

**NO: 01022052/INF/2021**

**MENINGKATKAN VARIASI TINDAKAN NON-PLAYABLE CHARACTER PADA  
GAME SURVIVAL MENGGUNAKAN METODE MARKOV**

**SKRIPSI**

Digunakan untuk memenuhi persyaratan penyelesaian program S-1  
Program Studi Informatika Fakultas Teknologi Industri  
Universitas Kristen Petra

Oleh :  
Hendra Winata  
NRP : C14170032

**PROGRAM STUDI INFORMATIKA**



**FAKULTAS TEKNOLOGI INDUSTRI  
UNIVERSITAS KRISTEN PETRA  
SURABAYA  
2021**

**LEMBAR PENGESAHAN**

**Skripsi**

**MENINGKATKAN VARIASI TINDAKAN NON-PLAYABLE CHARACTER PADA  
GAME SURVIVAL MENGGUNAKAN METODE MARKOV**

Oleh :

Hendra Winata

C14170032

Diterima Oleh :

Program Studi Informatika Fakultas Teknologi Industri  
Universitas Kristen Petra

Surabaya, 14 Juni 2021

Pembimbing 1

Pembimbing 2

LILIANA, S.T., M.Eng., Ph.D.  
NIP: 03-024

HANS JUWIANTHO, M.Kom.  
NIP : 45-240

Ketua Tim Penguji:

LEO WILLYANTO SANTOSO, M.T.

NIP: 03-023

Ketua Program Studi:

Henry Novianus Palit, Ph.D  
NIP: 14-001

**LEMBAR PERNYATAAN PERSETUJUAN PUBLIKASI KARYA ILMIAH UNTUK KEPENTINGAN  
AKADEMIS**

Sebagai mahasiswa Universitas Kristen Petra, yang bertanda tangan dibawah ini, saya:

Nama : Hendra Winata

NRP : C14170032

Demi mengembangkan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Kristen Petra Hak Bebas Royalti Non-Eksklusif (*Non- Exclusive Royalti-Free Rights*) atas karya ilmiah saya yang berjudul: Meningkatkan Variasi Tindakan Non-Playable Character Pada Game Survival Menggunakan Metode Markov. Dengan Hak Bebas Royalti Non-Eksklusif ini Universitas Kristen Petra berhak menyimpan, mengalih-media format-kan, mengelolanya dalam bentuk pangkalan data (*database*), mendistribusikannya dan menampilkan/mempublikasikannya di internet atau media lain untuk kepentingan akademis tanpa perlu meminta izin dari saya selama tetap mencantumkan nama saya sebagai penulis/pencipta.

Saya bersedia untuk menanggung secara pribadi, tanpa melibatkan pihak Universitas Kristen Petra, segala bentuk tuntutan hukum yang timbul atas pelanggaran Hak Cipta dalam karya ilmiah saya ini.

Demikian surat pernyataan ini saya buat dengan sebenarnya.

Surabaya, 14 juni 2021

Yang menyatakan,

Hendra Winata

## KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa atas berkat dan pimpinan-Nya yang telah diberikan selama pengerjaan skripsi ini, sehingga penulis dapat menyelesaikan skripsi ini dengan baik.

Pada kesempatan ini penulis menyampaikan ucapan terima kasih yang sebesar-besarnya kepada orang-orang yang telah berperan sehingga skripsi ini dapat terselesaikan, antara lain:

1. Henry Novianus Palit, Ph.D., selaku Ketua Program Studi Teknik Informatika dan Sistem Informasi Bisnis Universitas Kristen Petra.
2. Liliana, S.T., M.Eng., Ph.D., selaku dosen pembimbing I, yang telah memberikan arahan, motivasi serta meluangkan waktu selama proses pembuatan skripsi berlangsung.
3. Hans Juwiantho, M.Kom., selaku dosen pembimbing II, yang telah memberikan arahan, motivasi serta meluangkan waktu selama proses pembuatan skripsi berlangsung.
4. Silvia Rostianingsih, M.MT., selaku Koordinator Skripsi Program Teknik Informatika dan Sistem Informasi Bisnis Universitas Kristen Petra.
5. Segenap dosen dan staf pengajar di Program Studi Teknik Informatika Universitas Kristen Petra.
6. Keluarga yang telah banyak memberikan dukungan doa dan motivasi hingga penulis mampu menyelesaikan tugas akhir guna meraih gelar kesarjanaan ini.
7. Teman-teman tercinta yang telah menempuh perjalanan skripsi bersama.
8. Pihak-pihak lain yang telah memberikan bantuan secara langsung maupun tidak langsung dalam pembuatan tugas akhir ini yang tidak dapat disebutkan satu per satu.

Peneliti menyadari bahwa penulisan skripsi ini masih jauh dari sempurna. Oleh karena itu, penulis mengharapkan segala petunjuk, kritik, dan saran yang membangun dari pembaca agar dapat menunjang pengembangan dan perbaikan penulisan.

Akhir kata, penulis mohon maaf apabila ada kekurangan dalam penulisan tugas akhir ini dan penulis dengan senang hati menerima saran dan kritik yang membangun dari pembaca.

Surabaya, 14 Juni 2021

Penulis

## ABSTRAK

Hendra Winata:

Skripsi

Meningkatkan Variasi Tindakan Non-Playable Character Pada Game Survival Menggunakan Metode Markov

Permainan digital atau sering disebut *game* sudah tidak asing untuk didengar pada jaman saat ini. Berkembangnya varian *game* membuat *game* tidak pernah berhenti berkembang terutama pada bagian *Artificial Intelligence*. Setiap *game* memiliki kecerdasan buaatannya tersendiri sehingga banyak variasi yang dihasilkan dan membuat sebuah *game* menjadi unik.

Penelitian ini mencoba untuk membuat sebuah variasi tindakan yang dilakukan oleh *NPC* terhadap pemain. Dalam upaya membuat variasi tersebut, digunakan metode *Markov Chain* untuk membantu pemilihan *state*. Metode *Markov Chain* dikombinasikan dengan *Finite-State Machine* untuk pemilihan *state NPC*.

Berdasarkan hasil pengujian dan kuesioner, 80.4% sangat setuju dan 19.6% setuju bahwa *NPC* yang dihasilkan memiliki variasi tindakan yang banyak. Pada hasil kuesioner juga didapatkan 69.6% sangat tidak realistis dan 30.4% mengatakan bahwa *NPC* tidak realistis atau tidak meniru tingkah laku dari manusia.

Kata kunci:

*Markov Chain, Finite-State Machine, Game, Artificial Intelligence, Survival*

## **ABSTRACT**

Hendra Winata:

Undergraduate Thesis

Improving Non-Playable Character Behaviour Variation in Survival Game using Markov Method

Digital games or often called Video games are common today. The development of game variants makes games never stop improving, especially in the Artificial Intelligence section. Each game has its own artificial intelligence so that many variations are generated and make a game unique.

This research tries to make a variation of the actions taken by NPCs against players. In an effort to make these variations, the Markov Chain method is used to help state selection. Markov Chain method is combined with Finite-State Machine for NPC state selection.

Based on the results of testing and questionnaires, 80.4% strongly agree and 19.6% agree that the resulting NPC has a large variety of actions. The results of the questionnaire also found that 69.6% were very unrealistic and 30.4% said that NPCs were unrealistic or did not imitate human behavior.

Keywords:

Markov Chain, Finite-State Machine, Game, Artificial Intelligence, Survival

## DAFTAR ISI

HALAMAN JUDUL .....	i
LEMBAR PENGESAHAN.....	ii
LEMBAR PERSETUJUAN PUBLIKASI .....	iii
KATA PENGANTAR.....	iv
ABSTRAK .....	v
ABSTRACT .....	vi
DAFTAR ISI .....	vii
DAFTAR GAMBAR.....	ix
DAFTAR TABEL.....	xi
1. PENDAHULUAN .....	1
1.1. Latar Belakang .....	1
1.2. Rumusan Masalah .....	2
1.3. Tujuan Penelitian.....	2
1.4. Ruang Lingkup .....	2
1.5. Metodologi Penelitian.....	2
1.6. Sistematika Penulisan.....	3
2. LANDASAN TEORI .....	4
2.1. Tinjauan Pustaka .....	4
2.1.1 Permainan / Game .....	4
2.1.2 Non-Playable Character (NPC).....	4
2.1.3 Game Survival .....	4
2.1.4 Finite State Machine (FSM) .....	5
2.1.5 Markov Chain .....	5
2.2 Tinjauan Studi.....	6
2.2.1 Penerapan Metode Finite State Machine Pada Game “The Mahasiswa” Guna Membangun Perilaku Non Playable Character (Hernawan, 2018). ....	7
2.2.2 Multi-AI competing and Winning against Human in iterated Rock- Paper-Scissors game (Wang et al., 2020).....	7
2.2.3 Predicting Student Performance in an Educational Game Using a Hidden Markov Model (Tadayon & Pottie, 2020).....	8
3. ANALISIS DAN DESAIN SISTEM .....	10
3.1 Analisa Dan Gameplay.....	10

3.2	Desain Sistem .....	11
3.3	Desain User Interface .....	16
3.4	Desain Karakter NPC .....	17
4.	IMPLEMENTASI SISTEM.....	19
4.1	Melakukan inisialisasi <i>variable</i> input untuk tiap <i>state</i> .....	19
4.2	Mendapatkan <i>Value</i> dari Variabel Input Sekarang .....	20
4.3	Membuat Nilai <i>Current</i> ke dalam bentuk Float pada Range 0-1.....	21
4.4	Melakukan Perkalian Matriks Probabilitas dan Temporary Probability .....	22
4.5	Memilih <i>State</i> Dari Hasil Matriks Probabilitas .....	23
4.6	Melakukan <i>Random Enemy</i> yang Tampil .....	24
4.7	Membuat <i>Constructor</i> .....	26
4.8	<i>Set What to Do Every Time</i> .....	27
5.	PENGUJIAN SISTEM .....	39
5.1	Perangkat Lunak dan Spesifikasi yang dipakai .....	39
5.2	Pengujian Variasi .....	39
5.3	Pengujian <i>Main Menu</i> .....	48
5.4	Pengujian Gameplay.....	48
6.	KESIMPULAN DAN SARAN.....	49
6.1	Kesimpulan.....	49
6.2	Saran.....	49
	DAFTAR PUSTAKA.....	50
	LAMPIRAN .....	52



