

**TUGAS AKHIR**

**PENYESUAIAN RINTANGAN GAME  
HORROR SIDE-SCROLLING DARI EKSPRESI  
WAJAH PEMAIN MENGGUNAKAN LIBRARY  
MOODME UNITY**



**Oleh:**

**Axel Matthew Adiwijaya**

**219310432**

**PROGRAM SARJANA  
PROGRAM STUDI INFORMATIKA  
FAKULTAS SAINS DAN TEKNOLOGI  
INSTITUT SAINS DAN TEKNOLOGI TERPADU SURABAYA  
SURABAYA  
2024**

## **TUGAS AKHIR**

# **PENYESUAIAN RINTANGAN GAME HORROR SIDE-SCROLLING DARI EKSPRESI WAJAH PEMAIN MENGGUNAKAN LIBRARY MOODME UNITY**

**Diajukan Guna Memenuhi Sebagian Persyaratan Untuk Memperoleh Gelar  
Sarjana Komputer  
Pada  
Institut Sains dan Teknologi Terpadu Surabaya**

**Disetujui oleh Tim Penguji Tugas Akhir:**

1. Herman Thuan To Saurik, S.Kom., M.T. (Pembimbing)
2. Dr. Ir. Gunawan, M.Kom. (Penguji I)
3. Dr. Ir. Hj. Endang Setyati, M.T. (Penguji II)
4. Dr. Ir. Joan Santoso, S.Kom., M.Kom. (Penguji III)

**SURABAYA  
MARET 2024**

## **SURAT PERNYATAAN KEASLIAN**

Yang bertanda tangan di bawah ini:

Nama : Axel Matthew Adiwijaya  
Fakultas/ Prodi : Sains dan Teknologi/ Informatika  
NRP : 219310432

dengan ini menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir dengan judul:

### **PENYESUAIAN RINTANGAN GAME HORROR SIDE-SCROLLING DARI EKSPRESI WAJAH PEMAIN MENGGUNAKAN LIBRARY MOODME UNITY**

adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diizinkan dan bukan merupakan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka.

Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, 12-03-2024  
Yang Membuat Pernyataan,

**Axel Matthew Adiwijaya**  
219310432

## **ABSTRAK**

Genre game horror mengusung pendekatan yang menarik, dengan elemen rintangan yang terintegrasi untuk menghindari kebosanan dan mengatur tingkat kesulitan pada setiap level. Dengan memanfaatkan respons emosional pemain serta fitur Dynamic Difficulty Adjustment, konten permainan tidak hanya menghindari kebosanan, namun juga mampu menyesuaikan tantangan sesuai dengan kinerja dan preferensi pemain

Tugas akhir ini bertujuan untuk meningkatkan kualitas bermain pemain game horror dengan memanfaatkan penyesuaian rintangan pada ada pada tiap level. Penyesuaian dilakukan dengan menganalisis data emosi wajah pemain selama sesi bermain. Metode penelitian menggunakan waterfall, dimana semua proses dilakukan secara sekuensial mulai dari pembuatan dasar permainan hingga analisis data menggunakan library MoodME, dilanjutkan dengan pengujian White Box dan Black Box. Game dievaluasi menggunakan kuesioner yang diberikan kepada 30 pemain yang telah memainkan game. Kuesioner dianalisis menggunakan UEQ yang berfokus untuk mengukur user experience sejauh mana game dapat menyesuaikan rintangan terhadap performa pemainnya serta daya tarik game yang dirancang.

Hasil evaluasi penggunaan konsep pengaturan rintangan yang dapat menyesuaikan tingkat kesulitan berdasarkan ekspresi wajah pemain telah berhasil menarik minat pemain secara berulang (mean daya tarik = 1,59). Meskipun demikian, terdapat kebutuhan untuk memberikan penjelasan lebih lanjut tentang tujuan dan fungsi pengaturan rintangan berdasarkan emosi pemain (mean kejelasan = 1,14), serta memastikan ketepatan (mean ketepatan = 1,32) dan efisiensi (mean efisiensi = 1,47) dalam menyesuaikan tingkat kesulitan dengan performa pemain. Penggunaan rintangan yang dapat menyesuaikan pemain berhasil menstimulasi (mean stimulasi = 1,71) pemain untuk melanjutkan permainan hingga selesai. Fitur penyesuaian rintangan berdasarkan skor dan emosi pemain dinilai inovatif dan terbarukan (mean kebaruan = 1,45).

## **ABSTRACT**

The horror game genre adopts an engaging approach, integrating obstacles to avoid boredom and adjusting difficulty levels at each level. By leveraging players' emotional responses and Dynamic Difficulty Adjustment features, the game content prevents monotony and adapts challenges according to players' performance and preferences.

This final project aims to enhance players' gaming experience in horror games by utilising obstacle adjustments at each level, achieved through analysing players' facial emotional data. The research method employed the waterfall model, where all processes were sequentially executed, from basic game development to data analysis using the MoodME library, followed by White Box and Black Box testing. The game was evaluated using a questionnaire distributed to 30 players who had played the game. The questionnaire analysis was conducted using UEQ, focusing on measuring the user experience regarding the game's ability to adapt obstacles to players' performance and the designed game appeal.

The evaluation results of employing the concept of obstacle adjustment based on players' facial expressions successfully elicited repeated player engagement (mean attractiveness = 1.59). However, there is a need for further explanation regarding the purpose and function of obstacle adjustment based on player emotions (mean clarity = 1.14), as well as ensuring accuracy (mean accuracy = 1.32) and efficiency (mean efficiency = 1.47) in adjusting difficulty levels according to player performance. Using obstacles adaptable to players effectively stimulated (mean stimulation = 1.71) players to continue playing until completion. The feature of obstacle adjustment based on player scores and emotions was deemed innovative and sustainable (mean novelty = 1.45).

## KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa atas rahmat dan kebesaran-Nya, penulis dapat menyelesaikan laporan tugas akhir yang berjudul “PENYESUAIAN RINTANGAN GAME HORROR SIDE-SCROLLING DARI EKSPRESI WAJAH PEMAIN MENGGUNAKAN *LIBRARY MOODME UNITY*”. Dalam pembuatan tugas akhir ini ini tentu tidak lepas dari bantuan berbagai pihak. Oleh karena itu perkenankan penulis menyampaikan ucapan terima kasih kepada :

1. Ir. Arya Tandy Hermawan, M.T., selaku Rektor Institut Sains dan Teknologi Terpadu Surabaya.
2. Ir. Edwin Pramana, M.AppSc., Ph.D., selaku Dekan Fakultas Sains dan Teknologi Institut Sains dan Teknologi Terpadu Surabaya.
3. Dr. Ir. Esther Irawati Setiawan, S.Kom., M.Kom., selaku Kaprodi Program Profesional Informatika Institut Sains dan Teknologi Terpadu Surabaya.
4. Herman Thuan To Saurik, S.Kom., M.T., selaku dosen pembimbing yang senantiasa memberikan bimbingan dan dukungan moral yang tidak terhingga kepada penulis sehingga tugas akhir ini selesai dengan baik.
5. Seluruh Dosen Pengajar dan Staff Institut Sains dan Teknologi Terpadu Surabaya yang selama ini telah memberikan ilmu pengetahuan kepada penulis.
6. Keluarga tercinta, papa, mama dan saudara saya atas doa, nasehat dan dukungan yang tak terhingga baik dari segi moral maupun materiil yang diberikan sampai sekarang.
7. Kekasih saya Felicia Young, yang senantiasa memberikan dukungan selama kegiatan perkuliahan hingga penulisan Tugas Akhir.
8. Sahabat saya Robert Araya, Novanda Kensela, Vincent Soegiarto, Yeremia Pranata, Yoengky Kurnia. yang tak henti memberi semangat, motivasi dan dukungan selama kegiatan perkuliahan hingga penulisan Tugas Akhir.
9. Semua pihak yang turut berpartisipasi dan membantu dalam penyusunan laporan tugas akhir ini, yang tidak dapat disebutkan satu per satu.

Semoga bantuan yang telah Bapak, Ibu, dan Saudara berikan mendapat balasan dari Tuhan yang Maha Esa dan menjadi amal kebaikan dalam pengembangan dunia pendidikan.

Penulis menyadari laporan ini dibuat dengan berbagai upaya untuk menghasilkan yang terbaik, namun tugas akhir ini masih terdapat kekurangan dan belum sempurna. Untuk segala kesalahan dan kekurangan dalam penulisan laporan ini, penulis mengharapkan adanya kritik dan saran yang membangun agar tugas akhir ini menjadi lebih baik lagi. Akhir kata, semoga laporan tugas akhir ini dapat bermanfaat bagi pembaca.

Surabaya, Maret 2024  
Penulis

## DAFTAR ISI

	<b>Halaman</b>
HALAMAN JUDUL.....	i
HALAMAN PENGESAHAN.....	ii
SURAT PERNYATAAN KEASLIAN.....	iii
ABSTRAK .....	iv
ABSTRACT.....	v
KATA PENGANTAR .....	vi
DAFTAR ISI.....	vii
DAFTAR GAMBAR .....	xiii
DAFTAR TABEL.....	xiii
DAFTAR ALGORITMA.....	xv
DAFTAR SEGMENT PROGRAM .....	xvi
DAFTAR RUMUS .....	xviii
BAB I PENDAHULUAN.....	1
1.1 Latar Belakang.....	1
1.2 Rumusan Masalah.....	2
1.3 Batasan Masalah .....	3
1.4 Fitur.....	3
1.5 Tujuan .....	4
1.6 Metodologi .....	4
1.7 Sistematika Penulisan .....	6
BAB II TEORI PENUNJANG .....	9
2.1 Dynamic Difficulty Adjustment (DDA) .....	9
2.2 Game Horor .....	10
2.3 Game Side Scrolling .....	12
2.4 Dynamic Scripting .....	13
2.5 Mesin Permainan Unity .....	13
2.5.1 Unity 2D .....	13

2.5.2 Barracuda SDK .....	15
2.5.3 MoodMe 4 Emotion Barracuda SDK .....	16
2.5.4 Interaksi MoodMe dengan Barracuda .....	18
2.6 Metode Testing .....	21
2.6.1 Whitebox Testing .....	21
2.6.2 Blackbox Testing .....	22
<b>BAB III PERANCANGAN DAN ANALISA SISTEM .....</b>	<b>24</b>
3.1 Perencanaan .....	24
3.2 Analisa Game Horror 2D Skenario Alas Tilas, Jawa Timur	24
3.3 Elemen Dalam Game .....	25
3.4 Analisa Kebutuhan .....	27
3.4.1 Analisa Kebutuhan Hardware .....	27
3.4.2 Analisa Kebutuhan Software .....	27
3.5 Perancangan Game .....	28
3.5.1 Konsep .....	28
3.5.2 Plot Cerita .....	28
3.5.3 Use Case Diagram .....	29
3.5.4 State Diagram.....	31
3.6 Cara Bermain.....	33
3.7 Klasifikasi Emosi.....	34
3.8 Pendektsian Emosi MoodMe.....	36
3.9 Pengaturan Parameter DDA Rintangan	
Berdasarkan Pendektsian Emosi MoodMe.....	38
3.9.1 Parameter DDA Pada Rintangan Hantu .....	39
3.9.2 Parameter DDA Pada Rintangan Duri.....	40
3.9.3 Parameter DDA Pada Tempat Bersembunyi .....	41
3.9.4 Parameter DDA Pada Jumpscare Penampakan.....	42
<b>BAB IV DESAIN SISTEM .....</b>	<b>44</b>
4.1 Desain Arsitektural .....	44
4.1.1 Skenario Game.....	45
4.1.2 Arsitektur Pengaturan Rintangan Menggunakan DDA	46

4.2	Desain Interface .....	57
4.2.1	Menu Utama.....	58
4.2.2	Narasi Penutup .....	58
4.2.3	Cara Bermain.....	59
4.2.4	Interface HUD Karakter .....	60
4.2.5	Interface Pendeksiian Wajah .....	61
4.2.6	Interface Report Performa Pemain .....	61
4.3	Desain Procedural .....	62
4.3.1	Algoritma Kontrol Karakter .....	63
4.3.2	Algoritma Kontrol Merangkak .....	64
4.3.3	Algoritma Bangkit Dari Mode Merangkak.....	64
4.3.4	Algoritma Kontrol Melompat.....	64
4.3.5	Algoritma Karakter Mengambil Item.....	65
4.3.6	Algoritma Karakter Bersembunyi .....	65
4.3.7	Algoritma Karakter Dikejar Hantu.....	66
4.3.8	Algoritma Karakter Terkena Serangan Hantu.....	66
4.3.9	Algoritma Rintangan Hantu .....	67
4.3.10	Algoritma Jumpscare Penampakan.....	68
4.3.11	Algoritma Item.....	69
4.3.12	Algoritma Tempat Bersembunyi .....	69
4.3.13	Algoritma Rintangan Duri .....	70
4.3.14	Algoritma Pencatatan Log DDA .....	71
4.3.15	Algoritma Penyesuaian Rintangan DDA .....	71
4.3.16	Algoritma Pembentukan Rintangan Level.....	72
BAB V	IMPLEMENTASI PROGRAM .....	74
5.1	Sistem Pencatat dan Pembaca Log .....	74
5.1.1	Sistem Pencatatan Log .....	74
5.1.2	Sistem Pembacaan Log .....	77
5.2	Sistem Perubah Tingkat Kesulitan Rintangan .....	80
5.2.1	Sistem Penentuan Kategori Pemain .....	80
5.2.2	Sistem Perhitungan Skor Emosi .....	82

5.2.3 Sistem Penyesuai Tingkat Kesulitan Rintangan .....	84
5.3 Karakter .....	84
5.3.1 Kontrol Gerak Karakter .....	84
5.3.2 Karakter Merangkak.....	86
5.3.3 Karakter Melompat .....	87
5.3.4 Karakter Memukul .....	88
5.3.5 Karakter Kena Serang .....	91
5.4 Rintangan .....	93
5.4.1 Rintangan Hantu .....	94
5.4.2 Rintangan Duri.....	98
5.4.3 Jumpscare Penampakan .....	99
5.4.4 Tempat Bersembunyi.....	101
BAB VI UJI COBA .....	103
6.1 White Box Testing .....	103
6.1.1 White Box Testing Karakter.....	103
6.1.2 White Box Testing Rintangan Hantu .....	104
6.1.3 White Box Testing Rintangan Duri.....	105
6.1.4 White Box Testing Tempat Bersembunyi .....	106
6.1.5 White Box Testing Jumpscare Penampakan .....	107
6.1.6 White Box Testing Pencatat Log.....	107
6.1.7 White Box Testing Pembaca Log.....	108
6.1.8 White Box Testing Penentuan Kategori Pemain.....	109
6.2 Black Box Testing .....	111
6.2.1 Black Box Testing Karakter.....	111
6.2.2 Black Box Testing Rintangan Hantu.....	112
6.2.3 Black Box Testing Rintangan Duri .....	113
6.2.4 Black Box Testing Rintangan Kabut.....	113
6.2.5 Black Box Testing Item.....	114
6.2.6 Black Box Testing Jumpscare Penampakkan.....	115
6.2.7 Black Box Testing Tempat Bersembunyi .....	115
6.2.8 Black Box Testing Pencatat Log.....	116

6.2.9	Black Box Testing Pembaca Log .....	117
6.2.10	Black Box Testing Penentuan Kategori Pemain.....	118
6.3	Kuesioner .....	119
BAB VII	PENUTUP .....	126
7.1	Kesimpulan .....	126
7.2	Saran .....	127
DAFTAR PUSTAKA .....		129
RIWAYAT HIDUP .....		131

## DAFTAR GAMBAR

<b>Gambar</b>	<b>Halaman</b>
1.1 Alur dari Iterative Waterfall.....	5
2.1 Tampilan Antarmuka Unity 2D .....	14
2.2 Pemanfaatan Library Barracuda.....	16
2.3 Penggunaan MoodMe Unity .....	17
2.4 Penggunaan Neural Network yang Berasal Dari Barracuda Didalam Script MoodMe .....	20
2.5 Alur dari White Box Testing.....	21
2.6 Alur dari Black Box Testing .....	22
3.1 Diagram Use Case Game .....	29
3.2 Diagram State Game.....	31
3.2 Ilustrasi Penangkapan Wajah Oleh MoodMe .....	29
3.3 Karakteristik Wajah Marah.....	35
3.4 Karakteristik Wajah Takut.....	35
3.5 Ilustrasi Wajah Netral .....	36
3.6 Ilustrasi Penangkapan Wajah Oleh MoodMe .....	36
3.7 Alur Kerja Pendekripsi Emosi MoodMe .....	37
4.1 Desain Arsitektural Pengolahan Rintangan Oleh DDA .....	44
4.2 Simulasi Pembentukan Rintangan Level .....	57
4.3 Tampilan Menu Utama .....	58
4.4 Tampilan Narasi Penutup.....	59
4.5 Tampilan Cara Bermain.....	59
4.6 Tampilan Hud Karakter .....	60
4.7 Tampilan Pendekripsi Wajah .....	61
4.8 Tampilan Report Performa Pemain .....	62
6.1 26 Skala Pertanyaan UEQ.....	120
6.2 Grafik Hasil Benchmark UEQ .....	124

## DAFTAR TABEL

<b>Tabel</b>	<b>Halaman</b>
3.1 Parameter DDA Pada Rintangan Hantu .....	39
3.2 Tabel Keputusan DDA Rintangan Spawn Hantu .....	40
3.3 Parameter DDA Pada Rintangan Duri.....	40
3.4 Tabel Keputusan DDA Rintangan Duri.....	41
3.5 Parameter DDA Pada Tempat Bersembunyi .....	41
3.6 Tabel Keputusan DDA Tempat Bersembunyi .....	42
3.7 Parameter DDA Pada Jumpscare Penampakan.....	42
3.8 Tabel Keputusan DDA Jumpscare Penampakan.....	43
4.1 Tabel Simulasi Perhitungan Performa Mahir .....	48
4.2 Tabel Simulasi Perhitungan Performa Normal 1 .....	49
4.3 Tabel Simulasi Perhitungan Performa Normal 2 .....	51
4.4 Tabel Simulasi Perhitungan Performa Pemula 1 .....	52
4.5 Tabel Label Nilai Emosi .....	53
4.6 Tabel Pengaturan Weight Clipping DDA Rintangan Spawn Hantu ..	55
4.7 Tabel Pengaturan Weight Clipping DDA Rintangan Duri .....	55
4.8 Tabel Pengaturan Weight Clipping DDA Tempat Bersembunyi.....	56
4.9 Tabel Pengaturan Weight Clipping DDA Rintangan Jumpscare Penampakan .....	56
6.1 Tabel White Box Testing Karakter .....	103
6.2 Tabel White Box Testing Rintangan Hantu.....	105
6.3 Tabel White Box Testing Rintangan Duri .....	105
6.4 Tabel White Box Testing Tempat Bersembunyi.....	106
6.5 Tabel White Box Testing Jumpscare Penampakan .....	107
6.6 Tabel White Box Testing Pencatat Log.....	107
6.7 Tabel White Box Testing Pembaca Log.....	108
6.8 Tabel White Box Testing Penentuan Kategori Pemain .....	110
6.9 Tabel Black Box Testing Karakter.....	111

6.10	Tabel Black Box Testing Hantu .....	112
6.11	Tabel Black Box Testing Rintangan Duri.....	113
6.12	Tabel Black Box Testing Kabut .....	113
6.13	Tabel Black Box Testing Spawn Item.....	114
6.14	Tabel Black Box Testing Jumpscare Penampakan .....	115
6.15	Tabel Black Box Testing Tempat Bersembunyi .....	115
6.16	Tabel Black Box Testing Pencatat Log .....	116
6.17	Tabel Black Box Testing Pembaca Log .....	117
6.18	Tabel Black Box Testing Penentuan Kategori Pemain .....	118
6.19	Item Pertanyaan.....	121
6.20	Hasil Mean, Variance dan Standar Deviasi .....	122
6.21	Hasil Mean dan Variance 6 Skala UEQ .....	123
6.22	Hasil Benchmark UEQ .....	124

## DAFTAR ALGORITMA

<b>Algoritma</b>	<b>Halaman</b>
4.1 Kontrol Karakter .....	63
4.2 Kontrol Merangkak .....	64
4.3 Kontrol Bangkit Dari Mode Merangkak.....	64
4.4 Kontrol Melompat .....	64
4.5 Karakter Mengambil Item.....	65
4.6 Karakter Bersembunyi.....	66
4.7 Karakter Dikejar Hantu.....	66
4.8 Karakter Terkena Serangan Hantu .....	67
4.9 Rintangan Hantu.....	68
4.10 Rintangan Jumpscare Penampakan .....	68
4.11 Item.....	69
4.12 Tempat Bersembunyi.....	69
4.13 Rintangan Duri .....	70
4.14 Pencatatan Log DDA.....	71
4.15 Penyesuaian Rintangan DDA.....	71
4.16 Pembentukan Rintangan Level .....	72

## DAFTAR SEGMENT PROGRAM

<b>Segment Program</b>	<b>Halaman</b>
5.1 Pencatatan Emosi .....	74
5.2 Pencatatan Log .....	75
5.3 Pengecekan File Log .....	77
5.4 Sistem Pembaca Log .....	77
5.5 Pembacaan Log .....	78
5.6 Penentuan Kategori Pemain .....	80
5.7 Perhitungan Skor Emosi .....	82
5.8 Sistem Penyesuai Tingkat Kesulitan .....	84
5.9 Kontrol Gerak Karakter .....	85
5.10 Mode Merangkak .....	86
5.11 Kontrol Merangkak .....	87
5.12 Mode Lompat .....	87
5.13 Kontrol Lompat .....	88
5.14 Mode Pukul .....	89
5.15 Serangan Jenis Pukul .....	89
5.16 Kontrol Pukul .....	90
5.17 Kena Serangan .....	91
5.18 Karakter Mati .....	92
5.19 Karakter Langsung Mati .....	93
5.20 Generate Pijakan Hantu .....	94
5.21 Cek Penginjakan .....	95
5.22 Memunculkan Hantu .....	96
5.23 Generate Rintangan Duri .....	98
5.24 Pemain Terkena Duri .....	99
5.25 Generate Rintangan Jumpscare Penampakan .....	99
5.26 Pemain Menginjak Jumpscare .....	100
5.27 Generate Platform Bambu .....	101

5.28 Pemain Bersembunyi .....	102
-------------------------------	-----

## **DAFTAR RUMUS**

<b>Rumus</b>	<b>Halaman</b>
4.1 Perhitungan DDA Pemain Mahir 1 .....	47
4.2 Perhitungan DDA Pemain Mahir 2 .....	47
4.3 Perhitungan DDA Pemain Normal 1 .....	48
4.4 Perhitungan DDA Pemain Normal 2 .....	50
4.5 Perhitungan DDA Pemain Normal 3 .....	50
4.6 Perhitungan DDA Pemain Pemula 1 .....	51
4.7 Perhitungan DDA Pemain Pemula 2 .....	52
4.8 Perhitungan Skor Emosi Tidak Diinjak .....	53
4.9 Perhitungan Skor Emosi Jalan Dihindari Semua .....	54
4.10 Perhitungan Rata-Rata Skor Emosi .....	54

# BAB I

## PENDAHULUAN

Pada bab pendahuluan ini akan menjelaskan mengenai hal-hal dasar dari pembuatan Tugas Akhir. Hal-hal dasar tersebut meliputi latar belakang, rumusan masalah, batasan masalah dan tujuan dari pembuatan Tugas Akhir ini.

### 1.1 Latar Belakang

Di Indonesia yang memiliki banyak game tradisional saat ini tak luput dari digitalisasi permainan. Permainan atau yang kerap disebut game memiliki banyak genre salah satunya yaitu horror. Game horror sendiri berasal dari horror yang berkaitan dengan terror, kengerian, dan perasaan ngeri<sup>1</sup>. Perkembangan game horror sendiri di Indonesia dapat dilihat dari banyaknya pengembang yang berhasil menciptakan game bergenre horror terkenal seperti ‘DreadOut’ yang konsepnya unik dan ‘Pamali’ yang kental dengan dengan nuansa pantangan di masyarakat Indonesia<sup>2</sup>. Namun kebanyakan dari game horror yang beredar berperspektifkan First Person Perspective yaitu, dari sudut pandang orang pertama. Sedangkan game yang akan dikembangkan memilih sudut pandang Third Person Perspective yaitu sudut pandang orang ketiga dengan angle pemain side scrolling.

Side scrolling sendiri berarti game yang aksinya dilihat dari sudut kamera yang menyamping dengan layar yang mengikuti karakter pemain yang bergerak ke kiri atau kanan<sup>3</sup>. Game horror yang kebanyakan berperspektif kamera first person guna mengejar nuansa horror bukannya tidak mungkin diubah perspektifnya menjadi third person side scrolling.

Agar nuansa horror dari game side scroll yang akan dikembangkan bersaing

---

<sup>1</sup> Tobias Arnell, Nikola Stojanovic, *Horror game design – what instills fear in the player* 2020, hlm 1.

<sup>2</sup> Aliko Salsabila Marwanto, Wegig Murwonugroho, *Psikologis Pada Gamers Ketika Memainkan Survival Horror Game “DREADOUT”*, 2021, hlm 2.

<sup>3</sup> Nopi Ramsari, Gilang Ramadhan, *Pembuatan Game Side Scrolling 2D The Naila’s Survival Berbasis Android*, 2020, hlm 2.

dengan game horror yang berperspektif first person, digunakanlah Dynamic Difficulty Adjustment agar dapat menyesuaikan tingkat kesulitan tiap level yang ada.

Dynamic Difficulty Adjustment digunakan untuk menyeimbangkan tingkat kesulitan game secara dinamis dengan cara menyesuaikan tingkat kesulitan dan jenis rintangan yang telah dibuat<sup>4</sup>. Tujuan dipakainya Dynamic Difficulty Adjustment pada game ini agar pemain baru tidak mengalami kesulitan menjalankan permainan dan membuat game tidak monoton karena dapat menyesuaikan level dengan performa pemainnya.

## 1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, dapat disimpulkan permasalahan yang dihadapi yaitu bagaimana cara agar pemain bisa mendapatkan pengalaman bermain yang baik sehingga game horror dapat menarik minat pemain baru untuk memainkan game dengan genre horror sekaligus menarik minat pemain yang sering bermain game dengan genre horror.

## 1.3 Batasan Masalah

Dalam pembuatan Tugas Akhir “Penyesuaian Rintangan Game Horror Side-Scrolling Dari Ekspresi Wajah Pemain Menggunakan Library MoodME Unity” ini mempunyai batasan-batasan sebagai berikut :

1. Dynamic Difficulty Adjustment disini disinggung hanya sebagai pengolah dari output yang dihasilkan dari deteksi ekspresi wajah pemain setelah 1 level diselesaikan.
2. Facial expression mendeteksi 3 ekspresi yaitu Scared (Takut), Angry (Marah) dan biasa saja.
3. Platform level yang ada telah di desain tidak bisa di ubah, tetapi rintangannya saja yang bisa di ubah.

---

<sup>4</sup> Mohammad Zohaib, *Review Article Dynamic Difficulty Adjustment (DDA) in Computer Games: A Review*, 2018, hlm 1.

4. Rintangan sudah dibuat dan diberi kategori, jadi DDA hanya digunakan untuk menentukan jumlah rintangan kategori tertentu yang akan muncul lebih banyak pada level berikutnya.
5. Agar data ekspresi pemain valid, hanya diperbolehkan bermain sendiri tanpa wajah orang lain di dalam kamera dan berada pada ruang dengan pencahayaan cukup.

## 1.4 Fitur

Subbab ini menjelaskan fitur apa saja yang terdapat pada game yang dikembangkan. Berikut merupakan daftar fitur-fitur yang terdapat di dalam game yang dikembangkan :

1. Penyesuaian Tingkat Kesulitan Rintangan Level Berikutnya Secara Otomatis.  
Rintangan diatur secara otomatis dengan penggunaan Dynamic Scripting sebagai DDA pada game yang dipilih. Jadi tahap kerja dynamic scripting pada game yaitu
  1. Level berikutnya di bentuk sebelum level itu dimulai.
  2. Di dalam level, terdapat rintangan dengan parameter index array rintangan, jenis rintangan, dan checkpoint start dan finish dari rintangannya.
  3. Pengaturan jenis rintangan oleh DDA sesuai dengan pembacaan log emosi yang terdapat jenis rintangan yang berhasil dilalui. Kemudian DDA mengisi array rintangan pada level dengan jenis rintangan yang cocok dengan performa pemain.
2. Deteksi Emosi Wajah Secara Realtime.

Dengan penggunaan library MoodME dan Barracuda yang dapat berkerja secara realtime, emosi pemain sepanjang jalannya game dapat ditangkap secara akurat waktu dan dimana emosi tersebut terjadi disepanjang jalannya level game. Contohnya seperti saat pemain melewati rintangan jurang dengan dikejar oleh hantu lalu gagal, apakah ekspresi pemain tersebut takut, marah atau biasa saja, hal ini dapat diketahui secara akurat jenis rintangan dan checkpoint berserta ekspresi apa yang keluar sehingga data yang didapat akurasinya tinggi dan dapat dipertanggung jawabkan.

### 3. Deteksi Multiemosi.

Emosi yang dapat ditangkap ada 3 yaitu takut, marah dan biasa saja. Kemudian data yang ditangkap tersebut ditunjukkan seberapa kevalidanya dengan persentase 0-1 dimana 0 sedang tidak terjadi hingga angka 1 valid. Angka yang di tunjukan berjarak 0 hingga 1. Contohnya saat melewati rintangan pemain menunjukkan emosi biasa saja 0.6 dan takut 0.4, maka data yang diambil hanya emosi biasa saja di rintangan tersebut karena perbandingan emosi biasa saja lebih besar.

## 1.5 Tujuan

Dengan latar belakang yang dipilih oleh penulis, maka tujuan pembuatan tugas akhir ini yaitu :

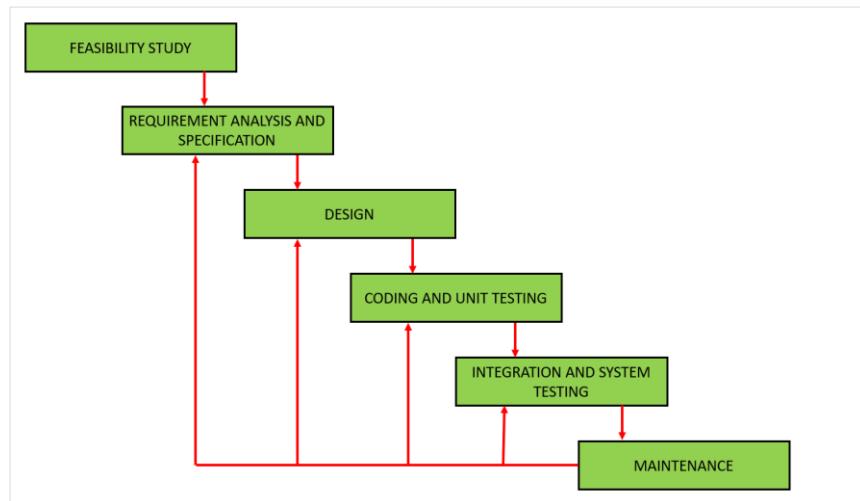
1. Berhasil memanfaatkan ekspresi wajah pemain untuk mengatur tingkat kesulitan rintangan game.
2. Menghasilkan produk game horror dengan rintangan yang dinamis.

## 1.6 Metodologi

Berikut adalah langkah-langkah yang dilakukan untuk pembuatan Tugas Akhir ini :

1. Mencari informasi, dokumentasi, dan tutorial tentang implementasi face recognition sebagai pembaca emosi pemain dan Unity Engine dari berbagai sumber.
2. Perancangan skenario penggunaan emotion detector untuk mendapatkan output agar game secara dinamis dapat menyesuaikan tingkat kesulitan levelnya.
3. Melakukan konsultasi dengan dosen pembimbing.
4. Melakukan testing game yang telah diintegrasikan dengan fitur emotion detector.
5. Melakukan perbaikan pada kesalahan yang ada.
6. Melakukan uji coba dengan membagi kuesioner.
7. Membuat buku laporan Tugas Akhir.

Berdasarkan susunan langkah-langkah yang dibuat di atas, maka dari itu dilakukan pemilihan metodologi untuk membantu proses pembuatan game. Metodologi yang digunakan pada pembuatan program ini adalah Iterative Waterfall Model. Iteratif Waterfall merupakan siklus pengembangan perangkat lunak di mana pengembangan awal dimulai berdasarkan spesifikasi dasar kemudian ditambahkan lebih banyak fitur dengan iterasi yang berkelanjutan hingga sistem akhir dibuat. Model Metode ini dipilih karena pengaplikasian metode ini cocok diaplikasikan untuk pembuatan game.



**Gambar 1.1**  
**Alur dari Iterative Water Fall**

Iterative Waterfall Model memiliki 6 tahapan pada alurnya, dapat di lihat pada gambar 1.1. Alur tersebut dimulai dari langkah feasibility study dan berakhir pada langkah maintenance. Berikut adalah langkah- langkah yang dilakukan berdasarkan alur dari Iterative Waterfall Model untuk pembuatan program ini :

1. Feasibility Study adalah tahapan melihat kelayakan program yang akan dibuat. Tahapan ini dilakukan dengan menanyakan apakah pengambangan dari program ini diperlukan atau tidak.
2. Requirement Analysis And Specification adalah tahapan yang dilakukan untuk menanyakan kebutuhan apa saja yang dibutuhkan agar program yang menjadi lebih efisien.

3. Design adalah tahapan dimana dilakukan perancangan database serta seperti apa tampilan dari program tersebut. Pada tahapan ini dilakukan pembuatan usecase agar memudahkan dalam perancangan.
4. Coding and Unit Testing adalah tahapan dimana dilakukan pembuatan program berdasarkan requirement yang diminta. Setelah program selesai maka akan dilakukan testing apakah program sudah berjalan sesuai dengan apa yang diharapkan. Pada tahapan ini dilakukan pembuatan game dan dilakukan testing di setiap requirement yang telah dibuat.
5. Integration and System Testing tahapan ini akan dilakukan penggabungan modul-modul yang sudah dibuat sebelumnya ke dalam program lalu dilakukan pengujian apakah modul-modul yang ditambahkan berfungsi dengan baik pada system.
6. Maintenance adalah tahapan terakhir dari Iterative Waterfall Model. Di sini game yang sudah jadi akan dijalankan atau dioperasikan oleh penggunanya. Disamping itu dilakukan pula pemeliharaan pada game yang telah dibuat.

