Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR - KI141502

PENERAPAN *DYNAMIC DIFFICULTY ADJUSTMENT* PADA *TYPING GAME WORD MASTER*

Antonio Cahyadi Limantara
NRP 5110100 187

Dosen Pembimbing
Imam Kuswardayan, S.Kom., M.T.
Ridho Rahman H., S.Kom., M.Sc.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

[Halaman ini sengaja dikosongkan]

TUGAS AKHIR – KI141502

PENERAPAN *DYNAMIC DIFFICULTY ADJUSTMENT* PADA *TYPING GAME WORD MASTER*

Antonio Cahyadi Limantara
NRP 5110100 187

Dosen Pembimbing
Imam Kuswardayam, S.Kom., M.T.
Ridho Rahman H., S.Kom., M.Sc.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

[Halaman ini sengaja dikosongkan]

FINAL PROJECT – KI141502

DYNAMIC DIFFICULTY ADJUSTMENT IMPLEMENTATION FOR WORD MASTER TYPING GAME

Antonio Cahyadi Limantara
NRP 5110100 187

Advisor
Imam Kuswardayan, S.Kom., M.T.
Ridho Rahman H., S.Kom., M.Sc.

DEPARTMENT OF INFORMATICS
Faculty of Information Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2017

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

Penerapan *Dynamic Difficulty Adjustment* pada *Typing Game Word Master*

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Interaksi Grafika Dan Seni
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh :
ANTONIO CAHYADI LIMANTARA
NRP : 5110 100 187

Disetujui oleh Dosen Pembimbing tugas akhir :

IMAM KUSWARDAYAN, S.Kom., M.Kom. .....
NIP: 197612152003121001 (pembimbing 1)

RIDHO RAHMAN H., S.Kom., M.Sc. .....
NIP: 198702132014041001 (pembimbing 2)

SURABAYA
JUNI 2017

[Halaman ini sengaja dikosongkan]

Penerapan *Dynamic Difficulty Adjustment* pada *Typing Game Word Master*

Nama Mahasiswa	: Antonio Cahyadi Limantara
NRP	: 5110100187
Jurusan	: Teknik Informatika FTIf-ITS
Dosen Pembimbing 1	: Imam Kuswardayan, S.Kom., M.T.
Dosen Pembimbing 2	: Ridho Rahman H., S.Kom, M.Sc.

ABSTRAK

Komputer sudah menjadi bagian penting dalam gaya hidup sebagian besar masyarakat. Kebanyakan pengguna komputer menggunakan komputer bukan hanya dalam kapasitas personal, tetapi juga dalam kapasitas akademis dan juga bisnis.

Kemampuan mengetik yang tinggi tentunya akan membantu dalam kegiatan keseharian semisal membuat laporan baik akademis maupun bisnis, membuat proposal, dan lain sebagainya. Akan tetapi banyak pengguna komputer tidak dapat mengetik dengan effisien.

Typing game Word Master dirancang untuk menjadi alat bantu pembelajaran kemampuan mengetik yang menarik. Typing game Word Master memiliki tingkat kesulitan yang dinamis sehingga dapat mengakomodasi pengguna dari berbagai tingkat kesulitan.

Hasil uji coba menunjukkan bahwa Dynamic Difficulty Adjustment dapat mengubah tingkat kesulitan game secara gradual. Perubahan tingkat kesulitan ini membuat game tetap dapat memberikan tantangan kepada pemain dengan tingkat kemampuan tinggi dan tetap dapat dinikmati oleh pemain dengan tingkat kemampuan rendah.

Kata kunci**: **Game, Typing Game, Unity, C#, Dynamic Difficulty Adjusment.

[Halaman ini sengaja dikosongkan]

DYNAMIC DIFFICULTY ADJUSTMENT IMPLEMENTATION FOR TYPING GAME WORD MASTER

Student Name	: Antonio Cahyadi Limantara
Student ID	: 5110100187
Major	: Teknik Informatika FTIf-ITS
Advisor 1	: Imam Kuswardayan, S.Kom., M.T.
Advisor 2	: Ridho Rahman H., S.Kom, M.Sc.

ABSTRACT

Computer has already become an important part of most people's lifestyle. Most users use computer not only in personal capacity, but also in academic or business capacity.

High competence in typing will surely help people in their daily activities such as making an academic or business report, making a proposal, et cetera. Alas, many computer users aren't capable of typing efficiently.

Typing game Word Master is designed to be a fun and interesting learning tool in learning to type. Typing game Word Master has a dynamic difficulty as to accommodate users from varying skill levels.

The result showed that Dynamic Difficulty Adjustment can make gradual change to the game's difficulty. This changes make it so that the game is challenging to players with high competence level while still being enjoyable to players with low competence level.

Keyword**: **Game, Typing Game, Unity, C#, Dynamic Difficulty Adjusment.

[Halaman ini sengaja dikosongkan]

KATA PENGANTAR

Penulis memanjatkan puji dan syukur kepada Tuhan Yang Maha Esa karena berkat segala rahmat-Nya penulis dapat menyelesaikan tugas akhir yang berjudul:

"Penerapan Dynamic Difficulty Adjustment pada Typing Game Word Master"

Semoga apa yang tertulis dalam buku tugas akhir ini dapat memberikan manfaat kepada pengembangan sejenis nantinya.

Penulis ingin mengucapkan terima kasih kepada:

1. Ayah Benny, Ibu Eny, Kak Andre dan Kak Ari karena penulis hanya dapat mencapai jenjang sejauh ini dengan bantuan penuh kasih selama ini.
2. Bapak Imam Kuswardayan dan Bapak Ridho Rahman Hariadi selaku dosen pembimbing yang telah sangat membantu dalam berbagai aspek pengerjaan tugas akhir ini, baik berupa saran, masukan, tambahan, dan nasihat.
3. Bapak, Ibu dosen Jurusan Teknik Informatika ITS yang telah membimbing penulis selama masa studi di Teknik Informatika ITS.
4. Staf dan karyawan Teknik Informatika ITS yang telah membantu penulis dari sisi administrasi akademik.
5. Angkatan 2010 Teknik Informatika ITS yang berjuang bersama selama ini.
6. Dan pihak-pihak lain yang turut membantu dalam pengerjaan tugas akhir ini.

Surabaya, Juni 2017

Antonio Cahyadi Limantara

[Halaman ini sengaja dikosongkan]

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

[Halaman ini sengaja dikosongkan]

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SUMBER

[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

Pada bab ini akan dipaparkan mengenai garis besar tugas akhir yang meliputi latar belakang, tujuan, rumusan dan batasan permasalahan, metodologi pembuatan tugas akhir, dan sistematika penulisan.

1.1. Latar Belakang

Kemampuan mengetik dengan 10 jari merupakan hal yang penting bagi seseorang yang menggunakan perangkat elektronik seperti komputer atau laptop dalam kesehariannya. Mengetik dengan menggunakan 10 jari dapat meningkatkan kecepatan mengetik tanpa perlu melihat *keyboard*. Tentu saja hal ini memberikan banyak keuntungan semisal dapat menyelesaikan laporan atau tugas dalam waktu yang lebih singkat.

Selain itu, memiliki kemampuan mengetik dengan menggunakan 10 jari juga menambah peluang untuk mendapatkan pekerjaan. Tidak sedikit perusahaan yang menambahkan persyaratan kecepatan mengetik minimum. Mengetik dengan 10 jari akan menghasilkan kecepatan mengetik yang maksimal.

Akan tetapi, banyak orang tidak dapat mengetik dengan efisien. Hal ini disebabkan kurangnya alat bantu pembelajaran mengetik dengan 10 jari dan alat bantu yang ada kurang memadai karena memiliki tingkat kesulitan yang statis sehingga tidak dapat mengakomodasi keberagaman tingkat kemampuan mengetik penggunanya.

Typing game Word Master dirancang untuk menjadi alat bantu pembelajaran kemampuan mengetik yang menarik dengan tingkat kesulitan yang dinamis sehingga dapat mengakomodasi pengguna dari berbagai tingkat kesulitan.

1.2. Tujuan

Tujuan dari pembuatan tugas akhir ini adalah membuat sebuah *typing game* bernama *Word Master*.

1.3. Rumusan Permasalahan

Rumusan masalah dari tugas akhir ini adalah sebagai berikut.

1. Bagaimana merancang aturan main untuk *typing game* pada *game Word Master*?
2. Bagaimana merancang scenario untuk *typing game* pada *game Word Master*?
3. Bagaimana menentukan tingkat kesulitan?
4. Bagaimana mengimplementasikan *Dynamic Diffculty Adjustment*?
5. Bagaimana mengimplementasikan *game*?

1.4. Batasan Permasalahan

Batasan masalah dari tugas akhir ini adalah sebagai berikut.

1. *Game* dirancang menggunakan *game engine* Unity.
2. *Game* dibuat menggunakan bahasa pemrograman C#

1.5. Metodologi

Langkah-langkah yang ditempuh dalam pengerjaan tugas akhir ini yaitu:

1. Studi literatur

Pada tahap ini dilakukan pengumpulan dan penggalian informasi dan literatur yang diperlukan dalam proses perancangan dan implementasi sistem yang akan dibangun. Literatur yang digunakan adalah teknik pemrograman untuk pengembangan *game* dua dimensi menggunakan Unity dengan bahasa pemrograman C#.

2. Analisis dan Perancangan Sistem

Pada tahap ini dilakukan analisis dan pendefinisian kebutuhan sistem untuk masalah yang sedang dihadapi. Selanjutnya, dilakukan perancangan sistem dengan beberapa tahap sebagai berikut:

a. perancangan aturan *game* ;
b. perancangan skenario;
c. perancangan antar muka sistem; dan
d. perancangan diagram kelas sistem.

3. Implementasi

Pada tahap ini dilakukan pembuatan elemen perangkat lunak. Sistem yang dibuat berpedoman pada rancangan yang telah dibuat pada proses perancangan dan analisis sistem.

Perincian tahap ini adalah sebagai berikut:

a. implementasi aturan *game* dengan memanfaatkan Unity; dan
b. implementasi *Dynamic Difficulty Adjustment*.

4. Pengujian dan evaluasi

Pada tahap ini akan dilakukan pengujian terhadap perangkat lunak menggunakan skenario yang telah disiapkan sebelumnya. Uji coba dan evaluasi dilakukan untuk mencari masalah yang memiliki kemungkinan muncul, mengevaluasi apakah jalan program sesuai dengan keinginan, dan melakukan perbaikan ketika kekurangan ditemukan. Pengujian bersifat *white-box* akan dilakukan pada tahap ini. Pengujian bersifat *white-box* adalah metode pengujian perangkat lunak yang memeriksa struktur dan cara kerja internal dari aplikasi tersebut, tanpa memeriksa fungsionalitasnya. Metode yang akan digunakan dalam pengujian adalah sebagai berikut.

1. *Unit Testing*

Unit testing dilakukan untuk memastikan kode berjalan sebagaimana mestinya. Metode ini mencoba apakah bagian dari kode, prosedur penggunaan, dan prosedur operasi layak digunakan. Metode ini akan menguji faktor *correctness*.

