BAB 4
IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi

Proses implementasi adalah tahapan dimana aplikasi yang sudah dibangun dan di rancang lalu di uji kelayakannya. Pemograman pada game ini menggunakan Construct 2, Selanjutnya dioperasikan sebagaimana mestinya sesuai fungsi dan kelayakannya untuk digunakan.

4.2 Perangkat Pendukung yang Digunakan

Dalam proses pembangunan 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

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

1. CPU : Intel Core i5 – 2430M 2.4GHz
2. Memory : 2 GB
3. VGA : AMD Radeon HD 6730M 2GB
4. Keyboard
5. Mouse
6. Speaker

4.2.2 Perangkat Lunak

Perangkat lunak yang digunakan untuk implementasi dan pengujian Game Edukasi Menjaga dan Memelihara Lingkungan menggunakan browser Google Chrome dan Mozilla Firefox.

4.3 Implementasi Aplikasi

Cara menjalankan Game Edukasi Mejaga dan memelihara Lingkungan ini hanya tinggal klik 2 kali pada index.html yang terdapat pada folder Game Edukasi Menjaga dan Memelihara Lingkungan Hidup.

4.4 Implementasi Antar Muka

4.4.1 Menu Utama



Gambar 4.1 Tampilan Menu Utama

Layar ini menampilkan menu utama yang di mana di dalamnya terdapat pilihan Main, Tentang Lingkungan, Tutorial, Keluar.

4.4.2 Tampilan Tentang Lingkungan



Gambar 4.2 Menu Tentang Lingkungan

Dalam menu Tentang Lingkungan terdapat 2 pilihan yaitu tentang Lingkungan Bersih dar Kotor dan Lingkungan Bersih dari Sampah.

4.4.3 Tampilan Lingkungan Bersih Dari kotor



Gambar 4.3 Tentang Kebersihan Lingkungan dari Kotor

Layar ini menjelaskan tentang lingkungan kotor dan cara kita mencegah agar lingkungan itu tidak kotor.

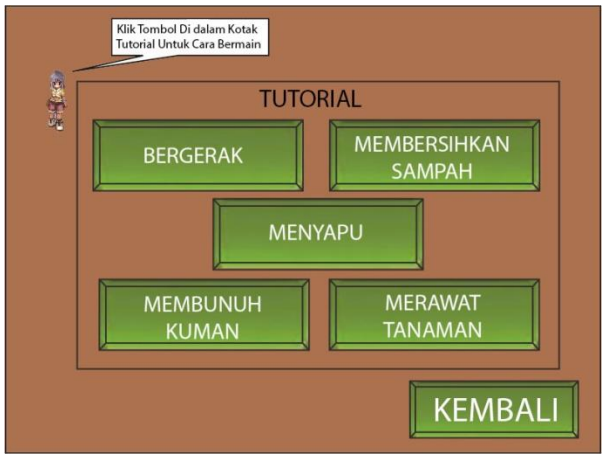
4.4.4 Tampilan Lingkungan Bersih dari Sampah



Gambar 4.4 Tentang Kebersihan Lingkungan dari Sampah

Layar ini menjelaskan tentang lingkungan kotor dan cara kita mencegah agar lingkungan itu tidak kotor.

4.4.5 Tampilan Tutorial



Gambar 4.5 Tutorial

Ini adalah tampilan Tutorial dalam game yang dimana terdapat tutorial bergarak, membersihkan sampah, menyapu, membunuh kuman, merawat tanaman.

4.4.6 Tampilan Turorial Bergerak



Gambar 4.6 Tutorial Bergerak

Layar ini menggambarkan tutorial bergerak agar user mengetahui bagaimana cara menggerakan karakter pada game ini

4.4.7 Tampilan Turorial Membunuh Kuman



Gambar 4.7 Tutorial Membunuh Kuman

Pada Layar ini akan di jelaskan bagaimana cara pemain membunuh kuman yang akan berada pada game ini.