## 1.7 Sistematika Penulisan

Tugas Akhir harus memiliki sistematika penulisan yang baik dan sesuai. Pedoman ini digunakan agar pembahasan runtut. Secara keseluruhan buku ini akan mencakup beberapa hal yang tertuang dalam rincian bab dibawah ini.

- BAB I : PENDAHULUAN

Bab ini menjelaskan tentang latar belakang, rumusan masalah, batasan masalah, tujuan penulisan, metodologi, dan sistematika penulisan buku Tugas Akhir “Penyesuaian Rintangan Game Horror Side-Scrolling Dari Ekspresi Wajah Pemain Menggunakan Library MoodME Unity”.

- BAB II : TEORI PENUNJANG

Bab ini menjelaskan mengenai teori-teori yang mendasari penulisan Tugas Akhir ini. Terdapat beberapa teori yang turut menunjang pelaksanaan Tugas Akhir ini, yaitu teori tentang DDA, Teori Pendukung tentang Game, *Dynamic Scripting*,

Mesin Permainan Unity 2D, Barracuda SDK, MoodME 4 Emotion Barracuda SDK, dan Metode Testing.

- BAB III : PERANCANGAN DAN ANALISA SISTEM

Bab ini menjelaskan tentang perancanaan rancangan, analisa game, uraian elemen-elemen dalam game, analisa kebutuhan, perancangan game, dan pengaturan parameter DDA lintasan yang ada didalam game. Analisa game membahas tema dan konsep yang akan diterapkan pada game. Analisa kebutuhan membahas kebutuhan agar game dapat beroperasional. Pengaturan parameter DDA akan menguraikan parameter yang digunakan DDA untuk mengatur lintasan disepanjang level game berjalan.

- BAB IV : DESAIN GAME

Bab ini menguraikan desain arsitektural, desain interface, desain procedural yang digunakan didalam game. Desain arsitektural akan membahas desain sistem DDA saat pengaturan lintangan dilakukan. Desain interface meliputi desain karakter, musuh, lintangan dan desain halaman-halaman didalam game. Desain procedural akan membahas algoritma penting yang ada pada game.

- BAB V : IMPLEMENTASI PROGRAM

Bab ini berisikan segmen-segmen program pada Sistem Pembaca dan Penulis Log, Sistem Perubahan Lintangan, Karakter, Lintangan. Setiap segmen dijelaskan bagaimana logika yang berjalan pada setiap baris segmen program yang ditulis.

- BAB VI : UJI COBA

Bab ini berisikan penjelasan tentang uji coba serta skenario yang dilakukan selama penggerjaan Tugas Akhir. Pada tahap ini dilakukan uji coba White Box Testing, Black Box Testing dan

Kuesioner yang diberikan pada responden yang telah memainkan game.

- BAB VII : PENUTUP

Bab ini berisikan kesimpulan dari penulisan Tugas Akhir yang dilakukan. Selain itu pada bab ini juga dicantumkan saran-saran yang didapatkan dalam penulisan Tugas Akhir.

## **BAB II**

### **TEORI PENUNJANG**

Pada bagian ini akan dijelaskan mengenai teori-teori yang menunjang dalam pembuatan game ini. Dimana selama pembuatan game ini berbagai macam teori diperlukan agar harapannya pembuatan game ini memenuhi standar dan ekspektasi pembuat. Teori-teori penunjang ini akan dijelaskan dalam bentuk poin-poin, sebagai berikut.

#### **2.1 Dynamic Difficulty Adjustment (DDA)**

Dynamic Difficulty Adjustment (DDA) adalah sebuah sistem dalam game yang secara otomatis menyesuaikan tingkat kesulitan permainan berdasarkan kinerja atau kemampuan pemain secara real-time. Tujuan dari DDA adalah untuk memberikan pengalaman bermain yang lebih menyenangkan, menantang, dan memuaskan bagi pemain, tanpa membuatnya terlalu mudah atau terlalu sulit<sup>5</sup>.

Pengaplikasian DDA dalam game dapat dilakukan dengan beberapa cara:

1. Penyesuaian Kesulitan: DDA dapat menyesuaikan tingkat kesulitan permainan berdasarkan performa pemain. Jika pemain mengalami kesulitan dalam menyelesaikan level atau tantangan tertentu, DDA dapat secara otomatis menurunkan tingkat kesulitan agar pemain dapat lebih mudah mencapai tujuan. Sebaliknya, jika pemain menunjukkan kinerja yang baik dan berhasil menyelesaikan tantangan dengan mudah, DDA dapat meningkatkan tingkat kesulitan untuk memberikan tantangan yang lebih besar.
2. Kontrol AI dan Musuh: DDA dapat digunakan untuk mengatur kecerdasan buatan (AI) musuh dalam game. Jika pemain menghadapi musuh yang terlalu kuat, DDA dapat menurunkan kecerdasan atau keterampilan musuh agar lebih mudah dihadapi. Sebaliknya, jika pemain merasa mudah mengalahkan musuh,

---

<sup>5</sup> Andrew, Adithya Nugraha Tjokrosetio, Andry Chowanda, *Dynamic Difficulty Adjustment With Facial Expression Recognition For Improving Player Satisfaction In A Survival Horror Game*, 2020, hlm 2.

- DDA dapat meningkatkan kecerdasan atau keterampilan musuh agar lebih menantang.
3. Pengaturan Keseimbangan Permainan: DDA juga dapat digunakan untuk mengatur keseimbangan permainan agar lebih sesuai dengan preferensi pemain. Misalnya, pemain dapat memilih untuk fokus pada aspek tertentu dari game, seperti eksplorasi, pertarungan, atau cerita. DDA dapat mengatur tingkat kesulitan dan frekuensi tantangan yang sesuai dengan preferensi pemain.
  4. Penyesuaian Tantangan Sampingan: DDA dapat membantu dalam menyesuaikan tantangan sampingan atau misi tambahan yang ada dalam game. Jika pemain kesulitan menyelesaikan misi sampingan tertentu, DDA dapat menawarkan opsi alternatif atau menurunkan tingkat kesulitan misi tersebut.
  5. Saran dan Petunjuk: DDA juga dapat memberikan saran dan petunjuk kepada pemain jika mereka mengalami kesulitan. Misalnya, DDA dapat menawarkan petunjuk atau tips untuk melewati bagian game yang sulit atau memberikan informasi tentang strategi yang efektif.

Pengaplikasian DDA dalam game dapat memberikan pengalaman bermain yang lebih dinamis dan disesuaikan dengan kemampuan dan preferensi masing-masing pemain. DDA membantu menjaga pemain terlibat dan terus bertantang, tanpa membuat mereka merasa terlalu mudah atau terlalu kesulitan<sup>6</sup>. DDA menjadi salah satu cara yang inovatif dan adaptif dalam meningkatkan kualitas gameplay dan kepuasan pemain dalam game modern.

## 2.2 Game Horor

Perbedaan paling mendasar antara game horor dan *thriller* adalah temanya. Game horor biasanya mengambil tema supernatural, mitos, legenda, dan hal-hal berbau mistis lainnya yang umumnya sulit dibuktikan kebenarannya dan sering terlihat kurang masuk akal.

---

<sup>6</sup> Demediuk, S., Tamassia, M., Raffe, W. L., Zambetta, F., Mueller, F. F., & Li, X., *Measuring player skill using dynamic difficulty adjustment*, In Proceedings of the Australasian Computer Science Week Multiconference, 2018, hlm 1-7.

Game bergenre horror merupakan genre permainan video yang bertujuan untuk menciptakan atmosfer menakutkan dan emosi tegang pada pemainnya. Fokus utama dari game ini adalah menghadirkan pengalaman yang intens dan mendebarkan, yang melibatkan pemain dalam lingkungan gelap dan mencekam dengan cerita yang menarik.

Salah satu aspek penting dari game horor adalah atmosfer yang diciptakan. Penelitian yang dilakukan oleh Lopes, P., Liapis, A., & Yannakakis, G. (2015) menunjukkan bahwa desain lingkungan yang gelap, terpencil, dan misterius memiliki efek langsung pada respons emosional pemain<sup>7</sup>. Atmosfer seperti ini dapat meningkatkan tingkat kecemasan dan ketegangan saat bermain.

Selain itu, game horor juga sering mengandalkan cerita-cerita menakutkan dan tema-tema supernatural. Menurut penelitian oleh E Kirkland (2009) tentang efek cerita menakutkan pada pemain, cerita-cerita yang melibatkan misteri, kejahatan, dan roh jahat dapat membangkitkan perasaan takut dan kecemasan yang intens pada pemain, sehingga membuat mereka merasa lebih terlibat dalam permainan<sup>8</sup>.

Dalam game horor, penggunaan musik dan efek suara juga memainkan peran penting dalam menciptakan atmosfer yang menegangkan. Penelitian oleh ROBERTS, Rebecca (2014) menunjukkan bahwa penggunaan suara yang efektif, seperti bunyi-bunyian misterius dan suara nafas berat, dapat meningkatkan tingkat ketegangan pada pemain dan memberikan pengalaman yang lebih mendalam dalam permainan<sup>9</sup>.

Secara keseluruhan, genre game horror menawarkan pengalaman yang unik dan mendebarkan bagi para pemainnya. Melalui desain lingkungan yang menakutkan, cerita yang menarik, penggunaan musik dan suara yang efektif, serta mekanik penyamaran, game horor mampu menciptakan atmosfer yang

---

<sup>7</sup> Lopes, P., Liapis, A., & Yannakakis, G. *Targeting horror via level and soundscape generation*, In Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment. Vol. 11, No. 1, 2015, hlm 37-43.

<sup>8</sup> Kirkland, E, *Storytelling in survival horror video games*, Horror video games: Essays on the fusion of fear and play, (2009), hlm 62-78.

<sup>9</sup> ROBERTS, Rebecca, *Fear of the unknown: Music and sound design in psychological horror games*, In: Music In Video Games. Routledge, 2014, hlm 138-150.

menegangkan dan menghadirkan pengalaman bermain yang penuh emosi bagi para pemainnya.

### 2.3 Game Side Scrolling

Video game sidescrolling adalah jenis permainan video di mana aksi permainan terjadi di bidang dua dimensi (2D) dan pandangan pemain mengikuti pergerakan karakter atau objek dari sisi. Dalam game sidescrolling, tampilan game akan terus bergerak secara horizontal dari kiri ke kanan (atau sebaliknya) seiring karakter atau objek utama bergerak melalui lingkungan game.

Ciri khas dari video game sidescrolling adalah pemain memiliki kendali atas pergerakan karakter atau objek secara horizontal untuk bergerak maju, mundur, melompat, dan berinteraksi dengan lingkungan serta musuh dalam game. Game sidescrolling sering kali menampilkan latar belakang yang bergulir secara horizontal untuk memberikan ilusi pergerakan yang kontinu, meskipun lingkungan sebenarnya terbatas pada layar dua dimensi.

Game sidescrolling telah menjadi populer sejak era konsol dan arkade pertama kali muncul pada tahun 1980-an hingga saat ini. Beberapa contoh game sidescrolling yang terkenal adalah "Super Mario Bros." dari Nintendo, "Sonic the Hedgehog" dari Sega, dan "Castlevania" dari Konami.

Keuntungan dari game sidescrolling adalah sederhana dalam mekanik permainan, sehingga mudah dipahami oleh pemain, namun tetap menawarkan tantangan dan kegembiraan. Tampilan sederhana dan desain karakter yang ikonik juga membuat game jenis ini mudah diakses oleh berbagai kalangan pemain, dari pemula hingga pemain berpengalaman.

Selama bertahun-tahun, game sidescrolling telah berevolusi dan menggabungkan berbagai elemen permainan, seperti elemen RPG (peran), platforming, aksi, dan bahkan elemen game pahlawan. Beberapa game sidescrolling modern juga menambahkan elemen tiga dimensi (3D) untuk memberikan pengalaman yang lebih mendalam dan memikat. Meskipun demikian, game sidescrolling tetap mempertahankan esensi gameplay-nya yang khas, yakni menjelajahi dunia horizontal dari sisi.

## 2.4 Dynamic Scripting

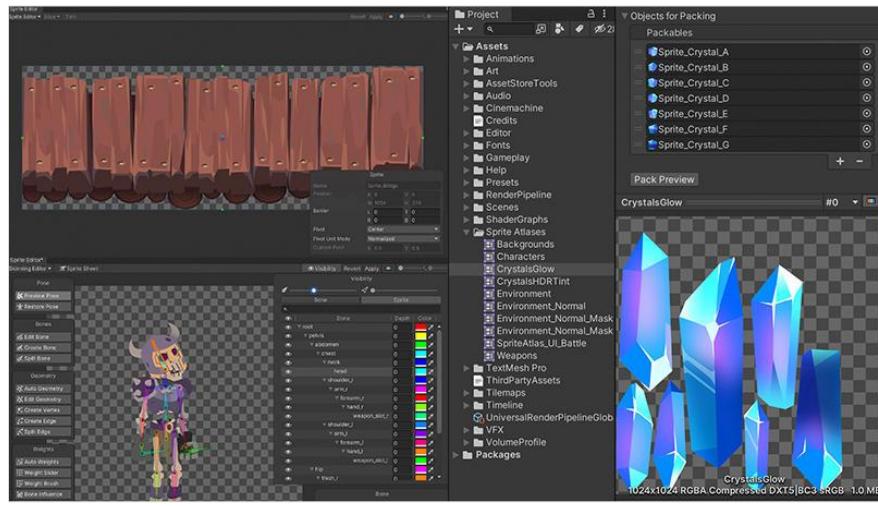
Dynamic scripting merupakan istilah yang digunakan secara luas dalam ilmu komputer untuk menjabarkan tingkatan dari bahas pemrograman yang mengeksekusi pada saat program berjalan (runtime) atas suatu struktur kode yang oleh bahasa lain dilakukan pada saat kompilasi. Perilaku ini termasuk pula pada ekstensi atas sebuah program, dengan cara menambahkan kode, dengan mengembangkan sebuah objek serta definisinya, atau mengubah suatu tipe data tertentu, kesemuanya dilakukan pada saat program berjalan. Perilaku seperti ini pada dasarnya dapat diemulasikan pada bahasa pemrograman apapun dengan tingkat kompleksitas yang berbeda-beda, hanya saja bahasa-bahasa pemrograman dinamis biasanya memiliki perangkat yang khusus didisain untuk kebutuhan tersebut

## 2.5 Mesin Permainan Unity 2D

Subbab ini membahas tentang apa itu Unity terkhususnya game engine Unity 2D, dimulai dengan penjelasan penjelasan seputar Unity 2D, fitur-fitur didalamnya, Barracuda SDK dan MoodME 4 Emotion Barracuda SDK. Dengan adanya subbab ini penulis terbantu untuk mempelajari secara cepat dan tepat mengenai game engine.

### 2.5.1 Unity 2D

Unity 2D adalah salah satu engine pengembangan perangkat lunak pembuat permainan dan aplikasi berbasis grafis dua dimensi. Engine ini dikembangkan oleh Unity Technologies, memiliki fokus khusus pada pengembangan permainan dengan tampilan 2D yang menarik dan interaktif. Dalam Unity 2D, elemen-elemen grafis seperti karakter, objek, dan latar belakang direpresentasikan sebagai sprite, yang merupakan gambar dua dimensi. Unity menyediakan editor yang kuat dan intuitif, serta memiliki kemampuan cross-platform, memungkinkan permainan yang dibuat dengan Unity 2D dapat dijalankan di berbagai platform.



**Gambar 2.1**  
**Tampilan Antarmuka Unity 2D**

Unity 2D banyak digemari oleh game developer karena sejumlah alasan yang membuatnya menjadi pilihan populer dalam pengembangan permainan 2D:

1. Mudah Dipelajari dan Digunakan: Unity 2D menawarkan antarmuka yang intuitif dan mudah dipelajar. Dokumentasi yang lengkap dan tutorial yang tersedia juga membuatnya lebih mudah bagi pemula untuk memulai dan mengembangkan permainan<sup>10</sup>.
2. Cross-platform: Unity 2D mendukung berbagai platform seperti PC, Mac, konsol game, perangkat seluler (smartphone dan tablet), dan platform lainnya<sup>11</sup>.
3. Komunitas Besar dan Dukungan: Unity memiliki komunitas yang besar dan aktif di seluruh dunia. Para pengembang dapat berbagi pengetahuan, sumber daya, dan dukungan melalui forum, blog, dan media sosial. Selain itu, Unity Asset Store menyediakan berbagai aset dan plugin yang dapat membantu dalam proses pengembangan permainan.
4. Fleksibilitas dan Customisasi: Unity 2D memberikan tingkat fleksibilitas dan customisasi yang tinggi. Pengembang dapat dengan mudah mengubah dan

<sup>10</sup> Unity Technologies, “Unity Documentation”, (<https://docs.unity.com/>, Diakses pada 20 Desember 2022)

<sup>11</sup> Unity Technologies, “Unity Manual – Platform development”, (<https://docs.unity3d.com/Manual/PlatformSpecific.html>, Diakses pada 21 Desember 2022)

mengatur elemen grafis, suara, musik, dan logika permainan sesuai dengan kebutuhan proyek mereka.

Kombinasi dari kemudahan penggunaan, dukungan cross-platform, komunitas aktif, dan fleksibilitas yang tinggi membuat Unity 2D menjadi pilihan yang menarik bagi banyak game developer untuk mengembangkan permainan 2D yang menarik, berkualitas tinggi, dan dapat diakses di berbagai platform. Melalui penggunaan Unity 2D sebagai landasan teori, diharapkan kemampuan untuk merancang, mengelola, dan mengembangkan permainan 2D yang menarik dan berkualitas tinggi dapat ditingkatkan secara signifikan.

### 2.5.2 Barracuda SDK

Barracuda merupakan library Deep Learning yang dirancang khusus untuk diintegrasikan dengan Unity. Library ini memungkinkan pengembang permainan menggunakan model Deep Learning yang telah dilatih untuk menambahkan kecerdasan buatan (artificial intelligence) dalam permainan<sup>12</sup>.

Dengan Barracuda, pengembang dapat menjalankan model Deep Learning pada platform Unity tanpa perlu tergantung pada plugin atau alat tambahan eksternal. Barracuda mendukung berbagai framework Deep Learning seperti TensorFlow, ONNX, dan Caffe. Hal ini memungkinkan pengembang untuk mengamplifikasi model yang telah dilatih menggunakan framework-framework ini ke dalam Unity dan menggunakan secara langsung dalam permainan mereka.

Barracuda SDK juga menawarkan keunggulan dalam hal kinerja dan efisiensi, memungkinkan penggunaannya dalam permainan dengan tingkat keterbatasan sumber daya yang rendah. Dengan dukungan untuk berbagai arsitektur perangkat keras, termasuk CPU, GPU, dan bahkan TPU (Tensor Processing Unit), Barracuda penerapan model Deep Learning sesuai dengan kebutuhan permainan dan platform yang dituju.

---

<sup>12</sup> Unity, “*Introduction to Barracuda*”, (<https://docs.unity3d.com/Packages/com.unity.barracuda@1.0/manual/index.html>, Diakses pada 12 Januari 2023)



**Gambar 2.2**  
**Pemanfaatan Library Barracuda**

Penggunaan Barracuda dalam Unity membuka berbagai kemungkinan untuk pengembangan permainan yang lebih cerdas dan kompleks. Pengembang dapat m实装implementasikan fitur-fitur seperti pengenalan objek, keputusan cerdas untuk musuh atau karakter non-pemain (NPC), dan berbagai tugas lainnya yang membutuhkan kecerdasan buatan.

Barracuda merupakan salah satu upaya dari Unity Technologies untuk memperluas kemampuan engine Unity dalam penggunaan teknologi kecerdasan buatan dan Deep Learning. Dengan adanya Barracuda, para pengembang permainan memiliki akses lebih mudah untuk menerapkan kecerdasan buatan dalam permainan mereka tanpa harus mengandalkan solusi pihak ketiga atau plugin eksternal.

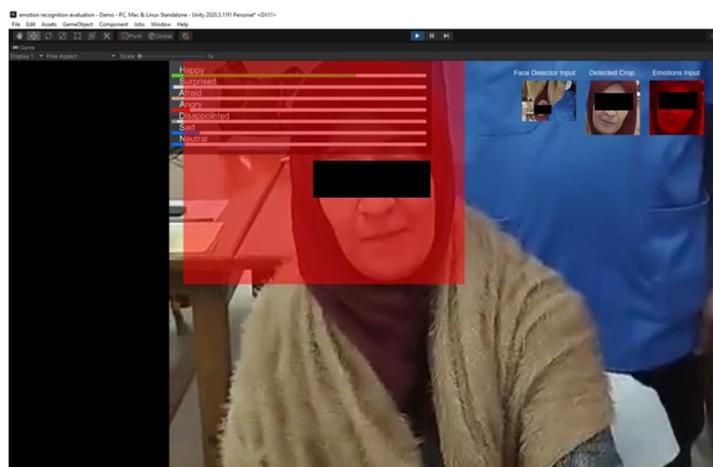
### 2.5.3 MoodMe 4 Emotion Barracuda SDK

MoodMe Unity Emotion Detection SDK dari MoodMe berfungsi sebagai plugin standar Unity 3D dan dapat dijalankan di semua platform yang didukung oleh Unity. SDK ini mengambil rekaman video dari kelas Webcam Texture standar Unity. Selain itu, SDK ini dapat diisi dengan gambar atau video jenis apapun yang dikonversi dalam format yang didukung (RGB, RGBA, BGRA, YUV, YUY2)<sup>13</sup>.

---

<sup>13</sup> MoodMe, “Unity AI - MoodMe”, (<https://www.mood-me.com/unity-ai/>, Diakses pada 8 Juni 2023)

Plugin MoodMe Unity Face Recognition SDK merupakan plugin multi-thread yang aman digunakan dan dapat di-skala untuk menggunakan setiap inti CPU yang tersedia. Plugin ini berjalan pada 60 FPS pada sistem mid-end dan hingga 85 FPS pada sistem high-end. Dukungan untuk CUDA dan OpenCL juga disediakan. Plugin Unity Face Recognition membuat data landmark 2D/3D 66/68 tersedia untuk Unity, serta menyediakan matriks transformasi kepala. Driver animasi dan retargeting juga tersedia.



**Gambar 2.3**  
**Penggunaan MoodMe Unity**

MoodMe Unity Face Recognition Emotion detection SDK memungkinkan Pengembang Game menciptakan skenario di mana mood pemain dapat dideteksi dan diterapkan pada karakter in-game mereka secara real-time. Selain itu, mood pemain dideteksi dan dibuat tersedia sebagai 7 nilai berbeda (senang, terkejut, marah, sedih, takut, jijik, netral).

Pengalaman bermain game dapat diperkaya dengan skenario adaptif yang disesuaikan dengan perasaan pemain. Sementara banyak game melakukan estimasi, perusahaan game yang menggunakan Unity3D dan keluarga SDK MoodMe Unity Face Recognition sekarang dapat membuat game yang beradaptasi dengan mood para pemain.

#### 2.5.4 Interaksi MoodMe dengan Barracuda

MoodMe dan Barracuda Unity SDK adalah dua teknologi yang dapat digunakan bersama-sama dalam pengembangan aplikasi dengan Unity. Berikut adalah penjelasan yang lebih mendetail tentang keduanya dan bagaimana MoodMe dan Barracuda dapat bekerja sama:

1. MoodMe Unity: MoodMe adalah sebuah platform yang menyediakan solusi pengenalan wajah dan emosi untuk aplikasi dan game. Dengan menggunakan teknologi Augmented Reality (AR) dan Machine Learning, MoodMe dapat menganalisis ekspresi wajah pengguna secara real-time dan memberikan umpan balik yang sesuai. Dalam game horor, MoodMe dapat digunakan untuk mengukur respons emosional pemain saat mereka menghadapi situasi menegangkan atau menakutkan. Misalnya, MoodMe dapat mendeteksi ekspresi wajah pemain saat mereka dihadapkan pada adegan mencekam, dan berdasarkan respons emosional tersebut, game dapat menyesuaikan tingkat ketegangan atau intensitas cerita untuk menciptakan pengalaman yang lebih personal dan menegangkan.

MoodMe juga dapat digunakan untuk menciptakan karakter dalam game horor yang merespons secara real-time terhadap ekspresi wajah pemain. Karakter dalam game dapat bereaksi terhadap ekspresi wajah pemain, seperti menangis, tertawa, atau menunjukkan ekspresi takut, sehingga menciptakan interaksi yang lebih mendalam antara pemain dan karakter dalam game.

2. Barracuda Unity SDK: Barracuda adalah sebuah Unity SDK yang dirancang untuk menjalankan model-model kecerdasan buatan (AI) langsung di dalam Unity. SDK ini memungkinkan pengembang untuk mengintegrasikan model-model AI yang telah dilatih, seperti jaringan saraf tiruan (neural networks), untuk melakukan tugas-tugas seperti pengenalan gambar, pemrosesan bahasa alami, atau prediksi. Barracuda Unity SDK, dengan kemampuannya untuk menjalankan model-model kecerdasan buatan (AI) di dalam Unity, dapat menyediakan data neural network yang dapat digunakan dalam integrasi dengan MoodMe. Dalam konteks penggunaan bersama MoodMe, Barracuda dapat memberikan akses ke model-model AI yang telah dilatih sebelumnya,

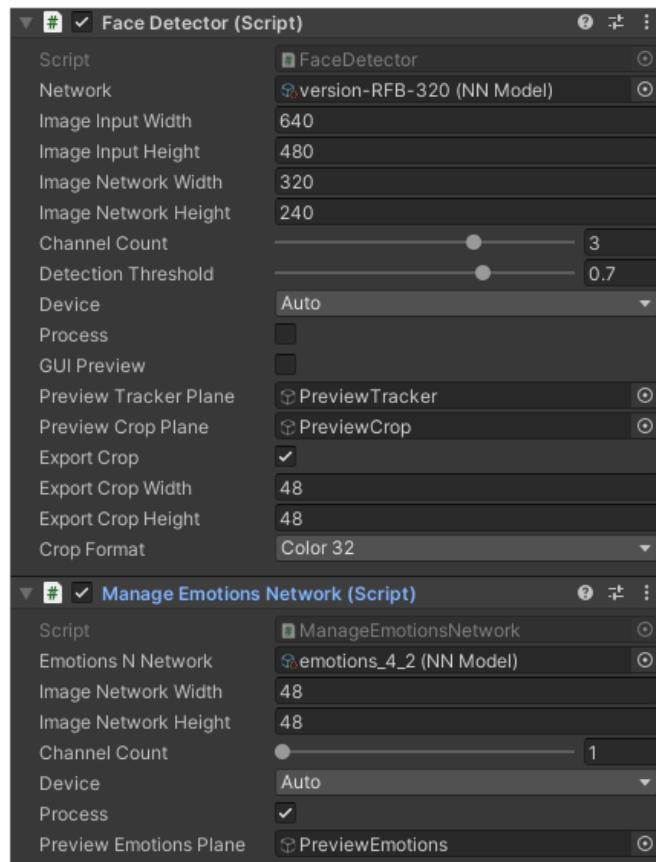
seperti jaringan saraf tiruan (neural networks), yang dapat digunakan untuk menganalisis data ekspresi wajah dari MoodMe secara lebih mendalam.

Dengan memanfaatkan data neural network dari Barracuda, MoodMe dapat meningkatkan kemampuannya dalam mengenali dan menganalisis ekspresi wajah pengguna dengan lebih akurat dan efisien. Misalnya, model neural network yang dijalankan oleh Barracuda dapat digunakan untuk mengidentifikasi pola-pola kompleks dalam ekspresi wajah yang mungkin sulit untuk dikenali secara manual, sehingga meningkatkan kualitas analisis emosi yang dilakukan oleh MoodMe.

Dalam konteks game, penggunaan MoodMe dan Barracuda Unity SDK dapat membawa pengalaman bermain game ke tingkat yang lebih responsif dan dinamis. Berikut adalah beberapa contoh penerapan integrasi MoodMe dengan Barracuda :

- Pengembang dapat menggunakan MoodMe dalam Unity untuk menambahkan fitur pengenalan wajah dan analisis emosi ke dalam aplikasi atau game mereka. MoodMe akan mengambil data dari kamera perangkat dan menganalisis ekspresi wajah pengguna secara real-time.
- Barracuda Unity SDK dapat digunakan untuk mengintegrasikan model AI yang telah dilatih ke dalam aplikasi Unity. Model ini bisa melakukan berbagai tugas, tidak terbatas pada pengolahan data visual atau audio yang diperoleh dari pengguna.
- Dengan menggabungkan kedua teknologi ini, pengembang dapat menciptakan aplikasi yang tidak hanya merespons terhadap ekspresi wajah pengguna tetapi juga menggunakan model AI untuk memperkaya pengalaman pengguna. Misalnya, sebuah game yang menggunakan MoodMe dapat menyesuaikan kesulitan atau narasi berdasarkan respons emosional pengguna yang dianalisis oleh AI.
- MoodMe dapat digunakan untuk membuat keputusan yang lebih kompleks dalam aplikasi berdasarkan pemrosesan informasi yang didapat melalui model Barracuda. Ini memungkinkan pengembangan aplikasi yang sangat

interaktif dan personal, di mana AI dapat menyesuaikan konten atau perilaku berdasarkan analisis emosi pengguna secara real-time.



**Gambar 2.4**  
**Penggunaan Neural Network yang Berasal Dari Barracuda**  
**Didalam Script MoodMe**

Dengan menggabungkan kemampuan analisis emosi wajah dari MoodMe Unity dan kecerdasan buatan dari Barracuda Unity SDK, pengembang game horor dapat menciptakan pengalaman bermain yang lebih menarik, menegangkan, dan personal bagi pemain. Kombinasi antara teknologi ini dapat membawa game horor ke tingkat baru dengan interaksi yang lebih mendalam antara pemain dan game, serta pengalaman bermain yang lebih responsif dan menantang. Dengan demikian, keterkaitan antara MoodMe Unity dan Barracuda Unity SDK terletak pada kemampuan mereka untuk saling melengkapi dalam menciptakan aplikasi yang responsif dan interaktif. MoodMe mengolah data emosional, sementara Barracuda

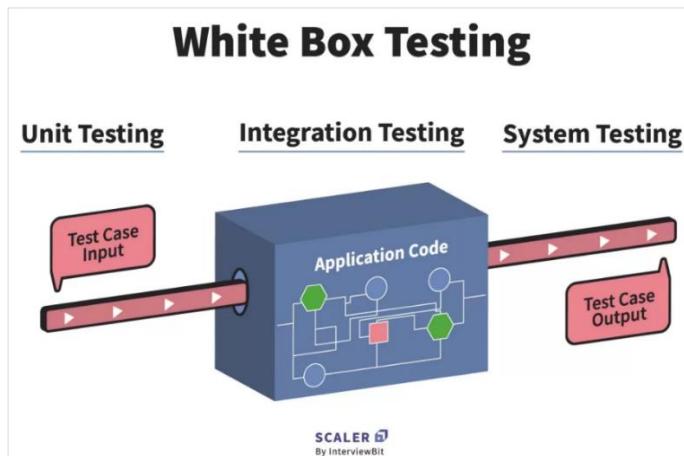
memungkinkan penggunaan data wajah pemain dalam konteks yang lebih luas melalui AI.

## 2.6 Metode Testing

Testing dalam pengembangan perangkat lunak adalah suatu proses yang dilakukan untuk mengevaluasi kinerja, dan kualitas keseluruhan dari sebuah software. Tujuan utama dari pengujian adalah untuk mengidentifikasi dan mengatasi potensi bug. Proses ini tidak hanya membantu dalam meminimalkan risiko kegagalan, tetapi juga memastikan bahwa pengembangan sesuai dengan standar yang ditetapkan.

### 2.6.1 Whitebox Testing

White box testing, adalah metode pengujian perangkat lunak yang melibatkan pemeriksaan dan evaluasi struktur internal dari *source code*. Dibutuhkan pemahaman yang mendalam terhadap logika dan implementasi program yang diuji.



**Gambar 2.5**  
**Alur dari White Box Testing**

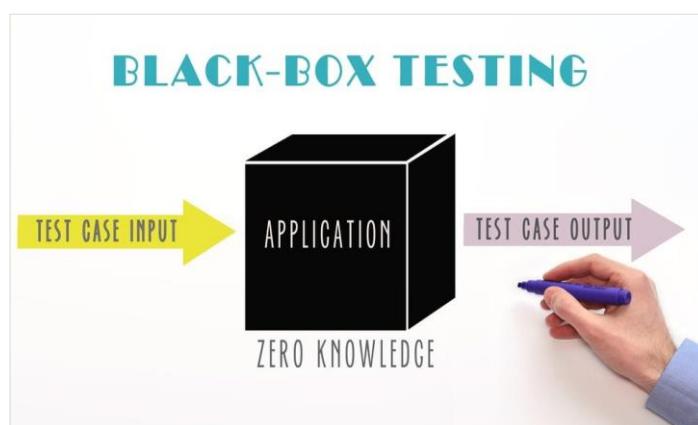
Tujuan utama dari white box testing adalah untuk memastikan bahwa semua jalur eksekusi di dalam program telah diuji dan setiap kondisi logika telah dievaluasi. Dengan demikian, white box testing membantu mengidentifikasi bug,

kesalahan logika, dan kekurangan dalam desain kode. Metode ini dapat diintegrasikan selama siklus pengembangan perangkat lunak untuk memastikan kualitas dan keandalan perangkat lunak yang dihasilkan.

Beberapa teknik yang sering digunakan dalam white box testing melibatkan analisis basis path, analisis kondisi, dan pengujian batas. Analisis basis path fokus pada evaluasi setiap jalur eksekusi yang mungkin di dalam program, sementara analisis kondisi memastikan bahwa setiap kondisi logika diverifikasi dengan benar. Pengujian batas, di sisi lain, mengevaluasi respons program terhadap input yang berada di batas kondisi.

### 2.6.2 Blackbox Testing

Black box testing adalah metode pengujian perangkat lunak tanpa memperhatikan struktur internal atau logika kode. Pendekatan ini lebih fokus pada input dan output yang dihasilkan oleh sistem tanpa mempertimbangkan bagaimana perangkat lunak mencapai hasil tersebut. Tujuan utama dari black box testing adalah untuk memastikan bahwa perangkat lunak memenuhi persyaratan spesifikasi fungsional dan memberikan hasil yang sesuai dengan harapan.



