REPOSITOR, Vol. 2, No. 3, Maret 2020, Pp. 277-286
ISSN : 2714-7975
E-ISSN : 2716-1382

Implementasi Utility AI untuk Pembuatan Fitur Auto-Battle pada Game Dude Knight

Syauqi Hidayatul Hakim[*1], Ali Sofyan Kholimi[2], Lailatul Husniah[3]
[1,2,3]Teknik Informatika/Universitas Muhammadiyah Malang
syauqi_437238@webmail.umm.ac.id[*1], kholimi@umm.ac.id[2], husniah@umm.ac.id[3]

Abstrak

Grinding adalah kegiatan repetitif yang dilakukan untuk meningkatkan kemampuan karakter (level) dalam game terutama dengan genre RPG. Dengan sistem seperti ini, konten game selanjutnya akan sulit diraih oleh pemain yang ingin menamatkan game tetapi tidak memiliki banyak waktu bermain. Fitur auto-battle hadir untuk mengatasi masalah tersebut. Pemain dapat melakukan hal lain selagi auto-battle berjalan untuk grinding. Fitur tersebut haruslah dilengkapi dengan kecerdasan buatan (AI) yang bisa menyelesaikan pertarungan dengan baik. Hal tersebut semakin dibutuhkan terutama pada game dengan genre turn-based strategy seperti Dude Knight dimana strategi dibutuhkan untuk menang. Meskipun game ini memiliki auto-battle bawaan dari game engine yang digunakan, terdapat beberapa kekurangan pada auto-battle bawaan tersebut. Penelitian ini menawarkan auto-battle yang lebih baik menggunakan Utility AI. Perilaku pada auto-battle yang dibuat akan difokuskan pada AI yang lebih stabil dalam memenangkan pertarungan dan dapat memenangkan pertarungan lebih cepat. Kedua AI akan diuji dan akan dilakukan t-test sebelum melakukan perbandingan data. Hasilnya adalah Utility AI memiliki rasio kemenagan yang lebih besar dari AI bawaan. Utility AI juga memenangkan pertarungan lebih cepat.

Kata Kunci: *Turn-based Strategy, Auto-Battle, Utility AI, T-Test*

Abstract

Grinding is a repetitive activity to increase character's power (level) in games especially in RPG genre. With this kind of system, the next content of the game will be harder to reach by players that wanted to finish the game but they don't have much time to play. Auto-battle feature came to solve that problem. Players could do something else while their games grinding using auto-battle. This feature should be equipped with an artificial intelligence (AI) that can finish a battle properly. That is even more so for turn-based strategy games like Dude Knight which require strategies to win. Although the game engine used to develop Dude Knight has built-in auto-battle, there are still some drawbacks. This research proposes a better auto-battle using Utility AI. Creating an AI with more stable win and faster win is what will be focused on. Both AI will be tested and t-test will be done before comparing the data. The results are Utility AI have higher win rate than original AI. Utility AI also win the battles faster.

Keywords: *Turn-based Strategy, Auto-Battle, Utility AI, T-Test*

1. Pendahuluan

Grinding adalah kegiatan repetitif yang dilakukan untuk meningkatkan kemampuan karakter (*level*) dalam *game*[1]. Meskipun beberapa orang menyukai kegiatan tersebut ketika bermain, pemain yang tidak memiliki banyak waktu luang namun tetap ingin bermain tidak akan merasakan hal yang sama. Pemain akan membutuhkan waktu yang lama untuk melihat konten atau cerita selanjutnya. Pada tahun 2017 sebanyak 72% *gamer* atau pemain *game* di Amerika Serikat berusia di atas 18 tahun. Sedangkan rata-rata usia *gamer* adalah 35 tahun[2]. Hal ini menunjukkan bahwa banyak orang yang masih bermain *game* meskipun sudah memasuki usia kerja.

Dalam *game*, *Auto-mode* adalah fase ketika pemain memilih untuk memberikan kendalinya kepada AI[3]. Beberapa *game* menghadirkan fitur *auto-battle,* dimana kegiatan *grinding* akan diambil alih oleh AI. Sehingga pemain bisa meninggalkan *game* ke AI untuk melakukan pekerjaan lain. Ketika kembali, pemain akan melihat peningkatan kemampuan pada karakter *game*-nya. Namun kebanyakan dari *game* tersebut memiliki mekanisme pertarungan yang sederhana.

