



TESIS - EE185401

VALIDASI PENGATURAN STATS OTOMATIS PADA PERMAINAN ACTION DAN TURN-BASED ROLE-PLAYING GAME (RPG) BERBASIS K-NN DAN NAIVE BAYES DENGAN DEEP LEARNING MULTICLASS CLASSIFICATION

NUR ROHMAN WIDHYANTO
07111650052005

DOSEN PEMBIMBING
Prof. Dr. Ir. Mauridhi Hery Purnomo, M.Eng.
Dr. Supeno Mardi Susiki Nugroho, ST., MT.

PROGRAM MAGISTER
BIDANG KEAHLIAN JARINGAN CERDAS MULTIMEDIA
DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI ELEKTRO DAN INFORMATIKA CERDAS
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2020



TESIS - EE185401

VALIDASI PENGATURAN STATS OTOMATIS PADA PERMAINAN ACTION DAN TURN-BASED ROLE-PLAYING GAME (RPG) BERBASIS K-NN DAN NAIVE BAYES DENGAN DEEP LEARNING MULTICLASS CLASSIFICATION

NUR ROHMAN WIDHYANTO
07111650052005

DOSEN PEMBIMBING
Prof. Dr. Ir. Mauridhi Hery Purnomo, M.Eng.
Dr. Supeno Mardi Susiki Nugroho, ST., MT.

PROGRAM MAGISTER
BIDANG KEAHLIAN JARINGAN CERDAS MULTIMEDIA
DEPARTEMEN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI ELEKTRO DAN INFORMATIKA CERDAS
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2020

LEMBAR PENGESAHAN TESIS

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar

Magister Teknik (MT)

di

Institut Teknologi Sepuluh Nopember

Oleh:

NUR ROHMAN WIDIYANTO

NRP: 07111650050205

Tanggal Ujian: 3 Maret 2020

Periode Wisuda: September 2020

Disetujui oleh:

Pembimbing:

1. Prof. Dr. Ir. Mauridhi Hery Purnomo, M.Eng
NIP: 195809161986011001

2. Dr. Supeno Mardi Susiki Nugroho, ST., MT
NIP: 197003131995121001

Pengaji:

1. Dr. Eko Mulyanto Yuniarno, ST., MT.
NIP: 196806011995121009

2. Dr. Diah Puspito Wulandari, ST. M.Sc
NIP: 198012192005012001

3. Reza Fuad Rachmadi, ST., MT., Ph.D
NIP: 198504032012121001

Kepala Departemen Teknik Elektro
Fakultas Teknologi Elektro

Dedet Candra Riawan, ST., M.Eng, Ph.D.

NIP: 197311192000031001

Halaman ini sengaja dikosongkan

PERNYATAAN KEASLIAN TESIS

Dengan ini saya menyatakan bahwa isi keseluruhan Tesis saya dengan judul “**VALIDASI PENGATURAN STATS OTOMATIS PADA PERMAINAN ACTION DAN TURN-BASED ROLE-PLAYING GAME (RPG) BERBASIS K-NN DAN NAIVE BAYES DENGAN DEEP LEARNING MULTICLASS CLASSIFICATION**” adalah benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan 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, Maret 2020

Nur Rohman Widiyanto
07111650050205

Halaman ini sengaja dikosongkan

Validasi Pengaturan *Stats* Otomatis pada Permainan *Action* dan *Turn-Based Role-Playing Games* (RPG) Berbasis K-NN dan Naive Bayes dengan Deep Learning Multiclass Classification

Nama Mahasiswa : Nur Rohman Widiyanto
NRP : 07111650050205
Pembimbing : 1. Prof. Dr. Ir. Mauridhi Hery Purnomo, M.Eng.
2. Dr. Supeno Mardi Susiki Nugroho, ST., MT.

ABSTRAK

Permainan dengan genre *action* atau *turn-based* RPG (*Role Playing Game*) merupakan permainan yang bersifat kompetitif, antara pemain melawan pemain ataupun melawan musuh yang berupa *non-player character* (NPC). Banyak pengembang permainan dalam pembuatan permainan itu sendiri masih menggunakan cara manual dalam penentuan data statistik dari karakter pemain ataupun musuh. Bila pada sebuah permainan memiliki banyak karakter, seperti hanyalnya banyak karakter pemain (contohnya pada *turn-based* RPG) dan banyak musuh. Pada penelitian ini akan diimplementasikan beberapa pendekatan seperti halnya *k*-NN (*Nearest Neighbor*), Distribusi Normal dan *Naive Bayes* yang akan digunakan dalam membantu dalam menghasilkan data statistik pada karakter dan musuh secara otomatis.

Kata Kunci : *Stats*, *k*-NN, Naive Bayes, Deep Learning, Klasifikasi, *Role-Playing Game* (RPG).

Halaman ini sengaja dikosongkan

Validating the Automatic Stats Adjustment in Action Games and Turn-Based Role-Playing Game (RPG) Based on K-NN and Naive Bayes with Deep Learning Multiclass Classification

By : Nur Rohman Widiyanto
Student Identity Number : 07111650050205
Supervisors : 1. Prof. Dr. Ir. Mauridhi Hery. P, M.Eng.
 2. Dr. Supeno Mardi Susiki. N, ST., MT.

ABSTRAK

Action or turn-based RPG (Role Playing Game) games are competitive games, between players against players or against enemies in the form of non-player characters (NPC). Many game developers in making the game itself still use manual methods in determining statistical data from the character of the player or enemy. If in a game has a lot of characters, such as only many player characters (for example in turn-based RPGs) and many enemies. In this research, several approaches such as k -NN (Nearest Neighbor), k -Means, Normal Distribution, and Naive Bayes will be implemented which will be used to help in producing statistical data on characters and enemies automatically.

Kata Kunci : Stats, k -NN, Naive Bayes, Deep Learning, Classification, Role-Playing Game (RPG).

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Puji dan syukur kehadiran Allah SWT atas segala limpahan berkah, rahmat, serta hidayah-Nya, penulis dapat menyelesaikan penelitian ini dengan judul **Validasi Pengaturan Stats Otomatis pada Permainan Action dan Turn-Based Role-Playing Games (RPG) Berbasis K-NN dan Naive Bayes dengan Deep Learning Multiclass Classification.**

Penelitian ini disusun dalam rangka pemenuhan bidang riset di Jurusan Teknik Elektro ITS, Bidang Studi Teknik Komputer dan Telematika, serta digunakan sebagai persyaratan menyelesaikan pendidikan S1. Penelitian ini dapat terselesaikan tidak lepas dari bantuan berbagai pihak. Oleh karena itu, penulis mengucapkan terima kasih kepada:

1. Keluarga, Ibu, Bapak dan Saudara tercinta yang telah memberikan dorongan spiritual dan material dalam penyelesaian penelitian ini.
2. Bapak Dr. Tri Arief Sardjono, ST., MT. selaku Dekan Fakultas Teknologi Elektro ITS, Bapak Dr. Ir. Wirawan, DEA selaku Ketua Program Pascasarjana FTE Teknik Elektro ITS, Serta Bapak Eko Mulyanto, S.T., M.T. Selaku Ketua Program bidang keahlian Jaringan Cerdas Multimedia FTE Teknik Elektro ITS.
3. Secara khusus penulis mengucapkan terima kasih yang sebesar-besarnya kepada Bapak Prof. Dr. Ir. Mauridhi Hery Purnomo, M.Eng. dan Bapak Dr. Supeno Mardi Susiki Nugroho, ST., MT. atas bimbingan selama mengerjakan penelitian.
4. Bapak-ibu dosen pengajar Bidang Keahlian Jaringan Cerdas Multimedia, atas pengajaran, bimbingan, serta perhatian yang diberikan kepada penulis selama ini.
5. Seluruh teman-teman *B201-crew* Laboratorium Bidang Studi Teknik Komputer dan Telematika.

Kesempurnaan hanya milik Allah SWT, untuk itu penulis memohon segenap kritik dan saran yang membangun. Semoga penelitian ini dapat memberikan manfaat bagi kita semua. Amin.

Surabaya, Maret 2020

Penulis

Halaman ini sengaja dikosongkan

DAFTAR ISI

LEMBAR PENGESAHAN	iii
PERNYATAAN KEASLIAN	v
ABSTRAK	vii
ABSTRACT	ix
KATA PENGANTAR	xi
DAFTAR ISI	xiii
DAFTAR GAMBAR	xv
DAFTAR TABEL	xvii
NOMENKLATUR	xix
1 PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Tujuan	2
1.4 Batasan Masalah	3
1.5 Kontribusi	3
2 KAJIAN PUSTAKA	5
2.1 Kajian penelitian terkait	5
2.2 Desain Permainan Turn-based RPG dan Beberapa Komponennya	6
2.2.1 Penempatan Peluang pada Permainan	6
2.2.2 Elemen Keterampilan Strategi	9
2.2.3 Menemukan Keseimbangan	12
2.3 Machine Learning	13
2.3.1 K-Nearest Neighbor	14
2.3.2 Naive Bayes	22
2.3.3 Klasifikasi dengan Naive Bayes	24
2.3.4 Gaussian Naive Bayes	28
2.4 RPG (Role-Playing Game)	30
3 METODOLOGI	33
3.1 Analisa Komponen Permainan Action dan Turn-based Role-Playing Game (RPG)	34
3.1.1 Desain Skenario Pertarungan	35

3.1.2	Hubungan Sistem Pertarungan dengan Cerita	43
3.2	Desain Level dan Stats pada Karakter Pemain	48
3.2.1	Distribusi Level, HP dan MP Pemain	50
3.2.2	Distribusi Elemen dan Kelemahan Pemain	53
3.2.3	Distribusi Stats Pemain	54
3.3	Desain Level dan Stats pada Karakter Musuh	57
3.3.1	Distribusi Level Musuh	60
3.3.2	Distribusi Tipe Musuh	69
3.3.3	Distribusi Elemen dan Kelemahan Musuh	74
3.3.4	Distribusi HP, MP dan Stats Musuh	78
4	PENGUJIAN	93
4.1	Pengujian dalam Pembuatan <i>Stats</i> Pemain pada Permainan dengan Genre <i>Turn-based</i> RPG	93
4.1.1	Distibusi Level, HP, dan MP Pemain	93
4.1.2	Distribusi Elemen dan Kelemahan Pemain	93
4.1.3	Distibusi stats pemain	94
4.2	Pengujian dalam Pembuatan <i>Stats</i> Musuh pada Permainan dengan Genre <i>Turn-based</i> RPG	94
4.3	Pengujian dalam Pembuatan <i>Stat</i> Pemain untuk Permainan dengan Genre <i>Action</i> RPG	94
4.4	Pengujian dalam Pembuatan <i>Stas</i> Musuh untuk Permainan dengan Genre <i>Action</i> RPG	94
5	PENUTUP	97
5.1	Kesimpulan	97
5.2	Saran	97
Lampiran	99	
Daftar Pustaka	119	
Biografi Penulis	121	

DAFTAR GAMBAR

1.1	Ilustrasi <i>Turn-based RPG</i>	1
2.1	Klasifikasi dengan k -NN.	16
2.2	Proses regresi dengan k -NN.	18
2.3	Contoh permainan digital dengan genre <i>turn-based</i> .	31
2.4	Contoh permainan digital dengan genre <i>RPG</i> .	32
3.1	Urutan metodologi.	33
3.2	Skema pertarungan antar pemain.	35
3.3	Status dari pemain pada permainan <i>turn-based</i> .	37
3.4	Skenario pertarungan <i>turn-based</i> .	39
3.5	Elemen pada permainan dengan mode pertarungan <i>turn-based</i> .	40
3.6	Pengaruh elemen pada efektifitas serangan.	40
3.7	Pengaruh cerita terhadap tingkat kesulitan.	44
3.8	Distribusi jenis musuh sesuai dengan cerita.	46
3.9	Proses pembuatan stats untuk karakter pemain.	48
3.10	<i>Class diagram</i> untuk <i>stats</i> pemain.	49
3.11	Kenaikan HP setiap levelnya.	52
3.12	Kenaikan MP setiap levelnya.	52
3.13	Kenaikan stats pemain setiap levelnya.	56
3.14	Proses pembuatan stats untuk karakter musuh.	57
3.15	<i>Class diagram</i> untuk <i>stats</i> musuh.	59
3.16	Distribusi Level Musuh.	67
3.17	Distribusi level musuh dalam bentuk distribusi normal.	68
3.18	Distribusi tipe musuh.	73
3.19	Distribusi kelemahan musuh.	77
3.20	Distribusi <i>stats</i> HP musuh.	85
3.21	Distribusi <i>stats</i> MP musuh.	85
3.22	Distribusi <i>stats</i> musuh secara keseluruhan.	89
3.23	Distribusi <i>Strength</i> musuh.	89
3.24	Distribusi <i>Magic</i> musuh.	90
3.25	Distribusi <i>Endurance</i> musuh.	90
3.26	Distribusi <i>Speed</i> musuh.	91
3.27	Distribusi <i>Luck</i> musuh.	91

Halaman ini sengaja dikosongkan

DAFTAR TABEL

3.1	Data masukan untuk pembuatan <i>stats</i> pemain.	50
3.2	Hasil Perhitungan HP dan MP	51
3.3	Sampel hasil perhitungan dan distribusi stats dengan <i>k</i> -NN . .	56
3.4	Data masukan untuk pembuatan program pada musuh.	59
3.5	Hasil level yang dibuat untuk musuh.	65
3.6	Hasil level yang dibuat untuk musuh.	72
3.7	Hasil level yang dibuat untuk musuh.	76
3.8	Distribusi <i>Stats</i> HP dan MP musuh.	84
3.9	Distribusi <i>Stats</i> musuh.	87
5.1	Hasil keseluruhan data HP dan MP pada pemain.	99
5.2	Hasil keseluruhan data <i>stats</i> pada pemain (Bag. 1).	100
5.3	Hasil keseluruhan data <i>stats</i> pada pemain (Bag. 2).	101
5.4	Hasil keseluruhan data <i>stats</i> pada pemain (Bag. 3).	102
5.5	Hasil keseluruhan data <i>stats</i> pada pemain (Bag. 4).	103
5.6	Hasil keseluruhan data <i>stats</i> pada musuh (Bag. 1).	104
5.7	Hasil keseluruhan data <i>stats</i> pada musuh (Bag. 2).	105
5.8	Hasil keseluruhan data <i>stats</i> pada musuh (Bag. 3).	106
5.9	Hasil keseluruhan data <i>stats</i> pada musuh (Bag. 4).	107
5.10	Hasil keseluruhan data <i>stats</i> pada musuh (Bag. 5).	108
5.11	Hasil keseluruhan data <i>stats</i> pada musuh (Bag. 6).	109
5.12	Hasil keseluruhan data <i>stats</i> pada musuh (Bag. 7).	110
5.13	Hasil keseluruhan data <i>stats</i> pada musuh (Bag. 8).	111
5.14	Hasil keseluruhan data <i>stats</i> pada musuh (Bag. 9).	112
5.15	Hasil keseluruhan data <i>stats</i> pada musuh (Bag. 10).	113
5.16	Hasil keseluruhan data <i>stats</i> pada musuh (Bag. 11).	114
5.17	Hasil keseluruhan data <i>stats</i> pada musuh (Bag. 12).	115
5.18	Hasil keseluruhan data <i>stats</i> pada musuh (Bag. 13).	116
5.19	Hasil keseluruhan data <i>stats</i> pada musuh (Bag. 14).	117
5.20	Hasil keseluruhan data <i>stats</i> pada musuh (Bag. 15).	118

Halaman ini sengaja dikosongkan

NOMENKLATUR

x	= Data x atau titik koordinat data.
x_0	= Titik koordinat data ke 0.
x_i	= Data x ke i .
x_n	= Data x ke n .
θ	= Data θ .
θ_i	= Data θ ke i .
$\min d$	= Jarak minimum
i	= Urutan atau iterasi.
Y	= Titik Y.
y	= Bagian dari Y dengan banyak data.
X	= Titik X.
D	= Distance atau jarak.
p	= Data p atau titik koordinat data.
p_i	= Titik koordinat data ke i .
y_{pred}	= Hasil prediksi dari titik koordinat yang dicari.
y_i	= Titik koordinat yang dicari oleh y_{pred} .
k	= Nilai k dalam k -NN.
$P(h d)$	= Probabilitas hipotesis h berdasarkan data d . Hal ini disebut probabilitas posterior.
$P(d h)$	= Probabilitas dari data d berdasarkan hipotesis h yang bernilai benar.
$P(h)$	= Probabilitas hipotesis h yang bernilai benar terlepas dari data secara keseluruhan.
$P(d)$	= Probabilitas data secara keseluruhan terlepas dari hipotesis.
$MAP(h)$	= Maximum a Posteriori Probability.
$P(C)$	= Class probability.
C	= Class untuk dicari probabilitasnya.
W	= Weather atau cuaca pada contoh kasus Naive Bayes.
n	= Jumlah data.
\bar{x}	= Rata-rata nilai x .
$\sigma(x)$	= Standar deviasi.
PDF	= Probability Density Function.
B_{total}	= Jumlah waktu melawan bos secara keseluruhan.
HP	= Health Point.
MP	= Magic Point.
$MaxSt$	= Jarak maksimum stats.
St_i	= Stats ke i .

Halaman ini sengaja dikosongkan

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Permainan dengan basis *Role-Playing Game* (RPG) adalah permainan yang mana pemain memerankan sebuah karakter khusus dalam sebuah cerita. Pada permainan tersebut, pemain memiliki tujuan untuk menjalankan misi dan mengikuti alur cerita dari permainan tersebut sampai selesai. Terdapat berbagai macam variasi jenis dari RPG, salah satunya adalah *turn-based*. Pada *turn-based* RPG, pemain dapat memainkan satu karakter atau lebih yang mana karakter tersebut melakukan serangan secara bergantian antara pemain atau musuh seperti pada Gambar 1.1. Biasanya musuh yang dilawan berupa *Non-Player Character* (NPC).



Gambar 1.1: Ilustrasi *Turned-based* RPG

Pada permainan dengan genre *turn-based* RPG yang terdiri dari banyak karakter baik itu karakter pemain dan musuh. Di mana stats tersebutlah yang akan menentukan jalannya pertarungan antara pemain dan musuh. Kemudian stats juga menunjukkan tingkat kesulitan dari musuh, kelemahan dari musuh, daya tahan musuh terhadap serangan dan lain-lain. Hal ini juga merupakan bagian utama dari desain permainan, yang mana nominal pada stats, jumlah pemain dan musuh dapat menentukan beberapa faktor penting lain seperti halnya lamanya permainan dan alur cerita.

Banyak penelitian yang mengaplikasikan berbagai bentuk algoritma dan metode untuk mempercepat pembuatan *video games* seperti halnya pengukuran tingkat kesulitan dengan menggunakan challenging rate (CR) pada permainan 2D *Real Time Strategy* (RTS) (Christyowidiasmoro et al., 2016), pembuatan *game engine* yang berorientasi pada *multi-agent systems* (Marín-Lora et al., 2020), pembuatan level secara otomatis pada permainan secara prosedular dengan tingkat kesulitan tertentu (Wu et al., 2018), ada juga ada yang memakai metode yang dapat menghasilkan *attribute space* untuk menganalisa keseimbangan dalam pertarungan *single unit* pada permainan RTS (Bangay and Makin, 2014). Berdasarkan beberapa penelitian tersebut maka penelitian ini dibuat guna memudahkan desainer permainan dalam mendesain sebuah permainan dengan genre *action* dan *turn-based RPG*.

Penggunaan berbagai metode seperti *k*-NN, Distribusi Normal dan Native bayes yang kemudian dilanjutkan dengan diperolehnya data statistik dari untuk pemain dan musuh yang siap digunakan. Kemudian dilanjutkan dengan proses validasi data statistik tersebut apakah sudah sesuai dengan apa yang direncanakan. Pada proses tersebut digunakanlah *Deep Learning* dengan basis *Neural Network* untuk *Multiclass Classification*.

1.2 Rumusan Masalah

Pembuatan statistik untuk karakter yang akan dimainkan oleh pemain dan juga karakter musuh pada permainan dengan genre action dan turn-based RPG yang terkadang masih dilakukan secara manual, terlebih lagi jika karakter dari pemain dan musuh pada permainan tersebut sangat banyak maka diperlukannya sebuah program yang mampu menyusun hal tersebut secara otomatis.

1.3 Tujuan

Dari permasalahan yang dirumuskan sebelumnya, maka penelitian ini memiliki tujuan sebagai berikut:

1. Mempercepat proses desain permainan dengan menggunakan program yang dapat menghasilkan data statistik untuk karakter dari pemain dan musuh.
2. Terujinya data statistik untuk karakter pemain dan musuh yang dihasilkan secara otomatis oleh program.

1.4 Batasan Masalah

Berdasarkan fokus permasalahan pada bagian sebelumnya, kemudian diambilah beberapa pembatasan masalah. Berikut adalah batasan-batasan masalah tersebut.

1. Data statistik yang dihasilkan oleh program hanya dapat dipakai oleh permainan dengan genre *action* dan *turn-based RPG*.
2. Program yang dibuat untuk menghasilkan data statistik dapat dipakai oleh lebih sama dengan satu karakter pemain dan musuh.
3. Metode yang digunakan dalam proses pembuatan data statistik pemain dan musuh diantaranya adalah *k–NN*, Normal Distribution dan Naive Bayes.
4. Sedangkan metode yang digunakan untuk menguji tingkat ke validitas dari data yang dibuat tadi digunakanlah metode *deep learning* berbasis *Neural Network* dengan *Multiclass Classification*.

1.5 Kontribusi

Di harapkan penelitian ini mampu menjadi rujukan dalam proses desain permainan, khususnya saat pembuatan data statistik untuk karakter pemain dan musuh. Selain itu menjadikan para pengembang permainan dengan genre RPG khususnya *action* dan *turn-based RPG* di masa mendatang menjadi lebih cepat.

Halaman ini sengaja dikosongkan

BAB 2

KAJIAN PUSTAKA

Demi mendukung penelitian ini, dibutuhkan beberapa teori penunjang sebagai bahan acuan dan referensi. Berikut ini akan dipaparkan teori dasar meliputi komponen-komponen desain permainan, penjelasan permainan *turn-based* dan *action RPG*, *Machine Learning* dan *Deep Learning*.

2.1 Kajian penelitian terkait

Beberapa penelitian sebelumnya yang berkaitan dengan penelitian ini dan mendukung penelitian ini antara lain:

1. Paper dengan judul “*Measuring Level of Difficulty in Game Using Challenging Rate (CR) on 2D Real Time Strategy Line Defense Game*” yang ditulis oleh Christyowidiasmoro dan rekan-rekan tahun 2015. Meneliti tentang formula *Challenging Rate* (CR) adalah formula yang menggunakan elemen dasar dalam permainan RTS (*Real Time Strategy*) sebagai parameternya. Dalam tulisan ini dibuat sebuah *prototipe* permainan dengan tingkat kesulitan yang diajukan untuk mencoba kemampuan formula *Challenging Rate* untuk mengukur nilai tingkat kesulitan.
2. Paper dengan judul “*A game engine to make games as multi-agent systems*” yang ditulis oleh Carlos Marín-Lora dan rekan-rekan tahun 2019. Pada tulisan ini meneliti tentang pengembangan *game engine* yang berfokus pada multi-agen. Di mana pada setiap aktor atau agen-agen memiliki sejumlah properti dan aturan kebiasaan untuk berinteraksi dengan lingkungan dalam permainan. Tujuan dari engine tersebut adalah memenuhi kebutuhan dasar dari sistem multi agent dengan merubah karakteristik dari system tanpa adanya pengaruh terhadap potensi karakter.
3. Paper dengan judul “*Procedurally Generating Game Level with Speci-*

fied Difficulty” yang ditulis oleh Zong-Han Wu dan rekan-rekan tahun 2018. Penelitian ini mencakup ranah pembuatan konten dalam sebuah permainan secara prosedural, hal tersebut bertujuan agar para pemain memperoleh pengalaman yang berbeda dalam bermain. Pada paper ini yang paling banyak di ulas adalah permainan PAC-MAN, dengan labirin yang dapat berubah secara procedural.

4. Paper dengan judul “*Generating an Attribute Space for analyzing Balance in Single Unit RTS Game Combat*” yang ditulis oleh Shaun Bangay dan Owen Makin tahun 2014. Fokus pada tulisan ini adalah tentang pencapaian keseimbangan dalam permainan *Real Time Strategy* (RTS), dengan dibuatnya sebuah framework yang dinamakan dengan *attribute space*. Pendekatan ini dievaluasi dan direpresentasikan untuk pertarungan *single unit* atau satu lawan satu pada permainan RTS. Melalui proses tersebut kemudian dibuatlah model prediksi terhadap simulasi pertarungan tersebut, dengan menggunakan attribut atau data statistik pada permainan tersebut seperti halnya *speed*, *range*, *health* dan *damage*.

2.2 Desain Permainan Turn-based RPG dan Beberapa Komponennya

Desain permainan adalah proses menciptakan konten dan aturan permainan. Desain permainan yang baik adalah proses menciptakan tujuan agar seorang pemain merasa termotivasi untuk mencapainya dan aturan yang dibuat dapat diikuti oleh seorang pemain saat membuat keputusan untuk mengejar tujuan-tujuan tersebut (Brathwaite and Schreiber, 2009). Berikut adalah sebagian elemen dari desain permainan yang akan digunakan pada permainan dengan genre *action* dan *Turn-based RPG* pada penelitian ini.

2.2.1 Penempatan Peluang pada Permainan

Sebagian besar permainan setidaknya mengandung beberapa faktor acak. Misalnya, banyak permainan kartu melibatkan proses acak. Adanya

variabel *damage* dipertarungan pada sebagian besar permainan dengan genre RPG. Permainan seperti *Rock-Paper-Scissors* memiliki pola yang tampak acak, meskipun terlihat tanpa mekanisme acak. Pada bagian ini akan banyak membahas mengenai aspek acak pada permainan dan bagaimana *developer* akan menggunakannya.

Pada dasarnya permainan yang memasukan unsur keberuntungan dapat dimainkan dan dimenangkan oleh khalayak luas. Bila dicari alasan penggunaannya, hasilnya sangatlah banyak. Pada akhirnya harus diakui bahwa hal-hal tersebut adalah keputusan dari desainer permainan untuk menciptakan dinamika yang dikehendakinya. Berikut adalah contoh-contoh penerapan penempatan peluang pada permainan yang biasanya diterapkan.

a). Menunda atau Mencegah Penyelesaian Masalah

Untuk permainan yang memiliki sedikit cara penyelesaian saat dipecahkan oleh komputer dan terutama manusia. Maka sesuatu harus dilakukan untuk menjaga permainan agar tetap *fresh* dan menyenangkan. Ditambahkanlah elemen acak untuk mencapai tujuan tersebut. Hal tersebut bertujuan agar pemain tidak terlalu menguasai permainan, karena saat pemain membuat keputusan yang sama persis dapat memberi hasil yang berbeda. Hal tersebut tentunya juga berpengaruh pada tingkat rasa bosan pemain dalam memainkan sebuah permainan, jika pola yang sama terus muncul.

b). Membuat Permainan Lebih Kompetitif

Elemen acak yang sesekali memungkinkan pemain yang kurang berpengalaman untuk menang atau setidaknya menawarkan keuntungan yang membuat permainan ini bertahan lebih lama dengan menggunakan dua cara. Pertama adalah selalu ada peluang kemenangan bagi setiap pemain. Kemudian yang kedua adalah efek dari kekalahan berkurang ketika seorang pemain bisa menyalahkan na-

sib buruknya sendiri atas kekalahannya. Hal tersebut tentunya juga dapat memunculkan rasa tidak terima akan sebuah kekalahan dan dimulainya permainan baru lagi.

c). Meningkatkan Keberagaman

Permainan tanpa elemen acak selalu dimulai dengan cara yang sama dan pola-pola tertentu seperti langkah awal pada permainan Catur. Akibatnya, pemain dapat memiliki pengalaman yang sama dari satu pertandingan ke pertandingan yang lain, dan para pemain dapat mejatuhkan pilihannya untuk selalu memakai strategi yang sama.

d). Menciptakan Momen Dramatis

Ketika seorang pemain dengan hati-hati menyusun strategi dan kemudian harus bergantung pada lemparan dadu atau apa pun yang bersifat acak untuk melihat apakah rencana tersebut berhasil, momen tersebut bisa menjadi sangat menegangkan. *Role-Playing Games* (RPG), *Real-Time Strategy* (RTS) dan banyak *board game* yang mengandalkan kondisi ini saat bermain. Akankah *healing spell* (mantra penyembuhan) yang digunakan akan sampai ke rekan bermainmu yang terluka sebelum monster menyerang? Apakah AI akan memberi pemain dua monster untuk bertempur ataukah hanya satu?

e). Meningkatkan Pengambilan Keputusan

Inti dari sebagian besar permainan adalah keputusan yang dibuat para pemain. Dalam permainan strategi, pemain memiliki informasi lengkap dan tahu hasil pasti dari setiap gerakan yang mereka lakukan. Karena semua variabel diketahui, beberapa keputusan menjadi tidak terlalu menarik, jika ada kesempatan untuk membunuh ratu lawan pada permainan catur secara cuma-cuma tanpa ada yang perlu dikorbankan, hal tersebut bukan merupakan keputusan yang menarik karena adanya jawaban “benar” yang jelas.

Ketika elemen acak ada dalam permainan, tidak ada lagi strategi yang selalu benar. Beberapa gerakan mungkin memiliki peluang kegagalan yang tinggi tetapi juga potensi hasil yang besar, hal tersebut menjadikannya sebuah pilihan berisiko, gerakan lainnya mungkin aman tetapi menghasilkan sedikit keuntungan. Hal tersebut menjadikan pemain akan melakukan analisis gerakan dengan cara yang berbeda berikut risiko dan keutungan yang akan didapat, kemudian tidak lupa mempertimbangkan posisi pemain tersebut dalam permainan atau langkah selanjutnya.

2.2.2 Elemen Keterampilan Strategi

Strategi adalah salah satu kekuatan yang dapat dilakukan oleh para pemain. Karena hal tersebutlah yang membuat para pemain tetap bermain. Pemain bermain *game* dan menikmati setiap prosesnya karena mereka berusaha untuk menguasai pola dalamnya (Koster, 2013). Saat pola berhasil dikuasai, maka selanjutnya pemain akan terus bersenang-senang. Mereka juga membentuk strategi berdasarkan pemahaman mereka tentang dinamika permainan. Penguasaan, pengembangan strategi, hal tersebut terjadi bukan karena ketidak sengajaan. Hal ini adalah tujuan dari desainer permainan atas penggunaan mekanika untuk menciptakan strategi dan taktik dalam permainan yang mereka buat Brathwaite and Schreiber (2009). Berikut beberapa poin penting pada penelitian ini yang mengacu tentang pentingnya Elemen Keterampilan Strategi.

a). Peran dari Keterampilan pada Permainan

Permainan yang bagus dibangun dari serangkaian keputusan menarik, membangun unit untuk menyerang atau bertahan, mencari tahu apa yang harus dilakukan oleh unit tersebut selanjutnya, apa saja kelemahannya dan lain sebagainya. Keberhasilan keputusan tersebut juga didasari oleh reaksi dari mental atau fisik pemain, karena hal tersebut adalah ukuran keterampilan dari seorang pemain.

Permainan yang bagus menyebabkan pemain sering melatih kemampuannya dan memberi *reward* atau bonus dengan pola timbal balik yang jelas. Kemudian sepanjang permainan pemain bertanya-tanya tentang apa yang harus dilakukan selanjutnya. Masuklah mereka ke dalam istilah yang disebut dengan “lingkaran sihir” sebuah *video game*, melalui monitor atau TV sebagai medianya terseraplah mereka ke dalam dunia game. Efek tersebut sama seperti ketika menonton film atau membaca buku yang sangat menarik, tetapi permainan yang baik memiliki daya tarik yang lebih kuat karena terjadi interaksi antara pemain dan keputusan yang dibuat yang terkonversi menjadi sebuah pengalaman.

Ketika pemain terus-menerus membuat keputusan, secara tidak sadar mereka telah memasuki kondisi yang oleh psikolog dan peneliti terkenal Mihaly Csikszentmihalyi disebut “*flow*”. Hal tersebut adalah keadaan permainan yang optimal dan merupakan hasil kerja keras dari seorang desainer permainan. Csikszentmihalyi menulis seluruh buku tentang topik, *Flow: The Psychology of Optimal Experience*. Buku ini tidak terbatas pada permainan saja, tetapi mencakup keadaan semacam ini dari perspektif yang lebih luas.

b). Strategi dan Taktik

Secara teknis strategi utama adalah keseluruhan cara untuk mencapai tujuan akhir jangka panjang (misalnya, kemenangan dalam permainan). Strategi utama terdiri dari beberapa strategi pendukung dengan tujuan jangka pendek atau menengah yang harus dilakukan untuk mencapai strategi besar (misalkan dalam peperangan besar, pilihan untuk bertarung dalam pertempuran tertentu adalah sebuah pilihan strategis). Taktik adalah keputusan mikro tingkat terendah yang dibuat ketika menjalankan strategi misalkan pada pergerakan pasukan, apakah akan diperintahkan untuk melakukan serangan udara, kapan pasukan harus mulai bergerak, dimana po-

sis yang tepat untuk menempatkannya, kapan mulai menembak dan lain-lain adalah contoh keputusan taktis yang dibuat selama pertem-puran militer. Secara informal pemain dalam permainan membuat keputusan strategis ketika membuat rencana jangka panjang (pan-jangnya strategi tersebut bersifat relatif terhadap panjangnya per-mainan), dan keputusan taktis ketika pemain ingin mencapai tujuan jangka pendek.

Pengorbanan mejadikan pengambilan keputusan atau pembuat-an taktik menjadi lebih menarik. Keputusan yang diambil secara cepat atau *twitch mechanics*, bisa disebut juga dengan ketangkas-an memiliki keterbatasan pada taktik. Hal ini menunjukkan bahwa permainan yang lebih fokus pada strategi, umumnya bersifat gilir-an atau *turn-based* seperti Catur dan *Go*, yang mana lebih terfokus pada keputusan yang melibatkan pengorbanan.

c). Game Berbasis Keterampilan Sepenuhnya

Permainan yang fokus pada strategi dan pengorbanan cenderung memiliki setidaknya beberapa elemen peluang. Ketika permainan-permainan ini murni berbasis keterampilan, seperti *Tic-Tac-Toe* atau sebagian besar *video game* petualangan, mereka dapat diselesaikan, dan keputusan-keputusan yang tadinya bervariasi kemudian menjadi keputusan-keputusan yang jelas. Misalkan dalam membuat keputus-an pada akhirnya hanya ada satu gerakan atau pilihan benar.