**Gambar 2.6**  
**Alur dari Black Box Testing**

Penguji black box tidak memiliki pengetahuan mendalam tentang implementasi atau desain internal perangkat lunak yang diuji. Mereka hanya fokus

pada fungsionalitas eksternal dan perilaku program. Beberapa teknik yang sering digunakan dalam black box testing melibatkan pengujian ekstensif atas berbagai jenis input untuk melihat respons yang dihasilkan, serta pengujian batas untuk memastikan bahwa perangkat lunak dapat menangani kondisi ekstrem dengan baik.

Keuntungan utama dari black box testing adalah bahwa ini memungkinkan penguji untuk fokus pada persyaratan fungsional dan pengalaman pengguna tanpa perlu mengetahui detail implementasi internal. Namun, kelemahannya adalah bahwa penguji mungkin tidak dapat mengidentifikasi kesalahan atau masalah yang mungkin terjadi di dalam kode.

## **BAB III**

### **PERANCANGAN DAN ANALISA SISTEM**

Bab ini menjelaskan mengenai tahap analisa game dan rancangan system pada pembuatan game. Tahapan dilakukan mulai dari perencanaan, analisa, penentuan elemen-elemen yang ada didalam game hingga perancangan metode emotion detection yang digunakan, dan rancangan parameter yang akan digunakan didalam game horror 2D skenario Alas Tilas, Jawa Timur.

#### **3.1 Perencanaan**

Dalam perencanaan penulis menganalisa hal yang dibutuhkan untuk membangun game, yaitu sebuah game yang dapat digunakan sebagai sarana untuk melestarikan cerita rakyat Indonesia dan game menjadi menarik untuk dimainkan. Game yang akan dibangun dalam tugas akhir ini adalah sebuah game cerita rakyat yang mengambil tema horror Alas Tilas, Jawa Timur. Game yang dibangun menggunakan bahasa pemrograman C# dan Unity sebagai game enginennya.

Berdasarkan judul yang dibuat yaitu fokusnya kepada produk game yang memanfaatkan emosi, maka hasil akhir dari game ini adalah sebuah game dengan judul “Game Horror 2D Skenario Alas Tilas, Jawa Timur” yang dapat menyesuaikan tingkat kesulitan level berdasarkan ekspresi yang terdeteksi pada wajah pemain dengan tema horror dan cerita horror Indonesia. Dengan adanya game horror yang mengambil tema Alas Tilas, Jawa Timur dengan tingkat kesulitan yang dapat beradaptasi dengan pemain, game ini diharapkan mampu menarik minat penikmat game horror di Indonesia.

#### **3.2 Analisa Game Horror 2D Skenario Alas Tilas, Jawa Timur**

Analisa game didefinisikan sebagai penguraian dari suatu game yang utuh ke dalam bagian komponennya dengan tujuan untuk mengidentifikasi dan mengevaluasi permasalahan. Analisa bertujuan mendapatkan pemahaman secara keseluruhan tentang game yang akan dibuat. Game horror ini diciptakan sebagai

media hiburan bagi penikmat game horror dan pemain yang hendak mencoba bermain game dengan genre horror.

### **3.3 Elemen Dalam Game**

Menurut Teresa Dillon terdapat 8 elemen-elemen dasar yang umumnya ada didalam game yaitu:

1. Peraturan Permainan

Peraturan permainan mencakup instruksi atau suatu aturan perintah, peran objek dan karakter dan metode penggunaanya didalam dunia permainan. Peraturan di game ini yaitu setelah mengklik tombol bermain pemain akan ditempatkan posisi awal pada hutan yang berbeda-beda setiap level. Pemain harus menginjak checkpoint untuk melanjutkan ke level selanjutnya.

2. Plot

Plot harus menguraikan dengan rinci aktivitas yang perlu dilakukan oleh pemain dalam game dan menjelaskan tentang tujuan yang harus dicapai. Dalam pembuatan game ini, menceritakan tentang seorang pendaki yang terpisah dari kerumunan pendaki lainnya dan terjebak di tengah hutan yang digambarkan sebagai lokasi Alas Tilas, Jawa Timur. Maka dari itu karakter utama harus berhasil bertahan dari rintangan dan gangguan dari makhluk supernatural serta mencari jalan keluar dari hutan belantara tersebut.

3. Objektifitas

Objektif merupakan sebuah hal yang penting dan biasanya digunakan dalam bentuk misi bagi pemain dan dituntut memecahkan masalah atau misi tersebut, sewaktu-waktu pemain dituntut punya keahlian dan pengetahuan untuk bisa memaninkannya. Pada game ini pemain dituntut untuk berhasil menyelesaikan game yang memiliki 3 level dengan melewati rintangan yang ada sepanjang level dan mencari jalan keluar hingga berhasil meloloskan diri dari hutan yang angker tersebut.

4. Karakter

Karakter disini yaitu karakter utama dan karakter lain-lain. Model permainan game horror ini yaitu single player. Pemain mengendalikan satu karakter yaitu

sang pendaki itu sendiri. Karakter yang tidak bisa di kendalikan atau yang umumnya disebut NPC (Non Playable Character) berupa hantu-hantu yang mengganggu pemain selama menyelesaikan tiap rintangan yang diberikan oleh sistem dalam game.

#### 5. Teks, grafik, dan suara

Pada umumnya pada game terdapat elemen grafik, suara, dan narasi. Hal tersebut dapat dimunculkan secara berkombinasi maupun terpisah. Meskipun mayoritas game yang beredar memenuhi seluruh aspek tersebut, kadangkala tidak selalu diperlukan elemen tersebut muncul secara bersamaan selama game dijalankan. Pada game ini unsur narasi terdapat pada banyak hal, salah satunya pada narasi pembuka dan penutup yang menceritakan kronologis karakter game.

#### 6. Tema

Tema game merupakan sesuatu yang membuat kehendak atau tujuan yang mau disampaikan oleh pembuat. Tema game yang dipilih merupakan game horor dengan mengambil suasana hutan belantara dengan vegetasi Jawa Timur yang disertai dengan hantu yang kerap dibahas didalam mitos hantu Jawa salah satunya yaitu kuntilanak. Tema ini diambil karena penulis hendak mempromosikan game horror dengan tema kearifan lokal.

#### 7. User Interface

User interface merupakan tampilan antarmuka yang mempermudah pemain berinteraksi dengan game. User interface dibuat selain membuat game menjadi lebih menarik, tujuan lain dibuatnya user interface yaitu memudahkan pemain untuk beradaptasi selama sesi permainan dijalankan. Game horror ini memiliki tampilan antarmuka pengguna yaitu dari menu utama, Sinopsis, Play, Cara Bermain, dan Keluar. Sedangkan saat didalam level, terdapat tampilan status pemain, item yang dibawa karakter, dan tombol interaksi pada objek.

#### 8. Animasi

Animasi merupakan hal yang harus ada pada dunia game, khususnya untuk gerakan karakter - karakter yang ada dalam properti dari objek game. Dengan adanya animasi, pemain dimudahkan untuk memahami suatu situasi yang akan

terjadi maupun apa yang sedang terjadi saat ini. Game ini memiliki banyak animasi pada karakter utama, objek yang ada dan hantu yang mengganggu sepanjang permainan.

### **3.4 Analisa Kebutuhan**

Analisis kebutuhan digunakan untuk mempermudah menganalisis sebuah aplikasi atau game yang dibutuhkan, ada dua jenis kebutuhan sistem :

#### **3.4.1 Analisa Kebutuhan Hardware**

Spesifikasi hardware yang digunakan dalam pembuatan Game Alas Tilas, Jawa Timur sebagai berikut :

- a. Processor : Ryzen 5 3500H
- b. Memory : 8 GB
- c. VGA : GTX1050 3GB Mobile
- d. Camera : Camera HD Laptop
- e. Perangkat Lainnya : Keyboard & Mouse

Sedangkan spesifikasi minimum pengguna agar dapat menjalankan game dengan lancar yaitu :

- a. CPU : Core I3 7xxx atau lebih tinggi
- b. Memory : 8 GB
- c. VGA : GT710 atau Radeon HD5500
- d. Camera : Minimum support resolusi 320p
- e. Perangkat Lainnya : Keyboard & Mouse

#### **3.4.2 Analisa Kebutuhan Software**

Software yang digunakan selama pembuatan Game Alas Tilas, Jawa Timur yaitu :

- a. Unity 2D
- b. Visual Studio Code
- c. Barracuda Unity SDK

d. MoodME Emotion Detector Unity

### **3.5 Perancangan Game**

Pada subbab ini, akan memasuki perancangan game horror Alas Tilas, Jawa Timur. Subbab ini membahas detail-detail penting terkait perancangan game, termasuk konsep, plot, usecase, cara bermain, karakter, dan elemen-elemen lain yang akan disajikan kepada pemain sehingga pengalaman yang didapat menjadi unik dan menantang.

#### **3.5.1 Konsep**

Konsep yang menjadi cerita dari game menggunakan papan dasar yang diperoleh dari sumber berita misteri<sup>14</sup>. Jejak Gunung Lali Jiwo Arjuno, Jawa Timur, sebagai lokasi yang dilaporkan memiliki hal-hal mistis di mana ditemukan insiden hilangnya pendaki dan tim SAR pendaki kemudian hanya ditemukan tulang-benulangnya saja.

Pemain dituntut untuk berhasil keluar dari hutan belantara yang ada pada Gunung Lali Jiwo tersebut. Indikator keberhasilannya yaitu pemain berhasil melewati 3 level yang diberikan. Setiap level yang muncul digenerate secara otomatis oleh sistem dengan menyesuaikan dengan data pencatatan ekspresi wajah pemain serta performa pemain dalam melewati rintangan yang diberikan.

#### **3.5.2 Plot Cerita**

Plot cerita yang dirancang menceritakan tentang seorang pendaki yang terpisah dari kerumunan pendaki lainnya dan terjebak di tengah hutan yang digambarkan sebagai lokasi Alas Tilas, Jawa Timur. Pendaki mencoba untuk melarikan diri dari hutan belantara yang dilaluinya dengan selamat dengan melalui

---

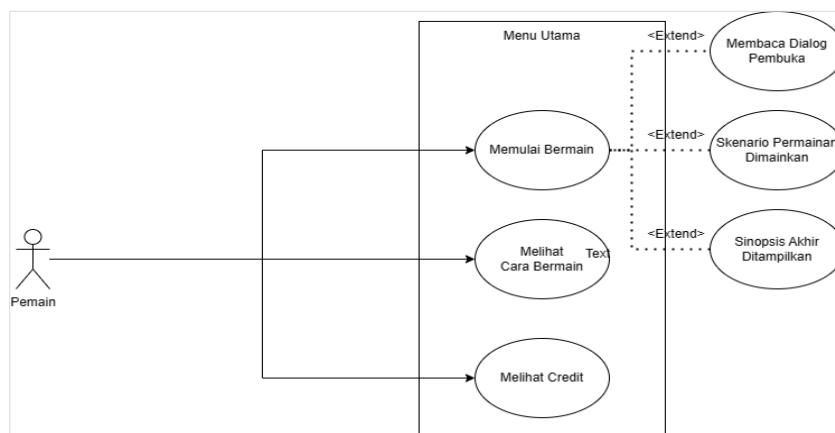
<sup>14</sup> Ihwal.Id, “Mengungkap Misteri Gunung Arjuna, 3 Penyebab Pendaki Mudah Tersesat dan Hilang di Alas Lali Jiwa”, (<https://www.ihwal.id/info/68211238086/mengungkap-misteri-gunung-arjuna-3-penyebab-pendaki-mudah-tersesat-dan-hilang-di-alas-lali-jiwa/>), Diakses pada 20 September 2022)

banyak rintangan dan hantu yang menghuni dan mengganggu manusia yang tersesat di dalam hutan tersebut.

Nahasnya meskipun pendaki merasa berhasil kabur dari hutan belantara yang dipenuhi hantu tersebut, rupanya itu merupakan halusinasi pendaki tersebut yang diakibatkan dari kekurangan darah setelah kecelakaan saat mencari jalan keluar dari hutan tersebut. Game ini mengambil cerita regional agar akrab dengan kehidupan sehari-hari pemain dan diharapkan memberikan efek manifestasi berlebihan pada pemain nantinya.

### 3.5.3 Use Case Diagram

Diagram use case adalah alat visual dalam rekayasa perangkat lunak yang berfungsi untuk menggambarkan interaksi antara pengguna dengan sistem yang akan dikembangkan. Dengan use case diagram, pengembang dapat mengidentifikasi fungsi atau tindakan yang akan dilakukan oleh pengguna dalam konteks sistem game. Ini adalah komponen penting analisis sistem yang membantu pengembangan memahami kebutuhan pengguna dan merencanakan fungsionalitas sistem dengan lebih baik. Berikut merupakan usecase diagram game yang dibuat.



**Gambar 3.1**  
**Diagram Use Case Game**

#### 1. Skenario Use Case Memulai Bermain

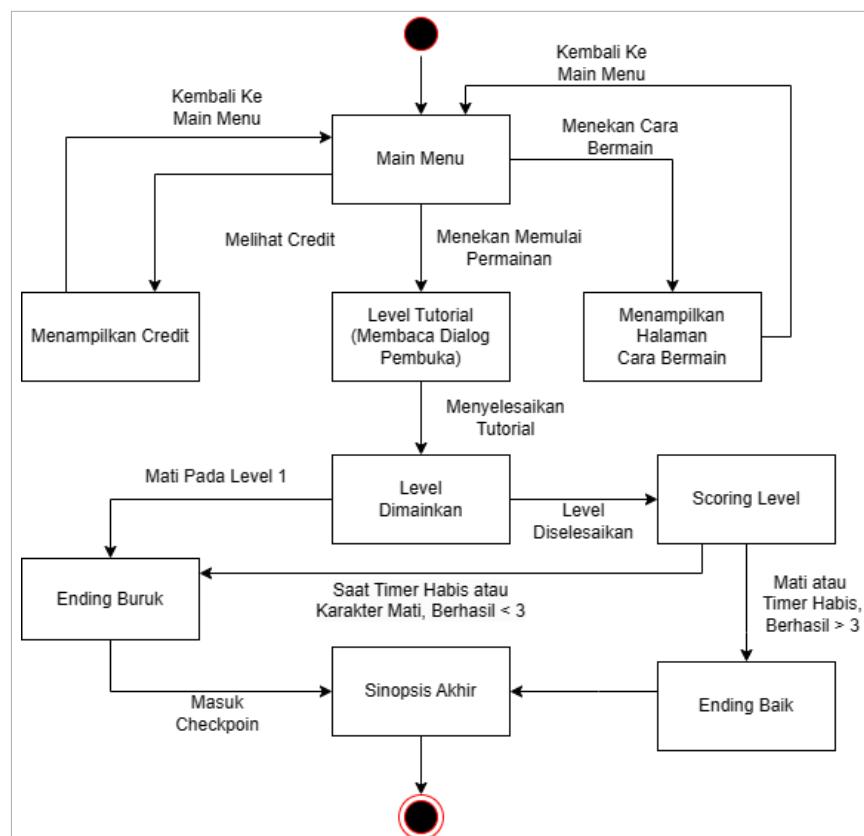
Aktor : Pemain

Nama Use Case : Memulai bermain

- Deskripsi : Pemain akan memulai bermain dari level 1 hingga game berhasil diselesaikan.
2. Skenario Use Case Mempelajari Bagaimana Cara Bermain
- Aktor : Pemain
- Nama Use Case : Mempelajari Cara Bermain
- Deskripsi : Memunculkan halaman yang memberi tahu tombol arah gerak dan hal-hal yang dapat dilakukan oleh pemain.
3. Skenario Use Case Membaca Credits
- Aktor : Pemain
- Nama Use Case : Membaca Credits
- Deskripsi : Memunculkan halaman yang memberi siapa saja yang berkontribusi selama pembuatan game ini.
4. Skenario Use Case Membaca Latar Belakang Karakter Pemain
- Aktor : Pemain
- Nama Use Case : Membaca Dialog Pembuka
- Deskripsi : Menampilkan dialog atau narasi pembuka yang memberikan latar belakang cerita sekaligus berfungsi sebagai tutorial kepada pemain.
5. Skenario Use Case Permainan Dimainkan
- Aktor : Pemain
- Nama Use Case : Skenario Permainan Dimainkan
- Deskripsi : Pemain berinteraksi dengan permainan setelah membaca dialog pembuka dan memulai permainan.
6. Skenario Use Case Penampilan Sinopsis Akhir Karakter Pemain
- Aktor : Pemain
- Nama Use Case : Sinopsis Akhir Ditampilkan
- Deskripsi : Menampilkan sinopsis akhir atau ringkasan cerita karakter setelah pemain menyelesaikan permainan.

### 3.5.4 State Diagram

State diagram merupakan diagram yang digunakan untuk menggambarkan aliran atau perpindahan keadaan dari suatu sistem. Diagram ini terdiri dari serangkaian keadaan, serta transisi antara keadaan yang digambarkan dengan panah. Dengan menggunakan state diagram, pengembang dapat dengan jelas memahami alur kerja sistem, memperkirakan kemungkinan keadaan yang terjadi, dan merancang logika yang sesuai. Berikut merupakan state diagram game yang dibuat.



**Gambar 3.2**  
**Diagram State Game**

Diagram diatas menggambarkan alur interaksi pengguna dengan permainan. Setiap kotak mewakili sebuah "state" atau kondisi dalam aplikasi/permainan, dan panah menunjukkan transisi antara state tersebut berdasarkan tindakan pengguna. Penjelasan setiap state akan dijabarkan dibawah ini :

1. Main Menu

Ini adalah titik awal ketika pengguna memulai game. Dari sini, pengguna dapat memilih untuk memulai permainan, melihat cara bermain, atau melihat credit dari aplikasi/permainan.

2. Menampilkan Halaman Cara Bermain

Jika pengguna memilih untuk melihat cara bermain, game akan menampilkan halaman yang menjelaskan aturan dan mekanisme permainan.

3. Menampilkan Credit

Jika pengguna memilih untuk melihat credit, game akan menampilkan informasi tentang siapa saja yang terlibat dalam pembuatan game.

4. Level Tutorial (Membaca Dialog Pembuka)

Jika pengguna memilih untuk memulai permainan, mereka akan diarahkan ke level tutorial di mana mereka dapat membaca dialog pembuka yang menjelaskan cerita dari karakter pemain.

5. Level Dimainkan

Setelah melewati tutorial, pengguna akan memasuki fase di mana mereka mulai bermain level yang sebenarnya.

6. Scoring Level

Setelah menyelesaikan level, game akan menampilkan skor atau penilaian berdasarkan performa pengguna selama bermain.

7. Ending Buruk

Ini adalah layar atau adegan yang ditampilkan jika pemain mati pada level 1 atau pada saat timer habis, jumlah level yang diselesaikan pemain dibawah 3.

8. Ending Baik

Jika pengguna berhasil menyelesaikan 3 atau lebih level permainan pada waktu habis atau mati pada level 3 atau lebih. Pemain akan ditampilkan adegan karakter berhasil kabur dari hutan.

9. Sinopsis Akhir

Jika pengguna berhasil menyelesaikan level atau seluruh permainan, mereka akan disajikan dengan sinopsis atau rangkuman dari cerita atau hasil dari permainan.

### 3.6 Cara Bermain

Game ini dirancang untuk dimainkan pada platform windows dengan memanfaatkan perangkat keyboard. Tombol-tombol pada keyboard digunakan untuk menggerakan karakter ke kanan dan ke kiri, loncat, dan menunduk. Berikut merupakan penjelasan tombol-tombol yang digunakan sebagai kontrol pemain berserta dengan penjelasannya :

- Tombol (A,S,D)

Tombol A digunakan untuk menggerakkan karakter ke kiri sedangkan Tombol D digunakan untuk menggerakkan karakter ke kanan. Tombol S digunakan untuk membuat karakter bergerak secara menunduk atau merangkak.

Kombinasi tombol ini digunakan dikarenakan agar tangan kiri fokus terhadap pergerakan karakter. Tombol yang dipilih juga dekat dengan tombol kontrol Shift, E, dan Space. Dengan pengaturan tombol sedemikian rupa pemain dapat melakukan pergerakan dalam game secara leluasa dan lebih lincah. Selain itu game serupa juga menerapkan kombinasi tombol ini sehingga pemain tidak kesulitan untuk beradaptasi dengan kontrol pada game ini.

- Tombol (Shift)

Tombol Shift digunakan untuk membuat karakter bergerak lebih cepat dengan cara berlari. Tombol Shift dipilih karena posisi Tombol Shift yang strategis dengan posisi jari kelingking tangan kiri, sehingga diperoleh ergonomic yang baik. Selain itu pada game serupa Tombol Shift juga digunakan secara *default* sebagai Tombol untuk membuat karakter berlari.

- Tombol (Space)

Tombol Space digunakan untuk membuat karakter melakukan aksi loncat. Tombol untuk loncat ini krusial dalam game karena digunakan agar pemain dapat bermanuver melewati rintangan dan jalan yang tidak rata. Tombol Space dipilih karena posisi Tombol Space yang dekat dengan jari jempol tangan kiri, sehingga tidak mengganggu jari-jari tangan kiri. Selain itu pada game serupa juga menerapkan Tombol Space secara *default* sebagai Tombol untuk membuat karakter melompat.

- Tombol (E)

Tombol E digunakan untuk mengambil item yang telah disebar di sepanjang map. Pemilihan tombol ini merujuk pada game serupa yang menerapkan kontrol Tombol E sebagai Tombol untuk mengambil item maupun melakukan aksi tertentu didalam game.

- Tombol (O)

Tombol O digunakan untuk melakukan aksi memukul oleh karakter. Pemilihan tombol ini merujuk pada game serupa yang menerapkan kontrol action untuk jari kanan sebagai Tombol untuk menggunakan melakukan aksi menghancurkan rintangan didalam game.

- Tombol (1,2, dan 3)

Tombol 1,2, dan 3 digunakan untuk mengkonsumsi atau menggunakan item yang telah dikumpulkan oleh karakter. Pemilihan tombol ini merujuk pada game serupa yang menerapkan kontrol Tombol 1, 2, dan 3 sebagai Tombol untuk menggunakan item didalam game.

### 3.7 Klasifikasi Emosi

Paul Ekman adalah seorang psikolog yang terkenal dengan penelitiannya tentang ekspresi wajah dan emosi. Menurut teori Ekman, ada enam emosi dasar yang diakui secara universal, yaitu: kebahagiaan, kesedihan, kejutan, rasa jijik, kemarahan, dan ketakutan<sup>15</sup>. Ekman dan rekan-rekannya melakukan serangkaian penelitian yang menunjukkan bahwa ekspresi wajah untuk emosi dasar ini konsisten lintas budaya<sup>16</sup>.

Berikut adalah deskripsi singkat tentang emosi manusia marah, takut, dan netral (biasa saja) berdasarkan penelitian Ekman:

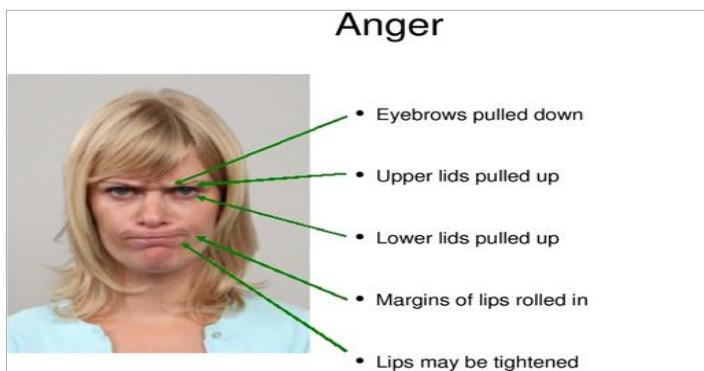
1. Marah (Anger): Emosi marah ditandai dengan kontraksi otot-otot tertentu di wajah, seperti mengernyitkan dahi, menatap tajam, mengencilkan mata, dan

---

<sup>15</sup> Ekman, P. *Emotions Revealed: Recognizing Faces and Feelings to Improve Communication and Emotional Life*. Times Books, 2003.

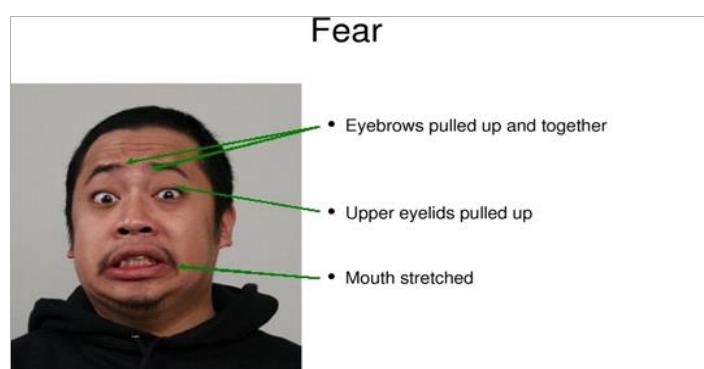
<sup>16</sup> Ekman, P., & Friesen, W. V. *Constants across cultures in the face and emotion*. Journal of Personality and Social Psychology. Vol 17(2), 1971, hlm 124–129.

kadang-kadang menunjukkan gigi atau mengetatkan mulut. Ekman menemukan bahwa ekspresi kemarahan memiliki fungsi adaptif, seperti mempersiapkan individu untuk bertindak melawan ancaman atau ketidakadilan.



**Gambar 3.3**  
**Karakteristik Wajah Marah**

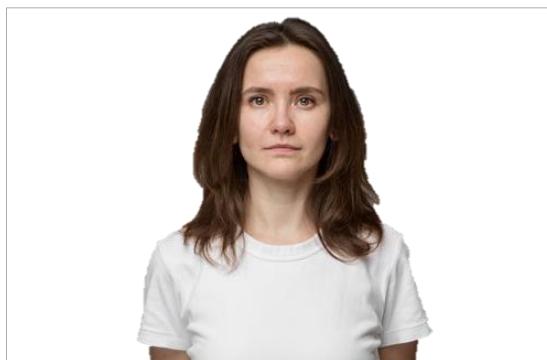
2. Takut (Fear): Ketakutan ditandai dengan ekspresi wajah yang melibatkan mengangkat alis dan memperlebar mata, seringkali untuk meningkatkan persepsi visual dan mempersiapkan tubuh untuk bereaksi terhadap potensi bahaya. Bibir bisa tertarik ke belakang atau sedikit terbuka sebagai bagian dari respons ini.



**Gambar 3.4**  
**Karakteristik Wajah Takut**

3. Netral (Neutral): Ekspresi netral bukanlah emosi khusus, tetapi lebih merupakan keadaan wajah tanpa aktivitas otot wajah yang signifikan. Wajah

netral sering digunakan sebagai titik referensi dalam penelitian untuk membandingkan dengan ekspresi emosi yang lebih aktif.

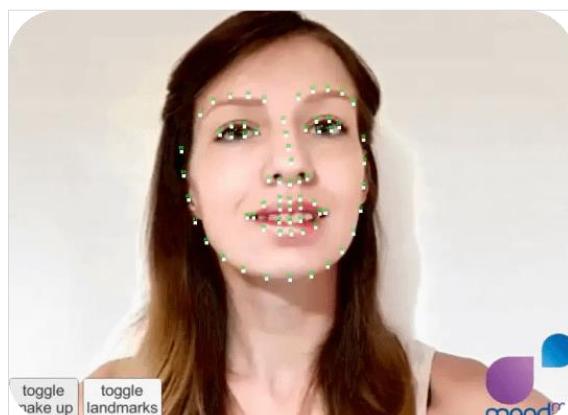


**Gambar 3.5**  
**Ilustrasi Wajah Netral**

Dengan memahami ekspresi wajah untuk emosi manusia yang telah diteliti oleh Paul Ekman, penulis dapat mengintegrasikan teknologi deteksi emosi MoodMe ke dalam pengalaman bermain game.

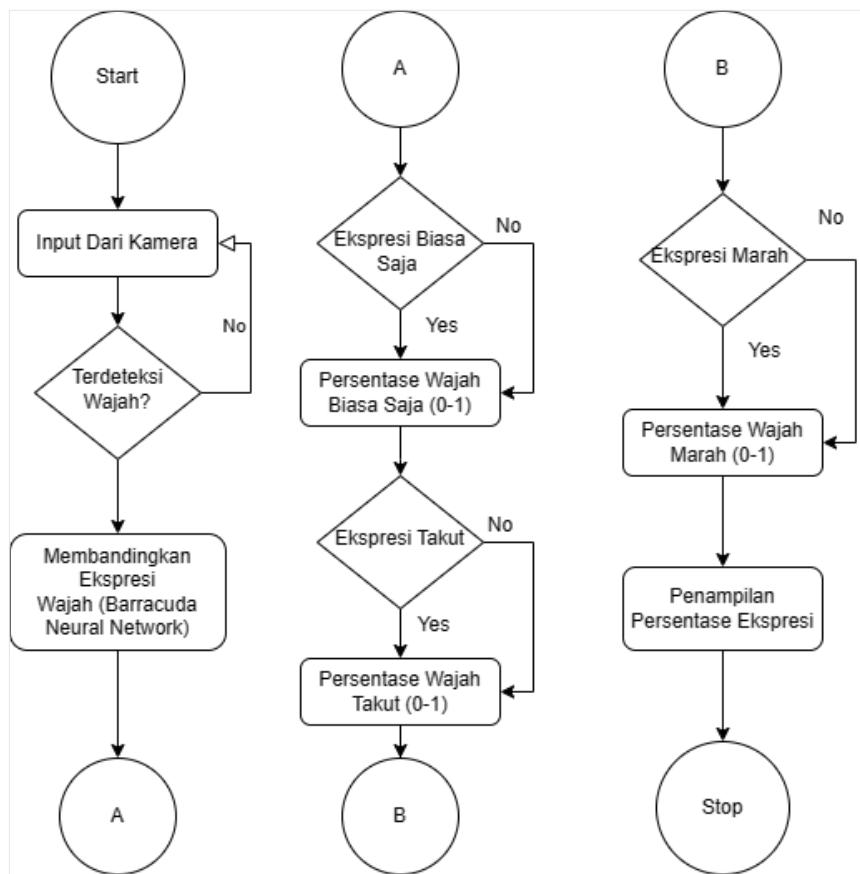
### 3.8 Pendekripsi Emosi MoodMe

Pada game ini emosi yang dideteksi ada 3 yaitu biasa saja atau netral, marah, dan takut. Parameter ekspresi diambil hanya itu saja dikarenakan keterbatasan pada Library MoodMe yang digunakan, yang hanya mendekripsi emosi Marah, Takut, Jijik, dan Netral.



**Gambar 3.6**  
**Ilustrasi Penangkapan Wajah Oleh MoodMe**

MoodMe mengklasifikaskan emosi yang terdeteksi dari data Neural Network Barracuda. Data yang sudah ada pada Neural Network Barracuda diambil oleh MoodMe dan diklasifikasikan berdasarkan label-label yang sudah dibuat oleh tim pengembang MoodMe.

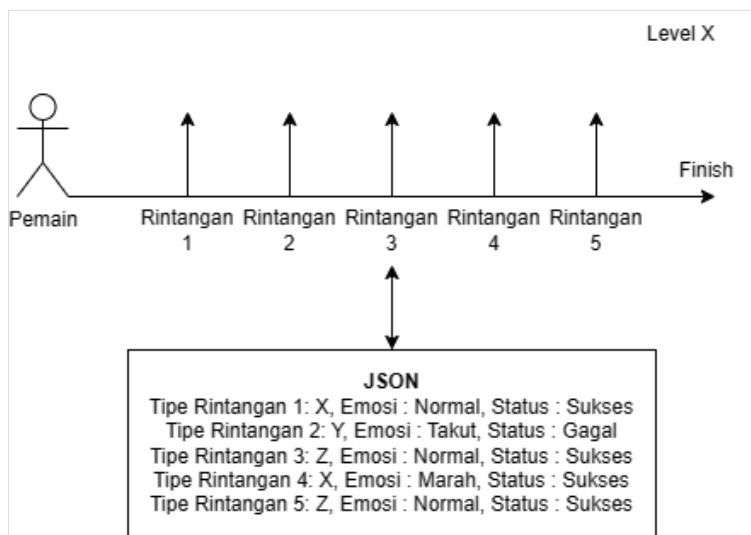


**Gambar 3.7**  
**Alur Kerja Pendekstrian Emosi MoodMe**

MoodMe mengambil tampilan dari standard Webcam Texture Unity class yang mensupport format RGB, RGBA, BGRA, YUV, YUY2. Setelah tampilan didapatkan maka akan dibandingkan dengan label-label yang sudah dibuat oleh MoodME. Hasilnya berupa indicator emosi ber-range 0-100.

### 3.9 Pengaturan Parameter DDA Rintangan Berdasarkan Pendeksi Emosi MoodMe

Cara meningkatkan pengalaman bermain salah satunya melalui penyesuaian tingkat kesulitan berdasarkan data pemain<sup>17</sup>. Didalam game terdapat beberapa variabel dinamis yang disiapkan oleh penulis. Variabel ini disiapkan untuk diubah oleh penulis berdasarkan data emosi yang sudah ditangkap menggunakan MoodMe. Variabel diolah untuk meningkatkan pengalaman bermain pemain game.



**Gambar 3.8**  
**Ilustrasi Pengolahan Parameter Menggunakan MoodME**

Pengolahan parameter dilakukan oleh DDA dengan sistem *dynamic scripting*. Konsep *weight clipping* diterapkan pada sistem *dynamic scripting* dengan menentukan optimalisasi nilai tertinggi dari suatu variable yang dapat dicapai. Nilai maksimum yang tinggi memungkinkan bobot meningkat menjadi nilai yang tinggi, sehingga nilai yang paling efektif hampir selalu dipilih. Demikian pula, nilai rendah untuk nilai maksimum menghambat pertumbuhan bobot. Ini menciptakan variasi besar dalam skrip yang dihasilkan, beberapa diantaranya mungkin ada yang tidak optimal.

<sup>17</sup> Constant, T., & Levieux, G, *Dynamic difficulty adjustment impact on players' confidence*, In Proceedings of the 2019 CHI conference on human factors in computing systems, 2019, hlm 1-12.

Dengan konsep ini game berjalan tidak terlalu mudah bagi pemain yang sudah mahir dan tidak terlalu sulit untuk dimainkan bagi pemain baru. Konsep *weight clipping* juga memudahkan pengembang untuk menentukan tingkat kesulitan dengan memberi pembatasan. Konsep ini diterapkan melalui beberapa parameter yang akan diolah kedalam DDA dan yang akan disesuaikan berdasarkan dari ekspresi wajah yang dideteksi oleh MoodMe. Uraian pengaturan parameter DDA rintangan berdasarkan ekspresi wajah pemain akan dibahas pada subbab di bawah.