Sehingga, AI untuk *auto-battle* juga tidak terlalu rumit. Bahkan untuk *game* dengan *genre turn-based strategy* sekalipun. *Game* dengan *genre* tersebut memiliki unsur strategi dalam memainkanya layaknya sebuah catur. Pemain harus mempertimbangkan langkah atau aksi yang akan dilakukannya di setiap giliran. Dude Knight adalah sebugh *game* dengan unsur RPG dan *turn-based strategy*. *Game* ini memiliki banyak parameter yang harus dipertimbangkan dalam strategi pertarungannya.

Dude Knight dibuat dengan *game engine* RPG Maker VX Ace. Dalam *game engine* tersebut terdapat fitur *auto-battle* bawaan. Namun fitur tersebut tidak bisa di aktifkan atau di non-aktifkan sesuai keinginan pemain layaknya fitur *auto-battle* pada umumnya. AI dari *auto-battle* bawaan tersebut hanya memilih *skill* atau jenis serangan yang dapat menghasilkan kerusakan terbesar pada musuh. Begitu juga dengan penyembuhan yang bisa diberikan ke karakter teman. *Skill* dengan kerusakan tinggi cenderung membutuhkan energi yang banyak. Jika energi tersebut habis maka karakter hanya bisa menggunakan serangan biasa. Energi pada Dude Knight disebut *MP (Mana Point)* dan *TP (Technique Point)*. Dude Knight juga memiliki *skill* yang bisa memberikan *status effect*. *Status effect* adalah status yang dapat membahayakan penderita. Contoh status tersebut adalah racun yang bisa mengurangi darah atau nyawa (*HP*) penderita setiap giliran. Adapula *silence* yang dapat membuat penderita tidak bisa menggunakan *skill* sihir. *Status effect* memiliki potensi untuk membalikkan keadaan. Status tersebut akan hilang setelah beberapa giliran telah berjalan, atau menggunakan *item* atau *skill* yang dapat menghilangkan status tertentu. Sayangnya *auto-battle* tidak akan menggunakan *item* atau *skill* tersebut. Kondisi *Auto-Battle* pada RPG Maker dan strategi bertarung pada Dude Knight yang cukup kompleks akan menyusahkan pemain yang tidak memiliki banyak waktu luang untuk *grinding.*

Dengan banyaknya parameter yang dimiliki *game* Dude Knight dalam memilih strategi, penulis mencoba menyelesaikan permasalahan tersebut dengan membuat *auto-battle* baru menggunakan Utility AI. Utility AI yang dikenalkan oleh Dave Mark adalah AI untuk pengambilan keputusan dan behavior yang menggunakan konsep utility atau kegunaan. Suatu benda memiliki nilai kegunaan tergantung dimana, kapan, dan bagaimana benda tersebut digunakan[4]. Begitu juga jika diterapkan pada AI pengambilan keputusan. Suatu keputusan akan sangat bermanfaat jika kondisi yang mempengaruhinya sesuai. Utility AI menghitung nilai kegunaan tiap keputusan atau aksi berdasarkan kondisi yang mempengaruhinya kemudian memilih aksi dengan nilai kegunaan terbaik. Kondisi tersebut berupa kombinasi dari kebutuhan pada saat itu dan bagaimana suatu aksi dapat memenuhi kebutuhan tersebut[5].

Pada penelitian[5][6][7], Utility AI digunakan untuk membuat suatu agen dalam suatu kondisi yang memiliki *behavior* layaknya manusia. Namun penelitian tersebut mengombinasikan Utility AI dengan Behavior tree karena peneliti menginginkan *behavior* tertentu. Parameter yang banyak dan kompleks pada strategi bertarung Dude Knight dapat diatasi dengan baik menggunakan Utility AI.

2. Metode Penelitian

Sebelum beranjak ke rancangan pembuatan *auto-battle* menggunakan utility AI, mula-mula perlu diketahui cara kerja *skill* di *game* Dude Knight. Kemudian performa *auto-battle* bawaan beserta perilaku yang ingin dicapai berdasarkan analisa tersebut.

