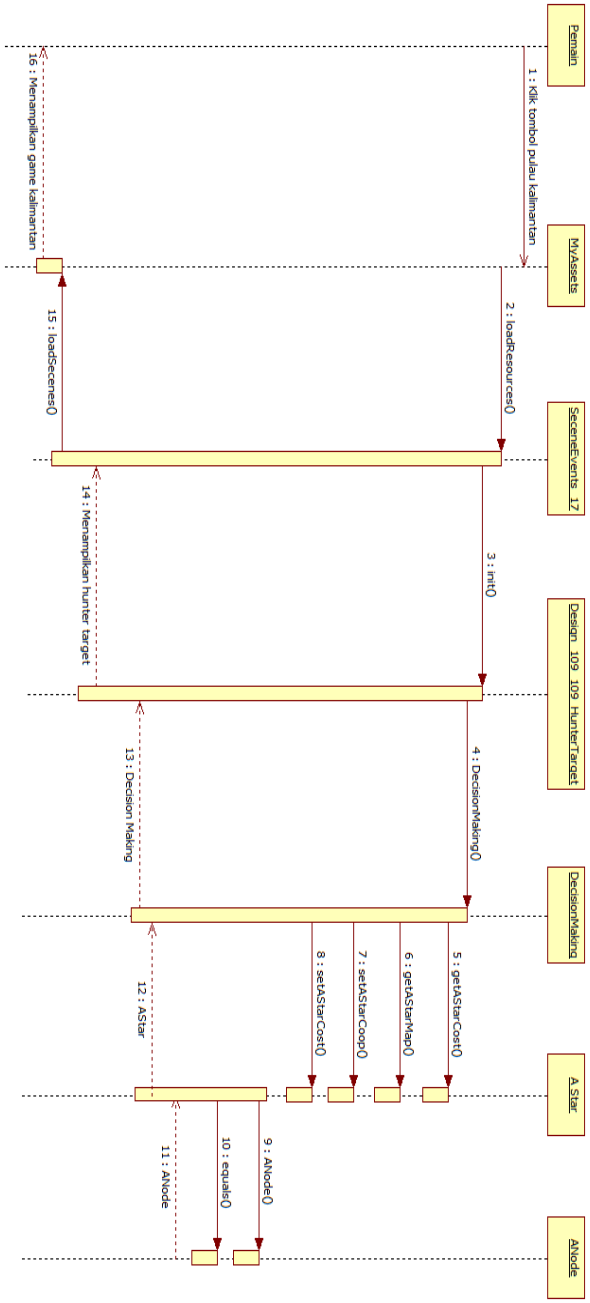
Berikut ini adalah gambar *sequence diagram* memilih *game* jawa dapat dilihat pada gambar 3.34.



Gambar 3.34 *Sequence Diagram* Memilih *Game* Jawa

4. *Sequence Diagram* Memilih *Game* Kalimantan.

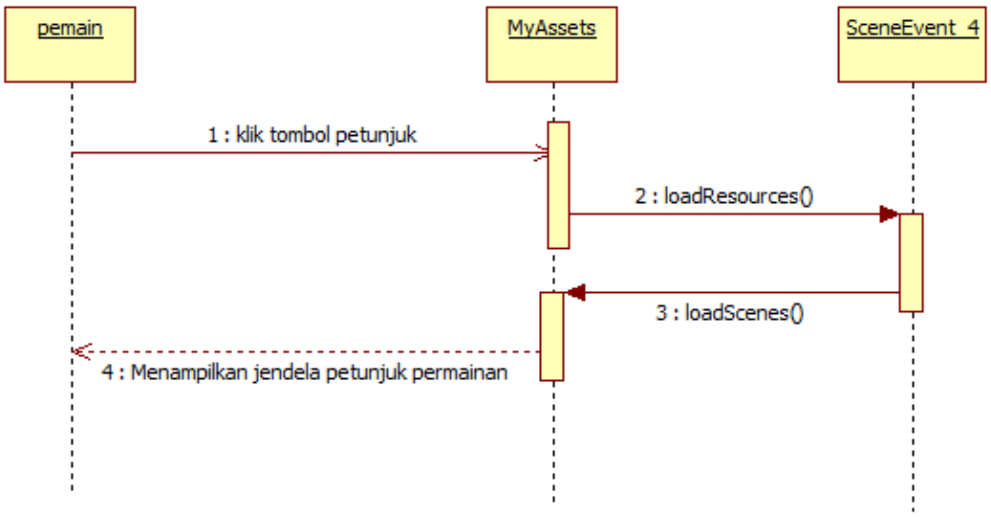
Berikut ini adalah gambar *sequence diagram* memilih *game* kalimantan dapat dilihat pada gambar 3.35.



Gambar 3.35 *Sequence Diagram* Memilih *Game* Kalimantan

5. ***Sequence Diagram* Memilih Petunjuk Permainan.**

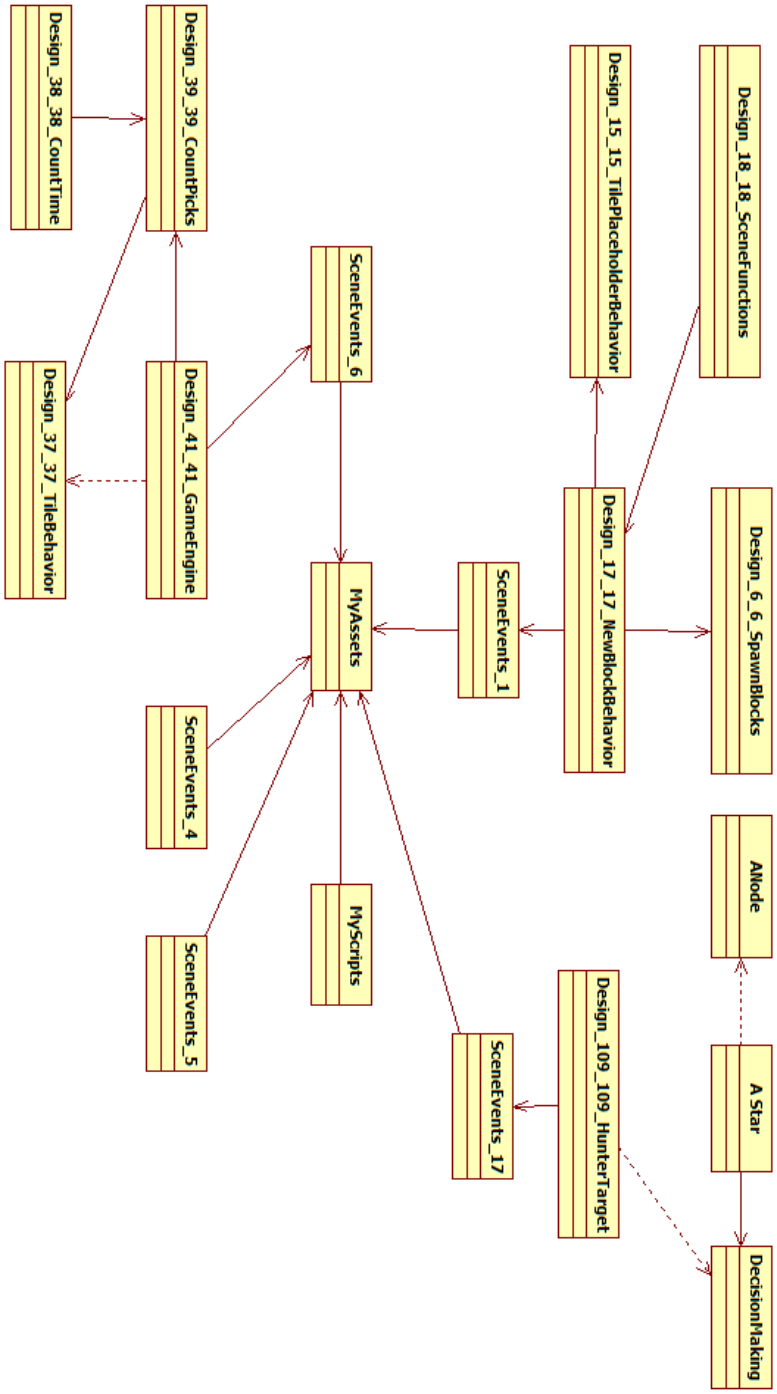
   Berikut ini adalah gambar *sequence diagram* memilih petunjuk permainan dapat dilihat pada gambar 3.36.



