BAB 4

IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi

Pada bagian tahap implementasi ini merupakan terjemahan perancangan yang berdasarkan hasil analisis pada bab sebelumnya kedalam bahasa pemograman yang dapat dimengerti oleh komputer.

4.2 Perangkat Pendukung yang Digunakan

Dalam proses pembuatan aplikasi ini, tentunya membutuhkan perangkat keras (*hardware*) dan perangkat lunak (*software*). Berikut merupakan penjelasan dari perangkat keras dan perangkat lunak yang digunakan.

4.2.1 Perangkat Keras (*Hardware)*

Perangkat keras yang digunakan selama pengujian aplikasi ini memiliki spesifikasi sebagai berikut :

a. Spesifikasi perangkat keras laptop

Laptop digunakan untuk menjalankan aplikasi yang telah dibangun. Adapun laptop yang digunakan dengan spesifikasi sebagai berikut:

1. Prosesor : 1,80 GHz
2. RAM 1 GB.
3. HARDDISK 250 GB.
4. MONITOR 14’.
5. Mouse dan Keyboard.

4.2.2 Perangkat Lunak (*Software)*

Perangkat lunak yang digunakan selama pembangunan aplikasi ini memiliki spesifikasi Sistem Operasi Windows 7

4.3 Implementasi Aplikasi

4.3.1 Sisi *User*

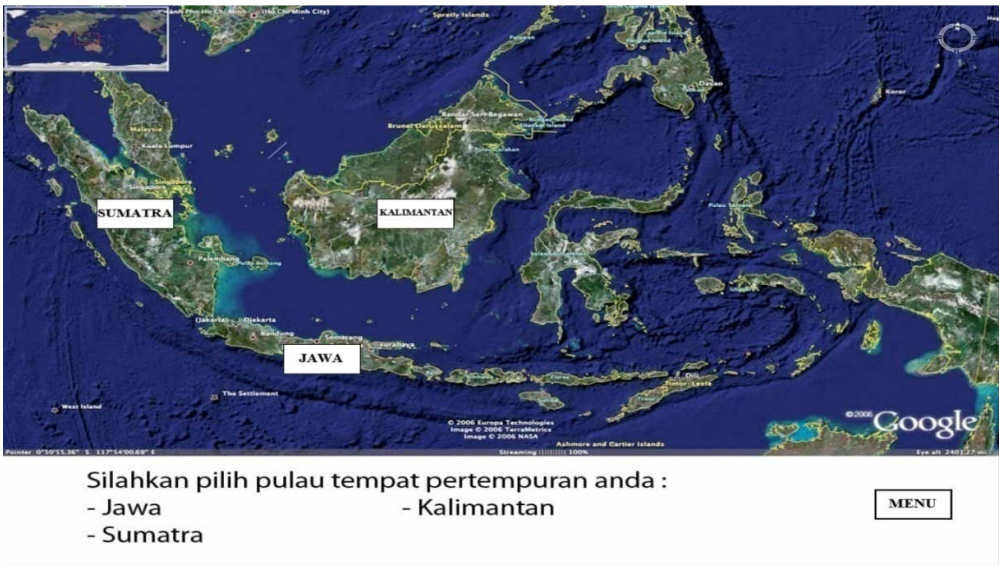
a. Tampilan Utama



Gambar 4.1 Tampilan Menu Utama

Layar ini berfungsi menampilkan Tampilan Menu Utama didalamnya tersedia enam *menu* yang terdiri dari menu mulai, menu cara bermain, menu *credit*, dan keluar game.

b. Tampilan Menu Pilih Pulau



Gambar 4.2 Tampilan Menu Pilih Pulau

Layar ini berfungsi menampilkan tampilan pulau, terdapat tiga pilihan pulau yang dapat dipilih yaitu pulau Jawa, pulau Sumatra, dan pulau Kalimantan. Tugas pemain adalah memilih satu pulau yang akan dimainkan.

c. Tampilan Menu Cara Bermain



Gambar 4.3 Tampilan Menu Cara Bermain

Layar ini berfungsi menampilkan tampilan cara bermain. Tugas pemain adalah memperhatikan langkah-langkah dalam menu cara bermain ini.

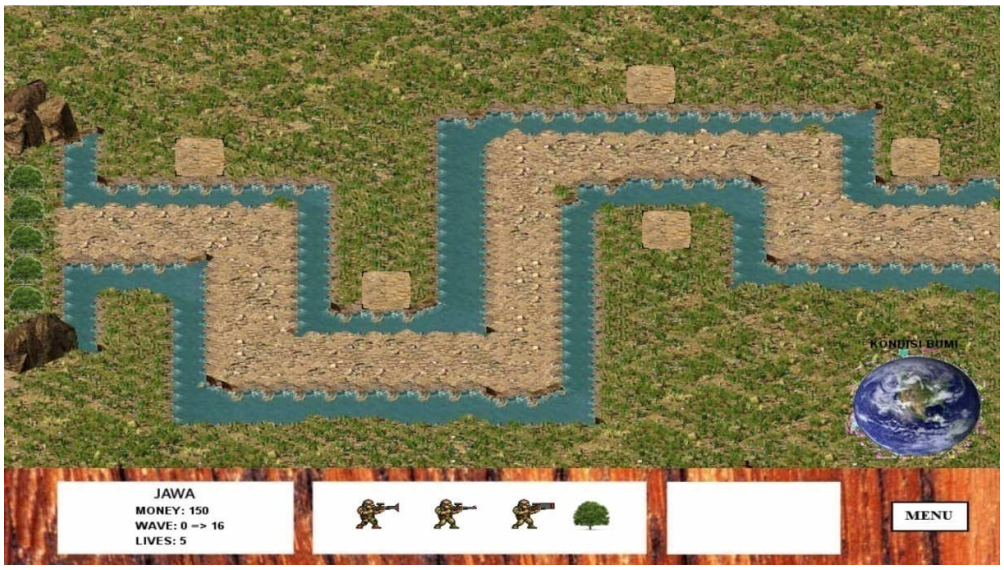
d. Tampilan Menu *Credit*



Gambar 4.4 Tampilan Menu *Credit*

Layar ini berfungsi menampilkan tampilan *credit*, terdapat informasi mengenai *game Tower Defense* Polisi Hutan dan pembuatnya.

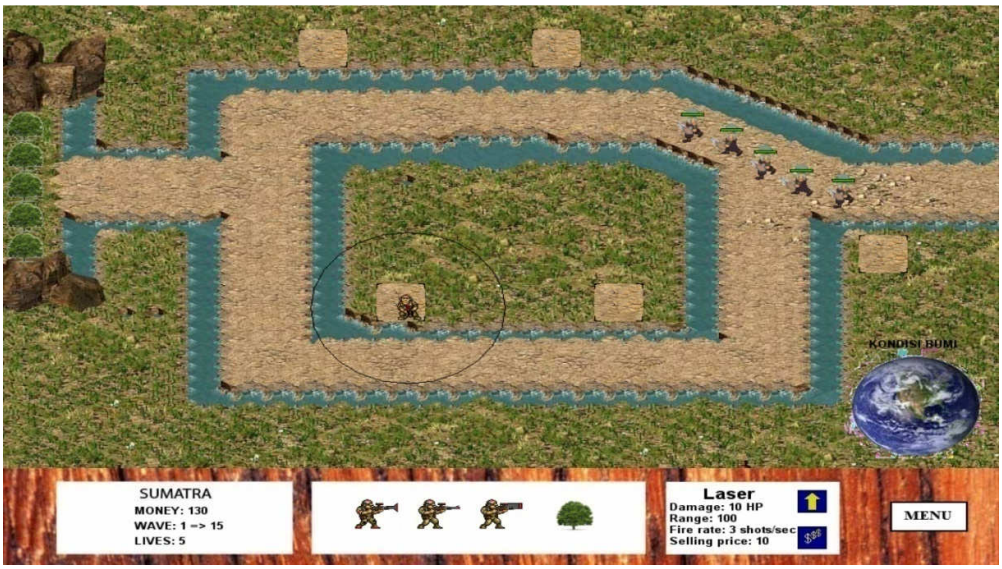
e. Tampilan Pulau Jawa



Gambar 4.5 Tampilan Menu Game Memilih jajanan

Layar ini merupakan tampilan Pulau Jawa, terdapat info mengenai pulau, pilihan tower dan tanaman, tampilan kondisi bumi, dan hutan. Tugas pemain adalah mencegah musuh merusak hutan.

