BAB 4

IMPLEMENTASI DAN PENGUJIAN

Pada Bab ini akan dilakukan implementasi dan pengujian terhadap aplikasi *game Death Castle* yang akan dibangun. Tahapan ini dilakukan setelah analisis dan perancangan selesai dilakukan dan selanjutnya akan diimplementasikan kedalam bahasa pemrograman. Setelah tahap implementasi akan dilakukan pengujian untuk mengetahui apakah sistem yang dibangun akan berjalan sesuai dengan tujuannya.

4.1 Implementasi

Tahap implementasi merupakan tahap penciptaan perangkat lunak kedalam bahasa pemrograman yang dimengerti oleh komputer, serta merupakan kelanjutan dari tahap kegiatan perancangan aplikasi, tahap ini merupakan tahap dimana aplikasi siap untuk dioperasikan, yang terdiri dari penjelasan mengenai lingkungan implementasi, dan implementasi program.

4.1.1 Implementasi Perangkat Keras

Perangkat keras Perangkat keras yang akan digunakan untuk menjalankan aplikasi permainan edukasi kolaboratif ini tidak harus komputer yang berspesifikasi tinggi. Hal disebabkan oleh penggunaan *resource* yang sedikit pada CPU dan tidak terlalu memakai *resource* grafis yang terlalu banyak. Sebagai ilustrasi sistem minimum yang dapat menjalankan aplikasi adalah sebagai berikut :

1. *Processor* dengan kecepatan ~ 1GHZ
2. RAM dengan kapasitas 512 MB
3. VGA *onboard*
4. *Harddisk* 40 GB
5. Monitor dengan resolusi 1024 X 768
6. *Mouse*
7. *Keyboard*

4.1.2 Implementasi Perangkat Lunak

Adapun spesifikasi perangkat lunak yang dibutuhkan untuk menjalankan aplikasi *game Death Castle* adalah sebagai berikut :

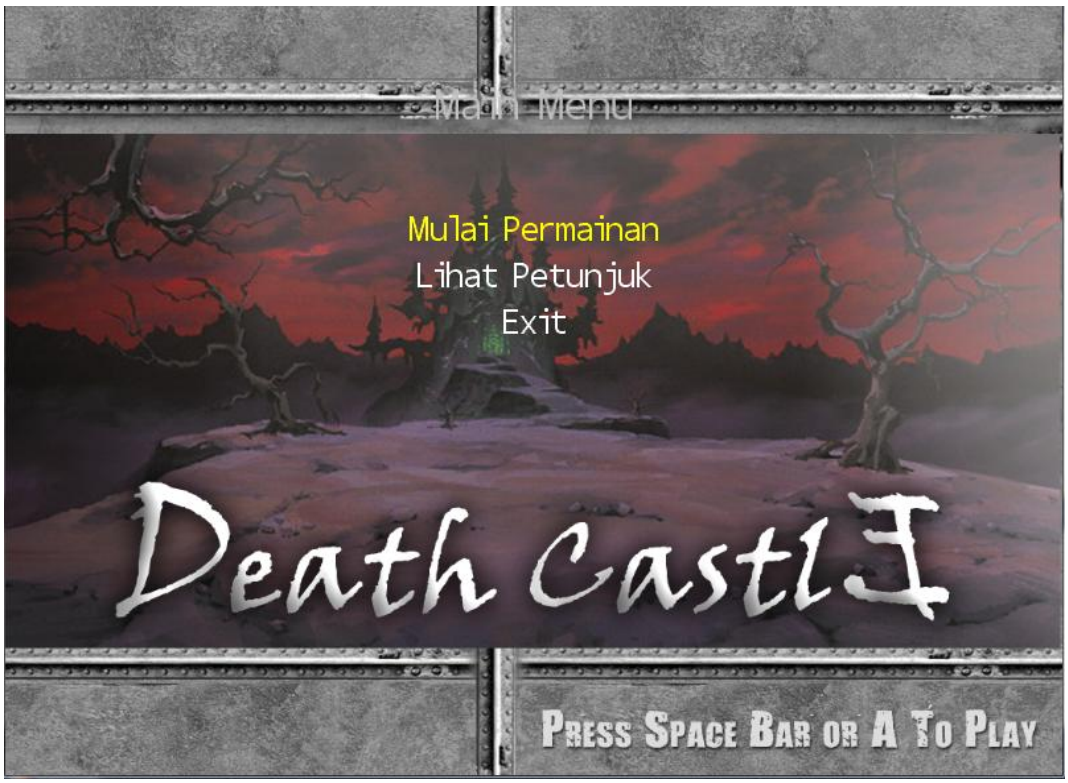
1. Sistem Operasi : Microsoft XP
2. Framework : .NET Framework 3.5 SP1

4.1.3 Implementasi Aplikasi *Game*

Aplikasi ini tidak perlu dilakukan instalasi, cukup dengan melakukan *doubel-klik* pada aplikasi ini. Akan tetapi harus diingat, pada sistem operasi Windows harus terdapat .NET Framework 3.5 SP1, supaya aplikasi dapat berjalan, apabila .NET Framework 3.5 SP1 belum terpasang, maka akan terdapat pesan pada sistem yang mengindikasikan user harus mendownload/memasang software tambahan ini.