2. *Integration Testing*

Integration testing dilakukan untuk menguji interaksi antar modul satu dengan yang lainnya. Modul yang ada akan digabungkan

menjadi satu untuk memastikan modul satu dengan yang lainnya dapat berinteraksi dengan lancar.

5. **Penyusunan buku tugas akhir**

Pada tahap ini dilakukan pendokumentasian dan pelaporan dari seluruh konsep, dasar teori, implementasi, proses yang telah dilakukan, dan hasil-hasil yang telah didapatkan selama pengerjaan tugas akhir.

1.6. Sistematika Penulisan

Buku tugas akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan tugas akhir ini. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku tugas akhir terdiri atas beberapa bagian seperti berikut ini.

Bab I Pendahuluan

Bab ini berisi latar belakang masalah, tujuan dan manfaat pembuatan tugas akhir, permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penyusunan tugas akhir.

Bab II Dasar Teori

Bab ini membahas beberapa teori penunjang yang berhubungan dengan pokok pembahasan dan mendasari pembuatan tugas akhir ini.

Bab III Analisis dan Perancangan Sistem

Bab ini membahas mengenai perancangan perangkat lunak. Perancangan perangkat lunak meliputi perancangan data, arsitektur, proses dan perancangan antarmuka pada kakas.

Bab IV Implementasi

Bab ini berisi implementasi dari perancangan perangkat lunak.

Bab V Pengujian dan Evaluasi

Bab ini membahas pengujian dengan metode pengujian subjektif untuk mengetahui penilaian aspek kegunaan

(*usability*) dari perangkat lunak dan pengujian hasil analisis kakas.

Bab VI Kesimpulan

Bab ini berisi kesimpulan dari hasil pengujian yang dilakukan. Bab ini membahas saran-saran untuk pengembangan sistem lebih lanjut.

Daftar Pustaka

Merupakan daftar referensi yang digunakan untuk mengembangkan tugas akhir.

Lampiran

Merupakan bab tambahan yang berisi daftar istilah yang penting pada aplikasi ini.

BAB II
DASAR TEORI

Pada bab ini akan dibahas mengenai teori-teori yang menjadi dasar dari pembuatan tugas akhir. Teori-teori tersebut meliputi Unity, C#, *typing game*, *dynamic difficulty adjusment*, Mono Framework, dan Mono Behaviour.

2.1. Unity

Unity merupakan sebuah ekosistem pengembangan *game* yang terintegrasi, dan kaya akan alat atau perlengkapan yang sangat berguna untuk membangun *game* interaktif seperti *lighting*, *special effect*, *animation* dan *physics engine*. Unity dapat digunakan untuk membangun *game* dengan grafis tiga dimensi atau dua dimensi. Unity juga dapat digunakan untuk melakukan perubahan maupun pengujian *game* yang dibuat, dan ketika sudah siap *game* dapat dipublikasikan ke berbagai macam perangkat yang mendukung Unity seperti Mac, PC, Linux, Windows Phone, Android, Blackberry, Wii U, PS3, dan Xbox 360. [1]

Gambar 2.1. Tampilan editor Unity.

2.1.1. Scene

Setiap object di dalam *game* yang dibuat oleh Unity diletakkan di dalam sebuah *scene*. *Scene* digunakan untuk membuat menu utama, level, dan lain sebagainya. *Scene* juga dapat dianalogikan sebagai level. Di dalam sebuah *scene*, penulis dapat meletakan entitas, dekorasi permaian dan lain sebagainya untuk dibangun menjadi sebuah *game*.

2.1.2. *GameObject*

GameObject adalah bagian terkecil dari sebuah *game* yang dibangun menggunakan Unity. *Programmer* dapat menambahkan atribut dan juga *class* program ke dalam sebuah *GameObject*. Setiap *GameObject* pada Unity memiliki fungsi yang berbeda - beda.

2.1.3. *Prefabs*

Prefabs adalah sebuah *GameObject* yang dapat digunakan pada beberapa *scene*. Sebuah *prefabs* juga dapat digunakan berulangkali pada sebuah *scene* yang sama tanpa merubah sifat asli dari prefabs tersebut. Sebuah *prefab* yang dipanggil melalui kode program akan memiliki akhiran *"clone"* pada nama *prefabs*. didalam jendela *Hierarchy*, *prefabs* dilambangkan dengan *GameObject* yang memiliki teks berwarna biru.

2.1.4. *Resources*

Resources merupakan kumpulan *asset game* baik berupa *AudioClip*, *GameObject*, *Image* ataupun animasi. *Resources* dapat dikatakan mirip dengan sebuah *prefab*, yang mana tujuan penggunaannya adalah untuk membuat sebuah template objek sehingga dapat dipanggil dan diduplikasi berulang-ulang. Perbedaan mendasar dari *Resources* dan *prefabs* adalah cara penggunaanya. *Resources* bisa dipanggil secara langsung melalui *code*, sedangkan *prefabs* digunakan langsung melalui Unity Editor. Perbedaan lain adalah *resources* harus ditempatkan di folder *Resources* sedangkan prefabs bisa diletakkan dimana saja.

2.2. C#

C# adalah bahasa pemrograman buatan Microsoft yang dikembangkan untuk bersaing dengan bahasa pemrogaman Java milik Sun. C# adalah sebuah bahasa pemrogaman berbasis objek yang digunakan dengan *web service* pada platform .NET dan dirancang untuk meningkatkan produktivitas pada pengembangan aplikasi jaringan. C# memiliki fitur-fitur seperti *type-safety*, *garbage collection*, *simplified type declarations*, *versioning* dan *scalability support*. [2]

2.3. *Typing Game*

Typing game adalah sebuah jenis *game* yang menggunakan *subgenre* dari *game* edukasi. *Game* edukasi sendiri didefinisikan sebagai sebuah *game* yang didesain untuk mengajarkan hal tertentu. *Typing game* dapat dipahami sebagai sebuah *game* yang didesain untuk mengajarkan kemampuan mengetik.[3]

Gambar 2.2.*Typing game* Learn with Pokemon.

2.4. Mono Framework

Mono adalah sebuah proyek *open source* dan gratis yang disponsori oleh Xamarin. Framework Mono adalah development

framework yang setara dengen framework .NET yang juga adalah milik Microsoft. Karena Mono memiliki Runtime tersendiri yaitu Mono Runtime yang menjalankan program yang memiliki arsitektur prosesor, Unity yang menggunakan Mono dapat mensupport berbagai macam platform. [4]

Mono Framework mensupport berbagai macam bahasa pemrograman, yaitu C#, Javascript dan Boo. C# yang tersedia pada Unity setara dengan yang tersedia pada Visual Studio 2005.

2.5. **Mono Behaviour**

Setiap kelas yang diimplementasikan ke *game object* pada Unity pasti merupakan turunan dari MonoBehaviour, karena MonoBehaviour adalah kelas induk dari *game object* pada Unity. Kelas turunan MonoBehaviour dapat menggunakan fungsi-fungsi seperti Start, Awake, Update, FixedUpdate, serta OnGUI. Pada C# dan Boo MonoBehaviour harus diturunkan secara eksplisit, namun pada Java setiap kelas akan otomatis merupakan turunan dari MonoBehaviour. [5]

Checkbox yang ada pada *editor* akan mengatur berjalannya fungsi yang ada dan tidak akan mempengaruhi berjalannya fungsi dasar lain yang ada pada MonoBehaviour.

2.6. ***Dynamic Difficulty Adjustment***

Dalam sebuah *game* dengan *Dynamic Difficulty Adjustment*, tingkat kesulitan yang dihadapi pemain berubah-ubah selama *game* berlangsung sesuai dengan tingkat kemampuan pemain. Apabila pemain memiliki tingkat kemampuan yang tinggi maka tingkat kesulitan *game* akan terus bertambah, dan akan menurun apabila tingkat kemampuan pemain rendah. Hal ini dapat dicapai dengan berbagai macam cara, terutama karena definisi sukses dapat berubah-ubah untuk tiap *game*.

Keberadaan DDA mengimplikasikan keberadaan cara untuk memantau tingkat kesuksesan pemain dalam *game*. Hal ini dapat meliputi parameter seperti waktu yang diperlukan pemain untuk

menyelesaikan objektif *game*, jumlah nyawa karakter yang hilang, dan lain sebagainya. [6]

2.7. *Fuzzy*

Logika *fuzzy* adalah pendekatan komputasi menggunakan dasar "derajat kebenaran" di mana pendekatan komputasi pada umumnya menggunakan dasar "benar atau salah". [7]

Kata "*fuzzy"* mengacu pada fakta bahwa logika ini dapat mengatasi konsep-konsep yang tidak dapat dinyatakan sebagai "benar" atau "salah" tetapi sebagai "sebagian benar". Pendekatan lain seperti *genetic algorithm* dan *neural network* juga dapat digunakan seperti *fuzzy*, logika *fuzzy* memiliki keunggulan di mana persoalan yang akan diselesaikan menggunakan *fuzzy* dapat diekspresikan menggunakan frase yang dapat dipahami oleh pengguna manusia dengan mudah. "Bila suhu rendah, maka naikkan suhu pemanas ruangan".

Penerapan utama logika *fuzzy* adalah dalam bidang *control system.* Sebuah *fuzzy* controller terdiri dari tiga prosess utama, *fuzzyfication*, *rule-based inference*, dan *defuzzyfication*. Proses pertama adalah *fuzzyfication*. Pada proses ini, nilai diskrit *input* dipetakan ke sekumpulan nilai fuzzy yang disebut *fuzzy sets* menggunakan sekumpulan fungsi matematika yang disebut sebagai *membership function*. Proses kedua adalah *rule-based inference*. Proses ini adalah proses untuk menentukan respons *fuzzy* berdasarkan aturan linguistic yang sudah ditentukan sebelumnya. Proses ketiga adalah *defuzzyfication.* Proses ini menentukan nilai *output* yang diskrit dari respons *fuzzy*.

2.7.1. *Fuzzyfication*

Setiap variable *input* dan *output* yang ada memerlukan dua atau lebih *membership function*, umumnya tiga namun bisa lebih apabila diperlukan. Tiap-tiap *membership function* perlu didefinisikan menjadi kategori kualitatif semisal, rendah, normal, atau tinggi. Bentuk dari *membership function* ini bisa berbeda-beda tergantung fungsi yang digunakan.

Membership function yang sederhana dapat dibentuk menggunakan garis lurus. Salah satu contohnya adalah *triangular membership function* yang biasa disebut trimf. Trimf merupakan koleksi tiga poin yang membentuk segitiga. Contoh lain adalah *trapezoidal membership function* yang biasa disebut trapmf. Membership function yang berbentuk garis lurus ini memiliki keunggulan di bidang simplisitas. Persamaan untuk trimf dan trapmf dapat dilihat pada persamaan (1) dan (2).