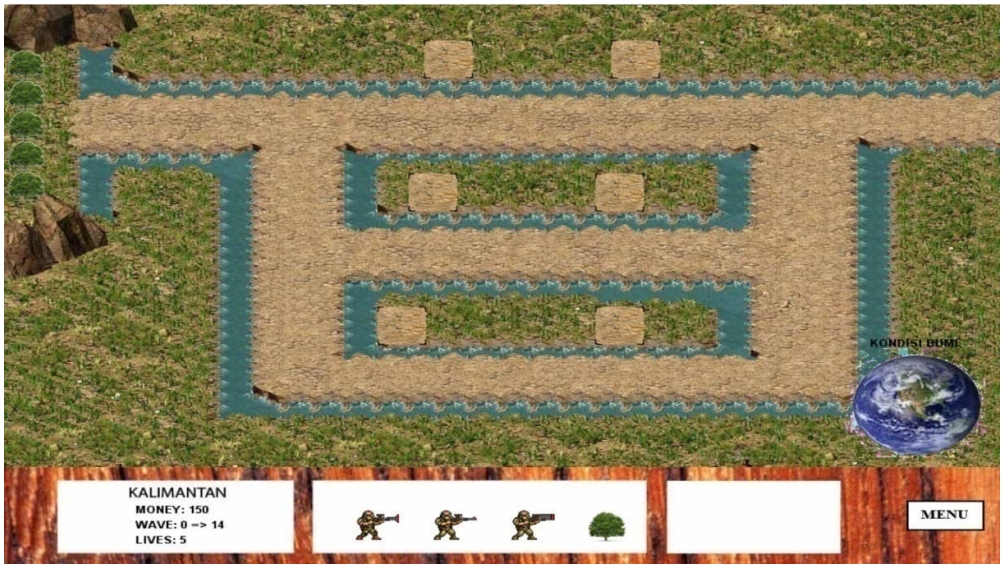
f. Tampilan Pulau Sumatra



Gambar 4.6 Tampilan Pulau Sumatra

Layar ini merupakan tampilan Pulau Sumatra, terdapat info mengenai pulau, pilihan tower dan tanaman, tampilan kondisi bumi, dan hutan. Tugas pemain adalah mencegah musuh merusak hutan.

g. Tampilan Pulau Kalimantan



Gambar 4.7 Tampilan Pulau Kalimantan

Layar ini merupakan tampilan Pulau Jawa, terdapat info mengenai pulau, pilihan tower dan tanaman, tampilan kondisi bumi, dan hutan. Tugas pemain adalah mencegah musuh merusak hutan.

4.3.2 Implementasi *Object*

Tabel 4.1 Implementasi *Object*

Object	Events	Actions
Tower	Create	Execute a piece of code
Source code		

```
{
    firing = 0;
    target = noone;
    active = 1;         //  indicates whether the tower
should fire or not
    range = 100;
    rate = 3;
```

```
    damage = 10;
    price = 20;
    name = 'Laser';     //  tower name
    coll_path = 0;      //  indicates a possible
collision
    selected = 0;       //  indicates whether selected
    level = 0;          //  indicates tower level
}
```

Tower	Step	Execute a piece of code

Source code

```
{
    if firing > 0 firing += 1;
    if firing >= 30/rate firing = 0;
    if not instance_exists(target) target = noone;
    if instance_exists(ob_enemy)
    {
        if target = noone target =
instance_nearest(x,y,ob_enemy);
        if point_distance(x,y,target.x,target.y) >
range
            target = noone;
        else if firing = 0
            firing = 1;
    }
    else if instance_exists(ob_enemy21)
    {
        if target = noone target =
instance_nearest(x,y,ob_enemy21);
        if point_distance(x,y,target.x,target.y) >
range
            target = noone;
        else if firing = 0
            firing = 1;
    }
    else if instance_exists(ob_enemy22)
    {
        if target = noone target =
instance_nearest(x,y,ob_enemy22);
        if point_distance(x,y,target.x,target.y) >
range
            target = noone;
        else if firing = 0
            firing = 1;
    }
```

```
    else if instance_exists(ob_enemy31)
    {
        if target = noone target =
instance_nearest(x,y,ob_enemy31);
        if point_distance(x,y,target.x,target.y) >
range
            target = noone;
        else if firing = 0
            firing = 1;
    }
    else if instance_exists(ob_enemy32)
    {
        if target = noone target =
instance_nearest(x,y,ob_enemy32);
        if point_distance(x,y,target.x,target.y) >
range
            target = noone;
        else if firing = 0
            firing = 1;
    }
    else if instance_exists(ob_enemy33)
    {
        if target = noone target =
instance_nearest(x,y,ob_enemy33);
        if point_distance(x,y,target.x,target.y) >
range
            target = noone;
        else if firing = 0
            firing = 1;
    }
    if instance_exists(ob_rute1)
    {
        if target = noone target =
instance_nearest(x,y,ob_rute1);
        if point_distance(x,y,target.x,target.y) >
range
            target = noone;
        else if firing = 0
            firing = 1;
    }
    if instance_exists(ob_rute21)
    {
        if target = noone target =
instance_nearest(x,y,ob_rute21);
        if point_distance(x,y,target.x,target.y) >
range
            target = noone;
```

```
        else if firing = 0
            firing = 1;
    }
    if instance_exists(ob_rute22)
    {
        if target = noone target =
instance_nearest(x,y,ob_rute22);
        if point_distance(x,y,target.x,target.y) >
range
            target = noone;
        else if firing = 0
            firing = 1;
    }
    if instance_exists(ob_rute212)
    {
        if target = noone target =
instance_nearest(x,y,ob_rute212);
        if point_distance(x,y,target.x,target.y) >
range
            target = noone;
        else if firing = 0
            firing = 1;
    }
    if instance_exists(ob_rute31)
    {
        if target = noone target =
instance_nearest(x,y,ob_rute31);
        if point_distance(x,y,target.x,target.y) >
range
            target = noone;
        else if firing = 0
            firing = 1;
    }
    if instance_exists(ob_rute32)
    {
        if target = noone target =
instance_nearest(x,y,ob_rute32);
        if point_distance(x,y,target.x,target.y) >
range
            target = noone;
        else if firing = 0
            firing = 1;
    }
    if instance_exists(ob_rute33)
    {
        if target = noone target =
instance_nearest(x,y,ob_rute33);
```

```
        if point_distance(x,y,target.x,target.y) >
range
            target = noone;
        else if firing = 0
            firing = 1;
    }
    if firing = 1 depth  = -3;
    else depth = -2;
}
```

Tower	Begin step	Execute a piece of code

Source code

```
{

    if !collision_rectangle(x-16,y-
16,x+15,y+15,ob_path_marker,0,1)

    if !collision_rectangle(x-16,y-
16,x+15,y+15,ob_tower_laser,0,1)

        coll_path = 0;

}
```