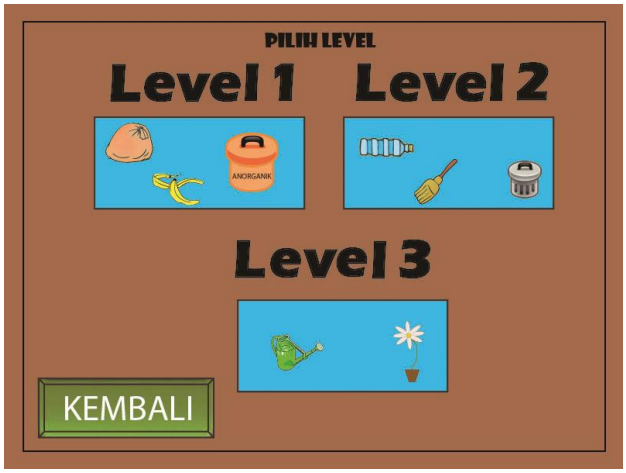
4.4.8 Tampilan Turorial Membersihkan Sampah



Gambar 4.8 Tutorial Membersihkan Sampah

Pada Layar ini akan di jelaskan bagaimana cara pemain membuang sampah yang akan berada pada game ini.

4.4.9 Tampilan Pilih Level



Gambar 4.9 Pilih Level

Layar ini menampilkan pilihan level pada game ini bila belum menyelesailkan level 1 maka pemain tidak akan bisa lanjut pada level selanjutnya

4.4.10 Tampilan Permainan



Gambar 4.10 Permainan

Layar ini menampilkan permainan dalam game ini yang di mana ter dapat karakter yang akan menyelesaikan misinya untuk membersihkan lingkungan

4.4.11 Tampilan Memenangkan Game



Gambar 4.11 Menang

Berikut adalah tampilan pemain yang telah menyelesaikan misi pada setiap game

4.4.12 Tampilan Kalah



Gambar 4.12 Game Over

Berikut adalah tampilan bila pemain gagal menyelesaikan misi.

4.5 Pengujian

Pengujian adalah hal yang perlu dilakukan untuk mengetahui 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.

4.5.1 Pengujian Alpha

Pengujian alpha dilakukan pada sisi pengembang yang dianggap pemakai dan merekam semua kesalahan dan masalah pemakaian. Pengujian alpha dilakukan pada sebuah lingkungan yang terkendali.

Pendekatan perancangan kasus uji menggunakan dua metode, yaitu white box dan Black Box Testing. White box adalah metode perancangan kasus uji yang menggunakan struktur kontrol program. Black box adalah metode pengujian yang berfokus pada persyaratan fungsional perangkat lunak.

4.5.1.1 Rencana Pengujian

Rencana pengujian yang akan dilakukan pada aplikasi ini selengkapnya terlihat pada Tabel 4.1

Tabel 4.1 Rencana Pengujian Aplikasi

No	Komponen Yang Diuji	Skenario Pengujian	Jenis Pengujian
1	Menu Utama	Memilih Main	Black Box
		Memilih Tentang Lingkungan	Black Box
		Memilih Turorial	Black Box
		Memilih Keluar	Black Box
2	Menu Main	Memlih Level	Black Box
		Mengklik tombol kiri mouse	Black Box
		Mengurangi darah karakter	Black Box
		Mendapatkan Nilai	Black Box
3	Menu Petunjuk	Memilih petunjuk bergerak	Black Box
		Memilih petunjuk membersihkan sampah	Black Box
		Memilih petunjuk membunuh kuman	Black Box
		Memilih petunjuk menyapu	Black Box
		Memlilih petunjuk menyiram	Black Box