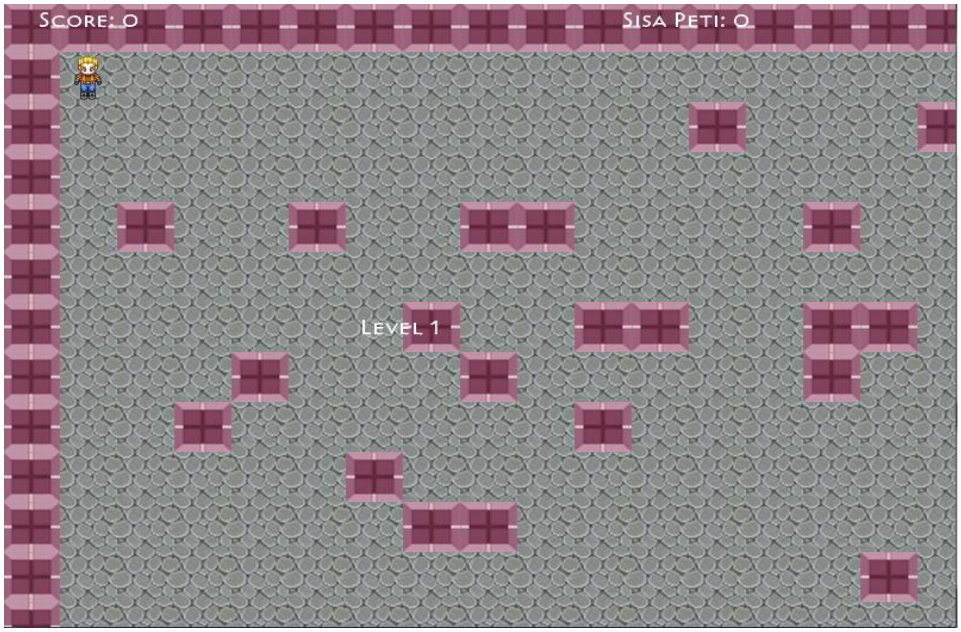
4.1.4 Implementasi Antarmuka

Saat pengguna membuka aplikasi pertama kalinya, maka antarmuka yang yang ditampilkan oleh sistem adalah antarmuka menu utama seperti pada gambar 4.1 dibawah ini :



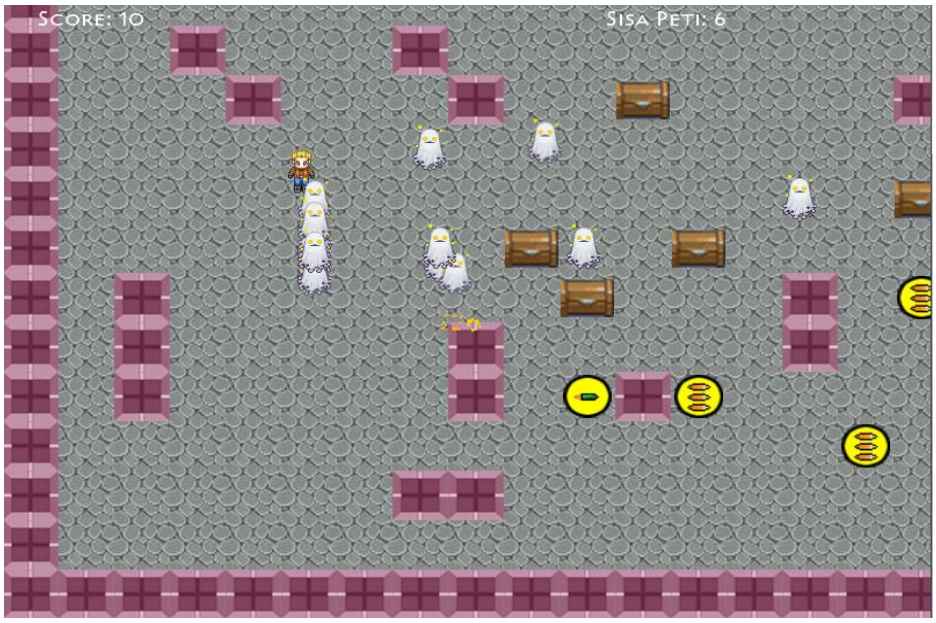
Gambar 4.1 Tampilan Menu utama *game Death Castle*.