Tower	Mouse (left pressed)	Execute a piece of code

Source code

```
{
    if active = 1
    if selected = 0
    {
        ob_tower_laser.selected = 0;
        selected = 1;
        controller.ii_select =
instance_position(x,y,ob_tower_laser);
    }
    if active = 0
    if coll_path = 0
    {
        active = 1;
        controller.money -= price;
    }
}
```

Tower	Draw	Execute a piece of code

Source code

```
{
    draw_set_alpha(0.5);
    if coll_path = 0 draw_set_color(c_lime);
    else draw_set_color(c_red);
    draw_circle(x,y,16,0);
    draw_set_color(c_white);
    draw_circle(x,y,range,1);
    draw_set_alpha(1);
}
```

Tower	Draw	Execute a piece of code

Source code

```
{
    draw_sprite(sprite_index,0,x,y);
    if selected = 1 draw_circle(x,y,range,1);
    draw_text(x,y,level);
    if not instance_exists(target) target = noone;
    if (target != noone && firing = 1)
    {
        draw_set_color(c_red);
        draw_line_width(x,y,target.x,target.y,4);
        target.own_health -= damage;
        if target.own_health < 1 target = noone;
        draw_set_color(c_white);
        sound_play(sd_laser);
    }
}
```

Enemy	Create	Execute a piece of code

Source code

```
{
    own_health =
round(40+power(controller.wave,1.5)*10);
    health_def =
round(40+power(controller.wave,1.5)*10);
    path_start(pth_level1,4,0,1);
}
```

Enemy	Step	Execute a piece of code
Source code		

```
{
    if own_health <= 0
    {
        sound_play(sd_explosion);
        instance_create(x,y,ob_explosion_large);
        controller.money += controller.wave;
        instance_destroy();
    }
}
```

Enemy	End of path	Execute a piece of code
Source code		

```
{
    instance_destroy();
    instance_create(-32,0,ob_tebang1);
}
```

Enemy	Draw	Execute a piece of code
Source code		

```
{
    draw_sprite_ext(sprite_index,-
1,x,y,1,1,direction,c_white,1);
    draw_set_alpha(0.8);
    draw_healthbar(x-12,y-16,x+11,y-
13,own_health/health_def*100,
                   c_black,c_red,c_lime,0,1,1);
    draw_set_alpha(1);
}
```

Ob_rute1, ob_rute21, ob_rute22, ob_rute212, ob_rute31, ob_rute32,	Create	Execute a piece of code

ob_rute33,		
Source code		

{
own_health = 1;
health_def = 0;
}		

Ob_rute1, ob_rute21, ob_rute22, ob_rute212, ob_rute31, ob_rute32, ob_rute33,	Step	Execute a piece of code
Source code		

{
if own_health <= 0
{
controller.jml_tower1 += 1;
instance_destroy();
}
}		

Btn_upgrade	Step	Execute a piece of code
Source code		

{
if controller.ii_select = noone
visible = 0;
else
{
visible = 1;
if controller.ii_select.level < 5
price = floor(controller.ii_select.price*0.4);
else price = 'MAX';
}
}		

Btn_upgrade	Mouse (left pressed)	Execute a piece of code

Source code

```
{
    if visible = 1
    {
        var ii;
        ii = controller.ii_select;
        if ii.level < 5
        if controller.money >= price   //cek money
        {
            controller.money -= price;     //money
berkurang
            ii.damage = round(ii.damage*1.1); //tower
level berikutnya
            ii.range = round(ii.range*1.1);
            ii.rate = min(ii.rate*1.2,10);
            ii.price = round(ii.price*1.5);
            ii.level += 1;
        }
    }
}
```

Btn_sell	Step	Execute a piece of code

Source code

```
{
    if controller.ii_select = noone
        visible = 0;
    else visible = 1;
}
```

Btn_sell	Mouse (left pressed)	Execute a piece of code

Source code

```
{
    if visible = 1
    {
        var ii;
        ii = controller.ii_select;
        controller.money += floor(ii.price/2);  //money
bertambah
        with ii instance_destroy();       //tower
dihapus
        controller.ii_select = noone;
```

```
    }
}
```

Btn_sell	Draw	Execute a piece of code

Source code

```
{
    draw_sprite(sprite_index,image_index,x,y);
}
```

Ob_tanaman_button	Create	Execute a piece of code

Source code

```
{
image_speed = 0;
    price = 30;
}
```

Ob_tanaman_button	Mouse (left pressed)	Execute a piece of code

Source code

```
{
if controller.money >= price
        {
        controller.money -= price;
        controller.lives += 1;
        }
}
```

Btn_tower_laser	Create	Execute a piece of code

Source code

```
{
    image_speed = 0;
    price = 20;
}
```

Btn_tower_laser	Begin step	Execute a piece of code

Source code

```
image_index = 0;
```

Btn_tower_laser	Mouse (no button)	Execute a piece of code
Source code		

```
image_index = 1;
```

Btn_tower_laser	Mouse (left pressed)	Execute a piece of code
Source code		

```
{
    var ii;
    if
instance_position(mouse_x,mouse_y,ob_tower_laser) exit;
    if controller.money < price exit;
    ii = instance_create(x,y,ob_tower_laser);
    ii.active = 0;
    ob_tower_laser.selected = 0;
    controller.ii_select = noone;
}
```

Btn_tower_laser	Draw	Execute a piece of code
Source code		

```
{
    draw_sprite(sprite_index,image_index,x,y);
    if image_index = 1
    {
        draw_set_color(c_red);
        draw_set_font(ft_hud);
        draw_text(x,y,price);
    }
}
```

Btn_tower_bullet, Btn_tower_bomb (Btn_tower_laser as parent)	Create	Execute a piece of code
Source code		

```
{
```

```
    image_speed = 0;
    price = 35;   // untuk yang bomb price 45
}
```

Btn_tower_bullet, Btn_tower_bomb (Btn_tower_laser as parent)	Mouse (left pressed)	Execute a piece of code

Source code

```
{
    var ii;
    if
instance_position(mouse_x,mouse_y,ob_tower_laser) exit;
    if controller.money < 15 exit;
    ii = instance_create(x,y,ob_tower_bomb);    //
disesuaikan dengan ob tower
    ii.active = 0;
    ob_tower_laser.selected = 0;
    controller.ii_select = noone;
}
```

Ob_explosion_small, ob_explosion_large	Other (animation end)	Execute a piece of code

Source code

```
instance_destroy();
```

Ob_explosion_small, ob_explosion_large	Draw	Execute a piece of code

Source code

```
{
    draw_set_alpha(1-image_index/4);
    draw_sprite(sprite_index,image_index,x,y);
    draw_set_alpha(1);
}
```

Ob_bullet, ob_bomb	Create	Execute a piece of code
Source code		

{
damage = 0;
sound_play(sd_cannon);
}		

Ob_bullet, ob_bomb	Collision with enemy	Execute a piece of code
Source code		

{
other.own_health -= damage;
sound_play(sd_bullet_hit);
instance_create(x,y,ob_explosion_small);
instance_destroy();
}		

Ob_bullet, ob_bomb	Collision with ob_enemy21	Execute a piece of code
	Collision with ob_enemy22	Execute a piece of code
	Collision with ob_enemy31	Execute a piece of code
	Collision with ob_enemy32	Execute a piece of code
	Collision with ob_enemy33	Execute a piece of code
Source code		

{
other.own_health -= damage;
sound_play(sd_bullet_hit);
instance_create(x,y,ob_explosion_small);
instance_destroy();		

} 		
Ob_bullet, ob_bomb	Collision with ob_rute1	Execute a piece of code
	Collision with ob_rute21	Execute a piece of code
	Collision with ob_rute22	Execute a piece of code
	Collision with ob_rute212	Execute a piece of code
	Collision with ob_rute31	Execute a piece of code
	Collision with ob_rute32	Execute a piece of code
	Collision with ob_rute33	Execute a piece of code
Source code		