## DAFTAR GAMBAR

2.1 Contoh Diagram State Sederhana .....	5
3.1 Flowchart Gameplay .....	11
3.2 Flowchart FSM.....	12
3.3 Flowchart Markov Chain .....	13
3.4 Tampilan HUD atau User Interface pada Main Menu .....	16
3.5 Tampilan HUD atau User Interface dari Gameplay .....	16
3.6 Karakter NPC 1 .....	17
3.7 Karakter NPC 2 .....	18
4.1 Inisialisasi Variabel Input Distance Terhadap Variabel Output.....	19
4.2 Inisialisasi Variabel Input Health Terhadap Variabel Output .....	19
4.3 Inisialisasi Variabel Input Speed Terhadap Variabel Output.....	20
4.4 Get Current Distance From Player to NPC .....	20
4.5 Get Current Speed of NPC.....	20
4.6 Get Total of All Variable Current .....	21
4.7 Make Float Probability Distance Variable .....	21
4.8 Make Float Probability Health Variable .....	22
4.9 Make Float Probability Speed Variable .....	22
4.10 Make Result From Matriks Probability and Temporary Probability Each Variable...	23
4.11 Find State .....	23
4.12 Select State.....	24
4.13 Random Mesh, Set Collision, dan Menentukan Anim.....	24
4.14 Set HP Text .....	24
4.15 Set Jenis Enemy Text .....	25
4.16 Attach Gun to Mesh .....	25
4.17 Attach Melee Collision to Mesh .....	26
4.18 Membuat Inisialisasi Matriks Probabilitas dan Memasang Markov Chain .....	26
4.19 Set Gun Instance .....	26
4.20 Select State.....	27
4.21 Set Rotation of HP Text .....	27
4.22 Set Rotation of Jenis Enemy Text.....	27
4.23 Markov Chain Process.....	28
4.24 Idle State .....	28
4.25 Patrol State-Part1.....	28
4.26 Patrol State-Part2.....	29
4.27 Patrol State-Part3.....	29
4.28 Pursue State .....	29
4.29 Attack State .....	30
4.30 Firing Part1 .....	30
4.31 Firing Part2 .....	31
4.32 Begin Play atau Constructor untuk Gun Blueprint .....	31
4.33 Start and Stop Shooting .....	32
4.34 Reloading.....	32

4.35 Fire Bullet Part 1.....	32
4.36 Fire Bullet Part 2.....	33
4.37 Fire Bullet Part 3.....	33
4.38 Fire bullet Part 4.....	33
4.39 Fire Bullet Part 5.....	34
4.40 Fire Bullet Part 6.....	34
4.41 Fire Bullet Part 7.....	34
4.42 Fire Bullet Part 8.....	35
4.43 Fire Bullet Part 9.....	35
4.44 Fire Bullet Part 10.....	35
4.45 Fire Bullet Part 11.....	36
4.46 Fire Bullet Part 12.....	36
4.47 Fire Bullet Part 13.....	36
4.48 Hit Enemy Event With Bullet BP.....	37
4.49 Damage Enemy Event .....	37
4.50 Get Score and Update Total Killed Enemy .....	37
4.51 Damage Player Event .....	37
4.52 Escape / Get Away.....	38
4.53 Suiciding Part 1.....	38
4.54 Suiciding Part 2.....	38
4.55 Reloading.....	38
5.1 Awal Game Berjalan Memilih State Patrol.....	39
5.2 <i>Current Distance from Player</i> .....	40
5.3 <i>Current Speed of NPC</i> .....	40
5.4 Current Health of NPC.....	40
5.5 Hasil perkalian Current Variabel dengan Matriks Probabilitas .....	41
5.6 <i>State Idle</i> .....	41
5.7 <i>Current Distance from Player</i> .....	42
5.8 <i>Current Speed of NPC</i> .....	42
5.9 <i>Current Health of NPC</i> .....	42
5.10 Hasil perkalian Current Variabel dengan Matriks Probabilitas .....	43
5.11 Awal Permainan, NPC Memilih State Idle .....	43
5.12 <i>Current Distance from Player</i> .....	44
5.13 <i>Current Speed of NPC</i> .....	44
5.14 Current Health of NPC.....	44
5.15 Hasil perkalian Current Variabel dengan Matriks Probabilitas .....	45
5.16 State Pursue .....	45
5.17 <i>Current Distance from Player</i> .....	46
5.18 <i>Current Speed of NPC</i> .....	46
5.19 <i>Current Health of NPC</i> .....	46
5.20 Hasil perkalian Current Variabel dengan Matriks Probabilitas .....	47
5.21 Hasil Responden Terhadap Variasi NPC .....	47
5.22 Hasil Responden Terhadap Tingkat Realistis NPC.....	47
5.23 State Yang Terpilih dari Proses Markov Chain .....	48

## DAFTAR TABEL

Table 2.1 Tabel Research Gap .....	8
Table 3.1 Nilai Variabel .....	14
Table 3.2 Matriks Probabilitas Transisi Parameter .....	14
Table 3.3 Matriks Probabilitas Transisi Parameter .....	14

## 1. PENDAHULUAN

### 1.1. Latar Belakang

*Game Survival Shooter* adalah *game* bertema survival dimana pemain akan bertahan hidup dengan cara mengalahkan semua musuh yang ada agar mencapai suatu tujuan dari sebuah misi. Dalam sebuah *game*, terdapat *NPC* yaitu karakter yang dikendalikan oleh komputer dan biasanya berupa lawan dari pemain (Kopel & Hajas, 2018). Perilaku *NPC* dipengaruhi oleh *AI* yang sangat menentukan tingkat kesulitan karena semakin variatif *NPC* semakin susah juga untuk ditebak perilaku yang dihasilkan (Kopel & Hajas, 2018).

Pemberian fleksibilitas terhadap *NPC* sangat diperlukan karena dengan ragam tindakan *NPC* akan membuat permainan semakin hidup dan tidak monoton dimana pemain dapat menebak dengan mudah pilihan tindakan yang akan diambil oleh *NPC* berikutnya (Zhu, 2019). Namun, tidak terlalu banyak permainan yang dibuat oleh developer ternama yang meningkatkan sistem *Artificial Intelligence* dari *NPC* mereka.

*Finite State Machine (FSM)* adalah sebuah metodologi perancangan sistem kontrol yang menggambarkan tingkah laku atau prinsip kerja sistem dengan menggunakan *State* (Keadaan), *Event* (kejadian), dan *Action* (Aksi) (Hidayat et al., 2019). *NPC* yang menggunakan *FSM* dinyatakan layak untuk digunakan dengan interpretasi baik (Hidayat et al., 2019). *FSM* pada *game* berguna untuk menentukan berbagai macam respon *NPC* terhadap pemain di dalam sebuah *game* berdasarkan interaksi yang dilakukan, hal ini disebabkan karena *FSM* dapat digunakan untuk gambaran awal, mendesain, dan menentukan respon perilaku yang dilakukan terhadap perubahan kondisi (Hidayat et al., 2019). *Markov Chain* adalah model universal untuk memprediksi *state* dari sistem diskrit (Ge et al., 2018). *Markov Chain* umumnya digunakan untuk pemodelan *state transition* dari sistem stokastik yang kompleks (Zhou et al., 2018). *Markov Chain* dibuat dengan menggunakan matrik transisi probabilitas (Ge et al., 2018). Pada penelitian mengenai Multi-AI competing and winning against humans in iterated Rock-Paper-Scissors game, diterapkan *Markov Chain* untuk meningkatkan *AI* dalam menghitung kemungkinan terbaik selanjutnya (Wang, 2020). Dalam penelitian tersebut, terbukti dengan *Markov Chain* dapat membuat *AI* mengalahkan kebanyakan orang dimana setiap orang memiliki pola tersendiri, namun terbatas pada permainan bertipe strategi atau *turn-based game*. Pada penelitian Predicting Student Performance pada Educational Game, diterapkan *Hidden Markov Model (HMM)* dimana pada model tersebut dilakukan untuk memberikan level yang sesuai terhadap siswa. *HMM* merupakan bagian dari *Markov Chain* namun tidak menggunakan

probabilitas. Penelitian tersebut berhasil melakukan generate level yang sesuai dan menghasilkan evaluasi berkala dari para siswa.

Pada penelitian skripsi sebelumnya, metode *FSM* hanya akan menghasilkan *NPC* yang bertindak secara monoton atau general, pada penelitian *Markov Chain* pada permainan bertipe strategi atau *turn-based game* sebelumnya bisa menghasilkan *AI* yang cenderung lebih bervariasi sehingga jika bisa diterapkan pada game yang memiliki *fast pace* seperti *game survival*, maka akan membuat game lebih unik dan membuat pemain ingin mencoba memenangkan permainan dengan mengalahkan *NPC* yang tidak terduga. Pada penelitian *ini*, pengembangan yang akan dilakukan adalah menggabungkan *Markov Chain* yang menggunakan probabilitas ke dalam *game survival* dengan menggunakan 3 variabel yaitu kesehatan, jarak, dan kecepatan yang bersifat universal sebagai variabel dasar untuk game 3D survival (K Fathoni, 2020) dan akan dibandingkan dengan *AI* dari *NPC* yang hanya menggunakan *FSM*.

#### **1.2. Rumusan Masalah**

Masalah yang ada dalam implementasi:

- Seberapa banyak variasi pemilihan tindakan yang bisa dihasilkan dengan *AI* yang menggunakan *Markov Chain*?
- Seberapa nyata pilihan tindakan yang dipilih *AI* berdasar kondisi yang dihadapi?

#### **1.3. Tujuan Penelitian**

Penelitian ini bertujuan menghasilkan *NPC* yang lebih variatif dengan umpan balik dari player.

#### **1.4. Ruang Lingkup**

Ruang lingkup dibatasi pada:

- Faktor penentu tindakan dibatasi hanya dengan nyawa, jarak, dan kecepatan.
- State *NPC* terdiri dari state idle, patrol, mengejar, menyerang, suicide, menembak, dan melarikan diri. *Markov Chain* diterapkan untuk menghitung perpindahan state.
- *Game* dibuat dengan Unreal Engine 4 versi 4.26.
- *FSM* hanya akan digunakan sebagai perbandingan dengan penelitian ini yang menggunakan *Markov Chain*.

#### **1.5. Metodologi Penelitian**

Langkah-langkah dalam pengerjaan skripsi:

- Studi literatur tentang *Markov Chain*, Teori *Unreal Engine 4*, dan *FSM*.
- Perencanaan dan Pembuatan *Markov Chain* pada *Game*.
- Penentuan variabel yang digunakan dalam *Markov Chain*.