Jika pengguna ingin langsung bermain, cukup dengan menekan tombol enter untuk memilih menu mulai permainan baru dan tampilannya akan terlihat seperti gambar 4.2 dibawah ini :



Gambar 4.2 Tampilan Menu mulai permainan *game Death Castle*.

Adapun interaksi pengguna dengan karakter pemain adalah mengontrol pergerakan karakter pemain, menembak musuh, mengambil *item* dan menutup peti seperti terlihat pada gambar 4.3 dibawah ini :



Gambar 4.3 Interaksi pengguna dengan aplikasi *game Death Castle*.

4.2 Pengujian Sistem

Pada umunya ada 2 (dua) macam teknik pengujian yaitu *white box testing* dan *black box testing*. Pengujian yang menggunakan teknik *white box testing* lebih menitikberatkan pada struktur internal (*source code*) program, di mana setiap baris kode yang akan di *compile* dicek satu persatu untuk mengetahui apakah masih terjadi *error*. Pengujian pada *black box testing* tidak terlalu memperhatikan struktur internal (*source code*) dari program, tetapi lebih kepada menemukan kesalahan pada program. Pada Aplikasi *game Death Castle* digunakan pengujian *black box testing*. Cara pengujian dilakukan melalui dua tahap, yaitu pengujian alpha (fungsional) dan pengujian beta.

4.2.1 Pengujian alpha

Pengujian dilakukan terhadap aplikasi untuk memastikan bahwa aplikasi dapat berjalan dengan benar sesuai dengan kebutuhan dan tujuan yang diharapkan. Proses pengujian terhadap aplikasi akan menggunakan *black box testing*.

4.2.1.1 Skenario Pengujian Aplikasi

Skenario pengujian menjelaskan pengujian terhadap sistem yang ada pada aplikasi *game Death Castle*

Berikut ini merupakan skenario dari pengujian menu utama yang dapat dilihat pada tabel 4.1 :

Tabel 4.1 Skenario Pengujian

No	Kelas Uji	Butir Uji	Jenis Pengujian
1.	Menu Utama	Tampilkan menu mulai permainan	*Black box*
		Tampilkan menu lihat petunjuk	*Black box*
		Tampilkan menu kredit	*Black box*
2.	Tombol kontrol pemain	Bergerak keatas	*Black box*
		Bergerak kebawah	*Black box*
		Bergerak kekanan	*Black box*
		Bergerak kekiri	*Black box*
3.	Menu Pause	Menampilkan menu pause	*Black box*
		Lanjutkan permainan	*Black box*
		Keluar permainan	*Black box*
4.	Menu Lihat Petunjuk	Memilih menu lihat petunjuk	*Black box*
		Menampilkan petunjuk	*Black box*
		Memilih *back*/kembali	*Black box*
5.	Menu Perhitungan A*	Menampilkan nilai node awal, node tujuan dan nilai cost.	*Black box*
6.	*Collision Detection*	*Collision Detection* terhadap objek pemain	*White Box*
		Collision Detection terhadap objek musuh	
7.	Implementasi Algoritma A* terhadap objek musuh	Implementasi Algoritma A* terhadap objek musuh dalam melakukan pencarian karakter pemain di arena permainan	*White Box*

4.2.1.2 Butir Pengujian

Pengujian ini dilakukan secara *black box* yaitu pengujian yang dilakukan dengan hanya memperhatikan masukan ke sistem dan respon sistem.

1. Pengujian Menu Utama

Pengujian menu utama merupakan proses dimana pengguna menyentuh menu-menu yang ada pada tampilan menu utama. Hasil dari proses pengujian dapat dilihat pada tabel 4.2 :

Tabel 4.2 Pengujian Menu Utama

Kasus dan Hasil Uji			
Masukan	**Yang Diharapkan**	**Pengamatan**	**Kesimpulan**
Pilih Menu Mulai Permainan	Sistem melakukan proses buka halaman menu Mulai Permainan	Sistem menampilkan halaman menu permainan	[√] Berhasil [ ] Tidak Berhasil
Pilih Menu Lihat Petunjuk	Sistem melakukan proses buka halaman menu Lihat petunjuk	Sistem menampilkan halaman menu Lihat petunjuk	[√] Berhasil [ ] Tidak Berhasil
Sentuh Menu Kredit	Sistem melakukan proses buka halaman menu kredit	Sistem menampilkan halaman menu kredit	[√] Berhasil [ ] Tidak Berhasil