{
    other.own_health -= damage;
    instance_destroy();
}		

Ob_bullet, ob_bomb	Other (outside room)	Execute a piece of code
Source code		
instance_destroy();		
Controller	Create	Execute a piece of code
Source code		

{
    money = 150;
    ii_select = noone;
    wave = 0;
    wave_enemies = 0;
    wave_time = 0;
    lives = 5;
    score = 0;
    level = 0;		

```
    jml_tower1 = 0;
    jml_tower2 = 0;
    jml_tower3 = 0;
    statusjml_tower1 = 0;
    statusjml_tower2 = 0;
    statusjml_tower3 = 0;
    statusjarak1 = 0;
    statusjarak2 = 0;
    statusjarak3 = 0;
    status_rute1 = 0;
    status_rute2 = 0;
    status_rute3 = 0;
}
```

Controller	Step	Execute a piece of code

Source code

```
{
    if not instance_exists(ii_select) ii_select =
noone;
    if mouse_check_button_pressed(mb_left)
        if not
instance_position(mouse_x,mouse_y,ob_tower_laser)
        if not
instance_position(mouse_x,mouse_y,btn_tower_laser)
        {
            ii_select = noone;
            ob_tower_laser.selected = 0;
        }
    if mouse_check_button_pressed(mb_right)
    {
        ii_select = noone;
        ob_tower_laser.selected = 0;
    }
    //  Execute wave system
    get_wave_enemies();
}
```

Controller	Step	Execute a piece of code

Source code

```
if level =1
{
if lives >= 5
    {
```

```
    instance_create(22,310,ob_tanaman5);
    instance_create(940,530,ob_bumi5);
    }
else if lives = 4
    {
    instance_create(22,310,ob_tanaman4);
    instance_create(940,530,ob_bumi4);
    }
else if lives = 3
    {
    instance_create(22,310,ob_tanaman3);
    instance_create(940,530,ob_bumi3);
    }
else if lives = 2
    {
    instance_create(22,310,ob_tanaman2);
    instance_create(940,530,ob_bumi2);
    }
else if lives = 1
    {
    instance_create(22,310,ob_tanaman1);
    instance_create(940,530,ob_bumi1);
    }
}
else if level =2
{
if lives >= 5
    {
    instance_create(22,240,ob_tanaman5);
    instance_create(940,530,ob_bumi5);
    }
else if lives = 4
    {
    instance_create(22,240,ob_tanaman4);
    instance_create(940,530,ob_bumi4);
    }
else if lives = 3
    {
    instance_create(22,240,ob_tanaman3);
    instance_create(940,530,ob_bumi3);
    }
else if lives = 2
    {
    instance_create(22,240,ob_tanaman2);
    instance_create(940,530,ob_bumi2);
    }
else if lives = 1
```

```
    {
    instance_create(22,240,ob_tanaman1);
    instance_create(940,530,ob_bumi1);
    }
}
else if level =3
{
if lives >= 5
    {
    instance_create(22,160,ob_tanaman5);
    instance_create(940,530,ob_bumi5);
    }
else if lives = 4
    {
    instance_create(22,160,ob_tanaman4);
    instance_create(940,530,ob_bumi4);
    }
else if lives = 3
    {
    instance_create(22,160,ob_tanaman3);
    instance_create(940,530,ob_bumi3);
    }
else if lives = 2
    {
    instance_create(22,160,ob_tanaman2);
    instance_create(940,530,ob_bumi2);
    }
else if lives = 1
    {
    instance_create(22,160,ob_tanaman1);
    instance_create(940,530,ob_bumi1);
    }
}
}
```

Controller	Draw	Execute a piece of code

Source code

```
{
    draw_set_color(c_black);
    draw_set_alpha(1);
    draw_set_font(ft_hud);
    draw_text(135,660,'MONEY: '+string(money));
    draw_text(135,680,'WAVE: '+string(wave)+' => '
        +string(floor((600-wave_time)/30)));
    draw_text(135,700,'LIVES: '+string(lives));
```

```
    draw_text(890,440,'KONDISI BUMI');
    //  Draw tower stats
    if ii_select != noone
    {
        draw_set_font(ft_header);
        draw_text(720,635,ii_select.name);
        draw_set_font(ft_hud);
        draw_text(685,655,'Damage:
'+string(ii_select.damage)+' HP');
        draw_text(685,670,'Range:
'+string(ii_select.range));
        draw_text(685,685,'Fire rate:
'+string(ii_select.rate)+' shots/sec');
        draw_text(685,700,'Selling price:
'+string(floor(ii_select.price/2)));
    }
}
```

Controller	Key press (space)	Execute a piece of code

Source code

```
{
    if wave_enemies = 0
    {
        wave += 1;
        wave_time = 0;
        wave_enemies = min(5+floor(wave*0.4),16);
        ee = wave_enemies;
    }
}
```

Ob_level1-3	End step	Execute a piece of code

Source code

```
{
controller.level = 1;       //nilai disesuaikan dengan
level
}
```

4.4 Pengujian Sistem

Pengujian sistem merupakan hal terpenting yang dilakukan untuk menemukan kekurangan atau kesalahan pada perangkat lunak yang diuji. Pengujian bermaksud untuk mengetahui perangkat lunak yang dibuat sudah memenuhi kriteria yang sesuai dengan tujuan perancangan perangkat lunak tersebut.

4.5 Pengujian Alpha (Fungsional)

Pengujian fungsional yang digunakan untuk menguji sistem yang baru adalah metode pengujian *alpha*. Pengujian *alpha* berfokus pada persyaratan fungsional perangkat lunak.

1. Rencana Pengujian

Rencana pengujian yang akan dilakukan pada aplikasi ini selengkapnya terlihat pada tabel di bawah.

Tabel 4.2 Rencana Pengujian Aplikasi

No.	Komponen yang diuji	Detail Pengujian	Jenis Uji
1	Menu	Memilih Tombol Mulai	*Black Box*
		Memilih Tombol Cara Bermain	*Black Box*
		Memilih Tombol *Credit*	*Black Box*
		Memilih Tombol Keluar	*Black Box*
2	Pilih Pulau	Memilih Tombol Pulau Jawa	*Black Box*
		Memilih Tombol Pulau Sumatra	*Black Box*

		Memilih Tombol Pulau Kalimantan	*Black Box*
		Memilih Tombol Menu	*Black Box*
3	Pulau Jawa	Pilih Tower	*Black Box*
		Pilih Tambah Tanaman	*Black Box*
		Upgrade Tower	*Black Box*
		Sell Tower	*Black Box*
		Memilih Tombol Menu	*Black Box*
4	Pulau Sumatra	Pilih Tower	*Black Box*
		Pilih Tambah Tanaman	*Black Box*
		Upgrade Tower	*Black Box*
		Sell Tower	*Black Box*
		Memilih Tombol Menu	*Black Box*
5	Pulau Kalimantan	Pilih Tower	*Black Box*
		Pilih Tambah Tanaman	*Black Box*
		Upgrade Tower	*Black Box*
		Sell Tower	*Black Box*
		Memilih Tombol Menu	*Black Box*
6	Cara Bermain	Memilih Tombol Sebelumnya	*Black Box*
		Memilih Tombol Berikutnya	*Black Box*

		Memilih Tombol Menu	*Black Box*
7	*Credit*	Memilih Tombol Menu	*Black Box*
8	Jalur Musuh	Penelusuran algoritma *fuzzy*	*Whitebox*

2. Pengujian *Blackbox*

Berdasarkan hasil pengujian dan yang telah disusun, maka dapat dilakukan pengujian sebagai berikut. Pengujian yang dilakukan meliputi pengujian tampilan pembuat.