- Pengujian dan Analisis *Markov Chain* pada *Game*.
- Pengambilan Kesimpulan.
- Pembuatan Laporan.

#### **1.6. Sistematika Penulisan**

Adapun sistematika penulisan yang digunakan untuk Menyusun skripsi ini adalah sebagai berikut:

##### **BAB I : PENDAHULUAN**

Bab ini membahas latar belakang permasalahan, perumusan masalah, tujuan skripsi, ruang lingkup, metode penelitian yang digunakan, dan relevansi skripsi ini.

##### **BAB II : LANDASAN TEORI**

Bab ini membahas mengenai teori-teori yang menjadi landasan dalam kegiatan penelitian ini, seperti penjelasan mengenai *Game*, *NPC*, *Game Engine*, teori *FSM* dan *Markov Chain*.

##### **BAB III : ANALISIS DAN DESAIN SISTEM**

Bab ini membahas analisis dari *gameplay* yang akan dibuat beserta penjelasan penggunaan *Markov Chain* dan *FSM*.

##### **BAB IV : IMPLEMENTASI SISTEM**

Bab ini berisikan kebutuhan implementasi metode dan membahas secara detail hasil pembuatan *game*.

##### **BAB V : PENGUJIAN DAN EVALUASI SISTEM**

Bab ini berisikan pembahasan kegiatan pengujian *game* dengan menggunakan metode *Markov Chain* dan evaluasi bug yang ada pada *game*.

##### **BAB VI : KESIMPULAN DAN SARAN**

Bab ini membahas kesimpulan dan saran yang berkaitan dengan metode pada *game* yang telah dibuat.

## 2. LANDASAN TEORI

### 2.1. Tinjauan Pustaka

#### 2.1.1 Permainan / Game

Dalam Bahasa Indonesia “*Game*” berarti permainan, dimana terdapat sebuah environment yang berisi keputusan dari aksi pemain serta target tertentu yang ingin dicapai. *Game* sendiri merupakan aktivitas yang *immersive*, *voluntary*, dan *fun* untuk mencapai target tertentu berdasar peraturan yang sudah disediakan (Saprudin et al., 2019). Pada konteks *e-learning*, sebuah game didefinisikan sebagai *online environment* yang kompetitif dan menantang untuk menuju sebuah *goal*, mempunyai sekumpulan aturan dan konteks (Saprudin et al., 2019). *Digital Game* terbukti sebagai media yang dapat meningkatkan semangat siswa untuk belajar, selain itu *Game* juga sangat erat dengan kompetisi dimana semakin banyak partisipan maka akan meningkatkan tingkat kesenangan dan tantangan (Saprudin et al., 2019).

#### 2.1.2 Non-Playable Character (NPC)

*NPC* yaitu karakter yang dikendalikan oleh komputer dan biasanya berupa lawan dari pemain (Kopel & Hajas, 2018). *Non-Playable Character* juga bisa disebut *autonomus agent* yang mewakili tokoh dalam cerita atau *game* dan memiliki kemampuan untuk improvisasi tindakan mereka.

*NPC* pada sebuah *game* memiliki peranan sebagai pelengkap agar *game* tersebut tidak membosankan. Umumnya, *NPC* ditempatkan sebagai musuh dari karakter yang dimainkan pemain, namun *NPC* juga bisa menjadi penolong dari karakter pemain. *NPC* pada *game survival* memiliki variabel dasar yang sering digunakan yaitu jarak, kesehatan, dan kecepatan (K Fathoni, 2020).

#### 2.1.3 Game Survival

*Survival Game* merupakan *sub-genre* dari *action video games* yang berlatarkan tempat berbahaya, mencekam, serta bersifat *open-world*. *Game survival* merupakan permainan bertahan hidup selama mungkin dengan *resource* yang tersedia (Hassan et al., 2018). Pemain umumnya akan mulai dengan *resource* yang terbatas dan harus bertahan selama mungkin. Banyak *survival game* yang berbasis *random* atau *procedural generated environments*. Pada umumnya, tidak ada batasan akhir dari *survival game* dan sering dikaitkan dengan *genre horror*.

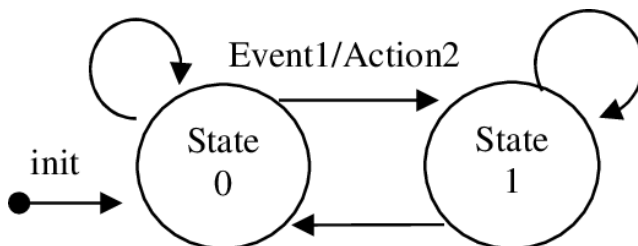
*Survival Game* umumnya dimainkan secara individu, namun saat ini sudah banyak yang mulai membuat dengan tipe *multiplayer* dengan world yang *persistent*. Target yang umumnya

bisa dicapai adalah waktu bermain dari pemain atau skor dari membunuh musuh sebanyak mungkin.

#### 2.1.4 Finite State Machine (FSM)

*Finite State Machine (FSM)* adalah sebuah metode pembuatan sistem kontrol yang menggambarkan tingkah laku berdasar 3 hal yaitu *state* (kondisi), *event* (kejadian), *action* (aksi/tindakan) (Hidayat et al., 2019). *State machine* dikenal sebagai teknik untuk pemodelan fenomena atau kondisi berbasis *event*, termasuk penguraiannya, serta desain *interface*. *Finite State Machine (FSM)* atau juga disebut sebagai *Finite State Automata*, dianggap sebagai teknik yang secara luas dipergunakan dalam merancang *AI* dalam *game* (Tirtana & Pumpungan, 1945; Yulsilviana & Ekawati, 2019). Penerapan *FSM* pada *game* berguna untuk menentukan berbagai macam respon *NPC* terhadap pemain di dalam sebuah *game* berdasarkan interaksi yang dilakukan, hal ini karena *FSM* dapat digunakan untuk gambaran awal, mendesain dan menentukan respon perilaku yang dilakukan terhadap perubahan kondisi (Hidayat et al., 2019).

Pada satu saat yang signifikan, sistem akan berada pada sebuah *state* yang aktif. Terjadinya transisi adalah ketika terdapat tindakan atau aksi yang menyebabkan sistem melakukan pergantian *state* sebagai bentuk respon, berupa aksi sederhana maupun aksi yang bersifat kompleks.



Gambar 2.1 Contoh Diagram State Sederhana

Diagram pada Gambar 2.1 adalah contoh sederhana dari *FSM* dengan dua buah *state*. *FSM* berperan juga dalam menentukan aksi 3D animasi objek bagaimana karakter bereaksi atau memutuskan tindakan berdasar keadaan atau kejadian tertentu (Hidayat et al., 2019).

#### 2.1.5 Markov Chain

*Markov Chain* adalah suatu metode universal untuk memprediksi *state* dari sistem diskrit (Ge et al., 2018). *Markov Chain* mempelajari sifat suatu variabel pada masa sekarang yang didasarkan pada sifat-sifatnya di masa lalu dalam usaha memprediksi sifat-sifat variabel tersebut di masa yang akan datang (Zou et al., 2018). Dalam analisis *Markov* yang dihasilkan adalah suatu informasi probabilitas yang dapat digunakan untuk membantu pembuatan keputusan. Analisis



Markov merupakan suatu bentuk khusus dari model probabilitas yang lebih umum dikenal sebagai proses stokastik (*Stochastic process*).

Proses stokastik merupakan proses yang dapat digunakan untuk memodelkan fenomena yang mengandung unsur ketidakpastian (S. Utami 1 , I W. Mangku 2, 2018). Konsep dasar dari analisis Markov adalah *state* dari sistem atau *state* transisi, sifat dari proses ini adalah apabila diketahui proses berada pada suatu keadaan tertentu, maka peluang berkembangnya proses di masa mendatang hanya bergantung pada keadaan saat ini dan tidak tergantung pada keadaan sebelumnya, dengan kata lain *Markov Chain* adalah rangkaian proses kejadian dimana peluang bersyarat kejadian yang akan datang bergantung pada keadaan sekarang (Kasse et al., 2020). *Markov Chain* memiliki syarat untuk dapat dijalankan yaitu memiliki state sekarang atau sebelumnya, jumlah probabilitas matrix nya adalah 1 bersifat steady state.

Pada *survival game*, metode Markov dapat diterapkan untuk membentuk syarat perpindahan dari *state* pada *finite-state machine*. Dalam membuat *Markov Chain*, umumnya dibentuk matriks dengan ukuran  $m \times n$  yang akan dikalikan dengan sebuah variable. Rumus dari *Markov Chain* dapat dituliskan sebagai berikut.

$$\Pr( X_{n+1} = x \mid X_1 = x_1, X_2 = x_2, \dots, X_n = x_n ) = \Pr( X_{n+1} = x \mid X_n = x_n ) \quad (2.1)$$

Dimana:

Pr : Probabilitas

$X_n$  : Random Variabel

$x$  : Kondisi saat ini

Rumus tersebut berlaku jika kondisi probabilitas telah ditentukan dan memenuhi syarat

$$\Pr( X_1 = x_1, \dots, X_n = x_n ) > 0. \quad (2.2)$$

Angka yang mungkin dari  $X_1$  membentuk *countable set* yang disebut *state space*. Jika *state space* bersifat *finite* maka setiap baris pada  $p$  jika dijumlahkan akan menjadi 1 dan setiap elemen tidak negatif sehingga probabilitas transisi dapat ditampilkan sebagai *matrix* yang disebut *transition matrix* dengan setiap elemen yang dapat didefinisikan dengan rumus berikut.

$$P_{ij} = \Pr(X_{n+1} = j \mid X_n = i) \quad (2.3)$$

Dimana :

$P_{ij}$  : Probabilitas pada element (i,j)

$j$  : kolom matriks

$i$  : baris matriks

## 2.2 Tinjauan Studi

Bagian ini akan menjelaskan mengenai penelitian-penelitian yang berhubungan, dimana menjadi referensi dalam penelitian kali ini. Tabel dapat dilihat pada Tabel 2.1.

### 2.2.1 Penerapan Metode Finite State Machine Pada Game “The Mahasiswa” Guna Membangun Perilaku Non Playable Character (Hernawan, 2018).

**Masalah :** Menentukan tingkat kesulitan pertanyaan berdasarkan jumlah jawaban benar yang berhasil dijawab oleh pemain. Pertanyaan memiliki variasi kesulitan yang belum tentu memiliki angka kesulitan yang sama.