2. Pengujian Tombol kontrol pemain

Pengujian tombol kontrol pemain merupakan proses yang dilakukan oleh pemain untuk mengerakkan karakter pemain diarena permainan dapat dilihat pada tabel 4.3 :

Tabel 4.3 Pengujian Tombol kontrol pemain

Kasus dan Hasil Uji			
Masukan	**Yang Diharapkan**	**Pengamatan**	**Kesimpulan**
Menekan tombol W	Karakter pemain akan bergerak kearah atas	Karakter pemain bergerak kearah atas	[√] Berhasil [ ] Tidak Berhasil
Menekan tombol S	Karakter pemain akan bergerak kearah bawah	Karakter pemain bergerak kearah bawah	[√] Berhasil [ ] Tidak Berhasil
Menekan tombol A	Karakter pemain akan bergerak kearah kiri	Karakter pemain bergerak kearah kiri	[√] Berhasil [ ] Tidak Berhasil
Menekan tombol D	Karakter pemain akan bergerak keanan	Karakter pemain bergerak kearah kanan	[√] Berhasil [ ] Tidak Berhasil

3. Pengujian Menu Pause

Pengujian menu utama merupakan proses dimana pengguna menekan tombol esc untuk melakukan jeda permainan.Hasil dari proses pengujian dapat dilihat pada tabel 4.4 :

Tabel 4.4 Pengujian Menu *Pause*

Kasus dan Hasil Uji			
Masukan	**Yang Diharapkan**	**Pengamatan**	**Kesimpulan**
Menekan tombol esc	Menampilkan menu pause ketika permainan telah berlangsung	Sistem Menampilkan menu pause	[√] Berhasil [ ] Tidak Berhasil
Memilih lanjut permainan	Kembali ke layar permainan	Sistem menampilkan kembali layar permainan	[√] Berhasil [ ] Tidak Berhasil
Menekan tombol A	Karakter pemain akan bergerak kearah kiri	Karakter pemain bergerak kearah kiri	[√] Berhasil [ ] Tidak Berhasil

4. Pengujian Menu Lihat Petunjuk

Pengujian menu lihat petunjuk merupakan proses dimana pengguna memilih menu lihat petunjuk pada menu utama. Hasil dari proses pengujian dapat dilihat pada tabel 4.5 :

Tabel 4.5 Pengujian Menu Lihat Petunjuk

Kasus dan Hasil Uji			
Masukan	**Yang Diharapkan**	**Pengamatan**	**Kesimpulan**
Memilih Menu lihat petunjuk	Sistem melakukan proses buka halaman menu lihat petunjuk	Sistem menampilkan informasi cara bermain	[√] Berhasil [ ] Tidak Berhasil
memilih back	Sistem melakukan proses kembali menu utama	Sistem menampilkan menu utama	[√] Berhasil [ ] Tidak Berhasil

5. Pengujian Menu Perhitungan Algoritma A*

Pengujian menu perhitungan Algoritma A* dilakukan untuk melihat performansi pada algoritma A* yang diterapkan pada objek musuh untuk melakukan pencarian objek pemain diarena arena permainan. Berikut ini adalah pengujian algoritma A* yang diterapkan pada objek musuh :

Tabel 4.6 Pengujian Menu Perhitungan A*

Kasus dan Hasil Uji			
Masukan	**Yang Diharapkan**	**Pengamatan**	**Kesimpulan**
Memilih Menu Perhitungan A*	Sistem menampilkan *node awal*, *node* tujuan dan nilai *cost* dari perhitungan A*	Sistem dapat menampilkan *node* awal, *node* tujuan dan nilai *cost* dari hasil perhitungan A*	[√] Berhasil [ ] Tidak Berhasil



Gambar 4.4 Pengujian Algoritma A*

Hasil pengujian performansi Algoritma A* pada gambar 4.4 dengan lima objek musuh dengan *node* tujuan atau posisi pemain pada koordinat(10,17) dapat dilihat pada tabel 4.7 berikut ini :

Tabel 4.7 Pengujian Perhitungan A* pada objek musuh

Musuh	Node Awal	Node Tujuan	Nilai Cost
Musuh 1	(4,12)	(3,12)	10
Musuh 2	(8,7)	(3,12)	16.40132
Musuh 3	(9,9)	(3,12)	15.83095
Musuh 4	(9,13)	(3,12)	15.09902
Musuh 5	(4,13)	(3,12)	11

