BAB 4
IMPLEMENTASI DAN PENGUJIAN

Pada Bab ini akan dilakukan implementasi dan pengujian terhadap aplikasi game Survive From Death yang akan dibangun. Tahap ini dilakukan setelah tahap analisis dan tahap perancangan selesai dilakukan yang selanjutnya akan diimplementasikan kedalam bahasa pemograman. Setelah tahap implementasi selesai, maka dilakukan pengujian terhadap aplikasi game Survive From Death.

4.1. Implementasi

Tahap implementasi merupakan tahap penciptaan perangkat lunak kedalam bahasa pemograman yang dimengerti oleh computer, serta merupakan tahap kelanjutan dari kegiatan perancangan sistem.

4.1.1. Implementasi Perangkat Keras

Perangkat keras yang akan digunakan untuk menjalankan aplikasi game Survive From Death ini tidak harus komputer yang berspesifikasi tinggi. Adapun spesifilasi perangkat keras yang digunakan saat implementasi dan pengujian aplikasi game Survive From Death dapat dilihat pada table 4.1

Tabel 4. 1 Spesifikasi perangkat keras

Spesifikasi Perangkat Keras	
Processor	AMD APU A8 3,0Ghz
RAM	Corsair Vengeance 4GB
Harddisk	Seagate 500GB
VGA	HIS Ati Radeon HD 6670 DDR5 2GB
Monitor	Resolusi 1280x720
Mouse	Optic 1 buah
Keyboard	1 buah
Speaker	1 pasang

4.1.2. Implementasi Perangkat Lunak

Spesifikasi perangkat lunak yang dibutuhkan untuk menjalankan aplikasi game Survive From Death yaitu sistem operasi widows XP SP2 dan directX9 untuk mendukung performa grafik untuk menjalankan aplikasi.

4.1.3. Implementasi Aplikasi

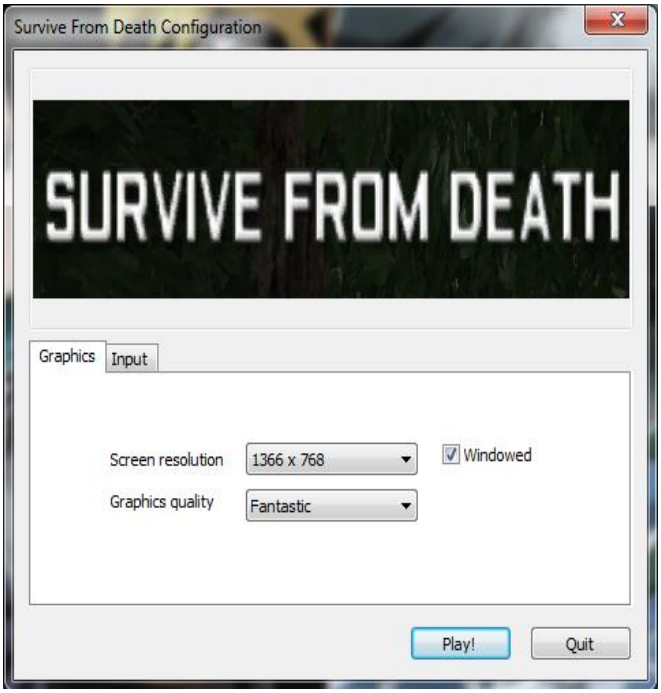
Aplikasi game Survive From Death merupakan sebuah game ber-genre FPS yang terdiri dari beberapa *Level* yang dimana pemain harus menyelesaikan misi disetiap *Level*nya untuk dapat melanjutkan ke*Level-Level* berikutnya. Aplikasi game ini tidak melakukan proses instalasi, cukup dengan melakukan double-klik pada aplikasi ini.

4.1.4. Implementasi Antarmuka

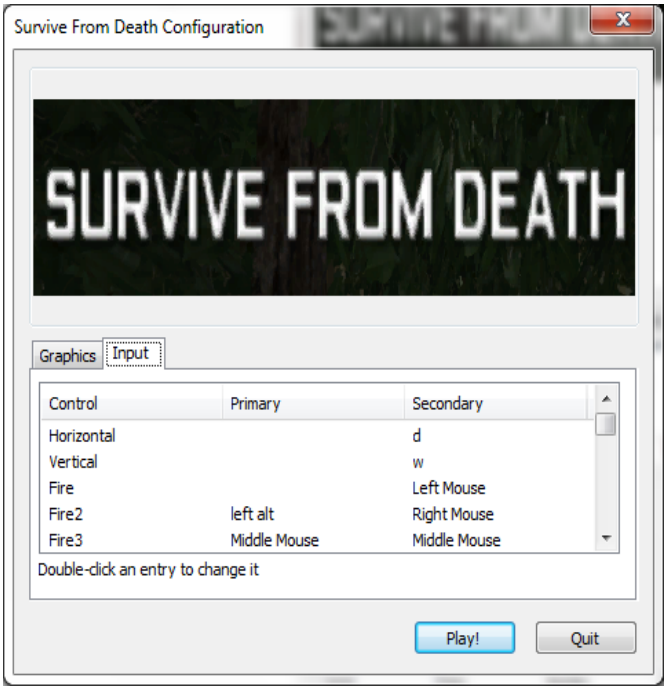
Adapun implementasi antarmuka pada aplikasi game Survive From Death adalah sebagai berikut :

1. Antarmuka Konfigurasi Default

Antarmuka konfigurasi default merupakan tampilan awal pada saat aplikasi dijalankan, tampilan tersebut merupakan tampilan default yang diberikan untuk semua project yang dibangun menggunakan Unity. Terdapat 2 menu untuk mensetting graphic seperti pada gambar 4.1 dan input dapat dilihat pada gambar 4.2 .



Gambar 4. 1 Menu Setting Graphic



Gambar 4. 2 Menu Input

2. Antarmuka Menu Utama

Antarmuka menu utama merupakan tampilan menu utama aplikasi game setelah pengguna mengklik play pada tampilan konfigurasi default. Terdapat 5 menu dapat dilihat pada gambar 4.2.



