BAB 4
IMPLEMENTASI DAN PENGUJIAN SISTEM

4.1 Implementasi

Tahap implementasi merupakan tahap penciptaan perangkat lunak dan juga tahap kelanjutan dari kegiatan perancangan aplikasi, tahap ini merupakan tahap dimana aplikasi siap untuk dioperasikan. Aplikasi ini diimplementasikan dengan nama Pertempuran Lima Hari Di Semarang dengan *file extension*-nya *.exe*

4.1.1 Implementasi Perangkat Keras

Perangkat keras yang akan digunakan untuk menjalankan *game* Pertempuran Lima Hari Di Semarang ini tidak harus komputer yang berspesifikasi tinggi. Adapun spesifikasi perangkat keras yang digunakan saat implementasi dan pengujian aplikasi *game* Pertempuran Lima Hari Di Semarang adalah sebagai berikut :

1. Prosesor Dual Core 2.0 GHz.
2. RAM 2 GB.
3. Harddisk dengan ruang kosong 1 GB.
4. VGA Card shared 128 MB.
5. Monitor dengan resolusi 1024 x 768 pixels.
6. *Speaker.*
7. *Keyboard* dan *mouse.*

4.1.2 Implementasi Perangkat lunak

Adapun spesifikasi perangkat lunak yang dibutuhkan untuk menjalankan aplikasi Pertempuran lima hari di semarang adalah :

1. Sistem operasi minimal Windows Xp.
2. Graphic card intel HD.
3. DirectX 11.

4.1.3 Implementasi Instalasi Aplikasi *Game*

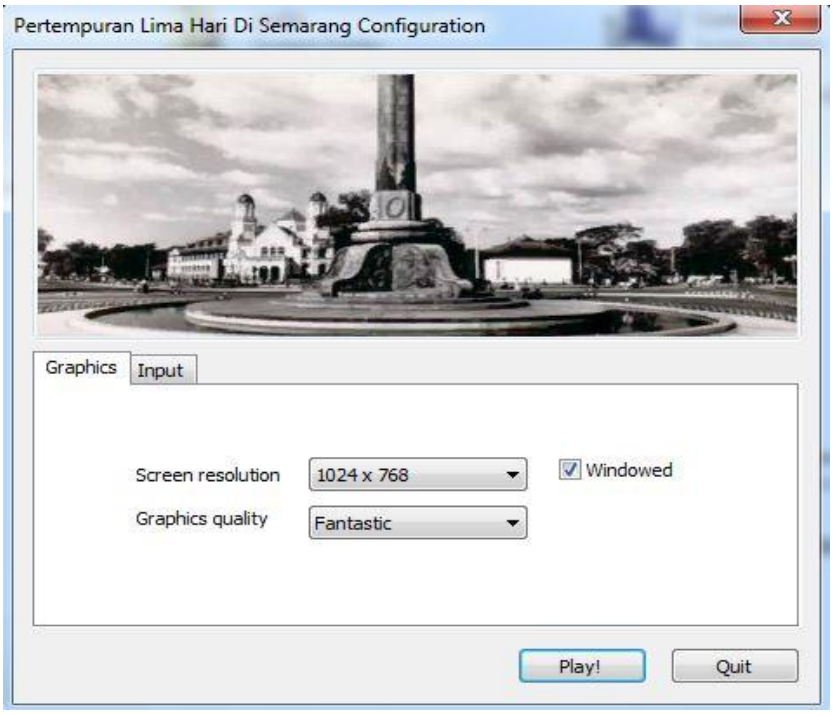
Untuk dapat memainkan *game* Pertempuran Lima Hari Di Semarang, pemain terlebih harus mempunyai *file* Pertempuran Lima Hari Di Semarang.*exe* . Untuk mulai memainkan *game,* pemain hanya perlu menjalankan *file* Pertempuran Lima Hari Di Semarang.*exe* yang telah ada dalam perangkat keras.

4.1.4 Implementasi Antarmuka

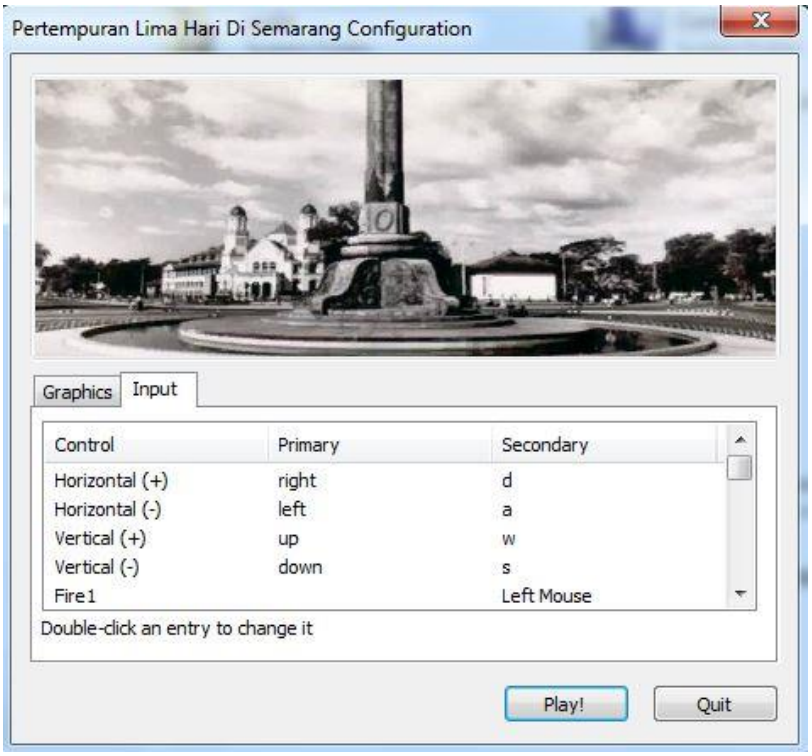
Adapun implementasi antarmuka pada aplikasi *game* Pertempuran Lima Hari Di Semarang adalah sebagai berikut :

4.1.4.1 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

4.1.4.2 Antarmuka Menu Utama

Antarmuka menu utama merupakan tampilan menu utama pada saat pertama kali pengguna membuka aplikasi *game*. Implementasi antarmuka untuk perancangan *game start menu* yang dibangun dapat dilihat dari gambar 4.3.



Gambar 4.3 Implementasi *Start Menu*

Di dalam *game* ini terdapat beberapa menu yang terbagi menjadi tiga yaitu menu mulai permainan, petunjuk dan keluar permainan. Menu mulai permainan berguna untuk memulai memainkan *game*, menu petunjuk untuk

menampilkan halaman cara bermainan atau informasi kontrol dapat dilihat pada gambar 4.4, menu dan terakhir menu keluar permainan digunakan untuk keluar dari aplikasi *game.*



Gambar 4.4 Implementasi Menu Petunjuk

4.1.4.3 Antarmuka Mulai Permainan

Antarmuka mulai permainan merupakan tampilan ketika pengguna mengklik tombol mulai permainan. Pertama adalah tampilan dari antarmuka *scene video* awal main atau *level* 1 dapat dilihat pada gambar 4.5.