6. *Collision Detection*

Pengujian *collision detection* dilakukan untuk melihat apakah metode *collision detection* yang diterapkan sudah berjalan sesuai dengan yang diharapkan. Berikut ini adalah pengujian *Collision Detection* terhadap objek pemain dan objek musuh :

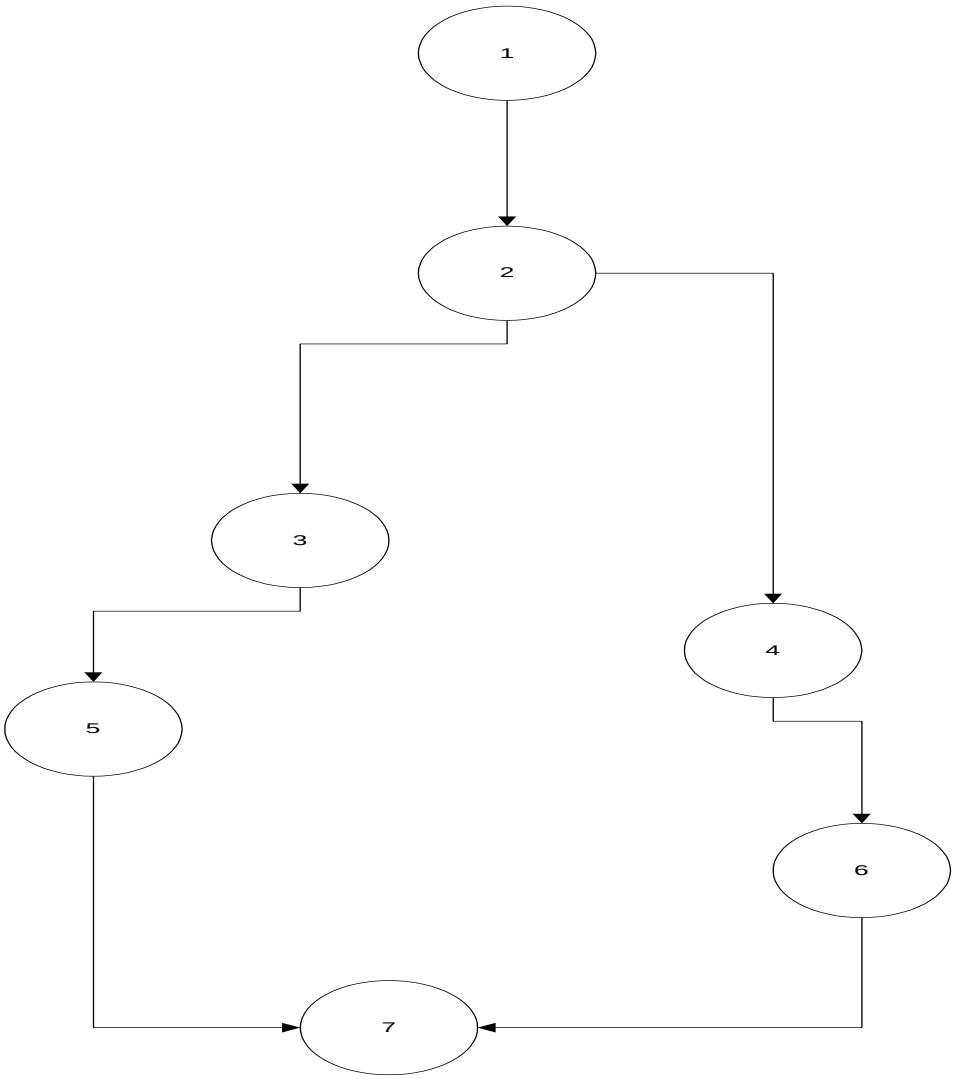
i). *Collision Detection* pada Objek Pemain

Berikut ini merupakan algoritma dari metode *Collision Detection* pada objek pemain yang penerapannya menggunakan bahasa pemrograman *C#* :

```
#region Collision Detection Methods
        public bool IsBoxColliding(Rectangle OtherBox)
        {
 1.           if ((Collidable) && (!Expired))
             {
                 return BoundingBoxRect.Intersects(OtherBox);
             }
 2.          else
             {
 3.              return false;
             }
        }
        public bool IsCircleColliding(
        Vector2 otherCenter,
        float otherRadius)
        {
 4.           if ((Collidable) && (!Expired))
             {
 5.              if (Vector2.Distance(WorldCenter, otherCenter) <
                (CollisionRadius + otherRadius))
                    return true;
 6.                else
                   return false;
             }
 7.          else
             {
                 return false;
             }
        }

 #endregion
```

b. *Flowgraph Collision Detection* pada Objek Pemain



Gambar 4.5 *Flowgraph collision detection* pada objek pemain

c. *Cyclomatic Complexity* V(G)