Gambar 4. 3 Tampilan Antarmuka Menu Utama

Pada tampilan menu utama jika pengguna mengklik menu petunjuk maka akan tampil seperti pada gambar 4.4.



Gambar 4. 4 Tampilan Antarmuka Petunjuk

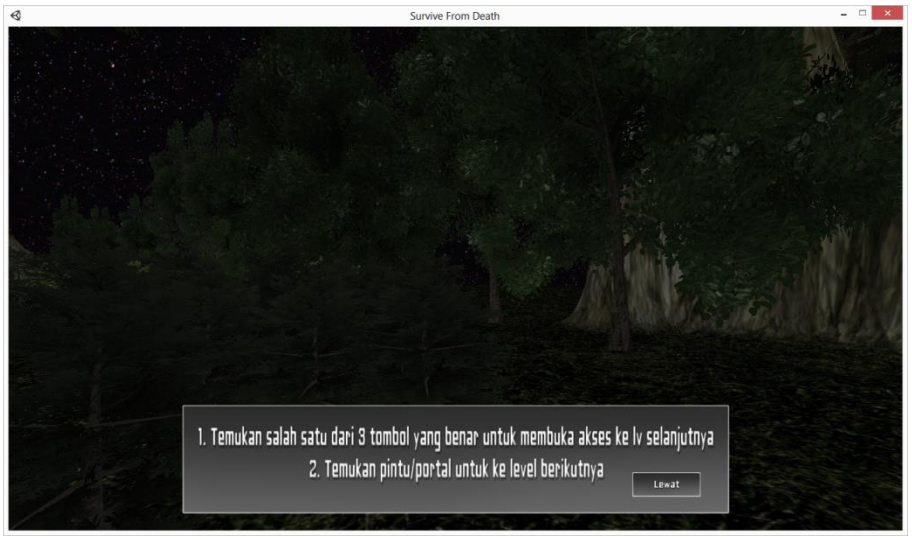
3. Antarmuka Mulai Permainan

Antarmuka mulai permainan merupakan tampilan yang akan ditampilkan ketika pengguna mengklik tombol mulai pada tampilan menu utama, berikut adalah tampilan Intro yang dapat dilihat pada gambar 4.5.



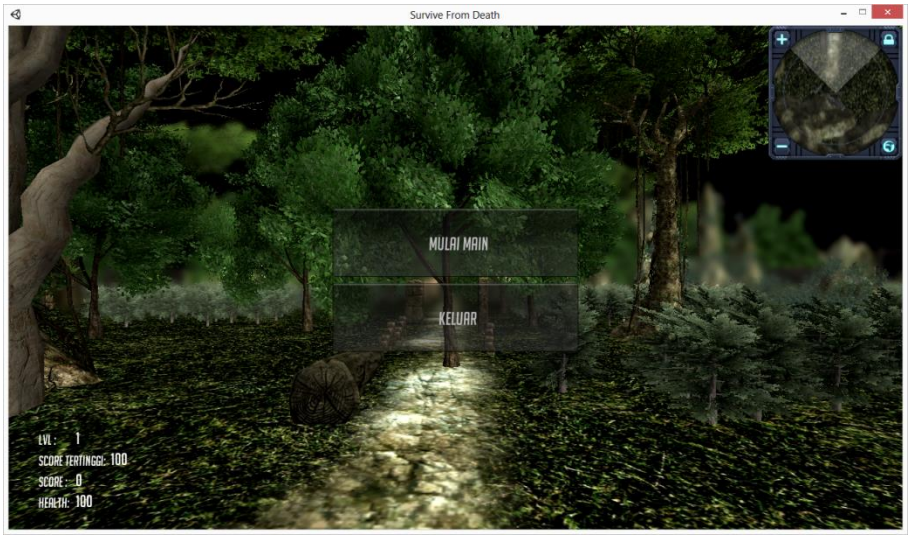
Gambar 4. 5 Tampilan Antarmuka Mulai Permainan Intro

Selanjutnya seletelah melihat intro dan mengklik mulai pemain akan diberikan misi pertamanya pada *Level* 1 yang dapat dilihat pada gambar 4.6.



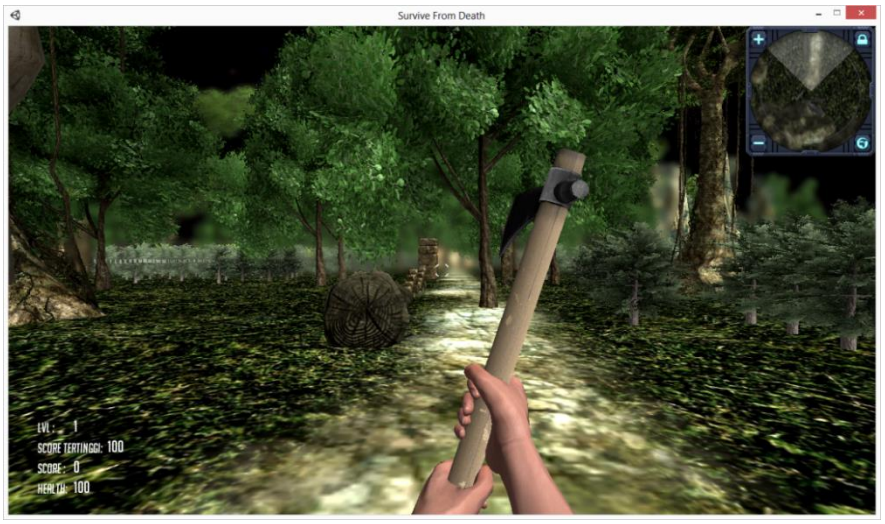
Gambar 4. 6 Tampilan Antarmuka Misi

Berikutnya tampilan pause menu pada aplikasi game, pada pause menu terdapat 2 menu mulai main dan keluar. Pause menu dapat diakses kembali ketika pengguna memencet tombol *Esc* pada *keyboard*, berikut tampilan pause menu yang dapat dilihat pada gambar 4.7.