Gambar 4.5 Implementasi *Scene Video level* 1

Tampilan antarmuka informasi misi pada *level* 1 dapat dilihat pada gambar 4.6



Gambar 4.6 Implementasi Informasi Misi *Level* 1

Gambar 4.7 merupakan antarmuka permainan *level* 1 yang terdapat pada *game* pertempuran lima hari di semarang. Gambar tersebut memperlihatkan posisi pemain ketika masuk ke dalam permainan dan berada di sebuah desa.



Gambar 4.7 Implementasi Memainkan *Level* 1-1

Gambar 4.8 merupakan antarmuka permainan *level* 1 yang terdapat pada *game* pertempuran lima hari di semarang. Gambar berada di sebuah jembatan menuju gudang gula.



Gambar 4.8 Implementasi Memainkan *Level* 1-2

Gambar 4.9 merupakan antarmuka permainan *level* 1 yang terdapat pada *game* pertempuran lima hari di semarang. Gambar tersebut berada di sebuah gudang gula.



Gambar 4.9 Implementasi Implementasi Memainkan *Level* 1-3

Gambar 4.10 merupakan tampilan antar muka scene video *level* 2



Gambar 4.10 Implementasi *Scene* Video *Level* 2

Gambar 4.11 merupakan tampilan antarmuka informasi misi di *level* 2



Gambar 4.11 Implementasi Memainkan *Level* 2

Gambar 4.12 merupakan antarmuka permainan *level* 2 yang terdapat pada *game* pertempuran lima hari di semarang. Gambar tersebut memperlihatkan posisi pemain ketika berada di simpang lima semarang.



Gambar 4.12 Implementasi Memainkan *Level* 2-1

Gambar 4.13 merupakan antarmuka permainan *level* 2 yang terdapat pada *game* pertempuran lima hari di semarang. Gambar tersebut memperlihatkan posisi pemain berada di salah satu jalan menuju lawang sewu.



Gambar 4.13 Implementasi Memainkan *Level* 2-2

Gambar 4.14 merupakan antarmuka permainan *level* 2 yang terdapat pada *game* pertempuran lima hari di semarang. Gambar tersebut memperlihatkan posisi pemain berada di lawang sewu dan mengambil bendera untuk menyelesaikan permainan.



Gambar 4.14 Implementasi Memainkan *Level* 2-3

Gambar 4.15 merupakan antarmuka *scene* video ketika pemain berhasil menyelesaikan permainan *game* pertempuran lima hari di semarang.



Gambar 4.15 Implementasi *Scene* Video Selesai 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*

Pengujian dilakukan terhadap aplikasi untuk memastikan bahwa aplikasi dapat berjalan dengan benar sesuai dengan kebutuhan dan tujuan yang diharapkan. Pengujian *alpha* berfokus pada persyaratan fungsional perangkat lunak.

4.2.1.1 Skenario Pengujian Aplikasi

Skenario pengujian menjelaskan pengujian terhadap sistem yang ada pada aplikasi *game* Pertempuran Lima Hari Di Semarang. Skenario pengujian yang akan dilakukan pada aplikasi ini selengkapnya dapat dilihat pada table 4.1 :

Tabel 4.1 Skenario Pengujian

No	Komponan Yang Diuji	Skenario Pengujian	Jenis Pengujian
1	Menu Mulai Permainan	Pilih Mulai Permainan	*Black Box*
2	Menu Petunjuk	Pilih Menu Petunjuk	*Black Box*
		Pilih Menu Kembali	*Black Box*
3	Menu Keluar Permainan	Pilih Menu Keluar Permainan	*Black Box*
4	Scene Video	Tekan Tombol Spasi	*Black Box*
5	Informasi Misi	Pilih Mainkan Sekarang	*Black Box*
6	Permainan	Tekan Tombol W Pada *keyboard*	*Black Box*
		Tekan Tombol A Pada *keyboard*	*Black Box*
		Tekan Tombol D Pada *keyboard*	*Black Box*
		Tekan Tombol S Pada *keyboard*	*Black Box*
		Tekan Klik kiri pada *mouse*	*Black Box*
		Tekan Tombol Esc pada Keyboard	*Black Box*
		Tekan Tombol 1 Pada *keyboard*	*Black Box*
		Tekan Tombol 2 Pada *keyboard*	*Black Box*
7	Algoritma	A* (A *Star*)	*White box*
		Fuzzy Logic	*White box*

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

Berikut ini merupakan kasus untuk menguji perangkat lunak yang sudah dibangun dengan menggunakan metode *black box* berdasarkan skenario pengujian aplikasi yang terdapat pada tabel 4.1 yang sudah dijabarkan. Hasil pengujian yang akan dilakukan pada aplikasi ini selengkapnya dapat dilihat pada tabel 4.2.

Tabel 4.2 Pengujian *Black Box*

No	Kasus yang akan di uji	Skenario dan Hasil Uji		
		Skenario Uji	**Hasil yang Diharapkan**	**Kesimpulan**
1.	Menu Mulai Permainan	Pilih Menu Mulai Permainan	Sistem menampilkan halaman *Scene Video*	[√] Berhasil [ ] Tidak Berhasil
2.	Menu Petunjuk	Pilih Menu Petunjuk	Sistem menampilkan petunjuk berupa informasi kontrol yang digunakan dalam *game*	[√] Berhasil [ ] Tidak Berhasil
		Pilih Kembali	Sistem menampilkan menu utama	[√] Berhasil [ ] Tidak Berhasil
3.	Menu Keluar Permainan	Pilih Menu Keluar Permainan	Menutup atau Keluar dari aplikasi	[√] Berhasil [ ] Tidak Berhasil
4.	*Scene Video*	Tekan Tombol Spasi pada *Keyboard*	Menghentikan *scene video* dan masuk ke dalam informasi misi	[√] Berhasil [ ] Tidak Berhasil
5.	Informasi Misi	Pilih Mainkan Sekarang	Masuk ke dalam *level* permainan	[√] Berhasil [ ] Tidak Berhasil
6.	Permainan	Tekan Tombol W Pada *keyboard*	Karakter pemain bergerak kedepan	[√] Berhasil [ ] Tidak Berhasil
		Tekan Tombol A Pada *keyboard*	Karakter pemain bergerak kekiri	[√] Berhasil [ ] Tidak Berhasil

No	Kasus yang akan di uji	Skenario dan Hasil Uji		
		Skenario Uji	**Hasil yang Diharapkan**	**Kesimpulan**
		Tekan Tombol S Pada *keyboard*	Karakter pemain bergerak kebelakang	[√] Berhasil [ ] Tidak Berhasil
		Tekan Tombol D Pada *keyboard*	Karakter pemain bergerak kekanan	[√] Berhasil [ ] Tidak Berhasil
		Tekan Tombol Kiri Pada Mouse	Karakter pemain menembak	[√] Berhasil [ ] Tidak Berhasil
		Tekan Esc Pada *Keyboard*	Menampilkan pause menu	[√] Berhasil [ ] Tidak Berhasil
		Tekan 1 Pada *Keyboard*	Karakter memakai senjata pistol	[√] Berhasil [ ] Tidak Berhasil
		Tekan 2 Pada *Keyboard*	Karakter memakai senjata machinegun	[√] Berhasil [ ] Tidak Berhasil

4.2.1.3 Kasus dan Hasil Pengujian (*White Box Testing)*

1. Pengujian *White Box* Algoritma A*

Di bawah ini merupakan kasus algoritma A* untuk menguji perangkat lunak yang sudah dibangun dengan menggunakan metode *white box*, berdasarkan skenario pengujian aplikasi yang sudah dijabarkan.