### 3.9.1 Parameter DDA Pada Rintangan Hantu

Parameter yang dipakai pada rintangan hantu terdapat 4 variabel parameter yang berpengaruh terhadap jumlah rintangan hantu pada level game. Beberapa variable tersebut adalah akan dijelaskan pada table dibawah ini.

**Tabel 3.1**  
**Parameter DDA Pada Rintangan Hantu**

Parameter	Tipe Data	Keterangan
jumlahJalanHantu	int	Jumlah blok spawn hantu.
ctr_dikejar_hantu	int	Counter pemain dikejar hantu.
emosi	int	Poin emosi yang terdeteksi.
status	bool	Cek kegagalan pemain.

Setiap pemain mengaktifkan trigger gameObject rintangan, maka secara otomatis akan memanggil function pencatat log. Semua variabel yang telah ditentukan dicatat dalam bentuk log berupa JSON. Setelah level selesai, log yang tercatat akan dibaca oleh pembentuk level.

Value parameter yang akan digunakan pada level berikutnya disesuaikan berdasarkan dengan patokan keputusan DDA. Dengan menerapkan konsep Weight Clipping pada DDA spawn hantu, diterapkan value minimum dan maksimum kemunculan blok spawn hantu. Penjelasan keputusan DDA untuk spawn hantu diuraikan dalam table dibawah.

**Tabel 3.2**  
**Tabel Keputusan DDA Rintangan Spawn Hantu**

Parameter				DDA		
JalanHantu	Dikejar	Status	Emosi	Skor	Label	Output
4	2	1,0	1+2/dikejar = 1,5 (Marah)	Normal (1 sukses, 1 gagal)	Normal	Jumlah = Rand (Weight.Clip Normal)

Kemudian level dibuat secara otomatis dengan mengadopsi keputusan dari parameter yang telah diolah DDA. Level dibentuk oleh pembentuk level sesuai dengan suplay data baru yang diolah DDA. Pengolahan dilakukan untuk meningkatkan kualitas pengalaman bermain pemain, sehingga terasa tidak terlalu mudah dan tidak terlalu mudah yang dibuktikan melalui kuesioner.

### 3.9.2 Parameter DDA Pada Rintangan Duri

Parameter yang dipakai pada rintangan duri terdapat 3 variabel parameter yang berpengaruh terhadap jumlah rintangan duri pada level game. Beberapa variabel tersebut adalah akan dijelaskan pada table dibawah ini.

**Tabel 3.3**  
**Parameter DDA Pada Rintangan Duri**

Parameter	Tipe Data	Keterangan
jumlahJalanJebakan	int	Jumlah blok rintangan duri.
emosi	int	Poin emosi yang terdeteksi.
status	int	Cek kegagalan pemain.

Setiap pemain mengaktifkan trigger gameObject rintangan, maka secara otomatis akan memanggil function pencatat log. Semua variabel yang telah ditentukan dicatat dalam bentuk log berupa JSON. Setelah level selesai, log yang tercatat akan dibaca oleh pembentuk level.

Value parameter yang akan digunakan pada level berikutnya disesuaikan berdasarkan dengan patokan keputusan DDA. Dengan menerapkan konsep Weight Clipping pada DDA rintangan duri, diterapkan value minimum dan maksimum kemunculan rintangan duri. Penjelasan keputusan DDA untuk rintangan duri diuraikan dalam table dibawah

**Tabel 3.4**  
**Tabel Keputusan DDA Rintangan Duri**

Parameter			DDA		
JalanJebakan	Status	Emosi	Skor	Label	Output
4	0,0,1,1	2+2+2+2/ jalanJebakan= 2 (Netral)	Normal (2 kena)	Mahir (Normal & Netral)	Jumlah = Rand (Weight.Clip Mahir)

Kemudian level dibuat secara otomatis dengan mengadopsi keputusan dari parameter yang telah diolah DDA. Level dibentuk oleh pembentuk level dengan suplay data baru yang telah diolah DDA. Jumlah rintangan duri yang muncul pada level berikutnya telah disesuaikan dengan keputusan DDA.

### 3.9.3 Parameter DDA Pada Tempat Bersembunyi

Parameter yang dipakai pada tempat bersembunyi terdapat 3 variabel parameter yang berpengaruh terhadap jumlah tempat bersembunyi pada level game. Beberapa variable tersebut adalah akan dijelaskan pada table dibawah ini.

**Tabel 3.5**  
**Parameter DDA Pada Tempat Bersembunyi**

Parameter	Tipe Data	Keterangan
jumlahJalanBambu	int	Jumlah blok bambu.
masuk	int	Hitung jumlah pemain bersembunyi.
emosi	int	Poin yang diberikan kepada emosi yang terdeteksi

Setiap pemain sedang dikejar hantu kemudian masuk dalam area trigger gameObject bambu dan durasi dikejar sudah habis, maka secara otomatis akan memanggil function pencatat log. Semua variabel yang telah ditentukan dicatat dalam bentuk log berupa JSON. Setelah level selesai, log yang tercatat akan dibaca oleh pembentuk level.

Dengan menerapkan konsep Weight Clipping pada DDA tempat bersembunyi, diterapkan value minimum dan maksimum kemunculan tempat bersembunyi. Penjelasan keputusan DDA untuk tempat bersembunyi diuraikan dalam table dibawah

**Tabel 3.6**  
**Tabel Keputusan DDA Tempat Bersembunyi**

Parameter			DDA		
JalanBambu	Masuk	Emosi	Skor	Label	Output
4	1	0	Pemula (1)	Pemula (Pemula & Takut)	Jumlah = Rand (Weight.Clip Pemula)

Kemudian level dibuat secara otomatis dengan mengadopsi keputusan dari parameter yang telah diolah DDA. Level dibentuk oleh pembentuk level dengan suplay data baru yang telah diolah DDA. Jumlah tempat bersembunyi yang muncul pada level berikutnya telah disesuaikan dengan keputusan DDA.

### 3.9.4 Parameter DDA Pada Jumpscare Penampakan

Parameter yang dipakai pada jumpscare penampakan terdapat 3 variable parameter yang berpengaruh terhadap jumpscare penampakan yang muncul pada level game. Beberapa variable tersebut adalah akan dijelaskan pada table dibawah.

**Tabel 3.7**  
**Parameter DDA Pada Jumpscare Penampakan**

Parameter	Tipe Data	Keterangan
jumlahJalanJumpscare	int	Jumlah blok jumpscare.
status	int	Status pemain menginjak kuburan.

**Tabel 3.7  
(Lanjutan)**

<b>Parameter</b>	<b>Tipe Data</b>	<b>Keterangan</b>
emosi	int	Poin yang diberikan kepada emosi yang terdeteksi.

Setiap pemain mengaktifkan trigger gameObject rintangan secara otomatis akan memanggil function pencatat log. Semua variabel yang telah ditentukan dicatat dalam bentuk log berupa JSON. Setelah level selesai, log yang tercatat akan dibaca oleh pembentuk level.

Value parameter yang akan digunakan pada level berikutnya disesuaikan berdasarkan dengan patokan keputusan DDA. Dengan menerapkan konsep Weight Clipping pada DDA jumpscare penampakan, diterapkan value minimum dan maksimum kemunculan jumlah jumpscare penampakan. Penjelasan keputusan DDA untuk jumpscare penampakan diuraikan dalam table dibawah

**Tabel 3.8  
Tabel Keputusan DDA Jumpscare Penampakan**

<b>Parameter</b>			<b>DDA</b>		
<b>JalanJumpscare</b>	<b>Status</b>	<b>Emosi</b>	<b>Skor</b>	<b>Label</b>	<b>Output</b>
6	3	1/status= 0,16 (Takut)	Normal (3)	Pemula (Normal & Takut)	Jumlah = Rand (Weight.Clip Pemula)

Kemudian level dibuat secara otomatis dengan mengadopsi keputusan dari parameter yang telah diolah DDA. Level dibentuk oleh pembentuk level dengan suplay data baru yang telah diolah DDA. Jumlah blok jumpscare penampakan yang muncul pada level berikutnya telah disesuaikan dengan keputusan DDA.

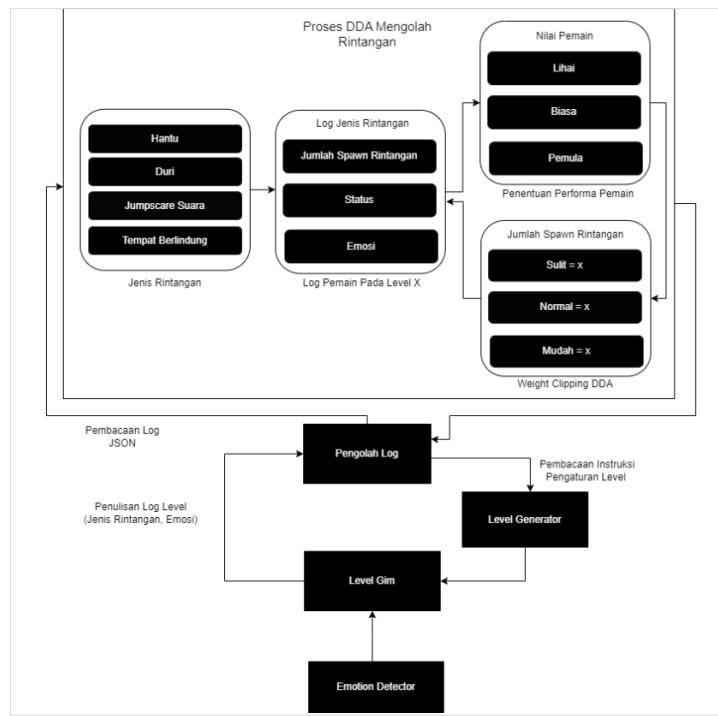
## BAB IV

# DESAIN SISTEM

Bab ini akan dijelaskan tentang tahapan pembuatan desain yang dilakukan selama pembentukan game. Berdasarkan dari konsep atau teori yang telah dibahas sebelumnya. Penulis menerapkannya kedalam game melalui Desain Arsitektural, Desain Interface, dan Desain Procedural berbentuk algoritma pada game yang dibuat.

### 4.1 Desain Arsitektural

Pada desain arsitektural digambarkan dengan beberapa modul-modul. Berikut merupakan gambar desain arsitektural pada game yang dibuat.



**Gambar 4.1**  
**Desain Arsitektural Pengolahan Rintangan Oleh DDA**

Gambar 4.1 merupakan desain arsitektural dari game. Pada saat level dijalankan emotion detector memberikan data emosi yang terdeteksi selama

rintangan dilewati oleh karakter pemain. Setelah level telah berhasil diselesaikan oleh pemain, data yang dicatat oleh pengolah log akan dikirim sebagai input pengolahan rintangan oleh DDA.

Didalam proses pengolahan rintangan, DDA membaca jenis rintangan yang muncul dengan atribut setiap masing-masing jenis. Kemudian skor pemain dibandingkan dengan data weight clipping yang sudah dibentuk. Data weight clipping mengatur intensitas spawn rintangan pada level berikutnya.

Setelah pengolahan data selesai dilakukan, dilakukan penulisan log rintangan yang telah diubah dan kemudian dilakukan pembacaan pada skrip level generator. Sebelum level berikutnya dimainkan, level dibentuk terlebih dahulu dengan suplai data yang telah diolah DDA. Kemudian proses tersebut diulang dari level 2 hingga semua level terselesaikan.

#### 4.1.1 Skenario Game

Skenario pada game yaitu di hutan Gunung Lali Jiwo Arjuno, Jawa Timur yang ditandai pembentukan level. Karakter yang dimainkan harus berjalan melewati rintangan serta menghindari kejaran hantu yang terus mengancam pemain. Bentuk level akan berubah setiap kali dimainkan. Rintangan yang muncul akan diubah sesuai dengan log performa pemain. Perubahan mempengaruhi rate spawn rintangan dan tempat berlindung. Rate diatur berdasarkan nilai *weight clipping* rintangan yang sudah ditetapkan oleh penulis sehingga sedari awal sudah terbentuk konsep rintangan dengan tingkatan mudah, sedang, dan susah.

Karena level yang selalu berubah, start dan finish level pada game ini dilakukan dengan berpatokan checkpoint. Level 1 dimulai setelah karakter menyelesaikan level tutorial. Pada level 1 karakter akan *spawn* di hutan dengan finishnya pemain harus mencapai ujung kanan level. Pada level 2 dan selanjutnya, garis startnya berada di hutan melanjut dengan garis finish yang ditandai dengan poin merah di ujung kanan level.

Pada setiap level terdapat area aman atau pengusir hantu yang ditandai dengan pohon bambu kuning. Terdapat juga spawn item pengusir hantu pada beberapa tempat pada level. Selain itu juga terdapat berbagai jenis rintangan, dan

platform naik turun dimana pemain dituntut untuk lihai dan diharapkan beruntung karena beberapa jenis rintangan melibatkan faktor keberuntungan pemain.

Permainan akan berlangsung hingga waktu habis. Pemain akan mendapat ending baik ketika berhasil melewati 3 level atau lebih hingga waktu yang ditentukan habis. Pemain akan mendapat ending buruk jika sisa nyawa yang diberikan habis dan gagal menyelesaikan 3 level setelah waktu habis.

#### **4.1.2 Arsitektur Pengaturan Rintangan Menggunakan DDA**

Dalam penggunaannya untuk mengatur rintangan, DDA pada game ini terbagi atas 2 tahap yaitu tahap processing dan obstacle generation.

##### **A. TAHAP PROCESSING**

Tahap *processing* melingkupi proses pengolahan data dari seluruh parameter yang sudah ditangkap dan dilakukan pencatatannya menjadi log. Dimulai dari pembacaan masing-masing parameter, melakukan scoring, dan kemudian membandingkannya dengan data weight clipping<sup>17</sup>. Berdasarkan dari parameter yang ditangkap pada setiap masing-masing jenis rintangan, DDA akan melakukan skoring pada performa pemain.

###### **1. SKORING DDA**

Skoring yang dilakukan akan dijabarkan dibawah ini. Skoring dibagi menjadi 2 tahapan. Tahapan pertama yaitu pelabelan performa pemain kemudian dilakukan perhitungan skor emosi pemain rintangan. Tahapan diterapkan pada masing-masing kategori rintangan.

Pada tahapan pelabelan performa pemain, digunakan rumus yang telah dipakai didalam penelitian oleh Missura, O pada 2015 yang berjudul *Dynamic difficulty adjustment*<sup>18</sup>. Rumus disesuaikan dengan beberapa parameter yang diubah pada game sehingga menjadi model sebagai berikut :

---

<sup>17</sup> Hunicke, R. *The case for dynamic difficulty adjustment in games*. In Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology (pp. 429-433), 2005, hlm 14.

<sup>18</sup> Missura, O. *Dynamic difficulty adjustment* (Doctoral dissertation, Universitäts-und Landesbibliothek Bonn), 2015, hlm 30-38.

a. Pemain dengan performa “Mahir”

Rumus dibawah ini digunakan untuk melabeli pemain mahir dengan cara menghindari semua rintangan, rumus yang digunakan yaitu jika rintangan kena sama dengan 0 maka pemain akan diberi label “Mahir”.

Sebagai simulasi pemain A telah menyelesaikan level tanpa kegagalan dengan tidak terkena rintangan yang ada, maka performa yang diberikan oleh sistem DDA kepada pemain yaitu ‘Mahir’. Namun ketika pemain B telah menyelesaikan level dengan satu atau lebih kegagalan pada rintangan yang terdapat pada level, maka akan dilakukan perhitungan performa pemain dengan rumus berikutnya.

Rumus dibawah ini digunakan untuk melabeli pemain mahir dengan cara menghitung persentase kesuksesan dalam melewati rintangan, jika lebih besar dibandingkan dengan jumlah kegagalannya dan memiliki persentase kesuksesan diatas 75 maka akan diberi label “Mahir”.

$$\text{Mahir} = \left( \frac{\text{ctr\_sukses}}{\text{jum\_jalan}} \times 100 \right) > \left( \frac{\text{ctr\_gagal}}{\text{jum\_jalan}} \times 100 \right) \wedge \left( \frac{\text{ctr\_sukses}}{\text{jum\_jalan}} \times 100 \right) > 75 \dots\dots (4.2)$$

Sistem DDA mencatat counter kesuksesan pemain dalam melewati rintangan yang ditampung dalam variable `ctr_sukses` pada masing-masing jenis rintangan, jumlah rintangan kategori tertentu juga akan dicatat yang ditampung pada variable `jumlah_jalan`, dan yang terakhir dilakukan pencatatan counter pemain gagal dalam melewati jenis rintangan tertentu. Untuk mensimulasikan setiap kemungkinan yang mungkin terjadi pada rumus diatas, maka digunakan tabel simulasi dibawah ini.

**Tabel 4.1**  
**Tabel Simulasi Perhitungan Performa Mahir**

No	Ctr_sukses	Ctr_gagal	Jum_jalan	Perhitungan	Keterangan
1	5	0	5	$(1 \times 100) > (0 \times 100) \wedge (1 \times 100) > 75$	Mahir.
2	4	1	5	$(0.8 \times 100) > (0.2 \times 100) \wedge (0.8 \times 100) > 75$	Mahir.
3	3	2	5	$(0.6 \times 100) > (0.4 \times 100) \wedge (0.6 \times 100) > 75$	Tidak memenuhi syarat kedua.
4	2	3	5	$(0.4 \times 100) > (0.6 \times 100) \wedge (0.4 \times 100) > 75$	Tidak memenuhi kedua syarat.
5	1	4	5	$(0.2 \times 100) > (0.8 \times 100) \wedge (0.2 \times 100) > 75$	Tidak memenuhi kedua syarat.
6	0	5	5	$(0 \times 100) > (1 \times 100) \wedge (0 \times 100) > 75$	Tidak memenuhi kedua syarat.

Pada contoh nomer 1 dan 2 didapati bahwa syarat rumus yang pertama yaitu nilai persentase kesuksesan lebih besar daripada nilai kegagalan dan nilai kesuksesan diatas range 75 terpenuhi semua. Maka DDA akan memberi skor pemain “Mahir”. Sedangkan pada contoh nomer 3 hingga nomer 6 memiliki keterangan tidak memenuhi syarat. Pada contoh nomer 3, syarat pertama terpenuhi namun persentase kesuksesan yang didapat berada dibawah 75. Sedangkan pada contoh nomer 4 hingga nomer 6 kedua syarat yang diberikan tidak dapat dipenuhi. Apabila kondisi tidak memenuhi syarat, maka akan dilakukan perhitungan menggunakan rumus berikutnya yaitu label “Normal”.

b. Pemain dengan performa “Normal”

Rumus dibawah ini digunakan untuk melabeli pemain dengan cara menghitung persentase kesuksesan dalam melewati rintangan. Jika nilai persentase kesuksesan yang didapat lebih besar dibandingkan dengan persentase kegagalannya dan memiliki persentase kesuksesan dibawah 75 maka akan diberi label “Normal”.

$$\text{Normal} = \left( \frac{\text{ctr\_sukses}}{\text{jum\_jalan}} \times 100 \right) > \left( \frac{\text{ctr\_gagal}}{\text{jum\_jalan}} \times 100 \right) \wedge \left( \frac{\text{ctr\_sukses}}{\text{jum\_jalan}} \times 100 \right) < 75 \dots (4.3)$$

Sistem DDA mencatat counter kesuksesan pemain dalam melewati rintangan yang ditampung dalam variable ctr\_sukses pada masing-masing jenis rintangan, jumlah rintangan kategori tertentu juga akan dicatat yang ditampung pada variable jum\_jalan, dan yang terakhir dilakukan pencatatan counter pemain gagal dalam melewati jenis rintangan tertentu. Untuk mensimulasikan setiap kemungkinan yang mungkin terjadi pada rumus diatas, maka digunakan tabel simulasi dibawah ini.

**Tabel 4.2**  
**Tabel Simulasi Perhitungan Performa Normal 1**

No	Ctr_sukses	Ctr_gagal	Jum_jalan	Perhitungan	Keterangan
1	5	0	5	$(1 \times 100) > (0 \times 100) \wedge (1 \times 100) < 75$	Tidak memenuhi syarat kedua.
2	4	1	5	$(0.8 \times 100) > (0.2 \times 100) \wedge (0.8 \times 100) < 75$	Tidak memenuhi syarat kedua.
3	3	2	5	$(0.6 \times 100) > (0.4 \times 100) \wedge (0.6 \times 100) < 75$	Normal
4	2	3	5	$(0.4 \times 100) > (0.6 \times 100) \wedge (0.4 \times 100) < 75$	Tidak memenuhi syarat pertama.
5	1	4	5	$(0.2 \times 100) > (0.8 \times 100) \wedge (0.2 \times 100) < 75$	Tidak memenuhi syarat pertama.
6	0	5	5	$(0 \times 100) > (1 \times 100) \wedge (0 \times 100) < 75$	Tidak memenuhi syarat pertama.

Pada contoh nomer 1 dan 2 didapatkan bahwa syarat rumus yang kedua yaitu nilai persentase dibawah range 75 tidak terpenuhi. Pada contoh nomer 3 didapatkan bahwa syarat rumus yang pertama yaitu nilai persentase kesuksesan lebih besar daripada nilai kegagalan dan nilai kesuksesan dibawah range 75 terpenuhi semua sehingga pemain diberi label “Normal”. Pada contoh nomer 4 hingga nomer 6 syarat pertama tidak terpenuhi karena persentase kesuksesan yang didapat lebih kecil daripada persentase kegagalan. Apabila kondisi masuk kedalam tidak memenuhi syarat pertama, maka akan dilakukan perhitungan menggunakan rumus berikutnya.

Rumus dibawah ini digunakan untuk melabeli pemain dengan cara menghitung persentase kesuksesan dalam melewati rintangan, jika sama dengan jumlah persentase kegagalannya, maka akan diberi label “Normal”.

$$\text{Normal} = \begin{cases} 1, & \left( \frac{\text{ctr\_sukses}}{\text{jumlah_jalan}} \times 100 \right) = \left( \frac{\text{ctr\_gagal}}{\text{jumlah_jalan}} \times 100 \right) \\ 0, & \left( \frac{\text{ctr\_sukses}}{\text{jumlah_jalan}} \times 100 \right) \neq \left( \frac{\text{ctr\_gagal}}{\text{jumlah_jalan}} \times 100 \right) \end{cases} \dots\dots\dots (4.4)$$

Sebagai simulasi pemain A telah menyelesaikan level dengan beberapa kegagalan. Langkah berikutnya DDA akan membandingkan persentase jumlah kegagalan pemain apakah sama dengan persentase jumlah keberhasilan pemain. Jika memenuhi syarat tersebut, maka pemain akan diberi label “Normal”. Jika tidak memenuhi syarat akan dilakukan perhitungan dengan rumus berikutnya.

Rumus dibawah ini digunakan untuk melabeli pemain yang memiliki persentase keberhasilan dalam melewati rintangan lebih kecil dibandingkan dengan jumlah kegagalannya tetapi memiliki persentase kesuksesan diatas 35.

$$\text{Normal} = \left( \frac{\text{ctr\_sukses}}{\text{jum\_jalan}} \times 100 \right) < \left( \frac{\text{ctr\_gagal}}{\text{jum\_jalan}} \times 100 \right) \wedge \left( \frac{\text{ctr\_sukses}}{\text{jum\_jalan}} \times 100 \right) > 35 \dots (4.5)$$

Sistem DDA mencatat counter kesuksesan pemain dalam melewati rintangan yang ditampung dalam variable ctr\_sukses pada masing-masing jenis rintangan, jumlah rintangan kategori tertentu juga akan dicatat yang ditampung pada variable jum\_jalan, dan yang terakhir dilakukan pencatatan counter pemain gagal dalam melewati jenis rintangan tertentu. Untuk mensimulasikan setiap kemungkinan yang mungkin terjadi pada rumus diatas, maka digunakan tabel simulasi dibawah ini.

**Tabel 4.3**  
**Tabel Simulasi Perhitungan Performa Normal 2**

No	Ctr_sukses	Ctr_gagal	Jum_jalan	Perhitungan	Keterangan
1	3	2	5	$(0.6 \times 100) < (0.4 \times 100) \wedge (0.6 \times 100) > 35$	Tidak memenuhi syarat pertama.
2	2	3	5	$(0.4 \times 100) < (0.6 \times 100) \wedge (0.4 \times 100) > 35$	Normal
3	1	4	5	$(0.2 \times 100) < (0.8 \times 100) \wedge (0.2 \times 100) > 35$	Tidak memenuhi syarat kedua.
4	0	5	5	$(0 \times 100) < (1 \times 100) \wedge (0 \times 100) > 35$	Tidak memenuhi syarat kedua.

Pada contoh nomer 1 syarat rumus yang pertama yaitu nilai persentase kesuksesan harus lebih rendah daripada persentase kegagalan tidak terpenuhi. Pada contoh nomer 2 didapati bahwa syarat rumus yang pertama yaitu nilai persentase kesuksesan lebih kecil daripada nilai kegagalan dan nilai kesuksesan diatas range 35 terpenuhi semua sehingga pemain diberi label “Normal”. Pada contoh nomer 3 dan nomer 4 syarat kedia tidak terpenuhi, yaitu persentase kesuksesan yang didapat lebih kecil dari range 35. Apabila kondisi masuk kedalam tidak memenuhi syarat kedua, maka akan dilakukan perhitungan rumus untuk menghitung label “Pemula”

c. Pemain dengan performa “Pemula”

Rumus dibawah ini digunakan untuk melabeli ‘Pemula’ pada pemain yang memiliki persentase kesuksesan dibawah 35.

$$\text{Pemula} = \left( \frac{\text{ctr\_sukses}}{\text{jum jalan}} \times 100 \right) < 35 \dots \quad (4.6)$$

Sistem DDA mencatat counter kesuksesan pemain dalam melewati rintangan yang ditampung dalam variable `ctr_sukses` dan jumlah rintangan kategori tertentu yang ditampung pada variable `jum_jalan`. Untuk mensimulasikan setiap kemungkinan yang mungkin terjadi pada rumus diatas, maka digunakan tabel simulasi dibawah.

**Tabel 4.4**  
**Tabel Simulasi Perhitungan Performa Pemula 1**

No	Ctr_sukses	Ctr_gagal	Jum_jalan	Perhitungan	Keterangan
1	3	2	5	$(0.6 \times 100) < 35$	Tidak memenuhi syarat.
2	2	3	5	$(0.4 \times 100) < 35$	Tidak memenuhi syarat.
3	1	4	5	$(0.2 \times 100) < 35$	Pemula
4	0	5	5	$(0 \times 100) < 35$	Pemula

Pada contoh nomer 1 dan nomer 2 syarat rumus yaitu nilai persentase kesuksesan harus dibawah 35 tidak terpenuhi. Pada contoh nomer 3 dan 4 didapatkan bahwa syarat rumus perhitungan persentase kesuksesan lebih kecil dari 35 terpenuhi sehingga pemain diberi label “Pemula”.

Rumus digunakan untuk melabeli pemain yang gagal dalam semua rintangan yang diberikan.

$$\text{Pemula} = \begin{cases} 1, & \text{ctr\_gagal} = \text{jumlahJalan} \\ 0, & \text{ctr\_gagal} \neq \text{jumlahJalan} \end{cases} \dots \quad (4.7)$$

Sebagai simulasi pemain A telah menyelesaikan level dengan sengaja mengenai semua rintangan jenis tertentu yang terdapat pada level. Langkah berikutnya DDA akan membandingkan counter kegagalan pemain apakah sama jumlah jalan rintangan yang muncul pada level. Jika hasilnya sama, maka pemain akan diberi label “Pemula”

Setelah tahapan pelabelan kemudian dilakukan perhitungan emosi rintangan pemain. Berikutnya dilakukan tahapan perhitungan emosi rintangan pemain, emosi diberi label sebagai berikut :

**Tabel 4.5**  
**Tabel Label Nilai Emosi**

No	Label Emosi	Nilai
1	Takut	0
2	Marah	1
3	Netral	2

Jika membagi nilai 2 yang merupakan range tertinggi emosi yaitu “Netral” dengan 3 jenis emosi yang ditangkap, maka akan didapat range 0.66 sebagai batas masing-masing kategori emosi. Ketika pemain mendapatkan nilai dengan range 0~0.66 maka akan digolongkan sebagai emosi “Takut”, berkelanjutan yaitu range 0.67~1.33 maka akan digolongkan sebagai emosi “Marah”, dan yang terakhir yaitu range 1.34~2 maka akan digolongkan sebagai emosi “Netral”.

Setelah menentukan batas pembagian nilai perkategori emosi, maka langkah selanjutnya dilakukan perhitungan nilai emosi pemain. Terdapat 3 rumus yang akan digunakan yang akan dijabarkan dibawah ini.

- a. Rumus perhitungan skor emosi jalan tidak diinjak

Rumus digunakan ketika pemain memiliki nilai rintangan yang tidak diinjak. Nilai emosi rintangan dijumlah dengan jalan tidak diinjak dikali dengan nilai dari label emosi “Netral” kemudian dibagi jumlah jalan yang diinjak untuk mencari rata-rata emosi yang dideteksi.

$$\text{nilai\_emosi} = \frac{\text{nilai\_emosi} + (\text{ctr\_tidak\_injak} \times \text{nilai\_neutra})}{\text{jumlahJalan}} \dots \quad (4.8)$$

Simulasinya sebagai berikut, Pemain A berhasil melewati 4 dari 5 rintangan hantu dengan “Takut” yang dibuktikan dengan rata-rata nilai emosi pada range 0,3 yang merupakan range emosi “Takut” dan pemain memiliki 1 rintangan hantu yang tidak diinjak. Dengan rumus diatas dilakukan perhitungan dengan tahapan  $0,3+(1x2)/5=0,46$ . Hasil yang didapat dari perhitungan termasuk dalam range emosi “Takut” sehingga pemain akan diberi label “Takut” pada rintangan jenis hantu.

- b. Rumus perhitungan skor emosi jalan dihindari semua

Rumus digunakan ketika pemain menghindari semua rintangan. Nilai emosi rintangan diisi dengan jumlahJalan \* 2 karena 2 merupakan skor emosi normal.

Rumus ini akan digunakan saat pemain menghindari semua rintangan tertentu. Dikarenakan berhasil menghindar maka pemain diberikan skor emosi “Netral”.

- c. Rumus perhitungan rata-rata skor emosi

Rumus ini akan digunakan untuk menentukan rata-rata emosi pemain dalam melewati jenis rintangan tertentu dalam 1 level. Nilai ditampung dan dibuat pengecekan dominasi emosi dengan cara menghitung seberapa sering emosi tersebut terjadi.

Jika didapati emosi ‘Netral’ sering keluar, performa akan diubah naik 1 level. Jika didapati emosi ‘Takut’ sering keluar, maka performa akan diturunkan 1 level. Simulasinya sebagai berikut, Pemain A mendapat nilai emosi 12 setelah mengaktifkan 7 rintangan jenis hantu. Maka ketika perhitungan dilakukan dengan rumus diatas, rata-rata emosi yang didapat sebesar 1.7. Karena nilai yang didapat berada pada range 1.34~2, maka pemain akan diberi label emosi “Netral”.

## 2. DATA WEIGHT CLIPPING

Setelah dilakukan skoring, maka proses berikutnya yaitu membentuk data weight clipping pada masing-masing rintangan. Terdapat 4 data weight clipping yang terdapat pada game ini.

- a. Rintangan hantu

Penjelasan pengaturan weight clipping spawn hantu akan dijabarkan melalui table dibawah.

**Tabel 4.6**  
**Tabel Pengaturan Weight Clipping DDA Rintangan Spawn Hantu**

No	Item	Rate Spawn	Label
1	JumlahJalanHantu	2	Mudah
2		4	Sedang
3		5	Susah

Rintangan hantu memiliki pengaturan batas maksimal rate spawn yang diberi label. Jika performa pemain mendapat skor ‘Mahir’ oleh sistem DDA, maka yang dipilih adalah weight clipping yang susah. Jika performa pemain mendapat skor ‘Sedang’ maka yang dipilih adalah weight clipping yang sedang, dan jika pemain mendapat skor ‘Pemula’ maka akan diturunkan kesulitannya menjadi mudah pada level berikutnya.

b. Rintangan duri

Penjelasan pengaturan rintangan duri akan dijabarkan melalui table dibawah.

**Tabel 4.7**  
**Tabel Pengaturan Weight Clipping DDA Rintangan Duri**

No	Item	Rate Spawn	Label
1	jumlahJalanJebakan	3	Mudah
2		4	Sedang
3		6	Susah

Rintangan duri memiliki pengaturan batas maksimal rate spawn yang diberi label. Jika performa pemain mendapat skor ‘Mahir’ oleh sistem DDA, maka yang dipilih adalah weight clipping yang susah. Jika performa pemain mendapat skor ‘Sedang’ maka yang dipilih adalah weight clipping yang sedang, dan jika pemain mendapat skor ‘Pemula’ maka akan diturunkan kesulitannya menjadi mudah pada level berikutnya.

c. Tempat bersembunyi

Penjelasan pengaturan tempat bersembunyi akan dijabarkan melalui table dibawah.

**Tabel 4.8**  
**Tabel Pengaturan Weight Clipping DDA Tempat Bersembunyi**

No	Item	Rate Spawn	Label
1	jumlahJalanBambu	4	Mudah
2		3	Sedang
3		1~2	Susah

Tempat bersembunyi memiliki pengaturan batas maksimal rate spawn yang diberi label. Jika performa pemain mendapat skor ‘Mahir’ oleh sistem DDA, maka yang dipilih adalah weight clipping yang susah. Jika performa pemain mendapat skor ‘Sedang’ maka yang dipilih adalah weight clipping yang sedang, dan jika pemain mendapat skor ‘Pemula’ maka akan diturunkan kesulitannya menjadi mudah.

d. Rintangan jumpscare penampakan

Penjelasan pengaturan rintangan jumpscare penampakan akan dijabarkan melalui table dibawah.