Gambar 3.36 *Sequence Diagram* Memilih Petunjuk Permainan.

3.4.7 *Class Diagram*

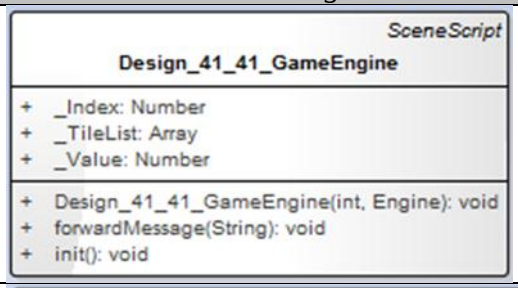
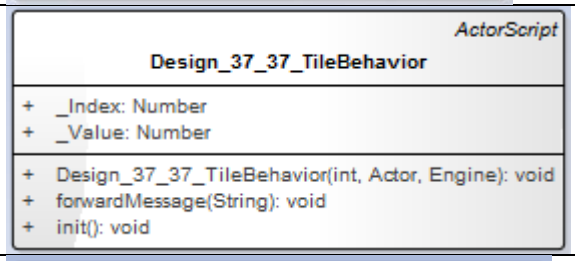
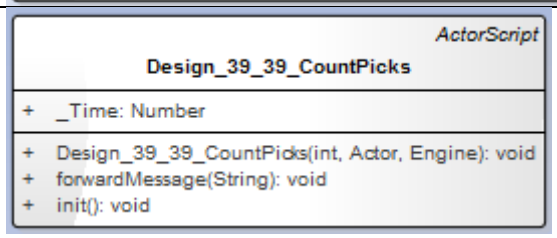
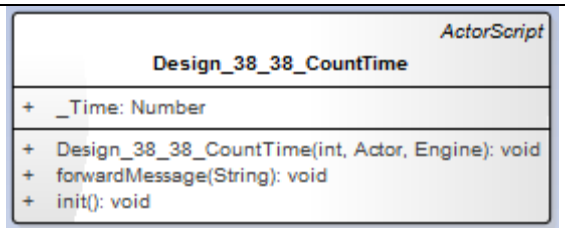
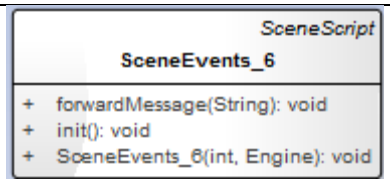
Class diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi. Atribut merupakan variabel - variabel yang dimiliki oleh suatu kelas sedangkan operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas. Berikut ini *class diagram game endemic Indonesia Island* dapat dilihat pada gambar 3.37.

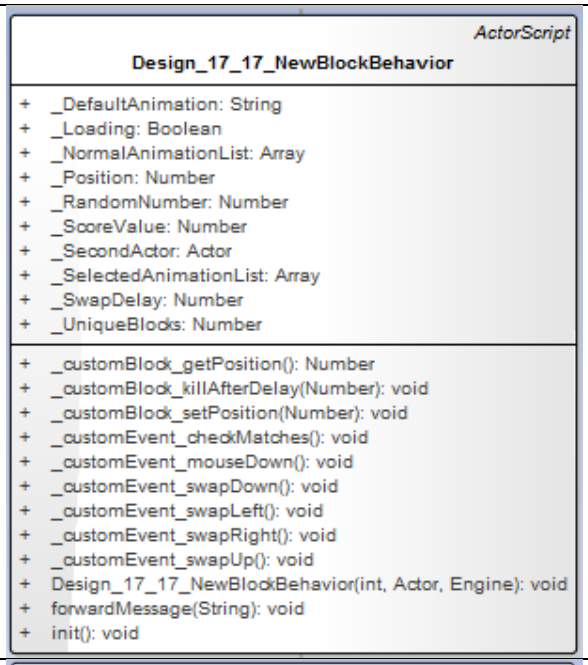
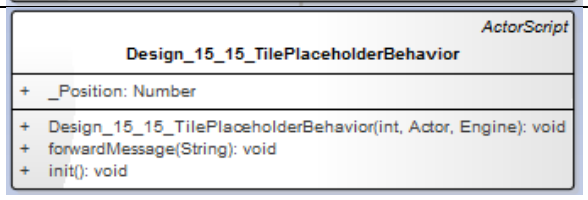
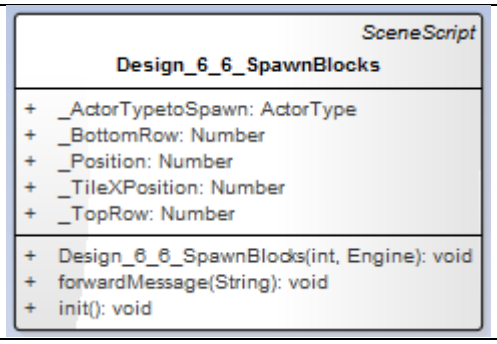