**Metode :** Pada penelitian Septian Rico Hernawan, dilakukan penggunaan *FSM* pada *NPC* untuk menentukan tingkat kesulitan pertanyaan berdasarkan jumlah jawaban benar yang berhasil dijawab oleh pemain. Penggunaan *FSM* juga diterapkan pada banyak hal seperti ketika terjadi interaksi baik antara pemain dan *NPC*, maka akan terjadi sesuatu, juga sebaliknya. Metode *FSM* disini lebih diterapkan pada saat menampilkan dan menjawab pertanyaan, juga pada musuh yaitu kera yang mengejar pemain.

**Hasil :** Pada penelitian ini, didapat hasil yang lebih baik dengan menggunakan *FSM* untuk memberikan pertanyaan yang sesuai berdasar tingkat kesulitan dan total jawaban benar.

### 2.2.2 Multi-AI competing and Winning against Human in iterated Rock-Paper-Scissors game (Wang et al., 2020).

**Masalah :** Pada permainan gunting,batu,kertas, AI harus menebak pilihan pemain sehingga AI dapat memberikan balasan yang sesuai untuk memenangkan permainan.

**Metode :** Pada penelitian Lei Wang, digunakan *Markov Chain* untuk menghitung probabilitas terbaik dari pilihan pemain. Eksperimen yang dilakukan adalah menggabungkan beberapa single model AI yang didasarkan pada 5 atau 10 round pertandingan terbaik, kemudian dilakukan kalkulasi dengan rumus yang dibuat dan diambil 1 dari hasil yang paling baik untuk digunakan pada round berikutnya. Perhitungan berdasar gunting, batu, kertas menghasilkan peluang masing-masing  $1/3$ . Percobaan dilakukan pada 52 manusia yang dilakukan pada 300 rounds dan dengan penggabungan multi-AI tersebut mampu memenangkan banyak pertandingan terhadap manusia, tapi bukan semuanya.

**Hasil :** Dengan menggabungkan AI dan single Markov Chain model, Lei Wang dapat membuat AI dengan kemampuan prediksi terbaik terhadap human behaviour.

**2.2.3 Predicting Student Performance in an Educational Game Using a Hidden Markov Model** (Tadayon & Pottie, 2020).

**Masalah :** Menentukan level secara dinamis untuk mengetahui seberapa paham siswa dalam menyelesaikan masalah pada Educational Game.

**Metode :** Pada penelitian Manie dan Greg, digunakan Hidden Markov Model (Markov Chain tanpa probabilitas) untuk memprediksi level secara berkala berdasar hasil yang didapat siswa pada tiap level sebelumnya. Pada penelitian tersebut, dilakukan dengan cara membagi 2 kelas berdasarkan score yang dihasilkan. Analisis dilakukan dengan menggunakan berbagai macam fitur pada game sebagai observasi.

**Hasil :** Metode HMM ini dapat menjadi evaluasi dinamis dari kemampuan siswa menyelesaikan level untuk dilakukan secara berkala sehingga level yang diberikan sesuai.

Table 2.1

Tabel Research Gap

Penelitian	Nama Peneliti	Tahun	Masalah	Metode	Hasil
<i>Penerapan Metode Finite State Machine Pada Game “The Mahasiswa” Guna Membangun Perilaku Non Playable Character</i>	Septian Rico Hermawan	2018	Menentukan tingkat kesulitan pertanyaan berdasar total jawaban benar pemain	<i>Finite State Machine</i>	<i>NPC</i> dapat memberikan pertanyaan sesuai tingkat kesulitan yang dicapai
Multi-AI Competing And Winning Against	Lei Wang	2020	Menentukan berapa AI model yang perlu	<i>Markov Chain</i>	Membuat AI dengan menggabungkan beberapa model

Human In Iterated Rock-Paper-Scissors Game			digabung untuk dapat mengalahkan pemain		sehingga menghasilkan prediksi terbaik untuk mengalahkan pemain
Predicting Student Performance In An Educational Game Using A Hidden Markov Model	Manie Tadayon, Greg Pottie	2019	Menentukan Level secara dinamis pada Educational Game	<i>Hidden Markov Model</i>	Membuat AI bisa menghasilkan level sesuai kemampuan siswa

Pada penelitian ini, artikel yang menjadi acuan adalah paper milik Lei Wang dimana penelitiannya berfokus pada perhitungan peluang dengan *Markov Chain*, namun hanya diterapkan sebatas perhitungan pada permainan catur sederhana tanpa menggunakan *FSM*. Penelitian ini dibuat dengan target 3D game yang lebih kompleks dan menggunakan *FSM* untuk menentukan pergerakan *NPC*.

Perbedaan dengan penelitian sebelumnya :

Metode yang akan digunakan dalam penelitian ini adalah Markov Chain yang akan digabungkan dengan *FSM*. Proses kombinasi ini menghasilkan beberapa kelebihan dari penelitian sebelumnya seperti *NPC* yang lebih variatif terhadap kondisi pemain serta memberikan tindakan sebagai responnya.

### 3. ANALISIS DAN DESAIN SISTEM

#### 3.1 Analisa Dan Gameplay

Pada game bertema *survival*, terdapat beberapa hal yang perlu diperhatikan beberapa hal, beberapa contohnya adalah variabel kesehatan, kecepatan, dan jarak. Variabel tersebut merupakan variabel universal yang ada pada game survival dan dapat dimanfaatkan dalam banyak hal seperti Path Finding, AI, dan masih banyak lagi. Variabel tersebut bermanfaat untuk menjadi nilai input yang akan diolah pada proses markov chain. Pada game yang akan dibuat, FSM dibentuk dengan menggunakan beberapa state yang nanti akan berubah mengikuti variabel jarak, kesehatan, kecepatan yang diproses oleh markov chain. State yang akan ada pada FSM adalah idle, patrol, mengejar, menyerang, suicide, menembak, dan melarikan diri.

Markov Chain memiliki fungsi untuk memperkirakan perubahan-perubahan yang kemungkinan akan terjadi di waktu yang akan datang. Terdapat beberapa syarat yang harus dipenuhi dalam proses Markov Chain yaitu jumlah probabilitas kejadian awal harus selalu 1, probabilitas transisi harus konstan atau tidak berubah sepanjang waktu, dan kondisi merupakan kondisi yang independent sepanjang waktu. Markov chain akan digunakan untuk membantu dalam pemilihan state pada FSM yang ditentukan berdasar variabel jarak, kesehatan, dan kecepatan NPC. Markov Chain akan melakukan perhitungan syarat berpindahnya sebuah state ke state yang lain dalam FSM.

### 3.2 Desain Sistem

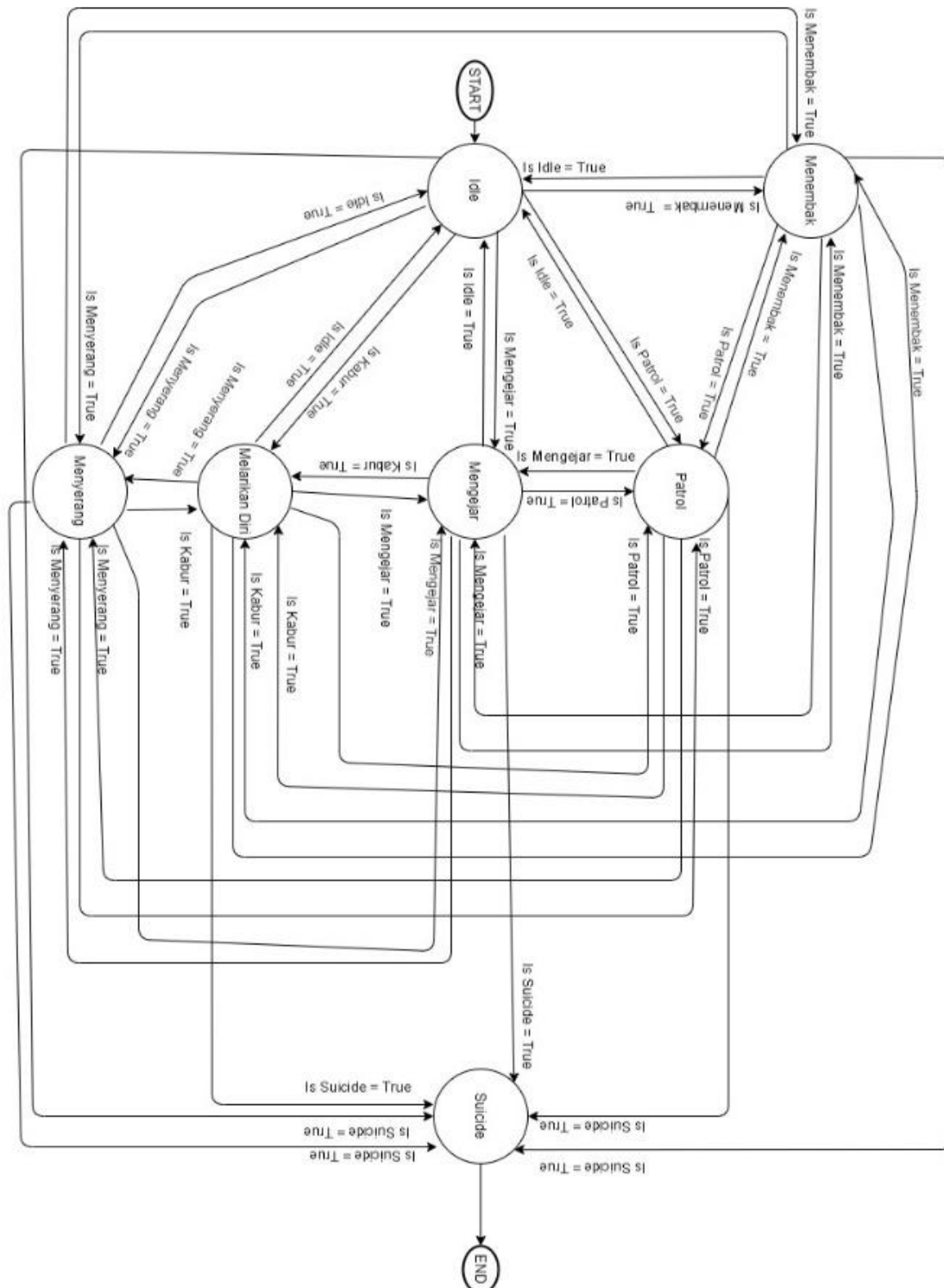
Flowchart Gameplay sebagai berikut.