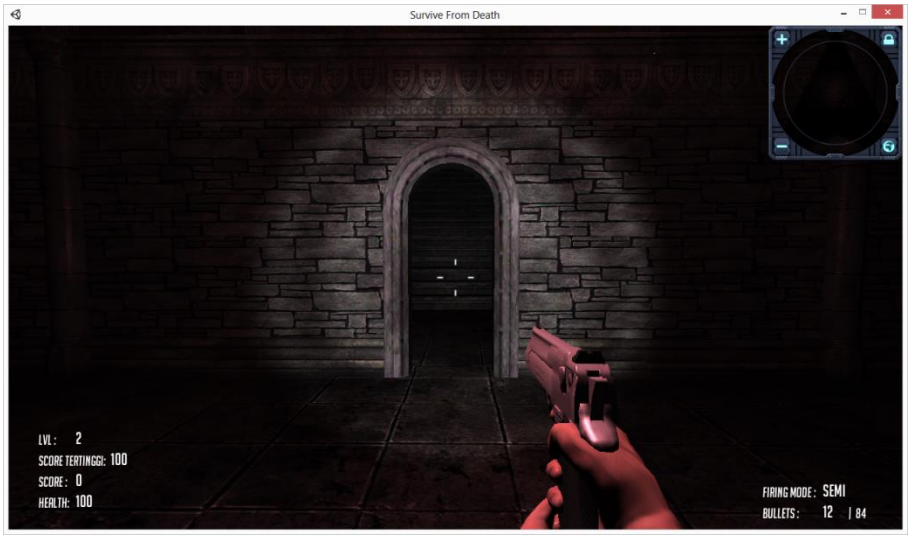
Gambar 4. 7 Tampilan Antarmuka Pause Menu

Selanjutnya pengguna akan berada di*Level* 1 permainan, dimana pengguna dapat mulai menggerakan karakter menggunakan *keyboard* dan *mouse* untuk menyerang dan pada *Level* 1 pengguna berada di hutan. Tampilan *Level* 1 dapat dilihat pada gambar 4.8.



Gambar 4. 8 Tampilan Antarmuka *Level* 1 Permainan

Setelah berhasil menyelesaikan *Level* 1 pengguna dapat melanjukan ke *Level* 2 dan dapat dilihat pada gambar 4.9.



Gambar 4. 9 Tampilan Antarmuka *Level* 2 Permainan

Setelah berhasil menyelesaikan *Level* 2 pengguna dapat melanjukan ke *Level* 3 dan dapat dilihat pada gambar 4.10.



Gambar 4. 10 Tampilan Antarmuka *Level* 3 Permainan

Setelah berhasil menyelesaikan *Level* 3 pengguna dapat melanjukan ke *Level* 4 dan dapat dilihat pada gambar 4.11.



Gambar 4. 11 Tampilan Antarmuka *Level* 4 Permainan

4.2. Pengujian

Pengujian dilakukan betujuan untuk menemukan kesalahan kekurangan pada perangkat lunak yang diuji. Pengujian tersebut dimaksudkan untuk mengetahui perangkat lunak yang dibuat sudah memenuhi kriteria yang sesuai dengan tujuan perancangan perangkat lunak.

Pengujian dilakukan dengan dua macam teknik yaitu pengujian *white box* dan Pengujian *black box*. Adapun pengujian *black box* lebih berfokus pada menemukan kesalahan program, sedangkan pengujian *white box* berfokus pada struktur internal (*source code*) program, dimana setiap baris kode yang akan di-*compile* dicek satu persatu untuk mengetahui apakah masih terjadi *error*. Cara pengujian dilakukan melalui dua tahap, yaitu tahap pengujian *alpha* dan tahap pengujian *beta.*

4.2.1. Pengujian *Alpha*

Pada tahap *alpha*, pengujian dilakukan untuk memastikan aplikasi *game Survive From Death* dapat berjalan sesuau dengan kebutuhan dan tujuan yang diharapkan. Pendekatan perancangan kasus uji menggunakan dua macam teknik yaitu *white box* dan *black box*.

4.2.1.1. Skenario Pengujian Aplikasi

Skenario pengujian menjelaskan pengujian terhadap sistem yang ada pada aplikasi *game Survive From Death*. Berikut ini merupakan skenario pengujian dari aplikasi game dapat dilihat pada tabel 4.2.

Tabel 4. 2 Skenario Pengujian

No	Komponen yang diuji	Butir Uji	Jenis Pengujian
1	Menu Utama	Klik Menu Mulai Main	*Black Box*
		Klik Menu Petunjuk	*Black Box*
		Klik Menu Cerita	*Black Box*
		Klik Menu Option	*Black Box*
		Klik MenuKeluar	*Black Box*
2	Menu Mulai Main	Klik Tombol Mulai	*Black Box*
		Intro *game*	*Black Box*
3	Menu Petunjuk	Klik Menu Petunjuk	*Black Box*
		Klik Tombol kembali	*Black Box*
4	Menu Cerita	Klik Menu Cerita	*Black Box*
		Klik Tombol Kembali	*Black Box*

5	Menu Option	Klik Menu Option	*Black Box*
		Klik Tombol Volume	*Black Box*
		Klik Tombol Grafik	*Black Box*
		Klik Tombol Status	*Black Box*
		Klik Tombol Kembali	*Black Box*
6	Intro	Klik Tombol Mulai	*Black Box*
		Halaman Misi	*Black Box*
7	Halaman Misi	Klik Tombol Lewati	*Black Box*
		Permainan *Level* 1	*Black Box*
8	Pause Menu	Tampil Pause Menu	*Black Box*
		Klik Tombol Mulai Main	*Black Box*
		Klik Tombol Keluar	*Black Box*
9	Kontrol Karakter Pemain	Karakter Bergerak Kedepan	*Black Box*
		Karakter Bergerak Kekanan	*Black Box*
		Karakter Bergerak Kekiri	*Black Box*
		Karakter Bergerak Kebelakang	*Black Box*
		Karakter Bergerak	*Black Box*

		Melompat	
		Karakter Menyerang	*Black Box*
		Karakter Me-*Reload* Senjata	*Black Box*
		Karakter Berlari	*Black Box*
		Karakter Mengganti Senjata	*Black Box*
10	Algoritma A*(A Star)	Implementasi Algoritma A* Pada Musuh Untuk Melakukan Pencarian Terhadap Karakter Pemain	*White Box*
11	*Finite State Machine* (FSM)	Implementasi FSM pada musuh untuk menentukan aksi atau prilaku.	*White Box*

4.2.1.2. Kasus dan Hasil Pengujian (*Black Box*)

Kasus dan hasil uji perangkat lunak yang sudah dibangun dengan menggunakan teknik *black box* berdasarkan pengamatan yang dilakukan dan menghasilkan kesimpulan dari pengujian tersebut adalah sebagai berikut:

1. **Pengujian Menu Utama**

   Pengujian menu utama merupakan proses dimana pengguna menyentuh menu-menu yang ada pada tampilan menu utama. Adapun hasil dari proses pengujian dapat dilihat pada tabel 4.3.