		tanaman	
4	Menu Tentang Lingkungan	Memeilih lingkungan bersih dari sampah	Black Box
		Memilih lingkungan bersih dari kotor	Black Box
5	Menggerakan Karakter	Menekan tombol atas keyboard	Black Box
		Menekan tombol bawah keyboard	Black Box
		Menekan tombol kiri keyboard	Black Box
		Menekan tombol kanan keyboard	Black Box
6	Menembak kuman	Mengambil antiseptik	Black Box
		Menembakan antiseptik dengan tombol spasi keyboard	Black Box
7	Membuang sampah	Collision Dengan objek keresek sampah dan Sampah	White Box
		Ambil keresek sampah	Black Box
		Ambil sampah	Black Box
		Drag sampah ke keresek sampah	Black Box
		Drag kresek sampah ke tong	Black Box

		sampah	
		Mendapat nilai	Black Box
8	Menyapu	Collision dengan Sapu	White Box
		Ambil sapu	Black Box
		Gunakan sapu	Black Box
		Tahan dan hapus tempat kotor	Black Box
		Mendapat nilai	Black Box
9	Menyiram tanaman	Collision dengan Penyiram	White Box
		Ambil penyiram	Black Box
		Gunakan penyiram	Black Box
		Drag siraman ke tumbuhan	Black Box
		Mendapat nilai	Black Box
10	Melawan Kuman	Menembakan antiseptik	Black Box
		Algoritma A*	White Box
		Mendapat nilai	Black Box

4.5.1.2 Pengujian Black Box

Tabel 4.2 Pengujian Black Box

No	Kasus yang Di uji	Skenario Uji	Hasil Yang Diharapkan	Kesimpulan
1	Menu Utama	Memilih Main	Sistem Menampilkan Pilihan Stage	Berhasil
		Memilih Tutorial	Menampilkan Turorial	Berhasil

		Memilih Tentang Lingkungan	Menampilkan plihan Tentang Lingkungan	Berhasil
		Memilih Menu Keluar	Keluar Dari game	Berhasil
2	Permainan	Menekan tombol 1	Peralatan ke 1 dipakai	Berhasil
		Menekan tombol 2	Peralatan ke 2 dipakai	Berhasil
		Menekan tombol 3	Peralatan ke 3 dipakai	Berhasil
		Menekan tombol 4	Peralatan ke 4 dipakai	Berhasil
		Menekan Tombol Spasi	Karakter menembak	Berhasil
		Drag Item Sampah ke Keresek Sampah	Sampah masuk ke keresek sampah	Berhasil
		Drag Keresek Sampah yang berisi sampah ke tong sampah	Keresek sampah masuk ke tong sampah	Berhasil
		Gesek tempat kotor oleh sapu	Tempat kotor menjadi bersih	Berhasil
		Drag penyiram ke Tanaman	Tanaman menjadi segar kembali	Berhasil
3	Pergerakan karakter	Menekan Atas	Bergerak ke atas	Berhasil
		Menekan Bawah	Bergerak ke bawah	Berhasil
		Menekan Kanan	Bergerak ke kanan	Berhasil
		Menekan Kiri	Bergerak ke kanan	Berhasil
4	Menembak	Menekan Spasi	Keluar Tembakan	Berhasil
5	Membuang sampah	Ambil sampah dengan arah	Collision terjadi	Berhasil
		Drag Sampah ke tong Sampah	Sampah masuk ke tong sampah	Berhasil
6	Menmbersihkan	Gunakan Sapu	Sapu terpakai	Bershasil

	tempat kotor	Gesek sapu ke tempat kotor	Tempat kotor hilang	Berhasil
7	Meyiram tanaman	Gunakan penyiram tanaman	Penyiram terpakai	Berhasil
		Drag ke tanaman	Tanaman Segar kembali	Berhasil

4.5.1.3 Pengujian White Box

Berikut ini adalah kasus menguji perangkat lunak yang telah dibangun menggunakan metode white-box berdasarkan tabel 4.3. Pengujian white-box menggunakan flow graph yang digunakan untuk menggambarkan alur dari algoritma dan grap matrix yang digunakan untuk menggenerasi flow graph. Berikut pengujian white-box dapat dilihat pada tabel 4.2