Sebagian besar game yang sepenuhnya berupa keterampilan ada-la game aksi berbasis fisik. Mungkin hal inilah sebabnya, tidak seperti pada pengorbanan yang bukan tentang mendapatkan jawab-an yang benar atau terbaik melainkan mendapatkannya dengan ce-pat. Waktu reaksi atau tindakan manusia dapat terus meningkat dari waktu ke waktu selamanya, terutama pada permainan dimana manusia saling berlawanan satu dengan yang lain atau *multiplayer*.

2.2.3 Menemukan Keseimbangan

Beberapa permainan sangat terlihat jelas bahwa semua membutuhkan keterampilan. Hal ini sangat bervariasi dari yang sederhana (*Tic-Tac-Toe*) menuju yang lebih kompleks (Catur dan *Go*). Ada juga perbedaan antara keterampilan berbasis strategis (seperti pada kebanyakan permainan berbasis strategi dan giliran) dan kemampuan pengambilan keputusan secara cepat atau *twitch* (biasa ditemukan dalam permainan berbasis keterampilan dan olahraga) atau bisa disebut juga dengan ketangkasan. Permainan ini bisa menjadi sangat menyenangkan karena terdapat keputusan berarti yang dibuat oleh pemain, baik itu keputusan yang dibuat dengan cepat (*twitch*) dan keputusan yang dibuat berdasarkan strategi (Brathwaite and Schreiber, 2009).

Banyak permainan yang menggabungkan kedua basis di atas. *Settlers of Catan* memiliki banyak elemen *gameplay* berbasis keterampilan, termasuk perdagangan, pembangunan, dan manajemen sumber daya. Namun permainan tersebut juga memiliki dadu, setumpuk kartu yang dikocok secara acak, dan pengaturan papan yang disusun dengan acak. *Backgammon* dan *Poker* keduanya memiliki elemen keterampilan dan keberuntungan yang kuat. Bayangkan sebuah permainan *Backgammon* tanpa dadu, atau *Poker* tanpa kemampuan untuk bertaruh atau menggertak lawan, dan menjadi jelas bahwa beberapa permainan mampu menjadi lebih baik dengan adanya campuran keterampilan dan peluang. Jika salah satunya dihapus maka permainan tersebut tidak akan menjadi menarik.

Bagaimana seorang desainer permainan memasukkan campuran keterampilan dan peluang yang benar dalam sebuah permainan? Kapan saat yang tepat untuk mengarahkan permainan ke arah itu? Jawabannya adalah kembali kepada siapakah pemainnya?

Hal tersebut adalah pertanyaan penting, seringkali pertanyaan pertama yang diajukan seorang desainer. Ketika merancang permainan un-

tuk anak berusia enam tahun hasil yang diperoleh akan sangat berbeda jika dibandingkan dengan merancang permainan yang diperuntukkan bagi mahasiswa. Pemain yang berbeda memiliki tingkat toleransi yang berbeda untuk setiap peluang dan keterampilannya. Permainan yang mungkin menjadi sangat populer bagi para gadis muda bisa jadi menjadi membosankan bagi orang dewasa, karena sebagai orang tua yang telah memainkan sejumlah permainan anak-anak dan dapat mengujinya. Beberapa contoh target pemain seperti anak-anak, pemain permainan kompetitif, permainan sosial, pemain profesional dan keluarga. Dalam penelitian ini akan difokuskan dengan permainan kompetitif, karena fokus utamanya adalah *fun games*.

Pemain permainan kompetitif cenderung lebih menyukai permainan dengan lebih banyak elemen keterampilan. Hal ini memberi mereka kesempatan untuk bermain satu lawan satu dengan pemain lain, mencocokan setiap pemikiran menjadi sebuah strategi, refleks melawan refleks, strategi melawan strategi. Hal terakhir yang diinginkan oleh pemain yang benar-benar kompetitif adalah proses acak seperti halnya dengan penggelindungan dadu yang merusak permainan yang dieksekusi dengan semipurna dan terampil.

Mengapa desainer menambah keberuntungan pada permainan berbasis keterampilan? Banyak alasan yang membuat hal-hal tidak dapat diprediksi, meningkatkan kemampuan pemain dengan memainkan permainan itu lagi, dan memungkinkan pemain dengan keterampilan yang sedikit berbeda untuk tetap dapat bersaing. Jika pemain yang lebih baik selalu menang, maka satu-satunya pertarungan yang menarik adalah antara dua pemain dengan kemampuan yang sama rata.

2.3 Machine Learning

Machine Learning (ML) adalah studi ilmiah tentang algoritma dan model statistik yang digunakan oleh komputer untuk melakukan tugas tertentu secara

ra efektif tanpa menggunakan instruksi secara eksplisit, dengan mengandalkan pola dan inferensi sebagai gantinya. *Machine learning* juga dilihat sebagai bagian dari kecerdasan buatan. Algoritma dari *machine learning* membangun model matematika berdasarkan data sampel, yang dikenal sebagai “data training” dalam membuat prediksi atau keputusan tanpa diprogram secara eksplisit untuk melakukan tugasnya (Koza et al., 1996).

Algoritma *machine learning* dapat digunakan pada berbagai aplikasi, seperti penyaringan email, dan visi komputer, yang mana tidak mungkin untuk mengembangkan algoritma instruksi khusus untuk melakukan tugas dengan sendirinya. *Machine learning* terkait erat dengan komputasi statistik yang berfokus pada pembuatan prediksi dengan menggunakan komputer. Studi tentang optimasi matematika memberikan metode, teori dan domain aplikasi ke bidang *machine learning*. Penambangan data atau *data mining* adalah bidang studi yang termasuk ke dalam cakupan *machine learning*, dan berfokus pada analisis dan eksplorasi data melalui pembelajaran yang mampu berjalan secara otomatis (Friedman, 1997).

Pada penelitian ini akan digunakan beberapa algoritma *machine learning* untuk menyelesaikan permasalahan dalam mendesain permainan. Penggunaan algoritma tersebut secara umum adalah membuat data baru yang terstruktur dengan referensi data awal yang disesuaikan dengan kebutuhan untuk mendesain permainan. Berikut algoritma-algoritma tersebut dan penjelasannya secara umum nantinya, di mana natinya akan dipakai dan dijelaskan secara detail pada BAB 3.

2.3.1 K-Nearest Neighbor

Di asumsikan terdapat sampel (x_i, θ_i) yang didistribusikan secara sesuai dengan distribusi (x, θ) argumen heuristik tertentu yang memungkinkan dalam prosedur pengambilan keputusan. Contohnya saat pengasumsian tentang jarak dari beberapa data, metrik atau koordinat yang sesuai akan diklasifikasi berdasarkan kesamaannya atau setidaknya me-

miliki kesamaan distribusi pada setiap datanya. Maka dalam klasifikasi sampel x , dipertimbangkan berdasarkan x_i terdekat yang diasumsikan sebagai kemungkinan terbesar. Sehingga prosedur pengambilan keputusan paling sederhana adalah aturan *nearest neighbor* (NN) dengan mengklasifikasikan x sebagai tetangga terdekat.

Di buatlah satu set n pasangan $(x_1, \theta_2), \dots, (x_n, \theta_n)$, yang mana x_i 's mengambil nilai dalam metrik X yang didefinisikan sebagai metrik d , dan O_i nilai yang diambil di set $\{1, 2, \dots, M\}$ seperti pada persamaan 2.1 dan 2.2. Setiap θ_i dianggap sebagai indeks dari kategori yang dimiliki oleh data ke- i , dan setiap x_i adalah hasil dari pengukuran setiap data. Lebih singkatnya pada “ x_i bagian dari θ_i ” adalah saat data ke- i yang menjadi dasar pengukuran x_i yang telah diamati dan diukur, termasuk juga pada kategori θ_i .

Di lakukan beberapa penambahan pasangan baru (x, θ) , yang mana pengukuran x yang akan dilakukan, dan digunakan hasilnya digunakan untuk memperkirakan θ dengan memanfaatkan hasil pengukuran atau klasifikasi sebelumnya. Pada persamaan 2.1 dinyatakan bahwa seluruh data x atau x_n yang akan digunakan untuk mencari nilai θ .

$$x'_n \in x_1, x_2, \dots, x_n \quad (2.1)$$

Akan termasuk ke dalam *nearest neighbor* terhadap x jika dilanjutkan dengan persamaan berikut.

$$\min d(x_i, x) = d(x'_n, x), \quad i = 1, 2, \dots, n \quad (2.2)$$

Aturan *nearest neighbor* menentukan, apakah x termasuk dalam kategori θ'_n dari tetangga terdekatnya x'_n seperti pada persamaan 2.2 di mana metrik d adalah *distance* atau jarak. Kesalahan dibuat jika $\theta'_n \neq \theta$. Perhatikan bahwa aturan *nearest neighbor* hanya menggunakan klasifikasi berdasarkan tetangga terdekat saja. Sedangkan $n - 1$ yang merupakan sisanya hasil klasifikasi θ_i akan diabaikan (Cover and Hart, 1967). Dengan

demikian, metode klasifikasi berbasis k -NN dapat dengan mudah diterapkan dalam banyak tugas klasifikasi. Secara umum, sebagian besar varian k -NN menentukan label kelas dari sampel data dengan menggunakan satu parameter k yang merupakan jarak dari keseluruhan data yang di anggap sebagai *neighbour* dan juga keputusan di ambil dengan mempertimbangkan jarak data secara mayoritas. Tetapi keputusan klasifikasi tersebut dapat dengan mudah dipengaruhi oleh sensitivitas k itu sendiri dan juga dapat memperburuk sensitivitasnya, sehingga sangat menurunkan kinerja klasifikasi dari k -NN terutama dalam kasus ukuran sampel yang kecil.

a). Klasifikasi

Dalam mendemonstrasikan analisis k -nearest neighbor, diperlukan juga proses klasifikasi objek baru (koordinat data) diantara sejumlah contoh yang diketahui. Hal ini ditunjukkan pada Gambar 2.1, yang menggambarkan representasi dari data atau *instance* dengan tanda plus dan minus dan titik dengan lingkaran merah. Dan permasalahan yg harus diselesaikan adalah memperkirakan (klasifikasi) hasil dari titik tersebut berdasarkan sejumlah tetangga terdekat atau *nearest neighbor* yang dipilih. Dengan kata lain, apakah titik tersebut dapat diklasifikasikan sebagai tanda plus atau minus.



Gambar 2.1: Klasifikasi dengan k -NN.

Kemudian dipertimbangkannya hasil k -NN berdasarkan tetangga terdekat pertama ($1\text{-nearest neighbor}$). Jelas bahwa dalam hal ini k -NN akan memprediksi hasil dari titik atau representasi data dengan nilai tambah (karena titik terdekat adalah tanda plus). Dilanjutkan dengan penambahan jumlah tetangga terdekat menjadi 2 ($2\text{-nearest neighbor}$). Kali ini k -NN tidak akan dapat mengklasifikasikan hasil dari titik dari data yang dicari, karena titik terdekat keduanya adalah minus, sehingga tanda plus dan minus mencapai jarak yang sama. Untuk langkah selanjutnya, ditambahkan lagi jumlah tetangga terdekat menjadi 5 ($5\text{-nearest neighbor}$). Hal ini akan menentukan daerah tetangga terdekat, yang ditunjukkan oleh lingkaran yang seperti pada Gambar 2.1. Karena ada 2 tanda plus dan 3 tanda minus. Pada lingkaran tersebut k -NN akan memberikan tanda minus pada hasil dari titik koordinat yang dicari.

b). Regresi

Pada bagian ini akan digeneralisasi konsep k -*nearest neighbor* untuk menyelesaikan permasalahan regresi. Permasalahan regresi sangat berkaitan dengan prediksi hasil dari variabel dependen berdasarkan beberapa variabel independen. Di mulai dengan mempertimbangkan skema yang ditunjukkan pada Gambar 2.2, di mana satu data *training* atau data sampel (kotak jingga) yang diambil dari hubungan antara variabel independen x dan variabel dependen y (kurva biru).

Kemudian ditandai dengan kotak jingga yang dilanjutkan dengan penerapan k -*nearest neighbor* untuk memprediksi hasil keluaran yang ditandai dengan X . Kemudian direpresentasikan juga dengan kotak jingga. Di mulai dengan mempertimbangkan metode $1\text{-nearest neighbor}$ sebagai data sampel. Dalam kasus ini akan dicari data sample (kotak jingga) yang paling dekat dengan titik X . Pada Gambar 2.2 yang dicari dinyatakan dalam koordinat (X, Y) sedangkan koordinat



Gambar 2.2: Proses regresi dengan k -NN.

data yang paling mendekati adalah (x_4, y_4) . Maka dapat disimpulkan bahwa tetangga terdekatnya atau *nearest neighbor* adalah (x_4, y_4) untuk *1-nearest neighbor*.

Langkah selanjutnya adalah mencari *2-nearest neighbor* seperti pada Gambar 2.2. Pada kasus ini ditemukan dua titik terdekat dengan X atau *2-nearest neighbor*, yang kebetulan ditunjukkan pada Gambar 2.2 adalah y_3 dan y_4 . Maka diambil rata-rata dari masing-masing nilai pada data tersebut yang dinyatakan pada persamaan 2.3.

$$Y = \frac{y_3 + y_4}{2} \quad (2.3)$$

Pada metode *k-nearest neighbor* hasil Y dari titik X dianggap sebagai hasil rata-rata dari nilai *k-nearest neighbor* dari tetangga terdekatnya.

c). Cross-Validation

Cross-validation adalah teknik yang dapat digunakan untuk mendapatkan estimasi model parameter yang tidak diketahui. Pembah-

hasan ini bertujuan untuk menerapan teknik dalam memperkirakan k .

Gagasan umum dari metode ini adalah untuk membagi sampel data ke dalam sejumlah kelompok atau *cluster* (diambil secara acak, *sub-samples* atau bagian-bagian yang terpisah). Untuk nilai tetap pada k , digunakanlah k -NN untuk membuat prediksi pada bagian ke v (mislahnya dengan penggunaan bagian $v-1$ sebagai contoh) dan kemudian digunakan untuk evaluasi kesalahan. Metode paling umum untuk analisa kesalahan pada regresi adalah dengan jumlah kuadrat atau *sum of square* dan pada klasifikasi hal tersebut didefinisikan sebagai akurasi atau tingkat kebenaran klasifikasi.

Proses ini kemudian diterapkan secara berurutan pada semua bagian atau v yang mungkin. Pada bagian terakhir, kesalahan yang dijumlahkan kemudian dirata-rata untuk menghasilkan model yang stabil (seberapa baik model memprediksi titik koordinat data). Langkah-langkah di atas kemudian diulang untuk berbagai nilai k yang lain, nilai yang mencapai kesalahan terendah atau akurasi klasifikasi tertinggi dijadikan sebagai nilai optimal untuk k .

Usaha dalam mencapai nilai optimal inilah yang kemudian disebut dengan *cross-validation*. Perlu diperhatikan bahwa *cross-validation* membutuhkan proses komputasi tinggi dan algoritma harus dibiarkan berjalan untuk beberapa waktu terutama saat ukuran sampel berjumlah besar. Atau nilai k dapat ditentukan sendiri. Ini mungkin tindakan yang wajar jika sudah megetahui apa yang akan diambil sebagai nilai k , dari analisis k -NN sebelumnya yang mungkin dilakukan pada data yang sama atau pengamatan secara manual.

d). Jarak Metrik

Seperti disebutkan sebelumnya, pada titik data yang dicari, k -NN membuat prediksi berdasarkan hasil k dari tetangga terdekat dengan

titik itu. Oleh karena itu, untuk membuat prediksi dengan k -NN, sangat diperlukan pendefinisian metrik untuk mengukur jarak antara titik koordinat data dan kasus dari data sampel.

Salah satu metode paling populer untuk mengukur jarak ini dikenal dengan istilah *Euclidean*. Langkah-langkah lain diantaranya adalah *Euclidean squared*, *City-block*, dan *Chebyshev* seperti yang ditunjukkan pada persamaan 2.4.

$$D(x, p) = \begin{cases} \sqrt{(x - p)^2} & \text{Euclidean} \\ (x - p)^2 & \text{Euclidean squared} \\ \text{abs}(x - p) & \text{Cityblock} \\ \text{Max}(|x - p|) & \text{Chebyshev} \end{cases} \quad (2.4)$$

Di mana pada persamaan 2.4 variabel x dan p masing-masing adalah titik koordinat data yang dicari dan data sampel. Kemudian yang dimaksud dengan D adalah *distance* atau jarak antara data sampel dengan koordinat data yang dicari.

e). Prediksi

Setelah memilih nilai k , maka prediksi berdasarkan data sampel dengan menggunakan k -NN dapat dilakukan. Sedangkan pada regresi seperti yang dijelaskan pada bagian sebelumnya, k -NN memprediksi hasil rata-rata dari k -nearest neighbor.

$$y_{pred} = \frac{1}{k} \sum_{i=1}^k y_i \quad (2.5)$$

Di mana y_i adalah kejadian ke- i pada data sampel dan y_{pred} adalah hasil prediksi dari titik koordinat data yang dicari. Berbeda dengan regresi, dalam masalah klasifikasi, prediksi k -NN didasarkan pada skema *voting* di mana pemenanglah yang akan digunakan untuk melabeli titik koordinat data yang dicari.

Untuk klasifikasi biasanya nilai ganjil seperti $y_{pred} = 1, 3, 5$, dan seterusnya sering digunakan untuk menghindari terjadinya *ties condition*, yaitu kondisi dimana terdapat dua label kelas yang mencapai skor yang sama.

f). Pembobotan Jarak

Karena prediksi k -NN didasarkan pada asumsi intuitif bahwa objek yang jaraknya dekat berpotensi dianggap sama. Hal tersebut sangat masuk akal dalam hal membedakan antara k pada tetangga-tetangga terdekat ketika membuat prediksi.

Dengan membiarkan titik terdekat antara k pada setiap tetangga terdekat maka semakin banyak jarak yang diperoleh dan akan mempengaruhi hasil dari titik data yang ingin diolah. Hal ini dapat dicapai dengan menerapkan bobot atau W pada setiap tetangga terdekat, yang ditentukan oleh jarak masing-masing tetangga dengan titik data yang ingin diolah seperti pada persamaan 2.6.

$$W(x, p_i) = \frac{\exp(-D(x, p_i))}{\sum_{i=1}^k \exp(-D(x, p_i))} \quad (2.6)$$

Di mana $D(x, p_i)$ adalah jarak antara titik data yang akan diolah, sementara x dan p_i adalah data sampel ke- i . Hal tersebut sudah menjelaskan bahwa bobot yang ditentukan dengan persamaan 2.6 akan dapat dipenuhi atau dibuktikan dengan persamaan 2.7.

$$\sum_{i=1}^k W(x_0, x_i) = 1 \quad (2.7)$$

Sedangkan untuk permasalahan regresi dapat dipenuhi atau dibuktikan dengan persamaan 2.8.

$$y = \sum_{i=1}^k W(x_0, x_i) y_i \quad (2.8)$$

Kemudian untuk permasalah klasifikasi, persamaan 2.7 dapat diambil untuk klasifikasi setiap variabelnya. Sudah jelas bahwa dari pemmbahasan ini, ketika $k > 1$, maka seseorang dapat secara langsung menentukan standar deviasi untuk prediksi pada regresi dengan menggunakan persamaan 2.9.

$$\text{error bar} = \mp \sqrt{\frac{1}{k-1}} \sum_{i=1}^k (y - y_1)^2 \quad (2.9)$$

2.3.2 Naive Bayes

Pada *machine learning* sering digunakannya hipotesis terbaik (h) dari data yang akan diproses (d). Dalam klasifikasi dengan menggunakan *naive bayes*, hipotesis (h) mungkin dapat dijadikan sebagai kelas untuk mengklasifikasi data baru (d).

Salah satu cara termudah untuk memilih hipotesis yang memungkinkan adalah dengan menggunakan data yang sudah ada. Kemudian digunakan sebagai referensi untuk menyelesaikan masalah tersebut. Teorema Bayes memberikan cara untuk menghitung probabilitas berdasarkan hipotesis yang diberikan berdasarkan pemahaman terhadap data yang ingin diolah. Maka teorema bayes dapat dinyatakan seperti pada persamaan 2.10

$$P(h|d) = \frac{(P(d|h) \times P(h))}{P(d)} \quad (2.10)$$

Di mana pada persamaan 2.10 akan dijelaskan menjadi beberapa poin, berikut poin-poin tersebut.

1. $P(h|d)$ adalah probabilitas hipotesis h berdasarkan data d . Hal ini disebut probabilitas posterior.
2. $P(d|h)$ adalah probabilitas dari data d berdasarkan hipotesis h yang bernilai benar.

3. $P(h)$ adalah probabilitas hipotesis h yang bernilai benar terlepas dari data secara keseluruhan.
4. $P(d)$ adalah probabilitas data secara keseluruhan terlepas dari hipotesis.

Dapat dilihat bahwa dalam menghitung probabilitas posterior $P(h|d)$ dari probabilitas sebelumnya $P(h)$ dengan $P(d)$ dan $P(d|h)$. Setelah menghitung probabilitas posterior dengan beberapa hipotesis, maka dapat dipilih hipotesis dengan probabilitas tertinggi. Ini adalah hipotesis dengan probabilitas paling tinggi biasanya disebut juga dengan *MAP* (*Maximum a Posteriori Probability*). Pada persamaan 2.11, 2.12, dan 2.13 dinyatakan bagaimana persamaan untuk mencari *MAP* dari seluruh probabilitas hasil *naive bayes*.

$$MAP(h) = \max(P(h|d)) \quad (2.11)$$

atau

$$MAP(h) = \max \left(\frac{P(d|h) \times P(h)}{P(d)} \right) \quad (2.12)$$

atau

$$MAP(h) = \max(P(d|h) \times P(h)) \quad (2.13)$$

$P(d)$ adalah aturan normalisasi yang biasanya digunakan dalam perhitungan probabilitas. Hal tersebut dapat tidak berlaku saat sudah dimukannya hipotesa yang paling mungkin selama hal tersebut hanya berupa aturan yang konstan dan digunakan saat normalisasi saja maka dari itu pada persamaan 2.13 dapat ditinggal atau tidak diperhitungkan. Sedangkan h dalam $MAP(h)$ maksudnya adalah hipotesa dari seluruh hasil perhitungan dengan *naive bayes* yang memiliki probabilitas yang paling besar.

Kembali lagi ke klasifikasi, jika telah memiliki sejumlah *instance* atau sekumpulan data dalam setiap kelas pada data *training*, maka probabilitas masing-masing kelas $P(h)$ akan sama karena sudah terklasifikasi sebelumnya. Hal tersebut $P(h)$ menjadi sebuah aturan yang bersifat konstan, sehingga dapat dihapus dari persamaan 2.13 menjadi persamaan 2.14

$$MAP(h) = \max(P(d|h)) \quad (2.14)$$

2.3.3 Klasifikasi dengan Naive Bayes

Naive Bayes adalah algoritma yang dapat digunakan untuk menyelesaikan masalah klasifikasi biner (dua kelas) dan multi kelas. Teknik ini paling mudah dipahami ketika dijelaskan dengan menggunakan nilai masukan biner atau terkategorisasi.

Metode ini dinamakan *naive* karena perhitungan probabilitas untuk setiap hipotesis disederhanakan agar kalkulasi dapat dilakukan. Jika dibandingkan dengan menghitung nilai dari masing-masing atribut $P(d_1, d_2, d_3)$ maka diasumsikan masing-masing atribut saling independen dengan target nilai dan dihitung sebagai $P(d_1|h) \times P(d_2|h)$ dan se-terusnya.

1). Representasi dari Naive Bayes adalah Probabilitas

Representasi dari *naive bayes* adalah probabilitas. Daftar probabilitas yang akan disimpan ke kesebuah variabel untuk dilakukan proses *learning* dengan model *naive bayes*. Berikut dua langkah yang biasanya dipakai dalam pengklasifikasian dengan *naive bayes*.

- a. ***Class probabilities:*** Di carinya probabilitas untuk setiap kelas dalam *dataset training*, jadi tujuan dari langkah ini adalah di-perolehnya probabilitas pada setiap kelas, sekelompok data atau data *clusster*.

- b. ***Conditional probabilities***: Sering disebut dengan probabilitas bersyarat, dicarinya probabilitas dari setiap nilai dalam *dataset training* terhadap kelas yang ada.

2). Proses Learning Data dengan Naive Bayes

Proses learning pada *data training* dengan menggunakan model *naive bayes* sangatlah cepat. Proses tersebut menjadi cepat karena hanya probabilitas pada masing-masing kelas dan probabilitas pada setiap kelas diberi nilai input (x) yang berbeda kemudian dihitung. Selain itu tidak ada koefisien yang perlu dicocokkan dengan menggunakan prosedur optimasi.

3). Perhitungan pada *Class Probabilities*

Class probabilities sebenarnya adalah frekuensi dari *instance* atau sekumpulan data pada masing-masing kelas dibagi dengan jumlah total *instance*. Sebagai contoh dalam klasifikasi biner atau dua kelas, probabilitas *instance* yang termasuk dalam kelas ke 1 dapat dinyatakan dengan persamaan 2.15.

$$P(C_1) = \frac{C_1}{(C_0 + C_1)} \quad (2.15)$$

Pada persamaan 2.15 terdapat dua buah kelas yang mewakili kombinasi angka biner, angka 1 yang diwakili dengan C_1 dan angka 0 yang diwakili dengan C_0 . Kemudian dicarilah peluang munculnya angka 1 atau $P(C_1)$. Dalam kasus paling sederhana setiap kelas akan memiliki probabilitas 0.5 atau 50% untuk masalah klasifikasi biner dengan jumlah *instance* yang sama di setiap kelas. Lebih jelasnya lagi bahwa terdapat sekumpulan data yang terkelompokkan menjadi dua yang diumpamakan dengan 0 dan 1.

4). Perhitungan *Conditional Probability*

Probabilitas bersyarat atau *Conditional probability* adalah frekuensi dari setiap atribut nilai terhadap setiap nilai pada sebuah

kelas dibagi dengan frekuensi dari *instances* atau nilai-nilai pada kelas itu. Misalkan terdapat sebuah atribut “cuaca” atau “*weather*” memiliki nilai “cerah” dan “hujan” kemudian akan diklasifikasikan berdasarkan atribut dari kelas yang memiliki nilai “jalan-jalan” dan “tetap di rumah” yang akan digunakan untuk mengklasifikasikan atribut dari “cuaca”, maka probabilitas bersyarat dari setiap atribut pada “cuaca” dapat dirumuskan dengan persamaan 2.16, 2.17, 2.18, dan 2.19.

$$P(W_1|C_1) = \frac{W_1 \times C_1}{C_1} \quad (2.16)$$

$$P(W_1|C_2) = \frac{W_1 \times C_2}{C_2} \quad (2.17)$$

$$P(W_2|C_1) = \frac{W_2 \times C_1}{C_1} \quad (2.18)$$

$$P(W_2|C_2) = \frac{W_2 \times C_2}{C_2} \quad (2.19)$$

Berdasarkan beberapa persamaan diatas atribut “cuaca” atau “*weather*” dinyatakan dengan W sedangkan atribut untuk mewakili kelas yang akan digunakan untuk mengklasifikasikan dinyatakan dengan C , kemudian hasil dari perhitungan probabilitas tersebut dinyatakan dengan P . Atribut dari “cuaca” terdiri dari “cerah” (W_1) dan “hujan” (W_2), sedangkan atribut dari kelas yang terdiri dari “jalan-jalan” (C_1) dan “tetap di rumah” (C_2). Kemudian $P(W_n|C_n)$ adalah probabilitas yang mungkin terjadi untuk “jalan-jalan” atau “tetap di rumah” saat kondisi “cerah” atau “hujan”, dengan n yang dapat diganti dengan angka 1 dan 2.

5). Membuat Prediksi dengan Naive Bayes

Dengan didapatkannya model *naive bayes*, maka dapat dilakukan prediksi data baru menggunakan teorema bayes. Dengan meng-

gunakan persamaan 2.13, di mana fokus dari persamaan tersebut adalah dipilihnya kelas dengan perhitungan probabilitas terbesar. Karena diperolehnya hipotesa “cuaca” yang mungkin untuk “jalan-jalan” adalah “cuaca” berada pada atribut “cerah”.

Di gunakannya contoh kasus yang sama seperti pada Sub-Bab 2.3.3 poin ke 4 dan dipilihnya atribut “cerah” dimana cuaca adalah *instance*. Kemudian dilakukan perhitungan untuk mencari nilai probabilitas maksimal dari “jalan-jalan” atau “tetap di rumah” yang merupakan bagian dari atribut kelas.

$$C_1 = P(W_1|C_1) \times P(C_1) \quad (2.20)$$

$$C_2 = P(W_1|C_2) \times P(C_2) \quad (2.21)$$

Seperti pada persamaan 2.20 dan 2.23 dimana C_1 adalah nilai maksimal probabilitas untuk atribut “jalan-jalan” dan C_2 adalah nilai maksimal probabilitas untuk “tetap di rumah” yang diperoleh dari kalkulasi probabilitas bersyarat dari kondisi “cerah” terhadap “jalan-jalan” dinyatakan dengan $P(W_1|C_1)$ dikalikan dengan probabilitas dari “jalan-jalan” $P(C_1)$ dan kondisi “cerah” terhadap “tetap di rumah” dinyatakan dengan $P(W_1|C_2)$ dikalikan dengan probabilitas dari “tetap di rumah”. Kemudian dapat dipilihlah dari kedua atribut C_1 dan C_2 yang memiliki nilai tertinggi, maka itulah hasil prediksinya. Nilai-nilai tersebut yang kemudian dapat diubah kedalam bentuk probabilitas dengan menormalkannya seperti pada persamaan berikut.

$$P(C_1|W_1) = \frac{C_1}{C_1 + C_2} \quad (2.22)$$

$$P(C_2|W_1) = \frac{C_2}{C_1 + C_2} \quad (2.23)$$

Di mana pada persamaan 2.22 dan 2.23, $P(C_1|W_1)$ adalah probabilitas untuk “jalan-jalan” dan $P(C_2|W_1)$ adalah probabilitas untuk “tetap di rumah”. Jika terdapat lebih banyak variabel input, maka contoh diatas dapat lebih diperluas lagi. Misalkan dengan ditambahkan atribut “mobil” dengan nilai “berjalan” dan “rusak”, maka probabilitasnya dikalikan menjadi sebuah persamaan. Sebagai contoh perhitungan pada kelas dengan atribut “jalan-jalan” ditambahkan *instance* baru yaitu “mobil” yang beri hipotesa “berjalan” maka persamaan 2.20 berubah menjadi persamaan berikut.

$$C_1 = P(W_1|C_1) \times P(V_1|C_1) \times P(C_1) \quad (2.24)$$

Di mana pada persamaan 2.24 terdapat variabel V yang menyatakan “*vehicle*” atau “mobil”, kemudian terdapat dua kondisi atau atribut pada variabel tersebut yaitu “berjalan” dan “rusak” yang masing-masing dapat dinyatakan dalam variabel V_1 dan V_2 .

2.3.4 Gaussian Naive Bayes

Penggunaan *Naive bayes* dapat diperluas ke atribut yang bernilai bilangan real, hal yang paling umum dilakukan adalah dengan membuat asumsi distribusi *Gaussian*.

Gaussian Naive Bayes adalah salah satu pengembangan dari *naive bayes*. Salah satu kegunaanya adalah dapat digunakan untuk memperkirakan distribusi data, tetapi istilah *Gaussian* (distribusi normal) adalah salah satu cara termudah untuk digunakan, karena hanya perlu memperkirakan rata-rata dan standar deviasi dari data *training* yang ingin digunakan.

1). Representasi Gaussian Naive Bayes

Pada bagian sebelumnya penghitungan probabilitas untuk nilai masukan data pada setiap kelas menggunakan frekuensi. Dengan

masukan yang berupa angka-angka, kemudian dihitung rata-rata dan standar deviasi dari masukan (x) pada setiap kelasnya untuk dilakukan distribusi data baru.

Hal tersebut menunjukkan bahwa selain terjadinya penambahan probabilitas pada setiap kelas, ditambahkan juga rata-rata dan standar deviasi pada setiap masukan variabel dari masing-masing kelas.

2). Proses Learning Data dengan Gaussian Naive Bayes

Hal ini sesederhana menghitung nilai rata-rata dan standar deviasi dari setiap nilai dari variabel masukan (x) pada setiap kelas.

$$\bar{x} = \frac{1}{n} \sum_{i=0}^k x \quad (2.25)$$

Di mana pada persamaan 2.25, variabel n adalah jumlah nilai dari sekumpulan data dan x adalah nilai-nilai untuk variabel masukan dalam data training. Kemudian k adalah banyaknya data *training*, dan \bar{x}_k adalah rata-rata nilai dari data *training*. Kemudian untuk standar deviasi dapat dihitung dengan menggunakan persamaan berikut ini.

$$\sigma(x) = \sqrt{\frac{\sum_{i=0}^k (x_i - \bar{x})^2}{n}} \quad (2.26)$$

Pada dasarnya standar deviasi pada persamaan 2.26 adalah akar kuadrat dari perbedaan rata-rata yang dikuadratkan dari setiap nilai x yang dinyatakan dengan x_i dari hasil rata-rata atau \bar{x} . Kemudian n adalah banyaknya data dan i adalah urutan dari setiap data tersebut. Hasil dari perhitungan standar deviasi tersebut dinyatakan dengan $\sigma(x)$ atau standar deviasi dari x .