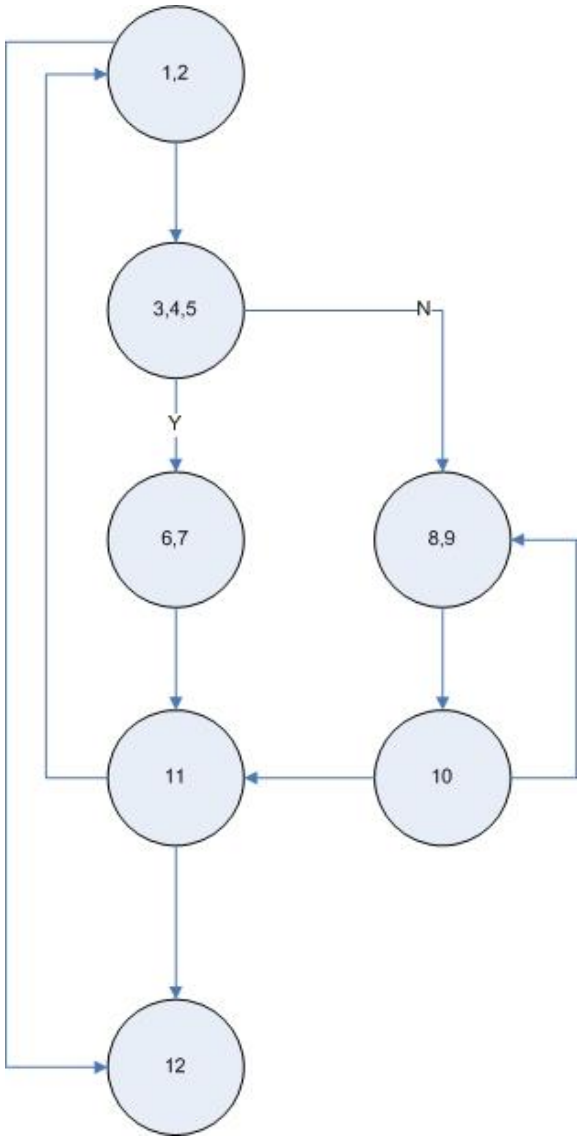
Pengujian Algoritma A* merupakan proses di mana akan diuji terhadap musuh dalam melakukan pencarian rute terpendek yang terdapat di dalam permainan.

a. *Flow Graph* Algoritma A* Prosedure Pencarian Jalan

```
1. openList.Add(hitInfo.collider.GetComponent(PathfindNode)
   ;
2. while(foundTarget==false){
3. checkingNode = GetSmallestFValueNode();
4. AddToClosedList(checkingNode);
5. if (checkingNode == targetNode)
6. foundTarget = true;
7. TraceBackPath(musuh);
8. else for (var node : PathfindNode in checkingNode.dekat)
```

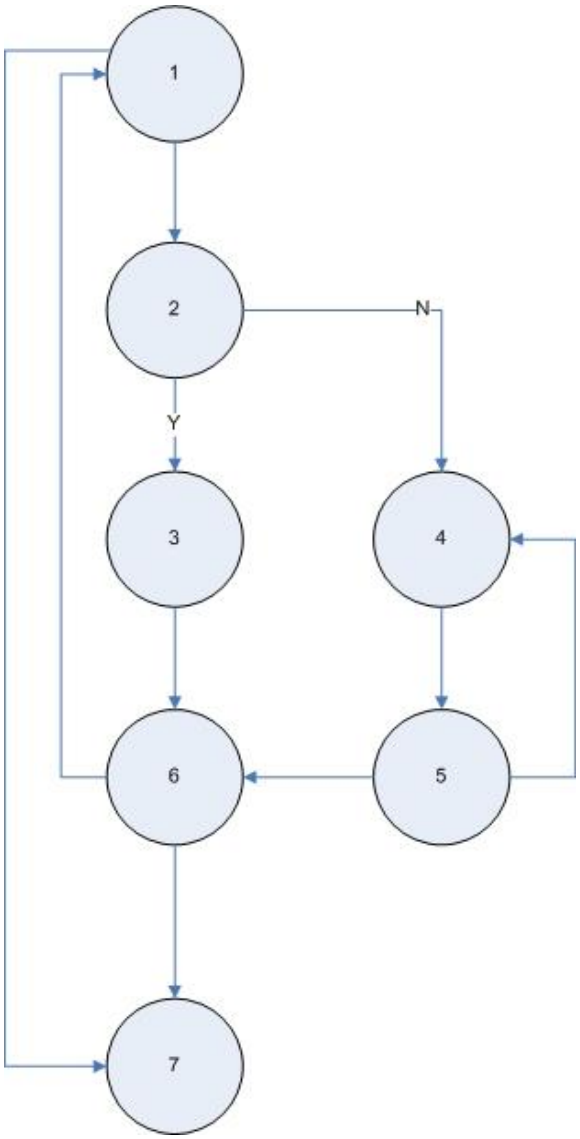
```
9. cekNode(checkingNode, node);
10.   endfor
11.   endif
12.   endwhile
```

Berdasarkan *pseudocode* prosedur pencarian jalan, maka dibuat *flow graph* yang bisa dilihat pada gambar 4.16.



Gambar 4.16 *Flow Graph* Prosedur Pencarian Jalan

Atau disederhanakan berdasarkan kondisi atau simpul sebagai berikut :



Gambar 4.17 Penyederhanaan *Flow Graph* Pencarian Jalan

Keterangan:

= Menggambarkan kondisi

= Menggambarkan aksi

a. *Cyclomatic Complexity* V(G)

E = 9, N = 7

V(G) = E – N + 2

V(G) = 9 – 7 + 2

V(G) = 2 + 2 = 4

Keterangan

E = Jumlah aksi

N = Jumlah Kondisi

b. *Independent Path*

Path 1 = 1-7

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

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

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

c. *Graph Matriks* Algoritma A* Pencarian Jalan

Penomoran ulang *flowgraph* atau penyederhanaan berdasarkan kondisi atau simpul dapat dilihat pada tabel 4.3:

Tabel 4.3 *Graph Matriks* Algoritma A* Pencarian Jalan

N	1	2	3	4	5	6	7	N(E) -1
1		1					1	1
2			1	1				1
3						1		0
4					1			0
5				1		1		1
6	1							0
7								0
							Sum(E)+1	4

Keterangan:

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

d. Hasil Uji Algoritma A* Pencarian Jalan

Tabel 4.4 Hasil Uji Algoritma A* Pencarian Jalan

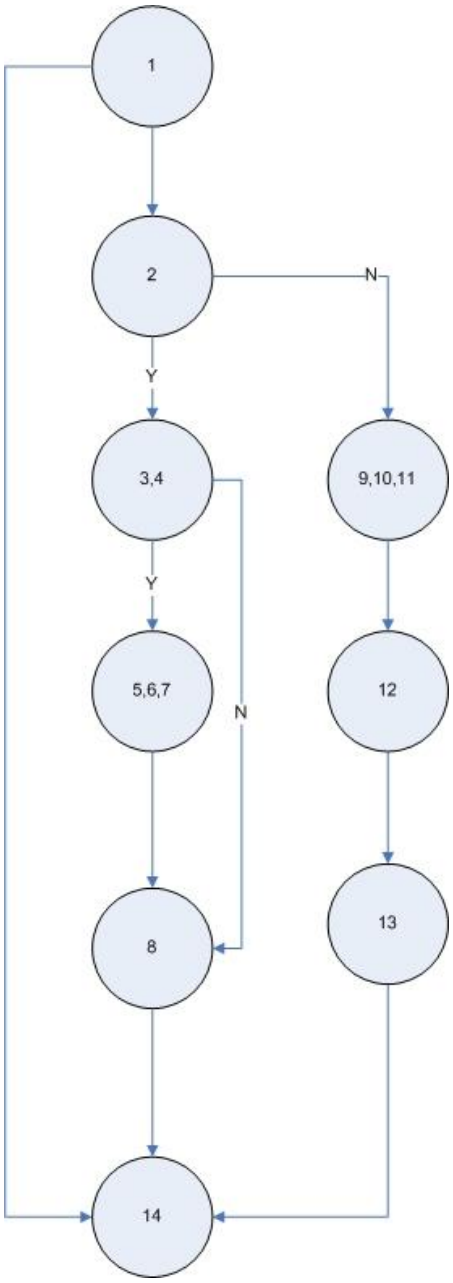
No	Kasus uji	Yang diharapkan	Pengamatan	Keterangan
1	`foundTarget = true`	Pencarian jalan gagal dilakukan	Pencarian jalan tidak dilakukan	[√] Alur terlewati

No	Kasus uji	Yang diharapkan	Pengamatan	Keterangan
				[ ] Alur tidak terlewati
2	- foundTarget = false - CheckingNode = target node	Pencarian jalan dilakukan dan fungsi TraceBackPath dilakukan	Pencarian jalan dilakukan dan fungsi TraceBackPath dilakukan	[√] Alur terlewati [ ] Alur tidak terlewati
3	- foundTarget = false - Checking node != target node	Pencarian jalan dilakukan dan pengecekan *node* dilakukan	Pencarian jalan dilakukan dan pengecekan *node* dilakukan	[√] Alur terlewati [ ] Alur tidak terlewati
4	- foundTarget = false - Checking node != target node - *Node* di checkingNode.dekat masih ada	Pengecekan *node* dilakukan lagi sebanyak jumlah *node* dekat	Pengecekan *node* dilakukan lagi sebanyak *node* dekat	[√] Alur terlewati [ ] Alur tidak terlewati

b. *Flow Graph* Algoritma A* Prosedure Pengecekan *Node* Terdekat

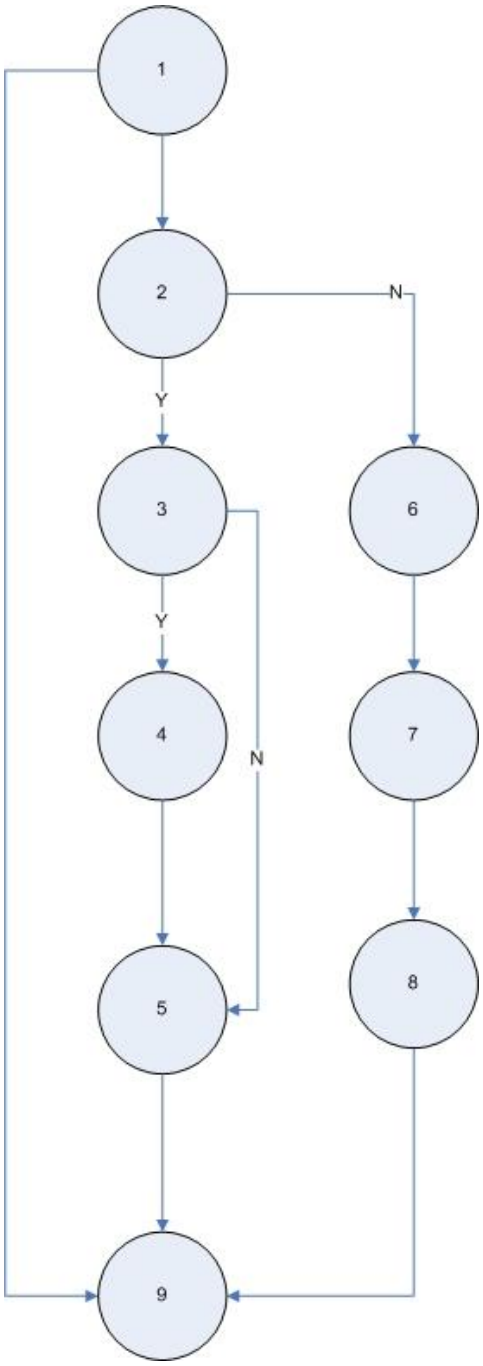
```
1. if (closedList.Contains(testing) == false)
2. then if (openList.Contains(testing) == true)
3. var newGCost : int = currentNode.getGValue() +
   baseMovementCost;
4. if (newGCost < testing.getGValue())
5. then testing.setParent(currentNode);
6. testing.setGValue(newGCost);
7. testing.CalculateFValue();
8. endif
9. else testing.setParent(currentNode);
10.testing.setGValue(currentNode.getGValue()+baseMovem
   entCost);
11.testing.CalculateFValue();
12.AddToOpenList(testing);
13. endif
14.   endif
```

Berdasarkan *pseudocode* prosedur pengecekan *node* terdekat, maka dibuat *flow graph* yang bisa dilihat pada gambar 4.16.



Gambar 4.18 *Flow Graph* Prosedur Pengecekan *Node* Terdekat

Atau disederhanakan berdasarkan kondisi atau simpul sebagai berikut :



Gambar 4.19 Penyederhanaan *Flow graph* Pengecekan *Node* Terdekat

Keterangan:

= Menggambarkan kondisi

= Menggambarkan aksi

a. *Cyclomatic Complexity* V(G)

E = 11, N = 9

V(G) = E – N + 2

V(G) = 11 – 9 + 2

V(G) = 2 + 2 = 4

Keterangan

E = Jumlah aksi

N = Jumlah Kondisi

b. *Independent Path*

Path 1 = 1-9

Path 2 = 1-2-3-5-9

Path 3 = 1-2-3-4-5-9

Path 4 = 1-2-6-7-8-9

c. *Graph Matriks* Algoritma A*

Penomoran ulang *flowgraph* atau penyederhanaan berdasarkan kondisi atau simpul dapat dilihat pada tabel 4.5:

Tabel 4.5 *Graph Matriks* Algoritma A* Pengecekan *Node* Terdekat

N	1	2	3	4	5	6	7	8	9	N(E) -1
1		1							1	1
2			1			1				1
3				1	1					1
4					1					0
5									1	0
6							1			0
7								1		0
8									1	0
9										0
Sum(E)+1										4

Keterangan:

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

d. Hasil Uji Algoritma A*

Tabel 4.6 Hasil Uji Algoritma A* Pengecekan *Node* Terdekat

No	Kasus uji	Yang diharapkan	Pengamatan	Keterangan
1	- `closedList.Contains(testing) = true`	Pengecekan *node* tidak dilakukan	Pengecekan *node* gagal dilakukan	[√] Alur terlewati [ ] Alur tidak terlewati
2	- `closedList.Contains(testing) = false` - `openList.Contains(testing) = true` - `newGCost > testing.getGValue()`	Hanya dilakukan perhitungan nilai g *cost* baru	Hanya dilakukan perhitungan nilai g *cost* baru	[√] Alur terlewati [ ] Alur tidak terlewati
3	- `closedList.Contains(testing) = false` - `openList.Contains(testing) = true` - `newGCost < testing.getGValue()`	Dilakukan perhitungan nilai g *cost* baru dan dilakukan penggantian *parent* dengan g *cost* yang lebih baik	Perhitungan nilai g *cost* baru dilakukan dan penggantian *parent* dengan g *cost* yang lebih baik	[√] Alur terlewati [ ] Alur tidak terlewati
4	- `closedList.Contains(testing) = false` - `openList.Contains(testing) = false` - `openList.Count < limit`	Dilakukan perhitungan nilai g dan f serta pemindahan *node* ke *open list*	Terjadi pemindahan *node* ke *open list* dengan perhitungan nilai g dan f-nya	[√] Alur terlewati [ ] Alur tidak terlewati

c. Kesimpulan

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

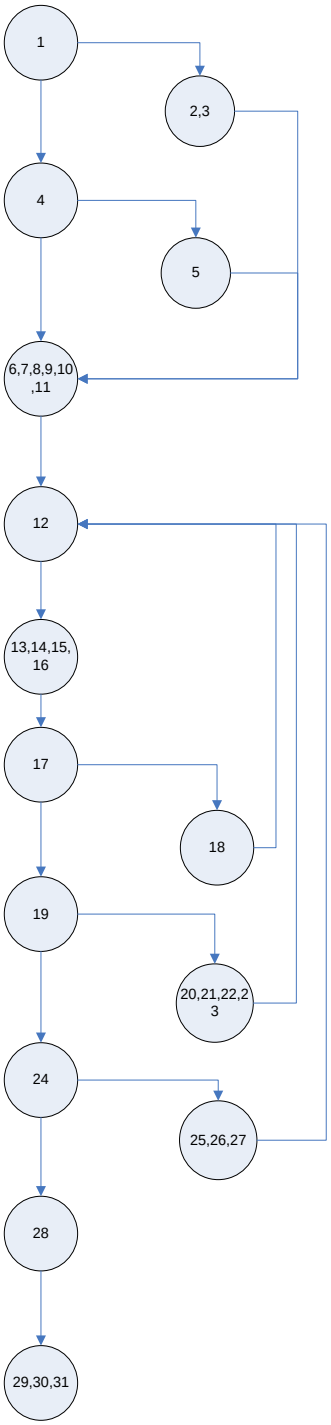
2. Pengujian *White Box* Algoritma *Fuzzy Logic*

Berikut ini merupakan algoritma *fuzzy logic* yang diterapkan pada *level* 1.

```
1.  nilaiJarak = jrk;
2.  fungsiKeanggotaanTrapesium ftrapesium = new
    fungsiKeanggotaanTrapesium();
3.  fungsiKeanggotaanSegitiga fsegitiga = new
    fungsiKeanggotaanSegitiga();
4.  nilaiKekuatan = kkt;
5.  fungsiKeanggotaanTrapesium ftrapesium = new
    fungsiKeanggotaanTrapesium();
6.  return nilaiImplikasi = Mathf.Min(n1, n2);
7.  implikasi imp = new implikasi();
8.  jarak jrk = new jarak(nilaiJarak);
9.  kekuatan kkt = new kekuatan(nilaiKekuatan);
10. jrk.periksaJarak();
11. kkt.periksaKekuatan();
12. jarak = (target.position -
    myTransform.position).magnitude;
13. rules rule = new rules(jarak,hitPoints);
14. defuzzyfikasi def = new defuzzyfikasi();
15. rule.periksaRule();
16. def. hasilDefuzzy();
17. if((balikKeMarkas.position -
    myTransform.position).magnitude < 1.0){
18. StartCoroutine(tambahDarah());}
19. else if(def.hasilDefuzzyfikasi <= 12.0f) {
20. balikKeMarkas();
21. myTransform.position += myTransform.forward * moveSpeed
    * Time.deltaTime;
22. anim.animation.CrossFade(walkAnim);
23. playWalkSounds();}
24. else if(def.hasilDefuzzyfikasi > 8.0f &&
    def.hasilDefuzzyfikasi <= 22.0f){
25. bertahan();
26. anim.animation.CrossFade(attackAnim2);
27. audio.Stop();}
28. else if(def.hasilDefuzzyfikasi > 18.0f &&
    def.hasilDefuzzyfikasi <= 30.0f){
29. menyerang();
30. damage = Random.Range(20, 30);
31. anim.animation.CrossFade(walkAnim);}
```

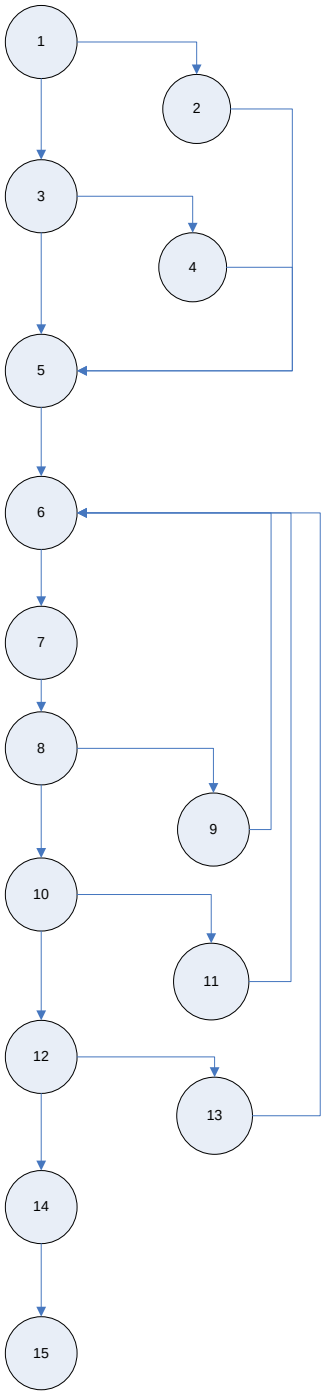
a. *Flow Graph* Algoritma *Fuzzy Logic*

Berdasarkan *pseudocode* di atas maka *flow graph* algoritma *fuzzy logic* adalah sebagai berikut:



Gambar 4.20 *Flow Graph* Algoritma *Fuzzy Logic*

Atau disederhanakan berdasarkan kondisi atau simpul sebagai berikut :



Gambar 4.21 Penyederhanaan *Flow Graph* Algoritma *Fuzzy Logic*

Keterangan:



= Menggambarkan kondisi

= Menggambarkan aksi

b. *Cyclomatic Complexity* V(G)

Cyclomatic complexity akan digunakan dalam menguji path yang nantinya akan digunakan untuk mencari jumlah path dalam satu dan juga sebagai penentuan jalur atau path pencarian NPC terhadap player. Apabila menggunakan basis path, nilai yang dihitung untuk Cyclomatic complexity menentukan jalur indpenden dalam basis set suatu program dan memberi batas atas untuk jumlah uji coba yang harus dikerjakan bahwa seluruh perintah sekurang-kurangnya telah dikerjakan sekali.