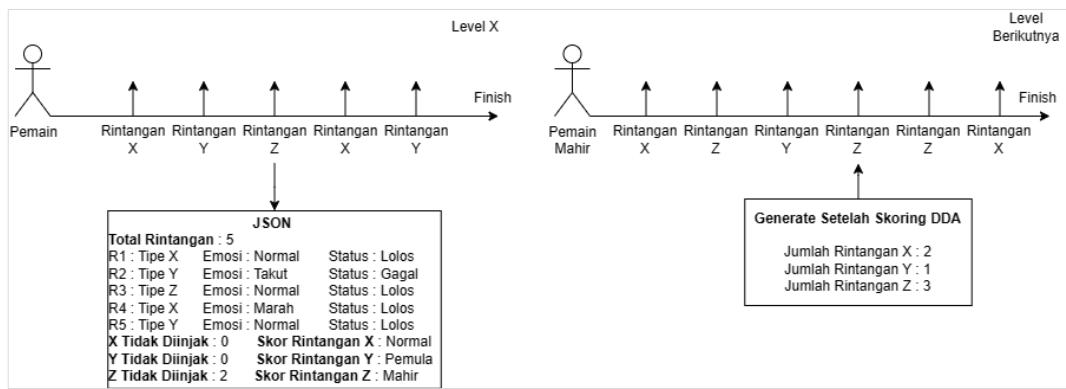
**Tabel 4.9**  
**Tabel Pengaturan Weight Clipping DDA**  
**Rintangan Jumpscare Penampakan**

No	Item	Rate Spawn	Label
1	jumlahJalanJumpscare	2	Mudah
2		4	Sedang
3		8	Susah

Rintangan jumpscare penampakan memiliki pengaturan batas maksimal rate spawn yang diberi label. Jika performa pemain mendapat skor ‘Mahir’ oleh sistem DDA, maka yang dipilih adalah weight clipping yang susah. Jika performa pemain mendapat skor ‘Sedang’ maka yang dipilih adalah weight clipping yang sedang, dan jika pemain mendapat skor ‘Pemula’ maka akan diturunkan kesulitannya menjadi mudah pada level berikutnya.

## B. TAHAP OBSTACLE GENERATION

Setelah DDA melakukan perancangan rintangan yang pada tahap *processing*, langkah selanjutnya dilakukan pembuatan level berikutnya. Penyesuaian Rintangan yang akan dimunculkan pada level berikutnya akan digambarkan melalui ilustrasi di bawah ini.



**Gambar 4.2**  
**Simulasi Pembentukan Rintangan Level**

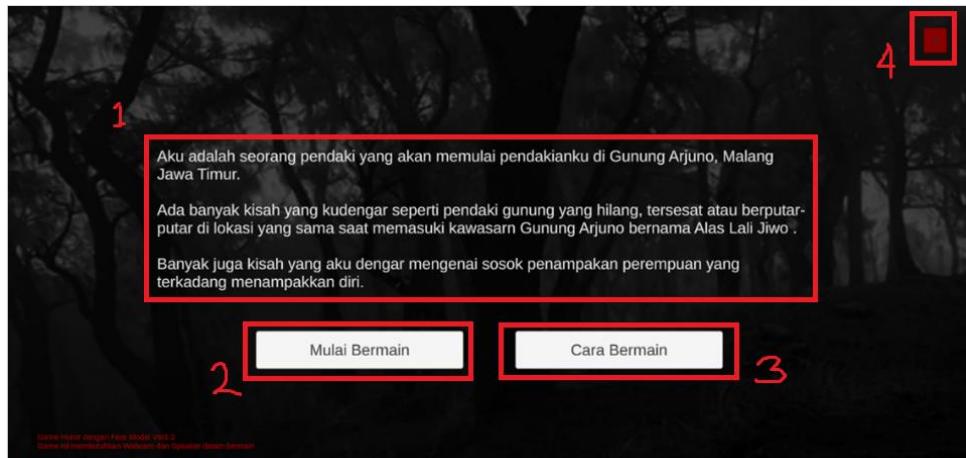
Pada tahap ini content generator akan disuplai data baru yang telah diolah DDA. Rancangan rintangan level berikutnya untuk pemain yang mahir pada rintangan jenis tertentu rintangan tersebut akan dimunculkan semakin banyak. Content generator akan membentuk jalan dengan rintangan berdasarkan instruksi yang telah didapat sehingga menjadi sebuah level. Siklus ini selalu diulang sebanyak level terdapat didalam game. Jadi setiap level yang akan muncul akan selalu berbeda dikarenakan kemunculan rintangan yang dilakukan berdasarkan data performa pemain pada tahapan sebelumnya.

## 4.2 Desain Interface

Subbab ini menjelaskan desain interface dari tampilan menu utama, narasi penutup, cara bermain, tampilan hud, dan deteksi wajah.

#### 4.2.1 Menu Utama

Halaman menu utama akan ditampilkan setiap permainan dibuka, dan setelah pemain telah menekan tombol Main Menu pada narasi penutup. Interface ini digunakan untuk membantu pemain dalam menjelajahi sistem permainan.



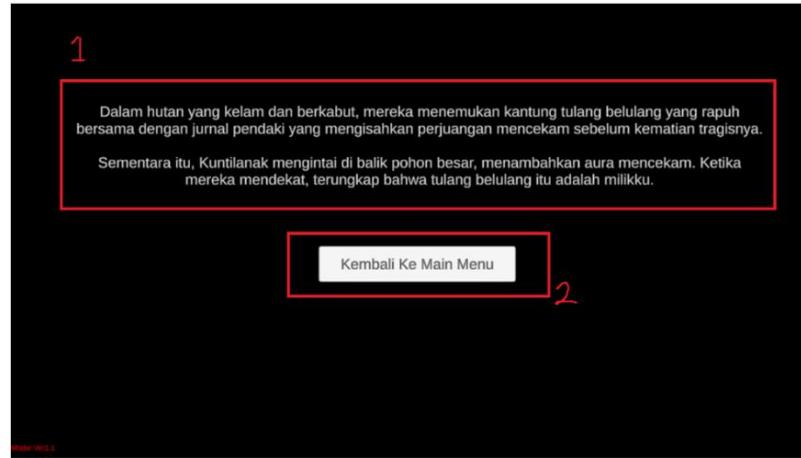
**Gambar 4.3**  
**Tampilan Menu Utama**

Di halaman menu utama terdapat beberapa komponen penting pada halaman ini yaitu narasi pembuka, tombol mulai bermain, tombol cara bermain, dan tombol exit. Penjelasan lebih lengkapnya diuraikan sebagai berikut:

1. Narasi pembuka seputar cerita karakter didalam game yang tersesat.
2. Tombol mulai bermain, berfungsi untuk memulai permainan.
3. Tombol cara bermain, berfungsi untuk menjelaskan kepada pemain mengenai kontrol pemain dan bagaimana game harus dimainkan.
4. Tombol exit, pada ujung kanan layar berfungsi untuk menutup permainan.

#### 4.2.2 Narasi Penutup

Halaman narasi penutup akan ditampilkan setiap pemain berhasil menyelesaikan semua level. Interface ini membantu pemain untuk memahami cerita dari keseluruhan game.



## Gambar 4.4

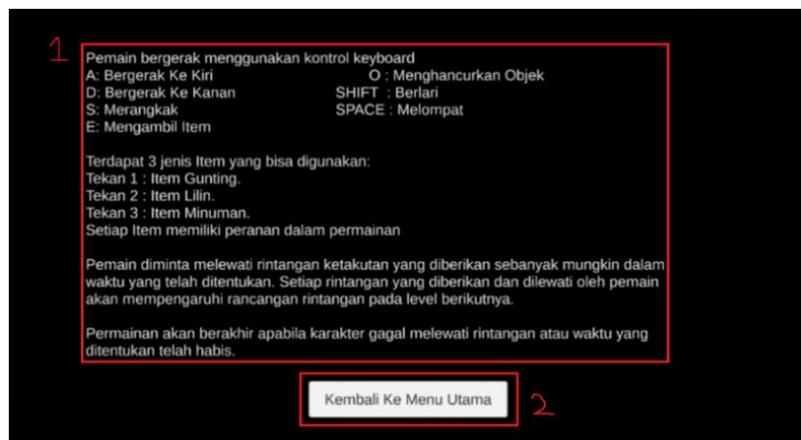
### Tampilan Narasi Penutup

Di halaman narasi penutup terdapat sebuah tombol yaitu tombol main menu, dan teks narasi tentang penutup cerita game. Penjelasan lebih lengkapnya diuraikan sebagai berikut :

1. Narasi penutup seputar cerita kondisi karakter setelah permainan dimainkan.
  2. Tombol main menu, berfungsi untuk membawa pemain ke halaman utama.

### 4.2.3 Cara Bermain

Halaman cara bermain akan muncul setiap pemain memainkan level. Fungsi interface ini agar pemain dapat mempelajari kontrol game, memahami tujuan game, dan memahami mekanik dasar game yang akan dimainkan.



## Gambar 4.5 Tampilan Cara Bermain

Pada gambar terdapat penanda dengan nomor yang akan membantu untuk memberi penjelasan tentang hal-hal yang terdapat pada tampilan cara bermain. Penjelasan lengkapnya diuraikan sebagai berikut :

1. Merupakan penjelasan cara bermain dan pengetahuan singkat tentang game.
2. Tombol untuk kembali menuju menu utama.

#### 4.2.4 Interface HUD Karakter

Interface hud player akan muncul setiap pemain memainkan level. Fungsi interface ini agar pemain dapat memonitor status karakter yang dimainkannya. Status yang ditampilkan mencakup darah, waktu, dan benda yang telah diambil.



**Gambar 4.6**  
**Tampilan HUD Karakter**

Pada gambar terdapat penanda dengan nomor yang akan membantu untuk memberi penjelasan tentang hal-hal yang terdapat pada hud karakter. Penjelasan lengkapnya diuraikan sebagai berikut :

1. Merupakan tampilan poin darah karakter yang dimainkan, apabila darah habis maka karakter akan mati.
2. Merupakan tampilan benda yang diambil pemain, jika hendak memakai benda yang diambil maka pemain harus menekan tombol “E” pada keyboard.
3. Merupakan timer sesi permainan, jika habis maka akan dimainkan scene ending yang berbeda-beda berdasarkan dengan nilai performa pemain.

#### 4.2.5 Interface Pendektsian Wajah

Interface pendektsian wajah akan muncul selama sesi permainan dijalankan. Fungsi interface ini agar membantu penulis memonitor ekspresi pemain selama sesi permainan dijalankan.



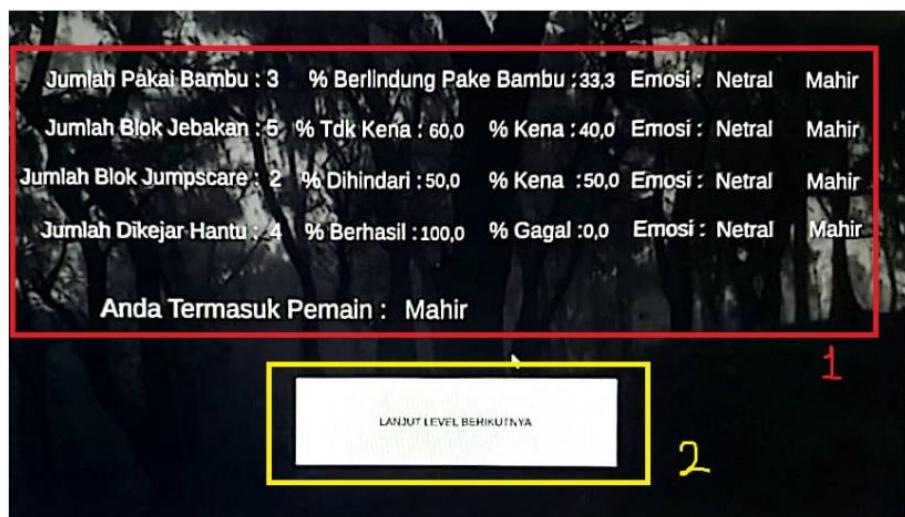
**Gambar 4.7**  
**Tampilan Pendektsi Wajah**

Pada gambar terdapat penanda dengan nomor yang akan membantu untuk memberi penjelasan tentang hal-hal yang terdapat pada interface pendektsian wajah. Penjelasan lengkapnya diuraikan sebagai berikut :

1. Merupakan tampilan penangkapan wajah pemain, ini dapat ditampilkan atau disembunyikan.
2. Merupakan bar emosi yang terdeteksi.

#### 4.2.6 Interface Report Performa Pemain

Interface report akan muncul setiap pemain telah berhasil melewati level. Fungsi interface ini agar pemain mengetahui skor yang didapatkannya berdasarkan penilaian DDA.



**Gambar 4.8**  
**Tampilan Report Performa Pemain**

Pada gambar terdapat penanda dengan nomor yang akan membantu untuk memberi penjelasan tentang hal-hal yang terdapat pada interface report performa. Penjelasan lengkapnya diuraikan sebagai berikut :

1. Merupakan tampilan poin hasil penilaian pemain yang diberikan oleh DDA.
2. Merupakan tombol untuk memulai level berikutnya.

### 4.3 Desain Procedural

Sub bab ini menjelaskan tentang algoritma-algoritma modul inti yang digunakan didalam game. Penjelasan berupa algoritma bagaimana modul harus berjalan sesuai dengan tujuan yang ditentukan penulis sebelum dilanjutkan menjadi penulisan kode. Algoritma yang akan dibahas yaitu algoritma karakter, algoritma rintangan, dan algoritma DDA.

Algoritma karakter akan membahas algoritma dan segmen program dari kontrol karakter, physics karakter, dan animasi karakter. Algoritma rintangan akan membahas algoritma rintangan dan segmen program yang digunakan untuk mengatur segala jenis yang berkaitan dengan rintangan yang berupa spawn blok hantu, duri, jumpscare penampakan, tempat bersembunyi, item dan kabut. Algoritma DDA akan membahas algoritma dan segmen program dari sistem pencatatan log, penyesuaian rintangan, dan pembentukan rintangan level.

### 4.3.1 Algoritma Kontrol Karakter

Pada bagian ini akan dijelaskan langkah-langkah pemain menggerakan karakter ke kanan atau ke kiri, berjalan, berlari. Langkah-langkah dijelaskan dalam baris yang mewakili 1 logic. Penjelasan detailnya dijelaskan dalam algoritma 4.1.

#### **Algoritma 4.1 Kontrol Karakter**

- 01: Cek apakah pemain menekan tombol gerak kekiri (a) atau tombol gerak kekanan (d).
- 02: Cek apakah pemain tidak kena serang, tidak melompat, dan tidak merangkak.
- 03: Jika tombol gerak ke kiri ditekan maka flip badan karakter ke kiri, kemudian kurangi koordinat x selama tombol ditekan sesuai dengan speed yang telah ditetapkan.
- 04: Jika tombol gerak ke kanan ditekan maka flip badan karakter ke kanan, kemudian tambah koordinat x selama tombol ditekan sesuai dengan speed yang telah ditetapkan.
- 05: Cek apakah tombol sprint/lari (shift) ditekan, jika iya ubah speed karakter.
- 06: Ketika move true dan sprint true, munculkan animasi berlari, jika move true dan sprint false, munculkan animasi jalan.
- 07: Cek koordinat x dari karakter setiap karakter bergerak, jika mengalami collision dengan objek lain maka ubah speed karakter jadi 0.

Ketika pemain menekan tombol "a" atau "d", karakter akan bergerak ke kiri atau ke kanan sesuai dengan tombol yang ditekan. Jika pemain menekan tombol untuk melompat, karakter akan melompat ke atas. Jika pemain menggunakan tombol sprint, kecepatan karakter akan meningkat. Animasi berlari akan muncul jika karakter bergerak dan menggunakan sprint, sementara animasi berjalan akan muncul jika karakter bergerak tanpa menggunakan sprint. Setiap kali karakter bergerak, koordinatnya akan diperiksa untuk tabrakan dengan objek lain di lingkungan game. Jika terjadi tabrakan dengan objek, kecepatan karakter akan diubah menjadi 0, menghentikan pergerakannya.

### 4.3.2 Algoritma Kontrol Merangkak

Pada bagian ini akan dijelaskan langkah-langkah pemain dalam mode merangkak. Langkah-langkah dijelaskan dalam baris yang mewakili 1 logic. Penjelasan detailnya dijelaskan dalam algoritma 4.2.

### **Algoritma 4.2 Kontrol Merangkak**

- 01: Cek apakah pemain menekan tombol (s).
- 02: Cek apakah karakter sedang berada ditanah atau tidak.
- 03: Jika tombol s ditekan maka set mode menjadi merangkak.
- 04: Kemudian jalankan animasi merangkak.
- 05: Ubah collider karakter menjadi collider karakter mode merangkak.

Ketika pemain menekan tombol "s", sistem akan memeriksa apakah karakter berada di tanah. Jika iya, karakter akan beralih ke mode merangkak, dan animasi merangkak akan dijalankan. Selain itu, collider karakter akan diubah sesuai dengan mode merangkak untuk interaksi dengan objek di sekitarnya.

### **4.3.3 Algoritma Bangkit Dari Mode Merangkak**

Pada bagian ini akan dijelaskan langkah-langkah karakter dalam kembali berdiri dari posisi merangkak. Langkah-langkah dijelaskan dalam baris yang mewakili 1 logic. Penjelasan detailnya dijelaskan dalam algoritma 4.3.

### **Algoritma 4.3 Kontrol Bangkit Dari Mode Merangkak**

- 01: Cek apakah pemain menekan tombol (s).
- 02: Cek apakah karakter sedang dalam mode merangkak.
- 03: Cek apakah karakter sedang menginjak tanah.
- 04: Cek apakah diatas karakter ada collider lain.
- 05: Jalankan animasi karakter bangkit dari merangkak.
- 06: Ubah collider karakter menjadi collider karakter mode berdiri.

Ketika pemain menekan tombol "s", sistem akan memeriksa apakah karakter sedang dalam mode merangkak. Jika iya, sistem akan memeriksa apakah karakter sedang menginjak tanah. Setelah itu, sistem akan memeriksa apakah di atas karakter ada collider lain. Jika semua kondisi terpenuhi, sistem akan menjalankan animasi karakter bangkit dari posisi merangkak dan mengubah collider karakter menjadi collider karakter dalam mode berdiri.

### **4.3.4 Algoritma Kontrol Melompat**

Pada bagian ini akan dijelaskan langkah-langkah karakter dalam melakukan tindakan melompat. Langkah-langkah dijelaskan dalam baris yang mewakili 1 logic. Penjelasan detailnya dijelaskan dalam algoritma 4.4.

#### **Algoritma 4.4 Kontrol Melompat**

- 01: Cek apakah pemain menekan tombol (space) .
- 02: Cek apakah karakter sedang berada ditanah atau tidak.
- 03: Jalankan animasi melompat
- 04: Kurangi koordinat Y karakter berdasarkan yang angka yang sudah diset.

Ketika pemain menekan tombol "space", sistem akan memeriksa apakah karakter berada di tanah. Jika iya, sistem akan menjalankan animasi melompat. Setelah itu, sistem akan mengurangi koordinat Y karakter berdasarkan angka yang telah diset, sehingga karakter tampak melompat ke atas.

#### **4.3.5 Algoritma Karakter Mengambil Item**

Pada bagian ini akan dijelaskan langkah-langkah karakter dalam mengambil item. Langkah-langkah dijelaskan dalam baris yang mewakili 1 logic. Penjelasan detailnya dijelaskan dalam algoritma 4.5.

#### **Algoritma 4.5 Karakter Mengambil Item**

- 01: Cek apakah karakter bertabrakan dengan trigger dari item.
- 02: Cek apakah pemain menekan tombol (e) .
- 03: Cek jenis item yang diambil.
- 04: Tambah jumlah item yang dibawa karakter.
- 05: Hilangkan item berserta triggernya.

Ketika karakter bertabrakan dengan trigger dari suatu item, sistem akan memeriksa apakah pemain menekan tombol "e". Jika iya, sistem akan memeriksa jenis item yang diambil. Selanjutnya, sistem akan menambah jumlah item yang dibawa karakter sesuai dengan jenis item yang diambil, dan menghilangkan item beserta triggernya dari level permainan.

#### **4.3.6 Algoritma Karakter Bersembunyi**

Pada bagian ini akan dijelaskan mengenai karakter dalam bersembunyi. Langkah-langkah dijelaskan dalam baris yang mewakili 1 logic. Penjelasan detailnya dijelaskan dalam algoritma 4.6.

#### **Algoritma 4.6 Karakter Bersembunyi**

- 01: Cek apakah karakter bertabrakan dengan trigger dari item.
- 02: Cek apakah karakter didalam area collider tempat sembunyi.
- 03: Cek apakah karakter dikejar hantu.
- 04: Status karakter diubah menjadi bersembunyi.

Ketika karakter bertabrakan dengan trigger dari suatu item, sistem akan memeriksa apakah karakter berada di dalam area collider tempat sembunyi. Jika iya, sistem akan memeriksa apakah karakter sedang dikejar oleh hantu. Jika kondisi tersebut terpenuhi, status karakter akan diubah menjadi "bersembunyi".

#### **4.3.7 Algoritma Karakter Dikejar Hantu**

Pada bagian ini akan dijelaskan mengenai karakter dikejar oleh hantu. Langkah-langkah dijelaskan dalam baris yang mewakili 1 logic. Penjelasan detailnya dijelaskan dalam algoritma 4.7.

#### **Algoritma 4.7 Karakter Dikejar Hantu**

- 01: Cek apakah karakter menginjak trigger spawn hantu.
- 02: Random mode hantu yang mengejar karakter.
- 03: Mainkan suara efek horror
- 04: Mainkan efek suara dari jenis hantu yang mengejar.
- 05: Status karakter diubah menjadi dikejar hantu.
- 06: Cek mode hantu yang mengejar.
- 07: Munculkan animasi hantu yang mengejar karakter.

Ketika karakter menginjak trigger spawn hantu, sistem akan memeriksa dan memilih secara acak mode hantu yang akan mengejar karakter. Setelah itu, sistem akan memainkan suara efek horror dan efek suara yang sesuai dengan jenis hantu yang mengejar. Selain itu, status karakter akan diubah menjadi "dikejar hantu", dan sistem akan memeriksa mode hantu yang mengejar untuk menampilkan animasi hantu yang sesuai.

#### **4.3.8 Algoritma Karakter Terkena Serangan Hantu**

Pada bagian ini akan dijelaskan mengenai karakter terkena serang oleh hantu. Langkah-langkah dijelaskan dalam baris yang mewakili 1 logic. Penjelasan detailnya dijelaskan dalam algoritma 4.8.

#### **Algoritma 4.8 Karakter Terkena Serangan Hantu**

- 01: Cek apakah karakter statusnya dikejar hantu.
- 02: Cek apakah pemain tidak menggunakan item dalam 3 detik.
- 03: Cek apakah karakter statusnya berubah menjadi bersembunyi dalam waktu 3 detik.
- 04: Cek jumlah nyawa karakter.
- 05: Mainkan suara hantu menangis.
- 06: Mainkan animasi hantu menyerang karakter.
- 07: Kurangi nyawa karakter.
- 08: Cek apakah karakter mati
- 09: Mainkan animasi karakter terserang.
- 10: Mainkan animasi karakter mati.
- 11: Status karakter diubah kembali menjadi tidak dikejar hantu.
- 12: Hilangkan animasi hantu mengejar.

Saat karakter sedang dikejar hantu, sistem akan memeriksa status karakter untuk mengetahui apakah pemain telah menggunakan item dalam 3 detik. Jika tidak, sistem akan memeriksa apakah status karakter telah berubah menjadi "bersembunyi" dalam waktu 3 detik. Selanjutnya, sistem akan memeriksa jumlah nyawa karakter dan memainkan suara hantu menangis serta animasi hantu menyerang karakter. Setelah itu, sistem akan mengurangi nyawa karakter dan memeriksa apakah karakter telah mati. Jika iya, sistem akan memainkan animasi karakter terserang dan animasi karakter mati, serta mengubah status karakter kembali menjadi tidak dikejar hantu. Terakhir, sistem akan menghilangkan animasi hantu mengejar.

#### **4.3.9 Algoritma Rintangan Hantu**

Pada bagian ini akan dijelaskan mengenai mekanisme rintangan hantu yang digenerate oleh content generator. Langkah-langkah dijelaskan dalam baris yang mewakili 1 logic. Penjelasan detailnya dijelaskan dalam algoritma 4.9.

#### **Algoritma 4.9 Rintangan Hantu**

- 01: Cek apakah gameobject bertag "Player" menginjak collider blok jalan hantu
- 02: Cek apakah status blok sudah dipijak.
- 03: Cek apakah pemain tidak menggunakan item gunting.
- 04: Munculkan hantu.
- 05: Ubah status dipijak menjadi true.

Saat gameobject dengan tag "Player" menginjak collider blok jalan hantu, sistem akan memeriksa apakah status blok tersebut sudah dipijak. Jika blok belum dipijak dan pemain tidak menggunakan item gunting, maka sistem akan mulai memunculkan hantu. Setelah itu, sistem akan mengubah status dipijak menjadi true untuk blok tersebut.

#### **4.3.10 Algoritma Jumpscare Penampakan**

Pada bagian ini akan dijelaskan mengenai mekanisme lintangan jumpscare penampakan yang digenerate oleh content generator. Langkah-langkah dijelaskan dalam baris yang mewakili 1 logic. Penjelasan detailnya dijelaskan dalam algoritma 4.10.

##### **Algoritma 4.10 Rintangan Jumpscare Penampakan**

- 01: Cek apakah gameobject bertag "Player" menginjak collider blok jumpscare.
- 02: Cek apakah status blok sudah dipijak.
- 03: Memanggil canvas jumpscare.
- 04: Memainkan suara jumpscare.
- 05: Mematikan collider blok.
- 06: Menghitung mundur canvas hingga hilang.
- 07: Menutup canvas jumpscare.

Ketika gameobject dengan tag "Player" menginjak collider blok jumpscare, sistem akan memeriksa apakah status blok tersebut sudah dipijak. Jika blok tersebut belum dipijak, maka sistem akan melakukan serangkaian tindakan. Pertama-tama, sistem akan memanggil canvas jumpscare untuk menampilkan efek jumpscare kepada pemain. Selanjutnya, sistem akan memainkan suara jumpscare untuk meningkatkan efek ketegangan. Setelah itu, sistem akan mematikan collider blok agar tidak ada lagi interaksi dengan pemain. Kemudian, sistem akan menghitung mundur pada canvas jumpscare hingga efeknya hilang. Setelah efek jumpscare hilang, sistem akan menutup canvas jumpscare untuk melanjutkan permainan.

### 4.3.11 Algoritma Item

Pada bagian ini akan dijelaskan mengenai mekanisme item yang digenerate oleh content generator. Langkah-langkah dijelaskan dalam baris yang mewakili 1 logic. Penjelasan detailnya dijelaskan dalam algoritma 4.11.

#### **Algoritma 4.11 Algoritma Item**

- 01: Mengecek pemain apakah memiliki item yang disimpan dalam tas.
- 02: Mengecek pemain menekan tombol item apa.
- 03: Mengurangi jumlah item yang dipakai.
- 04: Memunculkan efek sprite item telah terpakai.
- 05: Melakukan hitung mundur sebagai delay penggunaan item.

Sistem akan memeriksa apakah pemain memiliki item yang disimpan dalam tas. Jika iya, sistem akan memeriksa tombol item apa yang ditekan oleh pemain. Setelah itu, sistem akan mengurangi jumlah item yang dipakai dan memunculkan efek sprite untuk menunjukkan bahwa item tersebut telah digunakan. Selain itu, sistem akan memulai hitung mundur sebagai delay sebelum pemain dapat menggunakan item kembali.

### 4.3.12 Algoritma Tempat Bersembunyi

Pada bagian ini akan dijelaskan mengenai mengenai mekanisme tempat bersembunyi yang digenerate oleh content generator. Langkah-langkah dijelaskan dalam baris yang mewakili 1 logic. Penjelasan detailnya dijelaskan dalam algoritma 4.12.

#### **Algoritma 4.12 Algoritma Tempat Bersembunyi**

- 01: Cek apakah gameobject bertag "Player" menginjak collider blok bambu.
- 02: Memberi status masuk = true.
- 03: Cek apakah status masuk = true dan pemain tetap berada didalam.
- 04: Cek tipe bambu yang digunakan untuk bersembunyi.
- 05: Jika bambu pendek dan status karakter pemain jongkok = true
- 06: Memberi status bersembunyi = true.
- 07: Cek apakah karakter keluar dari trigger bambu.
- 08: Memberi status masuk = false.
- 09: Memberi status bersembunyi = false.

Saat gameobject dengan tag "Player" menginjak collider blok bambu, sistem akan memberi status masuk menjadi true. Selanjutnya, sistem akan memeriksa apakah status masuk sudah bernilai true dan apakah pemain masih berada di dalam collider bambu. Setelah itu, sistem akan memeriksa tipe bambu yang digunakan untuk bersembunyi. Jika bambu tersebut adalah tipe pendek dan status karakter pemain adalah jongkok (jongkok = true), maka sistem akan memberi status bersembunyi menjadi true. Kemudian, sistem akan memeriksa apakah karakter keluar dari trigger bambu. Jika karakter keluar dari trigger, sistem akan mengubah status masuk menjadi false dan status bersembunyi menjadi false.

#### **4.3.13 Algoritma Rintangan Duri**

Pada bagian ini akan dijelaskan mengenai mekanisme rintangan duri yang digenerate oleh content generator. Langkah-langkah dijelaskan dalam baris yang mewakili 1 logic. Penjelasan detailnya dijelaskan dalam algoritma 4.13.

##### **Algoritma 4.13 Rintangan Duri**

- 01: Cek apakah gameobject bertag "Area Pukul" menginjak collider rintangan duri.
- 02: Mainkan suara menghancurkan objek.
- 03: Hapus objek rintangan duri.
- 04: Cek apakah gameobject bertag "Player" menginjak collider rintangan duri.
- 05: Memberi status karakter kena serangan.
- 06: Memberi status game object active = false.

Saat gameobject dengan tag "Area Pukul" menginjak collider rintangan duri, sistem akan memainkan suara menghancurkan objek dan menghapus objek rintangan duri tersebut. Selanjutnya, sistem akan memeriksa apakah gameobject dengan tag "Player" menginjak collider rintangan duri. Jika iya, sistem akan memberi status karakter kena serangan dan menonaktifkan game object dengan status active = false.

### 4.3.14 Algoritma Pencatatan Log DDA

Pada bagian ini akan dijelaskan mengenai langkah-langkah pencatatan log. Langkah-langkah dijelaskan dalam baris yang mewakili 1 logic. Penjelasan detailnya dijelaskan dalam algoritma 4.14.

#### **Algoritma 4.14 Pencatatan Log DDA**

- 01: Cek apakah karakter menabrak trigger rintangan.
- 02: Cek jenis rintangan yang ditabrak collider karakter.
- 03: Cek emosi yang terdeteksi saat menginjak blok rintangan.
- 04: Cek status rintangan yang ditabrak.
- 05: Mencatat dalam array data yang telah ditangkap.
- 06: Cek jika level telah selesai dan checkpoint diaktifkan.
- 07: Melakukan generate json berdasarkan data yang ditampung dalam array.
- 08: Menyimpan log pada direktori yang dituju.

Saat karakter menabrak trigger rintangan, sistem akan memeriksa jenis rintangan yang ditabrak oleh collider karakter. Setelah itu, sistem akan memeriksa emosi yang terdeteksi saat karakter menginjak blok rintangan dan status rintangan yang ditabrak. Kemudian, sistem akan mencatat informasi tersebut dalam sebuah array data yang telah ditangkap. Selanjutnya, sistem akan memeriksa apakah level telah selesai dan checkpoint diaktifkan. Jika level telah selesai dan checkpoint diaktifkan, sistem akan melakukan generate JSON berdasarkan data yang telah ditampung dalam array. Terakhir, sistem akan menyimpan log pada direktori yang dituju.

### 4.3.15 Algoritma Penyesuaian Rintangan DDA

Pada bagian ini akan dijelaskan langkah-langkah rintangan disesuaikan oleh DDA. Langkah-langkah dijelaskan dalam baris yang mewakili 1 logic. Penjelasan detailnya dijelaskan dalam algoritma 4.15.

#### **Algoritma 4.15 Penyesuaian Rintangan DDA**

- 01: Membuka file Log.
- 02: Cek berapa banyak blok rintangan.
- 03: Cek jenis blok rintangan yang dicatat.
- 04: Cek emosi yang terdeteksi saat menginjak blok rintangan.
- 05: Cek status karakter setelah melewati rintangan.
- 06: Cek berapa kali pemain berhasil melewati rintangan.
- 07: Cek berapa kali pemain gagal melewati rintangan.
- 08: Cek berapa banyak blok rintangan yang tidak diinjak.

### **Algoritma 4.15 (Lanjutan)**

- 09: Memberi skor performa pemain.
- 10: Memberi skor emosi pemain.
- 11: Membandingkan dengan data *weight clipping* DDA yang dibuat.
- 12: Mengubah banyak blok rintangan pada level berikutnya.
- 13: Membersihkan file log DDA ketika level berikutnya telah dimuat.

Saat membuka file Log, sistem akan melakukan serangkaian pemeriksaan terkait data yang tercatat dalam log tersebut. Sistem akan memeriksa jumlah blok rintangan yang tercatat dalam file Log dan jenis blok rintangan yang dicatat. Saat karakter menginjak blok rintangan, sistem akan memeriksa emosi yang terdeteksi pada saat itu dan status karakter setelah melewati rintangan. Sistem akan mencatat berapa kali pemain berhasil dan gagal melewati rintangan, serta berapa banyak blok rintangan yang tidak diinjak. Berdasarkan data yang tercatat, sistem akan memberikan skor performa dan skor emosi pemain. Selanjutnya, sistem akan membandingkan data dengan pola atau aturan yang telah ditetapkan, seperti *weight clipping* Dynamic Difficulty Adjustment (DDA), dan mengubah banyaknya blok rintangan pada level berikutnya. Ketika level berikutnya dimuat, sistem akan membersihkan file log DDA yang terkait.

### **4.3.16 Algoritma Pembentukan Rintangan Level**

Pada bagian ini akan dijelaskan langkah-langkah rintangan pada level berikutnya dibentuk dengan data dari log DDA. Langkah-langkah dijelaskan dalam baris yang mewakili 1 logic. Penjelasan detailnya dijelaskan dalam algoritma 4.16.

### **Algoritma 4.16 Pembentukan Rintangan Level**

- 01: Membaca hasil pengolahan *weight ceiling* DDA.
- 02: Membuat checkpoint pada awal level
- 03: Membuat gameobject air dengan panjang dan tinggi yang ditentukan.
- 04: Menghitung persentase blok rintangan muncul terhadap panjang platform.
- 05: Membuat gameobject blok rintangan dengan posisi random sebanyak jumlah yang ditentukan DDA.
- 06: Membuat jalan biasa dengan posisi random sebanyak jumlah sisa dari perhitungan persentase.
- 07: Membuat gameobject blok naik-turun pada ujung kiri setiap jalan yang tidak memiliki jalan biasa.
- 08: Membuat checkpoint pada akhir level.