3). Prediksi dengan Model Gaussian Naive Bayes

Pada bagian ini akan dijelaskan prediksi nilai x selanjutnya dengan menggunakan *Gaussian PDF* (*Probability Density Function*). Saat membuat prediksi, parameter ini dimasukan ke *Gaussian PDF*

dengan masukan baru untuk variabel, dan sebagai gantinya *Gaussian* PDF akan memberikan perkiraan probabilitas nilai masukan baru untuk kelas tersebut.

$$PDF(x, \bar{x}, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x - \bar{x})^2}{2\sigma^2}\right) \quad (2.27)$$

Di mana maksud dari $PDF(x, \bar{x}, \sigma)$ pada persamaan 2.27 adalah *Gaussian* PDF dari x yaitu nilai masukan untuk variabel masukan untuk prediksi, \bar{x} atau rata-rata dan σ adalah standar deviasi yang sudah dihitung pada bagian atau berdasarkan data-data sebelumnya, π adalah konstanta numerik pada umumnya, kemudian fungsi $\exp()$ atau e adalah konstanta numerik yang betujuan untuk membentuk hasil prediksi dengan pendekatan eksponensial.

Kemudian pada persamaan 2.27 dapat dimasukan ke dalam persamaan 2.24 untuk membuat prediksi dengan masukan data baru. Sebagai contoh, dengan mengadaptasi salah satu perhitungan pada persamaan 2.24 untuk “cuaca” dan “car”.

$$C_1 = P(PDF(W)|C_1) \times P(PDF(V)|C_1) \times P(C_1) \quad (2.28)$$

Pada persamaan 2.28 penjelasan masih sama persis dengan 2.28, tetapi data hipotesa diganti dengan fungsi Gauss *PDF*. Di mana $PDF(W)$ adalah data hasil prediksi data baru untuk “cuaca”, kemudian digunakan hipotesa dalam memprediksi hasil prediksi. Hal tersebut berlaku juga untuk data hasil prediksi “mobil”, dimana akan digunakan hipotesa dalam memprediksi hasil prediksi.

2.4 RPG (Role-Playing Game)

Permainan berbasis *turn-based* (Panumate et al., 2015) adalah permainan yang memanfaatkan waktu secara diskrit, yang mana alur dari permainan khususnya pertarungan terbagi menjadi beberapa bagian yang disebut dengan giliran. Pada setiap giliran, pemain akan memiliki waktu yang terbatas

atau tidak terbatas dalam berpikir dalam membuat keputusan setiap langkahnya. Kemudian sistem dari permainan akan memproses tindakan dari pemain. Kemudian permainan dilanjutkan oleh pemain berikutnya atau musuh yang berupa AI akan melakukan giliran selanjutnya seperti pada Gambar 2.3.



Gambar 2.3: Contoh permainan digital dengan genre *turn-based*.

Role-playing Game (RPG) (Panumate et al., 2015) adalah jenis permainan yang mana pemain memainkan peran karakter dalam latar fiktif. Selain itu pemain juga bertanggung jawab untuk memainkan peran sesuai narasi cerita, baik melalui adegan secara literal atau melalui proses pembuatan keputusan dalam setiap adegannya yang berujung pada pengembangan karakter. Salah satu contoh permainan digital bagian dari genre RPG yang populer adalah *action RPG* yang mana pemain memerankan sebuah karakter dan menjalankan cerita dari permainan tersebut seperti pada Gambar 2.4.

Pada dasarnya *turn-based* RPG (Panumate et al., 2015) adalah bagian dari genre RPG. Pada *turn-based* RPG, pertarungan dilakukan secara bergantian yang mana pemain memerintahkan karakternya untuk melakukan berbagai tindakan dalam mengalahkan lawan. Jika dilihat maka *turn-based* RPG adalah kombinasi dari permainan *turn-based* dan RPG. Biasanya game dijalankan dengan model yang sama seperti RPG biasa. Saat terjadinya pertarungan dengan musuh, mode permainan akan berubah dari sistem RPG men-



Gambar 2.4: Contoh permainan digital dengan genre *RPG*.

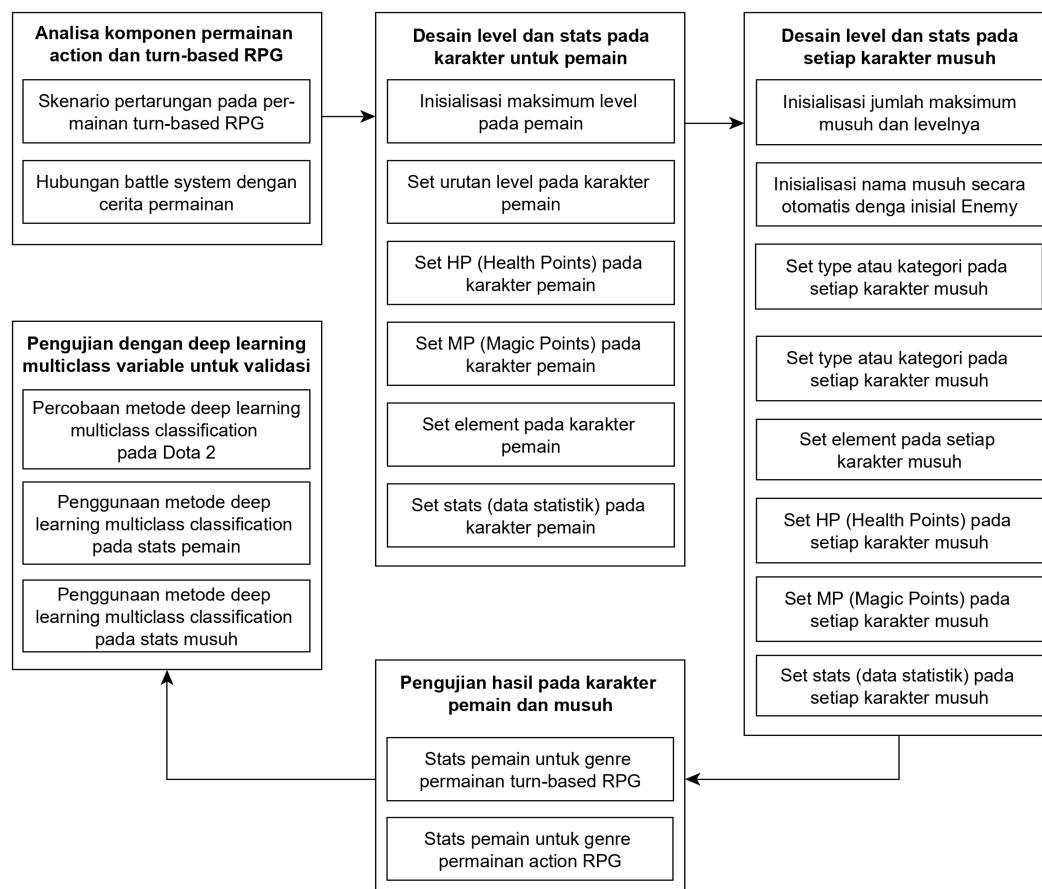
jadi sistem *turn-based*, dengan memilih tindakan yang ingin dilakukan pada setiap giliran seperti serangan, pertahanan dan sebagainya.

Inti dari mode pertarungan atau *battle stage* tersebut adalah sama seperti pada permainan *turn-based* pada umumnya. Saat pertempuran berakhir, permainan akan berubah dari sistem *turn-based* menjadi sistem RPG. Kemudian saat *mode* RPG seseorang harus mengikuti cerita seperti melakukan pencarian atau menemukan barang dan segala sesuatu yang dilakukan dalam sistem RPG. Hal tersebut mempengaruhi setiap attribut atau fungsional yang dipakai saat mode pertempuran dengan sistem *turn-based*.

BAB 3

METODOLOGI

Pada penelitian ini nantinya akan terdiri dari lima langkah utama yaitu analisa komponen permainan *action* dan *turn-based RPG*, desain level *stats* atau data statistik pada karakter pemain, desain level dan *stats* pada setiap karakter musuh, dilanjutkan pengujian dengan berbagai parameter untuk *stats* pemain, kemudian dilakukan lagi pengujian untuk validasi dengan *deep learning multiclass classification*. Dilanjutkan dengan beberapa langkah dibawahnya yang merupakan langkah-langkah detail dalam pengerjaanya seperti pada Gambar 3.1.



Gambar 3.1: Urutan metodologi.

Tujuan umum dari penelitian ini adalah membuat *stats* atau statistik untuk pemain dan musuh, sehingga dalam pembuatan permainan. Tujuan khusus dan fokus pada penelitian ini adalah untuk membuat data statistik untuk karakter pemain dan musuh dengan menggunakan metode *k*-NN dan *Naive Bayes* yang kemudian dilakukan validasi dengan menggunakan metode *Deep Learning (Multiclass Classification)*.

3.1 Analisa Komponen Permainan Action dan Turn-based Role-Playing Game (RPG)

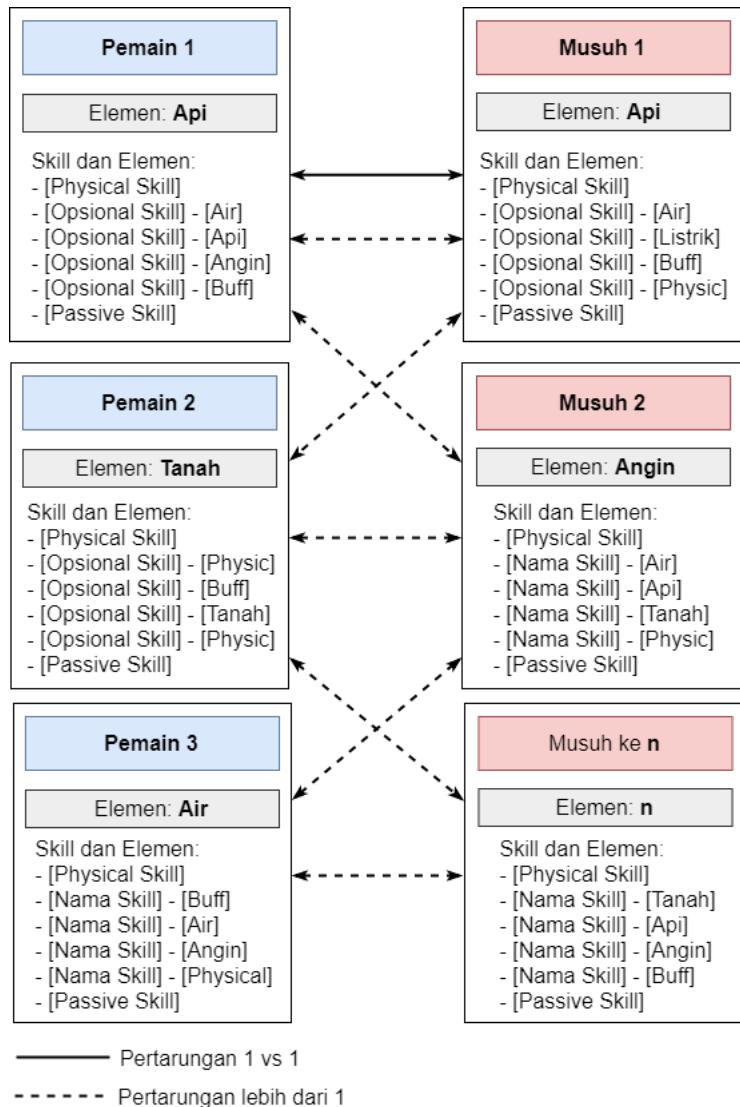
Pada bagian ini terdapat beberapa langkah yang akan menjelaskan penyusunan skenario pertarungan pada permainan RPG *turn-based*. Pertama yang akan dilakukan adalah pembuatan skenario pertarungan pada permainan *turn-based* RPG. Dari skenario tersebut tentunya harus ada hubungan dengan cerita dari permainan, apa saja parameter yang berpengaruh dari cerita terhadap skenario.

Di lanjutkan dengan desain level dan *stats* dari pemain, selain itu digunakan algoritma *k*-*Nearest Neighbor* (*k*-NN) untuk mempercepat proses pembuatannya. Kemudian dilanjutkan dengan desain level dan *stats* musuh yang juga dibuat secara otomatis dengan algoritma yang sama dengan pemain, namun tetap dipolakan oleh *gaussian naive bayes* atau distribusi normal.

Bagian selanjutnya adalah penambahan elemen pada pemain dan musuh yang membentuk kelebihan atau kekurangan pada masing-masing karakter. Pembagian elemen pada karakter yang dapat dimainkan oleh pemain dilakukan sesuai dengan cerita, sedangkan pembagian elemen pada musuh disebar secara acak berdasarkan *stats* yang telah dibuat. Hal ini berkaitan erat dengan penjelasan pada Sub-bab 2.2.1 tentang meningkatnya keberagaman, yang dibuktikan dengan banyaknya variasi musuh berikut dengan kelemahan dan kelebihannya.

3.1.1 Desain Skenario Pertarungan

Diumpamakan jumlah karakter yang rencananya akan digunakan berjumlah satu (*action RPG*) sampai tiga karakter (*turn-based RPG*) bergantung dengan alur cerita dari permainan dan jumlah musuh juga dimisalkan berjumlah antara satu sampai dengan enam, bergantung kepada tingkat kesulitannya seperti pada Gambar 3.2.



Gambar 3.2: Skema pertarungan antar pemain.

Sebelumnya pernah dijelaskan pada Sub-bab 2.2.1 tentang meningkatnya pengambilan keputusan, pada kondisi semakin banyak musuh mak-

an semakin banyak keputusan yang harus diambil oleh pemain. Selain itu setiap pemain memiliki elemen dan *skill*, setiap elemen dapat menjadi kelemahan dan kelebihan dari setiap karakter. Hal tersebut akan membangun sebuah momen dramatis berdasarkan kompleksitas kombinasi dari musuh, hal tersebut dibahas pada Sub-bab 2.2.1 tentang momen dramatis pada desain permainan.

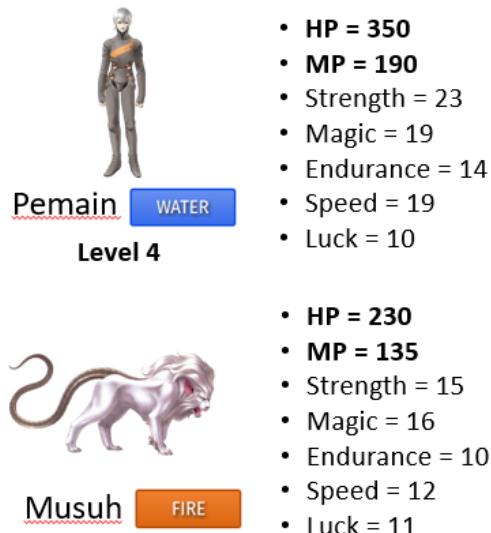
Pada bagian ini dijelaskan contoh perancangan dari skenario pertarungan dengan genre *action* dan *turn-based* yang dibandingkan secara langsung. Pada Gambar 3.2 jika dipecah dan dijelaskan lebih lanjut maka setiap karakter yang dimainkan oleh pemain atau musuh dalam bentuk NPC (*Non Playable Character*) maka bagian-bagian tersebut akan menjadi seperti beberapa poin dibawah ini.

1). Stats

Stats atau statistik yang diperhitungkan dan berpengaruh dalam proses pertarungan. Pada Gambar 1.1 ditunjukan tidak hanya karakter pemain saja, melainkan status dari pemain saat melakukan pertarungan. Pada Gambar 3.3 adalah penggambaran dari komponen atau *stats* pemain yang lebih detail seperti *Health Point* (HP), *Attack* atau serangan, *Defense* atau pertahanan, *Speed* atau kecepatan.

Berikut adalah penjabaran lebih dari beberapa komponen seperti HP, *Attack*, *Defense*, *Speed* yang terdapat pada Gambar 1.1.

- a). **Health Point (HP)** adalah indikator nyawa atau kehidupan dari pemain, jika HP bernilai 0 maka karakter tersebut akan mati atau kalah.
- b). **Magic Point (MP)** adalah indikator jumlah dari jurus yang dapat dikeluarkan oleh pemain, jika MP bernilai 0 maka karakter tersebut tidak bisa mengeluarkan jurus atau kemampuan khusus.
- c). **Strength** adalah jumlah atau nilai serangan yang akan dilakukan



Gambar 3.3: Status dari pemain pada permainan *turn-based*.

an untuk mengalahkan pemain lawan. Angka tersebut akan berlawanan atau dibandingkan dengan jumlah *endurance* musuh. Hal tersebut bertujuan untuk mengurangi HP dari musuh.

- d). **Magic** atau *Special Attack* biasanya tidak dimiliki oleh semua karakter pada permainan berbasis *turn-based*. *Special Attack* biasanya menjadi pembeda dalam setiap karakter berdasarkan jenis serangannya. Misalkan pada *strength* biasanya berupa serangan fisik sedangkan pada *special attack* berupa serangan *magic* atau sihir.
- e). **Endurance** adalah jumlah atau nilai ketahanan yang digunakan oleh pemain atau musuh dalam menerima serangan. Hal ini bertujuan agar mencegah penurunan HP secara signifikan, dengan membandingkan nilai serangan dengan nilai pertahanan.
- f). **Speed** atau kecepatan ada juga yang menyebutnya dengan *agility* bertujuan dalam menentukan giliran dan keberhasilan dalam melakukan serangan. Semakin tinggi nilai *speed*, biasanya semakin cepat melakukan serangan atau dapat mulai melakukan serangan lebih awal dibandingkan pemain atau musuh dengan nilai *speed* yang lebih kecil.

g). **Luck** atau keberuntungan adalah sebuah variabel yang digunakan menentukan hal yang bersifat acak, seperti bonus serangan atau *critical attack*, kesempatan saat melakukan serangan balik atau *counter attack* setelah diserang.

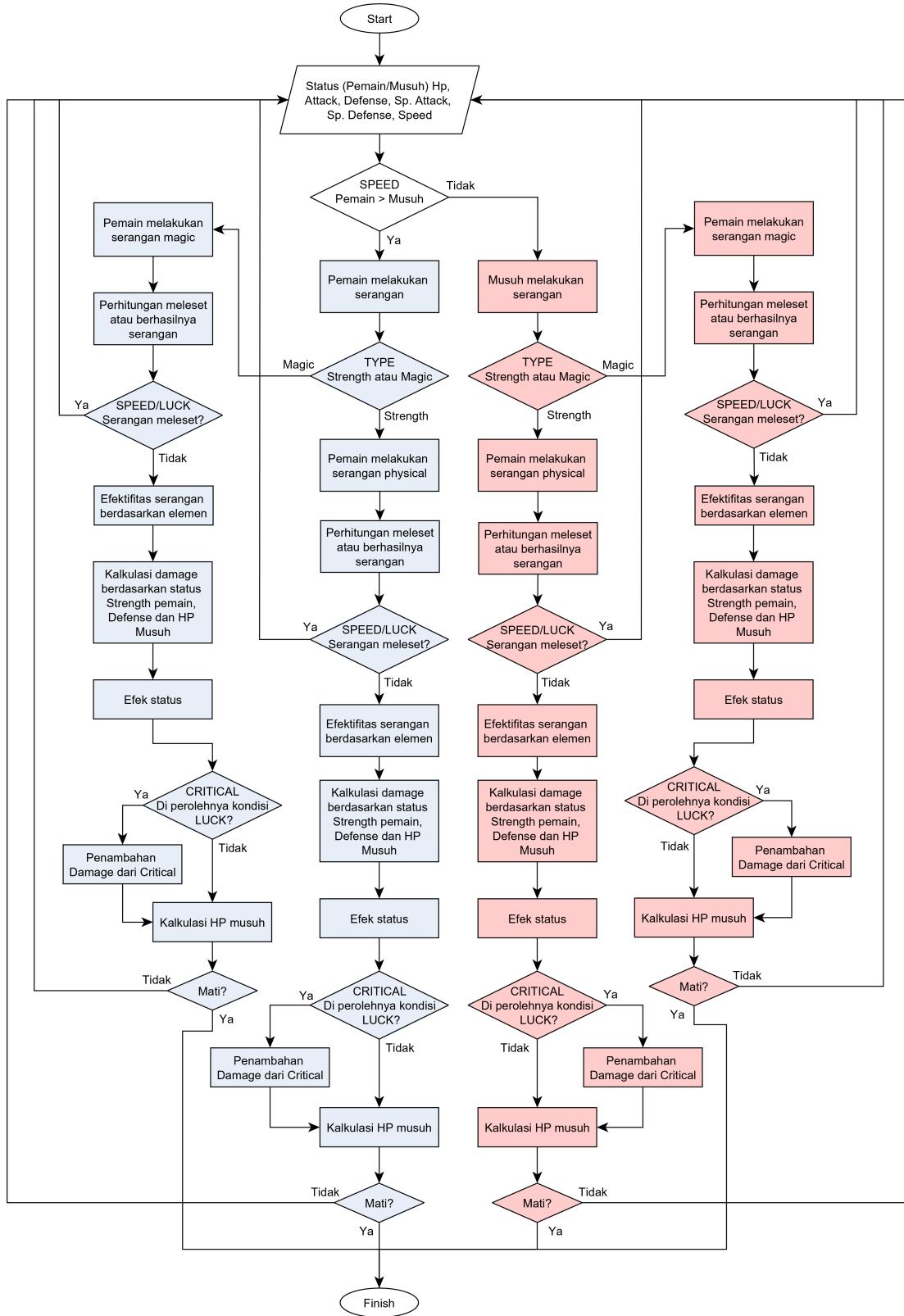
Sementara itu skenario pertarungan dengan melibatkan *stats* dijelaskan pada Gambar 3.4 yang mana pemain atau musuh melakukan serangan berdasarkan *speed*. Semakin tinggi *speed* maka akan memperoleh giliran pertama untuk menyerang.

Di lanjutkan lagi dengan perhitungan dengan membandingkan *Speed*, dan *Luck* antara karakter pemain dengan musuh, pada proses tersebutlah yang menentukan apakah serangan dari pemain dapat diterima atau meleset terhadap musuh dan sebaliknya. Tingginya *Speed* pada masing-masing karakter dapat diartikan perbandingan antara kecepatan untuk menyerang dan menghindar, sedangkan *Luck* adalah faktor keberuntungan yang mempengaruhi serangan tersebut. Kemudian dilanjutkan lagi dengan kalkulasi *attack* dan *defense* antara karakter yang menyerang dan target. Dari hasil kalkulasi tersebut akan berpengaruh pada jumlah HP dari karakter yang menjadi target.

Kemudian pada permainan RPG dengan genre *action* tidak membutuhkan skema berurutan serumit *turn-based*. Pada permainan tersebut lebih mengandalkan ketangkasan dari pemain dalam memainkan karakter yang ingin dimainkan seperti yang dijelaskan pada Sub-bab 2.2.2 pada bagian Peran dari keterampilan pada permainan. Maka terjadilah momen saling serang secara langsung antara pemain dan musuh.

2). Elemen dan Efektifitas Serangan

Pada Gambar 3.5 adalah contoh elemen yang digunakan dalam permainan dengan mode *turn-based*. Jumlah dari elemen dapat di-



Gambar 3.4: Skenario pertarungan *turn-based*.

tambah atau dikurangi sesuai dengan kebutuhan. Biasanya jumlah dari elemen ditentukan berdasarkan cerita. Mengapa terdapat elemen tersebut, bagaimana asal-usulnya dan sebagainya seperti yang dijelaskan pada Sub-bab 3.1.2.



Gambar 3.5: Elemen pada permainan dengan mode pertarungan *turn-based*.

Pembahasan ini mengacu kepada Gambar 3.4 pada bagian “efektifitas serangan berdasarkan elemen”. Sedangkan pada Gambar 3.6 adalah perbandingan pengaruh atau keterkaitan sebuah elemen dengan elemen yang lain. Di mana setiap elemen memiliki kelemahan yang berupa elemen lain dan sebaliknya. Elemen-elemen tersebut saling berlawanan satu dengan yang lainnya sehingga mampu membentuk sebuah permainan yang membutuhkan strategi khusus. Kemudian dilanjutkan dengan perhitungan *stats* seperti bagian sebelumnya.



Gambar 3.6: Pengaruh elemen pada efektifitas serangan.

Jika melihat pada Gambar 3.6 dapat disimpulkan bahwa beberapa element yang saling berlawanan kemudian efektifitas menjadi 2 kali lipat, contohnya pada elemen air terhadap api dan tanah terhadap angin begitu juga sebaliknya. Jika elemen yang sama saling bertarung maka efektifitasnya berkang 1/2 dari yang seharunya.

Kemudian pada elemen lain yang belum disebutkan berlaku efektifitas normal atau 1 kali. Pembagian elemen pada pemain dan musuh akan dibahas lebih detail pada Sub-bab 3.2.

Kebanyakan elemen dan efektifitas serangan berlaku pada permainan RPG dengan genre *turn-based*, berbeda halnya dengan *action* yang lebih mengandalkan keterampilan dari pemain dalam memaikan karakternya seperti yang dijelaskan pada Sub-bab 2.2.2 pada bagian pemain berbasis keterampilan sepenuhnya.

3). Efek Status

Pada Gambar 3.4 terdapat bagian “efek status”, maksud dari proses tersebut adalah penambahan efek yang merugikan terhadap pemain setelah diserang. Efek kerusakan dapat membawa kesan lebih taktis pada pertempuran. Berikut adalah efek status yang akan diterapkan pada sistem pertarungan.

- a). **Infected:** Karakter kehilangan 10% dari total HP setiap giliran.
- b). **Confused:** Karakter tidak dapat dikendalikan dan mungkin akan bertahan, menyerang dan tidak melakukan apa-apa. Ada juga kemungkinan mereka akan berbalik menyerang *party member* mereka sendiri.
- c). **Silence:** Karakter tidak dapat mengubah Personae atau menggunakan keterampilan Persona.
- d). **Tired:** Karakter kehilangan SP untuk setiap tindakan yang diambil, dan menerima lebih banyak *damage* dari musuh ketika diserang.
- e). **Hacked:** Efek yang muncul setelah target diserang sesuai dengan kelemahannya. Target kemudian akan menerima lebih banyak *damage* dan tidak dapat menghindar. Jika target diserang lagi, maka statusnya akan berubah menjadi disabled.
- f). **Disabled:** Target akan hilang 1 giliran untuk mengambil tin-

dakan, kemudian mendapat lebih banyak kerusakan dan serangan tidak dapat dihindari.

Tidak semua kemampuan pemain dapat memberi efek status, hal tersebut mengacu pada desain permainan yang mengatur keseluruhan *skill*, tidak hanya pada karakter utama melainkan juga pada musuh. Tetapi dalam penelitian ini, hal tersebut masih belum terpakai dikarenakan masih menyelesaikan perihal desain *stats* dari pemain dan musuh. Hal ini baru semacam perkiraan saja saat mendesain sebuah permainan.

Pada bagian efek status juga berlaku pada permainan RPG dengan genre *action*, setelah berlangsungnya pertarungan antara pemain dan musuh. Di mana pada sisi pemain dapat membangun karakternya sedemikian hingga demi memberi efek status ke pada musuh saat berlangsungnya pertarungan seperti yang dijelaskan pada Sub-bab 2.2.2 tentang peran dan keterampilan pemain serta strategi dan taktik.

4). Kondisi Kritis pada Serangan

Selain *attack*, elemen dan efek status, masih terdapat *damage* yang dapat ditimbulkan oleh penyerang kepada target yaitu kondisi kritis pada serangan. Dengan jumlah *attack* ditambah dengan jumlah *attack* yang dikalikan dengan persentase tingkat kondisi kritis (*critical rate*) pada *skill* yang dipilih untuk menyerang lawan. Kondisi tersebut didapat dengan membandingkan nilai *stats* dari *Strength* atau *Magic* dan *luck* milik penyerang dan target, pada proses tersebutlah yang menentukan apakah serangan tersebut diperoleh kondisi kritis atau tidak. Hal tersebut juga membangun sebuah momen dramatis berdasarkan peluang terjadinya kondisi kritis saat terjadinya serangan dari pemain terlebih lagi dari musuh seperti yang dibahas pada Sub-bab 2.2.1 tentang momen dramatis.

5). Kalkulasi HP dan MP

Pada akhirnya jumlah kerusakan yang ditimbulkan oleh penyerang dan HP dari target akan dikalkulasikan. Jika HP target habis atau sama dengan 0, maka terget tersebut dinyatakan mati. Dan jika HP target masih bersisa, maka pertarungan akan dilanjutkan oleh giliran karakter dari pemain atau musuh selanjutnya. Kemudian pada sisi penyerang juga ada yang dikorbankan dalam upaya melakukan serangan. Saat penyerang memilih serangan fisik maka pemain akan mengorbanakan sebagian HP yang dimiliki, jika yang dipilih adalah serangan *magic* maka yang dikorbankan adalah MP.

3.1.2 Hubungan Sistem Pertarungan dengan Cerita

Pada setiap permainan RPG khususnya *action* dan *turn-based* RPG pasti memiliki cerita yang menjadi latar belakang permainan seperti yang dijelaskan pada Sub-bab 2.4. Tentunya cerita tersebut juga memiliki pengaruh penting terhadap jumlah karakter yang dapat dimainkan oleh pemain, jumlah musuh, elemen apa saja yang akan dipakai, total waktu permainan, jumlah musuh yang harus dilawan dan lain sebagainya. Pada penelitian ini hal semacam itu akan dibuat menjadi sebuah estimasi yang kemudian disimulasikan ke dalam mode pertarungan dengan berbagai macam estimasi sebagaimana penjelasan berikut.

1). Tingkat Kesulitan Musuh

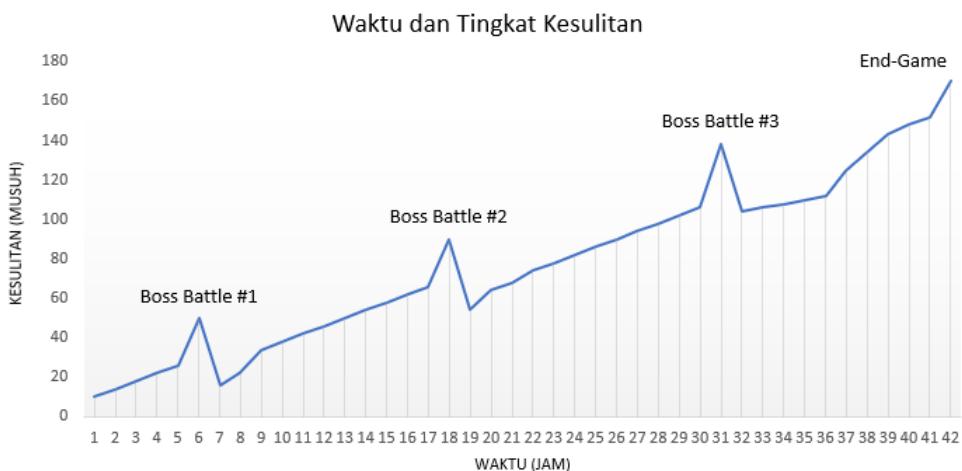
Berikut adalah beberapa pertanyaan yang harus ditanyakan kepada setiap pembuat desain permainan:

- a). Haruskah kebanyakan pemain nantinya dapat menyelesaikan permainan tanpa melakukan *side quest* (misi sampingan) atau melakukan *grinding* (menaikkan kemampuan karakter) diluar standar perkembangannya?
- b). Berapa banyak bos yang akan dilawan oleh pemain pada permainan ini, dan seberapa jauh jaraknya? Dan bagaimana dengan

penambahan bos?

- c). Berapa banyak *dungeon* (tempat muncul dan bertemu dengan musuh) yang akan disajikan, dan seberapa besar *dungeon* tersebut?
- d). Akankah pemain bisa menyimpan progres permainan kapan saja, atau hanya di titik penyimpanan tertentu yang sudah ditentukan?

Biasanya pada permainan *turn-based* RPG terdapat sebuah peta besar tentang lokasi yang merupakan latar dari cerita, tersebarlah berbagai jenis musuh yang relatif mudah dikalahkan. Kemudian ditampilkan juga beberapa *dungeon* yang akan terbuka satu demi satu, yang mana *dungeon* tersebut memiliki tingkat kesulitan dan kerumitan yang terus meningkat sampai dengan bertemu dengan Bos. Secara keseluruhan tingkat kesulitan juga akan terus meningkat sampai dengan akhir permainan seperti yang dicontohkan oleh Gambar 3.7.



Gambar 3.7: Pengaruh cerita terhadap tingkat kesulitan.

Adapun beberapa cara untuk meminimalisir *grinding* dan lamanya waktu permainan, dengan diberikan Exp (*Experience* adalah sebuah variabel untuk pemain agar naik level) kepada pemain untuk menyelesaikan misi dan mengalahkan musuh yang lebih sulit dari pa-

da mengalahkan musuh yang relatif mudah ditaklukan yang tersebar pada peta. Tentu saja musuh yang tersebar di peta juga memberikan Exp bagi pemain, namun seiring bertambahnya level pemain maka Exp yang diperoleh saat melawan musuh dengan level rendah akan semakin kecil.

Pada penelitian ini tingkat kesulitan langsung disimulasikan dengan pertarungan antara karakter-karakter yang dimainkan oleh pemain melawan musuh, dengan kondisi tingkat kesulitan musuh yang terus naik lalu turun kemudian naik lagi dan turun lagi, naik lagi dan seterusnya sampai dengan kondisi puncak. Hal ini mensimulasikan kondisi yang dilalui oleh pemain saat melawan *trash mobs*, memasuki *dungeon*, saat bertarung melawan bos dan kemudian pada akhirnya bertarung melawan bos terakhir. Lebih detailnya akan dijelaskan pada poin selanjutnya.

2). Waktu yang Diperlukan untuk Kalahkan Musuh

Musuh pada permainan *turn-based RPG* umumnya terbagi menjadi empat kategori diantaranya adalah:

- A). *Trash Mobs* adalah musuh yang tersebar pada seluruh area atau map.
- B). *Dungeon Mobs* dapat dibagi menjadi dua sub-kategori:
 - a). *Dungeon Trash* atau sama seperti *Trash Mobs* yang sebagian besar ditemukan awal sampai tengah *dungeon*.
 - b). *Difficult Dungeon Trash* atau yang lebih sulit terletak lebih dekat ke bos atau dari tengah ke akhir *dungeon*.
- C). *Mini-Boss/Boss Mobs*.
- D). *End-Game Boss/Secret Boss* (Bos yang bersifat opsional).

Pada penelitian ini keseimbangan permainan dirancang terus meningkat seperti yang dibahas pada bagian sebelumnya, dalam permainan ini sebagian besar waktu (asumsikan saja 80%) akan diha-

biskan bertarung di dalam *dungeon*. Kemudian 20% dari waktu pertempuran akan digunakan untuk bertarung melawan bos. Sedangkan sisanya 60% dari waktu bertarung akan dibagi antara pertarungan melawan musuh yang lemah dan juga kuat, bila digambarkan pembagian tersebut akan seperti pada Gambar 3.8.