Tabel 4. 3 Pengujian Menu Utama

Kasus dan Hasil Uji			
Masukan	**Keluaran yang diharapkan**	**Pengamatan**	**Kesimpulan**
Klik Menu Mulai Main	Sistem melakukan proses untuk menampilkan Intro permainan	Sistem menampilkan intro permainan	[√] Berhasil [ ] Tidak Berhasil
Klik Menu Petunjuk	Sistem melakukan proses untuk menampilkan halam petunjuk	Sistem menampilkan halaman petunjuk	[√] Berhasil [ ] Tidak Berhasil
Klik Menu Cerita	Sistem melakukan proses untuk menampilkan halaman cerita	Sistem menampilkan halaman cerita	[√] Berhasil [ ] Tidak Berhasil
Klik Menu Option	Sistem melakukan proses untuk menampilkan halaman option	Sistem menampilkan halaman option	[√] Berhasil [ ] Tidak Berhasil
Klik Menu Keluar	Sistem melakukan proses untuk menutup aplikasi	Sistem menutup aplikasi	[√] Berhasil [ ] Tidak Berhasil

2. Pengujian Menu Mulai Main

Pengujian menu mulai main merupakan proses setelah pengguna mengklik tombol mulai pada menu utama yang selanjutkan akan menampilkan intro permainan. Adapun hasil dari proses pengujian dapat dilihat pada tabel 4.4.

Tabel 4. 4 Pengujian Menu Mulai Main

Kasus dan Hasil Uji			
Masukan	**Keluaran yang diharapkan**	**Pengamatan**	**Kesimpulan**
Klik Tombol Mulai	Sistem melakukan proses untuk menampilkan halaman intro permainan	Sistem menampilkan halaman intro permainan	[√] Berhasil [ ] Tidak Berhasil

3. Pengujian Menu Petunjuk

Pengujian menu petunjuk merupakan proses dimana pengguna dapat melihat petunjuk permainan. Adapun hasil dari proses pengujian dapat dilihat pada tabel 4.5.

Tabel 4. 5 Pengujian Menu Petunjuk

Kasus dan Hasil Uji			
Masukan	**Keluaran yang diharapkan**	**Pengamatan**	**Kesimpulan**
Klik Menu Petunjuk	Sistem melakukan proses untuk menampilkan Petunjuk permainan	Sistem menampilkan petunjuk permainan	[√] Berhasil [ ] Tidak Berhasil
Klik Tombol Kembali	Sistem melakukan proses untuk menampilkan kembali menu utama	Sistem menampilkan menu utama	[√] Berhasil [ ] Tidak Berhasil

4. Pengujian Menu Cerita

Pengujian menu cerita merupakan proses dimana pengguna dapat melihat cerita keseluruhan dari permainan. Adapun hasil dari proses pengujian dapat dilihat pada tabel 4.6.

Tabel 4. 6 Pengujian Menu Cerita

Kasus dan Hasil Uji			
Masukan	**Keluaran yang diharapkan**	**Pengamatan**	**Kesimpulan**
Klik Menu Cerita	Sistem melakukan proses untuk menampilkan cerita permainan	Sistem menampilkan cerita permainan	[√] Berhasil [ ] Tidak Berhasil
Klik Tombol Kembali	Sistem melakukan proses untuk menampilkan kembali menu utama	Sistem menampilkan menu utama	[√] Berhasil [ ] Tidak Berhasil

5. Pengujian Menu Option

Pengujian menu option merupakan proses dimana pengguna dapat melihat dan menkofigurasi option yang tersedia pada game. Adapun hasil dari proses pengujian dapat dilihat pada tabel 4.7.

Tabel 4. 7 Pengujian Menu Option

Kasus dan Hasil Uji			
Masukan	**Keluaran yang diharapkan**	**Pengamatan**	**Kesimpulan**
Klik Menu Option	Sistem melakukan proses untuk	Sistem menampilkan	[√] Berhasil

	menampilkan menu option permainan	menu option permainan	[ ] Tidak Berhasil
Klik Tombol Volume	Sistem melakukan proses untuk menampilkan konfigurasi volume suara permainan	Sistem menampilkan konfigurasi volume suara permainan	[√] Berhasil [ ] Tidak Berhasil
Klik Tombol Grafik	Sistem melakukan proses untuk menampilkan konfigurasi grafik permainan	Sistem menampilkan konfigurasi grafik permainan	[√] Berhasil [ ] Tidak Berhasil
Klik Tombol Status	Sistem melakukan proses untuk menampilkan status aplikasi	Sistem menampilkan status aplikasi	[√] Berhasil [ ] Tidak Berhasil
Klik Tombol Kembali	Sistem melakukan proses untuk kembali menampilkan menu utama	Sistem menampilkan halaman menu utama	[√] Berhasil [ ] Tidak Berhasil

6. Pengujian Intro

Pengujian intro merupakan proses dimana pengguna dapat melihat intro atau cerita awal permainan. Adapun hasil dari proses pengujian dapat dilihat pada tabel 4.8.

Tabel 4. 8 Pengujian Intro

Kasus dan Hasil Uji			
Masukan	**Keluaran yang diharapkan**	**Pengamatan**	**Kesimpulan**
Klik Tombol Mulai	Sistem melakukan proses untuk menampilkan halaman misi permainan	Sistem menampilkan halaman misi permainan	[√] Berhasil [ ] Tidak Berhasil

7. Pengujian Halaman Misi

Pengujian halaman misi merupakan proses dimana pengguna dapat melihat misi yang harus dilakukan untuk menyelesaikan suatu level. Adapun hasil dari proses pengujian dapat dilihat pada tabel 4.9.