2.1 Skill yang Digunakan di Game Dude Knight

Dude Knight memiliki sebanyak 123 *skill* termasuk aksi *attack*, *guard,* dan *escape. Skill* pada game ini memiliki banyak atribut seperti nama, *icon,* deskripsi, formula, efek, *skill type* atau golongan *skill*, dll. Namun hanya ada 12 atribut yang dapat mempengaruhi strategi. Atribut tersebut dapat dilihat pada Tabel 1.

Tabel 1. Atribut Pada Setiap *Skill* yang Dapat Mempengaruhi Strategi

No	Atribut	Deskripsi
1	*Damage Type*	Tipe kerusakan. Tipe tersebut adalah *HP Damage, HP Recover, MP Recover, HP Drain, MP drain,* dan *None.*
2	*Element*	Jenis *element*. *Element* tersebut adalah normal, *none, physical, absorb, fire, ice, thunder, water, earth, wind, holy,* dan *dark.* Beberapa musuh kebal terhadap satu atau lebih *element*. Perhitungan *element* dilakukan setelah perhitungan *formula*.

3	*Formula*	Perhitungan *damage* mula-mula yang akan dihasilkan
4	*Variance*	Mencegah menghasilkan *damage* yang sama setiap kali menggunakan *skill.* Variasi *damage* dihitung terakhir setelah *formula* dan *element*.
5	*Critcal*	Bersifat *boolean. Critical* adalah kondisi khusus dimana suatu serangan menghasilkan beberapa persen (>=50%) lebih besar dari biasanya. Kemungkinan kondisi ini terjadi dan besarnya tambahan kerusakan berbeda-beda untuk tiap karakter.
6	*Cost*	(TP/MP) Biaya yang dibutuhkan ketika menggunakan suatu *skill*.
7	*Scope*	Target area yang dipengaruhi. Target tersebut adalah *none, one enemy, all enemies, 1 random enemy, 2 random enemies, 3 random enemies, 4 random enemies, one ally, all allies, one ally (dead), all allies (dead),* dan *the user.*
8	*Effect*	Efek-efek yang ditimbulkan selain *damage*.
9	*Speed*	Waktu yang dibutuhkan untuk melancarkan suatu serangan
10	*Success Rate*	Tingkat kesuksesan suatu serangan mengenai targetnya
11	*Repeats*	Jumlah serangan
12	*TP Gain*	Bersifat *boolean*. Memberikan sejumlah *TP* jika mengenai target.

2.2 Perilaku yang Ingin Dicapai untuk Auto-Battle Utility AI

Penentuan perilaku *auto-battle* yang baru bisa dilihat kembali ke alasan diciptakannya *auto-battle*, yaitu mengatasi permasalahan *grinding* untuk pemain yang memiliki waktu terbatas atau sedikit untuk bermain. Sehingga, perilaku yang diharapkan untuk *auto-battle* yang baru adalah sebagai berikut:

a. Tidak mudah kalah. Kekalahan akan mengakibatkan *game over* dan progress akan kembali ke titik penyimpanan sebelumnya. Bisa dikatakan *grinding* yang dilakukan akan sia-sia.
b. Dapat menyelesaikan setiap pertarungan dengan cepat, karena pemain sendiri tidak memiliki banyak waktu.

2.3 Pembagian Atribut

12 Atribut yang mempengaruhi strategi pada setiap *skill* akan dibagi menjadi dua jenis. Jenis pertama adalah yang nilai *utility*-nya tidak terpengaruh oleh kondisi pertarungan pada saat itu. Sehingga, atribut-atribut ini akan memiliki nilai *utility* yang tetap setiap saat. Sebut saja atribut tersebut memiliki *static utility* atau nilai utilitas tetap. Atribut-atribut dengan *static utility* adalah *variance, critical, speed, success rate,* dan *repeat.*

Sedangkan atribut yang perhitungan *utility*-nya menyesuaikan kondisi pertarungan pada saat itu disebut memiliki *dynamic utility* atau nilai utilitas dinamis. Atribut dengan *dynamic utility* adalah *damage type, formula, element, cost, scope, effect,* dan *TP Gain.* Pembagian atribut ini dilakukan hanya untuk memudahkan implementasi dan *debugging* terhadap perilaku yang dihasilkan apakah sudah sesuai dengan yang diharapkan atau belum. Selain itu, pembagian atribut ini dilakukan untuk menentukkan atribut apa yang memiliki pengaruh besar dan atribut apa yang memiliki pengaruh kecil dalam penentuan strategi. Atribut dengan *dynamic utility* memiliki pengaruh besar dalam penentuan strategi, sehingga perlu dicari rumus perhitungan nilai *utility* yang sesuai dengan rancangan. Pengembang dapat menggunakan rumus apapun untuk menciptakan perilakunya. Rumus tersebut bisa berupa grafik linier, kuadratik, atau eksponensial[8].