Berikut ini adalah perhitungan Cyclomatic complexity yang nantinya akan digunakan sebagai jumlah aksi pada independent path.

E = 19, N = 15

V(G) = E – N + 2

V(G) = 19 – 15 + 2

V(G) = 4 + 2 = 6

Keterangan

E = Jumlah aksi

N = Jumlah Kondisi

c. *Independent Path*

Path 1 = 1-2-5

Path 2 = 1-3-4-5

Path 3 = 1-3-5-6-7-8-9-6-15

Path 4 = 1-3-5-6-7-8-10-11-6-15

Path 5 = 1-3-5-6-7-8-10-12-13-6-15

Path 6 = 1-3-5-6-7-8-10-12-14-15

d. *Graph Matriks* Algoritma *Fuzzy Logic*

Graph metrik ini akan membantu uji coba independent path. Karena matrik persegi empat ini mempunyai ukuran yang sama dengan jumlah node pada flowgraph. Masing-masing baris dan kolom mempunyai hubugan dengan node yang telah ditentukan.

Penomoran ulang *flowgraph* atau penyederhanaan berdasarkan kondisi atau simpul dapat dilihat pada tabel 4.7 di bawah ini.:

Tabel 4.7 *Graph Matriks* Algoritma *Fuzzy Logic*

N	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	N(E) -1
1		1	1													1
2					1											0
3				1	1											1
4					1											0
5						1										0
6							1									0
7								1								0
8									1	1						1
9						1										0
10											1	1				1
11						1										0
12													1	1		1
13						1										0
14															1	0
15																0
															Sum(E)+1	6

Keterangan:

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

e. Hasil Uji Algoritma *Fuzzy Logic*

Tabel 4.8 Hasil Uji Algoritma *Fuzzy Logic*

No	Kasus uji	Yang diharapkan	Hasil Sesuai Kasus Uji	Keterangan
1	- `nilaiJarak = jrk;` - `fungsiKeanggotaanTrapesium ftrapesium = new fungsiKeanggotaanTrapesium();`	Mencari nilai jarak dengan menggunakan fungsi	nilai jarak didapat dengan menggunakan fungsi	[√] Alur terlewati [ ] Alur tidak terlewati

No	Kasus uji	Yang diharapkan	Hasil Sesuai Kasus Uji	Keterangan
	- `fungsiKeanggotaanSegitiga fsegitiga = new fungsiKeanggotaanSegitiga();` - `nilaiKekuatan = kkt;`	keanggotaan trapesium dan segitiga	keanggotaan trapesium dan segitiga	
2	- `nilaiKekuatan = kkt;` - `fungsiKeanggotaanTrapesium ftrapesium = new fungsiKeanggotaanTrapesium();`	Mencari nilai kekuatan dengan menggunakan fungsi keanggotaan trapesium	Nilai kekuatan di dapat dengan menggunakan fungsi keanggotaan trapesium	[√] Alur terlewati [ ] Alur tidak terlewati
3	- `balikKeMarkas.position myTransform.position.magnitude < 1.0` - `StartCoroutine(tambahDarah());`	NPC menambah *hitpoints* saat kembali ke markas	NPC menambah *hitpoints*	[√] Alur terlewati [ ] Alur tidak terlewati
4	- `balikKeMarkas.position myTransform.position.magnitude < 1.0` - `def.hasilDefuzzyfikasi <= 10.0f`	Hasil defuzzy <= 12 NPC melarikan diri ke markas	NPC melarikan diri ke markas	[√] Alur terlewati [ ] Alur tidak terlewati
5	- `balikKeMarkas.position myTransform.position.magnitude < 1.0` - `def.hasilDefuzzyfikasi > 8.0f && def.hasilDefuzzyfikasi <= 22.0f` - `bertahan();`	Hasil defuzzy > 8 dan <= 22 NPC bertahan	NPC bertahan	[√] Alur terlewati [ ] Alur tidak terlewati
6	- `def.hasilDefuzzyfikasi > 18.0f && def.hasilDefuzzyfikasi <= 30.0f` - `menyerang();` - `damage =`	Hasil defuzzy > 18 dan <= 30 NPC menyerang	NPC menyerang	[√] Alur terlewati [ ] Alur tidak terlewati

No	Kasus uji	Yang diharapkan	Hasil Sesuai Kasus Uji	Keterangan
	Random.Range(20, 30); - anim.animation.CrossFade(walkAnim);			

f. Kesimpulan

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

4.2.1.4 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 kuesioner mengenai kepuasa pengguna atas aplikasi yang telah dibangun. Adapun metode penilaian pengujian yang digunakan adalah metode kuantitatif berdasarkan data dari pengguna.

4.2.3 Kuesioner

Kuesioner disebarkan menggunakan teknik *Simple Random Sampling* dimana anggota *sample* dari populasi dilakukan secara acak tanpa memperhatikan strata yang ada dalam populasi. Kuesioner disebarkan kepada 30 responden dengan usia minimal 15 tahun, dari hasil kuesioner tersebut akan dilakukan perhitungan agar dapat diambil kesimpulan terhadap penilaian penerapan sistem yang telah dibangun. Kuesioner terdiri dari 8 pernyataan dengan menggunakan Sistem penskoran dengan skala pengukuran yang digunakan adalah skala likert, Berikut skor untuk jawaban kuesioner, dapat di lihat pada tabel 4.7.

Tabel 4.9 Skor Jawaban Pada Kuesioner

Skala Jawaban	Keterangan	Skor
SS	Sangat Setuju	4
ST	Setuju	3
RG	Ragu-ragu	2
TS	Tidak Setuju	1

Data hasil kuisioner yang telah diberikan dapat dicari presentasinya dari masing-masing jawaban dengan menggunakan rumus :

$$P = \frac{S}{skorideal} x100\% \quad (4.1)$$

Keterangan:

P = Nilai presentase yang dicari

S = Jumlah frekuensi dikalikan dengan nilai yang di tetapkan jawaban

Skor ideal = Nilai tertinggi dikalikan dengan jumlah sampel

Berikut ini adalah hasil persentase masing-masing jawaban yang sudah dihitung nilainya dengan menggunakan rumus diatas.

1. Apakah anda setuju sejarah pertempuran lima hari di semarang perlu dipelajari dan diketahui?

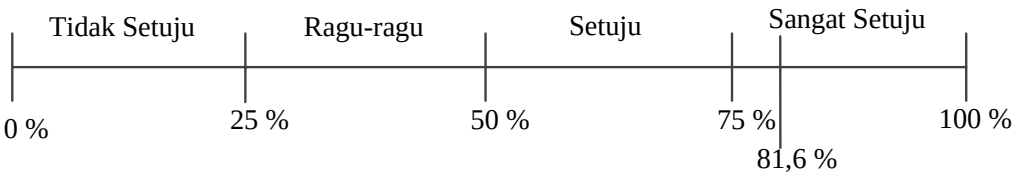
Tabel 4.10 Perhitungan Persentase Pertanyaan ke-1

Pertanyaan	Keterangan	Skor	Frekuensi	S
1.	Sangat setuju	4	13	52
	Setuju	3	12	36
	Ragu-ragu	2	5	10
	Tidak setuju	1	0	0
Jumlah			30	98

$$P = \frac{98}{120} x100 = 81{,}6\%$$

Berdasarkan hasil presentase nilai di atas, maka dapat disimpulkan bahwa penilaian terhadap pertanyaan tentang setuju tidakkah sejarah pertempuran lima

hari di semarang perlu dipelajari dan diketahui adalah 81,6% dari yang diharapkan 100%, atau dapat dikategorikan sebagai sangat setuju. Berikut kriteria interprestasi skor :



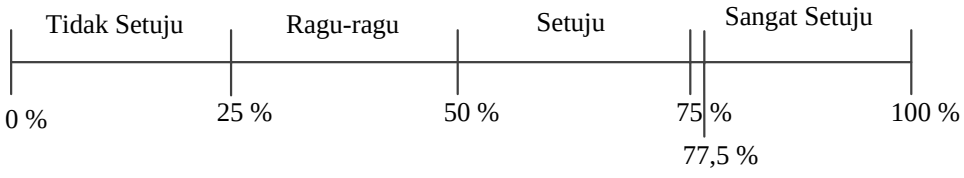
2. Apakah anda setuju media pembelajaran dalam bentuk *game*?