Sistem akan membaca hasil pengolahan weight ceiling DDA untuk menyesuaikan tingkat kesulitan permainan. Kemudian, sistem akan membuat checkpoint pada awal level untuk memudahkan pemain dalam melanjutkan permainan. Selanjutnya, sistem akan membuat gameobject air dengan panjang dan tinggi yang telah ditentukan. Selanjutnya, sistem akan menghitung persentase blok rintangan yang muncul terhadap panjang platform. Berdasarkan perhitungan tersebut, sistem akan membuat gameobject blok rintangan dengan posisi acak sesuai dengan jumlah yang ditentukan oleh DDA. Selain itu, sistem akan membuat jalan biasa dengan posisi acak sesuai dengan jumlah sisa dari perhitungan persentase. Jika sebuah jalan tidak memiliki jalan biasa, sistem akan membuat gameobject blok naik-turun pada ujung kiri setiap jalan tersebut. Terakhir, sistem akan membuat checkpoint pada akhir level untuk menandai pencapaian pemain.

# BAB V

## IMPLEMENTASI PROGRAM

Bab ini akan menjelaskan mengenai proses implementasi program hingga menjadi satu kesatuan program berbentuk game yang dapat dimainkan. Implementasi dilakukan berdasarkan hasil analisa dan desain sistem dengan menggunakan bahasa pemrograman. Hal yang akan dibahas melingkupi sistem pencatat dan pembaca log, penyesuai tingkat kesulitan rintangan, karakter, dan rintangan.

### 5.1 Sistem Pencatat dan Pembaca Log

Pada subbab 5.1 ini akan menjelaskan segmen program yang berkaitan dengan sistem pencatatan dan pembacaan log. Segmen program yang dibahas yaitu sistem pencatatan log, dan sistem pembacaan log.

#### 5.1.1 Sistem Pencatatan Log

Bagian ini menjelaskan sistem pencatatan log berdasarkan variabel yang ditampung selama melewati satu level dari berbagai jenis rintangan yang dilewati. Terdapat 2 segmen program pada bagian sistem pencatatan log yaitu segmen 5.1 Pencatatan Emosi, dan segmen 5.2 Pencatatan Log. Penjelasan segmen program akan dijabarkan setelah segmen program.

#### Segmen Program 5.1 Pencatatan Emosi

```
01: public string deteksi_emosi()
02: {
03:     string emotion = "";
04:     value_marah = EmotionsManager.Emotions.angry;
05:     value_neutral = EmotionsManager.Emotions.neutral;
06:     value_takut = EmotionsManager.Emotions.scared;
07:     if (value_neutral - value_takut > value_takut && value_neutral
08:         > value_marah){
09:         emotion = "Netral";
10:     }
11:     else if (value_takut > value_neutral - value_takut &&
12:             value_takut > value_marah){
```

### Segmen Program 5.1 (Lanjutan)

```

11:         emotion = "Takut";
12:     }
13:     else if (value_marah > value_neutral && value_takut <
14:             value_marah)
15:     {
16:         emotion = "Marah";
17:     }
18: }

```

Segmen Program 5.1 merupakan baris program yang digunakan untuk mencatat emosi yang dideteksi. Segmen program ini akan dipanggil setiap kali collider karakter pemain bertabrakan dengan collider rintangan dan ketika terkena serang hantu. Pada baris 1 merupakan nama function yang dipanggil oleh script collider rintangan dan berisi perintah hingga baris ke 17. Baris 3 merupakan deklarasi awal variable emosi. Baris 4 hingga 6 berisi deklarasi variable yang diisi oleh value deteksi emosi library MoodMe. Kemudian pada baris ke 7 hingga baris ke 9 merupakan kondisi pengecekan untuk menentukan emosi yang dideteksi yaitu “Netral”. Baris ke 10 hingga baris ke 12 merupakan fungsi yang serupa yaitu untuk mendeteksi emosi “Takut”. Baris ke 13 hingga baris ke 16 merupakan fungsi yang serupa yaitu untuk mendeteksi emosi “Marah”. Kemudian baris ke 17 berfungsi untuk mengembalikan value emotion untuk dibaca oleh skrip pencatat log.

### Segmen Program 5.2 Pencatatan Log

```

01: void AddData(string emotion)
02: {
03:     if (hantu)
04:     {
05:         obstacle = "hantu";
06:         status = hantu.status_karakter();
07:     }
08:     else if (jebakan)
09:     {
10:         obstacle = "duri";
11:         status = jebakan.status_karakter();
12:     }
13:     else if (jumpscares)
14:     {
15:         obstacle = "jumpscare_penampakan";
16:         status = jumpscares.status_karakter();
17:     }
18:     else if (bambus)
19:     {

```

### Segmen Program 5.2 (Lanjutan)

```

20:         obstacle = "bamboo";
21:         status = bambu.status_karakter();
22:     }
23:     Data data = new Data();
24:     data.emotion = emotion;
25:     data.obstacle = obstacle;
26:     data.status = status;
27:
28:     dataList.Add(data);
29: }
30: }
31: string ConvertListToJson(List<Data> list)
32: {
33:     string json = JsonConvert.SerializeObject(new {
34:         logEntries = list }, Formatting.Indented);
35:     return json;
36: }
37: void SaveJsonToFile(string json, string filename)
38: {
39:     string path =
40:         Path.Combine(Application.persistentDataPath, filename);
41:     File.WriteAllText(path, json);
42: }

```

Segmen Program 5.2 merupakan baris program yang digunakan untuk mencatat jenis lintangan, status, dan emosi lintangan yang dicatat dalam bentuk json. Baris 1 hingga baris 30 merupakan function yang mengolah bentuk data pencatatan. Baris ke 3,8,13, dan 18 merupakan pengecekan jenis lintangan. Baris ke 5 dan ke 6 akan mengisi jenis lintangan dan status dari jenis lintangan hantu. Baris ke 10 dan ke 11 akan mengisi jenis lintangan dan status dari jenis lintangan duri. Baris ke 15 dan ke 16 akan mengisi jenis lintangan dan status dari jenis lintangan jumpscare. Baris ke 20 dan ke 21 akan mengisi jenis lintangan dan status dari jenis lintangan bambu. Baris ke 23 hingga baris 28 berfungsi untuk menampung data menjadi bentuk array yang selanjutnya akan dikonversi menjadi json dengan function baris ke 31. Baris ke 31 merupakan function yang dipanggil untuk melakukan convert json, cara kerjanya pada baris ke 33 memanggil library JsonConvert dan melakukan serialize object yang berisi data array dengan nama list yang digunakan untuk menampung data jenis lintangan, status dan emosi. Kemudian baris ke 34 melakukan return hasil json. Baris ke 36 merupakan function untuk mengenerate file json dengan entri data berdasarkan isian json pada baris ke 34. Baris ke 38 berisi path yang dipilih dimana pada lokasi tersebut file akan

diexport, kemudian baris ke 39 melakukan write file dengan data dari json ke direktori yang ditentukan oleh path pada baris ke 38.

### 5.1.2 Sistem Pembacaan Log

Bagian ini menjelaskan sistem pembacaan log json yang telah digenerate. Terdapat 3 segmen program pada bagian sistem pembacaan log yaitu segmen 5.3 Pengecekan File Log, Segmen 5.4 Sistem Pembaca Log dan segmen 5.5 Pembacaan Log. Penjelasan segmen program akan dijabarkan setelah segmen program.

#### Segmen Program 5.3 Pengecekan File Log

```

01: public string LogFilePath
02: {
03:     get
04:     {
05:         if (string.IsNullOrEmpty(logFilePath))
06:         {
07:             logFilePath =
08:                 Path.Combine(Application.persistentDataPath,
09:                             "output.json");
10:         }
11:     }
12: }
```

Segmen Program 5.3 merupakan baris program yang digunakan untuk melakukan pengecekan apakah file log terdeteksi ada pada sistem. Baris ke 1 merupakan nama function yang dipanggil ketika log akan dibaca. Baris ke 5 melakukan pengecekan apakah filepath telah diset, jika belum maka file path pada baris ke 7 diisi oleh path default appData windows dan mencari file dengan nama output.json. Kemudian path direturn pada baris ke 9.

#### Segmen Program 5.4 Sistem Pembaca Log

```

01: public void ReadJsonLog()
02: {
03:     try
04:     {
05:         string jsonContent =
06:             System.IO.File.ReadAllText(LogFilePath);
07:         LogWrapper logWrapper =
08:             JsonUtility.FromJson<LogWrapper>(jsonContent);
09:         if (logWrapper != null && logWrapper.logEntries != null)
10:         {
11:             foreach (var entry in logWrapper.logEntries)
12:             {
13:                 LogEntry logEntry = new LogEntry();
14:                 logEntry.timeStamp = entry.timeStamp;
15:                 logEntry.message = entry.message;
16:                 logEntry.level = entry.level;
17:                 logEntries.Add(logEntry);
18:             }
19:         }
20:     }
21: }
```

### Segmen Program 5.4 (Lanjutan)

```

09:             logEntries.AddRange(logWrapper.logEntries);
10:         }
11:     }
12:     catch (System.Exception e)
13:     {
14:         Debug.LogError("Error reading JSON log: " + e.Message);
15:     }
16: }
```

Segmen Program 5.4 merupakan baris sistem yang digunakan untuk melakukan pembacaan file log json. Baris ke 3 melakukan try-catch pada file json, akan masuk kondisi catch jika tidak bisa membaca file log pada baris ke 13 dengan memunculkan debug error pada baris ke 14. Jika log dapat dibaca maka baris ke 5 akan melakukan perintah membaca semua teks dengan memanfaatkan library system.io dengan path yang telah dicari pada segmen program 5.3. Kemudian baris ke 6 memanggil library JsonUtility yang membaca konten hasil dari baris ke 5. Baris ke 9 melakukan penambahan data ke logEntries berdasarkan data yang diolah.

### Segmen Program 5.5 Pembacaan Log

```

01: public void BacaLog()
02: {
03:     pembacaLogs = GetComponent<PembacaLog>();
04:     List<PembacaLog.LogEntry> logEntries =
05:         pembacaLogs.LogEntries;
06:     foreach (PembacaLog.LogEntry entry in logEntries)
07:     {
08:         if (entry.obstacle == "hantu")
09:         {
10:             if (entry.status == "Lolos")
11:             {
12:                 ctr_hantu_sukses++;
13:                 if (entry.emotion == "Netral")
14:                 {
15:                     skor_emosi_hantu += 2;
16:                 }
17:                 else if (entry.emotion == "Marah")
18:                 {
19:                     skor_emosi_hantu++;
20:                 }
21:             else if (entry.status == "Gagal")
22:             {
23:                 ctr_hantu_gagal++;
24:             }
25:             hantu_tidak_injak--;
26:         }
27:         else if (entry.obstacle == "jumpscare_penampakan")
```

### Segmen Program 5.5 (Lanjutan)

```

28:         {
29:             if (entry.status == "Aktif")
30:             {
31:                 ctr_penampakan_muncul++;
32:                 if (entry.emotion == "Netral")
33:                 {
34:                     skor_emosi_penampakan += 2;
35:                 }
36:                 else if (entry.emotion == "Marah")
37:                 {
38:                     skor_emosi_penampakan += 1;
39:                 }
40:             }
41:             penampakan_tidak_injak--;
42:         }
43:         else if (entry.obstacle == "duri")
44:         {
45:             if (entry.status == "Gagal")
46:             {
47:                 ctr_duri_gagal++;
48:                 if (entry.emotion == "Marah")
49:                 {
50:                     skor_emosi_duri += 1;
51:                 }
52:             }
53:             else if (entry.status == "Sukses")
54:             {
55:                 ctr_duri_berhasil++;
56:                 skor_emosi_duri += 2;
57:             }
58:             duri_tidak_injak--;
59:         }
60:         else if (entry.obstacle == "bambu")
61:         {
62:             ctr_masuk_bambu++;
63:             if (entry.emotion == "Marah")
64:             {
65:                 skor_emosi_bambu += 1;
66:             }
67:             else if (entry.emotion == "Normal")
68:             {
69:                 skor_emosi_bambu += 2;
70:             }
71:             bambu_tidak_dimasuki--;
72:         }
73:     }
74:     HitungSkorDDARintangan(sukses, gagal, emosi, noninjak);
75: }

```

Segmen Program 5.5 merupakan baris program yang digunakan untuk memetakan menjadi kategori rintangan berdasarkan file json yang dibaca. Baris ke 3 melakukan get component apakah PembacaLog terbaca. Baris ke 5 melakukan

looping untuk membaca entri data pada logEntries yang dideklarasi pada baris ke 4. Baris ke 7, 27, 43, dan ke 60 berfungsi untuk memetakkan jenis rintangan. Baris ke 9, 21, 29, 45, dan 53 melakukan pengecekan log berisi status. Jika memenuhi syarat maka counter akan diubah seperti pada baris ke 11, 23, 25, 31, 41, 47, 55, 58, 62, dan 71. Jika gagal maka baris ke 21 akan berjalan dan counter hantu gagal akan ditambahkan pada baris ke 23. Kemudian baris ke 12, 16, 32, 36, 48, 63, dan 67 melakukan pengecekan emosi yang terdeteksi pada masing-masing kategori. Jika emosi netral maka skor\_emosi\_kategori akan ditambah 2, jika emosi marah maka skor\_emosi\_kategori akan ditambah 1. Kemudian jika baris log sudah habis terbaca, berikutnya dilakukan pemanggilan function DDA rintangan dengan parameter yang dibutuhkan pada baris ke 74.

## 5.2 Sistem Perubah Tingkat Kesulitan Rintangan

Pada subbab 5.2 ini akan menjelaskan segmen program yang berkaitan dengan sistem perubah tingkat kesulitan rintangan. Segmen program yang dibahas yaitu sistem penentuan kategori pemain, perhitungan skor emosi pemain, dan sistem penyesuai tingkat kesulitan rintangan.

### 5.2.1 Sistem Penentuan Kategori Pemain

Bagian ini menjelaskan sistem penentuan kategori pemain berdasarkan performa pemain selama melewati satu level dengan berbagai jenis rintangan yang diolah DDA. Terdapat 1 segmen program pada bagian sistem penentuan kategori pemain yaitu segmen 5.6 Penentuan Kategori Pemain. Penjelasan segmen akan dijabarkan setelah segmen program.

#### Segmen Program 5.6 Penentuan Kategori Pemain

```

01: jumlahJalanHantu = GlobalStatic.ctr_pjhantu;
02: float htg_jalahantu = jumlahJalanHantu*100;
03: float htg_hantusukses =
    (ctr_hantu_sukses+hantu_tidak_injak)*100;
04: float htg_hantugagal = ctr_hantu_gagal*100;
05: string kategori_pemain = "";
06: if (jumlahJalanHantu == 0)
07: {
08:     kategori_pemain = "Normal";

```

### Segmen Program 5.6 (Lanjutan)

```

09:     }
10:    else if(hantu_tidak_injak==jumlahJalanHantu)
11:    {
12:        kategori_pemain = "Mahir";
13:    }
14:   else
15:   {
16:       float persentase_sukses = (htg_hantusukses /
17:                                   htg_jalahanhantu)*100;
18:       float persentase_gagal = (htg_hantugagal /
19:                                   htg_jalahanhantu)*100;
20:       if (persentase_sukses<39)
21:       {
22:           kategori_pemain = "Pemula";
23:       }
24:       else if (persentase_sukses == persentase_gagal)
25:       {
26:           kategori_pemain = "Normal";
27:       }
28:       else if (persentase_sukses > persentase_gagal &&
29:                persentase_sukses > 50)
30:       {
31:           kategori_pemain = "Normal";
32:       }
33:       else if (persentase_sukses > persentase_gagal &&
34:                persentase_sukses > 75)
35:       {
36:           kategori_pemain = "Mahir";
37:       }
38:   }

```

Segmen Program 5.6 merupakan baris program untuk menentukan kategori pemain berdasarkan kategori yang telah dipetakkan. Segmen Program 5.6 ini mengambil contoh kategori hantu. Baris ke 1 hingga baris ke 5 merupakan variable deklarasi awal sebelum perhitungan. Baris ke 6 melakukan pengecekan apakah pada level tidak tedapat jalan hantu, jika iya maka kategori pemain akan dibuat normal yang berfungsi untuk merandom jalan dengan weight ceiling normal sehingga jalan tidak menjadi 0 lagi. Baris ke 10 melakukan pengecekan apakah jalan dilewati semua, jika iya maka performa pemain akan dibuat menjadi mahir. Kemudian baris ke 14 merupakan kondisi normal dan melakukan perhitungan didalamnya. Baris ke 16 dan 17 melakukan perhitungan persentase keberhasilan dan kegagalan pemain dalam melewati rintangan. Kemudian pada baris ke 18, 22,

26, 30 dan 34 melakukan pengecekan performa pemain berdasarkan perbandingan persentase keberhasilan dengan kegagalan pemain. Kemudian kategori pemain akan ditentukan pada baris ke 20, 24, 28, 32, dan 36. Selanjutnya data kategori pemain akan digunakan untuk Segmen Program 5.7 Perhitungan Skor Emosi.

### **5.2.2 Sistem Perhitungan Skor Emosi**

Bagian ini menjelaskan sistem perhitungan skor emosi pemain oleh DDA. Terdapat 1 segmen program pada bagian perhitungan skor emosi DDA yaitu segmen 5.7 Perhitungan Skor Emosi.

#### **Segmen Program 5.7 Perhitungan Skor Emosi**

```

01: skor_emosi_hantu = skor_emosi_hantu+(hantu_tidak_injak*2);
02: float persentaseKetakutan =
    Mathf.Clamp01((float)skor_emosi_hantu / jumlahJalanHantu);
03: if(hantu_tidak_injak==jumlahJalanHantu)
04: {
05:     skor_emosi_hantu=2*jumlahJalanHantu;
06:     kategori_pemain = "Mahir";
07: }
08: else if (kategori_pemain=="Pemula" && jumlahJalanHantu != 0)
09: {
10:     if (persentaseKetakutan < 1f)
11:     {
12:         kategori_pemain = "Pemula";
13:     }
14:     else
15:     {
16:         kategori_pemain = "Normal";
17:     }
18: }
19: else if (kategori_pemain=="Normal" && jumlahJalanHantu != 0)
20: {
21:     if (persentaseKetakutan > 1.5f)
22:     {
23:         kategori_pemain = "Mahir";
24:     }
25:     else
26:     {
27:         kategori_pemain = "Normal";
28:     }
29: }
30: else if (kategori_pemain == "Mahir" && jumlahJalanHantu != 0)
31: {
32:     if (persentaseKetakutan < 1.5f)
33:     {
34:         kategori_pemain = "Normal";

```

### Segmen Program 5.7 (Lanjutan)

```

35:     }
36: else
37: {
38:     kategori_pemain = "Mahir";
39: }
40: }
41: else
42: {
43:     kategori_pemain = "Mahir";
44: }

```

Segmen Program 5.7 merupakan baris program untuk memvalidasi skor emosi pemain yang terdeteksi. Baris 1 dan 2 merupakan deklarasi awal variable untuk menghitung persentase ketakutan. Baris ke 3 melakukan pengecekan hantu\_tidak\_injak, jika sama dengan jumlahJalanHantu maka skor emosi akan dikalikan 2 sebagai representasi emosi normal atau biasa saja. Kemudian baris ke 6 melakukan penggolongan pemain “Mahir”. Baris ke 8 dijalankan apabila kondisi baris ke 3 tidak terpenuhi dan kategori pemain yang didapat dari Segmen Program 5.6 hasilnya “Pemula”. Kemudian baris ke 10 melakukan pengecekan apakah persentase ketakutan yang diperoleh dibawah 1, jika iya maka pemain dianggap takut dan kategori pemainnya tetap “Pemula”, jika tidak maka baris perintah ke 14 akan berjalan dan men-set kategori pemain menjadi “Normal” pada baris ke 16. Baris ke 19 dijalankan apabila kondisi baris ke 10 tidak terpenuhi dan kategori pemain yang didapat dari Segmen Program 5.6 hasilnya “Normal”. Baris ke 21 melakukan pengecekan apakah persentase ketakutan yang diperoleh diatas 1.5, jika iya maka kategori pemain akan diubah menjadi “Mahir”, jika tidak akan diubah tetap menjadi “Normal”. Kemudian baris ke 30 dijalankan apabila kondisi baris ke 19 tidak terpenuhi dan kategori pemain yang didapat dari Segmen Program 5.6 hasilnya “Mahir”. Pengecekan dilakukan pada baris ke 32 apakah persentase ketakutan pemain dibawah 1.5, jika iya maka kategori pemain akan di-set “Normal” pada baris ke 34. Sedangkan baris ke 38 dijalankan jika tidak lolos pengecekan pada baris ke 32.

### 5.2.3 Sistem Penyesuai Tingkat Kesulitan Rintangan

Bagian ini menjelaskan sistem penyesuai tingkat kesulitan rintangan oleh DDA. Terdapat 1 segmen program pada bagian penyesuai kesulitan rintangan yaitu segmen 5.8 Sistem Penyesuai Tingkat Kesulitan.

#### Segmen Program 5.8 Sistem Penyesuai Tingkat Kesulitan

```

01: int[] wc_hantu = new int[] { 2, 4, 5 };
02: if (kategori_pemain == "Pemula")
03: {
04:     GlobalStatic.init_jumlahJalanHantu = wc_hantu[0];
05: }
06: else if (kategori_pemain == "Normal")
07: {
08:     GlobalStatic.init_jumlahJalanHantu = wc_hantu[1];
09: }
10: else if (kategori_pemain == "Mahir")
11: {
12:     GlobalStatic.init_jumlahJalanHantu = wc_hantu[2];
13: }
```

Segmen Program 5.8 merupakan baris program untuk mengatur weight ceiling dari rintangan kategori hantu. Baris ke 1 merupakan array yang menampung batas maksimum rintangan hantu yang akan spawn. Baris ke 2, 6 dan 10 merupakan baris pengecekan kategori pemain. Baris ke 4, 8, dan 12 memodifikasi nilai inisialisasi jumlah\_JalanHantu pada variabel global yang akan dibaca oleh generator level.

## 5.3 Karakter

Subbab 5.3 ini akan menjelaskan mengenai segmen program yang berkaitan dengan karakter. Segmen program yang dibahas yaitu karakter bergerak, merangkak, melompat, memukul dan karakter saat terkena serang.

### 5.3.1 Kontrol Gerak Karakter

Bagian ini menjelaskan kontrol pemain untuk menggerakkan karakter ke kiri dan kekanan, berjalan, dan berlari. Karakter dalam game ini digerakkan menggunakan kontrol keyboard. Isi dari kontrol gerak karakter akan dibahas dalam segmen program 5.9.

### Segmen Program 5.9 Kontrol Gerak Karakter

```

01: gerak = Input.GetAxisRaw("Horizontal");
02: if (Input.GetKeyDown(KeyCode.S))
03: {
04:     if (jongkok)
05:     {
06:         modeBerdiri();
07:     }
08:     else
09:     {
10:         modeJongkok();
11:     }
12: }
13: if (Input.GetKey(KeyCode.LeftShift) && !jongkok)
14: {
15:     modeLari();
16: }
17: else if (!Input.GetKey(KeyCode.LeftShift) && !jongkok)
18: {
19:     modeJalan();
20: }
21: if (Input.GetButtonDown("Jump") && !jongkok)
22: {
23:     modeLompat();
24: }
25: if (Input.GetKey(KeyCode.O) && isAttack == false)
26: {
27:     modePukul();
28: }
29:
30: public void modeBerdiri()
31: {
32:     if (!isCrawling)
33:     {
34:         jongkok = false;
35:         modeSerang = 0.8f;
36:         if (lari)
37:         {
38:             audioSource.clip = runSound;
39:             moveSpeed = dxLari;
40:         }
41:     }
42: }

```

Segmen Program 5.9 merupakan baris program yang digunakan untuk menggerakan karakter didalam game. Pada baris 1 dilakukan pengambilan input dari keyboard yang telah diatur dengan menggunakan tombol A untuk bergerak kekiri dan tombol D untuk bergerak kekanan. Baris ke 4 hingga baris ke 11 merupakan pengecekan apakah karakter berada dalam kondisi merangkak. Jika pemain menekan tombol S saat posisi karakter berada dalam kondisi merangkak,

maka karakter akan kembali kedalam mode berdiri. Jika pemain menekan tombol S pada saat karakter dalam posisi lain, maka karakter akan masuk kedalam mode merangkak. Baris ke 13 hingga baris ke 20 merupakan pengecekan kontrol karakter berlari. Terdapat pengecekan apakah pemain menekan tombol Shift dan tidak dalam mode merangkak, jika iya maka karakter masuk kedalam mode lari. Jika tidak maka karakter akan berada dalam mode jalan biasa. Baris ke 21 hingga baris ke 24 merupakan baris program untuk melakukan lompat pada karakter. Terdapat pengecekan apakah pemain menekan tombol Space dan tidak dalam mode merangkak, jika iya maka karakter masuk mode lompat. Baris ke 25 hingga baris ke 28 merupakan pengecekan apakah pemain menekan tombol O, jika iya maka karakter akan masuk ke mode pukul. Kemudian pada baris ke 30 merupakan function pengaturan saat pemain berdiri pemain dapat melakukan aksi lari dan pengaturan serangan menjadi mode memukul. Pada baris ke 32 dilakukan pengecekan apakah karakter saat ini sedang dalam mode menunduk, jika lolos pengecekan maka jongkok difalsekan dan mode serang diatur menjadi mode memukul pada baris ke 35 hingga baris ke 36. Kemudian pada baris ke 36 dilakukan pengecekan apakah karakter saat ini dalam mode lari, jika iya maka mainkan suara lari dan speed karakter diganti menjadi speed lari pada pada baris ke 38 dan baris ke 39.

### 5.3.2 Karakter Merangkak

Bagian ini menjelaskan segmen program saat pemain menekan tombol aksi untuk membuat karakter merangkak. Terdapat 2 segmen program pada ini, segmen 5.10 Mode Merangkak, dan segmen 5.11 Kontrol Merangkak.

#### Segmen Program 5.10 Mode Merangkak

```

01: public void modeJongkok()
02: {
03:     if (isGrounded)
04:     {
05:         jongkok = true;
06:         modeSerang = 0f;
07:     }
08: }
```

Segmen Program 5.10 merupakan baris program yang digunakan untuk membuat karakter melakukan aksi merangkak. Pada baris 1 merupakan nama function yang dipanggil ketika pemain menekan tombol S dan berisi perintah hingga baris ke 8. Baris 3 hingga baris 5 berisi pengecekan apakah pemain berada diatas tanah, jika iya maka karakter dapat melakukan aksi merangkak, dan mode serangnya dijadikan 0.

### **Segmen Program 5.11 Kontrol Merangkak**

```
01: if (jongkok)
02: {
03:     isCrawling = Physics2D.OverlapCircle(crawlCheck.position,
04:                                             upRadius, whatIsTop);
05: }
```

Segmen Program 5.11 merupakan baris program yang digunakan untuk membuat karakter melakukan aksi merangkak. Baris 1 berisi pengecekan apakah pemain sedang dalam mode merangkak, jika iya maka collider circle pemain diisi oleh crawlcheck, upradius dan whatistop saat itu. What is top merupakan variable yang diisi oleh gameobject yang ditentukan oleh pemain, agar bisa menseleksi layer platform diatas pemain jika ada. Up radius merupakan variable yang mengatur seberapa setinggi collider karakter saat mode merangkak.

### **5.3.3 Karakter Melompat**

Bagian ini menjelaskan segmen program saat pemain menekan tombol Space untuk membuat karakter melompat. Terdapat 2 segmen program pada bagian karakter melompat, segmen 5.12 Mode Loncat, dan segmen 5.13 Kontrol Lompat. Penjelasan setiap segmen akan dijabarkan setelah segmen program.

### **Segmen Program 5.12 Mode Lompat**

```
01: public void modeLompat()
02: {
03:     if (!jongkok && !lompat && (isGrounded || upWater))
04:     {
05:         lompat = true;
06:     }
07: }
```

Segmen Program 5.12 merupakan baris program yang digunakan untuk membuat karakter melakukan aksi loncat. Pada baris 1 merupakan nama function yang dipanggil ketika pemain menekan tombol Space dan berisi perintah hingga baris ke 7. Baris 3 hingga baris 5 berisi pengecekan jika karakter tidak dalam mode merangkak dan tidak lompat, dan sedang berada di atas tanah dan di atas permukaan air, maka pemain dapat melakukan aksi lompat dengan cara mengubah mode lompat menjadi true.

### **Segmen Program 5.13 Kontrol Lompat**

```

01: if (lompat)
02: {
03:     lompat = false;
04:     obj_sound.GetComponent<soundGlobal>().doLompatSound();
05:     if (upWater)
06:     {
07:         rb.velocity = Vector2.up * 1.5f * jumpVelocity;
08:         obj_sound.GetComponent<soundGlobal>().dobangkit();
09:     }
10:     else
11:     {
12:         rb.velocity = Vector2.up * jumpVelocity;
13:     }
14: }
```

Segmen Program 5.13 merupakan baris program yang digunakan untuk membuat karakter melakukan aksi loncat. Baris ke 1 berisi perintah pengecekan apakah saat ini status lompat dari karakter. Jika true, maka pada baris ke 3 lompat diganti menjadi false dan mainkan suara karakter lompat. Baris ke 5 hingga baris ke 9 merupakan pengecekan jika karakter berada diair maka akan merubah jumlah tinggi lompatan dan memainkan suara karakter keluar dari air. Pada baris ke 10 hingga baris ke 14 merupakan baris karakter loncat diatas tanah, lompatan dilakukan dengan cara mengganti value rigidbody karakter dengan fungsi vector2.up yang dimodifikasi dengan jumpvelocity yang sudah ditentukan.

#### **5.3.4 Karakter Memukul**

Bagian ini menjelaskan segmen program saat pemain menekan tombol O untuk membuat karakter memukul rintangan duri dan menghancurkan halangan

yang ada. Terdapat 3 segmen program pada bagian karakter memukul yaitu segmen 5.14 Mode Pukul, segmen 5.15 Serangan Jenis Pukul dan segmen 5.16 Kontrol Pukul. Penjelasan setiap segmen akan dijabarkan setelah segmen program.

### **Segmen Program 5.14 Mode Pukul**

```
01: public void modePukul()
02: {
03:     if (!isAttack)
04:     {
05:         init_delaySerang = init_delaySerangPukul;
06:         delaySerang = init_delaySerang;
07:         seranganPukul();
08:     }
09: }
```

Segmen Program 5.14 merupakan baris program yang digunakan untuk melakukan aksi memukul. Pada baris 1 merupakan nama function yang dipanggil ketika pemain menekan tombol O dan berisi perintah hingga baris ke 9. Baris ke 3 mengecek apakah pemain saat ini tidak dalam mode memukul, jika lolos pengecekan maka pada baris ke 5 hingga baris ke 6 terdapat perintah untuk memberi value delay serangan dengan delay serangan jenis pukulan yang sudah ditentukan. Kemudian pada baris ke 7 dilakukan pemanggilan function serangan jenis pukulan.

### **Segmen Program 5.15 Serangan Jenis Pukul**

```
01: private void seranganPukul()
02: {
03:     if (!isAttack && isGrounded)
04:     {
05:         isAttack = true;
06:         if (jongkok && isGrounded)
07:         {
08:             modeSerang = 0.8f;
09:             areaPukul.GetComponent
10:                 <AreaPukulKarakter>().doPukul();
11:         }
12:         areaPukul.SetActive(true);
13:         an.SetFloat("AttackType", modeSerang);
14:         an.SetBool("isAttack", true);
15:     }
}
```

Segmen Program 5.15 merupakan baris program yang dipanggil ketika pemain lolos pengecekan function modePukul(). Segmen program ini terdiri atas

15 baris perintah, setiap baris perintah akan dijelaskan pada kalimat berikut. Pada baris 3 dilakukan pengecekan jika pemain tidak sedang menyerang dan sedang berada diatas tanah, maka pada baris ke 5 mode pemain diubah menjadi attack sama dengan true. Kemudian pada baris ke 6 dilakukan pengecekan lagi jika pemain berada dalam mode jongkok dan sedang berada diatas tanah, maka serangan berupa bentuk pukulan pada baris ke 8. Baris ke 9 memanggil skrip AreaPukulKarakter dimana ini fungsinya serupa seperti baris 11 yaitu melakukan pukulan pada deteksi area pukul. Apabila layer yang dipukul terdeteksi sebagai layer rintangan, maka akan dilakukan function doPukul() yang menghilangkan gameobject rintangan yang dipukul. Kemudian pada baris 14 dan 15 memainkan animasi mode serang pukul dari karakter.

### **Segmen Program 5.16 Kontrol Pukul**

```

01: if (isAttack)
02: {
03:     delaySerang = delaySerang - Time.deltaTime;
04:     if (delaySerang <= 0)
05:     {
06:         if (modeSerang == 1 || modeSerang == 0.6f)
07:         {
08:             modeSerang = 0f;
09:         }
10:     else
11:     {
12:         areaPukul.SetActive(false);
13:     }
14:     isAttack = false;
15:     an.SetBool("isAttack", false);
16:     delaySerang = init_delaySerang;
17: }
18: }
```

Segmen Program 5.16 merupakan baris program yang digunakan untuk melakukan aksi memukul. Baris ke 1 berisi perintah pengecekan apakah saat ini status attack dari karakter yang dimainkan sama dengan true. Jika true, maka pada baris ke 3 delay serang diubah valuenya berdasarkan variable time yang sudah disiapkan. Kemudian pada baris ke 4 dilakukan pengecekan apakah waktu delayserang lebih kecil daripada 0 atau sama dengan 0. Jika iya pada baris ke 6 dilakukan pengecekan lagi apakah mode serang saat ini 1 atau modeserangnya sama

dengan 0.6f, jika iya maka pada baris ke 8 mode serangnya direset menjadi 0f. Jika tidak areaPukul difalsekan pada baris ke 12. Kemudian pada baris ke 14 dan 15 is attack dan animasi attack dari karakter difalse kan kembali. Dan pada baris 16 delay serang dikembalikan dengan value init delayserang awal.