Tabel 4.3 Kasus dan Hasil Uji

Kasus dan hasil uji (data normal)				
No.	**Kasus /diuji**	**Skenario Uji**	**Hasil yang diharapkan**	**Hasil Pengujian**
1.	Menu	Memilih Tombol Mulai	Menampilkan halaman Pilih Pulau	[√] diterima [ ] ditolak
		Memilih Tombol Cara Bermain	Menampilkan halaman Cara Bermain	[√] diterima [ ] ditolak
		Memilih Tombol *Credit*	Menampilkan halaman *Credit*	[√] diterima [ ] ditolak
		Memilih Tombol Keluar	Keluar Game	[√] diterima [ ] ditolak

2.	Pilih Pulau	Memilih Tombol Pulau Jawa	Menampilkan halaman Pulau Jawa	[√] diterima [ ] ditolak
		Memilih Tombol Pulau Sumatra	Menampilkan halaman Pulau Sumatra	[√] diterima [ ] ditolak
		Memilih Tombol Pulau Kalimantan	Menampilkan halaman Pulau Kalimantan	[√] diterima [ ] ditolak
		Memilih Tombol Menu	Menampilkan halaman Menu Utama	[√] diterima [ ] ditolak
3.	Pulau Jawa	Pilih Tower	Menambah tower di *map*, serta uang berkurang	[√] diterima [ ] ditolak
		Pilih Tambah Tanaman	Menambah tanaman di *map*, serta uang berkurang	[√] diterima [ ] ditolak
		Upgrade Tower	Tower ter-*upgrade* ke level berikutnya	[√] diterima [ ] ditolak
		Sell Tower	Tower dihapus dari *map*	[√] diterima [ ] ditolak
		Memilih Tombol Menu	Menampilkan halaman Menu Utama	[√] diterima [ ] ditolak

4.	Pulau Sumatra	Pilih Tower	Menambah tower di *map*, serta uang berkurang	[√] diterima [ ] ditolak
		Pilih Tambah Tanaman	Menambah tanaman di *map*, serta uang berkurang	[√] diterima [ ] ditolak
		Upgrade Tower	Tower ter-*upgrade* ke level berikutnya	[√] diterima [ ] ditolak
		Sell Tower	Tower dihapus dari *map*	[√] diterima [ ] ditolak
		Memilih Tombol Menu	Menampilkan halaman Menu Utama	[√] diterima [ ] ditolak
5.	Pulau Kalimantan	Pilih Tower	Menambah tower di *map*, serta uang berkurang	[√] diterima [ ] ditolak
		Pilih Tambah Tanaman	Menambah tanaman di *map*, serta uang berkurang	[√] diterima [ ] ditolak
		Upgrade Tower	Tower ter-*upgrade* ke level berikutnya	[√] diterima [ ] ditolak
		Sell Tower	Tower dihapus dari *map*	[√] diterima [ ] ditolak

		Memilih Tombol Menu	Menampilkan halaman Menu Utama	[√] diterima [ ] ditolak
6.	Cara Bermain	Memilih Tombol Sebelumnya	Menampilkan halaman sebelumnya	[√] diterima [ ] ditolak
		Memilih Tombol Berikutnya	Menampilkan halaman berikutnya	[√] diterima [ ] ditolak
		Memilih Tombol Menu	Menampilkan halaman Menu Utama	[√] diterima [ ] ditolak
7.	*Credit*	Memilih Tombol Menu	Menampilkan halaman Menu Utama	[√] diterima [ ] ditolak

3. Pengujian *Whitebox*

Pengujian Algoritma Fuzzy Jalur Musuh (*Whitebox*)

Pengujian pada proses jalur musuh merupakan proses pengujian dari pengimplementasian algoritma Fuzzy menggunakan pengujian *whitebox*, dan menguji apakah algoritma Fuzzy berjalan dengan seharusnya pada musuh.

a. *Pseudocode* Algoritma Fuzzy (get_wave_enemies)

1. //Inisialisasi variabel
   wave = 0;
   wave_enemies = 0;
   wave_time = 0;

```
level = 0;
jml_tower1 = 0;
jml_tower2 = 0;
jml_tower3 = 0;
statusjml_tower1 = 0;
statusjml_tower2 = 0;
statusjml_tower3 = 0;
statusjarak1 = 0;
statusjarak2 = 0;
statusjarak3 = 0;
status_rute1 = 0;
status_rute2 = 0;
status_rute3 = 0;
globalvar ee;
```

2. masukan nilai awal pada variabel
3. if wave_time = 600 then

```
{
    wave += 1;
    wave_time = 0;
    wave_enemies = min(15+floor(wave*0.4),12);
    ee = wave_enemies;
}
```

4. if wave_time < 600 then wave_time += 1;
5. if wave_enemies > 0

```
{
```

6. if wave_time = (ee-wave_enemies)*15+15

```
{
```

7. //fuzzification

```
if jml_tower1 <= 2
    {
    statusjml_tower1 = 1;
```

```
    }
 else if jml_tower1 >= 3
    {
    statusjml_tower1 = 2;
    }
 if jml_tower2 <= 2
    {
    statusjml_tower2 = 1;
    }
 else if jml_tower2 >= 3
    {
    statusjml_tower2 = 2;
    }
if jml_tower3 <= 2
    {
    statusjml_tower3 = 1;
    }
 else if jml_tower3 >= 3
    {
    statusjml_tower3 = 2;
    }
```