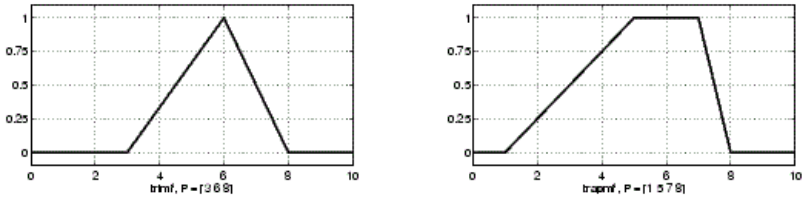
$$triangle(x;a,b,c) = \begin{cases} 0, & x \le a \\ \dfrac{x-a}{b-a}, & a \le x \le b \\ \dfrac{c-x}{c-b}, & b \le x \le c \\ 0, & c \le x \end{cases} \quad \textbf{(1)}$$

$$trapezoid(x;a,b,c,d) = \begin{cases} 0, & x \le a \\ \dfrac{x-a}{b-a}, & a \le x \le b \\ 1, & b \le x \le c \\ \dfrac{d-x}{d-c}, & c \le x \le d \\ 0, & d \le x \end{cases} \quad \textbf{(2)}$$

Persamaan trimf dan trapmf juga dapat dinyatakan menggunakan fungsi *min* dan *max* seperti terlihat pada persamaan (3) dan (4).

$$triangle(x;a,b,c) = \max(\min(\frac{x-a}{b-a}, \frac{c-x}{c-b}),0) \quad \textbf{(3)}$$

$$trapezoid(x;a,b,c,d) = \max(\min(\frac{x-a}{b-a}, 1, \frac{d-x}{d-c}),0) \quad \textbf{(4)}$$



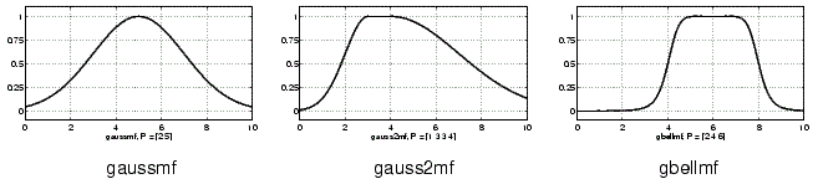
trimf trapmf

Gambar 2.3. Bentuk trimf dan trapmf.

Membership function juga dapat dibentuk menggunakan kurva distribusi G*aussian*: kurva *Gaussian* sederhana dan komposit dua kurva *Gaussian* yang berbeda. Kedua fungsi tersebut biasa disebut gaussmf dan gauss2mf. Ada juga *generalized bell membership function* yang memiliki tiga parameter dan biasa disebut gbellmf. Gbellmf memiliki lebih banyak parameter dari gaussmf dan gauss2mf sehingga memiliki tingkat kebebasan yang lebih tinggi dalam mengatur kecuraman bentuk fungsi. Karena bentuknya yang halus dan notasi yang ringkas, *membership function Gaussian* dan bell merupakan metode *fuzzyfication* yang popular. Kedua fungsi ini memiliki kelebihan dalam bentuknya yang halus dan nilai yang tidak pernah nol. Perumusan gaussmf dan gbellmf dapat dilihat pada persamaan (5) dan (6).

$$gaussian(x; c, \sigma) = e^{-\frac{1}{2}(\frac{x-c}{\sigma})^2} \quad \textbf{(5)}$$

$$gbell(x; a, b, c) = \frac{1}{1 + \left|\frac{x-c}{a}\right|^{2b}} \quad \textbf{(6)}$$



Gambar 2.4. Bentuk gaussmf, gauss2mf, dan gbellmf.

2.7.2. *Rule-Based Inference*

Setelah *membership function* selesai didefinisikan, langkah berikutnya adalah membuat kumpulan aturan *decision matrix*. Aturan ini akan mengubah variabel *input* menjadi *output*. Salah satu contoh aturan yang sederhana adalah aturan yang hanya melingkupi satu variabel *input* dan satu variabel *output* seperti "Bila suhu ruangan rendah maka suhu pemanas tinggi". Semakin banyak variabel yang ada, maka semakin banyak pula aturan yang harus dibuat. Bergantung pada jumlah variabel yang ada, aturan dapat melingkupi beberapa variabel input dan output, seperti "Bila suhu ruangan tinggi dan udara lembap maka suhu pemanas rendah dan kecepatan kipas angin tinggi". Apabila jumlah variabel input lebih dari satu, maka akan digunakan operan AND atau OR pada variabel input sesuai dengan aturan yang ada. Pada contoh aturan di atas, maka akan digunakan operan AND.

2.7.3. *Defuzzyfication*

Defuzzyfication adalah proses untuk mengubah nilai output yang masih berupa *fuzzy* menjadi nilai yang diskrit. Defuzzyfication dapat dilakukan melalui berbagai macam cara. Salah satu caranya adalah menggunakan metode *weighted average*.

Weighted average adalah metode menghitung rata-rata di mana tiap nilai dalam data set memiliki bobot yang berbeda dan bobot ini mempengaruhi tingkat kepentingan relative nilai tersebut. Dalam *defuzzyfication*, bobot ini ditentukan oleh nilai fuzzy set. Persamaan weighted average dapat dilihat pada persamaan (7).

$$z = \frac{\sum \mu(x)_i \times W_i}{\sum \mu(x)_i} \tag{7}$$

BAB III
ANALISIS DAN PERANCANGAN SISTEM

Bab ini membahas tahap analisis permasalahan dan perancangan dari sistem yang akan dibangun. Analisis permasalahan membahas permasalahan yang diangkat dalam pengerjaan tugas akhir. Selanjutnya dibahas mengenai perancangan sistem yang dibuat. Pendekatan yang dibuat dalam perancangan ini adalah pendekatan berorientasi objek. Perancangan direpresentasikan dengan diagram UML (*Unified Modelling Language*).

3.1. **Analisis**

Tahap analisis dibagi menjadi beberapa bagian antara lain cakupan permasalahan dan deskripsi umum sistem.

3.1.1. Analisis Permasalahan

Permasalahan utama yang diangkat dalam pengerjaan tugas akhir ini adalah bagaimana membuat implementasi aturan main yang dapat dipakai oleh *game Word Master*. Permasalahan kedua adalah bagaimana cara merancang *dynamic difficulty adjustment*.

3.1.2. Deskripsi Umum

Sistem yang akan dibuat berupa *game* yang mengimplementasikan *dynamic difficulty adjustment*. *Dynamic difficulty adjustment* akan diimplementasikan menggunakan logika *fuzzy*.

3.1.3. Analisis *Game*

Kelas-kelas yang digunakan oleh *game Word Master* dapat dibagi menjadi dua kelompok, kelompok kelas yang termasuk dalam bagian *MainGameplay*, dan kelas yang mengatur implementasi *dynamic difficulty adjustment*.

Kelompok kelas yang termasuk dalam bagian *MainGameplay*, adalah kelas-kelas yang mengimplementasikan fungsi fungsi yang diperlukan untuk mengimplementasikan aturan main dari *game Word Master*. Kelas yang mengatur implementasi

dynamic difficulty adjustment berisikan implementasi algoritma yang digunakan untuk penerapan konsep *dinamyc difficulty adjustment*. Dalam kasus ini, algoritma yang digunakan adalah algoritma *fuzzy*.

3.1.3.1. Aturan Main

Aturan main *typing game Word Master* adalah sebagai berikut.

1. Karakter pemain tidak dapat bergerak, memiliki sejumlah *live* dan berada di satu sisi layar.
2. Karakter musuh muncul dari sisi sebaliknya dan bergerak menuju karakter pemain.
3. Setiap kali karakter musuh muncul, muncul kata baru yang dipilih secara acak.
4. Apabila pemain berhasil memasukkan kata tersebut dengan benar, yaitu dengan mengetikkan kata tersebut lalu menekan tombol *enter* pada *keyboard*, maka karakter musuh akan hancur.
5. Apabila pemain memasukkan kata yang salah, maka kata yang baru akan dipilih dan pemain harus menunggu sejumlah waktu sebelum dapat kembali memasukkan kata.
6. Apabila karakter musuh menyentuh karakter pemain, maka karakter musuh akan hancur dan *live* pemain berkurang.
7. Apabila karakter musuh hancur maka akan muncul karakter musuh yang baru.
8. Apabila *live* pemain habis, maka permainan berakhir.

3.1.3.2. Skenario

3.1.3.2.1. *Style* dan *Tone*

Word Master menggunakan style gambar kartun dan tema fantasi.

Gambar 3.1. Contoh gambar dalam permainan *Word Master*.

3.1.3.2.2. Karakter

Karakter dalam *game* ini adalah karakter pemain dan karakter musuh. Karakter pemain berupa karakter humanoid dan karakter musuh berupa monster.

Gambar 3.2. Contoh gambar karakter permainan *Word Master*.

3.1.4. Fungsional Sistem

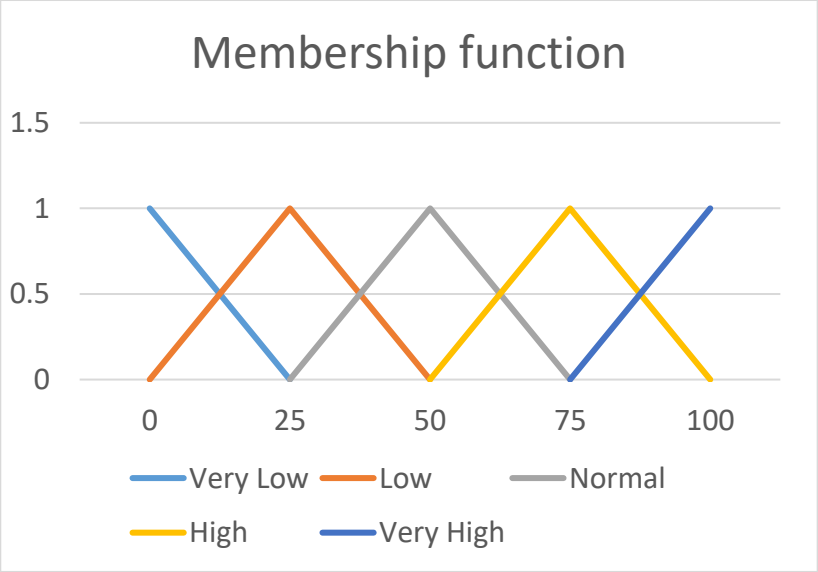
Fungsi-fungsi yang digunakan untuk implementasi DDA dapat dilihat pada Tabel 3.1.