Tabel 4. 9 Pengujian Halaman Misi

Kasus dan Hasil Uji			
Masukan	**Keluaran yang diharapkan**	**Pengamatan**	**Kesimpulan**
Klik Tombol Lewati	Sistem melakukan proses untuk menampilkan level 1 permainan	Sistem menampilkan level 1 permainan	[√] Berhasil [ ] Tidak Berhasil

8. Pengujian Pause Menu

Pengujian pause menu merupakan proses dimana pengguna dapat menghentikan sementara jalannya permainan sewaktu-waktu atau ingin keluar ditengah-tengah jalannya permainan. Adapun hasil dari proses pengujian dapat dilihat pada tabel 4.10.

Tabel 4. 10 Pengujian Pause Menu

Kasus dan Hasil Uji			
Masukan	**Keluaran yang diharapkan**	**Pengamatan**	**Kesimpulan**
Tekan Tombol Esc pada keyboard	Sistem melakukan proses untuk menampilkan halaman pause menu	Sistem menampilkan halaman pause menu	[√] Berhasil [ ] Tidak Berhasil
Klik Tombol Mulai Main	Sistem melakukan proses untuk menampilkan *Level* 1/2/3/4 permainan	Sistem menampilkan *Level* 1/2/3/4 permainan	[√] Berhasil [ ] Tidak Berhasil
Tekan Tombol Keluar	Sistem melakukan proses untuk menampilkan kembali menu utama	Sistem menampilkan menu utama	[√] Berhasil [ ] Tidak Berhasil

9. Pengujian Kontrol Karakter Pemain

Pengujian kontrol karakter pemain merupakan proses yang dilakukan oleh pemain untuk menggerakkan karakter pemain pada permainan. Adapun hasil dari proses pengujian dapat dilihat pada tabel 4.11

Tabel 4. 11 Pengujian Kontrol Karakter Pemain

Kasus dan Hasil Uji			
Masukan	**Keluaran yang diharapkan**	**Pengamatan**	**Kesimpulan**
Tekan Tombol W	Karakter akan bergerak kedepan	Karakter pemain bergerak kedepan	[√] Berhasil [ ] Tidak Berhasil
Tekan Tombol D	Karakter pemain akan bergerak kekanan	Karakter pemain bergerak kekanan	[√] Berhasil [ ] Tidak Berhasil
Tekan Tombol A	Karakter pemain akan bergerak kekiri	Karakter pemain bergerak kekiri	[√] Berhasil [ ] Tidak Berhasil
Tekan Tombol S	Karakter pemain akan bergerak kebelakang	Karakter pemain bergerak kebelakang	[√] Berhasil [ ] Tidak Berhasil
Tekan Tombol Space	Karakter pemain akan bergerak melompat	Karakter pemain melompat	[√] Berhasil [ ] Tidak Berhasil
Klik Kanan pada Mouse	Karakter pemain akan melakukan serangan	Karakter pemain melakukan serangan	[√] Berhasil [ ] Tidak Berhasil
Tekan Tombol R	Karakter pemain akan mengisi ulang peluru senjata	Karakter pemain melakukan isi ulang peluru senjata	[√] Berhasil [ ] Tidak Berhasil
Tekan Tombol Shift + W/D/A/S	Karakter pemain akan berlari	Karakter pemain berlari	[√] Berhasil [ ] Tidak Berhasil
Tekan Tombol 1	Senjata karakter	Senjata karakter	[√] Berhasil

atau 2	pemain akan berganti	pemain berganti	[ ] Tidak Berhasil

4.2.1.3. Kasus dan Pengujian (*White Box*)

Berikut ini merupakan kasus pengujian algoritma A* (A star) pada perangkat lunak yang sudah dibangun menggunakan teknik (*White Box*), berdasarkan skenario pengujian aplikasi yang sudah dijabarkan sebelumnya.

1. Pengujian Algoritma A* (A *Star*)

Pengujian algoritma A* merupakan proses dimana akan diuji implementasi A* terhadap objek musuh dalam melakukan pencarian karakter pemain.