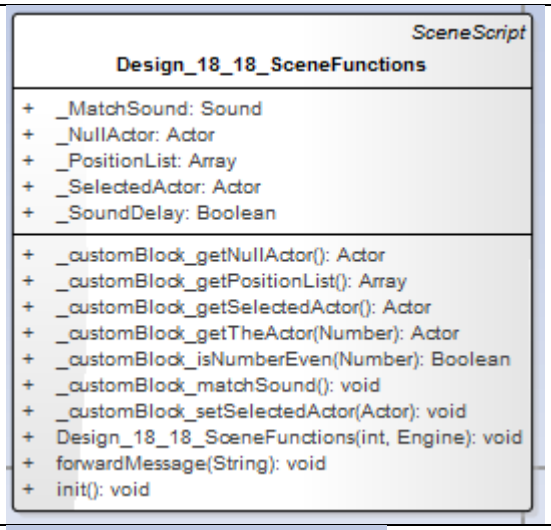
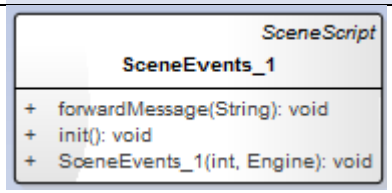
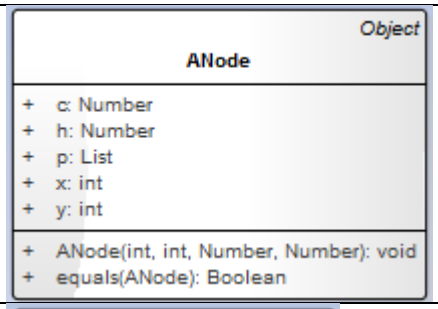
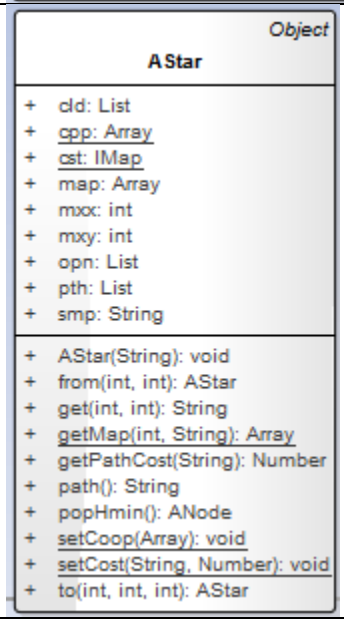
Gambar 3.37 ***Class Diagram***

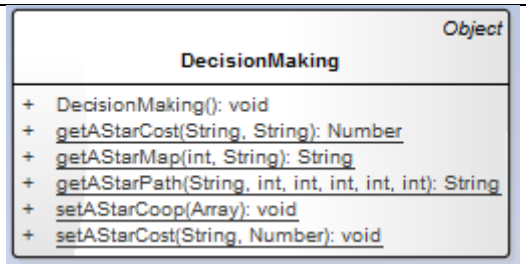
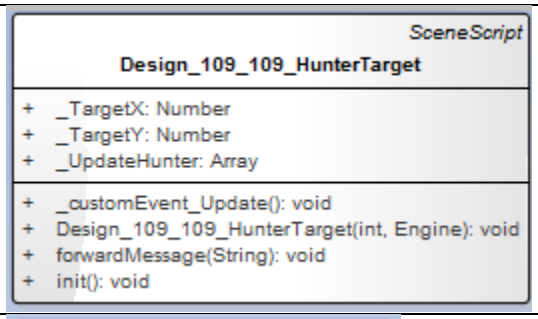
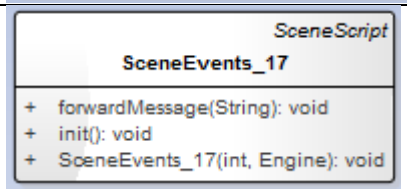
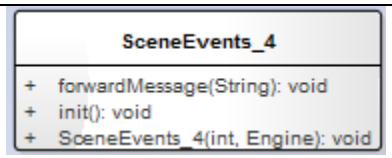
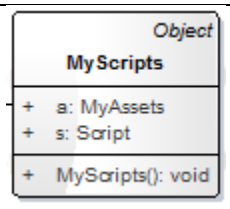
Berikut ini adalah keterangan *class diagram* yang ditunjukan pada tabel 3.15 :

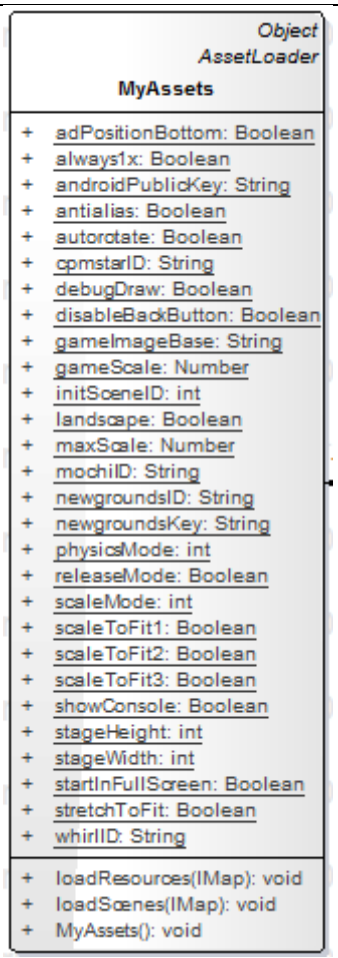
Tabel 3.15 Keterangan Class Diagram

No	*Class Diagram*	Keterangan
1		Merupakan kelas utama yang akan memanggil fungsi kelas lainnya untuk menjalankan *game* pada episode *game* sumatera.
2		Merupakan kelas untuk membuat deretan kartu yang akan ditampilkan pada *game* sumatera.
3		Merupakan kelas untuk menghitung jumlah kartu yang terklik oleh *mouse* baik yang benar maupun kartu yang salah pada *game* sumatera.
4		Merupakan kelas untuk menghitung waktu yang berjalan saat awal *game* hingga akhir *game* yang dimainkan pada episode sumatera.
5		Merupakan kelas yang akan menampilkan jendela *game* sumatera yang akan dimainkan.

6		Merupakan kelas untuk membuat dan mengurangi blok-blok *piece* yang telah dimainkan. Baik itu mengisi ruang blok kosong yang telah terselesaikan oleh pemain maupun menghilangkan *piece* yang sudah tersamakan pada episode *game* jawa.
7		Merupakan kelas untuk membuat deretan blok-blok yang akan ditampilkan pada jendela *game* jawa.
8		Merupakan kelas untuk membuat animasi *piece* saat ditukarkan dengan *piece* yang lainnya baik *piece* yang benar maupun salah.