E = 7, N = 7

V(G) = E – N + 2

V(G) = 7 – 7 + 2

V(G) = 0 + 2 = 2

Keterangan :

E = Jumlah aksi

N = Jumlah Kondisi

d. Independent Path

Path 1 = (1) – (2) – (3) – (5) – (7)

Path 2 = (1) – (2) – (4) – (6) – (7)

e. Graph Matriks

Tabel 4.8 *Graph* Matriks *Collision Detection*

	1	2	3	4	5	6	7
1		**1**					
2			**1**	**1**			
3					**1**		
4						**1**	
5							**1**
6							**1**
7							

Keterangan :

1. Baris dan kolom merepresentasikan simpul
2. Nilai 1 merepresentasikan adanya keterhubungan antar simpul

f. Hasil Uji penerapan *Collision Detection* pada objek pemain

Tabel 4.9 Hasil Uji penerapan *Collision Detection* pada objek pemain

Kasus dan Hasil Uji				
No. *Path*	**Node (n), input kondisi**	**Yang Diharapkan**	**Pengamatan**	**Kesimpulan**
1	1(*void*) 2(*void*) 4(*void*) 6(*true)* 7(*false*)	Pemain menabrak musuh	Pemain mati ketika menabrak musuh	*Path* dikerjakan
2	1(*void*) 2(*void*) 3(*void*) 5(*true*) 7(*true*)	Pemain menabrak item	Pemain menabrak item lalu kemudian peluru item akan di *upgrade*	*Path* dikerjakan

Dari hasil pengujian diatas yang dilakukan, maka didapat kesimpulan bahwa seluruh *node* yang ada pada setiap *independent path* telah dikerjakan, serta tidak terjadi pengulangan tak berhingga.

6. Pengujian Algoritma A*

Pengujian algoritma A* merupakan proses dimana akan diuji terhadap implementasi A* terhadap objek musuh dalam melakukan pencarian karakter Berikut ini merupakan algoritma A* yang penerapannya menggunakan bahasa pemrograman *C#* :

```
#region AI  Methods
endNode = new PathNode(null, null, endTile, 0);
            startNode = new PathNode(null, endNode, startTile, 0);

            addNodeToOpenList(startNode);

            while (openList.Count > 0)
```

```
            {
                PathNode currentNode = openList[openList.Count -
1];

                if (currentNode.IsEqualToNode(endNode))
                {
                    List<Vector2> bestPath = new List<Vector2>();

                    while (currentNode != null)
                    {
                        bestPath.Insert(0,
currentNode.GridLocation);
                        currentNode = currentNode.ParentNode;
                    }

                    return bestPath;
                }

                openList.Remove(currentNode);
                nodeCosts.Remove(currentNode.GridLocation);

                foreach (PathNode possibleNode in
findAdjacentNodes(currentNode, endNode))
                {
                    if
(nodeStatus.ContainsKey(possibleNode.GridLocation))
                    {
                        if (nodeStatus[possibleNode.GridLocation]
== NodeStatus.Closed)
                        {
                            continue;
                        }

                        if (nodeStatus[possibleNode.GridLocation]
== NodeStatus.Open)
                        {
                            if (possibleNode.TotalCost >=
nodeCosts[possibleNode.GridLocation])
                            {
                                continue;
                            }
                        }
                    }

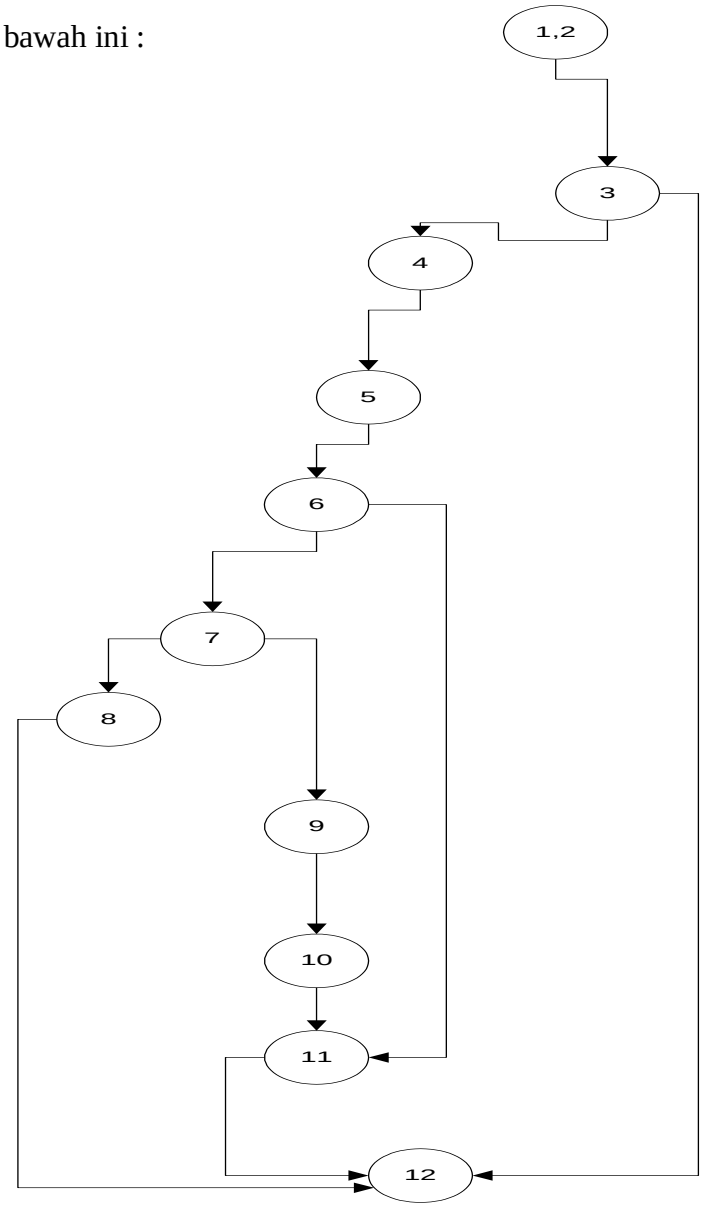
                    addNodeToOpenList(possibleNode);
                }

                nodeStatus[currentNode.GridLocation] =
NodeStatus.Closed;
            }


 #endregion
```