a. Algoritma

```
1.  public void Awake () {
                startEndModifier.Awake (this);
        }

        public void OnDestroy () {
                startEndModifier.OnDestroy (this);
        }

        private List<IPathModifier> modifiers = new List<IPathModifier> ();

        public void RegisterModifier (IPathModifier mod) {
2.  if (modifiers == null) {
                        modifiers = new List<IPathModifier> (1);
                }

                modifiers.Add (mod);
        }

3.  public void DeregisterModifier (IPathModifier mod) {
                if (modifiers == null) {
                        return;
                }
                modifiers.Remove (mod);
        }
        public enum ModifierPass {
                PreProcess,
                PostProcessOriginal,
                PostProcess
        }

4.  public void RunModifiers (ModifierPass pass, Path p) {

                bool changed = true;
                while (changed) {
                        changed = false;
5.  for (int i=0;i<modifiers.Count-1;i++) {
6.  if (modifiers[i].Priority < modifiers[i+1].Priority) {
                                        IPathModifier tmp = modifiers[i];
                                        modifiers[i] = modifiers[i+1];
                                        modifiers[i+1] = tmp;
                                        changed = true;
                                }
                        }
```

```
            }
7.  switch (pass) {
8.  case ModifierPass.PreProcess:
                            if (preProcessPath != null) preProcessPath
(p);
                            break;
                    case ModifierPass.PostProcessOriginal:
                            if    (postProcessOriginalPath    !=    null)
postProcessOriginalPath (p);
                            break;
                    case ModifierPass.PostProcess:
                            if (postProcessPath != null) postProcessPath
(p);
                            break;
            }

9.  if (modifiers.Count       == 0) return;

            ModifierData prevOutput = ModifierData.All;
            IPathModifier prevMod = modifiers[0];

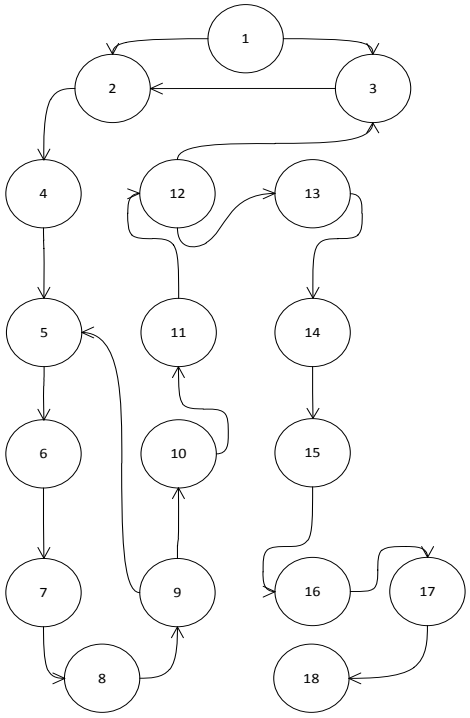
10. for (int i=0;i<modifiers.Count;i++) {
                    MonoModifier mMod = modifiers[i] as MonoModifier;
11. if (mMod != null && !mMod.enabled) continue;
12. switch (pass) {
                            case ModifierPass.PreProcess:
                                    modifiers[i].PreProcess (p);
                                    break;
                            case ModifierPass.PostProcessOriginal:
                                    modifiers[i].ApplyOriginal (p);
                                    break;
                            case ModifierPass.PostProcess:

13. ModifierData newInput = ModifierConverter.Convert
    (p,prevOutput,modifiers[i].input);
14. if (newInput != ModifierData.None) {
                                            modifiers[i].Apply
(p,newInput);
                                            prevOutput                  =
modifiers[i].output;
15. } else {
                                            UnityEngine.Debug.Log  ("Error
converting "+(i > 0 ? prevMod.GetType ().Name : "original")+"'s output to
"+(modifiers[i].GetType ().Name)+"'s input.\nTry rearranging the modifier
priorities on the Seeker.");

                                            prevOutput                  =
ModifierData.None;
                                    }
                                    prevMod = modifiers[i];

                                    break;
                    }
16. if (prevOutput == ModifierData.None) {
17. break;
                    }
            }
        }
18.end
```

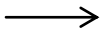
b. *Flowgraph*



Gambar 4. 12 Flowgraph A*

Keterangan :

 = Menggambarkan kondisi

 = Menggambarkan aksi

c. *Cyclomatic Complexity* V(G)

E = 20 , N = 18

V(G) = E – N +2

V(G) = 20 – 18 +2

V(G) = 2 + 2 = 4

Keterangan : E= jumlah aksi

N = jumlah kondisi

d. *Independent Path*

Path 1= 1-2-4-5-6-7-8-9-5-6-7-8-9-10-11-12-13-14-15-16-17-18

Path 2= 1-2-4-5-6-7-8-9-10-11-12-3-2-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18

Path 3= 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18

Path 4= 1-3-2-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18

e. *Graph* Matriks

Tabel 4. 12 *Graph Matriks* Algoritma A*

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1		**1**																
2			**1**	**1**														
3				**1**														
4					**1**													
5						**1**												
6							**1**											
7								**1**										
8									**1**									
9					**1**					**1**								
10											**1**							
11												**1**						
12			**1**										**1**					
13														**1**				
14															**1**			
15																**1**		
16																	**1**	
17																		**1**
18																		

V(G) = Jumlah *Graph Matriks* + 1

V(G) = 3 + 1

V(G) = 4

f. Kesimpulan

Berdasarkan hasil pengujian dihasilkan nilai *Cyclomatic Complexity* yang sama yaitu 4, maka dapat disimpulkan bahwa pada proses algoritma A* berjalan dengan baik, karena hasil pengujian memiliki nilai yang sama.

2. Pengujian *Finite State Machine* (FSM)

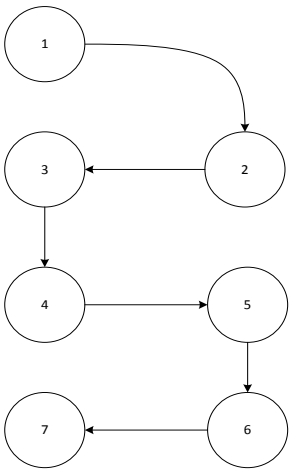
Pengujian FSM merupakan proses dimana akan diuji implementasi FSM terhadap objek musuh untuk menentukan prilaku atau aksi musuh.

a. Algoritma

```
1.  function Awake(){
        myTransform = transform;
       anim.animation.wrapMode = WrapMode.Loop;
       anim.animation[attackAnim].wrapMode = WrapMode.Once;
       anim.animation[attackAnim].layer = 2;
       anim.animation.Stop();
    }
2.  function Start(){
       target = GameObject.FindWithTag("Player").transform;
    }
3.  function Update () {
        var distance = (target.position - myTransform.position).magnitude;
4.  if (distance < dontComeCloserRange){
                        moveSpeed = 0;
                        anim.animation[idleAnim].speed = .4;
                        anim.animation.CrossFade(idleAnim);
                }else{
                        moveSpeed = Random.Range(4, 6);
                        anim.animation.CrossFade(walkAnim);}
5.  if (distance > giveUpRange) {
```

```
                chasing = false;
            }
6.  if (distance < attackRange && Time.time > attackTime) {
                    anim.animation[attackAnim].speed = 2.0;
                anim.animation.CrossFade(attackAnim);
                target.SendMessage( "PlayerDamage", maximumHitPoints);
                attackTime = Time.time + attackRepeatTime;
                    audio.PlayOneShot(attack, 1.0 / audio.volume); }
        } else {
              anim.animation[idleAnim].speed = .4;
              anim.animation.CrossFade(idleAnim);
7.  if (distance < chaseRange) {
                chasing = true;
            }}
    }
    function OnDrawGizmosSelected (){
       Gizmos.color = Color.yellow;
       Gizmos.DrawWireSphere (transform.position, attackRange);
       Gizmos.color = Color.red;
       Gizmos.DrawWireSphere (transform.position, chaseRange);
    }
```

a. *Flowgraph*



Gambar 4. 13 Flowgraph FSM

Keterangan :