Gambar 3.1 Flowchart Gameplay

Permainan akan dimulai seperti pada Gambar 3.1 dengan pemain yang akan di spawn di sebuah desa kecil yang berisi beberapa NPC, dimana pemain akan ditugaskan untuk membunuh semua NPC yang ada dan bertahan hidup. NPC akan memiliki variasi aksi tiap kali permainan dimainkan, misalnya NPC A akan idle, lalu mengejar, kemudian menyerang pemain pada percobaan pertama, kemudian pada percobaan kedua, NPC A akan patrol dan menembak saat melihat pemain. Pemain akan berusaha hidup selama mungkin dan mengumpulkan skor dari membunuh NPC.

Flowchart State FSM sebagai berikut.

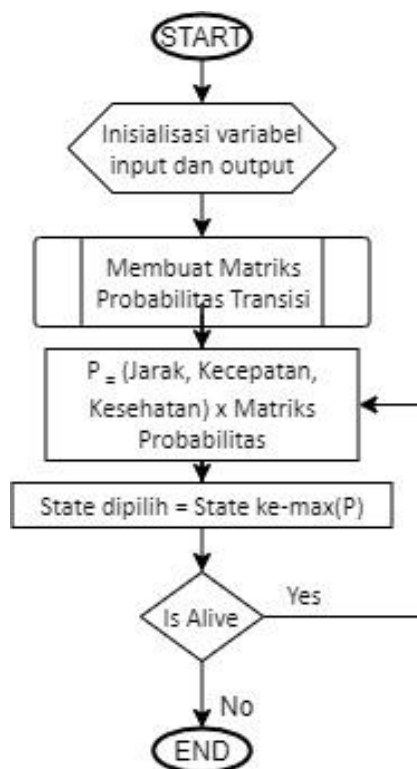


Gambar 3.2 Flowchart FSM

Pada Gambar 3.2 terdapat 7 state yang dibuat yaitu patrol, mengejar, menyerang, idle, suicide, menembak, dan melarikan diri. Tiap state memiliki kondisi atau syarat yang diperlukan

untuk masuk ke dalam state tersebut. Dalam hal ini syarat yang dibutuhkan berupa sebuah kondisi Boolean yang bernilai True / False. Proses Markov Chain akan menentukan nilai dari variable Boolean tersebut. Pada *state Idle*, maka *NPC* akan berdiam diri, kemudian jika memasuki *state* mengejar, *NPC* akan mengejar pemain. Jika *Is Attack* bernilai *true*, *NPC* akan berusaha menyerang pemain, jika *Is Kabur* bernilai *true*, maka *NPC* akan kabur menjauhi pemain. *NPC* akan melakukan patrol saat *Is Patrol* bernilai *true* dan *NPC* akan menembak pemain saat *Is Firing* bernilai *true*. *NPC* hanya akan memasuki *state suicide* saat *Is Suicide* bernilai *true* dan akan dilakukan *spawn NPC* baru.

Flowchart Markov Chain sebagai berikut.



Gambar 3.3 Flowchart Markov Chain

Pada Gambar 3.3, pertama kali akan dilakukan inisialisasi pada variabel input dan output. Variabel input berupa kondisi pemain yaitu jarak NPC terhadap pemain, kesehatan atau darah dari NPC, dan kecepatan NPC saat ini. Sedangkan variabel output berupa state FSM yang dihasilkan oleh NPC. Berikut contoh deklarasi variabel input dan output. Hal ini bisa dilihat pada Tabel 3.1 dan Tabel 3.2.



Table 3.1

Nilai Variabel

Variabel Output						Jumlah
Patrol	Kejar	Serang	Menembak	Melarikan diri	Suicide	
30	25	15	20	5	5	<b>100</b>
40	30	10	10	8	2	<b>100</b>
40	25	10	15	3	7	<b>100</b>

Table 3.2

Matriks Probabilitas Transisi Parameter

Variabel Input	Variabel Output					
	Patrol	Kejar	Serang	Menembak	Melarikan Diri	Suicide
Jarak	$30/100 = 0.3$	$25/100 = 0.25$	$15/100 = 0.15$	$20/100 = 0.2$	$5/100 = 0.05$	$5/100 = 0.05$
Kesehatan	$40/100 = 0.4$	$30/100 = 0.3$	$10/100 = 0.1$	$10/100 = 0.1$	$8/100 = 0.08$	$2/100 = 0.02$
Kecepatan	$40/100 = 0.4$	$25/100 = 0.25$	$10/100 = 0.1$	$15/100 = 0.15$	$3/100 = 0.03$	$7/100 = 0.07$

Berikut hasil dari penyederhanaan tabel diatas.

Table 3.3

Matriks Probabilitas Transisi Parameter

Variabel Input	Variabel Output					
	Patrol	Kejar	Serang	Menembak	Melarikan Diri	Suicide
Jarak	0.3	0.25	0.15	0.2	0.05	0.05
Kesehatan	0.4	0.3	0.1	0.1	0.08	0.02
Kecepatan	0.4	0.25	0.1	0.15	0.03	0.07

Berdasar tabel probabilitas transisi pada Tabel 3.3, maka nilai diatas merupakan nilai maksimal pada masing-masing sub variabel. Nilai terbesar merupakan kondisi pemain dimana

jarak masih jauh, kesehatan yang prima, dan kecepatan tinggi sehingga aksi dari *NPC* adalah patrol. Sedangkan nilai terkecil merupakan kondisi dimana pemain dekat dengan musuh, lemah, dan berjalan lambat, maka aksi dari *NPC* adalah suicide.

Nilai diatas merupakan matriks probabilitas transisi , sehingga dapat dituliskan sebagai berikut:

$$\text{Matriks Probabilitas Transisi} = \begin{bmatrix} 0.3 & 0.25 & 0.15 & 0.2 & 0.05 & 0.05 \\ 0.4 & 0.3 & 0.1 & 0.1 & 0.08 & 0.02 \\ 0.4 & 0.25 & 0.1 & 0.15 & 0.03 & 0.07 \end{bmatrix}$$

Pada prakteknya, matriks probabilitas transisi digunakan dalam rumus untuk menghitung kemungkinan yang terjadi di masa mendatang ketika pemain dalam kondisi saat ini. Sehingga dapat dirumuskan sebagai berikut:

$$[\text{Npatrol Nkejar Nserang Nmenembak Nmelarikandiri Nsuicide}] =$$

$$[\text{Jr Ks Kc}] \times \text{Matriks Probabilitas Transisi}$$

Keterangan:

Jr : Jarak

Ks : Kesehatan

Kc : Kecepatan

Contoh dari perhitungan diatas adalah sebagai berikut:

Diketahui pemain dengan kondisi sebagai berikut:

Jarak = 70, Kesehatan = 50, Kecepatan = 30

Maka akan didapat nilai variabelnya

$$\text{Jr} = \frac{70}{150} = 0.467$$

$$\text{Ks} = \frac{50}{150} = 0.333$$

$$\text{Kc} = \frac{30}{150} = 0.2$$

$$\begin{aligned} [\text{Jr Ks Kc}] &= [0.467 \ 0.333 \ 0.2] \quad \times \quad \begin{bmatrix} 0.3 & 0.25 & 0.15 & 0.2 & 0.05 & 0.05 \\ 0.4 & 0.3 & 0.1 & 0.1 & 0.08 & 0.02 \\ 0.4 & 0.25 & 0.1 & 0.15 & 0.03 & 0.07 \end{bmatrix} \\ &= [0.3533 \ 0.26665 \ 0.12335 \ 0.1567 \ 0.05599 \ 0.04401] \end{aligned}$$

Dari hasil perhitungan tersebut, maka didapatkan nilai probabilitas terbesar ada pada Patrol dengan nilai 0.3533. Maka *NPC* akan melakukan patrol.

### 3.3 Desain User Interface

Tampilan HUD atau User Interface dari Main Menu



Gambar 3.4 Tampilan HUD atau User Interface pada Main Menu

Tampilan Main Menu pada lobby sebelum permainan dimulai terlihat seperti pada Gambar 3.4. Terdapat 2 tombol utama pada sebelah kiri yaitu Campaign untuk memulai permainan dan tombol Exit untuk keluar dari permainan. Background menggunakan gambar statis jpg.

Tampilan HUD atau User Interface dari Gameplay



Gambar 3.5 Tampilan HUD atau User Interface dari Gameplay

Tampilan HUD atau User Interface terlihat pada Gambar 3.5 saat pemain berada di dalam game terdiri dari jumlah enemy atau NPC yang telah dibunuh saat bermain yang diletakkan di kiri atas agar memudahkan pemain melihat objective. Skor pada kanan atas didapat dari membunuh NPC dan merupakan nilai yang dikumpulkan sebanyak mungkin hingga akhir permainan. Skor diletakkan di kanan atas karena pada bagian atas umumnya berupa tempat untuk objective atau pencapaian. Crosshair terdapat di tengah layar untuk membantu pemain menembak NPC, crosshair merupakan sebuah titik yang membantu mengarahkan pemain kemana peluru akan mengarah. HP merupakan darah pemain dan ammo merupakan peluru yang dimiliki oleh pemain. Dua status ini diletakkan di bawah karena ini merupakan informasi general yang sering dilihat oleh pemain.

### 3.4 Desain Karakter NPC

Karakter pertama memiliki desain manusia dengan teknologi *sci-fi*, sedangkan karakter kedua memiliki desain seperti *tantara swat*. NPC ini akan mendapat *state* berdasar hasil dari perhitungan *Markov Chain* misalnya patrol dimana NPC akan melakukan patrol saat awal permainan. Kemudian, akan mengikuti hasil dari perhitungan *Markov Chain*. Desain karakter terlihat seperti pada Gambar 3.6 dan Gambar 3.7.



Gambar 3.6 Karakter NPC 1



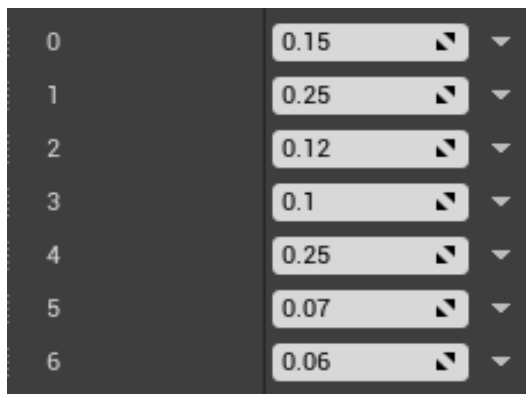
Gambar 3.7 Karakter *NPC* 2

## 4. IMPLEMENTASI SISTEM

Pada bab ini merupakan implementasi dari teori-teori dan desain sistem yang telah dibahas pada bab-bab sebelumnya.