b. *Flowgraph* Algoritma A*

Berikut ini merupakan *flowgraph* dari Algoritma A* yang dapat dilihat pada gambar di bawah ini :



Gambar 4.6 *Flowgraph* Algoritma A*

Keterangan :

: Menggambarkan Kondisi

: Menggambarkan Aksi

c. *Cyclomatic Complexity* V(G)

E = 14, N = 12

V(G) = E – N + 2

V(G) = 13 – 11 + 2

V(G) = 2 + 2 = 4

Keterangan :

E = Jumlah aksi

N = Jumlah Kondisi

d. Independent Path

Path 1 = (1,2) – (3) – (12)

Path 2 = (1,2) – (3) – (4) – (5) – (6) – (7) – (8) – (12)

Path 3 = (1,2) – (3) – (4) – (5) – (6) – (7) – (9) – (10) – (11) – (12)

Path 4 = (1,2) – (3) – (4) – (5) – (6) – (11) – (12)

e. Graph Matriks

Tabel 4.10 Graph Matriks Hasil Uji Algoritma A*

	1,2	3	4	5	6	7	8	9	10	11	12
1,2		**1**									
3			**1**								
4				**1**							
5					**1**						
6						**1**					**1**
7							**1**	**1**			
8											**1**
9									**1**		
10										**1**	
11											**1**
12											

Keterangan :

3. Baris dan kolom merepresentasikan simpul
4. Nilai 1 merepresentasikan adanya keterhubungan antar simpul

f. Hasil Uji penerapan Algoritma A* pada objek musuh

4.11 Hasil Uji penerapan Algoritma A* pada objek musuh

Kasus dan Hasil Uji				
No. *Path*	**Node (n), input kondisi**	**Yang Diharapkan**	**Pengamatan**	**Kesimpulan**
1	1,2(*void*) 3(*false*) 12(*void*)	Rute pemain dapat ditelusuri	Menelusuri rute pemain	*Path* dikerjakan
2	1,2(*void*) 3(*false*) 4,5,6,7(*void*) 8(*true*) 12(*void*)	Rute pemain dapat ditelusuri	Menelusuri rute pemain	*Path* dikerjakan
3	1,2(*void*) 3,(*false*) 4,5,6,7(*true*) 9(*true*) 10,11,12(*void*)	Rute pemain dapat ditelusuri	Menelusuri rute pemain	*Path* dikerjakan
4	1,2(*void*) 3(*false*) 4,5,6(*false*) 11(*true*) 12(*void*)	Rute pemain tidak dapat ditelusuri	Rute pemain tidak ditemukan	*Path* dikerjakan

Dari hasil pengujian diatas yang telah dilakukan, maka didapat kesimpulan bahwa seluruh *node* yang ada pada setiap *independent path* telah dikerjakan, serta tidak terjadi pengulangan tak berhingga.