9		Merupakan kelas untuk mengatur posisi blok yang akan ditampilkan pada *game* jawa menurut kelas *TilePlaceHolderehavior* pada *game* jawa.
10		Merupakan kelas untuk menampilkan jendela *game* yang akan dimainkan pada episode jawa.
11		Merupakan kelas untuk membuat *node* yang akan dijalankan oleh algoritma A* untuk membuat path pencarian untuk NPC.
12		Merupakan kelas untuk membuat algoritma A* untuk pencarian NPC dari node yang telah terdapat pada kelas node.

13		Merupakan kelas utama untuk memanggil algoritma A* yang akan diterapkan pada *behavior* musuh atau NPC dalam pencarian aktor *player*.
14		Merupakan kelas NPC yang diterapkan dengan kelas *decisionMaking* sebagai behavior untuk menjalankan NPC pada *game* Kalimantan.
15		Merupakan kelas untuk menampilkan jendela *game* yang akan dimainkan pada episode Kalimantan.
16		Merupakan kelas untuk memanggil jendela menu petunjuk permainan.
17		Merupakan kelas untuk memanggil jendela menu pilihan *game* yang akan dimainkan.
18		Merupakan kelas untuk membuat *package script* yang akan memanggil kelas *MyAssets.*

19		Merupakan kelas utama yang akan memanggil semua kelas untuk ditampilkan menjadi satu game yang utuh, yang mengontrol semua kelas yang ada.

3.5 Perancangan Sistem

Perancangan sistem adalah merancang atau mendesain sistem untuk memenuhi kebutuhan *user* mengenai gambaran yang jelas tentang perancangan sistem yang akan dibuat serta di implementasikan[37].

3.5.1 Perancangan Komponen Permainan

Berikut ini adalah perancangan komponen pada permainan *game endemic Indonesia Island*.

3.5.1.1 Deskripsi Karakter

Deskripsi karakter adalah karakter atau gambar yang dipakai dalam *game endemic Indonesia Island*.

Berikut ini adalah deskripsi karakter dalam *game endemic Indonesia island*.

1. Karakter Episode Sumatera.

   Berikut ini adalah karakter dari *game* episode sumatera dapat dilihat pada tabel 3.16.

Tabel 3.16 Karakter Episode Sumatera

No	Gambar	Keterangan/ Nama Hewan Endemik
1.		Karakter kartu *background* untuk tampilan awal dan menutup gambar yang di mainkan pada *matching pair game.*
2.		Karakter kartu burung beo nias (*Gracula religiosa robusta*).
3.		Karakter kartu badak sumatera (*Dicerorhinus sumatrensis*).
4.		Karakter kartu gajah sumatera (*Elephas maximus sumatrensis*).

5.		Karakter kartu harimau sumatera (*Panthera tigris sumatrae*).
6.		Karakter kartu kambing hutan Sumatera *(Capricornis sumatraensis).*
7.		Karakter kartu kelinci belang sumatera *(Nesolagus netscheri).*
8.		Karakter kartu orangutan sumatera (*Pongo abelli*).
9.		Karakter kartu siamang (*Hylobates syndactylus*).

2. Karakter Episode Jawa.

   Berikut ini adalah karakter dari *game* episode Jawa dapat dilihat pada tabel 3.17.

Tabel 3.17 Karakter Episode Jawa

No	Gambar	Keterangan/ Nama Hewan Endemik
		Karakter kartu badak bercula satu (*Rhinoceros sondaicus*).
		Karakter kartu elang jawa (*Spzaetus bartelis*).
		Karakter kartu kancil jawa (*Tragulus javanicus*).
		Karakter kartu kukang jawa (*Nycticebus javanicus*).
		Karakter kartu lutung jawa (*Trachypithecus auratus*).
		Karakter kartu trulek jawa (*Vanellus macropterus*).

3. Karakter Episode Kalimantan.

   Berikut ini adalah karakter dari game episode kalimantan dapat dilihat pada tabel 3.18.

Tabel 3.18 Karakter Episode Kalimantan

No	Gambar	Keterangan
1.		Karakter musuh atau NPC yang akan mengikuti aktor yaitu pemburu.
2.		Karakter hewan endemik yang akan dimainkan oleh pemain yaitu Bekantan (Nasalis larvatus).
3.		Karakter koin untuk menambah skor bagi *player*.

3.5.1.2 Story Board

Story board adalah organisator grafis dalam bentuk ilustrasi atau gambar yang ditampilkan secara berurutan untuk tujuan pra-visualisasi gambar gerak, animasi, *motion graphic* atau urutan media interaktif [24].

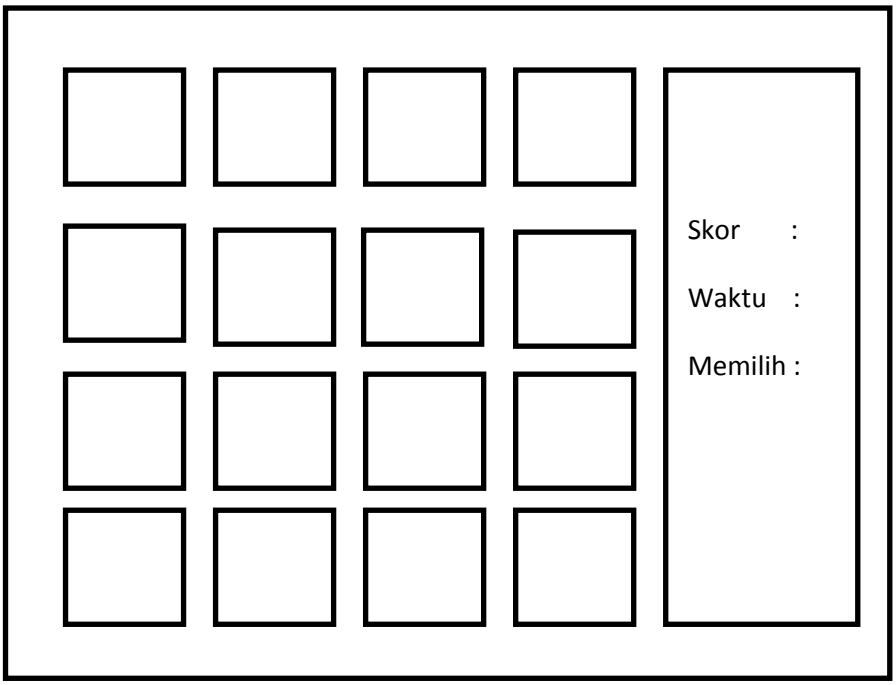
Berikut ini adalah detail *story board* pada *game Endemic Indonesia Island* dengan *gameplay* episode yang berbeda.