### 5.3.5 Karakter Kena Serang

Bagian ini menjelaskan segmen program saat pemain terkena serang oleh hantu atau rintangan yang terdapat pada level. Terdapat 3 segmen program pada bagian karakter terkena serang yaitu segmen 5.17 Kena Serangan, segmen 5.18 Karakter Mati dan segmen 5.19 Karakter Langsung Mati. Penjelasan setiap segmen akan dijabarkan setelah segmen program.

#### Segmen Program 5.17 Kena Serangan

```

01: public void kenaSerangan(GameObject paramObj = null)
02: {
03:     if (isHurting == false)
04:     {
05:         isHurting = true;
06:         healthPoints -= 1;
07:         dirX = 0;
08:         if (healthPoints > 0)
09:         {
10:             if (audioSource.isPlaying && !inWater)
11:             {
12:                 audioSource.Stop();
13:             }
14:             gameObject.GetComponent<SpriteRenderer>().color = Color.red;
15:             an.SetTrigger("isHurting");
16:             rb.velocity = Vector2.zero;
17:             if (paramObj.tag == "Jebakan")
18:             {
19:                 if (facingRight)
20:                 {
21:                     rb.AddForce(new Vector2(-200f, 200f));
22:                 }
23:                 else
24:                 {
25:                     rb.AddForce(new Vector2(200f, 200f));
26:                 }
27:             }
28:         }
29:     }
30:     {
31:         if (audioSource.isPlaying && !inWater)
32:     {

```

### Segmen Program 5.17 (Lanjutan)

```

33:             audioSource.Stop();
34:         }
35:         isDead = true;
36:         StartCoroutine("Die");
37:     }
38: }
39: }
```

Program 5.17 merupakan baris program yang digunakan untuk mendeteksi jika pemain sedang terkena serangan. Pada baris 1 merupakan nama function yang dipanggil ketika pemain terkena serang dengan parameter paramObj yang dinull kan valuenya. Segmen ini berisi perintah hingga baris ke 39. Pada baris ke 2 terdapat pengecekan apakah saat ini pemain sedang terkena serang, jika iya maka akan masuk ke baris 3. Baris ke 3 mengubah value isHurting menjadi true kemudian jumlah darah pemain dikurang 1 pada baris ke 4 dan gerakan karakter pada variable dirx dikembalikan menjadi 0. Pada baris ke 8 dilakukan pengecekan apakah sisanya wawa pemain diatas 0, jika iya maka pada baris ke 10 hingga baris ke 13 bertugas menghentikan suara karakter terkena serang. Baris ke 14 memberikan efek filter merah kepada pemain saat terkena serang. Baris ke 15 digunakan untuk memainkan animasi pemain terkena serang. Baris ke 16 membuat karakter tidak lanjut bergerak kedepan saat terkena serang. Baris ke 17 hingga baris ke 27 berfungsi untuk memberikan efek terpental saat pemain terkena rintangan jebakan, efek terpental diberikan sesuai dengan arah karakter saat itu. Kemudian baris ke 31 hingga baris ke 34 berfungsi sama seperti baris ke 10 hingga baris ke 13, yaitu menghentikan suara setelah pemain terkena serang. Baris ke 35 berfungsi untuk memberi status karakter mati ketika nyawanya mencapai 0. Baris ke 36 menjalankan segmen program Karakter Mati.

### Segmen Program 5.18 Karakter Mati

```

01: IEnumerator Die()
02: {
03:     yield return new WaitForSeconds(2f);
04:     Debug.Log("karakter mati");
05:
06: }
```

Segmen Program 5.18 merupakan baris program yang berguna untuk memberi jeda waktu saat karakter mati. Baris ke 1 merupakan penamaan fungsi IEnumarator Die. Baris ke 3 berfungsi untuk memberi jeda 2 detik sebelum perintah baris ke 4 dijalankan. Baris ke 4 berfungsi untuk memberikan status bahwa karakter mati.

### **Segmen Program 5.19 Karakter Langsung Mati**

```
01: public void langsungMati()
02: {
03:     healthPoints = 0;
04:     isDead = true;
05:     an.SetTrigger("isDead");
06:     StartCoroutine("Die");
07: }
```

Segmen Program 5.19 merupakan baris program yang dipanggil saat karakter ditentukan akan langsung mati tanpa memperhitungkan sisa nyawa yang tersisa. Baris ke 1 merupakan penamaan fungsi langsungMati. Baris ke 3 membuat sisa nyawa pemain menjadi 0, kemudian pada baris ke 4 status pemain mati dijadikan true. Baris ke 5 mengatur agar animasi mati pemain dijalankan. Baris ke 6 menjalankan segmen program Karakter Mati.

## **5.4 Rintangan**

Subbab 5.3 ini akan menjelaskan mengenai segmen program yang berkaitan dengan rintangan yang dimunculkan selama permainan. Segmen program yang dibahas yaitu rintangan hantu, rintangan duri, rintangan jumpscare penampakan, dan tempat bersembunyi.

### **5.4.1 Rintangan Hantu**

Bagian ini menjelaskan rintangan hantu berkerja dalam game. Rintangan jenis ini memiliki 3 segmen program yaitu segmen program 5.20 Generate Pijakan Hantu, 5.21 Cek Penginjakan, dan 5.22 Memunculkan Hantu. Penjelasan setiap segmen akan dijabarkan setelah segmen program.

### Segmen Program 5.20 Generate Pijakan Hantu

```

01: public void randomjalan_PCG(int x, int id)
02: {
03:     int min = 0;
04:     int max = 7;
05:     int randomValue = Random.Range(min, max);
06:
07:     if (randomValue == id)
08:     {
09:         randomValue++;
10:         randomjalan_PCG(x, id);
11:     }
12:     else
13:     {
14:         if (randomValue == 0)
15:         {
16:             jumlahJalanHantu--;
17:             if (jumlahJalanHantu >= 0)
18:             {
19:                 buatObject(jalan_hantu_single, x, tinggi,
20:                             layer_jalan);
21:             }
22:             else
23:             {
24:                 randomjalan_PCG(x, 0);
25:             }
26:         }
27:         {
28:             buatObject(rumput, x, tinggi, layer_jalan);
29:         }
30:     }
31: }
```

Segmen Program 5.20 merupakan baris program yang digunakan untuk membuat balok platform pijakan hantu. Baris ke 1 penamaan fungsi random jalan platform. Baris ke 3 hingga baris ke 5 merupakan deklarasi variable pengacak. Baris ke 9 mengecek apakah hasil random sama dengan value variable id, jika sama maka random value akan ditambah 1 pada baris ke 11 dan akan menjalankan fungsi randomjalan\_PCG sekali lagi. Jika hasil random tidak sama dengan value variable id, maka akan dilakukan pengecekan pada baris ke 16, apakah hasil random merupakan angka 0, jika iya maka variable jumlahJalanHantu akan dikurang 1 pada baris 18. Kemudian pada baris ke 19 di cek, apakah jumlahJalanHantu nilainya masih diatas atau sama dengan 0, jika iya maka akan membuat object jalan hantu single pada baris ke 21 dengan memanggil fungsi buatObject, jika tidak maka akan menjalankan fungsi randomjalan\_PCG dengan value id 0. Jika hasil random bukan 0

maka perintah baris ke 29 dijalankan, yaitu memanggil fungsi buatObject yang diisi dengan gameobject platform blok rumput.

## Segmen Program 5.21 Cek Penginjakan

```
01: private void OnTriggerEnter2D(Collider2D collision)
02: {
03:     penangkapEmosi.deteksi_emosi();
04:     if (useItems_ != null)
05:     {
06:         if (useItems_.GetComponent<glob_Items>
07:             ().getPakaiGunting() == true || sudahDipijak ==
08:                 true) {
09:             }
10:         else
11:         {
12:             if (collision.gameObject.CompareTag
13:                 ("Player") && sudahDipijak == false){
14:                 if (hantu != null){
15:                     if (hantu.GetComponent
16:                         <Hantu_single>() != null &&
17:                         hantu.GetComponent
18:                         <Hantu_single>().startMuncul ==
19:                             false){
20:                             hantu.GetComponent<Hantu_single>()
21:                             .munculHantu();
22:                             sudahDipijak = true;
23:                         }
24:                     }
25:                 }
26:             }
27:         }
28:     }
29: }
```

Segmen Program 5.21 merupakan baris program yang digunakan untuk melakukan cek penginjakan blok oleh karakter. Baris ke 3 mencatat emosi saat pemain melakukan penginjakan. Baris ke 4 melakukan pengecekan apakah pemain memiliki item. Jika lolos pengecekan maka perintah baris ke 6 dilakukan pengecekan apakah pemain menggunakan item gunting saat pijakan terinjak, jika iya maka akan hantu tidak akan muncul. Pada baris ke 10 terdapat pengecekan apakah pemain menginjak pijakan saat kondisi pijakan tidak pernah diinjak, jika iya maka dilakukan pengecekan pada baris ke 12. Baris ke 12 melakukan pengecekan apakah hantunya tidak null, jika iya maka masuk baris perintah ke 13 yang melakukan pengecekan apakah Hantu\_single sudah diisi gameobject dan

sedang tidak muncul. Ketika lolos penecekan maka baris ke 15 memanggil fungsi munculHantu() pada gameobject Hantu\_single yang sudah memiliki script dengan fungsi munculHantu(), kemudian variable sudahDipijak diberi value true.

## **Segmen Program 5.22 Memunculkan Hantu**

```
01: public void munculHantu()
02: {
03:     doAttack = true;
04:     gameObject.transform.position = karakter.transform.position;
05:     if (camera != null){
06:         camera.GetComponent<Camera_Karakter>().SetFovKarakterDekat();
07:     }
08:     startMuncul = true;
09:     poinLight.SetActive(true);
10:     karakter_hantu.SetActive(true);
11:     int min = 0;
12:     int max = 100;
13:     int randomValue = Random.Range(min, max);
14:     if (randomValue % 2 != 0 ){
15:         nempelPemain = true;
16:         randomValue = Random.Range(min, max);
17:         if (randomValue % 2 != 0 ){
18:             if (karakter.transform.localScale.x > 0){
19:                 karakter_hantu.transform.localScale = new Vector3(-0.3f, 0.3f, 0.3f);
20:                 karakter_hantu.transform.parent =
21:                     karakter.transform;
22:                 karakter_hantu.transform.position = new Vector2(karakter_hantu.transform.position.x - 1.5f, karakter_hantu.transform.position.y+0.5f);
23:                 poinLight.transform.parent = karakter.transform;
24:             }
25:             else{
26:                 karakter_hantu.transform.localScale = new Vector3(0.3f, 0.3f, 0.3f);
27:                 karakter_hantu.transform.parent =
28:                     karakter.transform;
29:                 karakter_hantu.transform.position = new Vector2(karakter_hantu.transform.position.x + 1.5f,karakter_hantu.transform.position.y+0.5f);
30:                 poinLight.transform.parent = karakter.transform;
31:             }
32:         else{
33:             if (karakter.transform.localScale.x < 0){
34:                 karakter_hantu.transform.localScale = new Vector3(-0.3f, 0.3f, 0.3f);
35:                 karakter_hantu.transform.parent = karakter.transform;
36:                 karakter_hantu.transform.position = new Vector2(karakter_hantu.transform.position.x - 2f, karakter_hantu.transform.position.y);
```

### Segmen Program 5.22 (Lanjutan)

```

37:         poinLight.transform.parent = karakter.transform;
38:     }
39:     else
40:     {
41:         karakter_hantu.transform.localScale = new
42:             Vector3(0.3f, 0.3f, 0.3f);
43:         karakter_hantu.transform.parent = karakter.transform;
44:         karakter_hantu.transform.position = new
45:             Vector2(karakter_hantu.transform.position.x + 2f,
46:                     karakter_hantu.transform.position.y);
47:         poinLight.transform.parent = karakter.transform;
48:     }
49:     nempelPemain = false;
50: }
51: }
```

Segmen Program 5.22 merupakan baris program yang digunakan untuk memunculkan hantu setelah karakter menginjak blok pijakan berisi hantu. Baris ke 1 merupakan penamaan fungsi munculHantu(). Baris ke 3 mengatur doAttack menjadi true dan baris ke 4 mengubah posisi gameobject sesuai dengan posisi karakter saat itu. Baris ke 5 mengatur pengecekan, jika kamera tidak null maka baris ke 6 mengatur fov dari kamera untuk menyempit disekitar karakter. Baris ke 8 mengubah variable muncul hantu menjadi true, kemudian memberi sorot point light pada baris ke 9 pada karakter hantu yang diatur pada baris ke 10. Baris ke 11 hingga baris ke 13 berguna untuk melakukan random value untuk memunculkan hantu jenis apa yang keluar, apakah model menempel dibelakng atau muncul didepan karakter. Jika hasil random lolos pengecekan baris ke 17, maka baris perintah ke 18 akan mengecek lagi apakah posisi karakter sedang menghadap kekiri, jika iya maka baris ke 19 hingga baris ke 21 akan mengatur pergerakan hantu kekiri dengan cara mengurangi value x. Sedangkan baris ke 22 merupakan pengaturan pointLight pada hantu yang mengikuti karakter. Sedangkan baris perintah ke 24 hingga ke 28, merupakan mekanisme pergerakan hantu jika karakter sedang menghadap kekanan. Value x pada transform akan ditambah pada baris perintah ke 25. Sedangkan baris ke 32 hingga baris ke 45, merupakan baris perintah untuk memunculkan hantu yang berada dibelakang pemain. Untuk pergerakannya diatur pada baris ke 33 jika

pemain menghadap kekiri, maka baris perintah ke 34 hingga baris ke 36 mengatur posisi x agar berkurang. Sedangkan jika pemain menghadap kanan maka baris perintah ke 39 akan dijalankan. Pengaturan gerak hantu diatur pada baris ke 41 hingga baris ke 43, dimana posisi x akan ditambah. Dan yang terakhir yaitu baris ke 47 jika hasil random yang dihasilkan tidak memenuhi syarat penyaringan, maka jenis hantu yang keluar hanya diam ditempat blok pijakan berhantu dengan cara memberi value false pada baris perintah ke 49.

#### **5.4.2 Rintangan Duri**

Bagian ini menjelaskan lintangan duri yang muncul didalam game. Rintangan jenis ini memiliki 2 segmen program yaitu segmen program 5.23 Generate Rintangan Duri, dan 5.24 Pemain Terkena Duri. Penjelasan berada diakhir segment.

##### **Segmen Program 5.23 Generate Rintangan Duri**

```

01: public void PCG_Jebakan()
02: {
03:     int min = 0;
04:     int max = 10;
05:     int randomValue = Random.Range(min, max);
06:     if (randomValue % 2 != 0)
07:     {
08:         index = 0;
09:     }
10:     else
11:     {
12:         index = 1;
13:     }
14:     sp.sprite = image_jebakan[index];
15:     randomValue++;
16: }
```

Segmen Program 5.23 merupakan baris program yang digunakan untuk membuat lintangan duri pada platform level. Baris ke 1 merupakan penamaan fungsi untuk membuat platform duri. Baris ke 3 hingga baris ke 5 merupakan variable untuk merandom keputusan. Baris ke 6 merupakan pengecekan value variable random pada baris ke 3 hingga baris ke 5, jika lolos pengecekan maka index atau jenis duri akan dimunculkan sesuai dengan baris perintah ke 8. Baris

perintah ke 10 hingga ke 13 mengatur index atau jenis duri yang akan dimunculkan, dengan mengubah value index menjadi 1 pada baris ke 12. Kemudian baris perintah ke 14 memunculkan sprite duri sesuai dengan index yang didapat sebelumnya. Terakhir baris ke 15 membuat value random ditambah 1.

### **Segmen Program 5.24 Pemain Terkena Duri**

```
01: private void OnTriggerEnter2D(Collider2D collision)
02: {
03:     if (collision.gameObject.CompareTag("Player"))
04:     {
05:         penangkapEmosi.deteksi_emosi();
06:         karakter.GetComponent<Karakter>().
07:             kenaSerangan(gameObject);
08:         gameObject.SetActive(false);
09:     }
}
```

Segmen Program 5.24 merupakan baris program yang digunakan untuk mendeteksi apakah pemain terkena rintangan duri. Baris ke 3 merupakan pengecekan apakah duri mengalami collision dengan gameobject dengan tag “Player” yang sudah dipasang pada karakter. Jika iya maka perintah baris ke 5 hingga baris ke 7 akan dijalankan. Baris ke 5 menyimpan deteksi wajah pemain saat terkena rintangan duri. Baris ke 6 memberi atribut kenaSerangan pada karakter. Dan kemudian pada baris ke 7 gameobject duri akan dihilangkan jika telah diinjak karakter pemain.

### **5.4.3 Jumpscare Penampakan**

Bagian ini menjelaskan tentang blok jumpscare penampakan didalam game. Rintangan jenis ini memiliki 2 segmen program yaitu segmen program 5.25 Generate Rintangan Jumpscare Penampakan, dan 5.26 Pemain Menginjak Jumpscare Penampakan. Penjelasan setiap segmen akan dijabarkan setelah segmen program.

### **Segmen Program 5.25 Generate Rintangan Jumpscare Penampakan**

```
01: public void PCG_kuburan_sesajen()
02: {
```

### Segmen Program 5.25 (Lanjutan)

```

03:     int min = 0; // Nilai terkecil yang ingin Anda hasilkan
        (termasuk)
04:     int max = image_kuburan_sesajen.Length-1;
05:     int randomValue = Random.Range(min, max);
06:     sp.sprite = image_kuburan_sesajen[randomValue];
07:     randomValue++;
08: }
```

Segmen Program 5.25 merupakan baris program yang digunakan untuk membuat rintangan jumpscare penampakan pada platform level. Baris ke 1 merupakan penamaan fungsi untuk membuat platform jumpscare. Baris ke 3 hingga baris ke 5 merupakan variable untuk merandom bentuk sprites kuburan yang akan muncul pada level. Baris ke 6 digunakan mengenerate blok jumpscare penampakan berdasarkan random value yang didapat dari baris ke 3 hingga baris ke 5. Kemudian baris ke 7 digunakan untuk mengubah nilai random value.

### Segmen Program 5.26 Pemain Menginjak Jumpscare

```

01: private void OnTriggerEnter2D(Collider2D collision)
02: {
03:     if (collision.gameObject.CompareTag("Player") &&
        useItems_.GetComponent<glob_Items>().getPakaiGunting() ==
        false)
04:     {
05:         canvasJumpScare.GetComponent<CanvasJumpScare>().doJumpSc
        are();
06:         gameObject.GetComponent<BoxCollider2D>().enabled =
        false;
07:         status = "Aktif";
08:         pencatatEmosi.RecordData();
09:     }
10: }
```

Segmen Program 5.26 merupakan baris program yang mengecek apakah pemain menginjak blok jumpscare penampakan. Baris ke 3 merupakan pengecekan apakah blok jumpscare mengalami collision dengan gameobject dengan tag “Player” yang sudah dipasang pada karakter dan status player sedang tidak menggunakan item gunting. Jika lolos pengecekan, maka baris ke 5 dijalankan yaitu memainkan jumpscare berdasarkan canvas jumpscare. Kemudian baris ke 6 melakukan disable gameobject blok jumpscare ketika sudah diinjak.. Kemudian

memberi status telah aktif pada baris ke 7 guna diolah oleh pencatat emosi pada baris ke 8.

#### **5.4.4 Tempat Bersembunyi**

Bagian ini menjelaskan tempat bersembunyi yang sekaligus menjadi tempat yang aman bagi karakter pemain yang berbentuk semak bambu didalam game. Tempat sembunyi memiliki 2 segmen program yaitu segmen program 5.27 Generate Platform Bambu, dan 5.28 Pemain Bersembunyi. Penjelasan setiap segmen akan dijabarkan setelah segmen program.

##### **Segmen Program 5.27 Generate Platform Bambu**

```

01: public void PCG_Bambu()
02: {
03:     int min = 0;
04:     int max = 10;
05:     int randomValue = Random.Range(min, max);
06:     if (randomValue % 2 != 0)
07:     {
08:         index = 0;
09:     }
10:     else
11:     {
12:         index = 1;
13:     }
14:     sp.sprite = image_bambu[index];
15:     randomValue++;
16: }
```

Segmen Program 5.27 merupakan baris program yang digunakan untuk membuat tempat persembunyian atau tempat aman pada platform level. Baris ke 1 merupakan penamaan fungsi untuk membuat platform tempat sembunyi. Baris ke 6 merupakan pengecekan value variable random pada baris ke 3 hingga baris ke 5, jika lolos pengecekan maka index atau jenis duri akan dimunculkan sesuai dengan baris perintah ke 8. Baris perintah ke 10 hingga ke 13 mengatur index atau jenis duri yang akan dimunculkan, dengan mengubah value index menjadi 1 pada baris ke 12. Kemudian baris perintah ke 14 memunculkan sprite bambu sesuai dengan index yang didapat sebelumnya. Terakhir baris ke 15 membuat value random ditambah 1.

### Segmen Program 5.28 Pemain Bersembunyi

```

01: if (masuk == true)
02: {
03:     if (index == 1)
04:     {
05:         if (karakter.GetComponent<Karakter>().jongkok == true){
06:             karakter.GetComponent<Karakter>().setStatusHide(true);
07:         }
08:         else{
09:             karakter.GetComponent<Karakter>().setStatusHide(false);
10:         }
11:     else
12:     {
13:         karakter.GetComponent<Karakter>().setStatusHide(true);
14:     }
15: else
16: {
17:     karakter.GetComponent<Karakter>().setStatusHide(false);
18: }

```

Segmen Program 5.x merupakan baris program yang mengecek apakah pemain statusnya sedang masuk didalam semak bambu. Baris ke 1 mendeteksi pemain dengan cara mengecek status trigger pemain apakah sedang true. Jika true, maka masuk di baris ke 3 yaitu pengecekan tipe bambu, jika bambu yang muncul dideteksi bambu pendek maka akan masuk baris perintah ke 5. Baris perintah ke 5 mengecek apakah karakter sedang jongkok, jika iya maka baris ke 6 mengubah statusHide karakter menjadi true. Jika tidak maka statusHide karakter diganti menjadi false pada baris ke 9. Kemudian pada baris ke 14 akan dijalankan jika bambu yang muncul berupa bambu panjang, jika karakter pemain mentrigger bambu maka statusHide pemain akan dijadikan true. Jika karakter statusnya tidak lolos pengecekan pada baris ke 1 maka statusHide karakter pada baris ke 18 akan diubah menjadi false.

## **BAB VI**

### **UJI COBA**

Bab ini akan menjelaskan tentang uji coba yang dilakukan selama pengerjaan Tugas Akhir ini. Uji coba yang dilakukan meliputi 3 metode uji coba yaitu White Box Testing, Black Box Testing, dan Kuesioner yang dijabarkan pertanyaan berserta jawabannya.

#### **6.1 White Box Testing**

White box testing, juga dikenal sebagai testing struktural, adalah metode pengujian perangkat lunak yang melibatkan pemeriksaan dan evaluasi struktur internal dari kode sumber. Pendekatan ini memerlukan pemahaman yang mendalam terhadap logika dan implementasi program yang diuji. White box testing yang dilakukan selama pengerjaan Tugas Akhir ini akan diuraikan dalam subbab di bawah ini.

##### **6.1.1 White Box Testing Karakter**

Bagian ini menjelaskan testing white box yang dilakukan pada karakter pemain. Terdapat beberapa testing yang dilakukan. Penjelasannya dijabarkan dalam Tabel 6.1.

**Tabel 6.1**  
**Tabel White Box Testing Karakter**

No	Keterangan	Skenario Testing	Target	Status
1	Gerak Kiri Kanan Karakter	Menekan tombol ‘A’ atau ‘D’	Value x position dikurang saat menekan tombol ‘A’. Value x position ditambah saat menekan tombol ‘D’. Play anims jalan. Flip jika tombol ‘A’ ditekan.	Berjalan sesuai target

**Tabel 6.1  
(Lanjutan)**

No	Keterangan	Skenario Testing	Target	Status
2	Karakter Merangkak	Menekan tombol ‘S’	Masuk kedalam kondisi modeJongkok(). Collider diubah menjadi jongkok, Play anims jongkok.	Berjalan sesuai target
3	Karakter Melompat	Menekan tombol ‘Space’	Masuk kedalam kondisi modeLompat(). Value y pos dikurangi kemudian ditambah lagi sesuai dengan gravity. Play anims lompat.	Berjalan sesuai target
4	Karakter Lari	Menekan tombol ‘Shift’	Value x position ditambah sejumlah dxLari saat menekan tombol ‘Shift’.	Berjalan sesuai target
5	Karakter Memukul	Menekan tombol ‘O’	Masuk kedalam kondisi modePukul(), status isAttack == true. Play anims memukul.	Berjalan sesuai target
6	Karakter Kena Serang	Tidak sembunyi saat terkejar hantu	isHurting == true, healthPoints -= 1, beri efek sprite merah. Play anims karakter terkena serang.	Berjalan sesuai target

### 6.1.2 White Box Testing Rintangan Hantu

Bagian ini menjelaskan testing white box yang dilakukan pada rintangan hantu. Terdapat beberapa testing yang dilakukan. Penjelasannya dijabarkan dalam Tabel 6.2.

**Tabel 6.2**  
**Tabel White Box Testing Rintangan Hantu**

No	Keterangan	Skenario Testing	Target	Status
1	Generate Blok Hantu	Memberi pembeda warna blok.	buatObject(jalan_hantu_single, x, tinggi, layer_jalan) tereksekusi.	Berjalan sesuai target
2	Cek Penginjakan	Melewati setiap blok yang tergenerate.	Collision.gameObject. CompareTag("Player") == true, sudahDipijak == true.	Berjalan sesuai target
3	Memunculkan hantu	Melewati setiap blok yang tergenerate	<Hantu_single>().startMuncul = true, karakter_hantu.SetActive(true);	Berjalan sesuai target

### 6.1.3 White Box Testing Rintangan Duri

Bagian ini menjelaskan testing white box yang dilakukan pada rintangan duri. Terdapat beberapa testing yang dilakukan. Penjelasannya dijabarkan dalam Tabel 6.3.

**Tabel 6.3**  
**Tabel White Box Testing Rintangan Duri**

No	Keterangan	Skenario Testing	Target	Status
1	Generate Rintangan Duri	Melewati setiap blok yang tergenerate.	Function PCG_Jebakan() dipanggil saat generate level. Sp.sprite=image_jebakan [index] berjalan.	Berjalan sesuai target

**Tabel 6.3  
(Lanjutan)**

No	Keterangan	Skenario Testing	Target	Status
2	Pemain Terkena Duri	Pemain menginjak duri secara sengaja	gameObject.setActive=false, collision.gameObject. CompareTag("Player") == true, karakter.kenaSerangan berjalan	Berjalan sesuai target

#### 6.1.4 White Box Testing Tempat Bersembunyi

Bagian ini menjelaskan testing white box yang dilakukan pada spawn tempat bersembunyi. Terdapat beberapa testing yang dilakukan. Penjelasannya dijabarkan dalam Tabel 6.4.

**Tabel 6.4  
Tabel White Box Testing Tempat Bersembunyi**

No	Keterangan	Skenario Testing	Target	Status
1	Generate Semak Bambu	Melewati setiap blok yang yang tergenerate.	Function PCG_Bambu() dipanggil saat generate level. Sp.sprite=image_bambu [index] berjalan.	Berjalan sesuai target
2	Pemain Bersembunyi	Pemain bersembunyi dalam semak saat dikejar hantu.	Saat karakter didalam collider, karakter.setStatusHide(true), jika tidak maka karakter.setStatusHide(false)	Berjalan sesuai target

### 6.1.5 White Box Testing Jumpscare Penampakan

Bagian ini menjelaskan testing white box yang dilakukan pada jumpscare penampakan. Terdapat beberapa testing yang dilakukan. Penjelasannya dijabarkan dalam Tabel 6.5.

**Tabel 6.5**  
**Tabel White Box Testing Jumpscare Penampakan**

No	Keterangan	Skenario Testing	Target	Status
1	Generate Rintangan Jumpscare Penampakan	Melewati setiap blok yang tergenerate.	Function PCG_kuburan_sesajen dipanggil saat generate level. Sp.sprite=image_kuburan_sesajen[randValue] berjalan.	Berjalan sesuai target
2	Pemain Menginjak Jumpscare Penampakan	Melewati setiap blok yang tergenerate.	doJumpscare() berhasil dipanggil saat collision.gameObject. CompareTag("Player") == true, dan canvasJumpscare muncul.	Berjalan sesuai target

### 6.1.6 White Box Testing Pencatat Log

Bagian ini menjelaskan testing white box yang dilakukan pada sistem pencatat log. Terdapat beberapa testing yang dilakukan. Penjelasannya dijabarkan dalam Tabel 6.6.

**Tabel 6.6**  
**Tabel White Box Testing Pencatat Log**

No	Keterangan	Skenario Testing	Target	Status
1	Pencatatan Emosi	Membuat raut wajah	Value_emosi mendapat suplay data dari EmotionsManager.Emotions	Berjalan sesuai target

**Tabel 6.6**  
**(Lanjutan)**

No	Keterangan	Skenario Testing	Target	Status
		takut, netral, dan marah.	library moodme. Dan variabel emotion dapat diisi label “emosi” dan return isi.	
2	Pencatatan Jenis Rintangan	Melewati masing-masing jenis rintangan dan didebug	Ketika jenis rintangan tertentu ditrigger, akan mengembalikan data jenis_rintangannya, dan berhasil mengisi jenis obstacle berserta status rintangan gagal atau sukses.	Berjalan sesuai target

### 6.1.7 White Box Testing Pembaca Log

Bagian ini menjelaskan testing white box yang dilakukan pada sistem pembaca log. Terdapat beberapa testing yang dilakukan. Penjelasannya dijabarkan dalam Tabel 6.7.

**Tabel 6.7**  
**Tabel White Box Testing Pembaca Log**

No	Keterangan	Skenario Testing	Target	Status
1	Pengecekan File Log	Membuat file log kosong dan menghapus file log dari direktori.	Function LogFilePath berhasil memberi return logFilePath Ketika ada file output.json, jika tidak ada maka akan muncul error pada console debug.	Berjalan sesuai target

**Tabel 6.7**  
**(Lanjutan)**

No	Keterangan	Skenario Testing	Target	Status
2	Membaca Isi File Json	File json ditempatkan didirektori seharusnya, namun isinya kosong, salah format, dan berisi benar.	LogWrapper.FromJson<LogWrapper>(jsonContent) mendapat supply data, kemudian jika tidak terpenuhi kondisinya, maka akan muncul error pada console debug sesuai dengan pesan penyebab error.	Berjalan sesuai target
3	Membaca Json sesuai dengan parameter yang dibutuhkan DDA	Menyortir data json menjadi supply data untuk DDA	foreach(PembacaLog.LogEntry entry in logEntries) berhasil melakukan looping data dan menyortir entry.obstacle json dan melakukan penyortiran entry.status perkategori rintangan.	Berjalan sesuai target

#### 6.1.8 White Box Testing Penentuan Kategori Pemain

Bagian ini menjelaskan testing white box yang dilakukan pada sistem penentuan kategori pemain melalui skoring. Terdapat beberapa testing yang dilakukan. Penjelasannya dijabarkan dalam Tabel 6.8.

**Tabel 6.8**  
**Tabel White Box Testing Penentuan Kategori Pemain**

No	Keterangan	Skenario Testing	Target	Status
1	Mendapatkan jumlahJalan pada level sebelumnya	Membuat debug log berisi variabel global ctr_pjrintangan	JumlahRintangan pada debug log level sebelumnya benar dan ditampung, pada variabel jumlahJalanRintangan.	Berjalan sesuai target
2	Mendapatkan value ctr_rintangan_sukses, gagal, dan tidak_injak	Melakukan debug log pada ke 3 variabel tersebut	Mendapatkan nilai yang benar dengan mencocokkan counter sukses dan gagal melewati rintangan serta jumlah rintangan yang tidak diinjak.	Berjalan sesuai target
3	Melakukan perhitungan performa pemain dan label pemain	Simulasi variabel pemain dengan angka statik, kemudian akan didebug log satu per satu kondisi pemain dilabeli “pemula”, “normal” dan “mahir”.	Semua kondisi untuk pelabelan berjalan sesuai dengan simulasi variabel statik yang menyerupai pemain menyelesaikan 1 level. Kemudian hasil debug kategori_pemain sesuai dengan kondisi percabangan.	Berjalan sesuai target
4	Melakukan perhitungan emosi pemain	Melakukan debug log pada value emosi yang terdeteksi oleh moodme, kemudian membandingkan hasil emosi yang lebih dominan.	Konsol log mengeluarkan nilai emosi pemain. Kemudian label emosi yang telah diproses divalidasi apakah emosinya dominan ‘takut’, ‘normal’ atau ‘marah’.	Berjalan sesuai target
5	Melakukan perbandingan label dengan pengatur jumlah rintangan	Init_rintangan diubah jumlahnya berdasarkan value wc_rintangan yang sesuai dengan label.	Value init_rintangan berubah sesuai dengan pengaturan weight ceiling DDA.	Berjalan sesuai target

Pada tahap ini, dilakukan pengujian berkali-kali untuk memastikan bahwa scoring yang dibuat tidak memiliki kendala. Selama pengetesan setidaknya terjadi error perhitungan sebanyak 20 kali yang berhasil diperbaiki selama testing kode sumber.

## 6.2 Black Box Testing

Black box testing adalah metode pengujian perangkat lunak tanpa memperhatikan struktur internal atau logika kode. Pendekatan ini lebih fokus pada input dan output yang dihasilkan oleh sistem tanpa mempertimbangkan bagaimana perangkat lunak mencapai hasil tersebut. Black box testing yang dilakukan selama pengerjaan Tugas Akhir ini akan diuraikan dalam subbab di bawah ini.

### 6.2.1 Black Box Testing Karakter

Bagian ini menjelaskan testing black box yang dilakukan pada karakter pemain. Terdapat 5 macam testing yang dilakukan. Testing yang dilakukan meliputi testing gerak karakter, karakter merangkak, karakter melompat, karakter memukul, dan karakter kena serang. Penjelasannya dijabarkan dalam Tabel 6.9.