Tabel 3.1. Daftar fungsi pada sistem

No	Fungsi
1	Penghitung nilai *fuzzy set*
2	Penghitung nilai *output*
3	Pemilih kata secara acak sesuai tingkat kesulitan

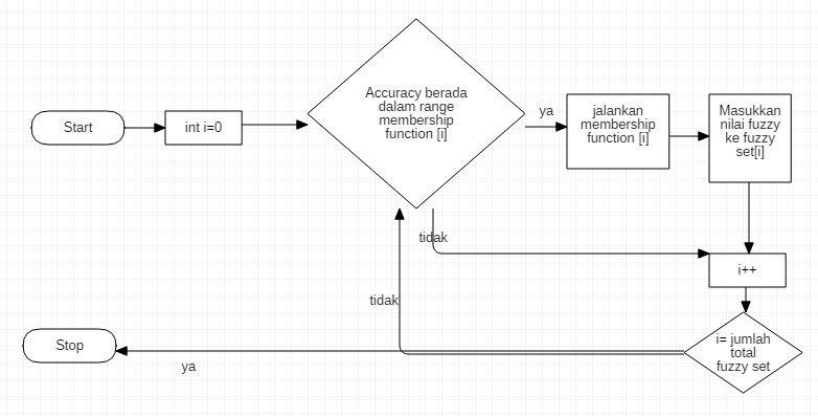
3.1.4.1. Penghitung Nilai *Fuzzy Set*

Parameter performa pemain yang berupa nilai scalar akan diubah menjadi nilai *fuzzy* yang disebut sebagai proses *fuzzyfication*. Nilai fuzzy ini berikutnya akan digunakan untuk dalam fungsi penghitung *nilai output*. Proses ini akan diimplementasikan menggunakan *triangular membership function* seperti terlihat pada gambar 3.3.



Gambar 3.3. Kurva membership function.

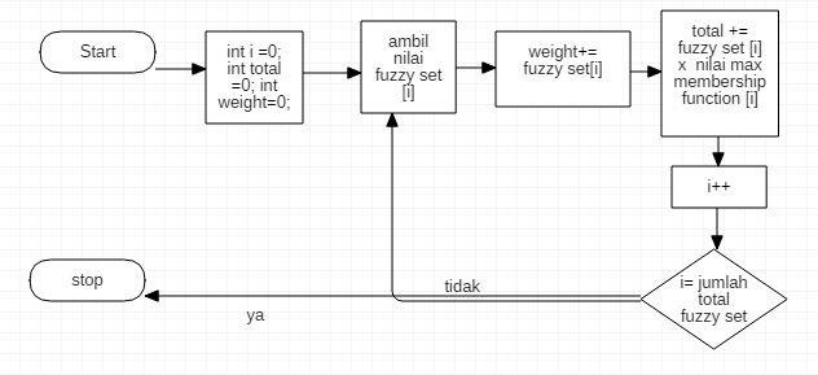
Diagram alir untuk fungsi penghitung nilai *fuzzy set* dapat dilihat pada Gambar 3.4.



Gambar 3.4. Diagram alir fungsi penghitung nilai *fuzzy set*.

3.1.4.2. Penghitung Nilai *Output*

Nilai-nilai *fuzzy* dalam *fuzzy set* harus dikembalikan menjadi nilai skalar supaya memiliki arti yang dikenal sebagai proses *defuzzyfication*. Proses ini akan diimplementasikan menggunakan metode *weighted average*. Diagram alir untuk fungsi penghitung nilai output dapat dilihat pada Gambar 3.5.

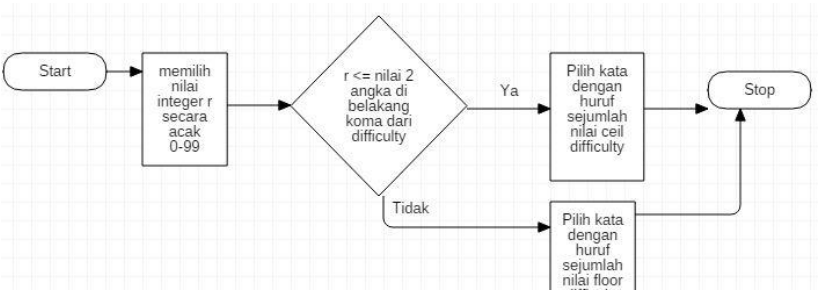


Gambar 3.5. Diagram alir fungsi penghitung nilai output.

3.1.4.3. Pemilih Kata

Proses fuzzification dan defuzzyfication seperti disebutkan di atas digunakan untuk menghitung perubahan tingkat kesulitan sesuai dengan performa pemain. Perubahan ini direfleksikan melalui jumlah huruf pada kata yang dipilih.

Kata dipilih secara acak dari kumpulan kata yang diambil dari permainan *scrabble*. Diagram alir pemilih kata dapat dilihat pada Gambar 3.3.



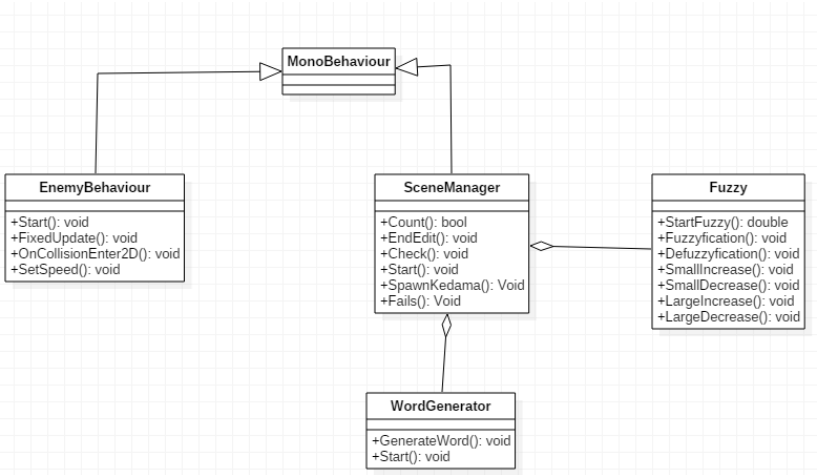
Gambar 3.6. Diagram alir pemilih kata.

3.2. Perancangan Sistem

Penjelasan tahap perancangan perangkat lunak dibagi menjadi beberapa bagian yaitu perancangan diagram kelas, perancangan proses analisis, dan perancangan antarmuka.

3.2.1. Perancangan Diagram Kelas

Perancangan diagram kelas berisi rancangan dari kelas-kelas yang digunakan untuk membangun *game*. Pada subbab ini, hubungan dan perilaku antar kelas digambarkan dengan lebih jelas.



Gambar 3.7. Desain diagram kelas seluruh system.

3.2.1.1. Kelompok kelas bagian MainGameplay

Kelompok kelas bagian MainGameplay mengandung kelas-kelas yang berhubungan dengan kelas-kelas utama *game* dan kelas-kelas yang mengatur logika *game*. Terdapat kelas SceneManager yang mengatur segala hal yang dilihat oleh pemain di permukaan meliputi membuat karakter musuh, perubahan UI, dan penerimaan masukan dari pemain. Kelas WordGenerator adalah kelas yang mengatur pemilihan kata yang yang harus diketikkan pemain. Kelas EnemyBehavior adalah kelas yang mengatur perilaku musuh seperti kecepatan gerak musuh..

3.2.1.2. Kelompok kelas bagian Algoritma

Kelompok kelas bagian Algoritma adalah kelompok kelas yang mengatur penerapan konsep DDA. Kelompok kelas terdiri dari kelas Fuzzy yang mengatur implementasi algoritma *fuzzy* sebagai penerapan konsep DDA.

BAB IV
IMPLEMENTASI

Bab ini membahas tentang implementasi dari perancangan *game Word Master*. Proses implementasi dari setiap kelas yang dibuat dengan mengacu pada rancangan yang telah dibahas sebelumnya akan dipaparkan pada bab ini. Bahasa pemrograman yang digunakan adalah bahasa pemrograman C#.

4.1. Kelas SceneManager.cs

Kelas SceneManager.cs adalah kelas yang mengatur implementasi aturan main *game* dan hal-hal lain yang dilihat oleh pemain, seperti memunculkan karakter, menerima masukan pemain, mengecek apakah masukan pemain benar atau salah, memutar *sound effect*, dan mengatur UI. Pada saat *game* dimulai, kelas akan memanggil fungsi Start untuk menginisialisasi hal-hal yang akan diperlukan dalam *game* seperti, memuat *sound effect* yang akan diputar, memuat *game object* yang akan digunakan,memilih kata pertama yang harus diketikkan pemain, dan memunculkan musuh melalui fungsi SpawnKedama. Musuh akan bergerak menuju karakter pemain. Pemain kemudian mengetikkan kata yang diminta kemudian menekan tombol *Enter* pada *keyboard* untuk mengkonfirmasi bahwa pemain sudah selesai mengetikkan kata. Hal ini akan mengaktifkan fungsi Endedit yang akan memanggil fungsi Check. Fungsi Check akan memeriksa apakah pemain mengetikkan kata yang benar atau tidak. Apabila benar, maka fungsi akan menghancurkan musuh, memanggil fungsi count dengan parameter *true*, lalu memunculkan musuh yang baru dan memilih kata yang baru. Apabila salah, maka fungsi akan memanggil fungsi count dengan parameter *false*. Apabila pemain tidak sempat mengetikkan kata yang diminta, maka kelas akan memanggil fungsi Fail. Fungsi Fail akan memanggil fungsi count dengan parameter *false*, lalu menghancurkan musuh dan memunculkan musuh dan kata baru. Fungsi Count adalah fungsi yang digunakan untuk menghitung jumlah kata yang diketikkan pemain, jumlah kata yang diketikkan pemain dengan benar, dan jumlah giliran

yang berlangsung. Fungsi Count akan menjalankan algoritma *fuzzy* setiap 5 giliran.

```
namespace Assets
{
    public class SceneManager : MonoBehaviour
    {
        public Text text;
        public Text level;
        public Text result;
        public InputField input;
        public GameObject kedama;
        AudioSource sound;
        public AudioClip clip;
        GameObject clone;
        Fuzzy f = new Fuzzy();
        WordGenerator wordGenerator = new
WordGenerator();
        double accuracy;
        double milisecs;
        int turncount = 0;
        int rightCount = 0;
        int typeCount = 0;
        Stopwatch sw = new Stopwatch();
        bool b;
        double difficulty = 3;
        // Use this for initialization
        void Start()
        {

sound=gameObject.AddComponent<AudioSource>();
            input.onEndEdit.AddListener(delegate {
Endedit(); });

EventSystem.current.SetSelectedGameObject(input.gameObje
ct, null);
            wordGenerator.Start();
            text.text = wordGenerator.GenerateWord(3);
            input.ActivateInputField();
            result.text = "";
```

```
            level.text = "3";
            sw.Reset();
            sw.Start();
            SpawnKedama();
            b = false;
            sound.clip =
Resources.Load<AudioClip>("smw_kick");
        }
        void SpawnKedama()
        {
            clone =
Instantiate(kedama,kedama.transform.position,kedama.tran
sform.rotation);

clone.gameObject.AddComponent<EnemyBehaviour>();
            clone.SendMessage("Start");
        }
        void Endedit()
        {
            StartCoroutine(Check());
        }
        IEnumerator Check()
        {
            sw.Stop();
            b =
text.text.ToLower().Equals(input.text.ToLower());
            sound.PlayOneShot(clip);
            if (b) { DestroyImmediate(clone, true);
rightCount++;turncount++; }
            result.text = b.ToString();
            Count();
            if(input.interactable == true)
input.interactable = false;
            yield return new WaitForSeconds(1);
            input.interactable = true;
            input.text = "";
            result.text = "";

EventSystem.current.SetSelectedGameObject(input.gameObje
ct, null);
```

```
            input.ActivateInputField();

text.text=wordGenerator.GenerateWord(difficulty);
            if(b) SpawnKedama();
            sw.Reset();
            sw.Start();
            b = false;

        }
        IEnumerator Fails()
        {
            sw.Stop();
            sound.PlayOneShot(clip);
            Destroy(clone);
            result.text = "Fail";
            turncount++;
            Count();
            input.interactable = false;

UnityEngine.Debug.Log(sw.ElapsedMilliseconds);
            yield return new WaitForSeconds(1);
            input.interactable = true;
            input.text = "";
            result.text = "";
            turncount++;

EventSystem.current.SetSelectedGameObject(input.gameObje
ct, null);
            input.ActivateInputField();
            text.text =
wordGenerator.GenerateWord(difficulty);
            SpawnKedama();
            sw.Reset();
            sw.Start();
        }
        public void Count()
        {
            typeCount++;
            if (turncount==5)
            {
```

```
                UnityEngine.Debug.Log(turncount);
                accuracy = (float)rightCount /
typeCount;
                //Console.WriteLine(acc);
                rightCount = 0;
                turncount = 0;
                typeCount = 0;
                UnityEngine.Debug.Log(turncount);
                difficulty +=f.StartFuzzy(accuracy);
                if (difficulty < 3) difficulty = 3;
                if (difficulty > 8) difficulty = 8;
                level.text = difficulty.ToString();
            }

        }
    }
}
```

Kode Sumber 1. Fungsi-fungsi kelas SceneManager.cs.