A. *Story Board* Pertama yaitu Episode Pulau Sumatera

1. Gambar menampilkan judul episode, *Skor,Waktu* dan *Memilih* dalam episode *game* ini.

2. Deretan 16 gambar yang berpasangan berbentuk persegi panjang yang berisi gambar hewan endemik pulau sumatera yang telah diacak isinya untuk kemudian dipasangkan dengan gambar yang sama berikut dengan nama hewannya untuk menguji memori.

Berikut ini adalah gambar *story board game* episode sumatera dapat dilihat pada gambar 3.38.

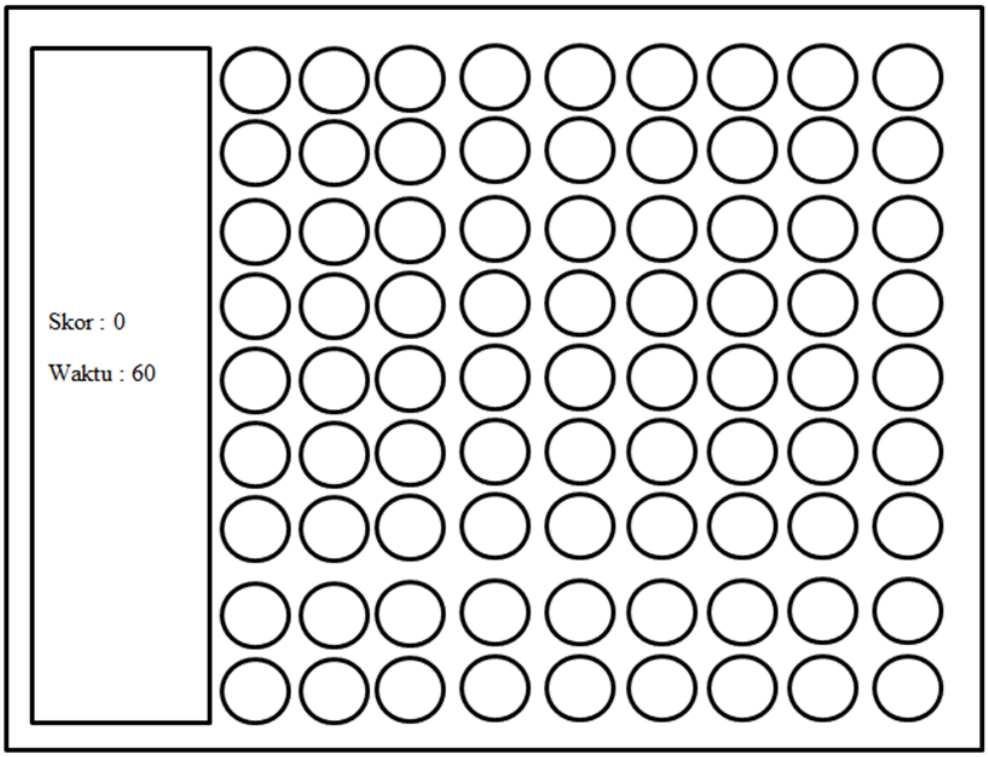


Gambar 3.38 *Story Board* Episode *Game* Sumatera

B. *Story Board* Kedua yaitu Episode Pulau Jawa

1. Gambar menampilkan judul episode *game* , *Waktu* dan *Skor* pada *game* ini.
2. Gambar empat *shape* bulat untuk masing – masing gambar hewan endemik pulau Jawa yang berbeda, *shape* bulat ini adalah hewan endemik, untuk memasangkan gambar – gambar yang sama baik secara vertikal atau horizontal minimal 3 gambar yang sama.

Berikut ini adalah gambar *story board* episode *game* jawa dapat dilihat pada gambar 3.39.

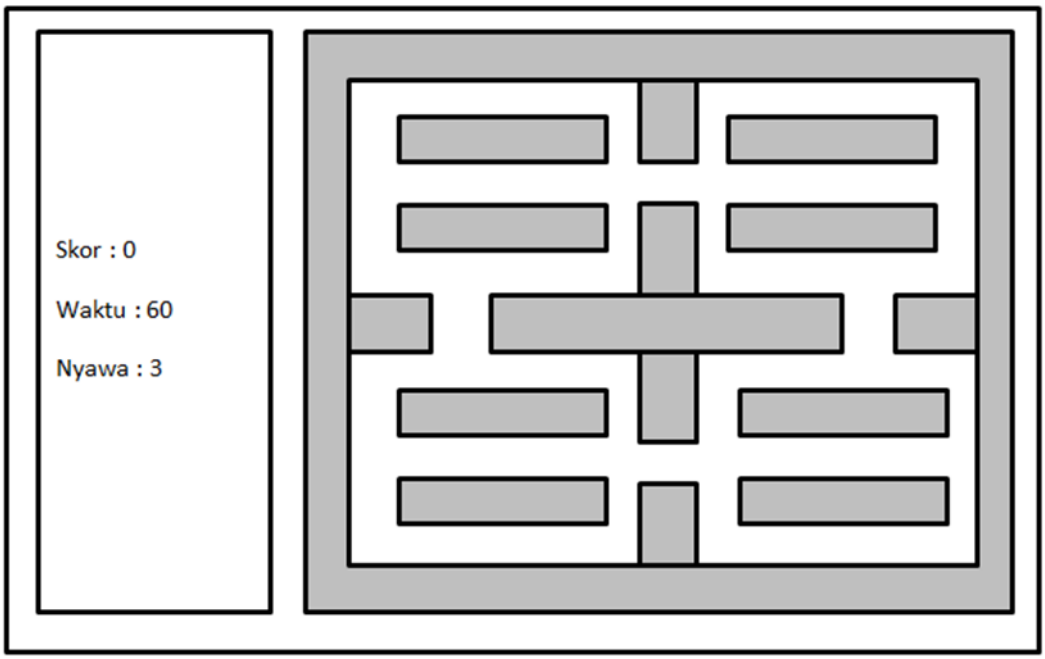


Gambar 3.39 *Story Board* Episode *Game* Jawa

C. *Story Board* Ketiga yaitu Episode Pulau Kalimantan

1. Gambar menampilkan judul episode *game* skor, waktu dan nyawa pada *game* ini.
2. Gambar menampilkan bentuk *maze* atau labirin untuk *game* pada episode ini, serta terdapat karakter pemburu dan hewan endemik, pemburu sebagai NPC/*enemy*, hewan endemik sebagai *player*. Pemburu yang berperan sebagai musuh yang mengejar hewan endemik yaitu bekantan.