2.4 Perhitungan Nilai Utilitas untuk Setiap Atribut

a. Variance

Variance yang terlalu rendah akan membuat *damage* yang dihasilkan cenderung stagnan atau tidak banyak perubahan. Sedangkan *variance* yang terlalu tinggi akan membuat *damage* tidak konsisten. Maka dari itu diambil nilai pertengahan yaitu 20-30. Nilai *variance* di luar jarak

tersebut tidak akan memiliki nilai utilitas. Sedangkan jika masuk dalam jarak tersebut akan mendapat penambahan utilitas sebesar 0.015.

b. Critical

Jika nilai *critical* pada suatu *skill* adalah *false,* maka tidak mendapatkan tambahan utilitas. Sedangkan jika *true*, maka nilai utilitasnya akan bertambah sebesar 0.05.

c. Speed

Rata-rata *skill* dalam *game* Dude Knight memiliki *speed* sebesar 0. Beberapa memiliki *speed* 1000 dan 2000 agar *skill* tersebut adalah yang pertama dilancarkan dalam suatu giliran. Terdapat satu *skill* dengan nilai *speed* 100. *Skill* ini memang lebih cepat dari pada *skill* pada umumnya, namun tidak didesain untuk dilancarkan pertama kali dalam suatu giliran. Sisanya memiliki *speed* sekitar -5 hingga -40. Maka dari itu hanya speed dibawah 101 yang akan memiliki perubahan nilai utilitas dengan perhitungan seperti pada Persamaan 1.

$$\frac{speed}{2000} \tag{1}$$

d. Success Rate

Tidak ada *skill* yang memiliki *success rate* di atas 100%, sehingga nilai utilitas untuk *skill* dengan *success rate* di bawah 100% akan berkurang sebesar hasil dari Persamaan 2.

$$\frac{100 - success\ rate}{1000} \tag{2}$$

e. Repeats

Setiap kelipatan *repeats* dalam suatu *skill* (kecuali kelipatan satu), nilai utilitasnya akan bertambah sebesar 0.025.

f. Damage Type

Setiap jenis *damage type* memiliki cara pertihungan nilai utilitas yang berbeda. Tipe kerusakan beserta penyebab dan perhitungannya dapat dilihat pada Tabel 2.

Tabel 2. Perhitungan Nilai Utilitas Dari Atribut Damage Type

Damage Type	Penyebab	Modifier
HP Damage	Jumlah musuh yang masih hidup	x 0.25
HP Recover	Setiap teman yang memiliki HP di bawah 85 %	$(e^{-hp\ rate})$ *
MP Recover	Jumlah musuh yang masih hidup MP pengguna	x 0.05 $(e^{-2(mp\ rate)})$*
HP Drain	HP pengguna Jumlah musuh yang masih hidup	$(e^{-2(hp\ rate)})$* x 0.05
MP Drain	MP pengguna Jumlah musuh yang masih hidup	$(e^{-2(mp\ rate)})$* x 0.05

*hp rate didapat dari hp saat itu dibagi maksimum hp suatu karakter. Hal yang sama juga berlaku untuk mp rate

g. Formula

Hasil perhitungan *damage* dari *formula* tidak langsung dikonversikan ke dalam nilai utilitas, melainkan diolah lagi oleh atribut *element*. Namun untuk *skill* jenis penyembuhan, akan dihitungan efektifitasnya. Maksud dari efektifas penyembuhan adalah tingkat kesamaan *hp* yang bisa disembuhkan dalam satu kali penggunaan *skill* penyembuhan dengan *hp* yang hilang pada suatu karakter. Jika sudah melalui perhitungan *element* atau efektifitas, maka nilai utilitasnya bisa didapat melalui Persamaan 3.