4.2.1.3 Kesimpulan Pengujian Alpha

Dari hasil pengujian yang dilakukan, dapat disimpulkan bahwa sistem sudah dapat berjalan sesuai dengan yang diharapkan beserta proses penanganan kesalahannya. Jadi, secara fungsional sistem sudah dapat menghasilkan keluaran yang diharapkan.

4.2.2 Pengujian Beta

Pengujian beta merupakan pengujian yang dilakukan secara objektif, dimana dilakukan pengujian secara langsung terhadap pengguna dengan menggunakan kuisioner mengenai kepuasan pengguna atas aplikasi yang telah dibangun. Adapun metode penilaian pengujian yang digunakan adalah metode kuantitatif berdasarkan data dari pengguna.

4.2.2.1 Pertanyaan Pengguna

Adapun pertanyaan yang diajukan dalam kuisioner untuk aplikasi *game Death Castle* adalah sebagai berikut :

1. Apakah tampilan aplikasi permainan ini secara umum sudah baik ?
    a. Sangat Baik
    b. Baik
    c. Cukup Baik
    d. Kurang Baik
    e. Tidak Baik

2. Apakah alur permainannya seru ?

   a. Sangat Seru
   b. Seru
   c. Cukup Seru
   d. Kurang Seru
   e. Membosankan

3. Apakah karakter pemain mudah untuk dikendalikan ?

   a. Sangat Mudah
   b. Mudah
   c. Cukup Mudah
   d. Kurang Mudah
   e. Tidak Mudah

4. Apakah anda menyukai aplikasi permainan ini ?

   a. Sangat Suka
   b. Suka
   c. Cukup Suka
   d. Kurang Suka
   e. Tidak Suka

5. Menurut Anda bagaimana tingkat kesulitan dari permainan ini ?

   a. Sangat Mudah
   b. Mudah
   c. Cukup Mudah

d. Agak Sulit

e. Sulit

4.2.2.2 Rumus Pengujian Beta

Berdasarkan data hasil kuisioner tersebut, dicari prosentase masing-masing jawaban dengan menggunakan rumus R=P/Q*100%,

dimana :

P = Banyaknya jawaban responden tiap soal

Q = Jumlah responden

R = Nilai prosentase

4.2.2.3 hasil Kuisioner

Berikut ini merupakan pengolahan data hasil kuesioner :

1. Apakah tampilan aplikasi permainan ini secara umum sudah baik ?

1.1 Hasil Kuesioner tampilan aplikasi

NO	Jawaban	Responden	Prosentase(%)
1	Sangat Baik	2	6.67
2	Baik	8	26.67
3	Cukup Baik	17	56.67
4	Kurang Baik	3	10
5	Tidak Baik	0	0

2. Apakah alur permainannya seru ?

1.2 Hasil Kuesioner alur permainan

NO	Jawaban	Responden	Prosentase(%)
1	Sangat Seru	3	10
2	Seru	6	20
3	Cukup Seru	13	43.33
4	Kurang Seru	6	20
5	Membosankan	2	6.67

3. Apakah anda menyukai aplikasi permainan ini?

1.3 Hasil Kuesioner tingkat kesukaan terhadap aplikasi

NO	Jawaban	Responden	Prosentase(%)
1	Sangat Suka	1	3.33
2	Suka	7	23.33
3	Cukup Suka	17	56.67
4	Kurang Suka	5	16.67
5	Tidak Suka	0	

4. Apakah karakter pemain mudah untuk dikendalikan?

1.4 Hasil Kuesioner pengendalian karakter pamain

NO	Jawaban	Responden	Prosentase(%)
1	Sangat Mudah	6	20
2	Mudah	12	40
3	Cukup Mudah	8	26.67
4	Kurang Mudah	4	13.33
5	Tidak Mudah	0	0

5. Menurut Anda bagaimana tingkat kesulitan dari permainan ini ?

1.5 Hasil Kuesioner Tingkat kesulitan aplikasi

NO	Jawaban	Responden	Prosentase(%)
1	Sangat Mudah	1	3.33
2	Mudah	10	33.33
3	Cukup Mudah	9	30
4	Agak Sulit	9	30
5	Sulit	1	3.33

4.2.2.4 Kesimpulan Pengujian Beta

Berdasarkan hasil dari pengujian beta dapat disimpulkan bahwa aplikasi yang telah dibangun dapat menumbuhkan ketertarikan pengguna terhadap *game* berjenis *Arcade*.