8.

```
//inference dan defuzification
 if statusjarak1 = 1
    {
    if statusjml_tower1 = 1
       {
       status_rute1 = 1;
       }
    else if statusjml_tower1 = 2
       {
       status_rute1 = 2;
```

```
        }
    }
else if statusjarak1 = 2
    {
    if statusjml_tower1 = 1
        {
        status_rute1 = 2;
        }
    else if statusjml_tower1 = 2
        {
        status_rute1 = 3;
        }
    }
if statusjarak2 = 1
    {
    if statusjml_tower2 = 1
        {
        status_rute2 = 1;
        }
    else if statusjml_tower2 = 2
        {
        status_rute2 = 2;
        }
    }
else if statusjarak2 = 2
    {
    if statusjml_tower2 = 1
        {
        status_rute2 = 2;
        }
    else if statusjml_tower2 = 2
```

```
                {
                status_rute2 = 3;
                }
            }
        if statusjarak3 = 1
            {
            if statusjml_tower3 = 1
                {
                status_rute3 = 1;
                }
            else if statusjml_tower3 = 2
                {
                status_rute3 = 2;
                }
            }
        else if statusjarak3 = 2
            {
            if statusjml_tower3 = 1
                {
                status_rute3 = 2;
                }
            else if statusjml_tower3 = 2
                {
                status_rute3 = 3;
                }
            }
9.      if level = 1
10.         {
            instance_create(-32,0,ob_enemy);
             wave_enemies -= 1;
            }
```

```
11.   if level = 2
      {
12.     if status_rute1 = 1
                {
                  if status_rute2 = 1
                  {
                  instance_create(-32,0,ob_enemy21);
                  wave_enemies -= 1;
                  }
                  else if status_rute2 = 2
                  {
                  instance_create(-32,0,ob_enemy21);
                  wave_enemies -= 1;
                  }
                  else if status_rute2 = 3
                  {
                  instance_create(-32,0,ob_enemy21);
                  wave_enemies -= 1;
                  }
                }
13.     else if status_rute1 = 2
                {
                  if status_rute2 = 1
                  {
                  instance_create(-32,0,ob_enemy22);
                  wave_enemies -= 1;
                  }
                  else if status_rute2 = 2
                  {
                  instance_create(-32,0,ob_enemy21);
```

```
                wave_enemies -= 1;
                }
                else if status_rute2 = 3
                {
                instance_create(-32,0,ob_enemy21);
                wave_enemies -= 1;
                }
            }
14.     else if status_rute1 = 3
            {
                if status_rute2 = 1
                {
                instance_create(-32,0,ob_enemy22);
                wave_enemies -= 1;
                }
                else if status_rute2 = 2
                {
                instance_create(-32,0,ob_enemy22);
                wave_enemies -= 1;
                }
                else if status_rute2 = 3
                {
                instance_create(-32,0,ob_enemy21);
                wave_enemies -= 1;
                }
            }
15.     }
16.     else if level = 3
        {
17.     if status_rute1 = 1
```

```
{
  if status_rute2 = 1
      {
    if status_rute3 = 1
      {
      instance_create(-32,0,ob_enemy31);
      wave_enemies -= 1;
      }
    else if status_rute3 = 2
      {
      instance_create(-32,0,ob_enemy31);
      wave_enemies -= 1;
      }
    else if status_rute3 = 3
      {
      instance_create(-32,0,ob_enemy31);
      wave_enemies -= 1;
      }
    }
  else if status_rute2 = 2
    {
    if status_rute3 = 1
      {
      instance_create(-32,0,ob_enemy31);
      wave_enemies -= 1;
      }
    else if status_rute3 = 2
      {
      instance_create(-32,0,ob_enemy31);
      wave_enemies -= 1;
      }
```

```
        else if status_rute3 = 3
          {
           instance_create(-32,0,ob_enemy31);
           wave_enemies -= 1;
           }
        }
      else if status_rute2 = 3
        {
          if status_rute3 = 1
            {
            instance_create(-32,0,ob_enemy31);
            wave_enemies -= 1;
            }
          else if status_rute3 = 2
            {
            instance_create(-32,0,ob_enemy31);
            wave_enemies -= 1;
            }
          else if status_rute3 = 3
            {
            instance_create(-32,0,ob_enemy31);
            wave_enemies -= 1;
            }
          }
      }
18.   else if status_rute1 = 2
    {
      if status_rute2 = 1
          {
        if status_rute3 = 1
          {
```

```
        instance_create(-32,0,ob_enemy32);
        wave_enemies -= 1;
        }
      else if status_rute3 = 2
        {
        instance_create(-32,0,ob_enemy32);
        wave_enemies -= 1;
        }
      else if status_rute3 = 3
        {
        instance_create(-32,0,ob_enemy32);
        wave_enemies -= 1;
        }
      }
  else if status_rute2 = 2
      {
      if status_rute3 = 1
        {
        instance_create(-32,0,ob_enemy33);
        wave_enemies -= 1;
        }
      else if status_rute3 = 2
        {
        instance_create(-32,0,ob_enemy31);
        wave_enemies -= 1;
        }
      else if status_rute3 = 3
        {
         instance_create(-32,0,ob_enemy31);
         wave_enemies -= 1;
         }
```

```
    }
  else if status_rute2 = 3
    {
      if status_rute3 = 1
        {
        instance_create(-32,0,ob_enemy33);
        wave_enemies -= 1;
        }
      else if status_rute3 = 2
        {
        instance_create(-32,0,ob_enemy31);
        wave_enemies -= 1;
        }
      else if status_rute3 = 3
        {
        instance_create(-32,0,ob_enemy31);
        wave_enemies -= 1;
        }
      }
  }
```

19.
```
else if status_rute1 = 3

    {

    if status_rute2 = 1

      {

      if status_rute3 = 1

        {

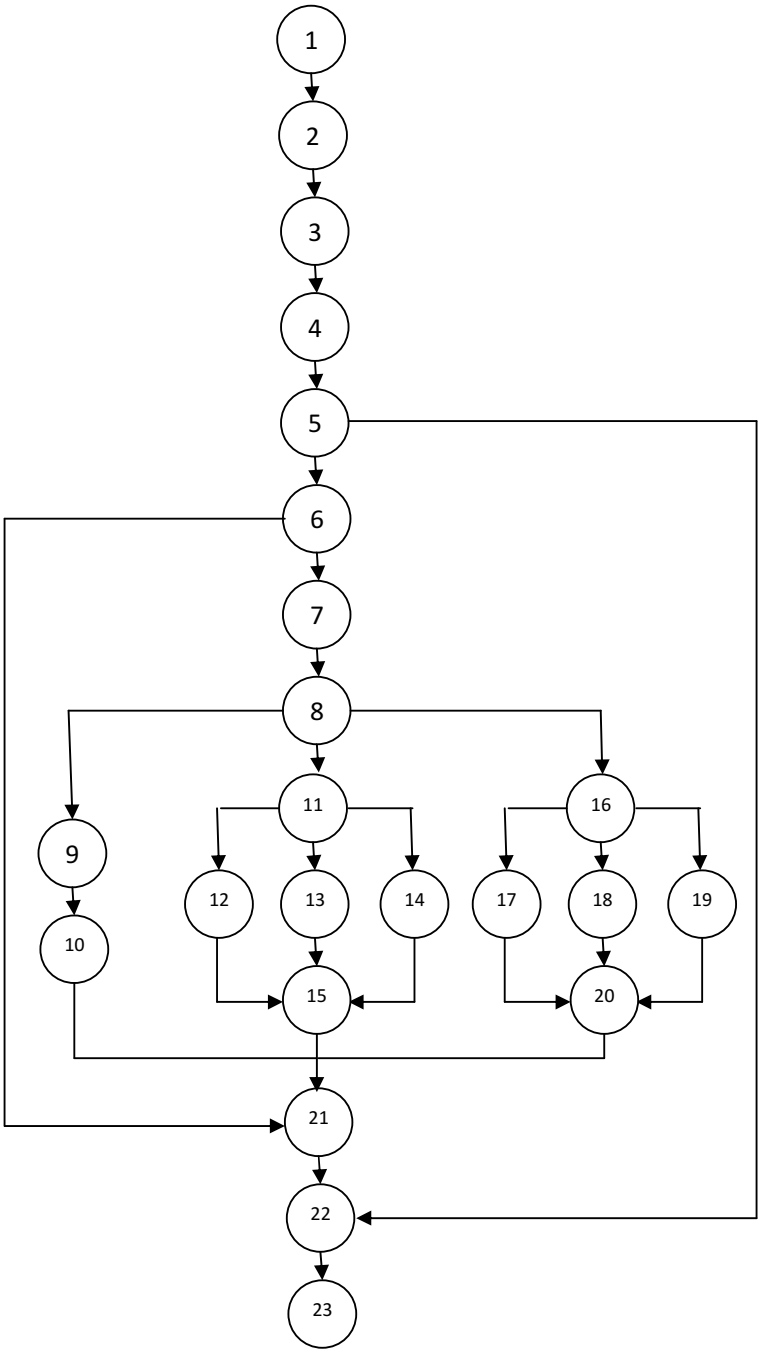
        instance_create(-32,0,ob_enemy32);
```

```
    wave_enemies -= 1;
    }
  else if status_rute3 = 2
    {
    instance_create(-32,0,ob_enemy32);
    wave_enemies -= 1;
    }
  else if status_rute3 = 3
    {
    instance_create(-32,0,ob_enemy32);
    wave_enemies -= 1;
     }
  }
else if status_rute2 = 2
  {
  if status_rute3 = 1
    {
    instance_create(-32,0,ob_enemy33);
    wave_enemies -= 1;
    }
  else if status_rute3 = 2
```

```
        {
        instance_create(-32,0,ob_enemy32);
        wave_enemies -= 1;
        }
      else if status_rute3 = 3
        {
        instance_create(-32,0,ob_enemy32);
        wave_enemies -= 1;
        }
      }
    else if status_rute2 = 3
      {
      if status_rute3 = 1
        {
        instance_create(-32,0,ob_enemy33);
        wave_enemies -= 1;
        }
      else if status_rute3 = 2
        {
        instance_create(-32,0,ob_enemy33);
        wave_enemies -= 1;
```

```
                    }
                else if status_rute3 = 3
                    {
                    instance_create(-32,0,ob_enemy31);
                    wave_enemies -= 1;
                    }
                }
            }
20.         }end if
21.       }end if
22.    }end if
23. End
```

b. *Flowgraph*



Gambar 4.8 *Flowgraph* Jalur Musuh (*Fuzzy*)

c. *Cyclomatic Complexity* V(G)