4.2. Kelas WordGenerator.cs

Kelas WordGenerator adalah kelas yang digunakan untuk memilih kata yang harus diketikkan pemain secara acak. Fungsi Start digunakan untuk memindai kata-kata dari *file*. Fungsi GenerateWord digunakan untuk memilih kata yang harus diketikkan pemain secara acak di mana jumlah huruf dari kata yang dipilih disesuaikan dengan tingkat kesulitan yang dihadapi pemain.

```
namespace Assets
{
    public class WordGenerator
    {

        public GameObject target;
        string[] letter3;
        string[] letter4;
        string[] letter5;
        string[] letter6;
        string[] letter7;
```

```
        string[] letter8;
        string s;
        float level;
        List<string[]> list = new List<string[]>();
        System.Random rand = new System.Random();
        // Use this for initialization
        public void Start()
        {
            letter3 =
System.IO.File.ReadAllLines(@"E:\3Letter.txt");
            letter4 =
System.IO.File.ReadAllLines(@"E:\4Letter.txt");
            letter5 =
System.IO.File.ReadAllLines(@"E:\5Letter.txt");
            letter6 =
System.IO.File.ReadAllLines(@"E:\6Letter.txt");
            letter7 =
System.IO.File.ReadAllLines(@"E:\7Letter.txt");
            letter8 =
System.IO.File.ReadAllLines(@"E:\8Letter.txt");
            list.Add(letter3);
            list.Add(letter4);
            list.Add(letter5);
            list.Add(letter6);
            list.Add(letter7);
            list.Add(letter8);
        }

        public string GenerateWord(double difficulty)
        {
            int count = list.Count;
            int diffCeil;
            int diffFloor;
            diffCeil =
(int)Mathf.Ceil((float)difficulty);
            diffFloor =
(int)Mathf.Floor((float)difficulty);
            string[] temp;
            int r = rand.Next(0, 100);
            float r2 = (float)r / 100;
```

```
            float difference = (float)difficulty -
diffFloor;
            if (r2 <= (float)(difference)) { temp =
list[diffCeil-3]; }
            else { temp = list[diffFloor-3]; }
            int count2 = temp.Count();
            s = temp[rand.Next(0, count2)];
            return s;
        }
    }
}
```

Kode Sumber 2. Fungsi-fungsi kelas WordGenerator.cs.

4.3. Kelas EnemyBehavior.cs

Kelas ini adalah kelas yang digunakan untuk mengatur perilaku musuh seperti arah pergerakan musuh, dan kecepatan gerak musuh.

```
public class EnemyBehaviour : MonoBehaviour {

    public float speed ;
    public GameObject SceneManager;
    public GameObject player;
    Vector2 position ;
    Vector2 playerPosition;
    // Use this for initialization
    void Start () {
        SceneManager = GameObject.Find("SceneManager");
        player = GameObject.Find("Player");
        playerPosition = player.transform.position;
        position = this.transform.position;
        setSpeed();
    }

    void FixedUpdate ()
    {
        position.x+=speed;
        transform.position = position;
```

```
    }
    private void OnCollisionEnter2D(Collision2D
collision)
    {
        SceneManager.SendMessage("Fails");
        Destroy(gameObject);
    }
    void setSpeed()
    {
        playerPosition = player.transform.position;
        position = this.transform.position;
        speed = (float)(playerPosition.x - position.x) /
260;
    }
}
```

Kode Sumber 3. Fungsi-fungsi kelas EnemyBehavior.cs.

4.4. Kelas Fuzzy.cs

Kelas ini adalah kelas yang digunakan untuk mengimplementasikan algoritma *fuzzy* sebagai penerapan konsep DDA. Fungsi Fuzzyfication adalah fungsi yang digunakan untuk menetapkan nilai masing-masing *membership function* dari *fuzzy*. Fungsi Defuzzyfication adalah fungsi yang digunakan untuk mengubah nilai *membership function* menjadi nilai *output* yang diskrit.

```
namespace Assets
{
    public class Fuzzy
    {
        double accuracy;
        double milisecs;
        double[,] accuracymatrixrule = new double[5, 3]
{ { 0,0,0.25},{0,0.25,0.50},{0.25,0.50,0.75
},{0.50,0.75,1.00 },{0.75,1.0,1.00 } };
        double[] accuracymatrix = new double[5];
        double result = 0;
```

```
        public double StartFuzzy(double accu)
        {
            accuracy = accu;
            result = 0;
            Fuzzyfication();
            return result;
        }
        private void Fuzzyfication()
        {
            for(int i=0;i<5;i++)
            {
                double b = 0;
                if (accuracy>=accuracymatrixrule[i,0] &&
accuracy < accuracymatrixrule[i,1])
                { b = (accuracy - accuracymatrixrule[i,
0])/(accuracymatrixrule[i,1]-accuracymatrixrule[i,0]);}
                if (accuracy> accuracymatrixrule[i, 1]
&& accuracy <= accuracymatrixrule[i, 2])
                { b = (accuracymatrixrule[i,2] -
accuracy) / (accuracymatrixrule[i, 2] -
accuracymatrixrule[i, 1]);}
                if (accuracy == accuracymatrixrule[i,
1]) { b = 1;}
                accuracymatrix[i] = b;
            }
            Defuzzyfication();
        }
        private void SmallIncrease(double accu)
        {
            result += accu*0.25;
        }
        private void LargeIncrease(double accu)
        {
            result += accu * 0.5;
        }
        private void SmallDecrease(double accu)
        {
            result += accu * -0.25;
        }
        private void LargeDecrease(double accu)
```

```
        {
            result += accu * -0.5;
        }
        private void Defuzzyfication()
        {
            for (int i = 0; i < 5; i++)
            {
                switch (i)
                {
                    case 0:
LargeDecrease(accuracymatrix[i]); break;
                    case 1:
SmallDecrease(accuracymatrix[i]); break;
                    case 2: break;
                    case 3:
SmallIncrease(accuracymatrix[i]); break;
                    case 4:
LargeIncrease(accuracymatrix[i]); break;
                }
            }
        }
    }
}
```

Kode Sumber 4. Fungsi-fungsi kelas Fuzzy.cs.

BAB V
PENGUJIAN DAN EVALUASI

Bab ini membahas pengujian dan evaluasi pada *game* yang dikembangkan. Kelas-kelas yang akan diuji adalah kelas pada kelompok HMCL. Pengujian fungsionalitas mengacu pada kasus penggunaan pada bab tiga. Pengujian yang dilakukan adalah pengujian terhadap kebutuhan fungsionalitas sistem dengan menggunakan metode *black box*. Hasil evaluasi menjabarkan tentang rangkuman hasil pengujian pada bagian akhir bab ini.

5.1. Lingkungan Pengujian

Lingkungan pengujian sistem pada pengerjaan tugas akhir ini dilakukan pada lingkungan dan alat kakas sebagai berikut:

Jenis Device : ASUS X455L
Prosesor : Intel i3-4030U Processor
Memori : 6GB RAM
Kartu Grafis : Intel HD Graphic
Sistem Operasi : Windows 10

5.2. Skenario Pengujian

Pada bagian ini akan dijelaskan tentang skenario pengujian yang dilakukan. Pengujian yang dilakukan adalah pengujian kebutuhan fungsionalitas dengan menggunakan metode *black box testing*.

5.2.1. Pengujian Fungsionalitas

Pengujian kebutuhan fungsionalitas sistem dilakukan dengan menyiapkan sejumlah skenario pengujian sebagai tolak ukur keberhasilan pengujian. Subbab 3.1.4. yang telah menjabarkan fungsional sistem dijadikan acuan untuk pengujian ini. Pengujian pada kebutuhan fungsionalitas dijabarkan pada subbab berikut.

5.2.1.1. Pengujian Fitur Penghitung Nilai *Fuzzy Set*

Pengujian fitur penghitung nilai *fuzzy set* bertujuan menguji apakah nilai *fuzzy set* yang dihasilkan sesuai dengan yang diharapkan. Skenario pengujian adalah dengan cara memasukkan kombinasi *input* yang menghasilkan nilai akurasi tertentu pada giliran kelima. Pada tahap ini sistem akan memunculkan *log* yang berisikan tingkat akurasi dan nilai *fuzzy set* . Hasil pengujian fitur ini dapat dilihat pada Tabel 5.1, Tabel 5.2, Tabel 5.3, Tabel 5.4, Tabel 5.25 Tabel 5.6, Tabel 5.7.

Tabel 5.1. Pengujian fitur penghitung nilai *fuzzy set.*

ID	**UJ.WM-001**
Referensi Fungsionalitas	1
Nama	Pengujian Fitur Penghitung Nilai *Fuzzy Set*.
Tujuan Pengujian	Menguji fitur penghitung nilai *fuzzy set* beserta hasilnya.
Skenario 1	Pengguna memiliki akurasi 100%.
Kondisi Awal	Aplikasi tidak menunjukkan log seperti ditunjukkan pada Gambar 5.1.
Masukan	Pemain mengetikkan 5 kata dengan benar berturut-turut.
Hasil yang Diharapkan	Nilai *fuzzy set* nomor 0,1,2, dan 3 = 0. Nilai *fuzzy set* nomor 4 = 1.
Hasil yang Didapat	Nilai *fuzzy set* nomor 0,1,2, dan 3 = 0. Nilai *fuzzy set* nomor 4 = 1..
Hasil Pengujian	Berhasil
Kondisi Akhir	Tampilan hasil pengujian dapat dilihat pada Gambar 5.2.