Gambar 3.8: Distribusi jenis musuh sesuai dengan cerita.

Perhatikan bahwa dalam distribusi yang dibuat, jumlah waktu yang dihabiskan untuk melawan bos sama seperti waktu yang dibutuhkan untuk memerangi *Trash Mobs*. Dalam proses replikasi distribusi dalam permainan, pertama tentukan jumlah total waktu yang dibutuhkan oleh pemain untuk mengalahkan naga, raksasa atau apa pun yang biasanya disebut dengan bos.

Misalnya, dalam permainan RPG, bos pertama idealnya akan membutuhkan 3 menit bagi pemain yang kompeten untuk mengalahkan, dan bos terakhir menghabiskan waktu 20 menit. Kemudian terjadi peningkatan kompleksitas pada bos di level menengah, untuk terdapat dua bos yang masing-masing membutuhkan waktu sekitar 10 menit untuk dikalahkan, dan bos kedua membutuhkan waktu 7 menit.

$$B_{total} = \sum_{n=0}^k B_n \quad (3.1)$$

Merujuk ke persamaan 3.1 bila dijabarkan maka B_{total} adalah jumlah waktu melawan bos secara keseluruhan, jumlah bos dinyata-

kan dengan k dan waktu yang dihabiskan untuk melawan satu bos dinyatakan dengan B kemudian diiterasi oleh n sejumlah k .

Di perlukannya penyesuaian terhadap model distribusi, salah satunya waktu yang habis untuk melawan bos setara untuk melawan musuh yang mudah atau *trash mobs*. Untung saja terdapat banyak cara untuk memodifikasi waktu yang akan dihabiskan saat bertarung melawan musuh yang mudah. Hal seperti menjelajah seluruh peta atau berpetualang menuju tempat-tempat sebelumnya juga tidak perlu dilakukan. Berikut adalah langkah-langkah yang dapat dilakukan:

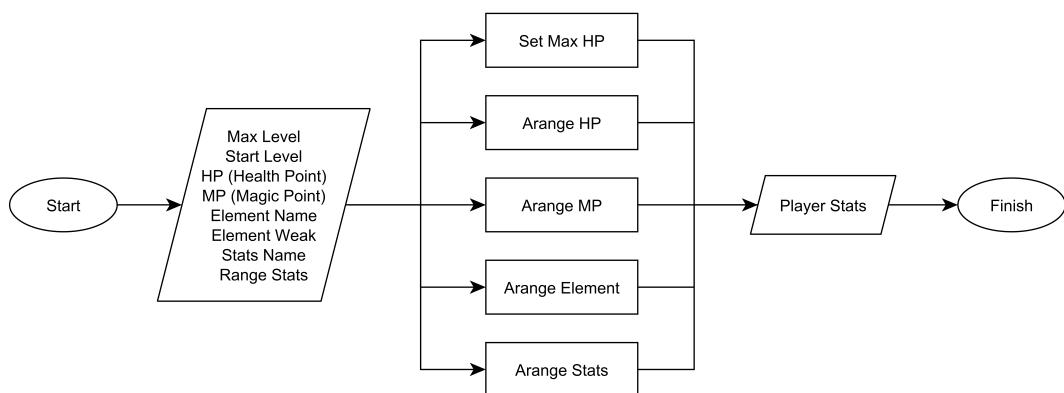
- a). Mengurangi atau meningkatkan tingkat pertemuan dengan musuh yang mucul secara acak atau yang biasa disebut dengan *random encounter rate*. Bisa juga dengan tingkat *spawn* (muncul lagi setelah mati).
- b). Menambah atau mengurangi jumlah musuh saat pertempuran.
- c). Membatasi kemampuan musuh yang menimbulkan efek status yang menyulitkan dan menghabiskan waktu seperti *confused*, *silence*, *tired* dan lain-lain.
- d). Menambah atau mengurangi kekuatan musuh.
- e). Menambah atau mengurangi kekuatan dari *party member*.

Terdapat banyak fleksibilitas di sini, developer dapat mengisi daftar musuh yang akan muncul secara acak dengan musuh yang sulit dikalahkan dengan tingkat probabilitas kemunculan yang kecil, atau lebih sering memunculkan musuh yang mudah dikalahkan. Alternatif lain adalah dengan meningkatkan kekuatan *party member* atau jumlah rata-rata musuh yang bertarung dalam satu kali pertarungan. Kebebasan desain semacam ini yang nantinya akan memudahkan penyeimbangan permainan. Sedangkan jumlah musuh dan panjangnya level dari pemain atau musuh sendiri menggambarkan akan lamanya permainan tersebut. Pada dasarnya semua proses diatas mengacu pada pokok

pembahasan dari referensi yang dibahas pada Sub-bab 2.2.3 tentang menemukan keseimbangan.

3.2 Desain Level dan Stats pada Karakter Pemain

Di buatlah sebuah program yang secara otomatis dapat membuat *stats* dari pemain dengan masukan sesuai dengan kebutuhan desainer permainan atau pengembang. Program tersebut terdiri dari beberapa fungsi yang pada awalnya adalah obyek yang memiliki masukan parameter-parameter yang nantinya akan menghasilkan sebuah data yang berupa *stats* dari karakter pemain seperti proses yang ditunjukkan oleh diagram alur pada Gambar 3.9.

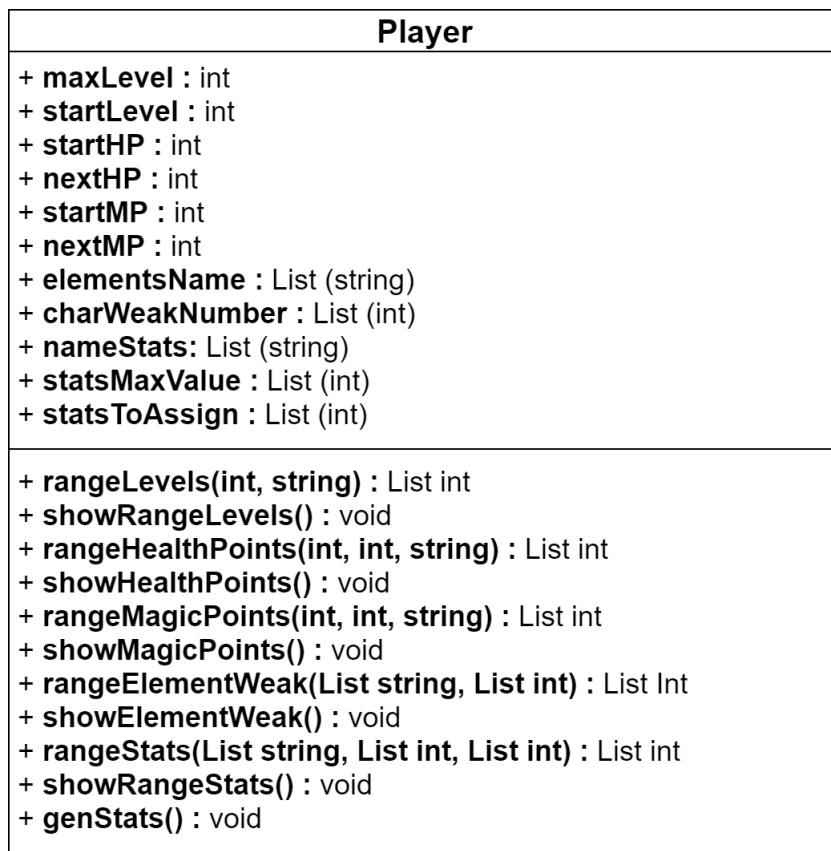


Gambar 3.9: Proses pembuatan stats untuk karakter pemain.

Pada permainan dengan genre *turn-based* RPG dengan jumlah karakter yang dapat dimainkan berjumlah lebih dari satu, maka program yang ditunjukkan melalui proses pada Gambar 3.9 akan dijalankan lebih dari satu kali. Hal tersebut dilakukan dengan tujuan agar karakter utama atau yang dapat dimainkan oleh pemain berjumlah lebih dari satu. Lain halnya dengan *action* RPG, cukup hanya dengan satu kali menjalankan program maka sudah diperolehnya *stats* dari karakter yang dibuat. Hal tersebut dikarenakan biasanya permainan dengan genre action RPG, jumlah karakternya hanya satu.

Pada Gambar 3.9 disisi masukan program terdapat banyak sekali masukan variabel seperti *Max Level* yang berupa maksimum level yang di inginkan, kemudian *Start Level* yang berupa level awal dari pemain, kemudian HP yang

berupa *range* atau jarak antara HP terendah dengan HP terendah setelah itu. Sama halnya dengan HP, dalam perhitungan *range* atau jarak pada *MP* juga menggunakan cara tersebut. Kemudian *Element Name* berisi daftar elemen apa saja yang ingin digunakan, begitu juga dengan *Name Stats*. Kemudian untuk *Range Stats* berisikan *stats* awal atau inisialisasi dan maksimum *stats*. Lebih detail tentang program yang dibuat dapat dilihat pada Gambar 3.10, yang merupakan *class* diagram untuk membuat *stats* pemain.



Gambar 3.10: *Class* diagram untuk *stats* pemain.

Karena program ini dibangun menggunakan OOP (*Object Oriented Programming*) maka program ini dapat dijabarkan menggunakan Gambar 3.10. Selain itu program ini juga dapat secara mudah dimodifikasi untuk keperluan pengembangan kedepannya, dengan fungsi-fungsi yang ada sangat memungkinkan dilakukan *override* atau pembuatan fungsi yang sama dengan isi atau proses yang berbeda.

Dalam menjelaskan proses pada BAB ini maka diambilah sebuah kasus dalam desain permainan, khususnya dalam penyusunan *stats* untuk pemain dengan *genre* permainan *turn-based* dan *action RPG*. Pada Tabel 3.1 adalah masukan untuk menguji program yang akan dijelaskan pada bagian selanjutnya, yang mana masukan pada Tabel 3.1 akan menghasilkan *stats* pada sebuah karakter untuk pemain. Hal ini merupakan perwujudan dari Sub-bab 3.1.1 bila diwujudkan dalam bentuk yang lebih teknis.

Tabel 3.1: Data masukan untuk pembuatan *stats* pemain.

Variabel	Input
<i>Max Level</i>	100
<i>Start Level</i>	1
<i>Start HP</i>	159
<i>Next HP</i>	163
<i>Start MP</i>	89
<i>Next MP</i>	93
<i>List Element</i>	[‘Phys’, ‘Water’, ‘Wind’, ‘Earth’, ‘Fire’]
<i>List Weaknessess</i>	[1, 2, 1, 1, 0]
<i>List Stats Name</i>	[‘Strength’, ‘Magic’, ‘Endurance’, ‘Speed’, ‘Luck’]
<i>Max Stats Value</i>	[74, 38, 63, 65, 60]
<i>Stats to Assign</i>	[2, 1]

3.2.1 Distribusi Level, HP dan MP Pemain

Berdasarkan Tabel 3.1 level untuk karakter pemain dimulai dari 1 dan level maksimalnya adalah 100, pada program ini level pemain akan terus naik satu demi satu level sampai ke tingkat maksimal. Selanjutnya adalah HP atau *Health Point* yang diberi masukan berupa *start HP*, bisa dinyatakan juga sebagai HP saat level satu. Kemudian variabel lanjutannya adalah *next HP* atau HP pada level selanjutnya, misalkan level dua. Muncul sebuah pertanyaan berapa HP selanjutnya sampai dengan level ke 100.

Cara semacam itu juga berlaku untuk perhitungan MP pada program ini, dengan pola masukan yang sama dengan HP yaitu *Start* MP dan *Next* MP. Pada persamaan 3.2 dan 3.3 adalah contoh persamaan yang digunakan untuk mencari nilai HP dan MP selanjutnya.

$$HP(N) = \sum_{n=0}^N HP(n+1) + (HP(n+1) - HP(n)) \quad (3.2)$$

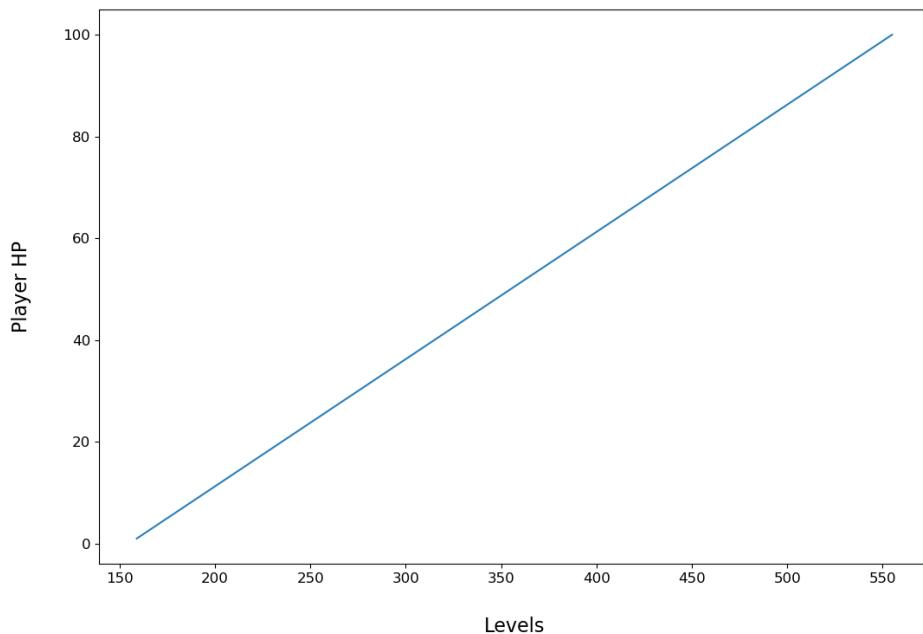
$$MP(N) = \sum_{n=0}^N MP(n+1) + (MP(n+1) - MP(n)) \quad (3.3)$$

Pada persamaan 3.2 dan persamaan 3.3 HP tetap dinyatakan sebagai HP, begitu juga dengan MP. Kemudian $HP(N)$ adalah nilai HP yang dicari, yang mana N adalah level maksimum, n adalah level mulai dan $n+1$ adalah level selanjutnya. Jadi pada tahap ini seperti yang dijelaskan pada Tabel 3.1 yang mana nilai n dan $n+1$ sudah diketahui sebagai inisialisasi, masing-masing adalah $HP(n)$ dan $HP(n+1)$. Penjelasan ini juga berlaku untuk mencari nilai $MP(N)$ yaitu nilai MP yang dicari. Hasil contoh perhitungan dari persamaan 3.2 dan 3.3 dengan masukan dari Tabel 3.1 dapat hasilnya dapat dilihat pada Tabel 3.2.

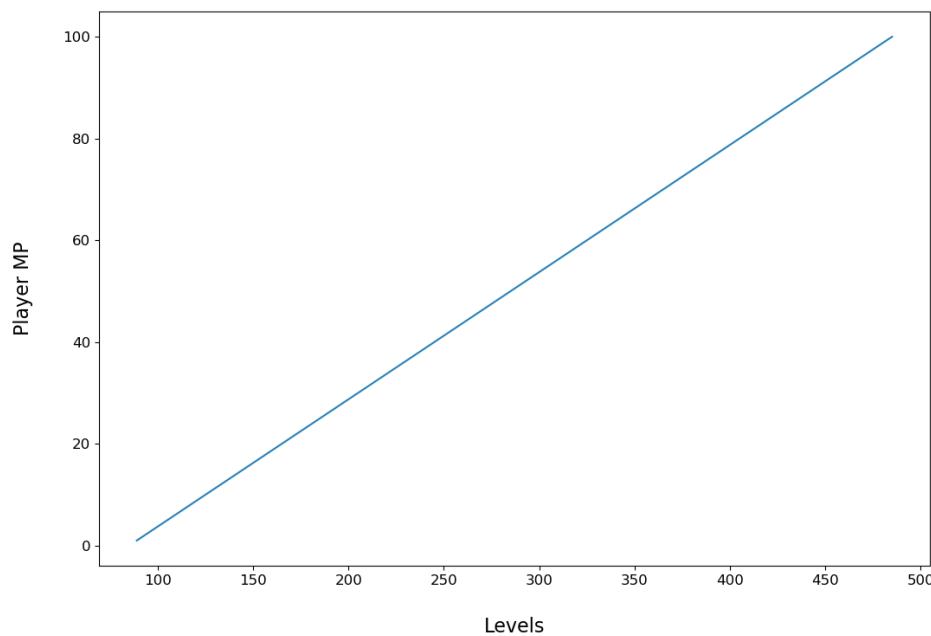
Tabel 3.2: Hasil Perhitungan HP dan MP

Levels	HP	MP
1	159	89
2	163	93
3	167	97
4	171	101
5	175	105
6	179	109
7	183	113
...
100	555	485

Hasil perhitungan tersebut terlihat membentuk pola linier, yang mana nilai HP dan MP terus naik secara konstan ke atas sesuai dengan kenaikan levelnya seperti yang direpresentasikan pada Gambar 3.11 dan Gambar 3.12.



Gambar 3.11: Kenaikan HP setiap levelnya.



Gambar 3.12: Kenaikan MP setiap levelnya.

Jika melihat Gambar 3.11 dengan jumlah HP dari pemain yang terus naik mengikuti pola yang dihasilkan pada Tabel 3.2, yang mana nilai tersebut diperoleh dari inisiasi variabel pada Tabel 3.1 yaitu *Max Level*, *Start HP* dan *Next HP*. Variabel-variabel tersebut dihitung dengan menggunakan persamaan 3.2 agar membentuk pola kenaikan HP setiap levelnya seperti yang ditunjukan pada Gambar 3.11.

Sama seperti pada Gambar 3.11, pada Gambar 3.12 dengan jumlah MP dari pemain yang terus naik mengikuti pola yang dihasilkan pada Tabel 3.2, yang mana nilai tersebut diperoleh dari inisiasi variabel pada Tabel 3.1 yaitu *Max Level*, *Start MP* dan *Next MP*. Kemudian variabel-variabel tersebut dihitung dengan menggunakan persamaan 3.3 agar membentuk pola kenaikan MP setiap levelnya seperti yang ditunjukan pada Gambar 3.12.

3.2.2 Distribusi Elemen dan Kelemahan Pemain

Kemudian untuk variabel *List Element* berisi elemen apa saja yang akan diterapkan pada permainan tersebut, seperti yang ditunjukan oleh Tabel 3.1. Yang mana maksud dari variabel ini adalah memberi penjelasan dari kelemahan dan keunggulan dari pemain berdasarkan elemen, seperti yang sudah dijelaskan pada Sub-bab 3.1.1 tentang elemen dan efektifitas serangan. Dilanjutkan dengan variabel *List Weaknesses* yang memuat angka-angka yang bertujuan menggambarkan saat pemain menerima serangan dari lawan, berikut adalah penjelasan dari angka-angka tersebut.

- a). **Angka 0** adalah *Normal* atau efek serangan bersifat normal tanpa tambahan bonus serangan dan lain sebagainya.
- b). **Angka 1** adalah *Repel* atau memiliki sifat menghindari serangan atau bahkan menghindari serangan.
- c). **Angka 2** adalah *Weaknessess* atau serangan tepat menyerang terhadap kelemahan dari pemain sehingga efek kerusakan atau *damage* menjadi lebih terasa, biasanya dua kali serangan normal.

Pembagian elemen pada pemain bersifat pada penelitian ini dibuat statis, maksudnya elemen yang dari awal didefinisikan tidak akan berubah sampai akhir level, untuk kedepannya hal seperti inilah yang akan menjadi konsetrasi pengembangan program ini berikut juga desainer permainan. Cukup dengan melakukan *override* maka diperolehlah fungsi baru yang bisa menghasilkan perubahan elemen di level tertentu.

Pada bagian selanjutnya adalah pembahasan mengenai pembagian *stats* jika merujuk pada Tabel 3.1 dengan variabel *List Stats Name* yang berisi nama atau info *stats* dari pemain yang akan digunakan. Kemudian diikuti dengan variabel *Max Stats Value* yang berisi nilai maksimum *stats* yang akan dihasilkan. Lebih detailnya untuk bagian ini, akan dibahas secara khusus pada bagian tersendiri yaitu pada Sub-bab 3.2.3.

3.2.3 Distribusi Stats Pemain

Pada bagian ini akan dibahas tentang pembagian *stats* dengan beracuan pada Tabel 3.1 dengan variabel *List Stats Name* yang berisi nama atau info *stats* dari pemain yang akan digunakan. Kemudian diikuti dengan variabel *Max Stats Value* yang berisi nilai maksimum *stats* yang akan dihasilkan. Pada tahap ini digunakannya metode *k*-Nearest Neighbor atau *k*-NN seperti yang sudah dijelaskan pada Sub-bab 2.3.1 dan Naive Bayes yang juga sudah dijelaskan pada Sub-bab 2.3.2. Pada persamaan 2.4 yang kemudian disesuaikan dengan pertambahan nilai *Stats to Assign* dari Tabel 3.1, yang mana nilai tersebut ditambahkan secara acak antara 2 dan 1 dengan perhitungan *class probability* pada persamaan 2.15 yang disesuaikan menjadi persamaan 3.4. Hasil proses acak tersebut juga harus dibatasi jumlahnya dengan persamaan 3.5. Maka hasil dari proses acak pada persamaan tersebut akan sangat menentukan perhitungan dari persamaan 3.6.

$$P(C_{St=1}) = \frac{C_{St=1}}{(C_{St=1} + C_{St=2})} \quad (3.4)$$

$$P(C_{St=2}) = \frac{C_{St=2}}{(C_{St=1} + C_{St=2})}$$

$$D(x, p) = \sqrt{(x - p)^2} \quad (3.5)$$

$$MaxSt = \begin{cases} \sum_{n=0}^N St_{(n)}, & \text{saat } D(x, p) \geq 2, St = 2 \\ \sum_{n=0}^N St_{(n)}, & \text{saat } D(x, p) \geq 1, St = 1 \\ 0, & \text{lainnya} \end{cases} \quad (3.6)$$

Pada persamaan 2.15, variabel $P(C_{St=1})$ adalah probabilitas munculnya *Stats To Assign* ke 1 dan $P(C_{St=2})$ adalah probabilitas munculnya *Stats To Assign* ke 2, kemudian $C_{St=1}$ adalah jumlah angka satu dan $C_{St=2}$ adalah jumlah angka satu pada *Stats To Assign*. Setiap stats hasil pengacakan tentunya juga akan dibatasi jumlahnya dengan melakukan *checking* pada maksimum stats, hal tersebut dijelaskan pada persamaan 3.5 yang mana x adalah maksimum stats, D adalah maksimum stats dan p adalah stats saat ini. Kemudian nilai $MaxSt$ pada persamaan 3.6 adalah nilai maksimum stats yang ingin dicapai dengan jumlah $St_{(n)}$ yang dihitung melalui persamaan 3.5 dan 3.4.

$$P(C_{St=n}) = \frac{C_{St=n}}{(C_{St=0} + C_{St=1} + C_{St=n})} \quad (3.7)$$

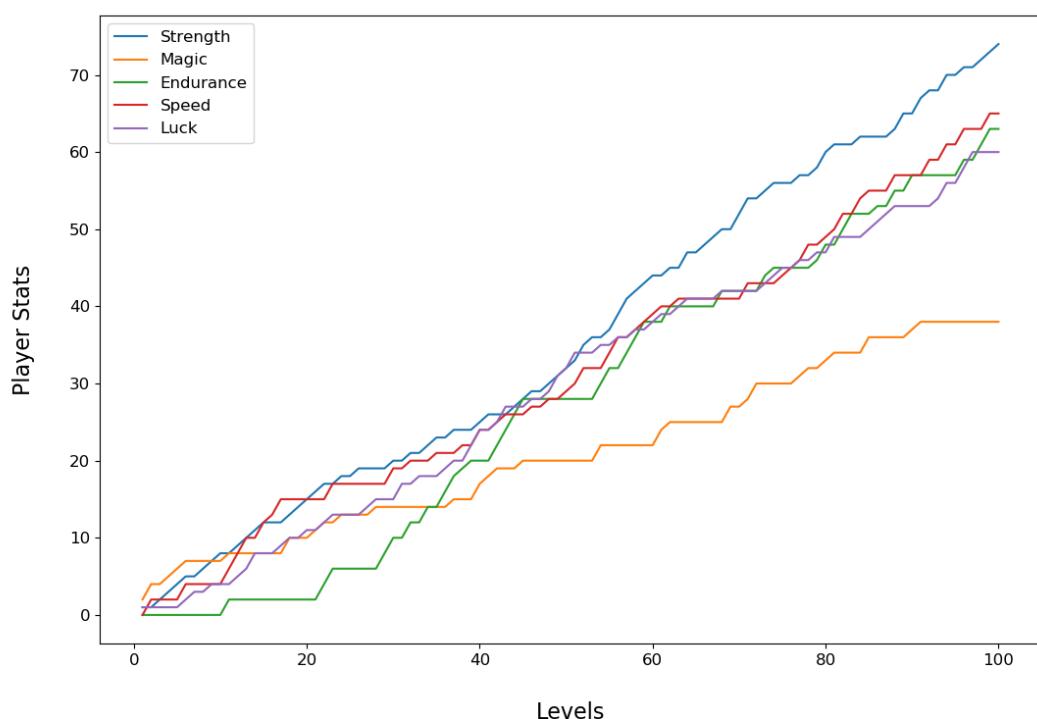
$$D(x, p) = \sqrt{(x - p)^2} \quad (3.8)$$

$$MaxSt = \begin{cases} \sum_{n=0}^N St_{(n)}, & \text{saat } D(x, p) \geq n_{st}, St = n_{st} \\ \sum_{n=0}^N St_{(n)}, & \text{saat } D(x, p) \geq 2, St = 2 \\ \sum_{n=0}^N St_{(n)}, & \text{saat } D(x, p) \geq 1, St = 1 \\ 0, & \text{lainnya} \end{cases} \quad (3.9)$$

Sedangkan pada persamaan 3.7, 3.8 dan 3.9 digunakan saat jumlah atau dimensi *Stats to Assign* lebih dari 2. Selanjutnya melalui persamaan 3.6 dan beberapa persamaan yang digunakan sebelumnya dihasilkan data seperti yang ditunjukan pada Tabel 3.3, dan bila divisualisasikan hasilnya akan tampak seperti pada Gambar 3.13.

Tabel 3.3: Sampel hasil perhitungan dan distribusi stats dengan k -NN

Levels	Strength	Magic	Endurance	Speed	Luck
1	1	2	0	0	1
2	0	2	0	2	0
3	1	0	0	0	0
4	1	1	0	0	0
5	1	1	0	0	0
6	1	1	0	2	1
7	0	0	0	0	1
8	1	0	0	0	0
...
100	1	0	0	0	0

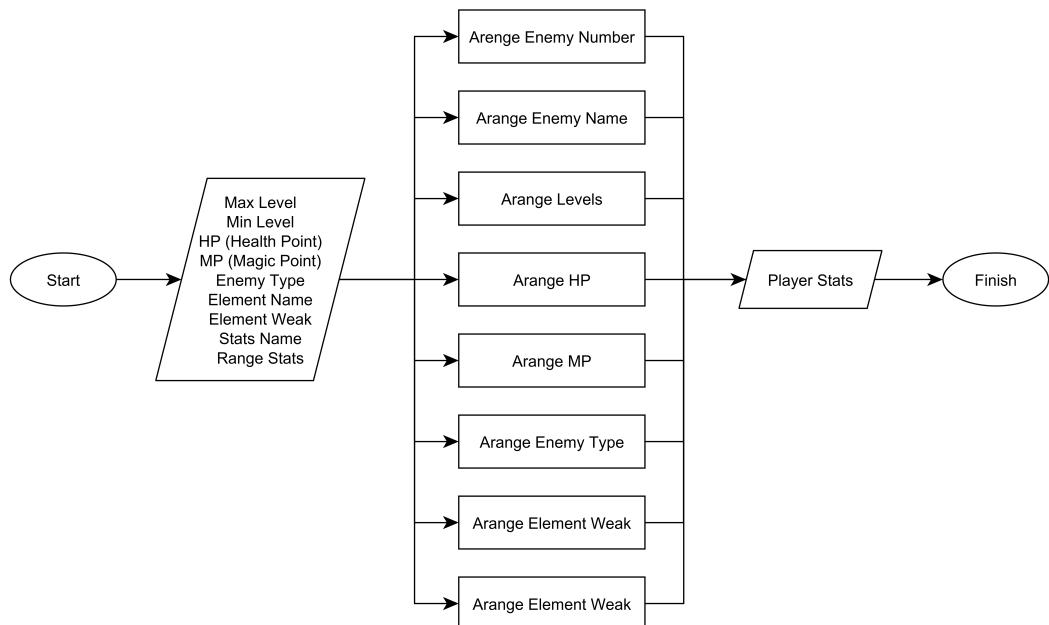


Gambar 3.13: Kenaikan stats pemain setiap levelnya.

Pada Tabel 3.3 adalah data *stats* dari pemain yang dihasilkan melalui penambahan nilai pada *stats* secara acak pada setiap *stats*. Seperti yang dijelaskan pada persamaan 2.15, 3.5, dan persamaan 3.6 saat nilai *stats* ditambahkan secara acak dari 1 sampai dengan 2 pada level yang berjarak antara 1 sampai dengan 100. Selanjutnya representasi dari hasil penambahan *stats* tersebut yang ditunjukkan melalui Gambar 3.13 yang mana nilai dari setiap *stats* terus naik sesuai dengan level pemain yang juga terus naik.

3.3 Desain Level dan Stats pada Karakter Musuh

Sama halnya dengan karakter pemain maka dibuatlah sebuah program yang secara otomatis dapat membuat *stats* musuh dengan masukan sesuai dengan kebutuhan desainer permainan atau pengembang. Program tersebut terdiri dari beberapa fungsi yang pada awalnya adalah objek yang memiliki masukan parameter-parameter yang nantinya akan menghasilkan sebuah data yang berupa *stats* dari karakter pemain seperti proses yang ditunjukkan oleh diagram alur sederhana pada Gambar 3.14.



Gambar 3.14: Proses pembuatan stats untuk karakter musuh.

Pada permainan dengan genre *turn-based* RPG dengan jumlah yang sangat banyak dan beragam, maka program yang ditunjukan melalui proses pada Gambar 3.14 dapat dijalankan satu kali saja, dan menghasilkan banyak musuh. Kecuali ingin menghasilkan kombinasi musuh yang berbeda, misalnya pada proses *generate* yang pertama menghasilkan musuh yang memiliki kemampuan *magic* dan kelemahan sedangkan pada kombinasi musuh selanjutnya tidak memiliki kemampuan *magic*, hanya mengandalkan kemampuan fisik saja. Sama halnya dengan permainan *action* RPG saat proses pembuatan stats musuh.

Pada Gambar 3.14 di sisi masukan program terdapat banyak sekali masukan variabel seperti *Max Level* yang berupa maksimum level yang diinginkan, kemudian *Min Level* yang berupa minimum level dari musuh, kemudian HP yang berupa *range* atau jarak antara HP minimum dengan HP tertinggi. Sama halnya dengan HP, dalam perhitungan *range* atau jarak, pada *MP* juga menggunakan perhitungan dengan cara tersebut. Kemudian *Enemy Type* yang berisi klasifikasi jenis stats musuh, tergolong musuh seperti apakah stats yang dihasilkan tersebut. Selanjutnya adalah *Element Name* yang berisi daftar elemen apa saja yang dapat dipilih saat pembuatan karakter musuh, selanjutnya *Element Weak* berisi tetang kelemahan dan keunggulan dari musuh tersebut ketika diserang, apakah saat diserang dengan menggunakan elemen tersebut akan mengalami kerusakan atau damage yang parah, normal atau tidak mempan sama sekali. Kemudian untuk *Stats Name* berisikan nama *stats* yang dipakai dalam membuat karakter musuh. Selanjutnya adalah isi atau data dari *stats* itu sendiri yang menentukan karakter dari musuh itu sendiri, seberapa kuat musuh tersebut dalam menyerang atau bertahan dan lain sebagainya. Lebih detail tentang program yang dibuat dapat dilihat pada Gambar 3.15, yang merupakan *class* diagram untuk membuat *stats* pemain.

Karena program ini dibangun menggunakan OOP (*Object Oriented Programming*) maka program ini dapat dijabarkan menggunakan Gambar 3.15. Selain itu program ini juga dapat secara mudah dimodifikasi untuk keperluan

Player
+ maxLevel : int + minLevel : int + minHP : int + maxHP : int + minMP : int + maxMP : int + enemyType : List (string) + elementsName : List (string) + damageName : List (string) + charWeakNumber : List (int) + nameStats : List (string) + basicMaxStats : List (int) + basicMinStats : List (int)
+ rangeLevels(int, string) : List int + showRangeLevels() : void + rangeHealthPoints(int, int, string) : List int + showHealthPoints() : void + rangeMagicPoints(int, int, string) : List int + showMagicPoints() : void + rangeElementWeak(List string, List int) : List Int + showElementWeak() : void + rangeStats(List string, List int, List int) : List int + showRangeStats() : void + genStats() : void

Gambar 3.15: Class diagram untuk *stats* musuh.

pengembangan kedepannya, dengan fungsi-fungsi yang ada sangat memungkinkan dilakukan *override* atau pembuatan fungsi yang sama dengan isi atau proses yang berbeda seperti pada penjelasan dibagian pemain pada Sub-bab 3.2. Seperti pada bagian sebelumnya maka dibuatlah Tabel 3.4 yang berupa masukan untuk menguji program yang akan dijelaskan pada bagian-bagian selanjutnya, yang mana masukan pada tabel tersebut akan menghasilkan stats pada sebuah karakter untuk pemain. Sama seperti pada Sub-bab 3.2 sebelumnya, yang mana pada bagian ini juga merupakan perwujudan teknis dari Sub-bab 3.1.1 dalam bentuk yang lebih teknis.

Tabel 3.4: Data masukan untuk pembuatan program pada musuh.