V(G) = E – N + 2

Di mana E adalah edge, dan N adalah node. Jadi :

V(G) = 30 edge – 23 node + 2 = 9

d. *Independent Path*

Dari hasil perhitungan cyclomatic complexity terdapat 8 independent path yaitu:

Path 1 = 1-2-3-4-5-22-23

Path 2 = 1-2-3-4-5-6-21-22-23

Path 3 = 1-2-3-4-5-6-7-8-9-10-21-22-23

Path 4 = 1-2-3-4-5-6-7-8-11-12-15-21-22-23

Path 5 = 1-2-3-4-5-6-7-8-11-13-15-21-22-23

Path 6 = 1-2-3-4-5-6-7-8-11-14-15-21-22-23

Path 7 = 1-2-3-4-5-6-7-8-16-17-20-21-22-23

Path 8 = 1-2-3-4-5-6-7-8-16-18-20-21-22-23

Path 9 = 1-2-3-4-5-6-7-8-16-19-20-21-22-23

Tabel 4.4 Graph Matriks Jalur Musuh (Fuzzy)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	N(E)-1
1		1																						1-1=0
2			1																					1-1=0
3				1																				1-1=0
4					1																			1-1=0
5						1																1		2-1=1
6							1														1			2-1=1
7								1																1-1=0
8									1		1					1								3-1=2
9										1														1-1=0
10																					1			1-1=0
11												1	1	1										3-1=2
12															1									1-1=0
13															1									1-1=0
1															1									1-

4																						1=0
15																			1			1-1=0
16																1	1	1				3-1=2
17																			1			1-1=0
18																			1			1-1=0
19																			1			1-1=0
20																				1		1-1=0
21																					1	1-1=0
22																						1	1-1=0
23																							0
SUM(E)+1																							8+1=9

Berikut ini adalah hasil dari pengujian *whitebox* penelusuran algoritma Fuzzy jalur musuh pada aplikasi *game* Tower Defense Polisi Hutan.

Tabel 4.5 Hasil Pengujian *Whitebox*

No	Path	Yang Diharapkan	Hasil Yang Dicapai	Kesimpulan
1	1-2-3-4-5-22-23	*Wave* musuh habis dan permainan selesai	Setelah *wave* musuh habis permainan selesai	[√] Diterima [ ] Ditolak

2	1-2-3-4-5-6-21-22-23	*Wave time* habis dan musuh dibuat	Setelah *wave time* habis musuh dibuat	[√] Diterima [ ] Ditolak
3	1-2-3-4-5-6-7-8-9-10-21-22-23	Membuat musuh di pulau Jawa	Musuh dibuat di pulau Jawa	[√] Diterima [ ] Ditolak
4	1-2-3-4-5-6-7-8-11-12-15-21-22-23	Status rute 1 untuk peta Sumatra aman dan rute 2 memiliki 1 status dari 3 status	Musuh memilih rute teraman di peta Sumatra	[√] Diterima [ ] Ditolak
5	1-2-3-4-5-6-7-8-11-13-15-21-22-23	Status rute 1 untuk peta Sumatra sedang dan rute 2 memiliki 1 status dari 3 status	Musuh memilih rute teraman di peta Sumatra	[√] Diterima [ ] Ditolak
6	1-2-3-4-5-6-7-8-11-14-15-21-22-23	Status rute 1 untuk peta Sumatra berbahaya dan rute 2 memiliki 1 status dari 3 status	Musuh memilih rute teraman di peta Sumatra	[√] Diterima [ ] Ditolak
7	1-2-3-4-5-6-7-8-16-17-20-21-22-23	Status rute 1 untuk peta Kalimantan aman dan rute 2 dan 3 memiliki 1 status dari 3 status	Musuh memilih rute teraman di peta Kalimantan	[√] Diterima [ ] Ditolak
8	1-2-3-4-5-6-7-8-16-18-20-21-22-23	Status rute 1 untuk peta Kalimantan sedang dan rute 2 dan 3 memiliki 1 status dari 3 status	Musuh memilih rute teraman di peta Kalimantan	[√] Diterima [ ] Ditolak
9	1-2-3-4-5-6-7-8-16-19-20-21-22-23	Status rute 1 untuk peta Kalimantan berbahaya dan rute 2 dan 3 memiliki 1 status dari 3 status	Musuh memilih rute teraman di peta Kalimantan	[√] Diterima [ ] Ditolak

Berdasarkan pengujian yang dilakukan, maka bisa diambil kesimpulan bahwa masing-masing *independent path* yang dihasilkan bisa dijalankan dan dieksekusi serta tidak ada perulangan yang tak berhingga.

4. Kesimpulan Hasil Pengujian Alpha

Berdasarkan hasil pengujian dan kasus sampel uji yang telah dilakukan didapat kesimpulan bahwa secara fungsional sistem sudah dapat bekerja dan menghasilkan output yang diharapkan.

4.6 Pengujian Beta

Pada pengujian beta ini, pengujian akan dilakukan secara objektif dengan cara dicoba langsung oleh beberapa responden. Sampel yang diambil adalah 30 responden, nantinya sampel akan dipersilahkan mencoba aplikasi dan mengisi kuesioner yang telah di sediakan. Teknik pengujian yang digunakan dalam pengujian beta ini adalah teknik skala likert.

1. Penafsiran bobot pengujian

   Pengujian menggunakan 3 kategori jawaban dengan bobot yang berbeda untuk setiap jawabannya seperti pada tabel berikut.

Tabel 4.6 Bobot Jawaban Pernyataan Positif

Jawaban	Setuju	Biasa Saja	Tidak Setuju
Bobot	2	1	0

Tabel 4.7 Bobot Jawaban Pernyataan Negatif

Jawaban	Tidak Setuju	Biasa Saja	Setuju
Bobot	2	1	0

2. Kualifikasi Skor

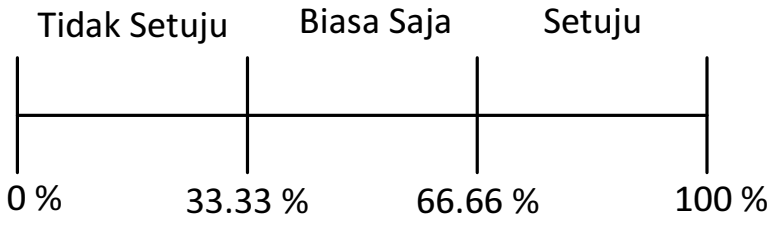
   Persentase skor=(jumlah skor / jumlah bobot nilai tertinggi) x100%

   Berikut adalah kriteria penilaian intrepetasi skor, kualifikasi pesentase = (bobot jawaban / bobot jawaban mak) x 100%;

a. Persentase nilai Setuju = (3 / 3) x 100% = 100%

b. Persentase nilai Ragu-Ragu= (2 / 3) x 100% = 66.66%

c. Persentase nilai Biasa Saja= (1 / 3) x 100% = 33.33%



Gambar 4.9 Kualifikasi Skala Pengukuran

Hasil pengujian *Beta* adalah sebagai berikut:

1. Pertanyaan Nomor 1

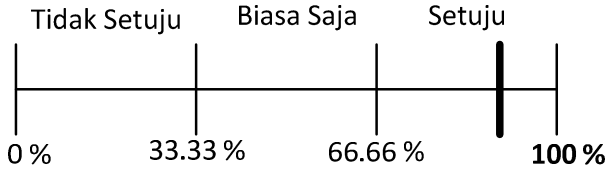
   (1) Apakah menurut anda tampilan antarmuka *game* ini menarik?