### 4.1 Melakukan inisialisasi *variable* input untuk tiap *state*

Pada tahap ini, langkah awal yang dilakukan adalah melakukan inisialisasi variabel input pada tiap variabel output dengan nilai yang telah di set secara manual. Berikut contoh set untuk variabel input distance seperti pada Gambar 4.1, variable input health seperti pada Gambar 4.2, variabel input speed seperti pada Gambar 4.3.



0	0.15	↗	▼
1	0.25	↗	▼
2	0.12	↗	▼
3	0.1	↗	▼
4	0.25	↗	▼
5	0.07	↗	▼
6	0.06	↗	▼

Gambar 4.1 Inisialisasi Variabel Input Distance Terhadap Variabel Output



0	0.4	↗	▼
1	0.2	↗	▼
2	0.1	↗	▼
3	0.08	↗	▼
4	0.12	↗	▼
5	0.06	↗	▼
6	0.04	↗	▼

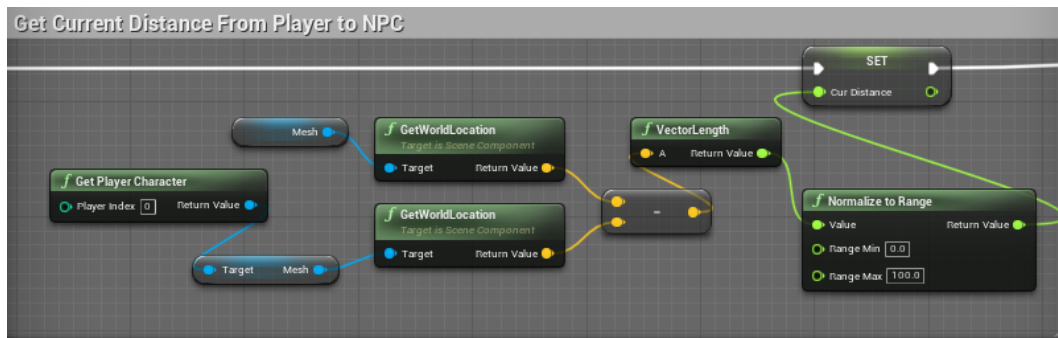
Gambar 4.2 Inisialisasi Variabel Input Health Terhadap Variabel Output

0	0.1
1	0.2
2	0.3
3	0.15
4	0.1
5	0.05
6	0.1

Gambar 4.3 Inisialisasi Variabel Input Speed Terhadap Variabel Output

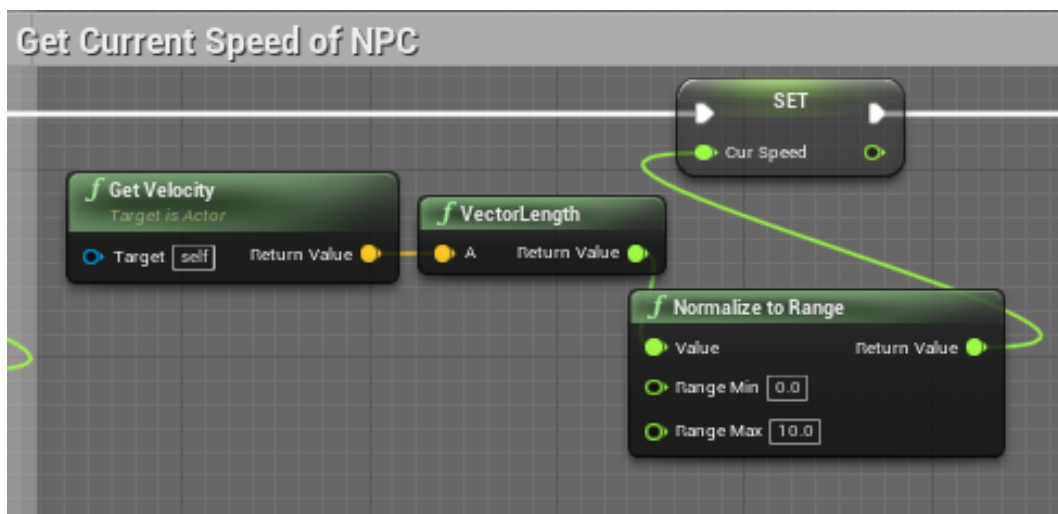
#### 4.2 Mendapatkan *Value* dari Variabel Input Sekarang

Mendapatkan *distance* saat ini dari *Player* ke *NPC* yang kemudian di masukkan dalam *variable* *cur distance* seperti Gambar 4.4.



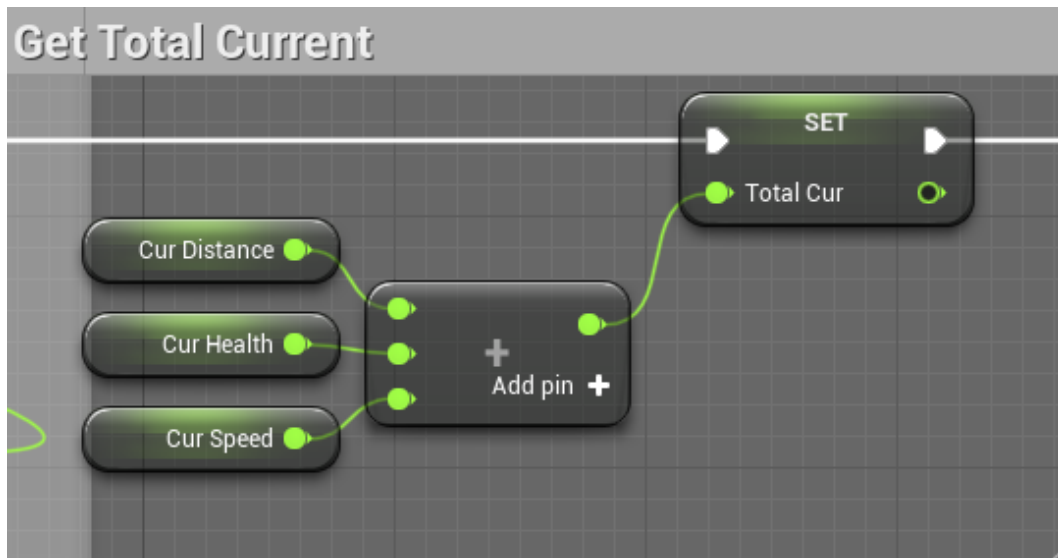
Gambar 4.4 Get Current Distance From Player to NPC

Mendapatkan *speed* dari *NPC* saat ini dan di normalisasi agar angka berada di range 0-100 sesuai batas *random* pada matriks probabilitas seperti Gambar 4.5.



Gambar 4.5 Get Current Speed of NPC

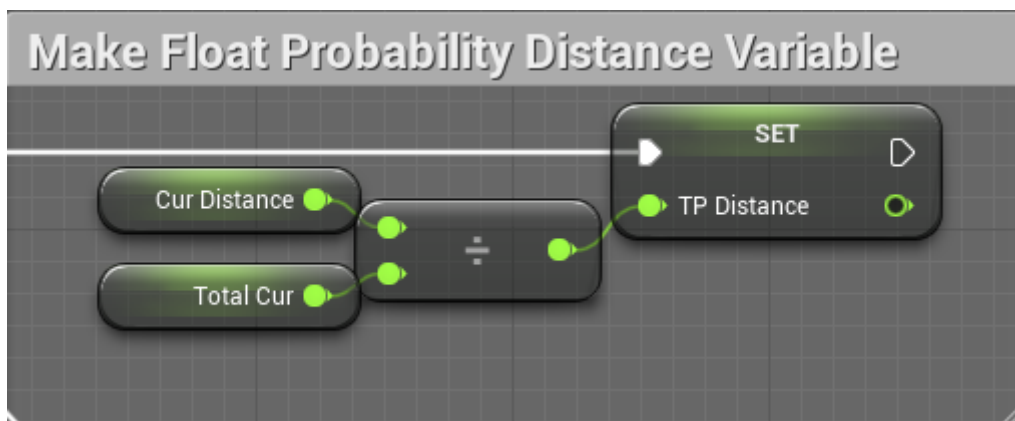
Mendapatkan total dari kondisi *NPC* sekarang dan memasukkan ke dalam total cur seperti Gambar 4.6.



Gambar 4.6 Get Total of All Variable Current

#### 4.3 Membuat Nilai *Current* ke dalam bentuk Float pada Range 0-1

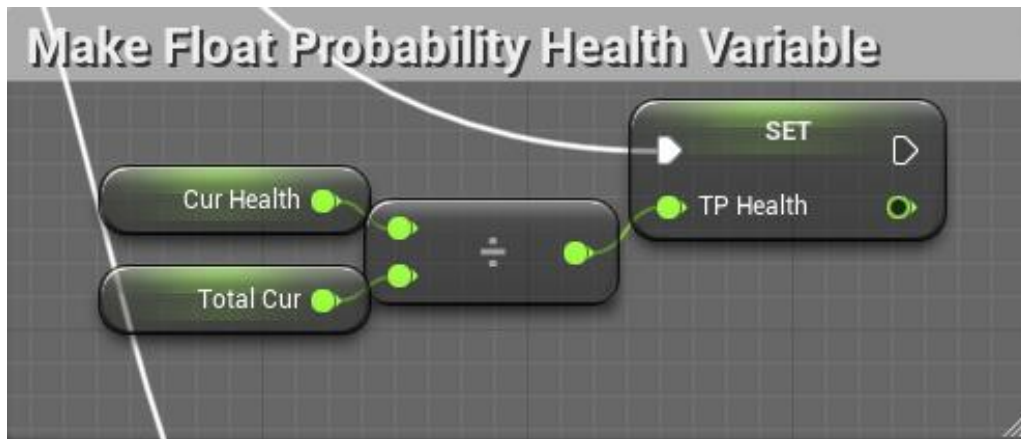
Menjadikan isi *variable* *cur distance* dalam range 0-1 seperti pada subbab 3.2 dan memasukkan hasilnya ke dalam *Temporary Probability* dari *distance* untuk nantinya dikalikan dengan matriks probabilitas seperti pada Gambar 4.7.