Tabel 4.2 Pengujian White Box

No	Coding

1	`function node()`
	`{`
	`this.parent = null;`
	
	`this.x = 0;`
	`this.y = 0;`
	
	`this.f = 0;`
	`this.g = 0;`
	`this.h = 0;`

2	`var nodeCache = []`

3	`function freeNode(n)`
	`{`
	`if (nodeCache.length < 64000)`
	`nodeCache.push(n);`
	`};`
	
	`function resultNode(x_, y_)`
	`{`
	`this.x = x_ || 0;`
	`this.y = y_ || 0;`
	`};`

4	`function allocNode()`
	`{`
	`var ret;`

5	else
	
	ret = new node();
	ret.parent = null;
	ret.x = 0;
	ret.y = 0;
	ret.f = 0;
	ret.g = 0;
	ret.h = 0;
	return ret;
	};

6	function freeNode(n)
	{
	if (nodeCache.length < 64000)
	nodeCache.push(n);
	};
	
	function resultNode(x_, y_)
	{
	this.x = x_ || 0;
	this.y = y_ || 0;
	};

7	function allocResultNode()
	{
	if (resultNodeCache.length)
	return resultNodeCache.pop();
	else
	return new resultNode(0, 0);
	};

8	function freeResultNode(n)
	{
	if (resultNodeCache.length < 10000)
	resultNodeCache.push(n);
	};

9	myInstance = new pathfinder();
10	myInstance.init(d.hcells, d.vcells, d.data, d.diagonals); } else if (d.cmd === "find")

11	myInstance.pathEnd.parent = null;
	myInstance.targetX = d.endX;
	myInstance.targetY = d.endY;
	
	if
	(myInstance.doLongFindPath(d.startX, d.startY))
	{
	self.postMessage({ cmd:
	"result", pathList: myInstance.pathList });

12	function pathfinder()

	{
	this.hcells = 0;
	this.vcells = 0;

13	this.pathEnd = allocNode();
	
	this.cells = null;
	this.openList = [];
	this.closedList = [];
	this.closedCache = {};
	this.pathList = [];
	this.currentNode = null;
	this.targetX = 0;
	this.targetY = 0;
	this.diagonalsEnabled = true;

14	this.worker = null; this.workerQueue = [];
15	pathfinder.prototype.findPath = function (startX, startY, endX, endY, successCallback, failCallback)

16	if (!this.cells)
	{
	failCallback();
	return;
	}
	
	startX = Math.floor(startX);
	startY = Math.floor(startY);
	endX = Math.floor(endX);
	endY = Math.floor(endY);
	
	this.targetX = endX;
	this.targetY = endY;
	this.pathEnd.parent = null;

17	pathfinder.prototype.setDiagonals = function (d)
	if (this.diagonalsEnabled)
	{
	var canMoveDirect = true

18	if (canMoveDirect)
	{
	for (i = 0, len = this.pathList.length; i < len; i++)
	
	freeResultNode(this.pathList[i]);

19	freeResultNode(this.pathList[i]);
	this.pathList.length = 0;
	
	this.pathEnd.x = endX;
	this.pathEnd.y = endY;

	this.pathEnd.parent = null;
	n = allocResultNode();
	n.x = endX;
	n.y = endY;
	this.pathList.push(n);
	successCallback();

20	add it in the right place
	c = allocNode();
	c.x = x_;
	c.y = y_;
	c.h = this.estimateH(x_, y_);
	c.g = this.currentNode.g + g_ + curCellCost;
	c.f = c.h + c.g;
	c.parent = this.currentNode;

21	pathfinder.prototype.estimateH = function (x_, y_)