$$\frac{(formula/100)}{level\ pengguna} \tag{3}$$

h. Element

Atribut *element* tidak menghasilkan nilai utilitas, namun tetap bisa mempengaruhi nilai utilitas dari atribut *formula*. Suatu musuh memiliki fitur yang disebut *element rate* yang berarti daya tahan terhadap suatu element. Musuh tidak lemah maupun tidak tahan terhadap suatu *element* jika memiliki *element rate* 100%. Contoh suatu musuh memiliki fitur *element rate [fire]* 200%, maka *skill* dengan *element* api akan menghasilkan kerusakan dua kali lebih banyak.

i. Cost

Nilai utilitas dari *TP Cost* didapat melalui Persamaan 4. Sedangkan nilai utilitas dari *MP Cost* didapat melalui Persamaan 5.

$$e^{-\frac{tp - tp.cost}{4}} + 0.15 \quad (4)$$

$$\frac{mp - mp\ cost}{max\ mp} x\ 0.3 \quad (5)$$

j. Scope

Atribut ini adalah atribut terakhir yang dihitung, karena melalui atribut ini, target akan dipilih. *Scope* hanya merubah nilai utilitas yang didapat dari atribut *formula*. Nilai utilitas *formula* akan dikali sejumlah musuh yang ada jika *skill* yang digunakan menarget semua musuh. Jika hanya satu musuh, maka akan dicari target dengan nilai utilitas paling tinggi.

k. Effects

Penentuan nilai utilitas *effect* dihitung dari banyaknya musuh yang belum terkena *effect* yang dihasilkan suatu *skill* dan musuh yang tidak memiliki ketahanan pada *effect* tersebut.

l. TP Gain

Nilai utilitas bertambah sebesar 0.025 setiap *TP* yang didapat.

2.5 Perhitungan Nilai Utilitas untuk Effect

Dude Knight memiliki tujuh jenis *status effects. Status effects* tersebut adalah *poison, blind, silence, confusion, sleep,* dan *paralyze*. Menurut kegunaannya, keenam efek tersebut dapat dibagi menjadi dua tipe, yaitu efek perusak dan efek pelumpuh.

a. Efek Perusak

Status effect yang tergolong efek perusak adalah *poison* dan *confusion. Poison* akan memberikan kerusakan setiap giliran tergantung maksimal *hp* yang dimiliki penderita. *Confusion* akan mengakibatkan penderita berkemungkinan menyerang diri sendiri atau teman. Efek jenis ini efektif digunakan pada musuh dengan tingkat *hp* yang tinggi. Penentuan nilai utilitas untuk jenis efek perusak didapat melalui Persamaan 6.

$$\frac{hp\ rate\ musuh}{10} \quad (6)$$

Musuh yang sudah menderita efek yang sama dan musuh yang kebal terhadap efek tersebut tidak akan dihitung.

b. Efek Pelumpuh

Status effect yang tergolong efek pelumpuh adalah *blind, silence, sleep,* dan *paralyze*. *Blind* mengakibatkan penderita tidak bisa membidik target dengan baik. Namun serangan *magic* tidak terpengaruh. *Silence* mengakibatkan penderita tidak bisa menggunakan serangan *magic. Sleep* mengakibatkan penderita tidur sehingga tidak bisa melancarkan serangan apapun. Namun jika penderita menerima kerusakan, efek *sleep* akan langsung hilang. Sedangkan *paralyze* mengakibatkan penderita berkemungkinan gagal melancarkan aksinya. Efek jenis ini efektif digunakan ketika terdapat banyak musuh. Penentuan nilai utilitas untuk jenis efek pelumpuh didapat dari Persamaan 7.

$$\frac{jumlah\ musuh\ yang\ tidak\ terpengaruh}{20} \quad (7)$$

Musuh yang tidak terpengaruh maksudnya adalah musuh yang tidak sedang menderita efek yang akan diberikan beserta musuh yang kebal terhadap efek tersebut.