Variabel	Input
<i>Enemy Numbers</i>	400

<i>Max Level</i>	80
<i>Min Level</i>	1
<i>Level Class</i>	[‘Easy’, ‘Medium’, ‘High’]
<i>Min HP</i>	159
<i>Max HP</i>	163
<i>Min MP</i>	89
<i>Max MP</i>	93
<i>Enemy Type</i>	[‘Mixed’, ‘Hard Magic’, ‘Soft Magic’, ‘Hard Strength’, ‘Soft Magic’]
<i>Distribution Percentage</i>	[40, 10, 20, 10, 20]
<i>List Element</i>	[‘Phys’, ‘Water’, ‘Wind’, ‘Earth’, ‘Fire’]
<i>List Damage</i>	[‘Normal’, ‘Repel’, ‘Weak’]
<i>List Stats Name</i>	[‘Strength’, ‘Magic’, ‘Endurance’, ‘Speed’, ‘Luck’]
<i>Max Stats Value</i>	[50, 60, 40, 55, 45]
<i>Min Stats Value</i>	[2, 2, 2, 2, 2]

3.3.1 Distribusi Level Musuh

Pada bagian ini akan dijelaskan tentang pembagian level pada musuh, dengan masukan seperti yang disebutkan pada Tabel 3.4. Beracuan pada tabel tersebut beberapa variabel utama yang akan digunakan diantaranya adalah “*Enemy Numbers*” yang menentukan jumlah musuh yang akan dibuat, “*Max Level*” dan “*Min Level*” adalah nilai maksimum dan minimum level dari musuh yang ingin dibuat.

Selanjutnya tingkat kesulitan dari musuh juga ditentukan disini, dengan variabel “*Level Class*” yang isinya dibagi menjadi “*Easy*”, “*Medium*” dan “*Hard*”. Musuh dengan “*Level Class*” atau tingkat kesulitan “*Easy*” akan menjadi yang paling mudah dikalahkan, diikuti dengan tingkat kesulitan “*Medium*” dan “*Hard*” secara berurutan.

Kemudian terdapat variabel pendukung yang akan menentukan data atau level yang ingin dihasilkan, yaitu *scale*. Selaanjutnya dilakukan beberapa proses seperti pada persamaan 3.10, 3.11, 3.12, dan persamaan 3.13 yang kemudian diperoleh hasil berupa level untuk banyak musuh sekaligus seperti yang ditunjukkan pada persamaan 3.14.

$$SL_N = \begin{cases} \sum_{i=0}^N \frac{LN}{LC_N} & \text{Saat } 0 \equiv LN \pmod{LC_N} \\ \sum_{i=0}^N \frac{LN}{LC_N} & \text{Saat } 0 \not\equiv LN \pmod{LC_N}, \\ & 0 \not\equiv LC_N \pmod{2} \Rightarrow SL_{(\lceil N/2 \rceil)} = \left\lceil \frac{LN}{LC_N} \right\rceil \\ & \Rightarrow SL_i = \left\lfloor \frac{LN}{LC_N} \right\rfloor \\ \sum_{i=0}^N \frac{LN}{LC_N} & \text{Saat } 0 \not\equiv LN \pmod{LC_N}, \\ & 0 \equiv LC_N \pmod{2} \Rightarrow SL_{N/2}, SL_{(N/2)+1} = \left\lceil \frac{LN}{LC_N} \right\rceil \\ & \Rightarrow SL_i = \left\lfloor \frac{LN}{LC_N} \right\rfloor \end{cases} \quad (3.10)$$

Pada persamaan 3.10 adalah proses pembagian dari jumlah level dari musuh yang dijelaskan pada Tabel 3.4 pada variabel *Max Level* dan *Min Level* sebagai *range* yang merupakan level dari musuh yang disimbolkan dengan *LN* yang kemudian dibagi dengan *LC_N* yang merupakan jumlah kelompok atau *cluster* data dari *level class*. Jika beracuan pada Tabel 3.4 maka jumlah *cluster* level adalah jumlah data atau level yang dimuat pada setiap variabel didalam “*Level Class*” seperti “*Easy*”, “*Medium*” dan “*High*”, jadi jumlah levelnya adalah 3 *cluster* level. Jadi pada kasus yang dicontohkan ini *LC* = {1, 2, 3}, kemudian jika jumlah *cluster* level ingin dibuat lebih dari contoh maka *LC* = {1, 2, 3, ..., *LC_N*} dengan *N* adalah jumlah *cluster* level yang ingin dibuat.

Dalam proses pencarian *SL_N* atau *cluster* level yang terbagi menjadi tiga tersebut, terdapat beberapa keputusan yang harus dijalankan. Seperti hanya hasil bagi antara *LN* dan *LC_N* harus bernilai bilangan tidak bulat, maka harus dilakukan proses pembulatan jumlah level terlebih dahulu dikarenakan nilai level tidak boleh bernilai angka yang tidak bulat. Seperti pada persamaan 3.10 untuk dilakukan pengujian apakah dapat dibulatkan maka dilakukan operasi

modulus atau *mod*, jika habis maka SL_N akan diisi secara merata oleh hasil bagi antara LN dan LC_N . Maka jumlah *cluster* sebanyak N pada LC_N dan banyaknya level adalah sebesar LN yang dibagi dengan LC_N yang kemudian hasil akhir tersebut disimpan dalam variabel SL_N .

Namun jika tidak dapat dibagi secara merata maka dilakukan terlebih dahulu proses pengecekan apakah jumlah LC_N apakah berjumlah ganjil atau genap, hal tersebut dilakukan dengan cara melakukan modulus dari LC_N dengan angka 2. Jika ternyata LC_N bernilai genap maka SL_N atau jumlah *cluster* akan dibagi menjadi dua bagian, kemudian dua nilai tengah setelah dibagi menjadi dua bagian tersebut diisi dengan jumlah level yang lebih tinggi jika dibandingkan dengan *cluster* level yang lain. Kemudian jika LC_N bernilai ganjil maka nilai tengah dari jumlah *cluster* tersebutlah yang akan memiliki jumlah level lebih banyak jika dibandingkan dengan *cluster* level yang lain.

$$SE_N = \begin{cases} \sum_{i=0}^N \frac{EN}{LC_N} & \text{Saat } 0 \equiv EN \pmod{LC_N} \\ \sum_{i=0}^N \frac{EN}{LC_N} & \text{Saat } 0 \not\equiv EN \pmod{LC_N}, \\ & 0 \not\equiv LC_N \pmod{2} \Rightarrow SE_{(\lceil N/2 \rceil)} = \left\lceil \frac{EN}{LC_N} \right\rceil \\ & \Rightarrow SE_i = \left\lceil \frac{EN}{LC_N} \right\rceil \\ \sum_{i=0}^N \frac{EN}{LC_N} & \text{Saat } 0 \not\equiv EN \pmod{LC_N}, \\ & 0 \equiv LC_N \pmod{2} \Rightarrow SE_{N/2}, SE_{(N/2)+1} = \left\lceil \frac{EN}{LC_N} \right\rceil \\ & \Rightarrow SE_i = \left\lceil \frac{EN}{LC_N} \right\rceil \end{cases} \quad (3.11)$$

Munculah pertanyaan berapakah jumlah level pada dua *cluster* level tengah pada kasus LC_N dengan nilai genap, dan berapakah jumlah level pada *cluster* level pada bagian tengah untuk kasus LC_N dengan nilai ganjil. Seperti pada persamaan 3.10 jika pada kondisi genap maka dua *cluster* level bagian tengah diisi dengan level sejumlah hasil pembagian LN dengan LC_N yang dibulatkan ke atas atau *ceil*, sedangkan pada kondisi ganjil maka 1 *cluster* level bagian tengahlah yang diisi dengan hasil pembagian tersebut. Kemudian untuk jumlah level pada *cluster* yang lain diisi dengan pembagian LN dengan LC_N yang dibulatkan ke bawah atau *floor*. Jika SL_N adalah jumlah *cluster* level, maka perlu dicari juga jumlah cluster untuk musuh dengan menggunakan

an persamaan 3.11 dengan cara yang sama dengan pencarian jumlah *cluster* level.

Dalam proses pencarian SE_N atau *cluster* musuh pada persamaan 3.11 digunakan cara yang sama seperti cara perhitungan pada persamaan 3.10 dengan membagi jumlah musuh ke dalam tiga bagian, dan dilanjutkan dengan pengambilan beberapa keputusan yang harus dijalankan. Sama seperti pada perhitungan hasil bagi antara LN dan LC_N pada persamaan 3.10, nilai pembagian EN atau jumlah musuh yang ingin dihasilkan dengan LC_N atau jumlah *cluster* musuh yang ingin dibuat dengan tiga tingkatan “*Easy*”, “*Medium*” dan “*Hard*” yang juga harus bernilai bilangan bulat positif, kemudian dilakukan juga proses pembulatan jumlah musuh terlebih dahulu dikarenakan jumlah musuh tidak boleh bernilai angka yang tidak bulat. Seperti pada persamaan 3.10 pada persamaan 3.11 juga dilakukan pengujian apakah dapat dibulatkan atau tidak maka dilakukan operasi modulus atau *mod*, jika habis maka SE_N akan diisi secara merata oleh hasil bagi antara EN dan LC_N . Maka jumlah *cluster* sebanyak N pada LC_N dan banyaknya musuh adalah sebesar EN yang dibagi dengan LC_N , kemudian hasil akhir tersebut disimpan dalam variabel SE_N . Untuk penjelasan langkah selanjutnya pada persamaan 3.11 sama persis dengan persamaan 3.10 yang sudah dijelaskan pada bagian sebelumnya, hanya saja objek pada persamaan 3.10 adalah level sedangkan pada persamaan 3.11 adalah musuh. Setelah SL_N dan SE_N diperoleh maka saatnya menuju bagian yang lebih dalam dan detail lagi. Bagaimana dengan pemberian level untuk setiap musuh, maka digunakanlah persamaan 3.12 dan persamaan 3.13 dengan penjelasan sebagai berikut.

$$SSL_N = \begin{cases} \sum_{i=0}^N \frac{SL}{Sc} & \text{Saat } 0 \equiv SL \pmod{Sc} \\ \sum_{i=0}^N \frac{SL}{Sc} & \text{Saat } 0 \not\equiv SL \pmod{Sc}, \\ & 0 \not\equiv Sc \pmod{2} \Rightarrow SSL_{(\lceil N/2 \rceil)} = \lceil \frac{SL}{Sc} \rceil \\ & \Rightarrow SSL_i = \lfloor \frac{SL}{Sc} \rfloor \\ \sum_{i=0}^N \frac{SL}{Sc} & \text{Saat } 0 \not\equiv SL \pmod{Sc}, \\ & 0 \equiv Sc \pmod{2} \Rightarrow SSL_{N/2}, SSL_{(N/2)+1} = \lceil \frac{SL}{Sc} \rceil \\ & \Rightarrow SSL_i = \lfloor \frac{SL}{Sc} \rfloor \end{cases} \quad (3.12)$$

Setelah diperolehnya SL_N dan SE_N yang masing-masing adalah cluster *level* dan *cluster* musuh, maka dicarilah SSL_N atau *sub-cluster* level pada persamaan 3.12 hal ini betujuan untuk mempersempit *range* dalam pembagian level musuh. Seperti pada penjelasan sebelumnya terdapat satu variabel lagi yang mempengaruhi proses ini, variabel tersebut adalah SC atau skala yang mentukarkan kerapatan pembagian level pada setiap musuh. Pada kasus ini SSL_N adalah jumlah *sub-cluster* dari setiap level *cluster* atau SL_N , jadi pada dasarnya persamaan 3.12 dijalankan setelah nilai SL_N diperoleh dan nilai SSL_N selalu berubah-ubah mengikuti skala atau Sc yang menjadi salah satu masukan untuk fungsi pada program ini. Perhitungan SSL_N atau jumlah *sub-cluster* level pada dasarnya sama dengan perhitungan dalam mencari SL_N atau SE_N pada bagian sebelumnya, hanya saja pada bagian sebelumnya jumlah *cluster* level dan musuh dipengaruhi oleh LC_N sedangkan pada SSL_N jumlah *sub-cluster* level dipengaruhi oleh Sc .

$$SSE_N = \begin{cases} \sum_{i=0}^N \frac{SE}{Sc} & \text{Saat } 0 \equiv SE \pmod{Sc} \\ \sum_{i=0}^N \frac{SE}{Sc} & \text{Saat } 0 \not\equiv SE \pmod{Sc}, \\ & 0 \not\equiv Sc \pmod{2} \Rightarrow SSE_{(\lceil N/2 \rceil)} = \lceil \frac{SE}{Sc} \rceil \\ & \Rightarrow SSE_i = \lfloor \frac{SE}{Sc} \rfloor & (3.13) \\ \sum_{i=0}^N \frac{SE}{Sc} & \text{Saat } 0 \not\equiv SE \pmod{Sc}, \\ & 0 \equiv Sc \pmod{2} \Rightarrow SSE_{N/2}, SSE_{(N/2)+1} = \lceil \frac{SE}{Sc} \rceil \\ & \Rightarrow SSE_i = \lfloor \frac{SE}{Sc} \rfloor \end{cases}$$

Kemudian dilakukan juga pengecekan dengan operasi modulus atau *mod* antara SL dengan SC , apakah SL akan habis jika dibagi dengan SC . Jika ternyata SL habis dibagi dengan Sc , maka level pada range SSL atau *sub-cluster* tersebut akan langsung dibagi secara merata. Sedangkan pada kondisi sebaliknya yaitu saat SE tidak habis dibagi dengan Sc maka akan dilakukan proses pembulatan ganjil dan genap yang dibulatkan ke atas atau *ceil* seperti pada persamaan 3.10 dan persamaan 3.11, kemudian untuk nilai yang lain dibulatkan ke bawah atau *floor*.

Kemudian untuk persamaan 3.13 sama seperti persamaan 3.13 hanya saja yang menjadi subjek disini adalah SSE_N atau *sub-cluster* jumlah musuh.

Kemudian cara perhitungannya sama seperti persamaan 3.12, yang dibagi adalah *cluster SE* dari musuh dibagi dengan skala atau *Sc*. Saat semua sudah selesai dilakukan, khususnya yang ada pada persamaan 3.10 dan persamaan 3.10. Kemudian dieksekusi setiap *cluster* level dan musuh ke dalam setiap *sub-cluster*, pada setiap *sub-cluster* tersebutlah proses pemberian level pada setiap musuh dilakukan. Pada persamaan 3.14 adalah penjelasan tentang peluang diperolehnya level pada setiap musuh, yang beracuan pada metode *Naive Bayes*, lebih tepatnya lagi adalah tentang peluang marginal.

$$P(Elv_k) = \sum_{i=0}^M \sum_{j=0}^N \sum_{k=0}^L \frac{Elv_k}{Lv_j + SSL_N} \quad (3.14)$$

Pada persamaan 3.14 $P(Elv_k)$ adalah peluang munculnya Elv ke k , sedangkan Lv ke j adalah batas bawah range level yang dapat diambil oleh musuh. Kemudian SSL_N sepeerti yang sudah dijelaskan pada bagian sebelumnya, variabel itu adalah *sub-cluster* dari level. Variabel tersebut dapat digunakan untuk mengubah jarak level saat terjadi perubahan nilai *sub-cluster* yang juga menentukan nilai dari Elv ke k . Setelah melalui sekian tahapan yang sudah dijelaskan sebelumnya dan dengan masukan variabel pada Tabel 3.4 maka level yang dihasilkan akan terlihat seperti pada Tabel 3.5.

Tabel 3.5: Hasil level yang dibuat untuk musuh.

No.	Name	Levels
1	Enemy 1	1
2	Enemy 2	1
3	Enemy 3	1
4	Enemy 4	1
5	Enemy 5	2
6	Enemy 6	2
7	Enemy 7	2
8	Enemy 8	2

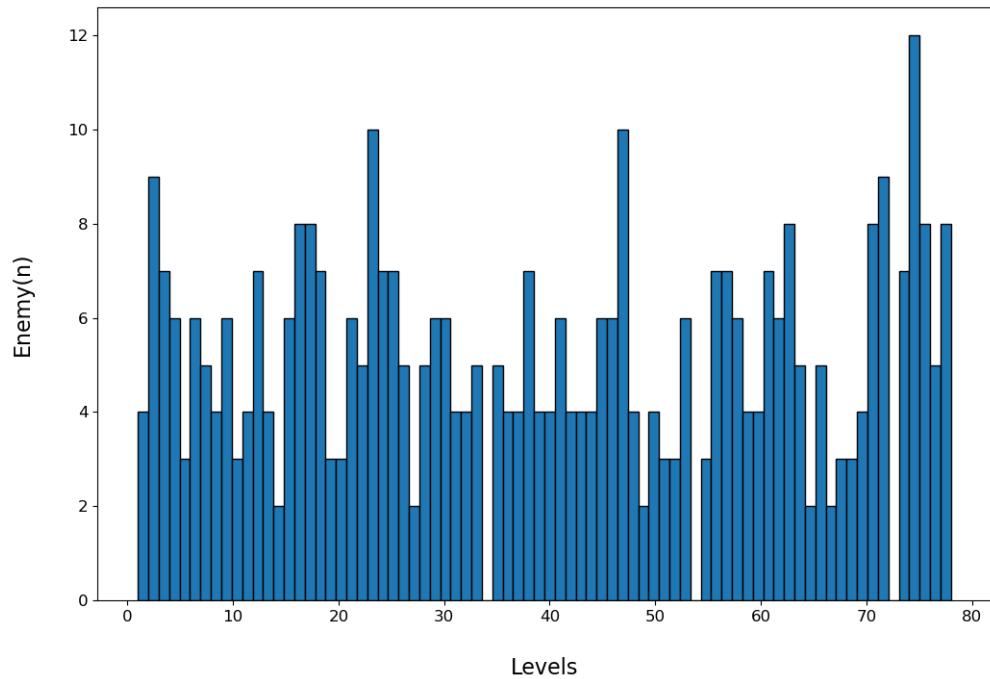
9	Enemy 9	2
10	Enemy 10	2
11	Enemy 11	2
12	Enemy 12	2
13	Enemy 13	2
14	Enemy 14	3
15	Enemy 15	3
...
400	Enemy 400	78

Pada Tabel 3.5 adalah sebagian data dari level yang dihasilkan oleh program, untuk hasil lebih lengkapnya bisa dilihat pada Bagian LAMPIRAN pada Tabel 5.6 sampai dengan Tabel 5.20 di kolom *Levels*. Kemudian persebaran level yang dihasilkan tersebut bila divisualisasikan maka hasilnya akan seperti yang ditujukkan pada Gambar 3.16. Grafik atau histogram yang ditunjukkan pada Gambar 3.16 tersebut sangatlah tidak merata, hal tersebut dikarenakan proses penentuan level yang ditentukan secara acak.

Selanjutnya adalah validasi dari keseimbangan persebaran level musuh, hal ini dilakukan dengan menggunakan persamaan melalui beberapa langkah seperti yang ditunjukkan oleh persamaan 3.15, 3.16, 3.17 dan persamaan 3.18. Konsep tersebut beracuan pada Sub-bab 2.3.4 tentang *Gaussian Naive bayes*, dengan harapan apakah setiap data yang dihasilkan sebelumnya sudah terdistribusi dengan normal atau tidak. Penjelasan berikut ini adalah beberapa langkah untuk validasi keseimbangan persebaran level musuh.

$$\bar{Elv} = \frac{\sum_{i=0}^N Elv_i}{N} \quad (3.15)$$

Adapun urutan metode dalam penggunaan *Gaussian Naive Bayes* adalah dengan mencari rata-rata dari data tersebut, kemudian diikuti dengan perhitungan standar deviasi. Pada persamaan 3.15 adalah rata-rata dari level



Gambar 3.16: Distribusi Level Musuh.

yang dihasilkan sama seperti pembahasan pada persamaan 2.25 yang kemudian disimbolkan dengan variabel \bar{Elv} . Selanjutnya adalah \bar{Elv}_i adalah setiap level dari musuh yang terus dijumlahkan sebanyak N , yang mana N sendiri adalah banyaknya musuh yang sudah dibuat. Setelah diperoleh rata-rata level maka dapat dilanjutkan dengan pencarian nilai *standar deviasi* seperti pada persamaan 3.17.

$$\sigma(Elv)^2 = \frac{\sum_{i=0}^N (Elv_i - \bar{Elv})^2}{N} \quad (3.16)$$

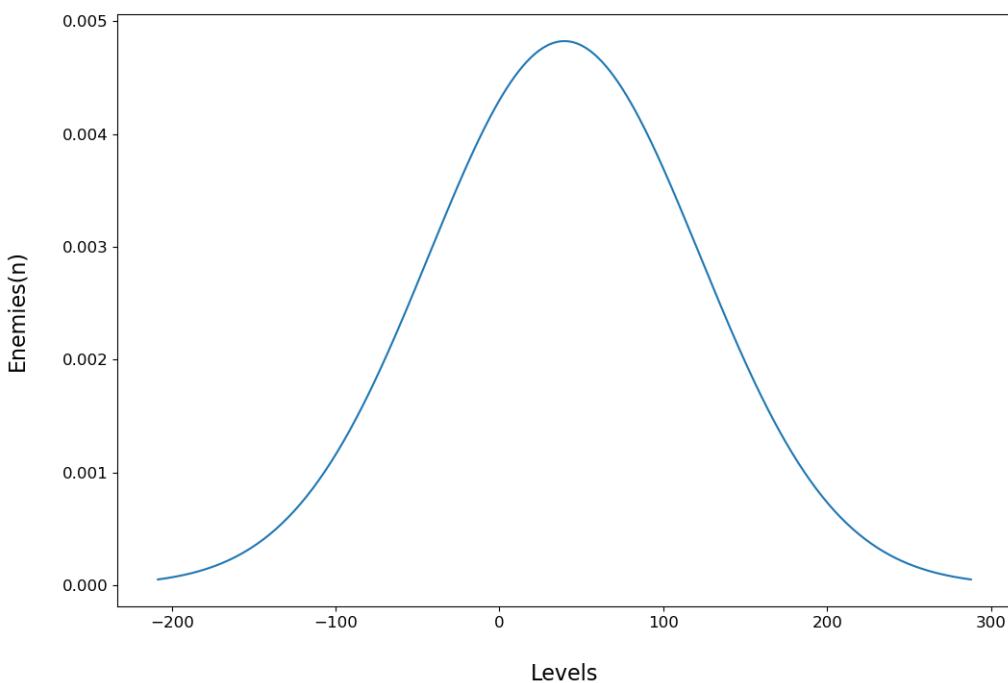
$$\sigma(Elv) = \sqrt{\frac{\sum_{i=0}^N (Elv_i - \bar{Elv})^2}{N}} \quad (3.17)$$

Pada persamaan 3.17 adalah persamaan untuk mencari standar deviasi atau $\sigma(Elv)$ setelah diperolehnya rata-rata atau \bar{Elv} melalui persamaan 3.15 yang kemudian dilanjutkan dengan pencarian varian atau $\sigma(Elv)^2$ melalui persamaan 3.16 yang selanjutnya diakar kuadratkan untuk memperoleh

nilai standar deviasi. Jika standar deviasi, varian dan rata-rata sudah diperoleh maka dapat dilanjutkan menuju pencarian nilai distribusi normal melalui *Gaussian Naive Bayes* seperti pada persamaan 3.18 berikut ini.

$$PDF(Elv, \bar{Elv}, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(Elv - \bar{Elv})^2}{2\sigma^2}\right) \quad (3.18)$$

Konsep pada persamaan 3.18 sudah sangat dijelaskan pada Sub-bab 2.3.4 poin ke 3. Hasil perhitungannya sebelumnya yang berupa varian σ^2 , rata-rata \bar{Elv} dan standar deviasi σ menjadi masukan pada persamaan tersebut. Variabel π adalah konstanta numerik pada umumnya, kemudian fungsi $\exp()$ atau e adalah konstanta numerik yang betujuan untuk membentuk hasil prediksi dengan pendekatan eksponensial. Jika memang benar data hasil program ini valid maka ketika musuh bertambah maka pola dari level dan jumlah musuh masih akan membentuk pola distribusi normal seperti pada Gambar 3.17.



Gambar 3.17: Distribusi level musuh dalam bentuk distribusi normal.

Pada Gambar 3.17 jika dilihat persebaran datanya dari kiri ke kanan menggambarkan tingkat kesulitan musuh berdasarkan level, jumlah musuh

pada sisi kiri berjumlah sedikit semakin ke kanan jumlahnya meningkat dan semakin ke kanan jumlah musuh kembali menjadi sedikit jumlahnya. Hal tersebut menggambarkan tingkat keseimbangan dari musuh yang dibuat, yang mana jumlah musuh dengan level menengah berjumlah paling banyak dan jumlah musuh yang sangat sulit dan sangat mudah dikalahkan berjumlah sedikit. Tujuan utama dari kondisi tersebut adalah terjadinya keseimbangan saat terjadinya pertarungan antara pemain dan musuh.

3.3.2 Distribusi Tipe Musuh

Di bagian distribusi tipe musuh akan dijelaskan tentang pembagian tipe musuh, dengan masukan seperti yang disebutkan pada Tabel 3.4. Beracuan pada tabel tersebut beberapa variabel utama yang akan digunakan diantaranya adalah “*Enemy Type*” yang menentukan tipe apa saja musuh yang akan dibuat. Kemudian *Distribute Percentage* adalah persentase distribusi tipe yang ingin dibuat. Kemudian variabel level musuh yang diambil dan dijelaskan pada Sub-bab 3.3.1 juga turut menentukan tipe musuh yang ingin dibuat.

Terdapat juga variabel pendukung yang membantu proses distribusi tipe, diantaranya adalah *Distribute Number*, *Distribute Level*. Kemudian dilakukan beberapa proses seperti pada persamaan 3.19, 3.20, 3.21, dan 3.23 yang kemudian diperolehlah hasil berupa probabilitas distribusi tipe musuh yang ditunjukkan pada persamaan 3.22 dan persamaan 3.24.

$$DN_N = \sum_{i=0}^M \sum_{j=0}^N EN_N \times \frac{DP_j}{100} \begin{cases} \text{Saat } \lceil DN_i \rceil - DN_i < DN_i - \lfloor DN_i \rfloor, \quad DN_i = \lceil DN_i \rceil \\ \text{Saat } \lceil DN_i \rceil - DN_i > DN_i - \lfloor DN_i \rfloor, \quad DN_i = \lfloor DN_i \rfloor \end{cases} \quad (3.19)$$

Pada persamaan 3.19 adalah persamaan untuk mencari *distribute number* atau distribusi musuh DN_N yang mana nilai dari variabel tersebut akan memiliki nilai sebenarnya yang diperoleh melalui jumlah musuh EN_N yang sudah dibahas pada Sub-bab 3.3.1, yang kemudian dibandingan dengan presentase yang disiapkan sebelumnya melalui variabel masukan *distribute percentage* atau DP_j dengan nilai yang ditunjukkan pada Tabel 3.4. Keluaran dari DN_N sendiri berupa angka bulat dan berupa *cluster* angka, maka dari itu setiap

angka yang membentuk *cluster* jumlah musuh tersebut harus dilakukan operasi pembulatan seperti pada DN_N dengan menggunakan syarat seperti pada persamaan 3.19. Selain itu angka bulat yang dihasilkan jumlahnya tidak hanya berjumlah satu, pemecahan jumlah musuh pada bagian ini mengikuti jumlah angka yang ada dalam variabel DP_j . Secara berurutan jumlah angka tersebut merealisasikan persentase jumlah tipe musuh pada variabel DP_j .

$$DL_N = \sum_{i=0}^M \sum_{j=0}^N \left\lfloor \frac{DN_j}{DP_i} \right\rfloor \begin{cases} \text{Saat } i = 0, & DL_i = DN_i \\ \text{Lainnya,} & DL_{i-1} < DL_{(i-1)+DN} \end{cases} \quad (3.20)$$

Selanjutnya adalah operasi pembagian musuh dan levelnya setelah ditemukannya DN_N , sebagai batasan dalam membagi setiap musuh berdasarkan presentase tipe DP_N pada persamaan 3.20. Dengan kata lain DP_N digunakan sebagai pembagi jumlah musuh ke dalam level, padalah DP_N adalah sebuah variabel yang bertujuan memuat presentase pembagian jumlah musuh dan level saja. Hal ini bersifat opsional dan dapat terus dikembangkan mungkin dengan menambah variabel baru untuk mendistribusikan jumlah musuh ke dalam level dengan tidak menggunakan variabel DP_N .

Kemudian pada persamaan 3.20 dicarilah nilai DL_N yang berisi urutan distribusi musuh DN_i yang dibagi dengan jumlah presentase tipe DP_N seperti pada persamaan 3.19. Hal tersebut sejatinya akan membentuk sebuah *cluster* kecil yang berisi distribusi musuh dan tipenya. Bila membagi setiap musuh dengan jumlah presetase tipe maka tidak semua musuh dapat terbagi secara merata atau habis, maka dari itu juga dicari sisa hasil pembagian antara distribusi musuh DN_i dengan jumlah presentase tipe DP_N pada persamaan 3.21 pada persamaan 3.21.

$$rDL_N \equiv \sum_{i=0}^N DN_i \pmod{DP_N} \quad (3.21)$$

Pada persamaan 3.21 diperolehlah sisa musuh yang tidak terdistribusi atau sisa dari jumlah musuh yang tidak habis saat dibagi dengan jumlah presentase tipe DP_N . Musuh yang belum terdistribusi nantinya akan ditambahan

atau didistribusi ke tipe sesuai urutan hasil yang tersimpan pada variabel 3.21 pada persamaan 3.21.

$$P(ET_i) = \sum_{i=0}^M \sum_{j=0}^N \frac{ET_i}{\sum_{k=0}^L DL_k} \begin{cases} \text{Saat } ET_i \leq DL_j \\ \text{Saat } DL_{(j-1)} < ET_i \leq DL_j \\ \text{Lainnya } 0 \end{cases} \quad (3.22)$$

Dari jumlah musuh yang terdistribusi kedalam *cluster* level tersebutlah kemudian dilakukan pemetaan tipe, pada kasus ini digunakanlah konsep probabilitas marginal seperti yang dijelaskan pada persamaan 2.3.2 poin ke 1, bila disesuaikan dengan kasus ini maka persamaan yang diperoleh akan menjadi seperti pada persamaan 3.22. Pada setiap karakter musuh ET_i akan melih satu tipe yang akan digunakannya, tipe tersebut akan menentukan *stats* atau data statistiknya yang akan dijelaskan pada bagian selanjutnya. Kemudian probabilitas tipe yang dipilih ET_i dinyatakan dengan variabel $P(ET_i)$, kisaran kemunculan nilai atau tipe musuh ET_i diperoleh dari sekumpulan data distribusi level DL_N .

$$rET = \sum_{i=0}^M \sum_{j=0}^N EN_M - ET_N \quad (3.23)$$

Selanjutnya pada persamaan 3.23 adalah langkah untuk melihat apakah ada sisa musuh yang belum terdistribusi rET dengan cara melakukan pengurangan antara jumlah musuh EN_M yang sudah dijelaskan pada Sub-bab 3.3.1, yang mana M pada variabel tersebut menandakan jumlah musuh sedangkan ET adalah tipe dengan N sebagai jumlah musuh yang sudah memiliki tipe. Pada persamaan 3.23 variabel rET dapat bernilai 0 saat semua musuh DN habis terbagi kedalam jumlah presentase dari tipe DP_N seperti pada persamaan 3.20 dengan sisa musuh yang belum terdistribusi atau memiliki level disimpan pada variabel rDL_N pada persamaan 3.21. Kemudian selanjutnya adalah mendistribusikan sisa musuh yang belum memilliki tipe tersebut, seperti yang dilakukan oleh persaman 3.24.

$$P(ET_i) = \sum_{i=0}^M \sum_{j=0}^N \frac{ET_i}{\sum_{k=0}^L rDL_k} \begin{cases} \text{Saat } ET_i \leq rDL_j, rET > 0 \\ \text{Saat } rDL_{(j-1)} < ET_i \leq rDL_j, rET > 0 \\ \text{Lainnya } 0 \end{cases} \quad (3.24)$$

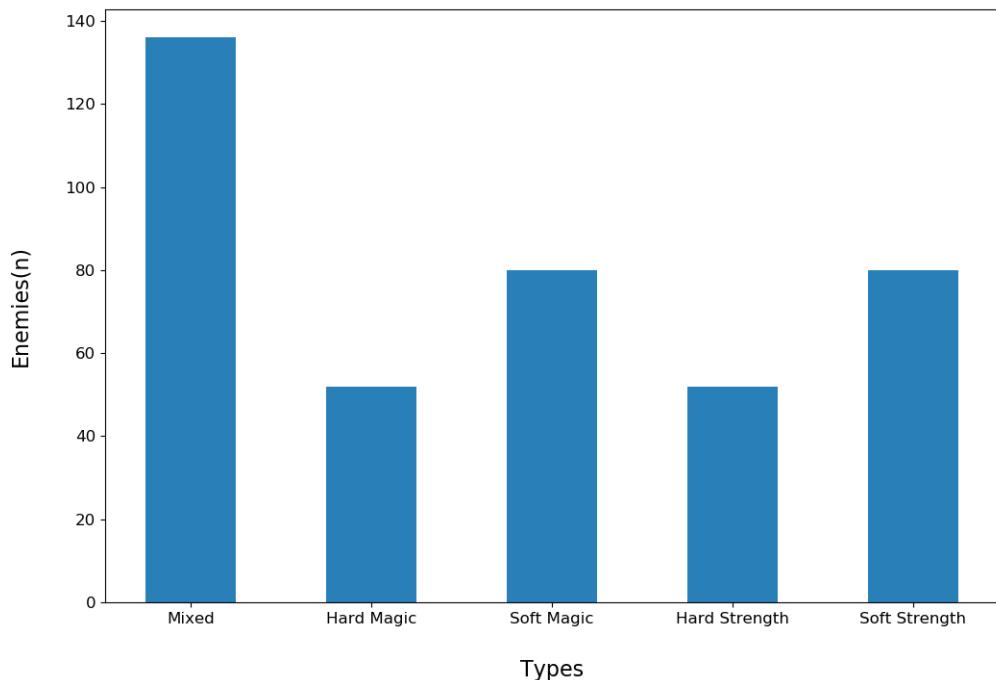
Melanjutkan persamaan 3.24, yang mana didistribusikan sisa musuh yang belum terdistribusi rDL_N . Kemudian setelah didistribusikan atau memilih tipe ET_i hasilnya digabung dengan tipe musuh yang sudah terdistribusi sebelumnya. Sama seperti pada persamaan 3.22 variabel $P(ET_i)$ adaalah probabilitas tipe musuh yang dipilih dari distribusi level rDL_k . Hasil distribusi dan penjeasan level yang dilakukan oleh beberapa persamaan diatas, maka hasilnya akan menjadi seperti Tabel 3.6.