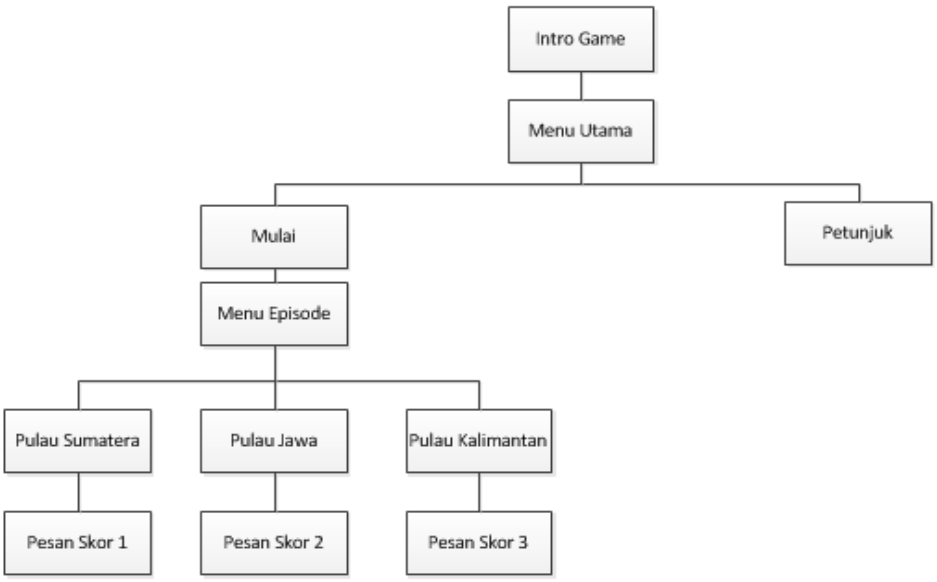
Berikut ini adalah *story board* dari *game* episode Kalimantan dapat dilihat pada gambar *3.40.*



Gambar 3.40 *Story Board* Episode Game Kalimantan.

3.5.2 Perancangan Struktur Menu

Perancangan struktur menu yaitu untuk memudahkan perancangan sistem pembangunan game. Perancangan struktur menu program ini membantu dalam merancang bagian-bagian sistem untuk mengetahui bagaian mana yang terlebih dulu diakses setelah *game* tersebut selesai. Berikut perancangan struktur menu dari sistem yang akan dibangun dapat dilihat pada gambar 3.41 di bawah ini :



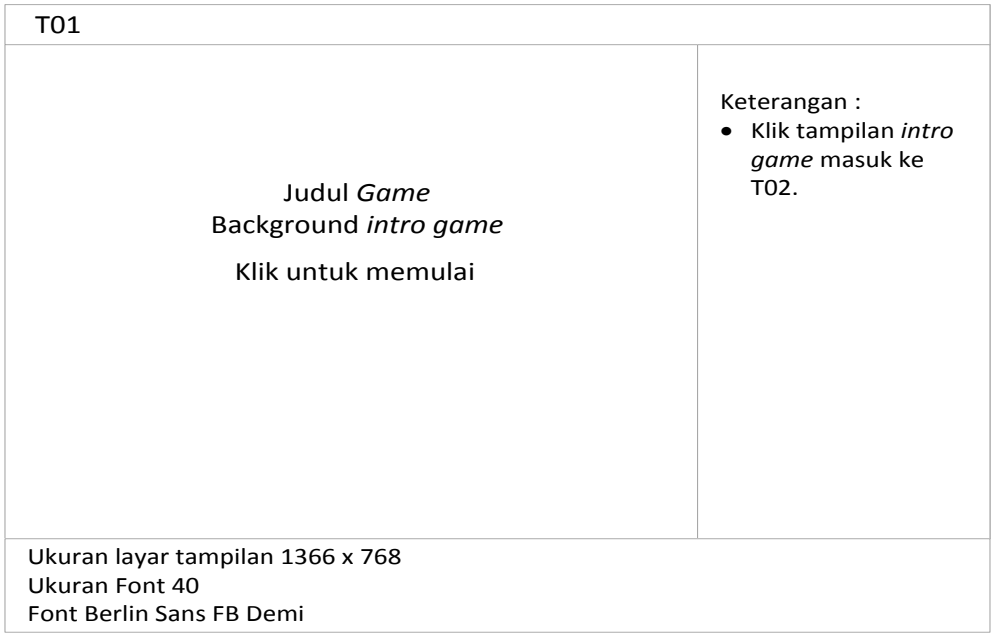
Gambar 3.41 Struktur Menu *Game Endemic Indonesia Island*

3.5.3 Perancangan Antarmuka (*Interface*)

Perancangan antarmuka adalah merancang atau mendesain antarmuka (*interface*) yang efektif untuk pembuatan sistem program aplikasi yang akan di implementasikan menjadi sebuah perangkat lunak (*software*). Antarmuka (*interface*) adalah sistem yang dirancang untuk mengolah *input* atau *output* dari data. Tujuan antarmuka (*interface*) adalah untuk mengkomunikasikan fitur-fitur sistem yang tersedia agar user mengerti dan dapat menggunakan sistem tersebut.

1. Perancangan Antarmuka *Intro Game*

   Berikut ini adalah perancangan antarmuka *Intro game*.



Gambar 3.42 Antarmuka *Intro Game*

2. Perancangan Antarmuka Menu Utama *Game*.

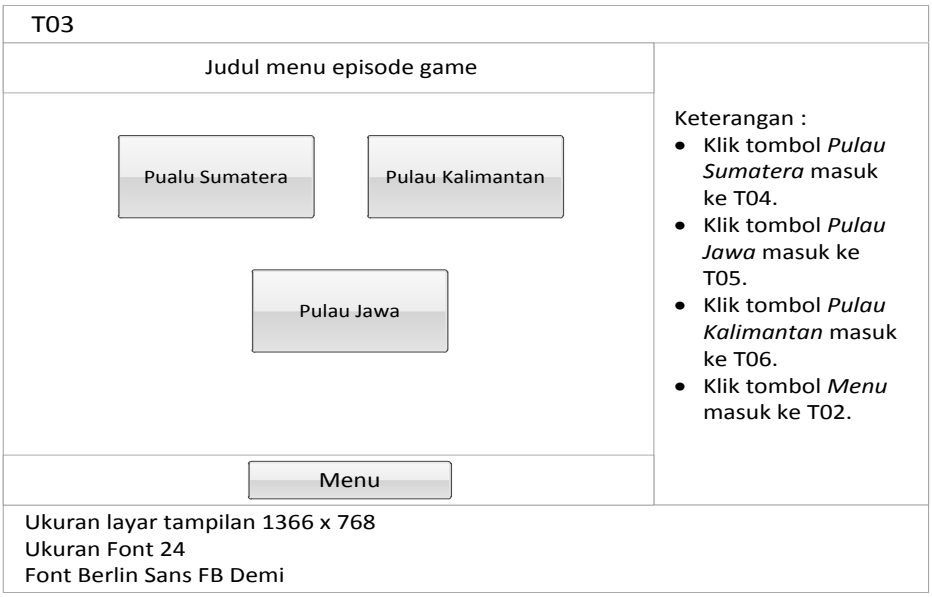
   Berikut ini adalah perancangan antarmuka menu utama *game*.