Gambar 4.7 Make Float Probability Distance Variable

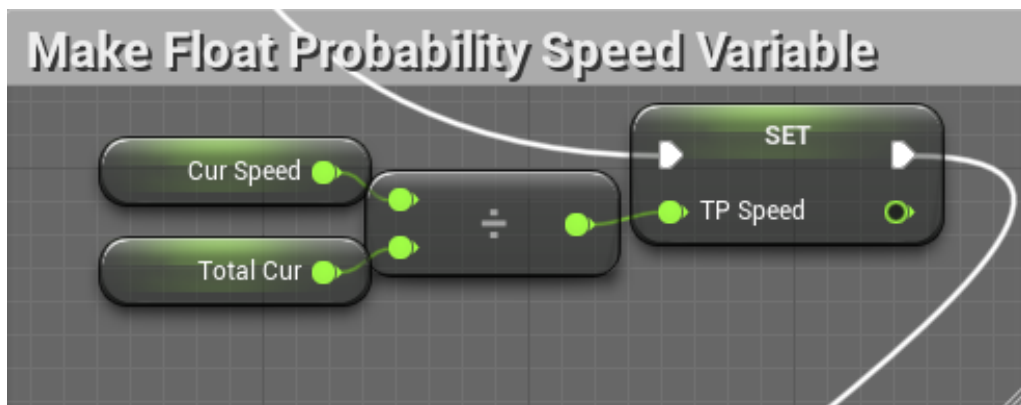
Menjadikan isi *variable* *cur health* dalam range 0-1 seperti pada subbab 3.2 dan memasukkan hasilnya ke dalam *Temporary Probability* dari *health* untuk nantinya dikalikan dengan matriks probabilitas seperti pada Gambar 4.8.





Gambar 4.8 Make Float Probability Health Variable

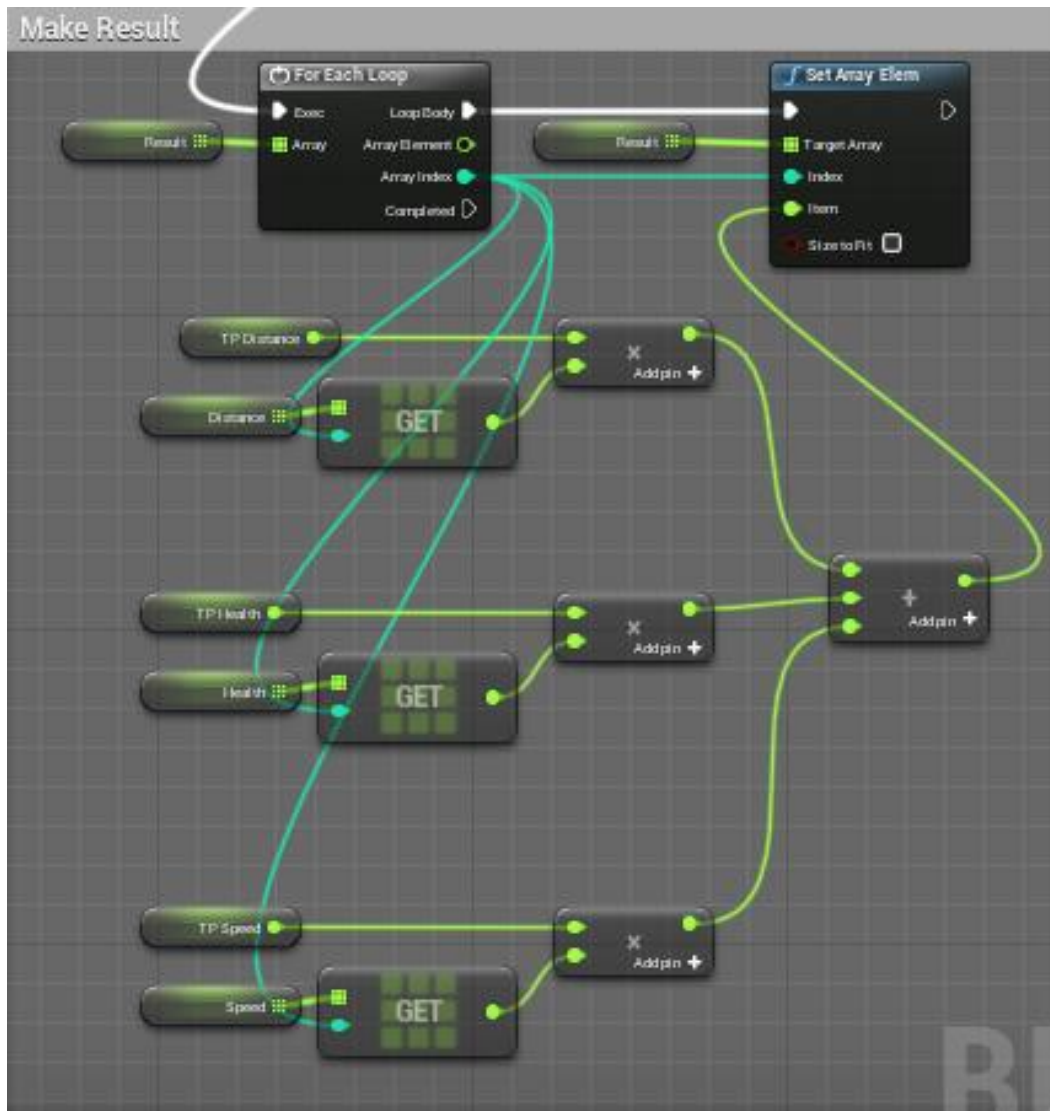
Menjadikan isi *variable* *cur speed* dalam range 0-1 seperti pada subbab 3.2 dan memasukkan hasilnya ke dalam *Temporary Probability* dari *speed* untuk nantinya dikalikan dengan matriks probabilitas seperti pada Gambar 4.9.



Gambar 4.9 Make Float Probability Speed Variable

#### 4.4 Melakukan Perkalian Matriks Probabilitas dan Temporary Probability

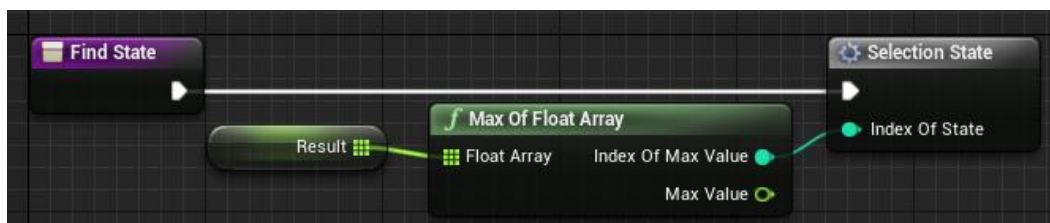
Melakukan perkalian matriks dari *Temporary Probability* dengan matriks probabilitas dan menyimpan hasilnya ke dalam array *result* seperti pada Gambar 4.10.



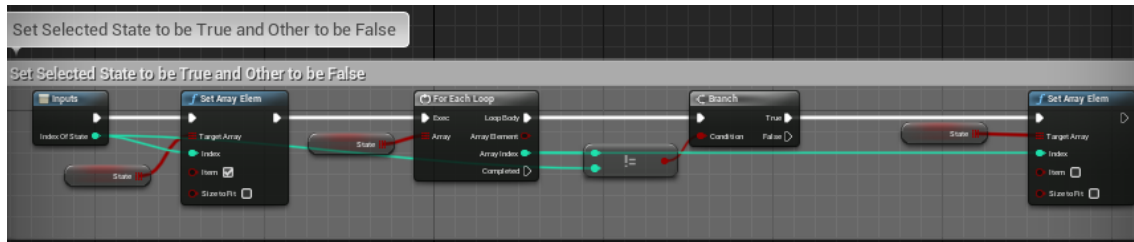
Gambar 4.10 Make Result From Matriks Probability and Temporary Probability Each Variable

#### 4.5 Memilih State Dari Hasil Matriks Probabilitas

Memilih state dari hasil matriks probabilitas dengan value terbesar seperti pada Gambar 4.11 dan Gambar 4.12.