Tabel 3.6: Hasil level yang dibuat untuk musuh.

No.	Name	Levels	Type
1	Enemy 1	1	0
2	Enemy 2	1	0
3	Enemy 3	1	2
4	Enemy 4	1	1
5	Enemy 5	2	2
6	Enemy 6	2	1
7	Enemy 7	2	2
8	Enemy 8	2	0
9	Enemy 9	2	2
10	Enemy 10	2	2
11	Enemy 11	2	4
12	Enemy 12	2	0
13	Enemy 13	2	0
14	Enemy 14	3	4
15	Enemy 15	3	4
...
400	Enemy 400	78	3

Dari data yang tedapat pada Tabel 3.6 hanya sebagian data saja, lebih lengkapnya dapat dilihat langsung pada Bagian LAMPIRAN dalam Tabel 5.6

sampai dengan Tabel 5.20 di kolom *Type*. Pada Tabel 3.6 dan tabel lain yang memuat tentang *Type* yang berisi data ET_i atau jenis musuh yang diisi dengan angka 0 sampai dengan 4, maksud dari angka tersebut adalah untuk mewakili indeks dari variabel *Enemy Type* pada Tabel 3.4. Secara berurutan dari 0 sampai dengan 4 adalah *Mixed*, *Hard Magic*, *Soft Magic*, *Hard Strength*, dan *Soft Magic*. Seluruh data tersebut jika divisualisasikan maka hasilnya seperti Gambar 3.18 sesuai yang ditargetkan oleh variabel *Distribute Percentage* pada Tabel 3.4 atau variabel *DP* pada persamaan 3.19.



Gambar 3.18: Distribusi tipe musuh.

Kemudian pada Gambar 3.18 adalah hasil dari proses pembuatan dan pengelompokan musuh berdasarkan tipe yang sudah sesuai dengan variabel masukan pada Tabel 3.4. Dengan musuh bertipe *Mixed* berjumlah yang paling banyak, diikuti *soft magic* dan *soft strength* dengan harapan bahwa ini adalah musuh yang memiliki karakter *magic* dan *strength* namun masih mudah dikalahkan, dilanjutkan dengan yang paling sedikit adalah *hard magic* dan *hard strength* dengan harapan menjadi musuh berkarakter *strength* dan *magic* yang sulit dikalahkan.

3.3.3 Distribusi Elemen dan Kelemahan Musuh

Pada persamaan 3.25 adalah penjelasan elemen yang digunakan oleh musuh seperti dideskripsikan pada Tabel 3.4 pada variabel *List Element*, di mana pada elemen tersebut terdapat kelemahan atau *weaknesses* dan kekebalan atau *repel* seperti yang dideskripsikan pada variabel *List Damage* pada Tabel 3.4. Seperti yang dijelaskan pada Sub-bab 3.1.1 poin ke dua tentang Elemen dan Efektifitas Serangan. Pada bagian ini elemen tersebut akan dibagi ke setiap musuh dengan kondisi yang berbeda, maksudnya nanti akan ada musuh yang memiliki kelemahan dan kekebalan yang berbeda antara musuh satu dengan musuh yang lain. Misalkan *Enemy 1* memiliki kelemahan api atau *fire* yang mana HP akan berkurang dua kali jika diserang dengan menggunakan elemen api dan memiliki kekebalan air atau *water* yang mana karakter tersebut kebal saat diserang dengan *skill* dari elemen air. Kemudian pada *Enemy 2* berbeda dengan *Enemy 1*, misalnya pada *Enemy 1* memiliki kelemahan api sedangkan pada *Enemy 2* memiliki kelemahan air misal dan kebal terhadap angin atau *wind*.

$$ElN = \begin{cases} Phys \\ Water \\ Wind \\ Earth \\ Fire \\ \dots \\ n \end{cases} \quad (3.25)$$

Pada dasarnya satu karakter musuh tidak akan menggunakan seluruh elemen yang ada pada persamaan 3.25 yang merupakan penggambaran dari variabel *List Element* pada Tabel 3.4, mungkin hanya sekitar satu sampai dengan tiga. Maka dari itu kondisi tersebut dapat dinyatakan dengan kondisi $DmgNa \in ElN$, yang mana seluruh elemen bisa menjadi kelemahan atau kekebalan dari musuh. Sama seperti pada persamaan 3.25 yang mana *ElN* adalah nama elemen, sedangkan *DmgNa* adalah *damage name* atau nama dari

efek serangan yang dilakukan. Di mana pada persamaan 3.26 $DmgNa$ akan dipilih secara acak dengan persamaan 3.27, yang akan menjadi kelemahan atau kekebalan terhadap serangan dari pemain seperti pada persamaan 3.26. Pada persamaan 3.26 sendiri adalah penggambaran dari variabel *List Damage* pada Tabel 3.4.

$$DmgNu = \begin{cases} 0, & Normal \\ 1, & Repel \\ 2, & Weak \\ \dots \\ n, & Defined Status Name \end{cases} \quad (3.26)$$

Sedangkan pada persamaan 3.26 variabel $DmgNu$ adalah respon terhadap serangan yang dikonversi menjadi angka, yang pada mulanya berupa nama respon atau status setelah diserang. Kemudian hasil pemilihan tersebut disimpan pada variabel $DmgNu$ yang dapat dinyatakan dengan persamaan 3.27.

$$P(DmgNu) = \frac{DmgNu}{\sum_{i=0}^N DmgNa_{(i-1)}} \quad (3.27)$$

Pada persamaan 3.26 variabel $DmgNa$ yang dipilih secara acak sebagai elemen yang memiliki respon khusus terhadap serangan yang kemudian disimpan pada variabel $DmgNu$, selanjutnya angka yang dipilih akan disimpan pada variabel $DmgNu$ yang selanjutnya dinyatakan dengan persamaan 3.27. Jadi dari $DmgNa$ akan dipilih satu sampai dengan tiga elemen secara acak yang kemudian menjadi kelemahan atau kekebalan dari karakter musuh tersebut.

$$P(DmgNu)_M = \sum_{i=0}^M \frac{DmgNu_i}{\sum_{j=0}^N DmgNa_{j-1}} \quad (3.28)$$

Sedangkan pada persamaan 3.28 adalah penjelasan tentang proses munculnya setiap $DmgNu$ pada satu karakter musuh, sehingga variabel tersebut berubah menjadi $DmgNu_i$. Seperti penjelasan sebelumnya bahwa pada sebuah karakter musuh dapat memiliki lebih dari satu kelemahan atau kekebalan

$DmgNu$ terhadap $DmgNa$ yang diset secara acak seperti yang sudah dijelaskan pada bagian sebelumnya.

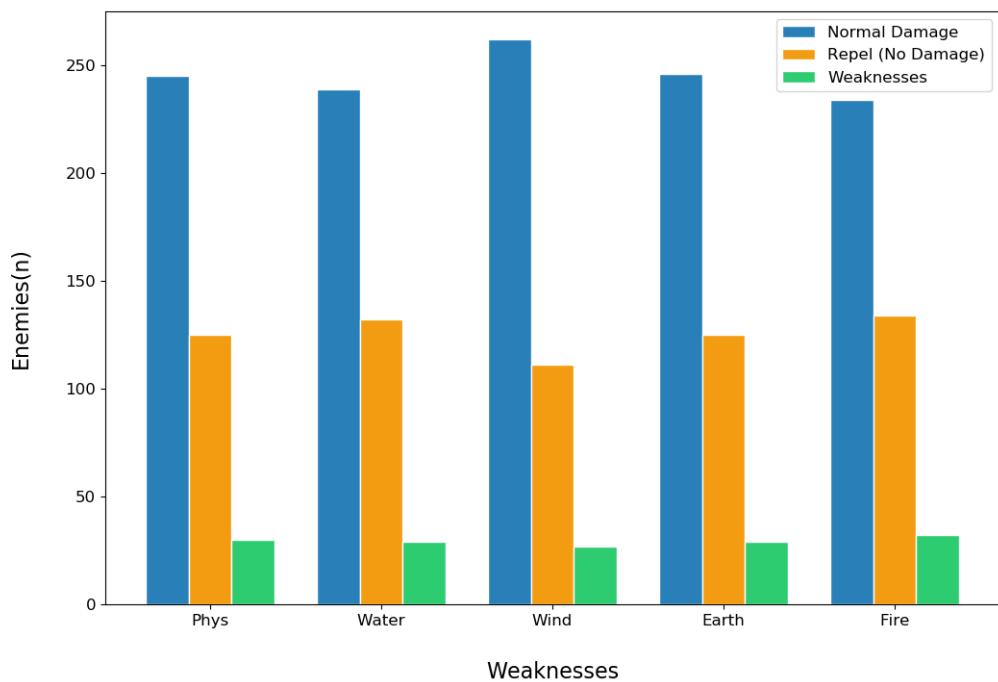
$$DmgEN_M = \sum_{i=0}^M \sum_{j=0}^N DmgNu_{ij} \quad (3.29)$$

Selanjutnya adalah proses penerapan persamaan 3.28 ke seluruh karakter musuh $DmgEN_M$ seperti pada persamaan 3.29. Sesuai dengan persamaan 3.29, yang mana M adalah jumlah musuh sedangkan N adalah jumlah $DmgNu$ pada satu musuh yang berjumlah satu sampai dengan tiga seperti yang sudah dijelaskan pada paragraf sebelumnya pada Sub-bab ini. Hasil dari proses dari keseluruhan proses yang dijelaskan pada Sub-bab ini menghasilkan persebaran kelemahan dan kekebalan musuh yang ditunjukkan pada Tabel 3.7.

Tabel 3.7: Hasil level yang dibuat untuk musuh.

No.	Name	Phys	Water	Wind	Earth	Fire
1	Enemy 1	1	1	0	1	0
2	Enemy 2	0	0	0	0	0
3	Enemy 3	0	1	0	1	1
4	Enemy 4	1	0	1	0	0
5	Enemy 5	0	1	0	0	1
6	Enemy 6	1	1	0	1	0
7	Enemy 7	1	0	0	0	0
8	Enemy 8	0	0	0	0	1
9	Enemy 9	1	1	0	0	0
10	Enemy 10	0	0	0	0	0
11	Enemy 11	1	0	0	1	0
12	Enemy 12	0	0	0	0	1
13	Enemy 13	0	0	2	1	0
...
400	Enemy 400	0	0	2	1	0

Jika melihat Tabel 3.7 dengan melihat pada kolom *Phys*, *Water*, *Wind*, *Earth*, dan *Fire*, isi dari kolom tersebut berupa angka nol sampai dengan dua adalah hasil dari pembuatan *stats* kelemahan musuh pada variabel $DmgEN_M$ atau untuk $DmgEN_M = \{DmgNu_0, DmgNu_1, DmgNu_2, \dots, DmgNu_M\}$ jika jumlah kelemahan musuh lebih dari tiga atau yang dicontohkan pada Tabel 3.4 dalam variabel *List Damage*. Hasil dari proses yang sebelumnya dijelaskan pada Sub-bab ini menghasilkan persebaran kelemahan dan kekebalan musuh yang ditunjukkan pada Gambar 3.19 dibawah ini.



Gambar 3.19: Distribusi kelemahan musuh.

Pada Gambar 3.19 dapat dilihat distribusi musuh dengan kelemahan dan kekebalannya. Terlihat pada kondisi tersebut jumlah musuh dengan kondisi *Normal* memiliki nilai paling tinggi pada setiap kelemahan, hal tersebut dapat diartikan bahwa sebagian besar elemen dari musuh dapat memperoleh *damage* secara normal jika diserang. Kemudian jumlah musuh pada kondisi *Repel* berjumlah terbanyak kedua, hal tersebut dapat diartikan musuh memiliki jumlah kekebalan terhadap serangan sejumlah angka pada setiap elemen pada Gambar 3.19. Begitu juga dengan kondisi *Weaknesses* dengan jumlah paling

sedikit, hal tersebut bertujuan agar menjadikan pertarungan antara pemain dan musuh menjadi tidak mudah dimenangkan oleh pemain. Tidak hanya dengan efek *Weaknesses*, efek *Repel* juga sangat berpengaruh dalam menjadikan permainan seperti layaknya sebuah *puzzle* atau teka-teki.

Sedangkan pada persamaan 3.28 adalah penjelasan tentang proses munculnya setiap $DmgNu$ pada satu karakter musuh, sehingga variabel tersebut berubah menjadi $DmgNu_i$. Seperti penjelasan sebelumnya bahwa pada sebuah karakter musuh dapat memiliki lebih dari satu kelemahan atau kekebalan $DmgNu$ terhadap $DmgNa$ yang diset secara acak seperti yang sudah dijelaskan pada bagian sebelumnya.

3.3.4 Distribusi HP, MP dan Stats Musuh

Bila melihat pada Tabel 3.4 terdapat beberapa variabel seperti *List Stats Name*, *Max Stats Value*, dan *Min Stats Value*. Variabel tersebut akan digunakan untuk membuat *stats* dari musuh dengan basis algorima Naive Bayes. Di awali dengan persamaan 3.30 yang menjadi pembagi setiap operasi pembagian *stats* berdasarkan tipenya ET_i .

$$ET_N = \sum_{i=0}^N ET_i \begin{cases} ET_i = 0, & MX \\ ET_i = 1, & HM \\ ET_i = 2, & SM \\ ET_i = 3, & HS \\ ET_i = 4, & SS \end{cases} \quad (3.30)$$

Pada persamaan 3.30 adalah persamaan yang disesuaikan untuk menyelesaikan masukan variabel pada Tabel 3.4, sedangkan pada persamaan 3.31 adalah pengembangan jika jumlah tipe musuh dinaikkan lebih dari yang dicontohkan pada Tabel 3.4 pada variabel *Enemy Type*. Pada kedua persamaan tersebut terdapat beberapa variabel diantaranya adalah ET_i yang merupakan enemy tipe ke i , kemudian secara berurutan masing-masing *MX*, *HM*, *SM*, *HS*, dan *SS* adalah musuh dengan tipe *mixed*, *hard magic*, *soft magic*, *hard strength*, dan *soft strength*.

$$ET_N = \sum_{i=0}^N ET_i \begin{cases} ET_i = 0, & MX \\ ET_i = 1, & HM \\ ET_i = 2, & SM \\ ET_i = 3, & HS \\ ET_i = 4, & SS \\ \dots & \dots \\ ET_n = n, & ST_n \end{cases} \quad (3.31)$$

Kemudian pada persamaan 3.31 secara spesifik terdapat variabel ET_n yang merupakan variabel tipe musuh ke n atau batas jumlah musuh. Variabel n adalah urutan dari index tipe musuh yang dideskripsikan, berikut juga variabel ST_n , dengan variabel ST adalah *stats* dan n adalah batas jumlah musuh. Selanjutnya adalah dilakukannya pembagian tipe musuh jika musuh tersebut memiliki tipe *MX* atau *mixed* seperti pada persamaan 3.32.

$$MX_N = \sum_{i=0}^N MX_i \begin{cases} MX_i = 1, & MP Focused \\ MX_i = 2, & HP Focused \end{cases} \quad (3.32)$$

Pada persamaan 3.32 terdapat variabel MX_i yang merupakan variabel satuan dari musuh yang bertipe *mixed*. Kemudian pada variabel MX_N adalah variabel yang bersifat jamak atau memuat banyak variabel musuh yang bertipe *mixed*. Pada persamaan 3.32 sendiri dijelaskan bahwa pada tipe *mixed* dipecah menjadi dua tipe lagi, yang satu berfokus pada *HP* dan satu lagi pada *MP*. Jadi isi dari variabel MX_N nantinya akan terdiri dari musuh bertipe *mixed* dengan sub-tipe *MP* atau berfokus pada *MP* dan sub-tipe *HP* atau berfokus pada *HP*.

$$Mx_N = \sum_{i=0}^N Mx_i \begin{cases} Mx_i = 1, & MP Focused \\ Mx_i = 2, & HP Focused \\ \dots & \dots \\ Mx_N = n, & Lainnya \end{cases} \quad (3.33)$$

Selanjutnya adalah perhitungan probabilitas diperolehnya tipe musuh dari keseluruhan tipe musuh yang berada pada variabel masukan. Misal pada

kasus yang ditunjukan pada persamaan 3.34, jumlah tipe musuh tentunya mengacu pada Tabel 3.4. Dalam menyelesaikan kasus tersebut digunakanlah Naive bayes yang berbasis pada *conditional probability* seperti yangg dijelaskan pada Sub-bab 2.3.3, yang mana pada persamaan 3.34 proses kemunculan tipe musuh tertentu misal $ET_0, ET_1, ET_2, ET_3, ET_4$ yang mana seluruh tipe musuh dimuat pada variabel ET_N .

$$\begin{aligned} ET_0 &= P(ET_N|ET_0) \times P(ET_0) \\ ET_1 &= P(ET_N|ET_1) \times P(ET_1) \\ ET_2 &= P(ET_N|ET_2) \times P(ET_2) \\ ET_3 &= P(ET_N|ET_3) \times P(ET_3) \\ ET_4 &= P(ET_N|ET_4) \times P(ET_4) \end{aligned} \quad (3.34)$$

Sedangkan jika jumlah tipe musuh tidak beracuan pada Tabel 3.4 atau tipe musuh berjumlah lebih dari empat maka persamaan 3.34 akan berubah menjadi persamaan 3.35. Semula berawal dari variabel ET_0 sampai ET_4 pada persamaan 3.34 yang kemudian berubah dari variabel ET_0 sampai ET_n dengan n adalah batas akhir jumlah tipe musuh.

$$\begin{aligned} ET_0 &= P(ET_N|ET_0) \times P(ET_0) \\ ET_1 &= P(ET_N|ET_1) \times P(ET_1) \\ ET_2 &= P(ET_N|ET_2) \times P(ET_2) \\ &\dots \\ ET_n &= P(ET_N|ET_n) \times P(ET_n) \end{aligned} \quad (3.35)$$

Saat sudah diperolehnya tipe musuh pada seperti yang dijelaskan pada persamaan 3.35 maka selanjutnya yang harus dibuat adalah stats dari musuh itu sendiri, seperti halnya HP, MP, Strength, Magic dan lain sebagainya seperti yang tercantum pada Tabel 3.4 pada variabel *ListStatsName* yang ditambah dengan variabel HP dan MP.

$$bHP = minHP \quad (3.36)$$

Pada persamaan 3.36 adalah penentuan batas bawah untuk HP atau bHP , nilai $minHP$ juga dicontohkan dalam daftar variabel masukan dengan

nama *Min HP* pada Tabel 3.4.

$$tHP = \sum_{i=0}^N bHP + \left| \frac{Elv_i}{100} \times maxHP \right| \quad (3.37)$$

Selanjutnya dilanjutkan dengan pencarian batas atas untuk HP atau *tHP*. Pada persamaan 3.37 terdapat beberapa variabel yang berpengaruh terhadap nilai dari *tHP* diantaranya adalah *bHP* yang merupakan batas bawah dari *HP*, *Elv_i* yang merupakan level dari setiap karakter musuh dengan tipenya masing-masing, dan *maxHP* sendiri yang merupakan batas atas dicontohkan pada Tabel 3.4 dengan nama variabel *max HP*. Pencarian batas atas dilakukan dengan dilakukannya penjumlahan antara *bHP* dengan presentase dari *maxHP* yang disesuaikan dengan *Elv_i*. Jadi dari perhitungan tersebut HP dari karakter musuh akan menyesuaikan dengan level pada karakter musuh tersebut.

$$P(HP_i) = \sum_{i=0}^N \frac{HP_i}{bHP_i - tHP_i} \quad (3.38)$$

Kemudian dilanjutkan dengan perhitungan probabilitas $P(HP_i)$ atau munculnya HP pada setiap karakter musuh pada persamaan 3.38. HP dari setiap karakter musuh itu sendiri dinyatakan dengan HP_i . Kemudian range nilai dari HP_i yang akan muncul terhitung mulai dari bHP_i sampai dengan tHP_i dari setiap karakter musuh. Variabel i pada persamaan tersebut merepresentasikan setiap musuh dengan variabel N yang menyatakan batas atau jumlah dari musuh yang ingin dibuat.

$$bMP = minMP \quad (3.39)$$

Mengulang persamaan 3.36 pada persamaan 3.39, hanya mengganti variabelnya saja. Jika pada persamaan 3.36 kasus yang ingin diselesaikan adalah pencarian *stats HP*, maka pada persamaan 3.39 kasus yang ingin diselesaikan adalah pencarian *stats MP*. Pada persamaan 3.36 dengan variabel *bHP* diganti dengan *bMP* dan variabel *minHP* diganti dengan *minMP* yang mana pada

Tabel 3.4 dicontohkan dengan variabel *Min MP*.

$$tMP = \sum_{i=0}^N bMP + \left| \frac{Elv_i}{100} \times maxMP \right| \quad (3.40)$$

Masih sama seperti penjelasan sebelumnya dimana pada persamaan 3.39 pada persamaan 3.40 juga sama dengan persamaan 3.37. Di lakukan penggantian variabel HP menjadi MP, sudah dijelaskan juga pada bagian sebelumnya tentang variabel Elv_i , dan pada variabel $maxMP$ yang merupakan variabel yang dicontohkan pada Tabel 3.4 dengan nama *Max MP*.

$$P(MP_i) = \sum_{i=0}^N \frac{MP_i}{bMP_i - tMP_i} \quad (3.41)$$

Pada persamaan 3.41 adalah probabilitas $P(MP_i)$ atau munculnya MP pada setiap karakter musuh pada persamaan 3.41. MP dari setiap karakter musuh itu sendiri dinyatakan dengan MP_i . Kemudian range nilai dari MP_i yang akan muncul terhitung mulai dari bMP_i sampai dengan tMP_i . Kasus ini sama seperti pada persamaan 3.38 hanya saja sama seperti pada pembahasan sebelumnya yang mana pada persamaan tersebut bertujuan untuk mencari HP , sedangkan pada kasus ini bertujuan untuk mencari MP .

Pada bagian ini akan membahas secara khusus untuk kondisi *mixed* melanjutkan seperti yang ada pada persamaan 3.32. Dalam kondisi *mixed* sendiri terdapat dua kondisi yaitu HP *focused* atau yang berfokus kepada HP, kemudian MP *focused* atau yang befokus pada MP. Bila masing-masing tipe tersebut dinyatakan dengan persamaan secara berurutan untuk HP *focused* dan MP *focused* dengan persamaan 3.42 dan persamaan 3.43.

$$Mx_1 = P(Mx_N|Mx_1) \times P(Mx_1) \quad (3.42)$$

Maka Mx_N adalah seluruh musuh yang bertipe *mixed*, kemudian dipilihlah musuh dengan tipe mixed yang berfokus pada HP atau Mx_1 . Beracuan pada konsep *conditional probability* yang sudah dijelaskan pada Sub-bab 2.3.2.

$$Mx_2 = P(Mx_N|Mx_2) \times P(Mx_2) \quad (3.43)$$

Jika Mx_1 adalah HP *focused* maka Mx_2 adalah MP *focused* maka sama seperti persamaan 3.42 dengan Mx_N adalah seluruh musuh yang bertipe *mixed*, kemudian dipilihlah musuh dengan tipe *mixed* yang berfokus pada MP atau Mx_2 . Pembahasan secara khusus lainnya adalah penambahan *stats* HP untuk musuh dengan tipe *mixed* yang berfokus pada HP seperti yang dinyatakan pada persamaan 3.44. Sedangkan pada musuh dengan tipe *mixed* yang berfokus pada MP tidak perlu diberi perlakuan khusus seperti tipe *mixed* yang berfokus pada HP, hal tersebut dikarenakan dengan melihat nilai MP pada setiap karakter musuh dengan tipe tersebut sudah tergolong tinggi. Kondisi ini beracuan terhadap konsep keseimbangan dalam mendesain seperti yang dijelaskan pada Sub-bab 2.2.3 dan Sub-bab 3.1.2 tentang penyesuaian keseimbangan dalam pertarungan antara pemain dengan musuh berikut juga dengan cerita dari permainan tersebut.

$$tHP = \sum_{i=0}^N bHP + bHP + \left| \frac{Elv_i}{100} \times maxHP \right| \quad (3.44)$$

Sedikit berbeda dengan persamaan 3.38 pada persamaan 3.44 dilakukan penambahan dua kali pada variabel bHP atau batas bawah HP. Hal tersebut dilakukan agar angka *stats* HP pada musuh dengan tipe *mixed* yang berfokus terhadap HP memiliki nilai HP yang lebih tinggi jika dibandingkan dengan tipe *mixed* yang berfokus MP.

$$P(HP_i) = \sum_{i=0}^N \frac{HP_i}{bHP_i - tHP_i} \quad (3.45)$$

Kemudian perhitungan probabilitas $P(HP_i)$ atau munculnya HP pada setiap karakter musuh pada persamaan 3.45. HP dari setiap karakter musuh itu sendiri dinyatakan dengan HP_i . Kemudian range nilai dari HP_i yang akan muncul terhitung mulai dari bHP_i sampai dengan tHP_i . Persamaan tersebut masih sama dengan probabilitas kemunculan HP dari *range* antara bHP dengan tHP pada persamaan 3.38. Kemudian *Stats* untuk HP dan MP dari setiap

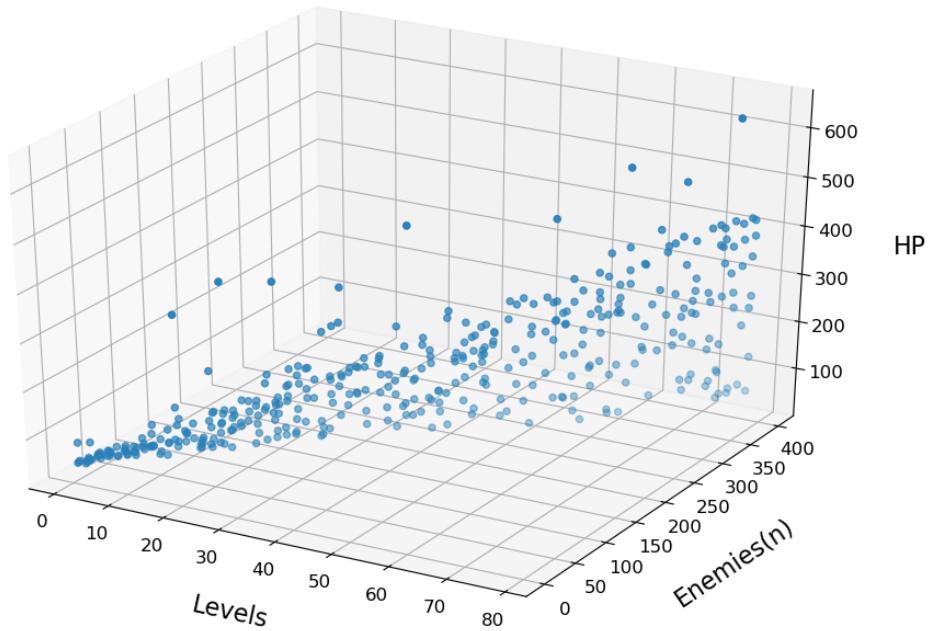
karakter musuh berhasil dibuat melalui beberapa langkah pada Sub-bab ini. Dari keseluruhan *stats* tersebut bila digabungkan maka hasilnya dapat dilihat pada Tabel 3.8.

Tabel 3.8: Distribusi *Stats* HP dan MP musuh.

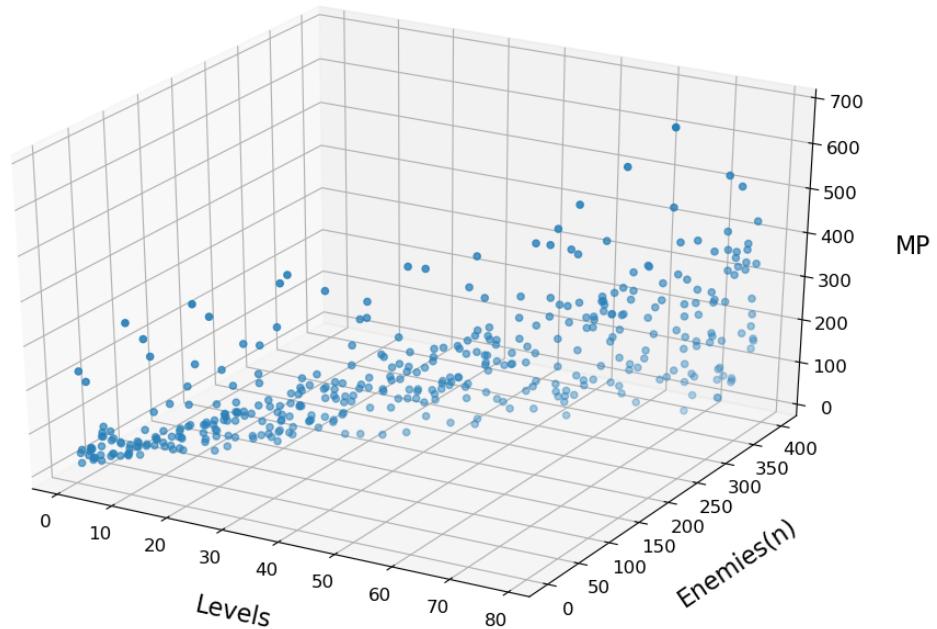
No.	Name	Levels	HP	MP
1	Enemy 1	1	42	45
2	Enemy 2	1	84	21
3	Enemy 3	1	43	51
4	Enemy 4	1	44	231
5	Enemy 5	2	40	40
6	Enemy 6	2	44	208
7	Enemy 7	2	44	47
8	Enemy 8	2	42	53
9	Enemy 9	2	49	53
10	Enemy 10	2	42	49
11	Enemy 11	2	44	23
12	Enemy 12	2	79	27
13	Enemy 13	2	41	26
14	Enemy 14	3	50	34
15	Enemy 15	3	54	20
...
400	Enemy 400	78	389	161

Pada Tabel 3.8 hanya sebagian data saja yang ditampilkan, untuk selengkapnya dapat dilihat pada Tabel 5.6 sampai dengan Tabel 5.20 pada kolom HP dan MP. Selanjutnya adalah penggambaran persebaran *stats* HP dan MP yang masing-masing digambarkan pada Gambar 3.20 dan Gambar 3.21.

Pada Gambar 3.20 dan Gambar 3.21 adalah visualisasi data HP dan MP dalam bentuk tiga dimensi, hal tersebut dilakukan agar terciptanya perbandingan antara level, jumlah musuh, dan tingkat tingginya nilai *stats* yang



Gambar 3.20: Distribusi *stats* HP musuh.



Gambar 3.21: Distribusi *stats* MP musuh.

bersangkutan. Selain pembuatan *stats* HP dan MP, *stats* lain seperti halnya *Strength*, *Magic*, *Endurance*, *Speed* dan *Luck* juga harus dicari. Daftar dari

stats tersebut dicontohkan pada Tabel 3.4 dengan nama variabel *List Stats Name*. Dalam kasus pembuatan *stats* musuh terdapat beberapa variabel lain digunakan selain *List Stats Name* diantaranya adalah *Max Stats Value* dan *Min Stats Value*, seperti yang ditunjukan pada Tabel 3.4. Dalam pembuatan *stats* musuh tersebut digunakanlah persamaan yang mirip dengan persamaan 3.36, 3.37, dan persamaan 3.38, hanya saja dilakukan sedikit perubahan pada persamaan tersebut seperti yang dinyatakan pada persamaan 3.46, 3.47, 3.48, 3.49, dan persamaan 3.50.

$$bSt = minSt \quad (3.46)$$

Jika hanya berlaku satu *stats* saja maka dapat digunakan persamaan 3.46 yang pada dasarnya sama seperti persamaan-persamaan sebelumnya dalam mencari batas bawah dari HP pada *bHP* dan MP pada *bMP*. Sedangkan pada persamaan 3.46 batas bawahnya adalah *bSt* yang diisi dengan nilai minimum dari *stats* atau *minSt*.

$$bSt_i = \sum_{i=0}^N minSt_i \quad (3.47)$$

Di karenakan *stats* dari musuh berjumlah lebih dari satu maka persamaan 3.46 harus disesuaikan dengan jumlah *stats* tersebut. Jumlah *stats* dari musuh sendiri berjumlah lima buah seperti yang tertulis pada variabel *List Stats Name*, maka persamaan 3.46 disesuaikan menjadi seperti persamaan 3.47. Dengan variabel *N* adalah batas maksimal dari jumlah *stats*, dengan nilai minimum *stats* atau *minSt* yang jumlahnya mengikuti jumlah maksimal *stats* menjadi *minSt_i* dan disimpan dalam setiap variabel *bSt* ke *i*. Kemudian selanjutnya adalah pencarian batas atas untuk masing-masing *stats*, sama halnya dengan pencarian batas bawah dari setiap *stats* dalam cara pencarian batas atas pada dasarnya juga sama dengan pencarian batas dari HP pada *tHP* dan MP pada *tMP*.

$$tSt_i = \sum_{i=0}^N bSt + \left| \frac{Elv_i}{100} \times maxSt_i \right| \quad (3.48)$$

Pada persamaan 3.48 batas atasnya adalah tSt yang diisi dengan nilai maximum dari $stats$ atau $maxSt$. Namun pada kasus ini jumlah $stats$ musuh berjumlah lebih dari satu seperti yang sudah dibahas pada bagian sebelumnya. Sama seperti pada pembahasan sebelumnya bahwa variabel N adalah jumlah $stats$ musuh, maka dari itu nilai maksimum $maxSt$ dan variabel penyimpan hasil batas maksimum tSt dieksekusi sesuai satu-satu sesuai jumlah $stats$ maka masing-masing variabel tersebut secara berurutan menjadi $maxSt_i$ dan tSt_i .

$$tSt_i = \sum_{i=0}^N bSt_i + \left| \frac{Elv_i}{100} \times maxSt_i \right| \quad (3.49)$$

Dalam pencarian nilai batas atas tSt_i tentunya harus melibatkan batas bawah. Jika melihat pada persamaan 3.48 maka terdapat variabel bSt , tapi pada variabel tersebut masih bersifat statis atau tidak menyesuaikan dengan jumlah $stats$ musuh. Digabunglah persamaan 3.47 dengan persamaan 3.48 sehingga diperolehnya persamaan 3.49 dengan variabel bSt yang mengikuti jumlah $stats$ musuh menjadi bSt_i dengan variabel N sebagai batasan maksimal jumlah $stats$. Dengan varaiabel i yang merepresentasikan urutan $stats$ musuh yang sedang diproses. Penjelasan untuk persamaan ini sebenarnya sama seperti persamaan 3.37 hanya saja jumlah $stats$ yang dibuat lebih banyak.

$$P(St_i) = \sum_{i=0}^N \frac{St_i}{bSt_i - tSt_i} \quad (3.50)$$

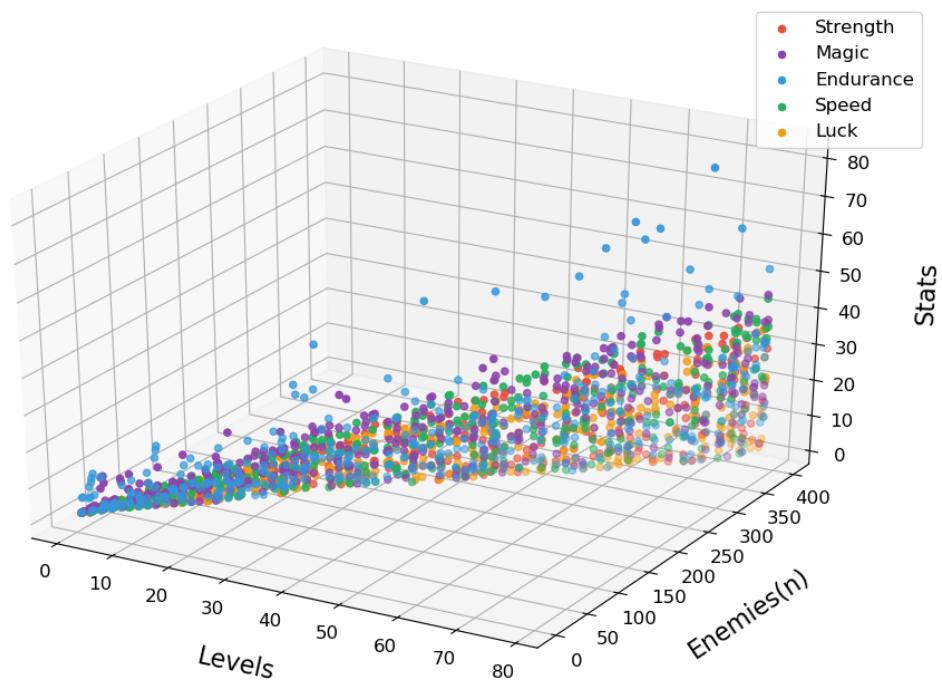
Kemudian pada persamaan 3.50 adalah probabilitas $P(St_i)$ saat munculnya $stats St_i$ dari batas bawah $stats bSt_i$ sampai batas atas $stats tSt_i$. $Stats$ dari setiap karakter musuh berhasil dibuat melalui banyak langkah pada Subbab ini, diantaranya adalah *Strength*, *Magic*, *Endurance*, *Speed*, dan *Luck*. Dari keseluruhan $stats$ tersebut bila digabungkan maka hasilnya dapat dilihat pada Tabel 3.8.