Tabel 4.11 Tabel Perhitungan Persentase Pertanyaan Ke-2

Pertanyaan	Keterangan	Skor	Frekuensi	S
2.	Sangat setuju	4	10	40
	Setuju	3	13	39
	Ragu-ragu	2	7	14
	Tidak setuju	1	0	0
Jumlah			30	93

$$P = \frac{93}{120} x100 = 77{,}5\%$$

Berdasarkan hasil presentase nilai di atas, maka dapat disimpulkan bahwa penilaian terhadap pertanyaan tentang setuju tidakkah media pembelajaran dalam bentuk *game* adalah 77,5% dari yang diharapkan 100%, atau dapat dikategorikan sangat setuju. Berikut kriteria interprestasi skor :



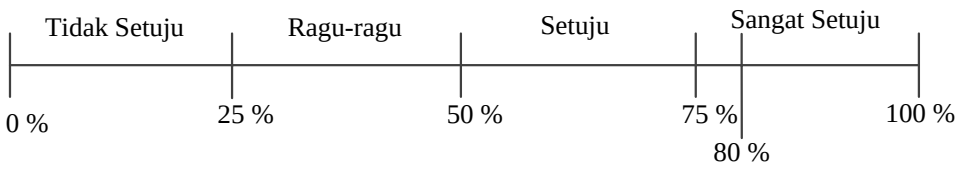
3. Apakah anda setuju sejarah pertempuran lima hari di semarang dijadikan sebuah *game* ?

Tabel 4.12 Tabel Perhitungan Persentase Pertanyaan Ke-3

Pertanyaan	Keterangan	Skor	Frekuensi	S
3.	Sangat setuju	4	11	44
	Setuju	3	14	42
	Ragu-ragu	2	5	10
	Tidak setuju	1	0	0
Jumlah			30	96

$$P = \frac{96}{120} x100 = 80\%$$

Berdasarkan hasil presentase nilai di atas, maka dapat disimpulkan bahwa penilaian terhadap pertanyaan tentang setuju tidakkah pertempuran lima hari di semarang dijadikan sebuah *game* adalah 80% dari yang diharapkan 100%, atau dapat dikategorikan sebagai sangat setuju. Berikut kriteria interprestasi skor :



4. Apakah anda setuju dengan bermain *game* anda dapat mengenal peristiwa sejarah pertempuran lima hari di semarang?

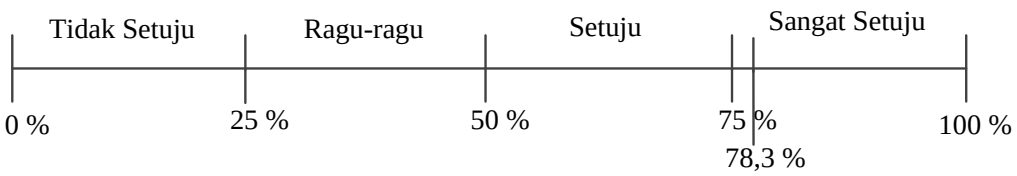
Tabel 4.13 Tabel perhitungan Persentase Pertanyaan Ke-4

Pertanyaan	Keterangan	Skor	Frekuensi	S
4.	Sangat setuju	4	11	44
	Setuju	3	12	36
	Ragu-ragu	2	7	14
	Tidak setuju	1	0	0
Jumlah			30	94

$$P = \frac{94}{120} x100 = 78{,}3\%$$

Berdasarkan hasil presentase nilai di atas, maka dapat disimpulkan bahwa dengan bermain *game* anda dapat mengenal peristiwa sejarah pertempuran lima

hari di semarang adalah 78,3% dari yang diharapkan 100%, atau dapat dikategorikan sebagai sangat setuju. Berikut kriteria interprestasi skor :



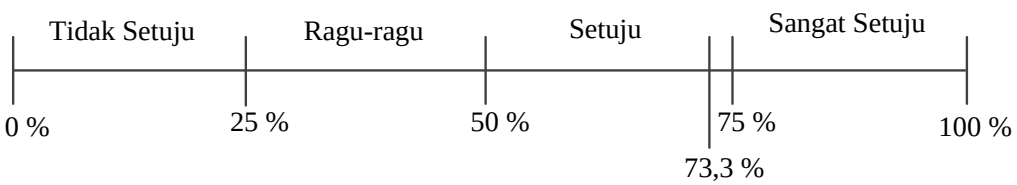
5. Apakah anda setuju *game* yang dibangun telah sesuai dengan cerita sejarah pertempuran lima hari di semarang?

Tabel 4.14 Perhitungan Persentase Pertanyaan Ke-5

Pertanyaan	Keterangan	Skor	Frekuensi	S
5.	Sangat setuju	4	8	32
	Setuju	3	12	36
	Ragu-ragu	2	10	20
	Tidak setuju	1	0	0
Jumlah			30	88

$$P = \frac{88}{120} x100 = 73{,}3\%$$

Berdasarkan hasil presentase nilai di atas, maka dapat disimpulkan bahwa *game* yang dibangun telah sesuai dengan cerita sejarah pertempuran lima hari di semarang adalah 73,3% dari yang diharapkan 100%, atau dapat dikategorikan sebagai setuju. Berikut kriteria interprestasi skor :



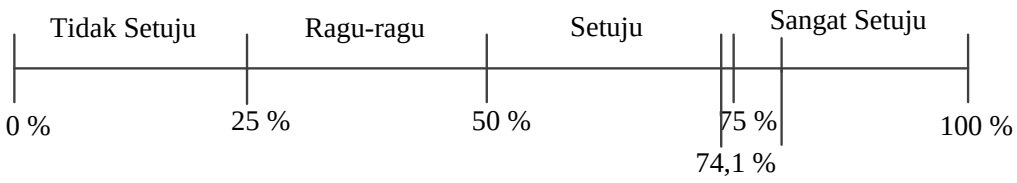
6. Apakah anda setuju materi atau informasi tentang sejarah pertempuran lima hari di semarang yang terdapat pada *game* ini mudah untuk diingat?