Gambar 3.43 Antarmuka Menu Utama *Game*

3. Perancangan Antarmuka Pilihan Episode *Game*

Berikut ini adalah perancangan antarmuka pilihan episode *game*.



Gambar 3.44 Antarmuka Pilihan Episode *Game*

4. Perancangan Antarmuka *Game* Pulau Sumatera

   Berikut ini adalah perancangan antarmuka episode *game* pulau sumatera.



Gambar 3.45 Antarmuka *Game* Pulau Sumatera

5. Perancangan Antarmuka *Game* Pulau Jawa

   Berikut ini adalah perancangan antarmuka episode *game* pulau jawa.



Gambar 3.46 Antarmuka *Game* Pulau Jawa

6. Perancangan Antarmuka *Game* Pulau Kalimantan

   Berikut ini adalah perancangan antarmuka episode *game* pulau kalimantan.



Gambar 3.47 Antarmuka *Game* Pulau Kalimantan

7. Perancangan Antarmuka Petunjuk Permainan

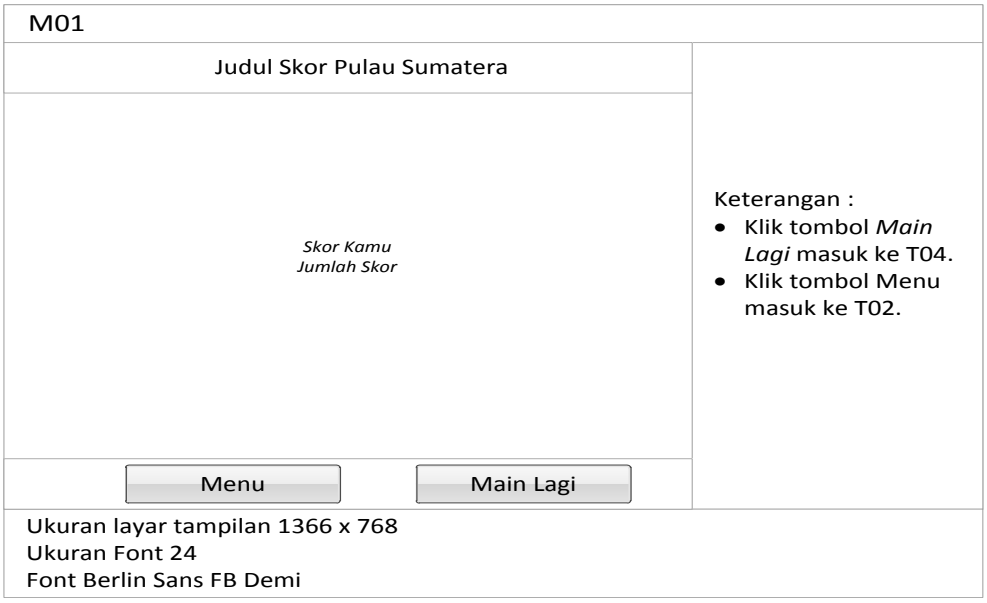
   Berikut ini adalah perancangan antarmuka petunjuk permainan.



Gambar 3.48 Antarmuka Petunjuk Permainan

8. Perancangan Antarmuka Pesan Skor Pulau Sumatera

Berikut ini adalah perancangan antarmuka pesan skor pulau sumatera.



Gambar 3.49 Antarmuka Pesan Skor Pulau Sumatera

9. Perancangan Antarmuka Pesan Skor Pulau Jawa

   Berikut ini adalah perancangan antarmuka pesan skor pulau jawa.



Gambar 3.50 Antarmuka Pesan Skor Pulau Jawa

10. Perancangan Antarmuka Pesan Skor Pulau Kalimantan

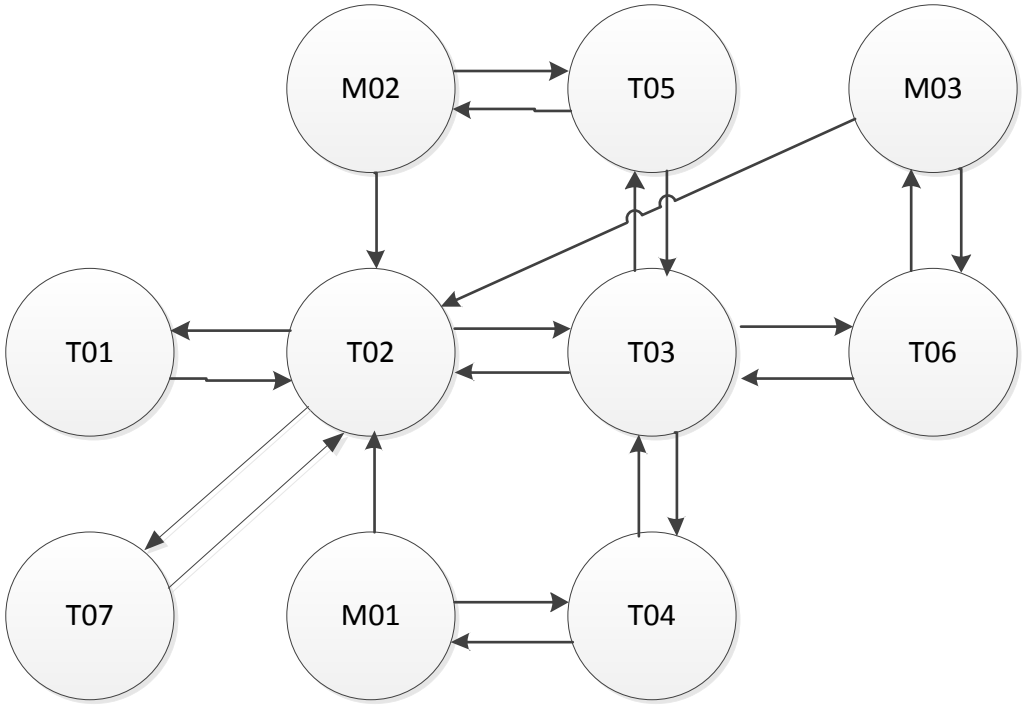
    Berikut ini adalah perancangan antarmuka pesan skor pulau kalimantan.