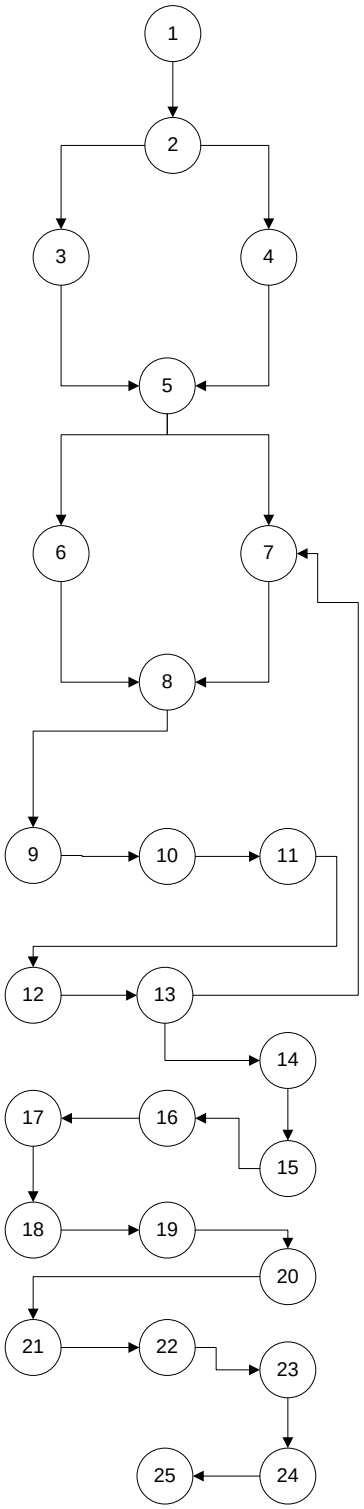
22	{
	var dx = quickAbs(x_ - this.targetX);
	var dy = quickAbs(y_ - this.targetY);

23	return dx * 10 + dy * 10; };

24	window.Pathfinder = pathfinder;
	
	window.ResultNode = resultNode;
	window.allocResultNode = allocResultNode;

25	window.freeResultNode = freeResultNode; }

a. Flowgraph Penelusuran algoritma A*



Gambar 4.13 Flow Graph Algoritma A*

b. Cyclomatic Complexity (VG)

V(G) = E – N + 2

V(G) = 25 – 23 +2

V(G) = 2+2 = 4

Keterangan : E = Jumlah aksi

N = Jumlah Kondisi

P = Jumlah bagian dari graph yang tidak terkoneksi

c. Independent Path

Path 1 = 1 – 2 – 3 – 5 – 6 – 8 – 10 – 11 – 12 – 13 – 14 – 15 -16 – 17 – 18 – 19 – 20 – 21 – 22 – 23 – 24 – 25

Path 2 = 1 – 2 – 4 – 5 – 7 – 8 – 10 – 11 – 12 – 13 – 14 – 15 -16 – 17 – 18 – 19 – 20 – 21 – 22 – 23 – 24 – 25

Path 3 = 1 – 2 – 4 – 5 – 7 – 8 – 10 – 11 – 12 – 13 – 7 – 8 – 10 – 11 – 12 – 13 – 14 – 15 -16 – 17 – 18 – 19 – 20 – 21 – 22 – 23 – 24 – 25

Path 4 = 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 – 9 – 12 – 13 7 – 8 – 10 – 11 – 12 – 13 – 14 – 15 -16 – 17 – 18 – 19 – 20 – 21 – 22 – 23 – 24 – 25

Graph Matriks penelusuran Algoritma A*

Tabel 4.3 Matriks penelusuran algoritma a*

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1		1																		
2			1														1			
3				1																
4					1															
5						1														
6							1													
7								1												
8									1											
9												1								
10									1		1									
11													1							
12													1							
13														1						
14				1											1					
15																1				
16																	1			
17																		1		
18		1																	1	
19	1																			1
20																				