2.5 Perhitungan Nilai Utilitas untuk Beberapa Skill Khusus

Terdapat beberapa *skill* yang memiliki perhitungan nilai utilitas sendiri dikarenakan *skill* tersebut berguna untuk disituasi tertentu. *Skill* tersebut adalah:

a. Guard

Guard akan mengurangi kerusakan yang diterima sebanyak 50%. *Skill* ini baik digunakan untuk karakter yang kondisi *hp* tinggal sedikit. Perhitungan nilai utilitas untuk *skill* ini adalah jika *hp* karakter dibawah 25%, maka nilai utilitas bertambah 1.25.

b. Cure I

Cure I dapat menghilangkan *status effect poison* dan *paraylsis* yang diderita suatu karakter. Sehingga setiap terdapat karakter yang menderita kedua efek tersebut, maka nilai utilitas akan bertambah sebesar 3.1

c. Cure II

Cure II dapat menghilangkan segala jenis *status effect* yang diderita suatu karakter. Sehingga setiap terdapat karakter yang menderita efek selain *poison* dan *paralysis,* maka nilai utilitas akan bertambah sebesar 3.2. Sedangkan setiap karakter yang menderita *poison* dan *paralyis,* maka nilai utilitas akan bertambah sebesar 3. *Cure I & II* mendapat penambahan nilai utilitas yang tinggi agar penyembuhan *status effect* menjadi prioritas utama.

d. Raise I & II

Raise I & II dapat menghidupkan kembali karakter yang telah gugur. Namun *hp* karakter yang dihidupkan hanya kembali beberap persen. Perbedaannya terletak pada *raise II* dapat mengembalikan *hp* lebih banyak ketika menghidupkan teman. Jika terdapat teman yang gugur, maka nilai utilitas akan bertambah 3.5 untuk *raise I* dan 3.6 untuk *raise II.*

e. Provoke

Provoke akan mengakibatkan musuh cenderung menyerang karakter yang menggunakan *skill* ini. *Provoke* sangat bermanfaat ketika terdapat teman yang hampir gugur, sehingga musuh tidak akan menyerang teman tersebut. Nilai utilitas untuk *skill* ini bertambah sebesar 0.3 setiap teman dengan *hp* dibawah 50%.

2.6 Pengujian

Pengujian yang dilakukan hanyalah performa *auto-battle* Utility AI ketika bertarung. Pengaruh *auto-battle* terhadap kesenangan ketika bermain tidak perlu diuji. Jika menggunakan taksonomi milik Bartle, pemain dengan kategori *explorers* lebih tertarik dengan meningkatkan kemampuan untuk mengeksplorasi dunia yang ditawarkan di *game.* Beberapa pemain juga merasakan kesenangan ketika melihat karakter yang mereka mainkan di dalam *game* menggunakan benda yang kuat yang didapat dari mengalahkan musuh[9].

Terdapat dua jenis pengujian untuk mengetahui performa *auto-battle* Utility AI. Pertama, baik *auto-battle* bawaan maupun *auto-battle* Utility AI akan dilakukan simulasi pertarungan melawan setiap *troops* dalam *game. Troops* adalah kombinasi musuh yang akan muncul untuk dilawan oleh pemain. Pada *game* Dude Knight, tidak ada lebih dari satu jenis musuh dalam satu *troops.* Melainkan suatu *troops* hanya terdiri dari satu jenis musuh sebanyak 1 hingga 3 unit. Empat dari karakter *game* akan dipilih untuk bertarung sebagai tim melawan *troops* tersebut. Tim karakter tersebut sama untuk semua pengujian dan untuk kedua AI. Setiap *troops* yang dilawan akan dilakukan pengujian sebanyak 30 kali. Data dari pengujian tersebut akan akan dibandingkan. Data yang akan dibandingkan adalah sebagai berikut

1. Menang atau kalah
2. Jumlah *turn* atau giliran yang dibutuhkan untuk menyelesaikan pertarungan, baik menang maupun kalah

Pengujian kedua adalah simulasi pertarungan langsung antar kedua AI. Setiap *unit* dari kedua kubu dibuat sama baik dari tingkat kekuatan maupun *skill* yang dimiliki. Simulasi pertarungan akan dilakukan sebanyak 30 kali. Data yang diambil hanya rasio kemenangan, kekalahan, dan jumlah giliran yang dibutuhkan untuk menyelesaikan pertarungan, baik menang maupun kalah.