 = Menggambarkan kondisi

 = Menggambarkan aksi

b. *Cyclomatic Complexity* V(G)

E = 6 , N = 7

V(G) = E – N +2

V(G) = 6 – 7 +2

V(G) = (-1) + 2 = 1

Keterangan : E= jumlah aksi

N = jumlah kondisi

c. *Independent Path*

Path 1= 1-2-3-4-5-6-7

d. *Graph* Matrik

Tabel 4. 13 *Graph* Matrik FSM

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

V(G) = Jumlah *Graph Matriks* + 1

V(G) = 0 + 1

V(G) = 1

e. Kesimpulan

Berdasarkan hasil pengujian dihasilkan nilai *Cyclomatic Complexity* yang sama yaitu 1, maka dapat disimpulkan bahwa pada proses algoritma FSM berjalan dengan baik, karena hasil pengujian memiliki nilai yang sama.

4.2.1.4. Kesimpulan Pengujian *Alpha*

Dari hasil pengujian yang dilakukan, dapat disimpulkan bahwa sistem sudah berjalan sesuai dengan yang diharapkan dan secara fungsional 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 kepuasa para pengguna terhadap aplikasi yang telah dibangun. Pengujian dilakukan terhadap 20 responden dengan usia minimal 15 tahun dan setiap responden diberikan 5 pertanyaan.

1. Pertanyaan Pengguna

Adapun pertanyaan yang diajukan dalam kuesioner untuk pengguna aplikasi ini adalah sebagai berikut :

1. Apakah tampilan grafis aplikasi *game Survive From Death* secara umum sudah baik?
   a. Sangat Baik
   b. Baik
   c. Cukup Baik
   d. Kurang Baik
   e. Tidak Baik
2. Apakah anda setuju AI musuh yang terdapat pada aplikasi *game* ini sudah berjalan dengan baik?
   a. Sangat Setuju
   b. Setuju
   c. Ragu-ragu
   d. Tidak Setuju
   e. Sangat Tidak Setuju

3. Apakah alur permainan aplikasi ini menarik ?
   a. Sangat Menarik
   b. Menarik
   c. Cukup Menarik
   d. Kurang Menarik
   e. Tidak Menarik
4. Menurut Anda bagaimana tingkat kesulitan dari aplikasi game ini?
   a. Sangat Mudah
   b. Mudah
   c. Cukup Mudah
   d. Agak Sulit
   e. Sulit
5. Apakah karakter pemain mudah untuk dikendalikan?
   a. Sangat mudah
   b. Mudah
   c. Cukup mudah
   d. Agak Sulit
   e. Sulit

2. Rumus Pengujian *Beta*

Untuk mencari nilai persentase dari masing-masing pendapat kuesioner digunakan rumus skala likert sebagai berikut :

$$P = \frac{\sum s}{skorideal} \times 100\%$$

Keterangan :

P = Nilai persentase yang dicari

$\sum s$ = Jumlah skor yang didapatkan dari setiap pilihan jawaban dikalikan dengan nilai yang telah ditetapkan sebelumnya

skorideal= Nilai tertinggi pada point pilihan dikalikan dengan jumlah responden

Berikut adalah data yang didapat dari hasil kuesioner terhadap 20 orang responden .

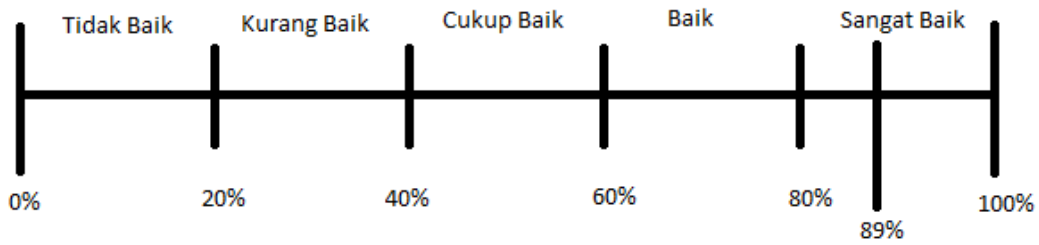
1. Apakah tampilan grafis aplikasi *game Survive From Death* secara umum sudah baik.

Tabel 4. 14 Hasil Pengujian Kuesioner No 1

Pertanyaan	Jawaban	Skor	Responden	S
1	Sangat Baik	5	9	45
	Baik	4	11	44
	Cukup Baik	3	0	0
	Kurang Baik	2	0	0
	Tidak Baik	1	0	0
Jumlah			**20**	**89**

$$P = \frac{89}{100} \times 100\% = 89\%$$

Berdasarkan hasil persentase nilai dapat disimpulkan bahwa penilaian terhadap tampilan grafis aplikasi *game* secara umum adalah 89% atau bisa dikategorikan sebagai sangat baik. Berikut kriteria interprestasi skor :



Gambar 4. 14 *Interprestasi* Skor Pertanyaan No 1

2. Apakah musuh yang terdapat pada aplikasi *game* ini sudah cerdas.

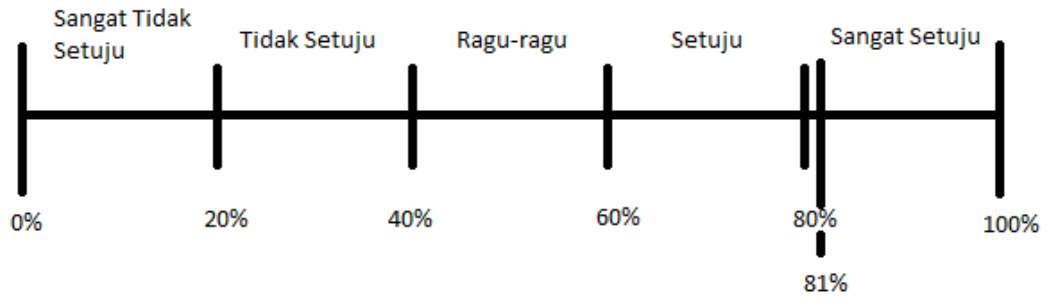
Tabel 4. 15 Hasil Pengujian Kuesioner No 2