Tabel 5.2. Pengujian fitur penghitung nilai *fuzzy set.*

ID	**UJ.WM-002**
Referensi Fungsionalitas	1
Nama	Pengujian Fitur Penghitung Nilai *Fuzzy Set*.
Tujuan Pengujian	Menguji fitur penghitung nilai *fuzzy set* beserta hasilnya.
Skenario 1	Pengguna memiliki akurasi 83,33%.
Kondisi Awal	Aplikasi tidak menunjukkan log seperti ditunjukkan pada Gambar 5.1.
Masukan	Pemain mengetikkan 1 kata salah lalu 5 kata dengan benar berturut-turut.
Hasil yang Diharapkan	Nilai *fuzzy set* nomor 0,1, dan 2 =0. Nilai *fuzzy set* nomor 3 = 0,67. Nilai *fuzzy set* nomor 4 = 0,33.
Hasil yang Didapat	Nilai *fuzzy set* nomor 0,1, dan 2 =0. Nilai *fuzzy set* nomor 3 = 0,67. Nilai *fuzzy set* nomor 4 = 0,33.
Hasil Pengujian	Berhasil
Kondisi Akhir	Tampilan hasil pengujian dapat dilihat pada Gambar 5.3.

Tabel 5.3. Pengujian fitur penghitung nilai *fuzzy set.*

ID	**UJ.WM-003**
Referensi Fungsionalitas	1
Nama	Pengujian Fitur Penghitung Nilai *Fuzzy Set*.
Tujuan Pengujian	Menguji fitur penghitung nilai *fuzzy set* beserta hasilnya.

Skenario 1	Pengguna memiliki akurasi 80%.
Kondisi Awal	Aplikasi tidak menunjukkan log seperti ditunjukkan pada Gambar 5.1.
Masukan	Pemain gagal mengetikkan 1 kata lalu mengetikkan 4 kata dengan benar berturut-turut.
Hasil yang Diharapkan	Nilai *fuzzy set* nomor 0,1, dan 2 = 0. Nilai *fuzzy set* nomor 3 = 0,80. Nilai *fuzzy set* nomor 4 = 0,2.
Hasil yang Didapat	Nilai *fuzzy set* nomor 0,1, dan 2 = 0. Nilai *fuzzy set* nomor 3 = 0,80. Nilai *fuzzy set* nomor 4 = 0,2.
Hasil Pengujian	Berhasil
Kondisi Akhir	Tampilan hasil pengujian dapat dilihat pada Gambar 5.4.

Tabel 5.4. Pengujian fitur penghitung nilai *fuzzy set.*

ID	**UJ.WM-004**
Referensi Fungsionalitas	1
Nama	Pengujian Fitur Penghitung Nilai *Fuzzy Set*.
Tujuan Pengujian	Menguji fitur penghitung nilai *fuzzy set* beserta hasilnya.
Skenario 1	Pengguna memiliki akurasi 60%.
Kondisi Awal	Aplikasi tidak menunjukkan log seperti ditunjukkan pada Gambar 5.1.
Masukan	Pemain gagal mengetikkan 2 kata lalu mengetikkan 3 kata dengan benar berturut-turut.
Hasil yang Diharapkan	Nilai *fuzzy set* nomor 0,1,dan 4 = 0 Nilai *fuzzy set* nomor 2 = 0.6. Nilai fuzzy set nomor 3 = 0,4.

Hasil yang Didapat	Nilai *fuzzy set* nomor 0,1,dan 4 = 0 Nilai *fuzzy set* nomor 2 = 0.6. Nilai fuzzy set nomor 3 = 0,4.
Hasil Pengujian	Berhasil
Kondisi Akhir	Tampilan hasil pengujian dapat dilihat pada Gambar 5.4.

Tabel 5.5. Pengujian fitur penghitung nilai *fuzzy set.*

ID	**UJ.WM-005**
Referensi Fungsionalitas	1
Nama	Pengujian Fitur Penghitung Nilai *Fuzzy Set*.
Tujuan Pengujian	Menguji fitur penghitung nilai *fuzzy set* beserta hasilnya.
Skenario 1	Pengguna memiliki akurasi 40%.
Kondisi Awal	Aplikasi tidak menunjukkan log seperti ditunjukkan pada Gambar 5.1.
Masukan	Pemain gagal mengetikkan 3 kata lalu mengetikkan 2 kata dengan benar berturut-turut.
Hasil yang Diharapkan	Nilai *fuzzy set* nomor 0,3,dan 4 = 0 Nilai *fuzzy set* nomor 1 = 0.4. Nilai fuzzy set nomor 2 = 0,6.
Hasil yang Didapat	Nilai *fuzzy set* nomor 0,3,dan 4 = 0 Nilai *fuzzy set* nomor 1 = 0.4. Nilai fuzzy set nomor 2 = 0,6.
Hasil Pengujian	Berhasil
Kondisi Akhir	Tampilan hasil pengujian dapat dilihat pada Gambar 5.6.

Tabel 5.6. Pengujian fitur penghitung nilai *fuzzy set.*

ID	**UJ.WM-006**
Referensi Fungsionalitas	1
Nama	Pengujian Fitur Penghitung Nilai *Fuzzy Set*.
Tujuan Pengujian	Menguji fitur penghitung nilai *fuzzy set* beserta hasilnya.
Skenario 1	Pengguna memiliki akurasi 20%.
Kondisi Awal	Aplikasi tidak menunjukkan log seperti ditunjukkan pada Gambar 5.1.
Masukan	Pemain gagal mengetikkan 4 kata lalu mengetikkan 1 kata dengan benar berturut-turut.
Hasil yang Diharapkan	Nilai *fuzzy set* nomor 2,3,dan 4 = 0 Nilai *fuzzy set* nomor 0 = 0.2. Nilai fuzzy set nomor 1 = 0,8.
Hasil yang Didapat	Nilai *fuzzy set* nomor 2,3,dan 4 = 0 Nilai *fuzzy set* nomor 0 = 0.2. Nilai fuzzy set nomor 1 = 0,8.
Hasil Pengujian	Berhasil
Kondisi Akhir	Tampilan hasil pengujian dapat dilihat pada Gambar 5.7.

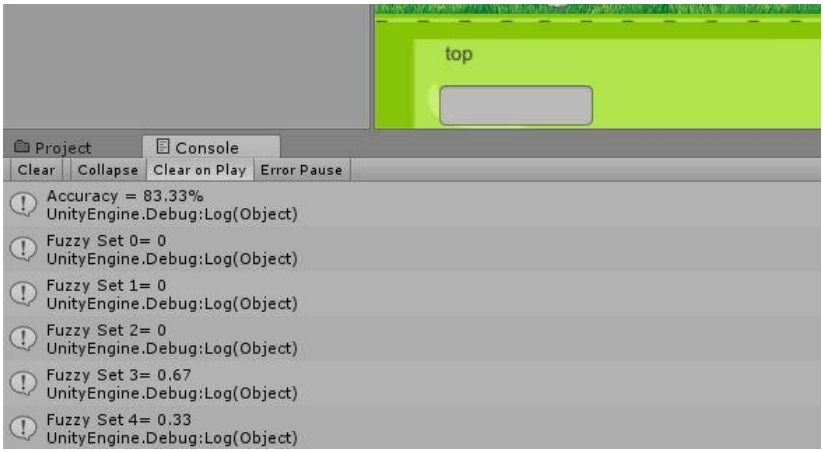
Tabel 5.7. Pengujian fitur penghitung nilai *fuzzy set.*

ID	**UJ.WM-007**
Referensi Fungsionalitas	1
Nama	Pengujian Fitur Penghitung Nilai *Fuzzy Set*.
Tujuan Pengujian	Menguji fitur penghitung nilai *fuzzy set* beserta hasilnya.

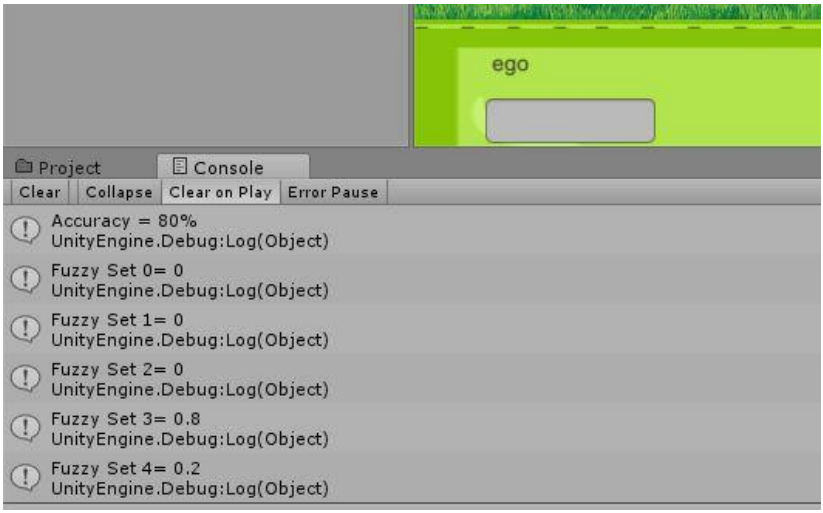
Skenario 1	Pengguna memiliki akurasi 0%.
Kondisi Awal	Aplikasi tidak menunjukkan log seperti ditunjukkan pada Gambar 5.1.
Masukan	Pemain gagal mengetikkan 5 kata berturut-turut.
Hasil yang Diharapkan	Nilai *fuzzy set* nomor 1,2,3,dan 4 = 0 Nilai *fuzzy set* nomor 0 = 1.
Hasil yang Didapat	Nilai *fuzzy set* nomor 1,2,3,dan 4 = 0 Nilai *fuzzy set* nomor 0 = 1.
Hasil Pengujian	Berhasil
Kondisi Akhir	Tampilan hasil pengujian dapat dilihat pada Gambar 5.8.

Gambar 5.1. Kondisi awal ketika pemain baru memulai.

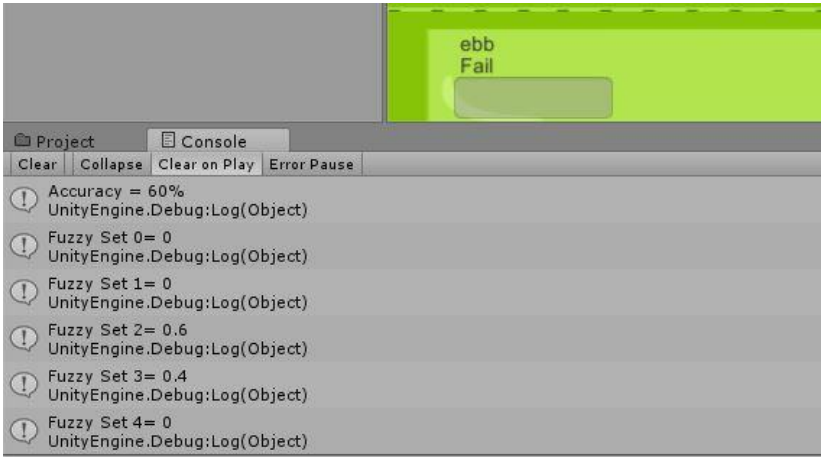
Gambar 5.2. Kondisi ketika pemain mencapai akurasi 100%.