Merujuk pada tabel 3.3 pada bab 3 untuk pengujian nomor path pada butir uji, maka dibuatlah tabel kesimpulan algoritma A* seperti yang ditunjukkan pada tabel 4.13 sebagai berikut:

Tabel 4.4 Kesimpulan Algoritma A*

No path	Butir Uji	Hasil yang diharapkan	Hasil sesuai kasus uji	Kesimpulan
1	Path.name = posisi target (6,3)	Membuat jalur pencarian	Membuat jalur pencarian	Sesuai dengan yang diharapkan
2	Path.name dengan kondisi penghalang ((4,2),(4,3),(4,4)	Musuh menemukan penghalang dan tidak dapat dilalui	Musuh menemukan penghalang dan tidak dapat dilalui	Sesuai dengan yang diharapkan
3	Path.name mendapatkan nilai terendah (3,3), (3,4) , (4,5), (5,3)	Musuh dapat menemukan posisi dengan nilai terendah	Musuh dapat menemukan posisi dengan nilai terendah	Sesuai dengan yang diharapkan

4.5.2 Pengujian Beta

Pengujian beta merupakan pengujian yang dilakukan secara objektif, dimana pengujian langsung melibatkan para pengguna aplikasi game secara langsung dengan menggunakan kuesioner mengenai kepuasan para pengguna terhadap aplikasi game yang dibangun. metode penilaian pengujian yang digunakan adalah metode kuantitatif berdasarkan data dari pengguna.

4.5.2.1 Kuesioner

Data hasil kuesioner yang telah diberikan dapat dicari presentasenya dari masing-masing jawaban dengan menggunakan rumus:

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

dimana,

P = Nilai presentase 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

Data yang didapat dari hasil kuesioner terhadap 30 orang responden adalah sebagai berikut:

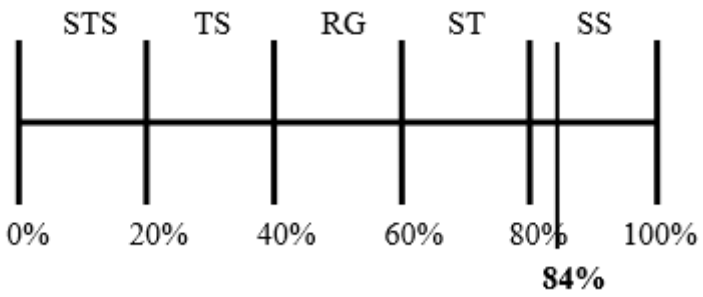
1. Apakah Game Edukasi Menjaga dan Memelihara Lingkungan Hidup dapat memberi pemahaman tentang membersihkan lingkungan?
   Hasil kuesioner untuk pertanyaan ke-1 dapat dilihat pada tabel 4.14.

Tabel 4.5 Hasil Kuisioner Pertanyaan 1

Pertanyaan	Jawaban	Skor	Responden	S
1	Sangat Setuju	5	8	40
	Setuju	4	20	80
	Biasa Saja	3	2	6
	Tidak Setuju	2	0	0
	Sangat Tidak Setuju	1	0	0
Jumlah			30	126

$$P = \frac{126}{150} x\ 100\ \% = \ 84\ \%$$

Dapat disimpulkan bahwa penilaian anak terhadap ketertarikan aplikasi Game Menjaga dan memelihara Lingkungan adalah 84% dari nilai 100% dan dapat dikategorikan sebagai sangat setuju



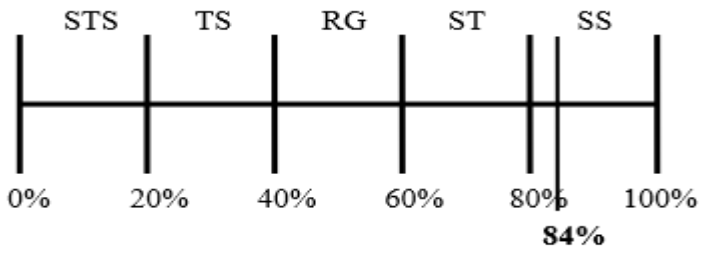
2. Apakah anda jadi tertarik menjaga lingkungan sekitar anda?