Pertanyaan	Jawaban	Skor	Responden	S
	Sangat Setuju	5	2	10
	Setuju	4	17	68

2	Ragu-ragu	3	1	3
	Tidak Setuju	2	0	0
	Sangat Tidak Setuju	1	0	0
Jumlah			**20**	**81**

$$P = \frac{81}{100} \times 100\% = 81\%$$

Berdasarkan hasil persentase nilai dapat disimpulkan bahwa penilaian terhadap kecerdasan musuh yang terdapat pada aplikasi *game* adalah 81% atau bisa dikategorikan sebagai sangat setuju. Berikut kriteria interprestasi skor :



Gambar 4. 15 *Interprestasi* Skor Pertanyaan No 2

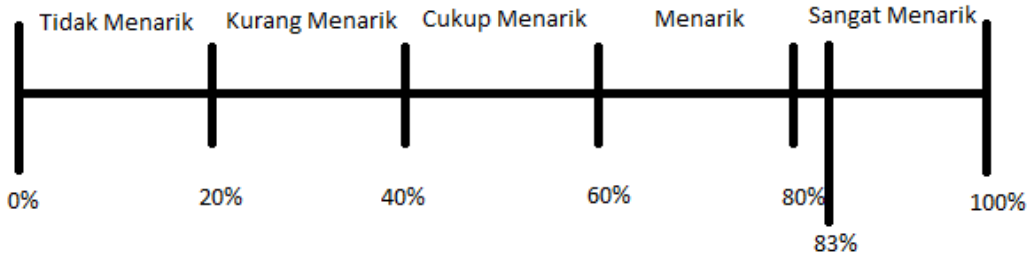
3. Apakah alur permainan aplikasi ini menarik.

Tabel 4. 16 Hasil Pengujian Kuesioner No 3

Pertanyaan	Jawaban	Skor	Responden	S
3	Sangat Menarik	5	5	25
	Menarik	4	13	52
	Cukup Menarik	3	2	6
	Kurang Menarik	2	0	0
	Tidak Menarik	1	0	0
Jumlah			**20**	**83**

$$P = \frac{83}{100} \times 100\% = 83\%$$

Berdasarkan hasil persentase nilai dapat disimpulkan bahwa penilaian terhadap alur permainan aplikasi *game* adalah 83% atau bisa dikategorikan sebagai sangat menarik. Berikut kriteria interprestasi skor :



Gambar 4. 16 *Interprestasi* Skor Pertanyaan No 3

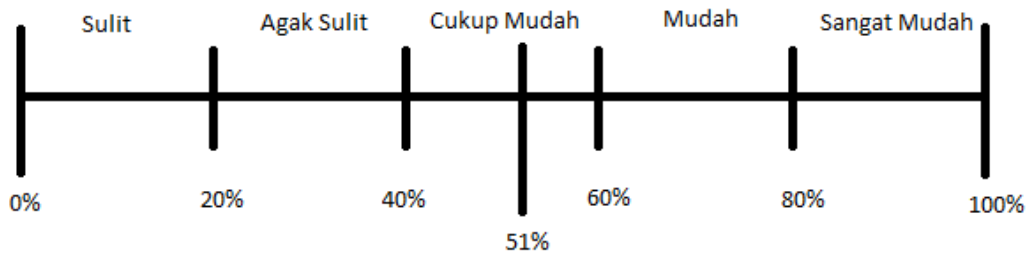
4. Menurut Anda bagaimana tingkat kesulitan dari aplikasi game ini.

Tabel 4. 17 Hasil Pengujian Kuesioner No 4

Pertanyaan	Jawaban	Skor	Responden	S
4	Sangat Mudah	5	1	5
	Mudah	4	2	8
	Cukup Mudah	3	6	18
	Agak Sulit	2	9	18
	Sulit	1	2	2
Jumlah			**20**	**51**

$$P = \frac{51}{100} \times 100\% = 51\%$$

Berdasarkan hasil persentase nilai dapat disimpulkan bahwa penilaian terhadap tingkat kesulita aplikasi *game* adalah 51% atau bisa dikategorikan sebagai cukup mudah. Berikut kriteria interprestasi skor :



Gambar 4. 17 *Interprestasi* Skor Pertanyaan No 4

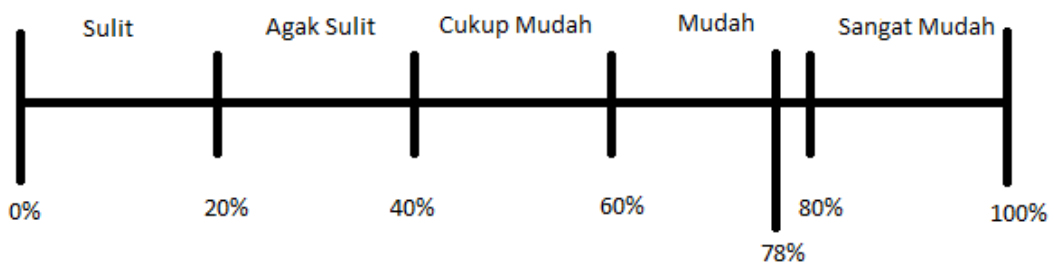
5. Apakah karakter pemain mudah untuk dikendalikan.

Tabel 4. 18 Hasil Pengujian Kuesioner No 5

Pertanyaan	Jawaban	Skor	Responden	S
5	Sangat Mudah	5	3	15
	Mudah	4	13	52
	Cukup Mudah	3	3	9
	Kurang Mudah	2	1	2
	Sulit	1	0	0
Jumlah			**20**	**78**

$$P = \frac{78}{100} \times 100\% = 78\%$$

Berdasarkan hasil persentase nilai dapat disimpulkan bahwa penilaian terhadap pengendalian karakter pemain pada aplikasi *game* adalah 78% atau bisa dikategorikan sebagai mudah. Berikut kriteria interprestasi skor :



Gambar 4. 18 *Interprestasi* Skor Pertanyaan No 5