Gambar 5.3. Kondisi ketika pemain mencapai akurasi 83,33%.



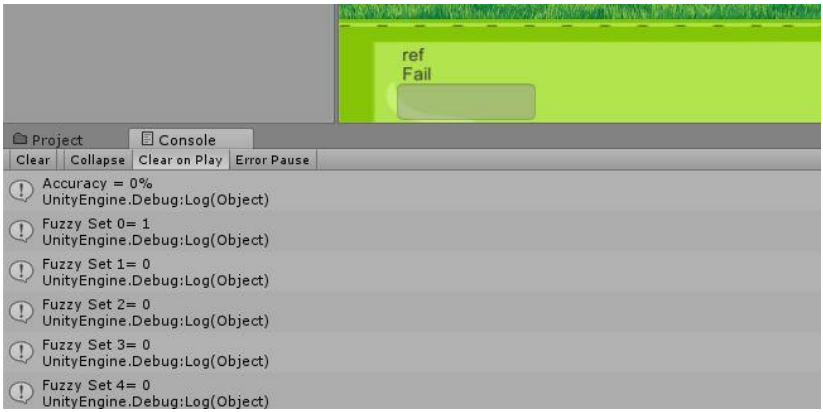
Gambar 5.4. Kondisi ketika pemain mencapai akurasi 80%.



Gambar 5.5. Kondisi ketika pemain mencapai akurasi 60%.

Gambar 5.6. Kondisi ketika pemain mencapai akurasi 40%.

Gambar 5.7. Kondisi ketika pemain mencapai akurasi 20%.



Gambar 5.8. Kondisi ketika pemain mencapai akurasi 0%.

5.2.1.2. Pengujian Fitur Penghitung Nilai Output

Pengujian fitur penghitung nilai *output* bertujuan menguji apakah proses *defuzzyfication* menghasilkan nilai output yang benar Skenario pengujian adalah dengan cara memasukkan kombinasi *input* yang menghasilkan nilai akurasi tertentu pada giliran kelima. Pada tahap ini dilihat apakah perubahan tingkat kesulitan sesuai dengan yang diharapkan. Hasil pengujian fitur ini dapat dilihat pada Tabel 5.8, Tabel 5.9, Tabel 5.10, Tabel 5.11, Tabel 5.12, dan Tabel 5.13.

Tabel 5.8. Pengujian fitur penghitung nilai *output.*

ID	**UJ.WM-008**
Referensi Fungsionalitas	2
Nama	Pengujian Fitur Penghitung Nilai *Output*
Tujuan Pengujian	Menguji fitur penghitung nilai *output* beserta hasilnya.
Skenario 1	Pemain mencapai tingkat kesulitan 3.5.

Kondisi Awal	Aplikasi menunjukkan tingkat kesulitan = 3 seperti ditunjukkan pada Gambar 5.9.
Masukan	Pemain mengetikkan 5 kata dengan benar berturut-turut.
Hasil yang Diharapkan	Tingkat kesulitan berubah menjadi 3.5.
Hasil yang Didapat	Tingkat kesulitan berubah menjadi 3.5.
Hasil Pengujian	Berhasil
Kondisi Akhir	Tampilan hasil pengujian dapat dilihat pada Gambar 5.10.

Tabel 5.9. Pengujian fitur penghitung nilai *output.*

ID	**UJ.WM-009**
Referensi Fungsionalitas	2
Nama	Pengujian Fitur Penghitung Nilai *Output*
Tujuan Pengujian	Menguji fitur penghitung nilai *output* beserta hasilnya.
Skenario 1	Pemain mencapai tingkat kesulitan 3.3.
Kondisi Awal	Aplikasi menunjukkan tingkat kesulitan = 3.5 seperti ditunjukkan pada Gambar 5.10.
Masukan	Pemain gagal mengetikkan kata lalu mengetikkan 4 kata dengan benar berturut-turut.
Hasil yang Diharapkan	Tingkat kesulitan berubah menjadi 3.8.
Hasil yang Didapat	Tingkat kesulitan berubah menjadi 3.8.

Hasil Pengujian	Berhasil
Kondisi Akhir	Tampilan hasil pengujian dapat dilihat pada Gambar 5.11.

Tabel 5.10. Pengujian fitur penghitung nilai *output.*

ID	**UJ.WM-0010**
Referensi Fungsionalitas	2
Nama	Pengujian Fitur Penghitung Nilai *Output*
Tujuan Pengujian	Menguji fitur penghitung nilai *output* beserta hasilnya.
Skenario 1	Pemain mencapai tingkat kesulitan 3.9.
Kondisi Awal	Aplikasi menunjukkan tingkat kesulitan = 3.8 seperti ditunjukkan pada Gambar 5.11.
Masukan	Pemain gagal mengetikkan 2 kata lalu mengetikkan 3 kata dengan benar berturut-turut.
Hasil yang Diharapkan	Tingkat kesulitan berubah menjadi 3.9.
Hasil yang Didapat	Tingkat kesulitan berubah menjadi 3.9.
Hasil Pengujian	Berhasil
Kondisi Akhir	Tampilan hasil pengujian dapat dilihat pada Gambar 5.12.

Tabel 5.11. Pengujian fitur penghitung nilai *output.*

ID	**UJ.WM-0011**
Referensi Fungsionalitas	2

Nama	Pengujian Fitur Penghitung Nilai *Output*
Tujuan Pengujian	Menguji fitur penghitung nilai *output* beserta hasilnya.
Skenario 1	Pemain mencapai tingkat kesulitan 3.8.
Kondisi Awal	Aplikasi menunjukkan tingkat kesulitan = 3.9 seperti ditunjukkan pada Gambar 5.12.
Masukan	Pemain gagal mengetikkan 3 kata lalu mengetikkan 2 kata dengan benar berturut-turut.
Hasil yang Diharapkan	Tingkat kesulitan berubah menjadi 3.8.
Hasil yang Didapat	Tingkat kesulitan berubah menjadi 3.8.
Hasil Pengujian	Berhasil
Kondisi Akhir	Tampilan hasil pengujian dapat dilihat pada Gambar 5.13.

Tabel 5.12. Pengujian fitur penghitung nilai *output.*

ID	**UJ.WM-0012**
Referensi Fungsionalitas	2
Nama	Pengujian Fitur Penghitung Nilai *Output*
Tujuan Pengujian	Menguji fitur penghitung nilai *output* beserta hasilnya.
Skenario 1	Pemain mencapai tingkat kesulitan 3.5.
Kondisi Awal	Aplikasi menunjukkan tingkat kesulitan = 3.8 seperti ditunjukkan pada Gambar 5.13.

Masukan	Pemain gagal mengetikkan 4 kata lalu mengetikkan 1 kata dengan benar.
Hasil yang Diharapkan	Tingkat kesulitan berubah menjadi 3.5.
Hasil yang Didapat	Tingkat kesulitan berubah menjadi 3.5.
Hasil Pengujian	Berhasil
Kondisi Akhir	Tampilan hasil pengujian dapat dilihat pada Gambar 5.14.

Tabel 5.13. Pengujian fitur penghitung nilai *output.*

ID	**UJ.WM-0013**
Referensi Fungsionalitas	2
Nama	Pengujian Fitur Penghitung Nilai *Output*
Tujuan Pengujian	Menguji fitur penghitung nilai *output* beserta hasilnya.
Skenario 1	Pemain mencapai tingkat kesulitan 3.
Kondisi Awal	Aplikasi menunjukkan tingkat kesulitan = 3.5 seperti ditunjukkan pada Gambar 5.14.
Masukan	Pemain gagal mengetikkan 5 kata.
Hasil yang Diharapkan	Tingkat kesulitan berubah menjadi 3.
Hasil yang Didapat	Tingkat kesulitan berubah menjadi 3.
Hasil Pengujian	Berhasil

Kondisi Akhir	Tampilan hasil pengujian dapat dilihat pada Gambar 5.15.

Gambar 5.9. Kondisi awal tingkat kesulitan.

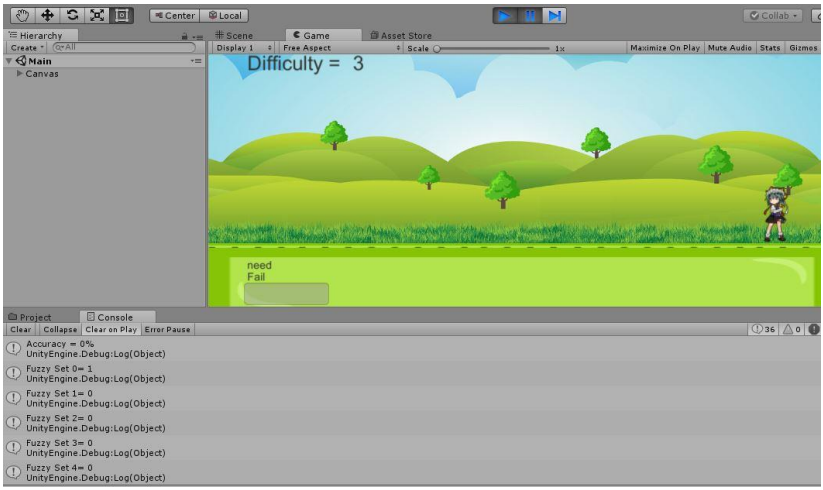
Gambar 5.10. Kondisi tingkat kesulitan setelah lima giliran dengan akurasi 100%.

Gambar 5.11. Kondisi tingkat kesulitan setelah lima giliran dengan akurasi 80%.

Gambar 5.12. Kondisi tingkat kesulitan setelah lima giliran dengan akurasi 60%.

Gambar 5.13. Kondisi tingkat kesulitan setelah lima giliran dengan akurasi 40%.

Gambar 5.14. Kondisi tingkat kesulitan setelah lima giliran dengan akurasi 20%.



Gambar 5.15. Kondisi tingkat kesulitan setelah lima giliran dengan akurasi 0%.

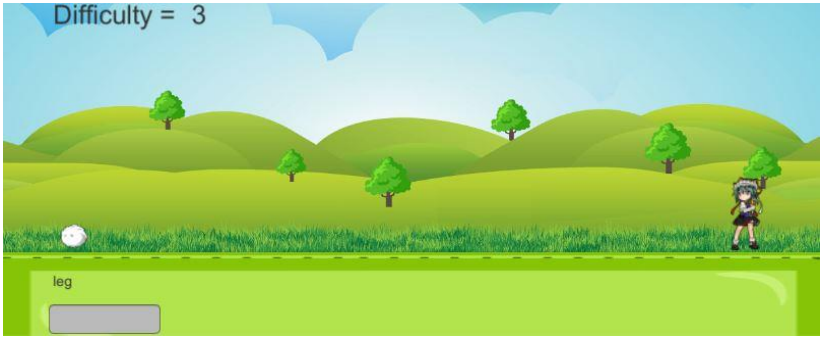
5.2.1.3. Pengujian Fitur Pemilih Kata

Pengujian fitur pemilih kata bertujuan menguji kata yang dipilih sesuai dengan tingkat kesulitan. Skenario pengujian adalah dengan cara menjalankan aplikasi hingga mencapai tingkat kesulitan 3.5. Pada tahap ini dilihat apakah terdapat perbedaan jumlah huruf pada tingkat kesulitan 3 dan 3.5. Hasil pengujian fitur ini dapat dilihat pada Tabel 5.3.