Khusus untuk pengujian pertama, data yang didapat akan dilakukan t-test untuk mengetahui adanya perbedaan signifikan antar kedua AI ketika melawan suatu *troops*[10]. Jika ketika melawan suatu *troops* terbukti adanya perbedaan signifikan, barulah akan dibandingkan

AI mana yang memiliki rasio kemenangan lebih tinggi dan giliran yang dibutuhkan untuk menang lebih cepat. Adanya *status effect* dan *critical* yang sangat berhubungan dengan peluang akan membuat data yang didapat bisa bervariasi. Maka dari itu, *level* signifikansi(α) yang digunakan untuk t-test adalah 0.025.

Dude Knight memiliki 10 karakter dan 9 jenis *class*. *Class* inilah yang menentukkan peran suatu karakter dalam pertarungan. *Class* ini pula yang menentukan *skill* yang akan diterima suatu karakter ketika mencapai *level* yang dibutuhkan. Namun dalam pertarungan, hanya ada 4 karakter yang dapat digunakan. Maka dipilihlah 4 karakter dengan *class* berbeda atau peran yang dapat menyeimbangkan tim. Peran tersebut adalah peran yang umum digunakan pada *game* RPG, yaitu *damage dealer, tanker,* dan *support/healer*. Sehingga, *class* yang dipilih adalah *Soldier, Priestess, Paladin,* dan *Witch*. Tim dengan kombinasi *class* ini akan digunakan di setiap simulasi pertarungan. Terdapat sebanyak 28 kombinasi musuh (*troops*) bukan tergolong boss yang dapat dihadapi pemain selama bermain. Sehingga terdapat 840 simulasi pertarungan untuk setiap AI.

3. Hasil dan Diskusi

3.1 Hasil Pertarungan Melawan Troops

Dari hasil simulasi pertarungan yang dilakukan, kedua AI memiliki rasio kemenangan 100% untuk setiap *troops* kecuali ketika melawan Sahagin. Ketika melawan Sahagin, kedua AI mengalami kekalahan sebanyak 2 kali dari 30 pertarungan. Sedangkan dari hasil t-test melawan 28 *troops*, 14 diantaranya terbukti tidak ada perbedaan signifikan, dan 14 sisanya terbukti terdapat perbedaan yang signifikan. Setelah mengetahui *troops* mana yang terbukti adanya perbedaan, langkah selanjutnya adalah melakukan perbandingan antar kedua AI melawan 14 *troops* dimana perbedaan telah terbukti tersebut.

Tabel 3. Rata-rata Giliran yang Dibutuhkan Untuk Menang Pada Kedua AI

No	Troops	Jenis AI	Rata-rata turn
1	Rate	Original	2.767
		Utility AI	2.467
2	Wisp	Original	3.567
		Utility AI	3.000
3	Scorpion	Original	3.867
		Utility AI	3.467
4	Gazer	Original	5.067
		Utility AI	3.933
5	Puppet	Original	2.367
		Utility AI	2.000
6	Cockatrice	Original	3.267
		Utility AI	2.433
7	Mimic	Original	2.933
		Utility AI	2.267
8	Werewolf	Original	4.967
		Utility AI	3.567
9	Ogre	Original	2.867
		Utility AI	2.033
10	Gargoyle	Original	8.167
		Utility AI	5.533
11	Lamia	Original	7.433
		Utility AI	3.900
12	Vampire	Original	5.167
		Utility AI	3.967
13	Succubus	Original	27.933
		Utility AI	16.467
14	Demon	Original	10.067
		Utility AI	6.533

Dari setiap *troops* pada Tabel 3, *auto-battle* menggunakan Utility AI memiliki rata-rata giliran yang dibutuhkan untuk menang lebih kecil dari *auto-battle* bawaan. Hal ini menunjukkan bahwa *auto-battle* Utility AI dapat memenangkan pertarungan lebih cepat dari pada *auto-battle* bawaan.

3.2 Hasil Pertarungan Antara Utility AI dengan AI Bawaan

Pada Tabel 4, dapat dilihat bahwa Utility AI memenangkan 26 dari 30 pertarungan atas AI bawaan. Jumlah giliran yang dibutuhkan oleh Utility AI untuk menang juga lebih cepat dari pada jumlah giliran yang dibutuhkan untuk kalah.