Tabel 3.9: Distribusi $Stats$ musuh.

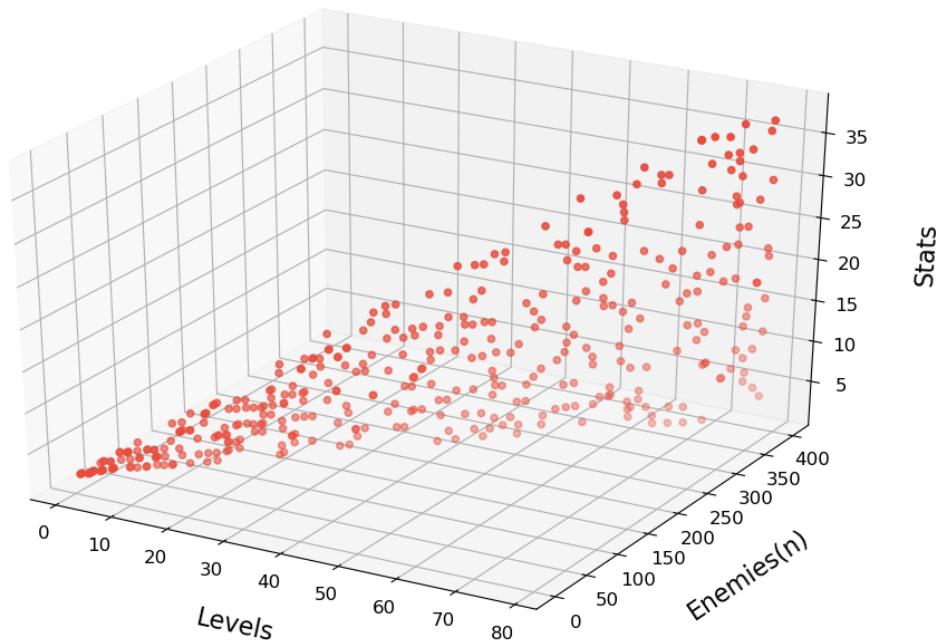
No.	Name	Lv.	Str	Mag	Endr	Spd	Luck
1	Enemy 1	1	2	2	2	2	2

2	Enemy 2	1	2	2	2	2	2
3	Enemy 3	1	2	2	6	2	2
4	Enemy 4	1	2	2	6	2	2
5	Enemy 5	2	2	3	6	3	2
6	Enemy 6	2	2	3	9	2	2
7	Enemy 7	2	2	3	6	2	2
8	Enemy 8	2	2	3	2	2	2
9	Enemy 9	2	2	3	10	3	2
10	Enemy 10	2	2	3	11	3	2
11	Enemy 11	2	2	3	2	2	2
12	Enemy 12	2	2	3	12	3	2
13	Enemy 13	2	2	3	6	2	2
14	Enemy 14	3	2	8	2	2	2
15	Enemy 15	3	3	11	2	3	2
...
400	Enemy 400	78	30	15	11	36	22

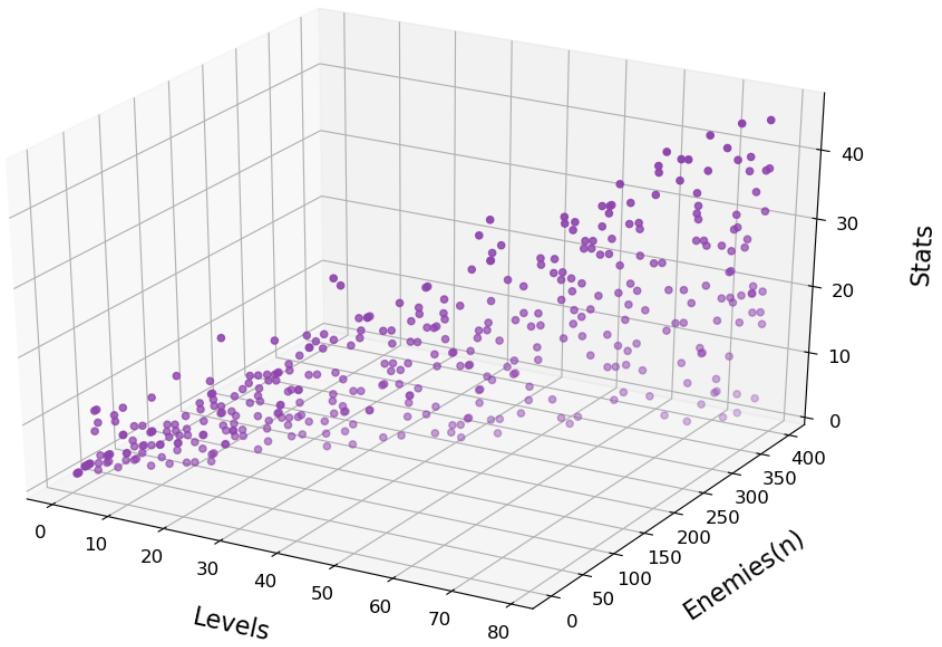
Selanjutnya adalah penggambaran persebaran *stats* yang disebutkan pada Tabel 3.9 yang digambarkan pada Gambar 3.22. Pada Gambar 3.23, 3.24, 3.25, 3.26 dan Gambar 3.27 adalah pecahan dari Gambar 3.22 secara berurutan diantaranya adalah *Strength*, *Magic*, *Endurance*, *Speed* dan *Luck*. Data *Stats* selengkapnya dapat dilihat pada Tabel 5.6 sampai dengan Tabel 5.20 pada kolom *Strength*, *Magic*, *Endurance*, *Speed*, *Luck* dan *stats* yang lain jika dilakukan penambahan *stats* pada program.



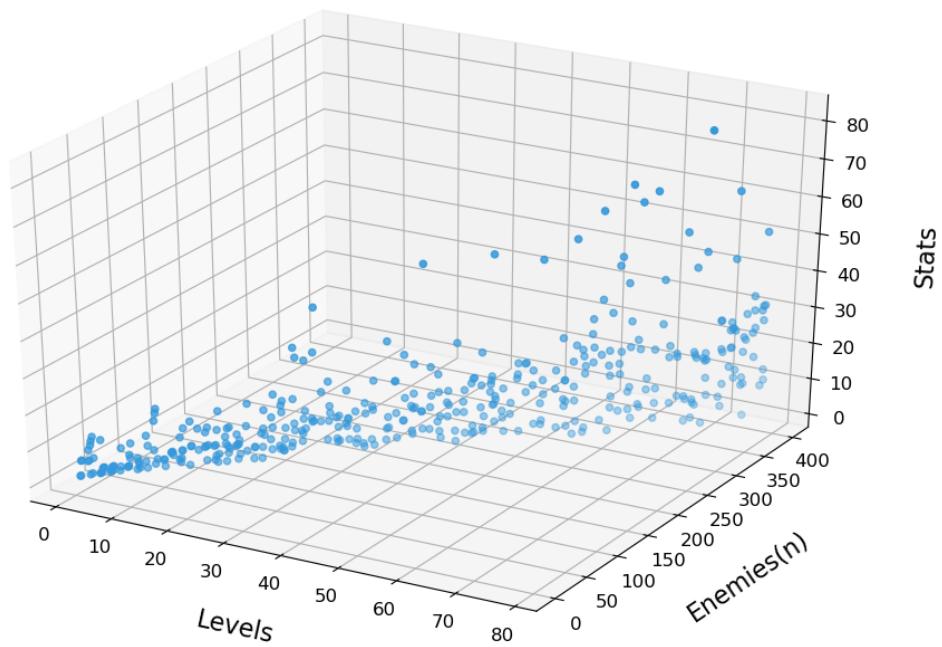
Gambar 3.22: Distribusi *stats* musuh secara keseluruhan.



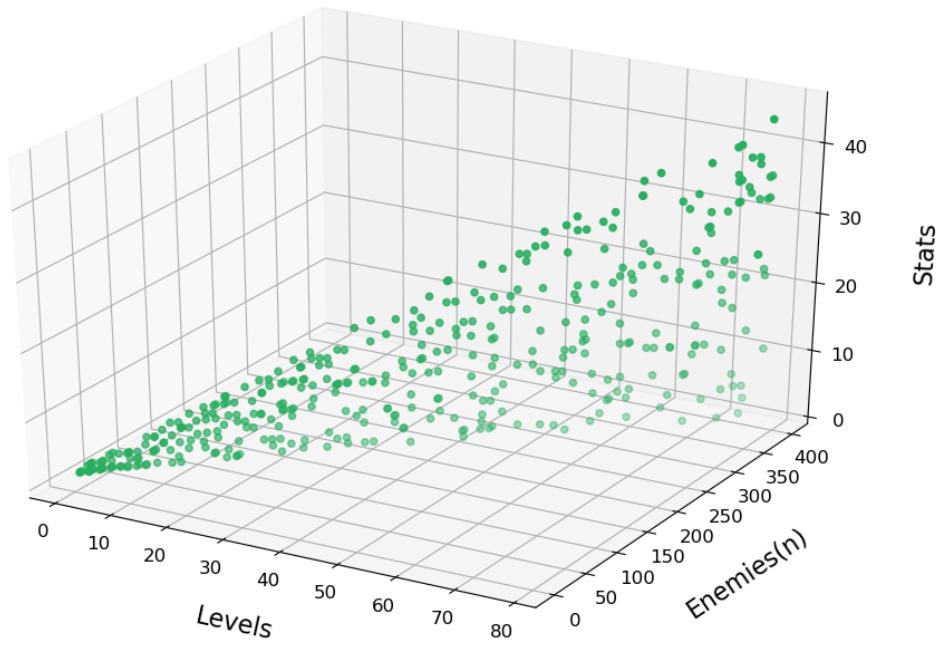
Gambar 3.23: Distribusi *Strength* musuh.



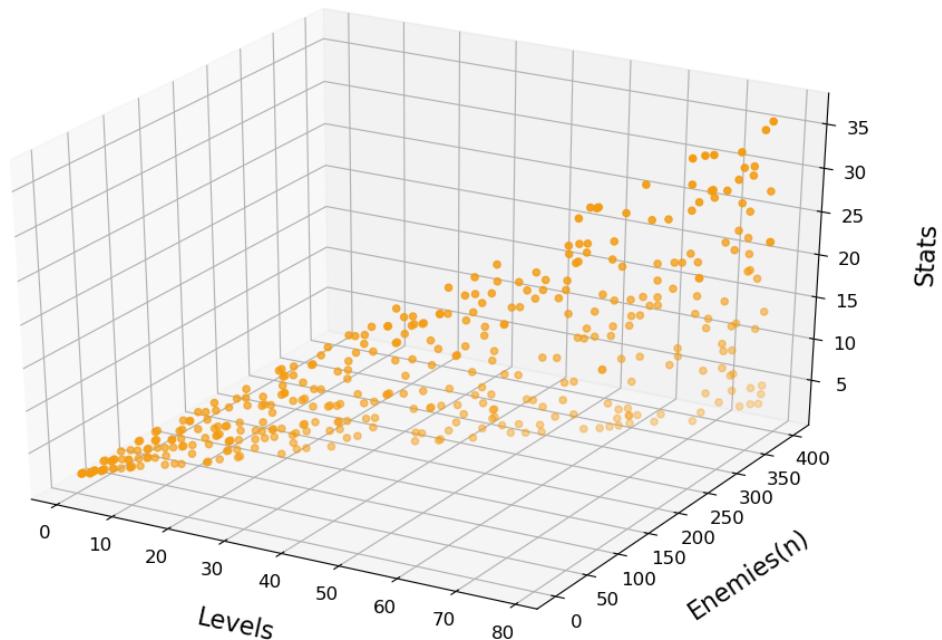
Gambar 3.24: Distribusi *Magic* musuh.



Gambar 3.25: Distribusi *Endurance* musuh.



Gambar 3.26: Distribusi *Speed* musuh.



Gambar 3.27: Distribusi *Luck* musuh.

Halaman ini sengaja dikosongkan

BAB 4

PENGUJIAN

Inti dari BAB 4 pada penelitian adalah pengujian dengan berbagai parameter untuk *stats* pemain dan musuh. Seperti halnya *stats* pemain dan musuh pada permainan dengan genre *turn-based* dan *action* RPG.

4.1 Pengujian dalam Pembuatan *Stats* Pemain pada Permainan dengan Genre *Turn-based* RPG

Pada bagian ini setiap langkah yang akan diuji sudah dijelaskan pada Sub-bab 3.2, dimana pada bagian ini membahas tentang pembuatan *stats* pemain untuk permainan dengan genre *turn-based* RPG. Melalui berbagai proses seperti yang dijelaskan pada Sub-bab 3.2, maka pada beberapa Sub-bab dibawah ini adalah langkah-langkah dalam proses pengujinya.

4.1.1 Distibusi Level, HP, dan MP Pemain

Pada bagian ini setiap langkah yang akan diuji sudah dijelaskan pada Sub-bab 3.2, dimana pada bagian ini membahas tentang pembuatan *stats* pemain untuk permainan dengan genre *turn-based* RPG. Melalui berbagai proses seperti yang dijelaskan pada Sub-bab 3.2, maka pada beberapa Sub-bab dibawah ini adalah langkah-langkah dalam proses pengujinya.

4.1.2 Distribusi Elemen dan Kelemahan Pemain

Pada bagian ini setiap langkah yang akan diuji sudah dijelaskan pada Sub-bab 3.2, dimana pada bagian ini membahas tentang pembuatan *stats* pemain untuk permainan dengan genre *turn-based* RPG. Melalui berbagai proses seperti yang dijelaskan pada Sub-bab 3.2, maka pada beberapa Sub-bab dibawah ini adalah langkah-langkah dalam proses pengujinya.

4.1.3 Distibusi stats pemain

Pada bagian ini setiap langkah yang akan diuji sudah dijelaskan pada Sub-bab 3.2, dimana pada bagian ini membahas tentang pembuatan *stats* pemain untuk permainan dengan genre *turn-based* RPG. Melalui berbagai proses seperti yang dijelaskan pada Sub-bab 3.2, maka pada beberapa Sub-bab dibawah ini adalah langkah-langkah dalam proses pengujinya.

4.2 Pengujian dalam Pembuatan *Stats* Musuh pada Permainan dengan Genre *Turn-based* RPG

Pada bagian ini setiap langkah yang akan diuji sudah dijelaskan pada Sub-bab 3.3, dimana pada bagian ini membahas tentang pembuatan *stats* musuh untuk permainan dengan genre *turn-based* RPG. Melalui berbagai proses seperti yang dijelaskan pada Sub-bab 3.2, maka pada beberapa Sub-bab dibawah ini adalah langkah-langkah dalam proses pengujinya.

4.3 Pengujian dalam Pembuatan *Stat* Pemain untuk Permainan dengan Genre *Action* RPG

Sama seperti pengujian yang dilakukan pada Sub-bab 4.1 yang mana langkahnya sudah dijelaskan pada Sub-bab 3.2, hanya saja pada bagian ini membahas tentang pembuatan dan penyesuaian *stats* pada permainan untuk genre *action* RPG. Melalui berbagai proses seperti yang dijelaskan pada Sub-bab 4.1 tentang *turn-based* RPG, hanya saja pada bagian ini tidak memerlukan elemen yang menggambarkan kelemahan dari karakter pemain yang mana hal inilah yang disebut dengan penyesuaian. Melalui beberapa Sub-bab dibawah ini adalah langkah-langkah dalam proses pengujinya.

4.4 Pengujian dalam Pembuatan *Stas* Musuh untuk Permainan dengan Genre *Action* RPG

Sama seperti pengujian yang dilakukan pada Sub-bab 4.1 yang mana langkahnya sudah dijelaskan pada Sub-bab 3.2, hanya saja pada bagian ini

membahas tentang pembuatan dan penyesuaian *stats* pada permaian untuk genre *action* RPG. Melalui berbagai proses seperti yang dijelaskan pada Subbab 4.1 tentang *turn-based* RPG, hanya saja pada bagian ini tidak memerlukan elemen yang menggambarkan kelemahan dari karakter pemain yang mana hal inilah yang disebut dengan penyesuaian. Melalui beberapa Sub-bab dibawah ini adalah langkah-langkah dalam proses pengujianya.

Halaman ini sengaja dikosongkan

BAB 5

PENUTUP

Setelah penerapan metode terhadap masalah yang ingin diselesaikan pada Bab ?? dan dilakukan pengujian dari metode tersebut pada Bab ?? maka didapatkan kesimpulan yang akan dijabarkan pada Sub-bab berikut. Kemudian dilanjutkan dengan saran untuk penelitian kedepannya pada Sub-bab selanjutnya.

5.1 Kesimpulan

Dengan menggunakan metode k -NN dalam melakukan proses ditribusi, khususnya pada stats dalam permainan *turn-based* dan *action* RPG. Maka stats dapat terdistribusi dengan baik. Hanya saja belum dicoba pada permainan yang sesungguhnya. Selain itu hal semacam ini juga sangat mempermudah dalam desain permainan, terlebih lagi untuk mendesain permainan dengan musuh dan karakter yang sangat banyak.

5.2 Saran

Dalam penelitian kedepannya sangat disarankan untuk meneliti tentang bagaimana mensimulasikan pertarungan secara otomatis dengan jenis *Action* dan *Turn-based* RPG. Hal tersebut bertujuan guna mencoba mensimulasikan nilai yang dihasilkan dari penelitian ini yang berupa *stats*.

Halaman ini sengaja dikosongkan

LAMPIRAN

Tabel 5.1: Hasil keseluruhan data HP dan MP pada pemain.

Levels	HP	MP
1	159	89
2	163	93
3	167	97
4	171	101
5	175	105
6	179	109
7	183	113
8	187	117
9	191	121
10	195	125
11	199	129
12	203	133
13	207	137
14	211	141
15	215	145
16	219	149
17	223	153
18	227	157
19	231	161
20	235	165
21	239	169
22	243	173
23	247	177
24	251	181
25	255	185
26	259	189
27	263	193
28	267	197
29	271	201
30	275	205
31	279	209
32	283	213
33	287	217
34	291	221
35	295	225

Levels	HP	MP
36	299	229
37	303	233
38	307	237
39	311	241
40	315	245
41	319	249
42	323	253
43	327	257
44	331	261
45	335	265
46	339	269
47	343	273
48	347	277
49	351	281
50	355	285
51	359	289
52	363	293
53	367	297
54	371	301
55	375	305
56	379	309
57	383	313
58	387	317
59	391	321
60	395	325
61	399	329
62	403	333
63	407	337
64	411	341
65	415	345
66	419	349
67	423	353
68	427	357
69	431	361
70	435	365

Levels	HP	MP
71	439	369
72	443	373
73	447	377
74	451	381
75	455	385
76	459	389
77	463	393
78	467	397
79	471	401
80	475	405
81	479	409
82	483	413
83	487	417
84	491	421
85	495	425
86	499	429
87	503	433
88	507	437
89	511	441
90	515	445
91	519	449
92	523	453
93	527	457
94	531	461
95	535	465
96	539	469
97	543	473
98	547	477
99	551	481
100	555	485

Tabel 5.2: Hasil keseluruhan data *stats* pada pemain (Bag. 1).

Levels	HP	MP	Phys	Water	Wind	Earth	Fire	Strength	Magic	Endurance	Speed	Luck
1	159	89	0	2	0	0	1	1	1	0	0	0
2	163	93	0	2	0	0	1	2	0	2	0	0
3	167	97	0	2	0	0	1	1	0	1	1	0
4	171	101	0	2	0	0	1	2	2	2	1	0
5	175	105	0	2	0	0	1	0	0	0	0	0
6	179	109	0	2	0	0	1	2	0	0	1	0
7	183	113	0	2	0	0	1	0	0	0	1	0
8	187	117	0	2	0	0	1	0	0	0	0	0
9	191	121	0	2	0	0	1	0	0	2	1	0
10	195	125	0	2	0	0	1	2	0	0	1	0
11	199	129	0	2	0	0	1	0	0	0	0	0
12	203	133	0	2	0	0	1	1	0	0	1	2
13	207	137	0	2	0	0	1	1	2	0	0	0
14	211	141	0	2	0	0	1	0	2	2	1	0
15	215	145	0	2	0	0	1	1	0	0	2	0
16	219	149	0	2	0	0	1	0	0	2	0	0
17	223	153	0	2	0	0	1	2	0	0	2	
18	227	157	0	2	0	0	1	1	0	0	1	2
19	231	161	0	2	0	0	1	1	0	2	0	0
20	235	165	0	2	0	0	1	1	0	0	0	2
21	239	169	0	2	0	0	1	1	0	2	1	0
22	243	173	0	2	0	0	1	1	0	0	1	1
23	247	177	0	2	0	0	1	0	0	0	1	0
24	251	181	0	2	0	0	1	0	0	2	2	1
25	255	185	0	2	0	0	1	0	1	0	1	0

Tabel 5.3: Hasil keseluruhan data *stats* pada pemain (Bag. 2).

Levels	HP	MP	Phys	Water	Wind	Earth	Fire	Strength	Magic	Endurance	Speed	Luck
26	259	189	0	2	0	0	1	1	0	0	0	2
27	263	193	0	2	0	0	1	1	0	0	1	0
28	267	197	0	2	0	0	1	1	0	0	1	0
29	271	201	0	2	0	0	1	1	0	2	0	1
30	275	205	0	2	0	0	1	1	0	0	0	0
31	279	209	0	2	0	0	1	1	2	0	0	1
32	283	213	0	2	0	0	1	1	0	0	1	1
33	287	217	0	2	0	0	1	1	0	0	1	2
34	291	221	0	2	0	0	1	0	0	0	1	2
35	295	225	0	2	0	0	1	0	0	0	1	1
36	299	229	0	2	0	0	1	1	0	0	0	1
37	303	233	0	2	0	0	1	0	0	2	0	0
38	307	237	0	2	0	0	1	0	0	2	1	0
39	311	241	0	2	0	0	1	0	2	2	2	0
40	315	245	0	2	0	0	1	2	0	0	1	1
41	319	249	0	2	0	0	1	1	0	0	1	0
42	323	253	0	2	0	0	1	2	0	2	0	2
43	327	257	0	2	0	0	1	0	2	2	1	2
44	331	261	0	2	0	0	1	2	1	2	0	0
45	335	265	0	2	0	0	1	1	0	0	0	0
46	339	269	0	2	0	0	1	1	0	2	0	0
47	343	273	0	2	0	0	1	0	0	1	1	0
48	347	277	0	2	0	0	1	0	2	0	1	0
49	351	281	0	2	0	0	1	0	2	0	1	2
50	355	285	0	2	0	0	1	2	0	0	1	1

Tabel 5.4: Hasil keseluruhan data *stats* pada pemain (Bag. 3).

Levels	HP	MP	Phys	Water	Wind	Earth	Fire	Strength	Magic	Endurance	Speed	Luck
51	359	289	0	2	0	0	1	1	0	0	0	0
52	363	293	0	2	0	0	1	0	2	0	1	2
53	367	297	0	2	0	0	1	1	0	2	0	0
54	371	301	0	2	0	0	1	2	0	0	1	0
55	375	305	0	2	0	0	1	2	0	0	0	0
56	379	309	0	2	0	0	1	2	0	0	0	2
57	383	313	0	2	0	0	1	2	0	0	0	0
58	387	317	0	2	0	0	1	0	0	2	0	1
59	391	321	0	2	0	0	1	1	2	0	1	1
60	395	325	0	2	0	0	1	1	0	2	1	0
61	399	329	0	2	0	0	1	0	0	0	0	1
62	403	333	0	2	0	0	1	0	2	2	1	0
63	407	337	0	2	0	0	1	0	2	2	1	0
64	411	341	0	2	0	0	1	0	2	0	1	1
65	415	345	0	2	0	0	1	1	0	0	1	0
66	419	349	0	2	0	0	1	0	0	2	1	2
67	423	353	0	2	0	0	1	1	0	0	1	0
68	427	357	0	2	0	0	1	1	0	2	1	0
69	431	361	0	2	0	0	1	0	0	0	1	0
70	435	365	0	2	0	0	1	2	1	2	1	0
71	439	369	0	2	0	0	1	1	0	0	1	1
72	443	373	0	2	0	0	1	0	0	0	0	0
73	447	377	0	2	0	0	1	0	0	0	1	1
74	451	381	0	2	0	0	1	1	0	0	1	2
75	455	385	0	2	0	0	1	1	0	0	0	0

Tabel 5.5: Hasil keseluruhan data *stats* pada pemain (Bag. 4).

Levels	HP	MP	Phys	Water	Wind	Earth	Fire	Strength	Magic	Endurance	Speed	Luck
76	459	389	0	2	0	0	1	1	0	0	1	0
77	463	393	0	2	0	0	1	0	0	2	1	2
78	467	397	0	2	0	0	1	1	0	2	1	1
79	471	401	0	2	0	0	1	2	0	0	0	1
80	475	405	0	2	0	0	1	1	0	2	1	0
81	479	409	0	2	0	0	1	1	0	0	1	2
82	483	413	0	2	0	0	1	1	0	0	0	0
83	487	417	0	2	0	0	1	0	2	2	1	0
84	491	421	0	2	0	0	1	0	2	0	0	0
85	495	425	0	2	0	0	1	0	0	0	1	0
86	499	429	0	2	0	0	1	0	0	0	0	0
87	503	433	0	2	0	0	1	1	0	2	0	0
88	507	437	0	2	0	0	1	0	2	0	1	0
89	511	441	0	2	0	0	1	0	0	2	0	0
90	515	445	0	2	0	0	1	1	0	0	0	0
91	519	449	0	2	0	0	1	0	0	2	0	0
92	523	453	0	2	0	0	1	0	0	0	1	0
93	527	457	0	2	0	0	1	1	0	0	1	2
94	531	461	0	2	0	0	1	1	2	0	1	2
95	535	465	0	2	0	0	1	0	0	0	0	0
96	539	469	0	2	0	0	1	1	0	0	2	0
97	543	473	0	2	0	0	1	2	0	0	1	2
98	547	477	0	2	0	0	1	0	0	0	0	2
99	551	481	0	2	0	0	1	0	0	2	1	1
100	555	485	0	2	0	0	1	0	0	0	1	0
										74	38	63
										TOTAL	74	65
											60	

Tabel 5.6: Hasil keseluruhan data *stats* pada musuh (Bag. 1).

Name	Levels	HP	MP	Type	Phys	Water	Wind	Earth	Fire	Strength	Magic	Endurance	Speed	Luck
Enemy 1	1	44	53	2	0	1	0	1	0	2	2	6	2	2
Enemy 2	1	41	21	0	0	0	0	1	0	2	2	5	2	2
Enemy 3	1	43	63	0	1	0	0	0	0	2	5	2	2	2
Enemy 4	1	42	52	0	1	0	0	0	2	0	2	3	2	2
Enemy 5	1	44	50	0	0	0	0	0	0	2	7	2	2	2
Enemy 6	1	76	23	0	0	0	0	1	0	2	2	5	2	2
Enemy 7	2	49	279	1	0	1	1	1	0	2	2	9	2	2
Enemy 8	2	41	53	0	0	1	1	0	1	2	10	2	3	2
Enemy 9	2	45	50	2	0	1	0	0	1	2	3	5	3	2
Enemy 10	2	42	24	4	2	0	0	2	0	2	2	2	2	2
Enemy 11	2	87	27	0	0	0	2	1	0	2	3	12	3	2
Enemy 12	2	46	48	2	2	0	0	0	0	2	2	5	2	2
Enemy 13	3	46	46	2	0	0	0	0	0	2	2	3	2	2
Enemy 14	3	54	25	3	1	0	1	0	0	3	7	3	2	2
Enemy 15	3	44	63	0	0	0	0	0	1	3	3	2	2	2
Enemy 16	4	99	21	0	0	0	0	2	0	3	4	6	4	3
Enemy 17	4	52	60	2	0	1	1	0	0	3	4	6	4	2
Enemy 18	4	44	37	4	1	1	0	0	3	4	3	4	4	2
Enemy 19	4	40	41	2	0	1	2	0	0	2	4	11	4	3
Enemy 20	5	65	36	3	0	0	1	1	0	4	7	2	3	4
Enemy 21	5	57	215	1	1	0	1	1	0	2	3	12	4	3
Enemy 22	5	46	149	1	1	1	0	1	0	2	2	7	2	4
Enemy 23	5	86	41	0	0	1	0	0	0	0	4	2	12	2
Enemy 24	5	40	22	4	2	0	0	2	0	2	3	2	2	3
Enemy 25	5	50	104	0	0	1	1	0	4	3	2	4	4	3

Tabel 5.7: Hasil keseluruhan data *stats* pada musuh (Bag. 2).

Name	Levels	HP	MP	Type	Phys	Water	Wind	Earth	Fire	Strength	Magic	Endurance	Speed	Luck
Enemy 26	5	60	62	2	1	0	0	1	0	3	2	3	3	3
Enemy 27	5	61	274	1	1	0	1	1	0	4	4	6	3	2
Enemy 28	6	57	59	2	1	1	0	0	1	2	3	7	5	2
Enemy 29	6	51	36	0	0	0	2	2	0	3	2	4	5	2
Enemy 30	6	41	22	4	1	0	0	1	1	3	6	3	2	2
Enemy 31	6	57	233	1	0	0	0	0	2	2	4	4	5	4
Enemy 32	6	62	24	4	0	2	0	0	1	3	11	2	2	3
Enemy 33	6	99	40	0	0	0	0	2	0	3	2	12	3	3
Enemy 34	6	94	36	0	0	0	0	1	0	4	2	3	2	2
Enemy 35	6	59	24	4	0	2	0	0	0	4	3	2	5	2
Enemy 36	7	57	75	0	0	0	1	0	1	5	10	4	3	4
Enemy 37	7	63	26	0	0	0	2	0	1	2	5	14	4	5
Enemy 38	7	63	330	1	1	0	0	0	1	2	3	7	4	4
Enemy 39	8	51	59	0	0	0	1	2	0	2	12	5	5	4
Enemy 40	8	78	50	3	2	0	0	2	0	2	10	4	3	5
Enemy 41	8	72	25	0	0	0	1	1	0	4	5	18	2	2
Enemy 42	8	66	73	2	0	0	1	1	1	3	2	3	5	3
Enemy 43	8	58	51	4	2	0	2	0	0	5	4	2	5	5
Enemy 44	8	75	50	0	1	0	1	0	0	3	5	14	4	2
Enemy 45	9	75	79	2	0	0	0	0	0	4	3	10	4	2
Enemy 46	9	49	50	0	1	0	0	0	1	5	7	2	6	3
Enemy 47	9	54	47	0	2	0	1	0	0	2	3	6	2	5
Enemy 48	9	64	291	1	1	1	1	0	0	6	7	10	4	4
Enemy 49	9	81	28	0	0	0	0	0	1	6	2	3	4	4
Enemy 50	9	53	57	0	0	1	0	0	2	6	8	4	3	2

Tabel 5.8: Hasil keseluruhan data *stats* pada musuh (Bag. 3).