Tabel 4.8 Perhitungan Pertanyaan Nomor 1

Pertanyaan	Kategori Jawaban	Bobot	Frekuensi	Skor
1	Setuju	2	21	42
	Biasa Saja	1	9	9
	Tidak Setuju	0	-	-

Jumlah	51

Persentase = (51 / 60) x 100% = 85%



Gambar 4.10 Hasil Pengukuran Pertanyaan Nomor 1

Berdasarkan persentase diatas maka dapat disimpulkan bahwa penilaian tampilan antarmuka *game* ini menarik adalah 85% atau bisa dikategorikan sebagian besar responden setuju.

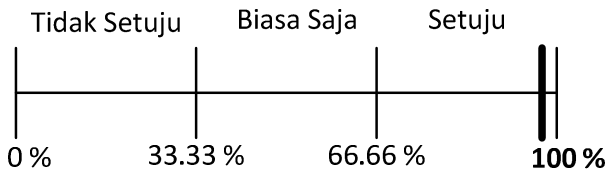
2. Pertanyaan Nomor 2

(2) Apakah menurut anda *game* ini mudah untuk dimainkan?

Tabel 4.9 Perhitungan Pertanyaan Nomor 2

Pertanyaan	Kategori Jawaban	Bobot	Frekuensi	Skor
2	Setuju	2	28	56
	Biasa Saja	1	2	2
	Tidak Setuju	0	-	-
Jumlah				58

Persentase = (58 / 60) x 100% = 96%



Gambar 4.11 Hasil Pengukuran Pertanyaan Nomor 2

Berdasarkan persentase diatas maka dapat disimpulkan bahwa *game* ini mudah untuk dimainkan adalah 96% atau bisa dikategorikan sebagian besar responden setuju.

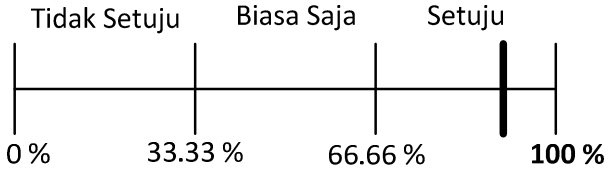
3. Pertanyaan Nomor 3

(3) Apakah tema Hutan dan Polisi Hutan yang ada pada *game* ini baik?

Tabel 4.10 Perhitungan Pertanyaan Nomor 3

Pertanyaan	Kategori Jawaban	Bobot	Frekuensi	Skor
3	Setuju	2	22	44
	Biasa Saja	1	5	5
	Tidak Setuju	0	3	0
Jumlah				49

Persentase = (49 / 60) x 100% = 82%



Gambar 4.12 Hasil Pengukuran Pertanyaan Nomor 3

Berdasarkan persentase diatas maka dapat disimpulkan bahwa tema Hutan dan Polisi Hutan yang ada pada *game* ini baik adalah 82% atau bisa dikategorikan sebagian besar responden setuju.

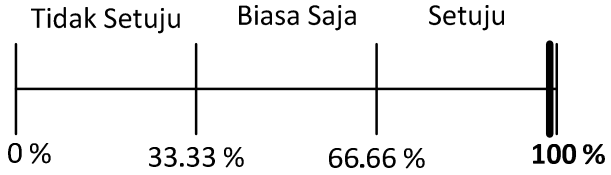
4. Pertanyaan Nomor 4

(4) Dengan musuh yang cerdas untuk memilih rute teraman, apakah tantangan yang diberikan di dalam *game* ini cukup menantang?

Tabel 4.11 Perhitungan Pertanyaan Nomor 4

Pertanyaan	Kategori Jawaban	Bobot	Frekuensi	Skor
4	Setuju	2	29	58
	Biasa Saja	1	1	1
	Tidak Setuju	0	-	-
Jumlah				59

Persentase = (60 / 60) x 100% = 98%



Gambar 4.13 Hasil Pengukuran Pertanyaan Nomor 4

Berdasarkan persentase diatas maka dapat disimpulkan bahwa penilaian musuh yang cerdas untuk memilih rute teraman dalam *game* ini cukup menantang adalah 98% atau bisa dikategorikan sebagian besar responden setuju.

5. Pertanyaan Nomor 5

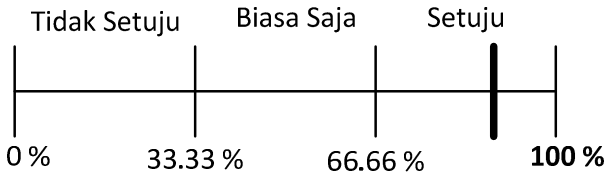
(5) Apakah musik latar yang ada pada *game* ini sudah bagus?

Tabel 4.12 Perhitungan Pertanyaan Nomor 5

Pertanyaan	Kategori Jawaban	Bobot	Frekuensi	Skor
5	Setuju	2	20	40
	Biasa Saja	1	9	9
	Tidak Setuju	0	1	0

Jumlah	49

Persentase = (49 / 60) x 100% = 81%



Gambar 4.14 Hasil Pengukuran Pertanyaan Nomor 5

Berdasarkan persentase diatas maka dapat disimpulkan bahwa penilaian musik latar yang ada pada *game* ini sudah bagus adalah 81% atau bisa dikategorikan sebagian besar responden setuju.

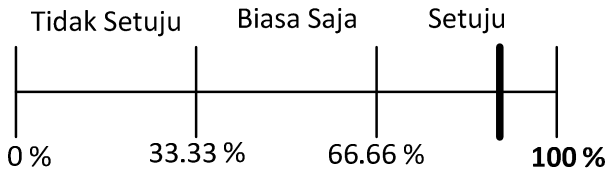
6. Pertanyaan Nomor 6

(6) Apakah *game* ini dapat membantu anda dalam mengenal akibat dari perusakan hutan?

Tabel 4.13 Perhitungan Pertanyaan Nomor 6

Pertanyaan	Kategori Jawaban	Bobot	Frekuensi	Skor
1	Setuju	2	21	42
	Biasa Saja	1	9	9
	Tidak Setuju	0	-	-
Jumlah				51

Persentase = (51 / 60) x 100% = 85%



Gambar 4.15 Hasil Pengukuran Pertanyaan Nomor 6

Berdasarkan persentase diatas maka dapat disimpulkan bahwa penilaian *game* ini dapat membantu dalam mengenal akibat dari perusakan hutan adalah 85% atau bisa dikategorikan sebagian besar responden setuju.

4.6.1 Kesimpulan Pengujian Beta

Berdasarkan pengujian beta yang telah didapat dari hasil kuesioner yang telah dijawab oleh para responden, maka bisa diambil kesimpulan bahwa *game* ”*Tower Defense* Polisi Hutan” ini :

1. Pemain dapat tertarik memainkan *game* ini dengan desain yang menarik.
2. Pemain merasa sangat mudah dalam memainkan *game* ini.
3. Kesesuaian dengan tema Hutan dan Polisi Hutan membuat *game* ini dinilai baik oleh pemain.
4. Pemain merasa tertantang dengan adanya musuh yang cerdas dan jalur yang dinamis.
5. Pemain beranggapan musik yang melatarbelakangi *game* ini bagus.
6. Pemain merasa terbantu dalam mengenal akibat dari perusakan Hutan.