Tabel 4 Hasil Pengujian Auto-Battle *Utility AI Melawan* Auto-Battle *Bawaan*

Pengujian ke	Pemenang	Jumlah giliran
1	Utility AI	10
2	Utility AI	5
3	Utility AI	6
4	Utility AI	10
5	Utility AI	6
6	Utility AI	6
7	Utility AI	7
8	Utility AI	8
9	Utility AI	8
10	Utility AI	7
11	Original	6
12	Utility AI	6
13	Utility AI	7
14	Utility AI	7
15	Utility AI	6
16	Utility AI	5
17	Original	6
18	Utility AI	6
19	Original	7
20	Utility AI	9
21	Original	12
22	Utility AI	7
23	Utility AI	10
24	Utility AI	7
25	Utility AI	8
26	Utility AI	8
27	Utility AI	7
28	Utility AI	7
29	Utility AI	8
30	Utility AI	6
Rata-rata	Turn (Utility AI)	7.192308
	Turn (Original)	7.75

4. Kesimpulan

Berdasarkan hasil pengujian yang telah dilakukan dapat disimpulkan bahwa fitur *auto-battle* menggunakan Utility AI telah berhasil dibuat. Pengujian *auto-battle* dapat dilakukan dengan membandingkan *auto-battle* bawaan Dude Knight dengan *auto-battle* Utility AI yang telah dibuat. Hasil pengujian pertama adalah *auto-battle* Utility AI dan *auto-battle* bawaan memiliki rasio kemenangan yang sama untuk setiap *troops* yang dilawan yaitu sebanyak 27 dari 28 *troops,* dengan rasio kemenagan 100%. Sedangkan satu *troops* sisanya, kedua AI memiliki rasio kemenangan 93%. Hasil t-test dari pengujian pertama adalah 14 dari 28 *troops* yang dilawan memiliki p-value dibawah *level* signifikansi (α=0.025) yang berarti perbedaan yang terjadi pada 14 *troops* tersebut bukan karena kebetulan. Pada 14 *troops* tersebut, *auto-battle* Utility AI memenangkan pertarungan lebih cepat dibandingkan *auto-battle* bawaan. Hasil pengujian kedua adalah *auto-battle* Utility AI memenangkan 26 dari 30 pertarungan melawan *auto-battle* bawaan.

Sehingga bisa disimpulkan bahwa *auto-battle* Utility AI memiliki kemenangan yang lebih stabil dan memenangkan pertarungan lebih cepat seperti yang sudah diharapkan.

Referensi

[1] F. Mayra, *An Introduction to Game Studies - Games in Culture*. London: SAGE Publications, 2008.
[2] Entertainment Software Association, "2017 Essential Facts About The Computer And Video Game Industry," 2017.
[3] S. Fizek, "The work of game in the age of automation," *Gaming and Virtual Worlds*, pp. 197–201, 2018.
[4] D. Mark, *Behavioral Mathematics For Game AI*. Boston: Course Technology, 2009.
[5] M. N. M. Othman and H. Haron, "Implementing game artificial intelligence to decision making of agents in emergency egress," *2014 8th Malaysian Softw. Eng. Conf. MySEC 2014*, pp. 316–320, 2014.
[6] K. Dill, "A Game AI Approach to Autonomous Control of Virtual Characters A Game AI Approach to Autonomous Control of Virtual Characters," *Interservice/Industry Training, Simulation, Educ. Conf.*, no. 11136, pp. 1–11, 2011.
[7] S. Jadon, A. Singhal, and S. Dawn, "Military Simulator -A Case Study of Behaviour Tree and Unity based architecture," *Int. J. Comput. Appl.*, vol. 88, no. 5, pp. 26–29, 2014.
[8] D. Mark and K. Dill, "Improving AI Decision Modeling Through Utility Theory," in *AI Summit GDC 2010*, 2010.
[9] H. Wang and C.-T. Sun, "Game Reward Systems : Gaming Experiences and Social Meanings," in *Proceedings of DiGRA 2011 Conference: Think Design Play*, 2011.
[10] S. Boslaugh and P. A. Watters, *Statistics in a Nutshell: A Desktop Quick Reference*, 1st ed. Sebastopol: O'Reilly Media, 2008.

ISSN: 2714-7975; E-ISSN: 2716-1382