Gambar 3.51 Antarmuka Pesan Skor Pulau Kalimantan.

3.5.4 Perancangan Jaringan Semantik

Jaringan semantik adalah gambaran pengetahuan grafis yang menunjukkan hubungan antar berbagai objek, terdiri dari lingkaran-lingkaran yang dihubungkan dengan anak panah yang menunjukkan objek dan informasi tentang objek-objek tersebut. Berikut ini adalah jaringan semantik pad setiap episode *game endemic Indonesia Island*.



Gambar 3.52 Jaringan Semantik

Keterangan :

T01 : Antarmuka *Intro Game Endemic Indonesia Island*

T02 : Antarmuka Menu Utama *Game*

T03 : Antarmuka Menu Pilihan Epsiode *Game*

T04 : Antarmuka *Game* Pulau Sumatera

T05 : Antarmuka *Game* Pulau Jawa

T06 : Antarmuka *Game Pulau* Kalimantan

T07 : Antarmuka Petunjuk Permainan

M01 : Antarmuka Pesan Skor *Game* Sumatera

M02 : Antarmuka Pesan Skor *Game* Jawa

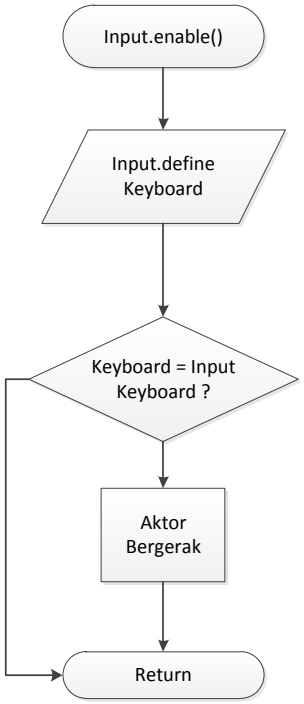
M03 : Antarmuka Pesan Skor *Game* Kalimantan

3.5.5 Perancangan Method

Perancangan komponen *method* merupakan perancangan yang dibuat perancangan struktur menu, antarmuka, pesan, dan jaringan semantik. Perancangan setelah ini berfungsi untuk mendeskripsikan *method-method* yang berada di dalam aplikasi permainan *Game Endemic Indonesia Island*. Adapun *method-method* yang terdapat dalam aplikasi permainan *Game Endemic Indonesia Island* yang akan dibangun adalah sebagai berikut :

A. Method MyAssets()

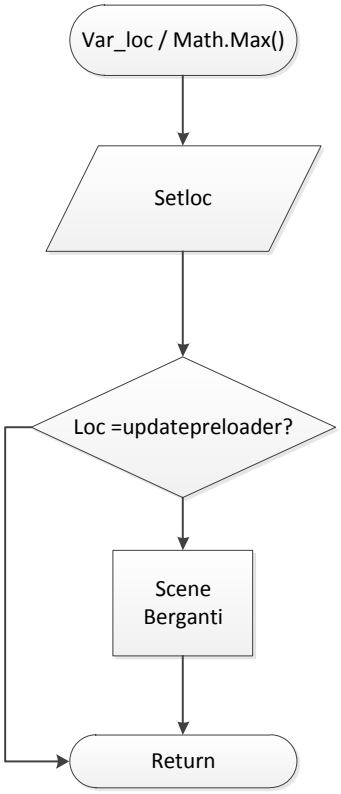
Method MyAssets() akan dipanggil ketika akan memainkan kontrol permainan.



Gambar 3.53 *Flowchart Method* MyAssets()

B. Method loadScenes()

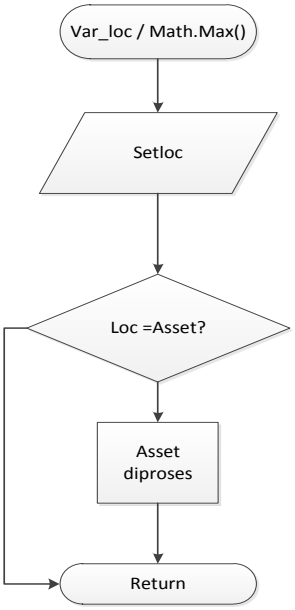
Method loadScenes() akan dipanggil ketika akan menampilkan menu seluruh *scene* permainan.



Gambar 3.54 *Flowchart* Method loadScenes().

C. Method loadResources()

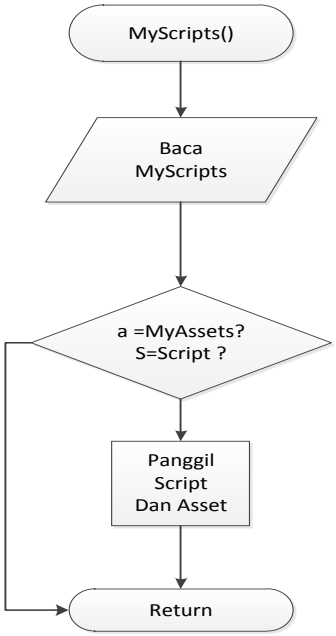
Method loadResources() akan dipanggil ketika akan menampilkan *resource* seluruh permainan.



Gambar 3.55 *Flowchart* Method loadResources()

D. Method MyScripts()

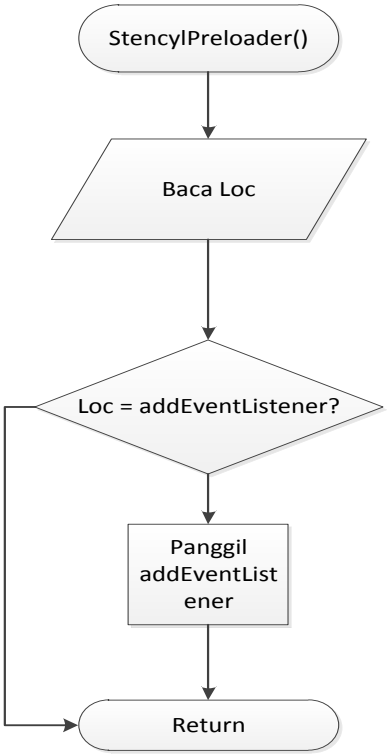
Method MyScripts() akan dipanggil ketika akan memanggil seluruh *script* yang berhubungan dengan method MyAssets() dan MyScripts().



Gambar 3.56 *Flowchart* Method MyScripts().

E. Method StencylPreloader()

Method StencylPreloader() akan dipanggil ketika akan menampilkan preloader permainan yaitu sebelum masuk ke permainan.



Gambar 3.57 *Flowchart* Method StencylPreloader().