Tabel 5.14. Pengujian fitur penghitung kata.

ID	**UJ.WM-0014**
Referensi Fungsionalitas	3
Nama	Pengujian Fitur Pemilih Kata.
Tujuan Pengujian	Menguji fitur pemilih kata beserta hasilnya.

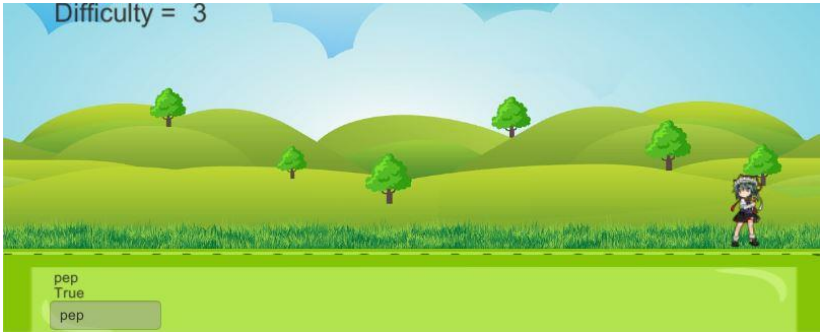
Skenario 1	Pemain mencapai tingkat kesulitan 3.5.
Kondisi Awal	Pemain memulai dari tingkat kesulitan 3.
Masukan	Pemain memasukkan 10 kata dengan benar secara berturut-turut.
Hasil yang Diharapkan	Kata berjumlah huruf lebih dari 3 baru muncul pada tingkat kesulitan lebih dari 3.
Hasil yang Didapat	Kata berjumlah huruf lebih dari 3 baru muncul pada tingkat kesulitan lebih dari 3.
Hasil Pengujian	Berhasil
Kondisi Akhir	Tampilan hasil pengujian dapat dilihat pada Gambar 5.16, 5.17, 5.18, 5.19, 5.20, 5.21, 5.22, 5.23, 5.24, dan 5.25



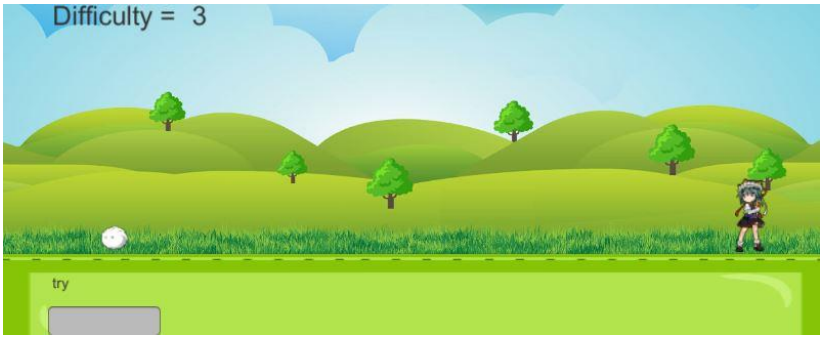
Gambar 5.16. Kata pertama yang harus diketikkan pemain.

Gambar 5.17. Kata kedua yang harus diketikkan pemain.

Gambar 5.18. Kata ketiga yang harus diketikkan pemain.



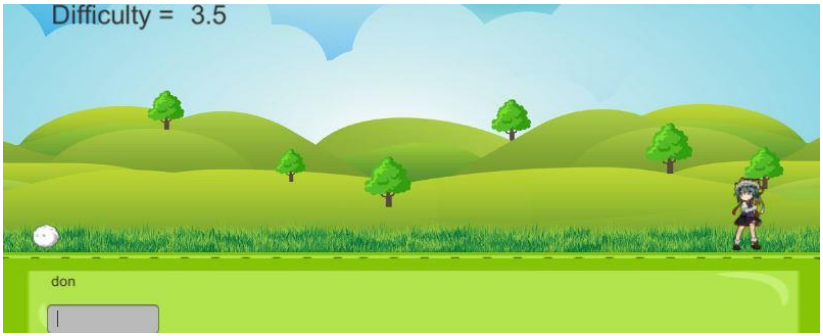
Gambar 5.19. Kata keempat yang harus diketikkan pemain.



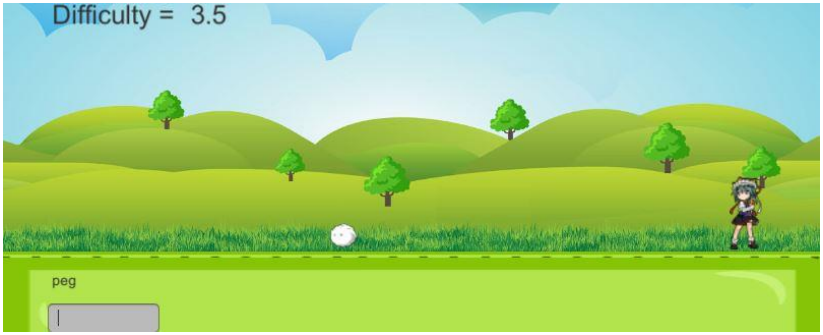
Gambar 5.20. Kata kelima yang harus diketikkan pemain.

Gambar 5.21. Kata keenam yang harus diketikkan pemain.

Gambar 5.22. Kata ketujuh yang harus diketikkan pemain.



Gambar 5.23. Kata kedelapan yang harus diketikkan pemain.



Gambar 5.24. Kata kesembilan yang harus diketikkan pemain.

Gambar 5.25. Kata kesepuluh yang harus diketikkan pemain.

5.3. **Evaluasi hasil pengujian**

Pada subbab ini hasil dari pengujian yang telah dilakukan pada fungsi-fungsi yang ada akan dibahas. Hasil pengujian ini merupakan hasil pengujian secara *black box*.

1. Pengujian fungsi penghitung nilai *fuzzy set* menunjukkan hasil sesuai seperti yang diharapkan. Pada pengujian tersebut fungsi *fuzzyfication* dapat menghitung nilai *fuzzy set* dengan benar.
2. Pengujian fungsi penghitung nilai *output* menunjukkan hasil sesuai seperti yang diharapkan. Pada pengujian tersebut transisi fungsi *defuzfyfication* dapat mengubah nilai *fuzzy* menjadi nilai skalar dengan benar.
3. Pengujian fungsi pemilih kata berhasil sesuai dengan yang diharapkan. Pada pengujian tersebut dapat terlihat kata yang dipilih merepresentasikan tingkat kesulitan.

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini akan diberikan kesimpulan yang diambil selama pengerjaan tugas akhir serta saran-saran tentang pengembangan yang dapat dilakukan terhadap tugas akhir ini di masa yang akan datang.

6.1. Kesimpulan

Dalam proses pengerjaan Tugas Akhir ini dimulai dari pendahuluan, kajian pustaka, analisis, perancangan, implementasi, hingga akhirnya pengujian diperoleh kesimpulan sebagai berikut.

1. *Typing game* dapat menjadi perantara yang baik untuk pembelajaran mengetik 10 jari.
2. DDA dapat dipergunakan untuk membuat *game* lebih menarik karena dapat membuat *game* tetap menantang bagi pemain dengan kemampuan tinggi dan tetap bisa dinikmati oleh pemain dengan kemampuan lebih rendah.
3. Unity mempermudah pembuatan *game* dengan berbagai fitur yang disediakannya, didukung dengan MonoFramework yang berkapabilitas *cross-platform.*

6.2. Saran

Berikut ini adalah saran-saran yang ditujukan untuk pengembangan dan perbaikan sistem yang mirip atau sejenis di masa yang akan datang. DDA dapat membuat *game* menjadi lebih menarik. Sebagai kakas yang berkembang, Unity juga terus mengeluarkan versi terbarunya dengan fitur-fitur baru yang dapat membantu membuat *game* yang dibuat menjadi lebih bervariasi, karena itu pastikanlah penggunaan versi terbaru.

DAFTAR PUSTAKA

[1] Unity, "Unity - create games," Unity3D, [Online]. Available: http://unity3d.com/pages/create-games. [Accessed 1 March 2014].

[2] Webopedia, "C# Definition," [Online]. Available: http://www.webopedia.com/TERM/C/C_sharp.html. [Accessed 1 March 2014].

[3] G. S. Keesee, "PBWorks," 27 May 2011. [Online]. Available: http://teachinglearningresources.pbworks.com/w/page-revisions/35130965/Educational%20Games. [Accessed 1 March 2017].

[4] Mono Project, "Mono Project," [Online]. Available: http://www.mono-project.com/docs/about-mono/. [Accessed 1 March 2017].

[5] Unity, "Unity3D," [Online]. Available: https://docs.unity3d.com/356/Documentation/ScriptReference/MonoBehaviour.html. [Accessed 1 March 2017].

[6] Game Ontology Project, "Game Ontology," 18 December 2008. [Online]. Available: http://www.gameontology.com/index.php/Dynamic_Difficulty_Adjustment. [Accessed 1 March 2017].

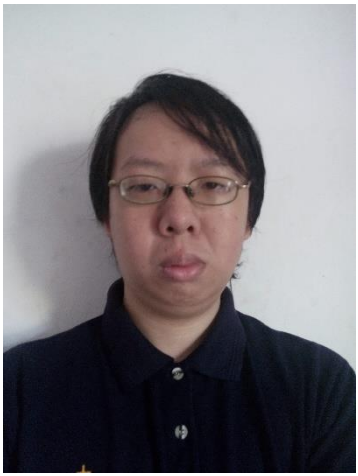
[7] E. Cox, The Fuzzy Systems Handbook, Boston: AP Professional, 1994.

[8] K. Salen and Z. Eric, Rules of Play, MIT Press, 2003.

[9] M. A. Arifianto, "Celotehan Mahasiswa," 13 March 2016. [Online]. Available: http://vanillabluse.blogspot.co.id/2016/03/penjelasan-tentang-desain-skenario.html. [Accessed 1 March 2017].

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS



Antonio Cahyadi Limantara, lahir di Magelang pada tanggal 22 November 1992, merupakan anak ketiga dari lima bersaudara. Penulis menempuh pendidikan dasar mulai kelas 1 hingga kelas 6 di SD Tarakanita, Magelang. Penulis melanjutkan pendidikan menengah di SMP Tarakanita, Magelang. Kemudian penulis melanjutkan pendidikan di SMAN 1 Magelang. Selanjutnya penulis melanjutkan pendidikan sarjana di Jurusan Teknik Informatika, Institut Teknologi Sepuluh Nopember Surabaya.

Di Jurusan Teknik Informatika, penulis mengambil bidang minat Interaksi Grafika dan Seni (IGS) . Penulis dapat dihubungi melalui alamat email liem.antonio@gmail.com