**Tabel 6.9**  
**Tabel Black Box Testing Karakter**

No	Keterangan	Skenario Testing	Target	Status
1	Gerak Kiri Kanan Karakter	Menekan tombol ‘A’ atau ‘D’	Karakter akan berjalan kekiri dan kekanan saat pemain menekan tombol ‘A’ atau ‘D’	Berjalan sesuai target
2	Karakter Merangkak	Menekan tombol ‘S’	Karakter akan bergerak merangkak	Berjalan sesuai target
3	Karakter Melompat	Menekan tombol ‘Space’	Karakter akan melompat	Berjalan sesuai target
4	Karakter Lari	Menekan tombol ‘Shift’	Karakter akan berlari	Berjalan sesuai target

**Tabel 6.9  
(Lanjutan)**

No	Keterangan	Skenario Testing	Target	Status
5	Karakter Memukul	Menekan tombol ‘O’	Karakter akan melakukan aksi memukul.	Berjalan sesuai target
6	Karakter Kena Serang	Tidak sembunyi saat terkejar hantu	Nyawa karakter akan berkurang 1	Berjalan sesuai target

### 6.2.2 Black Box Rintangan Hantu

Bagian ini menjelaskan testing black box yang dilakukan pada rintangan hantu. Terdapat 3 macam testing yang dilakukan. Testing yang dilakukan meliputi testing generate pijakan hantu, cek penginjakan, dan memunculkan hantu. Penjelasannya dijabarkan dalam Tabel 6.10.

**Tabel 6.10  
Tabel Black Box Testing Hantu**

No	Keterangan	Skenario Testing	Target	Status
1	Generate Blok Hantu	Melewati setiap blok.	Terdapat blok spawn hantu sejumlah X pada level yang muncul.	Berjalan sesuai target
2	Cek Penginjakan	Melewati setiap blok.	Berhasil mendeteksi pemain telah melewati blok dengan memunculkan hantu.	Berjalan sesuai target
3	Memunculkan hantu	Melewati setiap blok.	Animasi hantu berhasil digenerate dan dimunculkan saat pemain telah menginjak blok hantu.	Berjalan sesuai target

### 6.2.3 Black Box Rintangan Duri

Bagian ini menjelaskan testing black box yang dilakukan pada rintangan hantu. Testing yang dilakukan meliputi testing generate rintangan duri, dan pemain terkena duri. Penjelasannya dijabarkan dalam Tabel 6.11.

**Tabel 6.11**  
**Tabel Black Box Testing Rintangan Duri**

No	Keterangan	Skenario Testing	Target	Status
1	Generate Rintangan Duri	Melewati setiap blok yang tergenerate.	Terdapat blok dengan rintangan duri diatasnya sejumlah X pada level yang muncul.	Berjalan sesuai target
2	Pemain Terkena Duri	Pemain menginjak duri secara sengaja	Nyawa pemain berhasil berkurang sebanyak 1 dan rintangan duri menghilang setelah diinjak.	Berjalan sesuai target

### 6.2.4 Black Box Rintangan Kabut

Bagian ini menjelaskan testing black box yang dilakukan pada rintangan kabut. Terdapat 2 macam testing yang dilakukan. Testing yang dilakukan meliputi testing generate rintangan kabut, dan pemain menginjak blok rintangan kabut. Penjelasannya dijabarkan dalam Tabel 6.12.

**Tabel 6.12**  
**Tabel Black Box Testing Kabut**

No	Keterangan	Skenario Testing	Target	Status
1	Generate Rintangan Kabut	Melewati setiap blok yang tergenerate.	Terdapat blok kabut suara sejumlah X pada level yang muncul.	Berjalan sesuai target

**Tabel 6.12  
(Lanjutan)**

No	Keterangan	Skenario Testing	Target	Status
2	Pemain Menginjak Blok Kabut	Melewati setiap blok yang tergenerate.	Intensitas kabut dilayar bertambah tebal.	Berjalan sesuai target

### 6.2.5 Black Box Testing Item

Bagian ini menjelaskan testing black box yang dilakukan pada spawn item. Terdapat 3 macam testing yang dilakukan. Testing yang dilakukan meliputi testing generate item, mengambil item dan menggunakan item. Penjelasannya dijabarkan dalam Tabel 6.13.

**Tabel 6.13  
Tabel Black Box Testing Spawn Item**

No	Keterangan	Skenario Testing	Target	Status
1	Generate Item Pada Map	Melewati setiap blok yang tergenerate.	Terdapat sejumlah item pada level yang digenerate.	Berjalan sesuai target
2	Pemain Mengambil Item	Saat berdiri diatas item, karakter menekan tombol ‘E’.	Item yang terdapat pada level hilang 1 dan inventory item pemain bertambah 1.	Berjalan sesuai target
3	Pemain Menggunakan Item	Pada Rintangan X Pemain Menggunakan Item X.	Rintangan tertentu menjadi tidak efektif terhadap pemain dan jumlah item di inventory pemain berkurang 1.	Berjalan sesuai target

### 6.2.6 Black Box Testing Jumpscare Penampakan

Bagian ini menjelaskan testing black box yang dilakukan pada rintangan jumpscare penampakan. Terdapat 2 macam testing yang dilakukan. Testing yang dilakukan meliputi testing generate blok jumpscare penampakan, dan pemain menginjak jumpscare penampakan. Penjelasannya dijabarkan dalam Tabel 6.14.

**Tabel 6.14**  
**Tabel Black Box Testing Jumpscare Penampakan**

No	Keterangan	Skenario Testing	Target	Status
1	Generate Rintangan Jumpscare Penampakan	Melewati setiap blok yang tergenerate.	Terdapat blok jumpscare penampakan sejumlah X pada level yang muncul.	Berjalan sesuai target
2	Pemain Menginjak Jumpscare Penampakan	Melewati setiap blok yang tergenerate.	Muncul canvas hantu mengagetkan serta sound effect mengagetkan saat pemain menginjak blok.	Berjalan sesuai target

### 6.2.7 Black Box Testing Tempat Bersembunyi

Bagian ini menjelaskan testing black box yang dilakukan pada tempat bersembunyi. Terdapat 2 macam testing yang dilakukan. Testing yang dilakukan meliputi testing generate semak bambu, dan pemain bersembunyi. Penjelasannya dijabarkan dalam Tabel 6.15.

**Tabel 6.15**  
**Tabel Black Box Testing Tempat Bersembunyi**

No	Keterangan	Skenario Testing	Target	Status
1	Generate Semak Bambu	Melewati setiap blok pada level.	Terdapat semak bambu sejumlah X pada level yang muncul.	Berjalan sesuai target

**Tabel 6.15  
(Lanjutan)**

No	Keterangan	Skenario Testing	Target	Status
2	Pemain Bersembunyi	Pemain bersembunyi dalam semak saat dikejar hantu.	Saat karakter dikejar hantu dan bersembunyi didalam semak, jumlah nyawanya tetap tidak berkurang.	Berjalan sesuai target

#### 6.2.8 Black Box Testing Pencatat Log

Bagian ini menjelaskan testing black box yang dilakukan pada sistem pencatat log. Terdapat beberapa testing yang dilakukan. Penjelasannya dijabarkan dalam Tabel 6.16.

**Tabel 6.16  
Tabel Black Box Testing Pencatat Log**

No	Keterangan	Skenario Testing	Target	Status
1	Pencatatan Emosi	Membuat raut wajah takut, netral, dan marah.	Slider emosi pada game bergerak menunjukan persentase emosi yang dideteksi.	Berjalan sesuai target
2	Pencatatan Jenis Rintangan	Melewati masing-masing jenis rintangan	Ketika level selesai, nilai performa pemain ditampilkan. Dapat dibandingan dengan isi file log yang digenerate.	Berjalan sesuai target

### 6.2.9 Black Box Testing Pembaca Log

Bagian ini menjelaskan testing black box yang dilakukan pada sistem pembaca log. Terdapat beberapa testing yang dilakukan. Penjelasannya dijabarkan dalam Tabel 6.17.

**Tabel 6.17**  
**Tabel Black Box Testing Pembaca Log**

No	Keterangan	Skenario Testing	Target	Status
1	Pengecekan File Log	Membuat file log kosong dan menghapus file log dari direktori.	Jika file log tidak ada didalam direktori, maka game membuat file log baru.	Berjalan sesuai target
2	Membaca Isi File Json	File json ditempatkan didirektori seharusnya, namun isinya kosong, salah format, dan berisi benar.	File log tetap dapat terbaca oleh game dan tidak berdampak apapun terhadap jalannya permainan.	Berjalan sesuai target
3	Membaca Json sesuai dengan parameter yang dibutuhkan DDA	Menyortir data json menjadi supply data untuk DDA	Panel report pada ending menampilkan performa pemain dengan benar.	Berjalan sesuai target

### **6.2.10 Black Box Testing Penentuan Kategori Pemain**

Bagian ini menjelaskan testing black box yang dilakukan pada sistem penentuan kategori pemain melalui skoring. Terdapat beberapa testing yang dilakukan. Penjelasannya dijabarkan dalam Tabel 6.18.

**Tabel 6.18**  
**Tabel Black Box Testing Penentuan Kategori Pemain**

No	Keterangan	Skenario Testing	Target	Status
1	Mendapatkan nilai jumlahJalan pada level sebelumnya	Menyelesaikan level yang telah diberikan.	Jumlah jalan dan rintangan pada panel report sesuai dengan level yang diselesaikan.	Berjalan sesuai target
2	Mendapatkan skor performa ‘Pemula’	Menyelesaikan level yang telah diberikan dengan ceroboh dan membuat ekspresi “Takut”.	Mendapatkan nilai ‘Pemula’ pada panel report dan jenis rintangan yang digenerate pada level berikutnya dipermudah.	Berjalan sesuai target
3	Mendapatkan skor performa ‘Mahir’	Menyelesaikan level yang telah diberikan dengan sengaja menghindari sebagian besar rintangan dan membuat ekspresi “Netral”.	Mendapatkan nilai ‘Mahir’ pada panel report dan jumlah jenis rintangan yang digenerate pada level berikutnya dipertahankan, dan akan diturunkan saat mendapat skor Normal.	Berjalan sesuai target

**Tabel 6.18  
(Lanjutan)**

No	Keterangan	Skenario Testing	Target	Status
4	Mendapatkan skor performa ‘Normal’	Menyelesaikan level dengan sengaja mengenai sebagian rintangan.	Mendapatkan nilai ‘Normal’ pada panel report dan jumlah jenis rintangan berubah lebih sulit.	Berjalan sesuai target
5	Mendapatkan report rata-rata emosi pada jenis rintangan tertentu	Menyelesaikan level dengan rintangan yang ada.	Rata-rata emosi yang terdeteksi pada jenis rintangan tertentu berhasil ditampilkan pada panel report.	Berjalan sesuai target

### 6.3 Kuesioner

Pada tahap ini metode yang digunakan yaitu kuesioner yang dirancang menggunakan UEQ. Game dimainkan oleh 30 responden yang memenuhi kriteria yaitu pernah memainkan game horror dan percaya akan hantu atau cerita mistis. Kemudian setelah bermain, responden diminta untuk mengisi form yang telah dirancang metode evaluasi UEQ dipilih karena kekuatan dan kelemahan dari game yang dibuat dapat diidentifikasi. Dan yang paling penting yaitu dengan UEQ pengembang akan mendapatkan masukan yang berharga untuk meningkatkan kualitas pengalaman bermain.

UEQ sendiri merupakan alat evaluasi yang dapat diterapkan dalam desain pengalaman pengguna untuk menilai pandangan pengguna terhadap suatu produk atau layanan. Salah satu keunggulan utamanya adalah kemampuannya untuk memberikan gambaran menyeluruh tentang beragam aspek pengalaman pengguna. Proses kerjanya melibatkan pengumpulan data melalui kuesioner yang dirancang secara khusus, yang kemudian dianalisis untuk mengenali titik kuat dan kelemahan suatu produk atau layanan serta memberikan panduan untuk perbaikan lebih lanjut.

	1	2	3	4	5	6	7	
menyusahkan	<input type="radio"/>	menyenangkan 1						
tak dapat dipahami	<input type="radio"/>	dapat dipahami 2						
kreatif	<input type="radio"/>	monoton 3						
mudah dipelajari	<input type="radio"/>	sulit dipelajari 4						
bermanfaat	<input type="radio"/>	kurang bermanfaat 5						
membosankan	<input type="radio"/>	mengasyikkan 6						
tidak menarik	<input type="radio"/>	menarik 7						
tidak dapat diprediksi	<input type="radio"/>	dapat diprediksi 8						
cepat	<input type="radio"/>	lambat 9						
berdaya cipta	<input type="radio"/>	konvensional 10						
menghalangi	<input type="radio"/>	mendukung 11						
baik	<input type="radio"/>	buruk 12						
rumit	<input type="radio"/>	sederhana 13						
tidak disukai	<input type="radio"/>	menggembirakan 14						
lazim	<input type="radio"/>	terdepan 15						
tidak nyaman	<input type="radio"/>	nyaman 16						
aman	<input type="radio"/>	tidak aman 17						
memotivasi	<input type="radio"/>	tidak memotivasi 18						
memenuhi ekspektasi	<input type="radio"/>	tidak memenuhi ekspektasi 19						
tidak efisien	<input type="radio"/>	efisien 20						
jelas	<input type="radio"/>	membingungkan 21						
tidak praktis	<input type="radio"/>	praktis 22						
terorganisasi	<input type="radio"/>	berantakan 23						
atraktif	<input type="radio"/>	tidak atraktif 24						
ramah pengguna	<input type="radio"/>	tidak ramah pengguna 25						
konservatif	<input type="radio"/>	inovatif 26						

**Gambar 6.1  
26 Skala Pertanyaan UEQ**

Kuesioner disusun dengan mengikuti pedoman 26 item pertanyaan yang mewakili 6 skala pada panduan UEQ yaitu daya tarik, kejelasan, efisiensi, ketepatan, stimulasi dan kebaruan seperti pada gambar 6.1 diatas<sup>19</sup>. Pertanyaan yang dibuat melalui google form akan dijabarkan pada tabel dibawah ini.

**Tabel 6.19  
Item Pertanyaan**

No	Pertanyaan	Skala
1	Apakah rancangan rintangan yang dibuat membuat game menjadi menyenangkan untuk dimainkan?	Daya tarik
2	Seberapa mudah anda memahami konsep dan tujuan dari pengaturan rintangan yang dibuat?	Kejelasan

<sup>19</sup> Hinderks, Andreas; Schrepp, Martin; Domínguez Mayo; Francisco José; Escalona, M.J.; Thomaschewski, Jörg. *Developing a UX KPI based on the User Experience Questionnaire*. Computer Standards & Interfaces, 2019.

**Tabel 6.19  
(Lanjutan)**

3	Seberapa kreatif menurut anda pengaturan tingkat kesulitan?	Kebaruan
4	Seberapa cepat anda dapat menguasai keterampilan yang diperlukan untuk mengatasi rintangan?	Kejelasan
5	Seberapa bermanfaatkah pengaturan rintangan dalam game?	Stimulasi
6	Seberapa mengasyikkan pengalaman bermain game ini?	Stimulasi
7	Seberapa menarik game dengan pengaturan rintangan ini?	Stimulasi
8	Seberapa konsisten tingkat kesulitan disesuaikan?	Ketepatan
9	Seberapa cepat tingkat kesulitan level berikutnya berubah?	Efisiensi
10	Menurut anda penggunaan ekspresi wajah sebagai pengatur tingkat kesulitan mengadopsi ide-ide baru?	Kebaruan
11	Seberapa efektif pengaturan rintangan level berikutnya?	Ketepatan
12	Seberapa baik pengaturan rintangan dalam level berikutnya?	Daya tarik
13	Seberapa rumit pengaturan rintangan dalam level berikutnya?	Kejelasan
14	Seberapa menggembirakan pengalaman bermain yang anda dapatkan dengan adanya pengaturan rintangan level berikutnya?	Daya tarik
15	Seberapa terdepan ide pengaturan rintangan level yang diterapkan?	Kebaruan
16	Seberapa nyaman pengalaman bermain yang anda dapatkan dengan adanya pengaturan rintangan dalam level berikutnya?	Daya tarik
17	Seberapa aman konsep pengolahan data emosil pada game?	Ketepatan
18	Seberapa besar pengaruh pengaturan rintangan memotivasi anda untuk menyelesaikan permainan?	Stimulasi
19	Sejauh mana pengalaman bermain level berikutnya memenuhi ekspektasi anda?	Ketepatan
20	Seberapa efisien rancangan rintangan level berikutnya?	Efisiensi
21	Sejauh mana tujuan perancangan rintangan yang dibentuk dengan sedemikian rupa dapat dipahami dengan jelas oleh anda sebagai pemain?	Kejelasan
22	Seberapa praktis menurut anda persiapan dalam memainkan game ini?	Efisiensi
23	Seberapa terorganisasi menurut anda alur game yang dibuat?	Efisiensi
24	Seberapa atraktif menurut anda fitur-fitur dalam permainan ini?	Daya tarik
25	Seberapa ramah pengguna menurut anda keseluruhan game ini?	Daya tarik
26	Seberapa inovatif penerapan konsep penyesuaian berdasarkan emosi?	Kebaruan

Kemudian hasil yang diperoleh akan diolah menggunakan data analyst UEQ untuk mendapatkan hasil benchmark yang digunakan sebagai acuan perbaikan

desain game kedepannya agar game yang dibuat mendapat user experience yang lebih baik. Hasil dari jawaban item pertanyaan berupa angka ber-range 1 hingga 7 yang harus dimasukkan kedalam sheet excel data analyst UEQ. Hasil pengolahan data melalui sheet data analyst berupa mean, variance, dan standar deviasi dari data kuesioner yang perhitungannya berdasarkan rumus yang telah dibuat oleh tim UEQ.

**Tabel 6.20**  
**Hasil Mean, Variance dan Standar Deviasi**

No	Mean	Variance	Std. Dev.	Left	Right	Scale
1	1,6	0,9	1,0	menyusahkan	menyenangkan	Daya tarik
2	1,6	1,5	1,2	tak dapat dipahami	dapat dipahami	Kejelasan
3	1,3	1,8	1,3	kreatif	monoton	Kebaruan
4	1,5	1,7	1,3	mudah dipelajari	sulit dipelajari	Kejelasan
5	1,8	1,3	1,1	bermanfaat	kurang bermanfaat	Stimulasi
6	1,6	1,1	1,0	membosankan	mengasyikkan	Stimulasi
7	1,8	0,6	0,8	tidak menarik	menarik	Stimulasi
8	0,7	2,7	1,6	tak dapat diprediksi	dapat diprediksi	Ketepatan
9	1,2	1,2	1,1	cepat	lambat	Efisiensi
10	1,3	1,8	1,3	berdaya cipta	konvensional	Kebaruan
11	1,5	0,7	0,9	menghalangi	mendukung	Ketepatan
12	1,7	1,0	1,0	baik	buruk	Daya tarik
13	-0,1	2,6	1,6	rumit	sederhana	Kejelasan
14	1,3	1,1	1,0	tidak disukai	menggembirakan	Daya tarik
15	1,5	0,7	0,9	lazim	terdepan	Kebaruan
16	1,3	0,9	0,9	tidak nyaman	nyaman	Daya tarik
17	1,6	1,2	1,1	aman	tidak aman	Ketepatan
18	1,7	1,3	1,1	memotivasi	tidak memotivasi	Stimulasi
19	1,5	1,2	1,1	memenuhi ekspektasi	tidak memenuhi ekspektasi	Ketepatan
20	1,4	1,1	1,0	tidak efisien	efisien	Efisiensi
21	1,6	1,6	1,3	jelas	membingungkan	Kejelasan
22	1,4	1,4	1,2	tidak praktis	praktis	Efisiensi
23	1,9	0,6	0,8	terorganisasi	berantakan	Efisiensi
24	1,7	1,3	1,1	atraktif	tidak atraktif	Daya tarik
25	1,9	1,0	1,0	ramah pengguna	tidak ramah pengguna	Daya tarik
26	1,8	0,5	0,7	konservatif	inovatif	Kebaruan

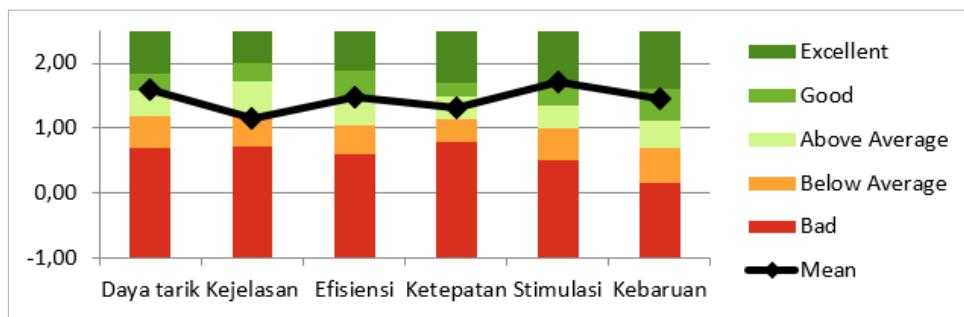
Nilai rata-rata  $>0.8$  menunjukkan evaluasi positif (panah ke atas), sementara nilai rata-rata  $<-0.8$  menunjukkan evaluasi negatif (panah ke bawah) dan jika nilai rata-rata berada di antara nilai positif dan negatif, akan ditandai dengan panah ke kanan. Kemudian angka evaluasi per item pertanyaan yang didapat, langkah selanjutnya yaitu mencari mean pada setiap skala yang dibuat sehingga akan ditemukan nilai mean dan variance yang didapat pada masing-masing 6 skala yang dievaluasi. Berikut merupakan hasil dari 6 skala yang dievaluasi berdasarkan data yang diperoleh.

**Tabel 6.21  
Hasil Mean dan Variance 6 Skala UEQ**

<b>UEQ Scales (Mean and Variance)</b>		
Daya Tarik	1,589	0,42
Kejelasan	1,142	0,92
Efisiensi	1,467	0,43
Ketepatan	1,317	0,54
Stimulasi	1,708	0,63
Kebaruan	1,450	0,42

Hasil yang didapat pada seluruh skala mendapatkan nilai positif. Skala daya tarik mendapat nilai positif dengan mean atau rata-rata 1,589. Skala kejelasan mendapat nilai positif dengan mean atau rata-rata 1,142. Skala efisiensi mendapat nilai positif dengan mean atau rata-rata 1,467. Skala ketepatan mendapat nilai positif dengan mean atau rata-rata 1,317. Skala stimulasi mendapat nilai positif dengan mean atau rata-rata 1,708. Skala kebaruan mendapat nilai positif dengan mean atau rata-rata 1,450.

Setelah pengolahan data selesai dilakukan, untuk memperoleh feedback yang diperlukan untuk pengembangan rancangan rintangan atau pengembangan game kedepannya pada tahapan selanjutnya akan dilakukan benchmark UEQ yang berdasarkan dari data 21175 orang dari 467 studi produk yang telah menggunakan UEQ. Data yang diperoleh akan dijabarkan dibawah.



**Gambar 6.2**  
**Grafik Hasil Benchmark UEQ**

Berdasarkan hasil benchmark yang dilakukan, nilai yang didapat pada skala daya tarik dan skala kebaruan mendapatkan nilai benchmark ‘Baik’. Nilai yang didapat pada skala efisiensi dan ketepatan mendapatkan benchmark ‘Diatas rata-rata’. Nilai “Luar biasa” didapat pada skala stimulasi. Dan skala kejelasan mendapatkan nilai “Dibawah rata-rata”. Berdasarkan data diatas, dapat disimpulkan bahwa pada skala kejelasan perlu perbaikan lebih lanjut untuk meningkatkan experience bermain game yang dikembangkan.

**Tabel 6.22**  
**Hasil Benchmark UEQ**

Scale	Mean	Comparisson to benchmark	Interpretation
Daya Tarik	1,59	Good	10% of results better, 75% of results worse
Kejelasan	1,14	Below Average	50% of results better, 25% of results worse
Efisiensi	1,47	Above Average	25% of results better, 50% of results worse
Ketepatan	1,32	Above Average	25% of results better, 50% of results worse
Stimulasi	1,71	Excellent	In the range of the 10% best results
Kebaruan	1,45	Good	10% of results better, 75% of results worse

Dari aspek daya tarik yang mendapat nilai mean 1,59 dari keseluruhan impresi pemain game ini, mengindikasikan penggunaan konsep pengaturan rintangan yang dapat menyesuaikan tingkat kesulitan berdasarkan ekspresi wajah pemain didalam game ini membuat game menarik untuk dimainkan bahkan untuk kesekian kalinya. Dari aspek kejelasan yang mendapatkan nilai mean 1,14 dari keseluruhan impresi pemain game ini, dapat ditarik bahwa diperlukan penjelasan

lebih lanjut tentang tujuan dan fungsi pengaturan rintangan berdasarkan emosi pemain yang digunakan didalam game sehingga pemain menjadi paham bahwa tujuan utamanya yaitu meningkatkan kualitas bermain pemain serta membuat pemain tidak terlalu takut sehingga akan meninggalkan permainan. Dari aspek efisiensi yang mendapat nilai mean 1,47 dari keseluruhan impresi pemain game ini, tingkat efisiensi pengaturan rintangan yang dilakukan sudah cukup baik untuk menyesuaikan tingkat kesulitan terhadap performa pemain yang sedang memainkan game. Dari aspek ketepatan yang mendapat nilai mean 1,32 dari keseluruhan impresi pemain game ini, tingkat ketepatan pengaturan setiap jenis rintangan sudah cukup akurat menyesuaikan dengan performa masing-masing rintangan yang dilalui pemain didalam game. Dari aspek stimulasi yang mendapat nilai mean 1,71 dari keseluruhan impresi pemain game ini, dapat disimpulkan bahwa penggunaan rancangan rintangan yang dapat menyesuaikan pemainnya cukup berhasil dalam menstimulasi untuk pemain melanjutkan game hingga finish dan tidak berhenti ditengah jalan. Dan yang terakhir yaitu aspek kebaruan yang mendapat nilai mean 1,45 dari keseluruhan impresi pemain game ini, game yang dilengkapi dengan penyesuaian rintangan berdasarkan skor dan emosi pemain termasuk terbarukan atau inovatif menurut sebagian besar pemain.

## **BAB VII**

## **PENUTUP**

Bab ini akan membahas kesimpulan dan saran yang didapat selama pembuatan Tugas Akhir ini. Kesimpulan yang diambil dibuat berdasarkan data yang diperoleh selama penggerjaan Tugas Akhir hingga selesai. Saran diambil berdasarkan hasil masukan-masukan pemain dan data uji coba yang telah dilakukan.

### **7.1 Kesimpulan**

Pada subbab ini akan membahas mengenai kesimpulan yang didapat setelah tugas akhir ini selesai dibuat. Berdasarkan data evaluasi yang diambil dari respon 30 responden, dapat diambil beberapa kesimpulan sebagai berikut:

1. Daya Tarik: Game ini sangat menarik bagi pemain, terutama karena penggunaan konsep pengaturan rintangan yang dapat menyesuaikan tingkat kesulitan berdasarkan ekspresi wajah pemain. Hal ini membuat game tetap menarik bahkan setelah dimainkan berkali-kali.
2. Kejelasan: Meskipun permainan menarik, terdapat kebutuhan untuk memberikan penjelasan lebih lanjut tentang tujuan dan fungsi pengaturan rintangan berdasarkan emosi pemain. Hal ini akan membantu pemain memahami tujuan utama penerapannya didalam game yaitu untuk mencegah pemain merasa terlalu takut sehingga meninggalkan permainan dan membuat level game ber-variatif.
3. Efisiensi: Pengaturan rintangan dalam permainan ini dinilai cukup efisien dalam menyesuaikan tingkat kesulitan terhadap performa pemain yang sedang memainkan game.
4. Ketepatan: Tingkat ketepatan pengaturan setiap jenis rintangan dinilai cukup akurat, yang membantu meningkatkan pengalaman bermain pemain dalam melewati rintangan.

5. Stimulasi: Penggunaan rancangan rintangan yang dapat menyesuaikan pemainnya dinilai berhasil dalam menstimulasi pemain untuk melanjutkan permainan hingga selesai.
6. Kebaruan: Fitur penyesuaian rintangan berdasarkan skor dan emosi pemain dinilai sebagai inovatif dan terbaharukan oleh sebagian besar pemain.

Selama penggeraan Tugas Akhir ini terdapat beberapa kekurangan dan kendala yang ditemui oleh penulis. Kendala dan kekurangan yang ada pada pembuatan Tugas Akhir ini yaitu :

1. Game terkadang dapat merandom sebagian besar level dengan jumlah air yang terlalu banyak.
2. Pendeteksian wajah kurang akurat jika menggunakan kamera yang kurang bagus, atau berada pada kondisi kurang cahaya.
3. Pembentukan atmosfer menakut-nakuti dirasa masih kurang diterapkan didalam game, sehingga beberapa pemain tidak merasa ditakuti-takuti terlebih dahulu.
4. Susunan rintangan yang dibuat terkadang kurang terasa perbedaannya bagi sebagian pemain dikarenakan jumlah pengaturan weight ceiling yang agak kecil yang dibuat sedemikian rupa menyesuaikan dengan panjang dan waktu level yang terbatas.

Kesimpulannya, game ini memiliki potensi yang besar dalam mempertahankan minat pemain dengan daya tarik yang tinggi, tetapi perlu diperhatikan untuk memberikan penjelasan yang lebih jelas tentang pengaturan rintangan dan memastikan ketepatan serta efisiensi dalam menyesuaikan tingkat kesulitan dengan performa pemain.

## 7.2 Saran

Setelah penggeraan Tugas Akhir ini selesai, terdapat saran yang diberikan oleh pemain agar game berkembang menjadi lebih matang dan lebih baik. Saran-saran yang didapat akan diuraikan sebagai berikut :

1. Tampilan user interface diperbaiki sehingga desainnya menjadi lebih fresh dan menarik bagi calon pemain serta mempermudah pemain dalam menavigasi game.
2. Rintangan ditambah lagi modelnya dan intensitas yang muncul bagi pemain ‘Mahir’ diatur lebih intens.
3. Perbaikan kualitas grafik terutama pada bagian kabut yang kadang mengganggu visibilitas pemain.
4. Deteksi emosi tambahan seperti ‘Senang’, ‘Sedih’, dan lainnya sesuai support library MoodMe.
5. Mungkin untuk hantu dapat dibuat lebih bervariasi.
6. Vision di buat lebih menarik lagi.
7. Kalau bisa dibuat versi mobile, karena kalau di PC dan tidak punya webcam kesulitan untuk memainkan game ini.
8. Zoom pada karakter ketika hantu mengikuti terlalu dekat sehingga rintangan di depan sulit terlihat.
9. Mungkin Tombol untuk memukul bisa di dekatkan dengan tombol pergerakan lainnya.
10. Kalau misalnya tingkat kesulitannya naik, dibuat agak ekstrim dong perbedaanya biar kelihatan, kalau buat kesulitan yang menurun kelihatan banget rintangan bisa jadi tidak ada atau jadi sedikit banget. Kalau misalnya tingkat kesulitannya naik, dibuat agak ekstrim dong perbedaanya biar kelihatan, kalau buat kesulitan yang menurun kelihatan banget rintangan bisa jadi tidak ada atau jadi sedikit banget.
11. Susah keluar dari air.

## DAFTAR PUSTAKA

Alika Salsabila Marwanto, Wegig Murwonugroho, *Psikologis Pada Gamers Ketika Memainkan Survival Horror Game "DREADOUT"*, 2021, hlm 2.

Andrew, Adithya Nugraha Tjokrosetio, Andry Chowanda, *Dynamic Difficulty Adjustment With Facial Expression Recognition For Improving Player Satisfaction In A Survival Horror Game*, 2020, hlm 2.

Demediuk, S., Tamassia, M., Raffe, W. L., Zambetta, F., Mueller, F. F., & Li, X., *Measuring player skill using dynamic difficulty adjustment*, In Proceedings of the Australasian Computer Science Week Multiconference, 2018, hlm 1-7.

Ekman, P. Emotions Revealed: Recognizing Faces and Feelings to Improve Communication and Emotional Life. Times Books, 2003.

Ekman, P., & Friesen, W. V. Constants across cultures in the face and emotion. Journal of Personality and Social Psychology. Vol 17(2), 1971, hlm 124–129.

Hunicke, R. The case for dynamic difficulty adjustment in games. In Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology (pp. 429-433), 2005, hlm 14.

Ihwal.Id, “Mengungkap Misteri Gunung Arjuna, 3 Penyebab Pendaki Mudah Tersesat dan Hilang di Alas Lali Jiwa”, (<https://www.ihwal.id/info/68211238086/mengungkap-misteri-gunung-arjuna-3-penyebab-pendaki-mudah-tersesat-dan-hilang-di-alas-lali-jiwa/>), Diakses pada 20 September 2022)

Kirkland, E, Storytelling in survival horror video games, Horror video games: Essays on the fusion of fear and play, (2009), hlm 62-78.

Lopes, P., Liapis, A., & Yannakakis, G. *Targeting horror via level and soundscape generation*, In Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment. Vol. 11, No. 1, 2015, hlm 37-43.

Missura, O. Dynamic difficulty adjustment (Doctoral dissertation, Universitäts- und Landesbibliothek Bonn), 2015, hlm 30-38.

Mohammad Zohaib, Review Article Dynamic Difficulty Adjustment (DDA) in Computer Games: A Review, 2018, hlm 1.

MoodMe, “Unity AI - MoodMe”, (<https://www.mood-me.com/unity-ai/>), Diakses pada 8 Juni 2023)

Nopi Ramsari, Gilang Ramadhan, Pembuatan Game Side Scrolling 2D The Naila's Survival Berbasis Android, 2020, hlm 2.

ROBERTS, Rebecca, *Fear of the unknown: Music and sound design in psychological horror games*, In: Music In Video Games. Routledge, 2014, hlm 138-150.

Tobias Arnell, Nikola Stojanovic, Horror game design – what instills fear in the player 2020, hlm 1.

Unity Technologies, “*Unity Documentation*”, (<https://docs.unity.com/>, Diakses pada 20 Desember 2022)

Unity Technologies, “*Unity Manual – Platform development*”, (<https://docs.unity3d.com/Manual/PlatformSpecific.html>, Diakses pada 21 Desember 2022)

Unity, “*Introduction to Barracuda*”, (<https://docs.unity3d.com/Packages/com.unity.barracuda@1.0/manual/index.html>, Diakses pada 12 Januari 2023)

## **RIWAYAT HIDUP**



**Nama** : Axel Matthew Adiwijaya  
**Alamat Asal** : Kedinding Lor 3/59 A10  
**Tempat/Tanggal Lahir** : Surabaya, 25 Juli 2001

### **Jenjang Pendidikan:**

- 2006 – 2012      SDK Pencinta Damai
- 2012 – 2015      SMPK Stella Maris
- 2016 – 2019      SMAK Stella Maris
- Sejak 2019      Institut Sains dan Teknologi Terpadu Surabaya, Surabaya  
(Program Studi S1 Informatika Professional)