Tabel 4.6 Hasil Kuisioner Pertanyaan 2

Pertanyaan	Jawaban	Skor	Responden	S
2	Sangat Setuju	5	10	50
	Setuju	4	14	56
	Biasa Saja	3	3	9
	Tidak Setuju	2	3	6
	Sangat Tidak Setuju	1	0	0
Jumlah			30	115

$$P = \frac{126}{150} x\ 100\ \% = \ 84\ \%$$

Dari hasil kuisioner di atas maka dapat disimpulkan bahwa animasi dari game ini adalah 84% dan dapat dikategorikan sebagai sangat setuju



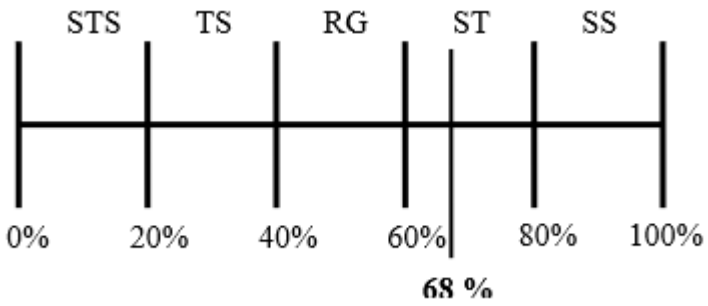
3. Apakah game ini membuat anda menyukai membersihkan lingkungan seperti karakter pada game ini?

Table 4.7 Hasil Kuisioner Pertanyaan 3

Pertanyaan	Jawaban	Skor	Responden	S
3	Sangat Suka	5	3	15
	Suka	4	8	32
	Cukup Suka	3	17	51
	Sedang	2	2	4
	Tidak Suka	1	0	0
Jumlah			30	80

$$\frac{102}{150} x\ 100\ \% =\ 68\ \%$$

Berdasarkan hasil penilaian diatas maka dapat disimpulkan, bahwa penilaian terhadap tingkat kesulitan dari permainan di dalam game ini adalah 68% dari yang diharapkan 100% atau bisa dikategorikan sebagai sulit



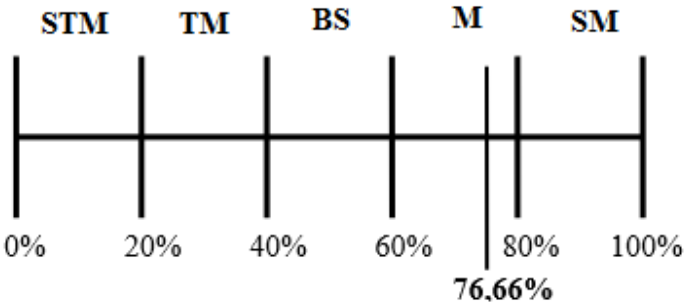
4. Apakah game ini memotivasi anda dalam menjaga dan memelihara lingkungan?

Tabel 4.8 Hasil Kuisioner Pertanyaan 4

Pertanyaan	**Jawaban**	**Skor**	**Responden**	**S**
4	Sangat Memotivasi	5	10	50
	Memotivasi	4	14	56
	Biasa Saja	3	3	9
	Tidak Memotivasi	2	3	6
	Sangat Tidak Menmotivasi	1	0	0
Jumlah			30	115

$$P = \frac{115}{150} x\ 100\ \% =\ 76{,}66\ \%$$

Berdasarkan hasil kuisioner di atas maka dapat disimpulkan bahwa setelah anak memainkan game kemauan anak untuk menjaga lingkungan sebesar 76,66% atau di kategorikan sebagai Memotivasi


STM
TM
BS
M
SM
0%
20%
40%
60%
80%
100%
76.66%