Name	Levels	HP	MP	Type	Phys	Water	Wind	Earth	Fire	Strength	Magic	Endurance	Speed	Luck
Enemy 51	9	69	27	3	1	0	0	1	2	16	4	3	6	
Enemy 52	9	43	25	4	0	0	0	1	1	6	7	2	6	2
Enemy 53	9	74	57	4	1	0	1	1	0	3	7	2	3	6
Enemy 54	10	74	62	3	0	0	1	1	0	4	8	4	7	5
Enemy 55	10	80	52	4	1	0	0	0	1	5	5	4	5	2
Enemy 56	11	53	49	4	0	0	1	1	1	3	8	4	4	5
Enemy 57	11	71	78	2	0	0	0	0	0	6	8	2	4	4
Enemy 58	11	67	64	4	0	1	1	0	0	5	7	3	8	6
Enemy 59	11	65	77	0	1	0	0	0	0	7	7	2	3	2
Enemy 60	11	64	292	1	0	0	1	0	0	4	8	12	6	3
Enemy 61	11	56	88	0	0	1	1	0	1	7	5	3	2	4
Enemy 62	11	95	98	0	2	0	1	0	0	7	4	2	7	6
Enemy 63	12	80	56	3	0	0	2	0	0	3	8	2	3	2
Enemy 64	12	68	48	4	1	0	0	0	1	4	5	6	4	6
Enemy 65	12	50	77	0	0	0	0	1	2	2	9	3	6	4
Enemy 66	12	66	64	0	1	0	1	0	1	4	8	25	3	4
Enemy 67	12	90	59	4	0	0	2	0	0	3	4	3	4	5
Enemy 68	12	49	53	2	0	0	0	0	2	7	4	3	6	
Enemy 69	12	60	71	2	1	1	0	0	0	4	8	11	7	4
Enemy 70	12	50	66	0	0	0	0	0	0	6	9	5	2	3
Enemy 71	13	64	57	3	1	0	0	1	0	6	12	4	3	6
Enemy 72	13	96	51	4	0	1	0	0	0	7	11	6	8	3
Enemy 73	13	98	105	2	0	1	0	0	1	8	2	7	7	3
Enemy 74	14	54	54	0	0	0	0	0	0	9	7	23	7	3
Enemy 75	14	132	44	0	1	1	0	0	0	2	5	9	2	2

Tabel 5.9: Hasil keseluruhan data *stats* pada musuh (Bag. 4).

Name	Levels	HP	MP	Type	Phys	Water	Wind	Earth	Fire	Strength	Magic	Endurance	Speed	Luck
Enemy 76	14	105	41	3	1	0	0	0	1	8	19	5	8	7
Enemy 77	15	115	195	0	0	1	0	0	0	6	11	7	6	4
Enemy 78	15	102	107	2	0	0	1	0	1	3	8	6	2	7
Enemy 79	15	77	86	2	0	0	0	0	0	7	8	6	5	8
Enemy 80	15	101	67	4	2	0	2	0	0	8	6	2	6	2
Enemy 81	15	74	73	0	0	2	0	0	0	8	6	9	6	5
Enemy 82	15	260	82	3	0	0	0	0	0	2	10	7	8	4
Enemy 83	16	116	46	4	0	0	0	0	0	4	7	4	10	2
Enemy 84	17	126	40	0	0	0	2	0	0	4	12	20	11	4
Enemy 85	17	95	47	3	0	0	1	0	2	8	8	7	10	2
Enemy 86	17	104	20	0	0	1	1	0	0	5	12	22	3	8
Enemy 87	17	94	50	0	0	0	1	0	0	6	11	11	3	2
Enemy 88	17	57	100	0	1	0	1	0	1	8	3	2	6	8
Enemy 89	17	100	72	0	0	2	0	0	2	9	10	10	6	4
Enemy 90	17	73	69	0	0	0	0	0	0	8	7	17	8	9
Enemy 91	17	117	121	2	0	1	0	1	0	10	5	7	8	8
Enemy 92	17	63	63	2	0	0	2	0	0	2	12	9	2	4
Enemy 93	18	82	39	3	0	0	1	0	0	4	15	6	6	10
Enemy 94	18	127	89	3	0	0	0	2	0	6	8	3	6	4
Enemy 95	18	109	78	4	0	2	1	0	0	10	2	3	8	2
Enemy 96	18	112	300	1	1	0	0	1	0	10	12	20	3	3
Enemy 97	18	88	33	3	0	0	1	1	1	5	16	2	2	6
Enemy 98	18	90	101	0	1	0	1	0	1	6	8	2	11	2
Enemy 99	19	109	263	1	1	0	1	1	0	8	10	12	8	8
Enemy 100	19	132	137	2	0	0	0	0	1	9	5	9	4	2

Tabel 5.10: Hasil keseluruhan data *stats* pada musuh (Bag. 5).

Name	Levels	HP	MP	Type	Phys	Water	Wind	Earth	Fire	Strength	Magic	Endurance	Speed	Luck
Enemy 101	19	56	49	3	0	1	0	1	1	9	14	3	8	5
Enemy 102	19	89	80	0	0	1	0	1	0	4	10	28	11	7
Enemy 103	19	50	34	4	0	2	0	2	0	3	8	3	3	5
Enemy 104	19	71	75	0	0	1	0	1	1	7	4	2	4	7
Enemy 105	19	84	92	0	2	0	0	2	0	7	7	3	6	2
Enemy 106	19	354	90	3	0	0	0	0	2	10	10	4	4	10
Enemy 107	20	126	128	2	0	2	0	0	0	2	6	4	5	8
Enemy 108	20	100	108	2	0	2	1	0	0	6	7	9	9	9
Enemy 109	20	54	51	0	0	0	1	0	1	2	6	9	10	6
Enemy 110	21	73	88	0	0	0	0	0	0	8	10	7	3	4
Enemy 111	21	110	101	4	1	0	0	1	1	8	3	9	13	4
Enemy 112	21	105	33	4	0	0	2	0	2	10	13	10	9	10
Enemy 113	21	54	120	0	0	0	1	1	0	10	13	3	12	3
Enemy 114	21	56	115	0	1	0	0	1	1	8	10	3	13	6
Enemy 115	21	60	59	4	1	0	0	1	1	3	14	9	2	3
Enemy 116	22	51	82	0	0	0	0	0	2	4	7	4	13	9
Enemy 117	22	149	261	0	2	0	0	2	0	2	18	10	5	5
Enemy 118	22	115	120	2	1	0	0	0	1	8	6	6	11	4
Enemy 119	22	148	236	0	0	0	0	1	1	5	9	6	8	2
Enemy 120	22	81	87	2	0	0	0	0	0	10	14	7	13	8
Enemy 121	22	101	103	2	0	0	1	0	0	10	10	5	9	4
Enemy 122	23	124	333	1	0	2	0	1	0	4	4	18	8	10
Enemy 123	23	105	153	0	2	0	0	1	0	12	14	4	4	2
Enemy 124	23	138	66	4	1	0	1	0	0	9	12	11	13	6
Enemy 125	23	141	36	3	0	1	1	1	0	11	20	11	8	4

Tabel 5.11: Hasil keseluruhan data *stats* pada musuh (Bag. 6).

Name	Levels	HP	MP	Type	Phys	Water	Wind	Earth	Fire	Strength	Magic	Endurance	Speed	Luck
Enemy 126	23	132	134	2	0	1	0	1	1	5	6	11	5	9
Enemy 127	23	159	112	0	0	0	1	0	0	11	5	7	8	4
Enemy 128	23	94	36	4	2	0	0	1	0	8	2	7	13	4
Enemy 129	24	139	236	0	1	0	0	1	0	4	15	7	9	9
Enemy 130	24	126	48	0	0	0	0	0	0	8	16	26	8	3
Enemy 131	24	57	123	1	1	0	0	1	1	4	6	7	10	3
Enemy 132	24	106	105	0	0	1	0	2	0	8	10	27	5	10
Enemy 133	24	131	133	1	1	0	0	1	1	8	2	13	11	9
Enemy 134	25	102	102	2	0	1	0	1	1	9	4	8	5	9
Enemy 135	25	78	126	0	1	1	0	0	0	4	7	5	6	7
Enemy 136	25	140	131	4	1	1	0	0	0	13	7	9	4	3
Enemy 137	26	72	72	2	0	0	1	1	0	3	5	5	16	2
Enemy 138	26	103	102	4	0	2	0	2	0	11	5	4	15	12
Enemy 139	26	81	73	4	0	1	1	0	1	11	10	7	16	4
Enemy 140	26	75	257	1	1	2	0	0	0	11	14	22	7	10
Enemy 141	27	138	132	4	0	2	0	0	1	6	5	12	16	11
Enemy 142	27	152	69	0	1	0	1	0	0	4	8	24	13	8
Enemy 143	27	192	57	0	0	0	0	0	0	8	18	37	4	2
Enemy 144	28	85	94	2	0	2	0	1	0	15	15	3	4	4
Enemy 145	28	48	84	0	0	0	2	2	0	15	15	3	6	13
Enemy 146	28	98	83	4	1	0	2	0	0	4	12	10	16	11
Enemy 147	28	151	159	2	0	1	1	1	0	16	16	4	15	14
Enemy 148	28	126	22	4	0	0	2	0	0	7	3	13	2	11
Enemy 149	28	130	291	1	1	1	0	0	1	12	4	17	4	9
Enemy 150	29	48	29	0	0	0	2	2	0	7	18	43	11	4

Tabel 5.12: Hasil keseluruhan data *stats* pada musuh (Bag. 7).

Name	Levels	HP	MP	Type	Phys	Water	Wind	Earth	Fire	Strength	Magic	Endurance	Speed	Luck
Enemy 151	29	143	136	4	0	0	0	1	0	13	9	12	7	2
Enemy 152	29	127	118	0	0	0	0	1	1	7	4	5	13	12
Enemy 153	29	95	267	1	0	0	1	0	1	12	19	21	17	5
Enemy 154	29	124	130	2	1	1	0	0	0	9	14	9	6	15
Enemy 155	30	47	124	2	1	0	1	0	0	9	3	8	8	2
Enemy 156	30	113	210	0	0	1	2	0	0	6	13	11	11	13
Enemy 157	30	121	121	4	0	1	1	0	1	13	18	4	12	7
Enemy 158	30	175	354	0	0	1	1	0	0	5	15	6	3	14
Enemy 159	30	151	162	2	0	0	1	0	0	13	5	11	16	9
Enemy 160	30	148	163	0	0	0	0	0	0	13	11	9	8	12
Enemy 161	30	158	152	4	0	0	0	2	0	10	8	13	15	8
Enemy 162	30	163	168	2	0	1	0	1	1	13	18	8	7	8
Enemy 163	31	59	70	0	1	1	0	0	0	2	16	11	9	2
Enemy 164	31	180	188	2	2	0	0	0	2	10	15	6	13	7
Enemy 165	31	125	68	4	0	1	1	0	1	16	17	3	9	14
Enemy 166	31	159	151	3	0	0	0	0	0	10	9	5	10	4
Enemy 167	31	189	342	1	0	0	1	1	0	10	20	25	5	9
Enemy 168	32	102	162	2	1	0	1	0	0	14	10	8	4	5
Enemy 169	32	85	138	0	1	0	0	1	1	10	15	2	13	12
Enemy 170	33	42	49	2	0	2	0	0	0	7	15	5	13	11
Enemy 171	33	117	111	4	0	0	1	0	2	15	20	13	13	16
Enemy 172	33	230	69	0	2	0	0	2	0	15	15	20	13	11
Enemy 173	33	108	187	0	2	0	0	0	2	0	8	17	15	11
Enemy 174	33	70	74	2	2	0	0	0	0	1	12	13	14	8
Enemy 175	33	40	96	0	1	0	0	1	0	5	14	11	12	16

Tabel 5.13: Hasil keseluruhan data *stats* pada musuh (Bag. 8).

Name	Levels	HP	MP	Type	Phys	Water	Wind	Earth	Fire	Strength	Magic	Endurance	Speed	Luck
Enemy 176	33	124	170	2	2	0	1	0	0	18	10	5	16	8
Enemy 177	33	128	232	1	1	0	1	0	1	6	18	24	14	6
Enemy 178	35	271	90	3	0	0	1	0	2	14	17	15	10	10
Enemy 179	35	167	164	4	0	1	0	1	1	4	9	3	10	4
Enemy 180	35	124	96	3	1	1	1	0	0	16	22	9	7	12
Enemy 181	35	139	100	4	2	0	0	0	1	18	11	14	10	6
Enemy 182	35	144	148	2	0	0	2	2	0	17	21	11	4	3
Enemy 183	36	57	166	0	2	0	0	0	2	18	14	3	2	17
Enemy 184	36	138	134	4	0	0	0	1	1	9	23	14	5	15
Enemy 185	36	133	135	0	0	1	0	0	0	19	7	2	5	3
Enemy 186	37	189	181	4	1	1	1	0	0	16	8	2	3	5
Enemy 187	37	191	115	0	0	0	2	0	0	13	20	44	4	17
Enemy 188	37	200	193	0	1	2	0	0	0	3	22	25	9	8
Enemy 189	37	67	102	0	1	2	0	0	0	8	15	12	2	12
Enemy 190	38	189	161	0	0	2	0	0	0	16	13	14	3	12
Enemy 191	38	50	145	2	0	0	1	1	1	20	9	4	8	3
Enemy 192	38	382	197	3	0	0	0	2	1	19	10	4	19	3
Enemy 193	39	116	259	1	0	1	0	1	0	5	7	14	11	9
Enemy 194	39	139	216	0	0	0	0	0	1	12	19	12	9	18
Enemy 195	39	53	128	0	2	0	1	0	0	3	11	10	16	10
Enemy 196	39	209	133	3	1	1	0	0	1	10	20	6	23	14
Enemy 197	39	124	70	4	0	0	1	0	0	14	13	9	19	19
Enemy 198	39	163	161	0	1	0	0	0	1	11	14	22	12	15
Enemy 199	39	162	171	1	0	1	0	0	1	19	14	20	6	17
Enemy 200	39	149	202	0	0	0	1	0	2	9	20	16	16	7

Tabel 5.14: Hasil keseluruhan data *stats* pada musuh (Bag. 9).

Name	Levels	HP	MP	Type	Phys	Water	Wind	Earth	Fire	Strength	Magic	Endurance	Speed	Luck
Enemy 201	39	79	51	0	0	0	1	0	0	15	11	13	19	14
Enemy 202	39	184	116	4	1	0	0	1	0	3	19	13	2	14
Enemy 203	40	101	216	1	0	0	0	0	2	12	25	26	11	2
Enemy 204	40	226	27	3	0	0	1	1	0	14	16	3	5	11
Enemy 205	40	73	135	2	0	0	1	0	1	8	11	10	20	12
Enemy 206	40	397	208	3	0	1	0	0	1	11	15	14	20	16
Enemy 207	40	102	100	4	0	0	0	0	0	17	7	3	2	11
Enemy 208	40	168	21	4	2	0	0	0	0	3	3	16	6	14
Enemy 209	41	214	413	0	0	0	0	1	1	2	20	11	9	7
Enemy 210	41	124	210	0	0	2	0	0	1	9	23	8	5	7
Enemy 211	41	170	314	0	0	2	0	0	1	10	20	8	12	12
Enemy 212	42	64	164	2	0	1	1	0	0	20	16	2	8	4
Enemy 213	42	213	211	0	0	1	0	1	1	15	26	51	21	18
Enemy 214	42	192	254	0	1	1	0	0	0	12	19	17	16	14
Enemy 215	42	40	211	2	1	1	0	0	0	9	19	4	9	8
Enemy 216	43	206	294	0	1	1	0	0	0	1	3	17	16	3
Enemy 217	43	237	245	2	1	0	2	0	0	11	4	18	8	13
Enemy 218	43	76	34	0	0	0	1	2	0	8	10	29	14	17
Enemy 219	43	165	93	4	0	0	1	0	0	12	19	16	14	21
Enemy 220	43	94	171	2	2	0	1	0	0	15	22	2	16	21
Enemy 221	44	179	128	4	1	0	0	1	1	4	3	8	14	2
Enemy 222	44	235	236	2	0	0	1	1	0	6	21	7	15	18
Enemy 223	44	134	94	0	0	0	1	0	1	20	6	8	10	21
Enemy 224	45	193	140	0	0	1	1	0	0	2	14	16	7	4
Enemy 225	45	71	184	2	0	0	0	2	2	21	25	9	7	13

Tabel 5.15: Hasil keseluruhan data *stats* pada musuh (Bag. 10).

Name	Levels	HP	MP	Type	Phys	Water	Wind	Earth	Fire	Strength	Magic	Endurance	Speed	Luck
Enemy 226	45	204	198	0	0	0	1	0	0	14	10	16	26	22
Enemy 227	45	136	136	0	0	0	0	0	2	5	7	11	13	3
Enemy 228	45	247	396	1	1	0	2	0	0	21	4	19	10	8
Enemy 229	45	78	357	1	0	0	1	0	1	7	4	15	21	11
Enemy 230	45	191	57	4	0	0	1	0	0	8	23	15	11	3
Enemy 231	45	173	170	4	1	0	0	0	0	3	5	14	17	10
Enemy 232	46	209	0	0	0	2	0	0	1	2	20	28	19	18
Enemy 233	46	137	244	0	1	0	0	0	0	18	20	19	2	21
Enemy 234	46	152	235	0	1	0	1	0	1	2	24	9	16	3
Enemy 235	46	300	68	0	1	1	0	0	1	19	7	17	16	20
Enemy 236	46	74	40	3	1	0	1	0	0	17	15	2	2	10
Enemy 237	46	156	186	0	0	2	2	0	0	14	16	3	2	14
Enemy 238	47	227	376	1	0	0	0	1	0	18	10	10	8	13
Enemy 239	47	74	49	4	1	0	1	1	0	20	10	3	2	9
Enemy 240	47	103	106	2	0	0	2	0	2	7	19	11	26	13
Enemy 241	47	157	65	0	0	1	1	0	0	11	10	19	17	3
Enemy 242	47	265	148	0	0	1	0	0	1	10	29	58	21	23
Enemy 243	47	215	221	2	0	0	2	0	0	13	25	7	3	19
Enemy 244	48	307	238	0	0	0	0	2	2	21	3	5	28	15
Enemy 245	48	286	450	1	1	0	0	2	0	25	9	12	17	19
Enemy 246	48	130	338	1	1	1	0	1	0	11	6	15	21	3
Enemy 247	49	109	83	4	0	0	0	1	0	16	3	10	2	5
Enemy 248	49	218	344	0	0	0	1	1	0	5	14	2	4	15
Enemy 249	49	203	212	1	1	1	0	1	0	19	21	22	13	8
Enemy 250	49	290	181	3	0	0	0	0	2	20	24	2	24	9

Tabel 5.16: Hasil keseluruhan data *stats* pada musuh (Bag. 11).

Name	Levels	HP	MP	Type	Phys	Water	Wind	Earth	Fire	Strength	Magic	Endurance	Speed	Luck
Enemy 276	56	314	196	0	0	1	0	1	0	15	25	50	30	24
Enemy 277	57	312	69	0	0	0	2	1	0	11	19	23	26	10
Enemy 278	57	185	180	4	0	2	0	0	2	21	6	23	26	26
Enemy 279	57	461	263	3	0	0	0	0	0	13	35	19	12	3
Enemy 280	57	242	324	0	1	0	1	0	0	4	28	22	33	3
Enemy 281	57	67	90	1	2	0	0	0	2	7	34	43	21	10
Enemy 282	57	166	267	0	1	0	0	0	2	9	14	11	23	10
Enemy 283	57	266	267	2	0	0	1	0	2	26	8	2	29	24
Enemy 284	57	283	165	3	0	0	0	0	2	9	29	24	2	18
Enemy 285	57	124	131	2	0	1	1	0	1	3	18	5	4	21
Enemy 286	58	54	31	4	0	1	1	1	0	22	25	11	25	15
Enemy 287	58	314	97	4	1	1	0	0	1	20	22	19	19	5
Enemy 288	58	149	30	0	0	0	2	1	0	4	15	27	21	5
Enemy 289	58	311	184	3	0	0	0	1	0	11	25	2	5	12
Enemy 290	59	256	256	0	0	0	0	0	0	25	23	46	28	9
Enemy 291	59	71	199	0	1	0	0	1	0	21	35	8	28	23
Enemy 292	59	59	300	0	0	0	0	0	0	20	25	16	10	28
Enemy 293	59	215	206	3	0	0	2	0	2	29	19	7	8	20
Enemy 294	59	303	300	0	0	0	0	0	0	31	5	24	6	19
Enemy 295	59	112	117	2	0	1	0	1	0	8	23	9	8	11
Enemy 296	60	343	594	1	0	0	0	2	0	17	37	39	27	11
Enemy 297	60	159	154	4	0	0	0	1	1	9	8	4	6	14
Enemy 298	60	52	56	2	0	0	1	1	1	2	33	7	17	5
Enemy 299	61	347	351	0	1	0	0	1	1	28	38	5	20	5
Enemy 300	61	144	172	0	0	2	0	0	29	27	21	10	10	24

Tabel 5.17: Hasil keseluruhan data *stats* pada musuh (Bag. 12).

Name	Levels	HP	MP	Type	Phys	Water	Wind	Earth	Fire	Strength	Magic	Endurance	Speed	Luck
Enemy 301	61	284	252	0	0	0	1	0	1	7	38	67	12	6
Enemy 302	61	306	310	2	0	0	0	0	0	7	6	10	19	20
Enemy 303	61	121	99	4	0	0	0	0	1	12	34	7	22	12
Enemy 304	61	274	272	4	0	1	0	0	1	28	29	26	5	20
Enemy 305	62	215	208	0	1	0	0	1	1	16	32	47	10	14
Enemy 306	62	92	72	0	1	0	0	0	0	5	4	11	6	26
Enemy 307	62	249	251	2	0	0	1	0	0	0	29	35	7	10
Enemy 308	62	283	281	4	0	0	1	0	1	17	13	12	31	22
Enemy 309	62	186	422	1	0	2	0	0	1	11	8	25	30	24
Enemy 310	62	296	59	3	1	0	1	0	1	26	22	10	34	13
Enemy 311	62	75	46	4	0	0	0	1	0	27	6	15	7	12
Enemy 312	63	324	43	4	1	0	0	1	1	32	26	23	21	21
Enemy 313	63	400	265	0	1	1	1	0	0	6	31	54	21	30
Enemy 314	63	277	180	0	0	0	0	1	0	9	5	13	33	3
Enemy 315	63	46	118	0	1	1	1	0	0	28	16	16	32	25
Enemy 316	63	206	206	2	1	0	0	1	1	33	15	7	24	26
Enemy 317	63	102	32	4	1	0	1	1	0	8	18	7	7	12
Enemy 318	64	299	43	4	0	0	2	1	0	11	9	24	22	11
Enemy 319	64	132	139	2	0	0	1	1	0	4	7	18	29	30
Enemy 320	64	323	314	4	0	0	0	0	0	20	21	11	32	28
Enemy 321	64	306	297	4	0	0	0	1	1	11	17	7	31	11
Enemy 322	64	230	444	1	2	0	0	0	2	16	5	20	31	29
Enemy 323	65	332	112	0	0	1	1	0	1	22	17	32	29	11
Enemy 324	65	315	303	0	1	0	0	0	1	2	3	14	7	26
Enemy 325	65	250	301	2	0	0	0	1	0	33	28	6	7	19

Tabel 5.18: Hasil keseluruhan data *stats* pada musuh (Bag. 13).

Name	Levels	HP	MP	Type	Phys	Water	Wind	Earth	Fire	Strength	Magic	Endurance	Speed	Luck
Enemy 326	66	250	253	2	0	1	0	2	0	28	10	8	4	17
Enemy 327	66	293	318	2	0	0	1	0	1	25	7	19	15	
Enemy 328	66	268	318	0	0	0	0	1	0	23	28	28	30	14
Enemy 329	66	284	137	0	0	0	0	1	0	2	41	84	22	23
Enemy 330	67	179	275	1	0	1	0	0	0	7	32	41	38	4
Enemy 331	67	193	240	2	0	0	1	1	0	30	12	13	2	14
Enemy 332	67	410	208	0	0	0	2	0	2	2	13	21	4	10
Enemy 333	67	580	286	3	0	0	0	0	0	31	38	11	16	29
Enemy 334	68	187	179	3	0	1	0	1	0	14	23	18	24	23
Enemy 335	68	296	595	1	1	2	0	0	0	14	34	34	13	16
Enemy 336	68	243	358	1	1	0	0	1	1	21	38	43	24	4
Enemy 337	69	198	192	4	0	0	0	1	0	14	9	9	36	15
Enemy 338	69	104	337	2	1	0	0	0	0	19	19	27	32	22
Enemy 339	69	301	341	0	0	0	0	1	1	31	15	12	39	20
Enemy 340	70	250	244	4	0	0	0	0	0	8	43	4	15	18
Enemy 341	70	307	298	0	0	0	1	0	1	34	7	10	4	9
Enemy 342	70	362	61	4	0	1	2	0	0	4	10	27	23	22
Enemy 343	70	55	333	0	0	0	0	0	0	5	10	2	25	23
Enemy 344	70	356	171	3	0	0	0	0	0	5	21	8	34	13
Enemy 345	70	273	264	0	0	0	2	0	0	8	43	67	2	4
Enemy 346	70	280	231	4	0	0	0	0	0	23	31	24	22	27
Enemy 347	70	222	137	3	0	1	0	0	0	3	41	7	2	12
Enemy 348	71	224	220	0	1	1	0	0	0	30	42	45	3	2
Enemy 349	71	141	186	2	0	1	0	1	1	24	9	27	15	5
Enemy 350	71	310	309	4	0	0	1	1	0	16	23	5	36	26

Tabel 5.19: Hasil keseluruhan data *stats* pada musuh (Bag. 14).

Name	Levels	HP	MP	Type	Phys	Water	Wind	Earth	Fire	Strength	Magic	Endurance	Speed	Luck
Enemy 351	71	40	350	0	1	0	1	0	1	4	40	7	22	29
Enemy 352	71	240	279	0	0	1	1	0	0	14	29	27	26	23
Enemy 353	71	226	226	2	0	2	0	0	2	28	3	19	3	3
Enemy 354	71	409	120	0	2	0	0	1	0	34	43	54	27	11
Enemy 355	71	83	281	2	0	0	1	1	1	37	39	7	11	14
Enemy 356	71	404	407	2	1	0	1	0	0	20	34	2	21	30
Enemy 357	71	245	236	4	1	0	1	1	0	23	5	16	7	2
Enemy 358	71	153	365	1	1	0	0	0	0	20	8	17	17	16
Enemy 359	72	138	31	4	1	0	1	1	0	33	11	29	37	22
Enemy 360	72	433	158	0	1	1	0	0	1	15	42	81	9	33
Enemy 361	72	41	139	2	0	0	0	1	1	22	35	4	37	12
Enemy 362	72	268	268	2	1	0	1	1	0	2	14	30	22	28
Enemy 363	72	354	58	4	1	0	0	0	1	23	9	27	14	19
Enemy 364	72	195	346	0	1	0	1	1	0	4	32	29	6	30
Enemy 365	72	220	25	4	0	2	2	0	0	2	6	28	26	27
Enemy 366	72	152	155	2	0	1	1	0	0	3	33	3	30	25
Enemy 367	72	341	252	4	0	0	0	1	1	15	4	29	7	33
Enemy 368	73	138	132	4	0	0	0	2	2	17	12	7	18	29
Enemy 369	73	229	346	2	0	0	0	0	1	14	19	9	5	8
Enemy 370	73	217	387	0	1	0	0	0	0	33	42	4	6	22
Enemy 371	73	257	260	2	0	1	0	0	0	33	25	8	5	23
Enemy 372	74	61	127	0	1	0	1	0	0	36	46	8	33	35
Enemy 373	74	332	620	0	1	0	0	1	0	33	30	23	16	24
Enemy 374	74	194	155	0	0	0	1	0	2	26	19	26	29	12
Enemy 375	74	345	337	0	2	0	2	0	0	18	6	11	38	17

Tabel 5.20: Hasil keseluruhan data *stats* pada musuh (Bag. 15).

Name	Levels	HP	MP	Type	Phys	Water	Wind	Earth	Fire	Strength	Magic	Endurance	Speed	Luck
Enemy 376	74	295	289	0	0	1	0	0	1	15	9	12	18	35
Enemy 377	75	268	501	1	2	0	0	2	0	7	35	38	12	26
Enemy 378	75	323	228	0	1	0	0	0	1	9	25	32	11	6
Enemy 379	75	374	366	4	0	0	0	1	0	19	28	23	42	33
Enemy 380	75	371	34	0	0	0	1	0	1	1	26	10	18	30
Enemy 381	75	60	58	4	1	0	1	1	0	9	35	27	29	24
Enemy 382	75	178	223	0	0	1	0	0	0	38	22	7	15	26
Enemy 383	76	294	289	4	1	1	0	1	0	4	5	30	38	15
Enemy 384	76	222	62	4	0	1	0	0	1	9	41	12	39	11
Enemy 385	76	579	303	3	2	0	2	0	0	11	38	24	12	7
Enemy 386	76	197	290	0	0	0	0	0	1	12	18	5	28	36
Enemy 387	76	154	292	1	0	1	0	0	0	33	39	40	22	4
Enemy 388	76	177	162	0	1	1	1	0	0	24	28	31	36	29
Enemy 389	76	433	268	0	1	0	0	1	1	20	39	65	15	13
Enemy 390	77	300	400	1	0	1	1	0	0	17	18	26	21	9
Enemy 391	77	427	113	0	0	0	1	0	0	10	39	53	19	11
Enemy 392	77	222	40	3	1	1	0	0	1	29	35	30	13	3
Enemy 393	77	467	296	3	0	2	0	0	0	21	44	14	31	4
Enemy 394	77	350	359	2	0	1	1	0	1	2	7	14	6	12
Enemy 395	77	475	358	0	1	0	0	1	0	3	36	43	4	23
Enemy 396	78	300	199	3	0	2	0	1	0	27	29	12	23	5
Enemy 397	78	267	441	0	0	1	0	0	1	37	29	17	12	25
Enemy 398	78	146	201	0	0	0	0	0	1	31	23	19	11	10
Enemy 399	78	75	261	2	0	0	1	0	0	22	30	2	43	15
Enemy 400	78	63	337	0	0	1	0	0	0	24	41	33	20	34

Daftar Pustaka

- Bangay, S. and Makin, O. (2014). Generating an attribute space for analyzing balance in single unit RTS game combat. IEEE Conference on Computational Intelligence and Games, CIG, pages 1–8. (Dikutip pada halaman 2).
- Brathwaite, B. and Schreiber, I. (2009). Challenges for game designers. Course Technology. (Dikutip pada halaman 6, 9, 12).
- Christyowidiasmoro, Putra, R. C. A., and Susiki, S. M. (2016). Measuring level of difficulty in game using challenging rate (CR) on 2D Real time Strategy Line Defense game. Proceedings - 2015 International Electronics Symposium: Emerging Technology in Electronic and Information, IES 2015. (Dikutip pada halaman 2).
- Cover, T. M. and Hart, P. E. (1967). Nearest Neighbor Pattern Classification. IEEE Transactions on Information Theory. (Dikutip pada halaman 15).
- Friedman, H. J. (1997). Data mining and statistics: What's the connection? Department of Statistics and Standford Linear Accelerator Center, Standford University. (Dikutip pada halaman 14).
- Koster, R. (2013). Theory of fun for game design. “O'Reilly Media, Inc.”. (Dikutip pada halaman 9).
- Koza, J. R., Bennett, F. H., Andre, D., and Keane, M. A. (1996). Automated Design of Both the Topology and Sizing of Analog Electrical Circuits Using Genetic Programming. Springer Netherlands. (Dikutip pada halaman 14).
- Marín-Lora, C., Chover, M., Sotoca, J. M., and García, L. A. (2020). A game engine to make games as multi-agent systems. Advances in Engineering Software, 140(April 2019):102732. (Dikutip pada halaman 2).
- Panumate, C., Xiong, S., and Iida, H. (2015). An approach to quantifying Pokemon's entertainment impact with focus on battle. Proceedings - 3rd International Conference on Applied Computing and Information Technology and 2nd International Conference on Computational Science and Intelligence, ACIT-CSI 2015, pages 60–66. (Dikutip pada halaman 30, 31).
- Wu, Z. H., Lai, K., Lin, L. A., Huang, M. H., and Tai, W. K. (2018). Procedurally Generating Game Level with Specified Difficulty. 2018 IEEE Games, Entertainment, Media Conference, GEM 2018, pages 71–78. (Dikutip pada halaman 2).

Halaman ini sengaja dikosongkan

BIOGRAFI PENULIS



Nur Rohman Widiyanto, lahir pada 26 Maret 1991 di Lamongan, Jawa Timur. Penulis menempuh pendidikan Sarjana di Jurusan Teknik Elektro ITS Surabaya angkatan 2009, kemudian mengambil bidang studi Teknik Komputer dan Telematika. Saat di kuliah penulis aktif menjadi asisten laboratorium Telematika (B201) dan pernah menjabat sebagai koordinator asisten Lab B201 periode 2011/2012. Selama masa kuliah penulis aktif dalam mengikuti ajang perlombaan seperti PKM (Program Kreatifitas Mahasiswa), aktif dalam *development group networking* dan juga sebagai *administrator* jaringan Lab B201. Kemudian penulis menyelesaikan penelitian atau Tugas Akhir dengan judul “Pengukuran Performansi Mesin Virtual pada Komputasi Awalan”. Di sisi lain penulis sangat tertarik dengan segala hal yang berhubungan dengan komputer, dan berencana mendalami cabang ilmu komputer lain selain jaringan komputer. Pada tahun 2017 penulis resmi melanjutkan studi Master di bidang keahlian Jaringan Cerdas Multimedia, Jurusan Teknik Elektro, FTE ITS. Sebelum resmi melanjutkan studi, penulis pernah bekerja sebagai *Lead Network Engineer* di PT. Niltava Teknologi Indonesia. Selama menempuh studi Master penulis memfokuskan diri dalam penelitian tentang kecerdasan buatan, maka dibuatlah penelitian dengan judul “Validasi Pengaturan Stats Otomatis pada Permainan Action dan Turn-Based Role-Playing Games (RPG) Berbasis K-NN dan Naive Bayes dengan Deep Learning Multiclass Classification”. Penulis dapat dihubungi melalui email: rohwid@gmail.com dan telepon di +6285731925725.

Halaman ini sengaja dikosongkan