Tabel 4.15 Perhitungan Persentase Pertanyaan Ke-6

Pertanyaan	Keterangan	Skor	Frekuensi	S
6.	Sangat setuju	4	7	28
	Setuju	3	15	45
	Ragu-ragu	2	8	16
	Tidak setuju	1	0	0
Jumlah			30	89

$$P = \frac{89}{120} x100 = 74{,}1\%$$

Berdasarkan hasil presentase nilai di atas, maka dapat disimpulkan bahwa materi atau informasi yang diberikan pada *game* Pertempuran Lima Hari di Semarang mudah untuk diingat adalah 74,1% dari yang diharapkan 100%, atau dapat dikategorikan sebagai setuju. Berikut kriteria interprestasi skor :



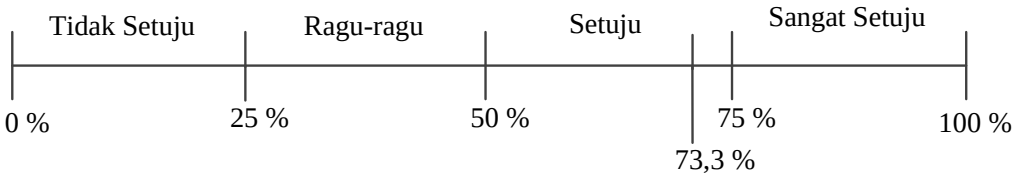
7. Apakah anda setuju *game* sejarah Pertempuran lima hari di semarang mudah untuk dimainkan?

Tabel 4.16 Perhitungan Persentase Pertanyaan Ke-7

Pertanyaan	Keterangan	Skor	Frekuensi	S
7.	Sangat setuju	4	9	36
	Setuju	3	10	30
	Ragu-ragu	2	11	22
	Tidak setuju	1	0	0
Jumlah			30	88

$$P = \frac{88}{120} x100 = 73{,}3\%$$

Berdasarkan hasil presentase nilai di atas, maka dapat disimpulkan bahwa *game* sejarah Pertempuran lima hari di semarang mudah untuk dimainkan adalah 73,3% dari yang diharapkan 100%, atau dapat dikategorikan setuju. Berikut kriteria interprestasi skor :



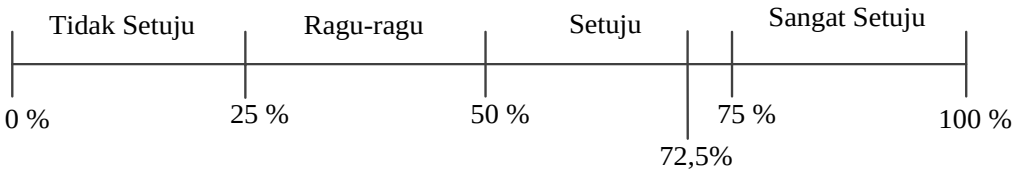
8. Apakah anda setuju musuh dalam permainan *game* ini sulit untuk dikalahkan?

Tabel 4.17 Perhitungan Persentase Pertanyaan Ke-8

Pertanyaan	Keterangan	Skor	Frekuensi	S
8.	Sangat setuju	4	9	36
	Setuju	3	9	27
	Ragu-ragu	2	12	24
	Tidak setuju	1	0	0
Jumlah			30	87

$$P = \frac{87}{120} x100 = 72{,}5\%$$

Berdasarkan hasil presentase nilai di atas, maka dapat disimpulkan bahwa musuh dalam permainan *game* ini sulit untuk dikalahkan adalah 72,5% dari yang diharapkan 100%, atau dapat dikategorikan setuju. Berikut kriteria interprestasi skor :



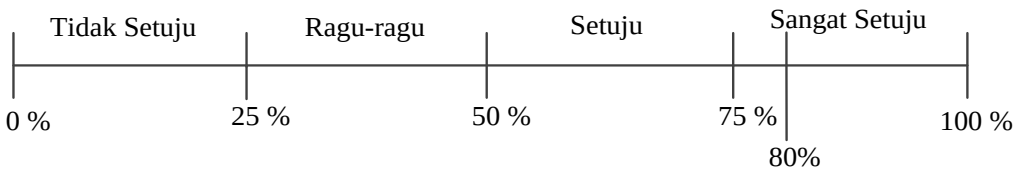
9. Apakah anda setuju secara umum tampilan *game* Pertempuran lima hari di semarang ini sudah menarik?

Tabel 4.18 Perhitungan Persentase Pertanyaan Ke-9

Pertanyaan	Keterangan	Skor	Frekuensi	S
9.	Sangat setuju	4	11	44
	Setuju	3	14	42
	Ragu-ragu	2	5	10
	Tidak setuju	1	0	0
Jumlah			30	96

$$P = \frac{96}{120} x100 = 80\%$$

Berdasarkan hasil presentase nilai di atas, maka dapat disimpulkan bahwa secara umum tampilan *game* Pertempuran lima hari di semarang ini sudah menarik adalah 80% dari yang diharapkan 100%, atau dapat dikategorikan sangat setuju. Berikut kriteria interprestasi skor :



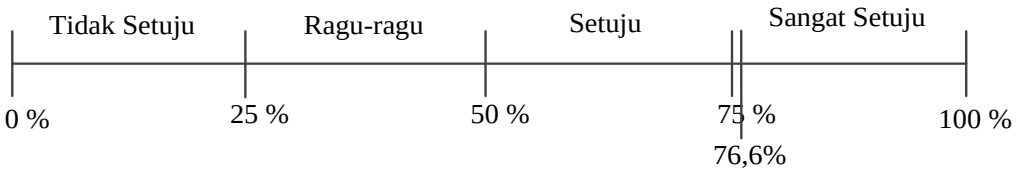
10. Apakah anda setuju setelah memainkan *game* ini anda merasa terhibur?

Tabel 4.19 Perhitungan Persentase Pertanyaan Ke-10

Pertanyaan	Keterangan	Skor	Frekuensi	S
10.	Sangat setuju	4	9	36
	Setuju	3	14	42
	Ragu-ragu	2	7	14
	Tidak setuju	1	0	0
Jumlah			30	92

$$P = \frac{92}{120} x100 = 76{,}6\%$$

Berdasarkan hasil presentase nilai di atas, maka dapat disimpulkan bahwa setelah memainkan *game* ini anda merasa terhibur adalah 76,6% dari yang diharapkan 100%, atau dapat dikategorikan sangat setuju. Berikut kriteria interprestasi skor :



4.2.4.3 Kesimpulan Pengujian Beta

Berdasarkan hasil kuesioner yang didapatkan dari pengujian beta bahwa responden sangat setuju sejarah pertempuran lima hari di semarang perlu dipelajari dan diketahui serta media pembelajarannya dalam bentuk game. Sejarah pertempuran lima hari di semarang yang akan dijadikan ke dalam bentuk *game* diharapkan dapat mengenal peristiwa sejarah pertempuran lima hari di Semarang, dalam *game* pertempuran lima hari di semarang ini di dalamnya terdapat cerita atau informasi tentang peristiwa sejarah Pertempuran Lima Hari Di Semarang. Musuh yang terdapat di dalam *game* pertempuran lima hari di semarang mempunyai tingkat kesulitan yang berbeda disetiap level, tampilan *game* pertempuran lima hari di Semarang secara umum sudah menarik dan pengguna merasa terhibur setelah memainkan *game* tersebut. Jadi, secara keseluruhan responden setuju menjadikan *game* sebagai salah satu alternatif untuk media pembelajaran.