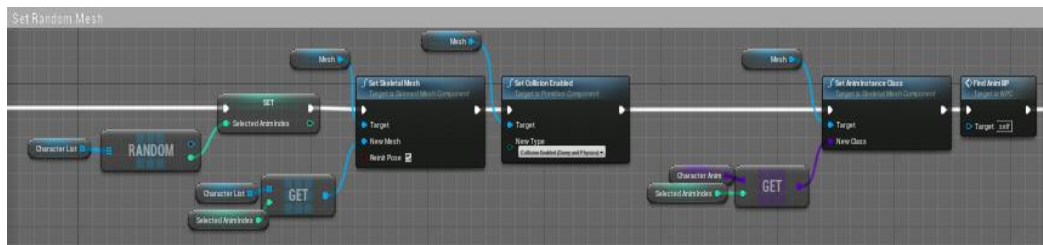
Gambar 4.11 Find State



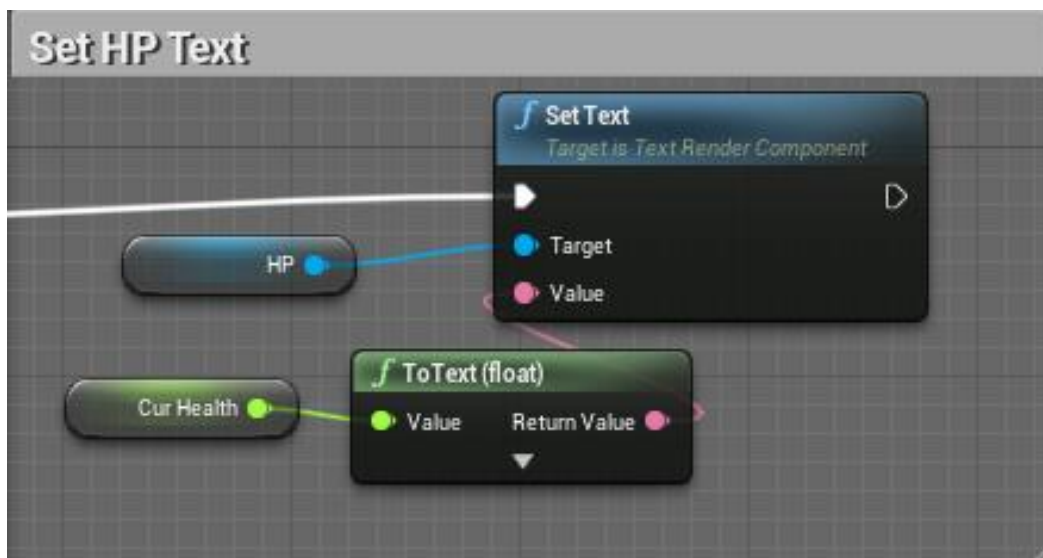
Gambar 4.12 Select State

#### 4.6 Melakukan *Random Enemy* yang Tampil

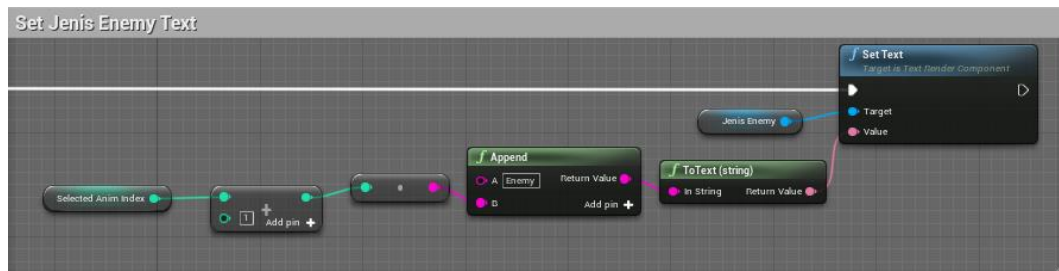
Melakukan *random* pada *mesh enemy* yang tampil, memasang *Collision* pada *mesh*, menentukan *anim instance* yang akan dipakai, kemudian mencari *Blueprint* dari *anim*. Selanjutnya melakukan *binding* atau mengikat *text* HP dengan *variable* *cur health*, kemudian melakukan *binding text* jenis *enemy* dengan jenis *enemy* yang terpilih. Hal ini dapat terlihat pada Gambar 4.13 hingga Gambar 4.15.



Gambar 4.13 Random Mesh, Set Collision, dan Menentukan Anim

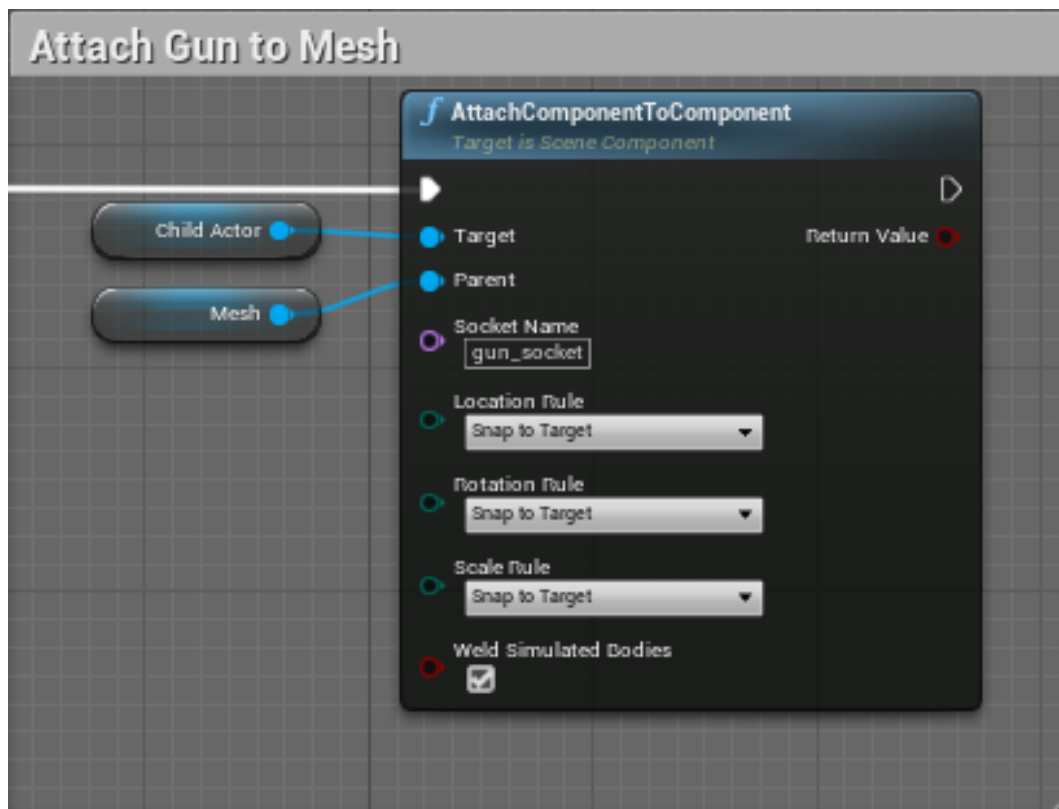


Gambar 4.14 Set HP Text



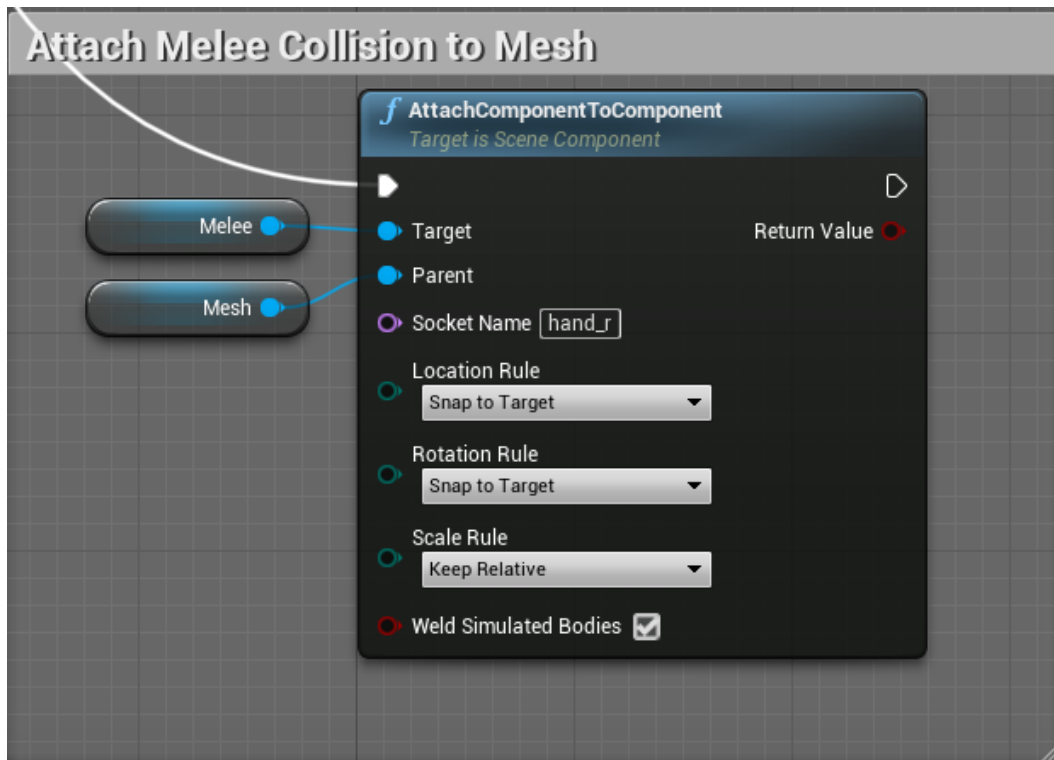
Gambar 4.15 Set Jenis Enemy Text

Setelah itu, memasang *gun* pada *mesh* pada *socket* yang dibuat seperti Gambar 4.16.



Gambar 4.16 Attach Gun to Mesh

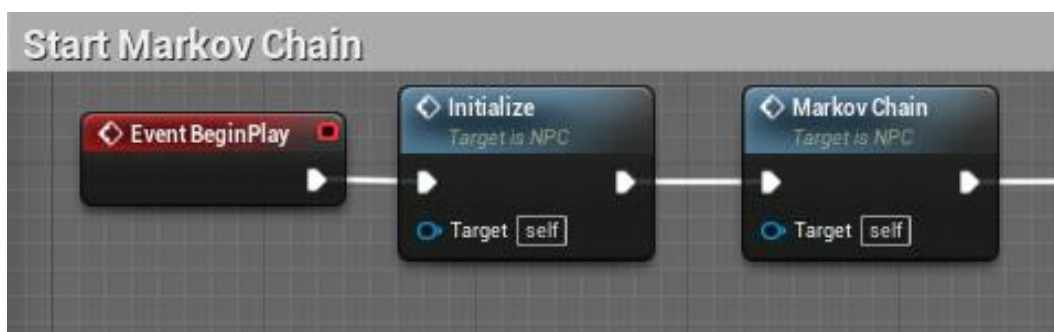
Terakhir memasang *collision detection* untuk melee *attack* seperti pada Gambar 4.17.



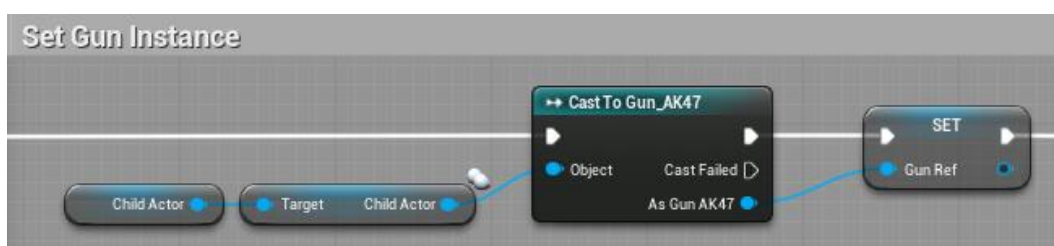
Gambar 4.17 Attach Melee Collision to Mesh

#### 4.7 Membuat Constructor

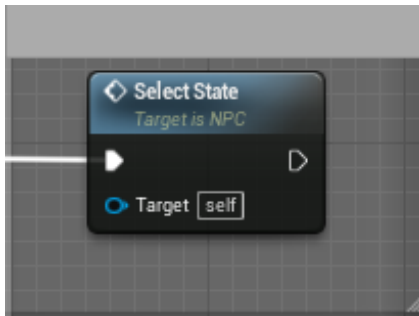
Membuat inisialisasi matriks probabilitas dan memasang proses Markov ke *constructor* seperti pada Gambar 4.18. Memasang *gun instance* seperti pada Gambar 4.19, kemudian memilih *state* seperti pada Gambar 4.20.



Gambar 4.18 Membuat Inisialisasi Matriks Probabilitas dan Memasang Markov Chain



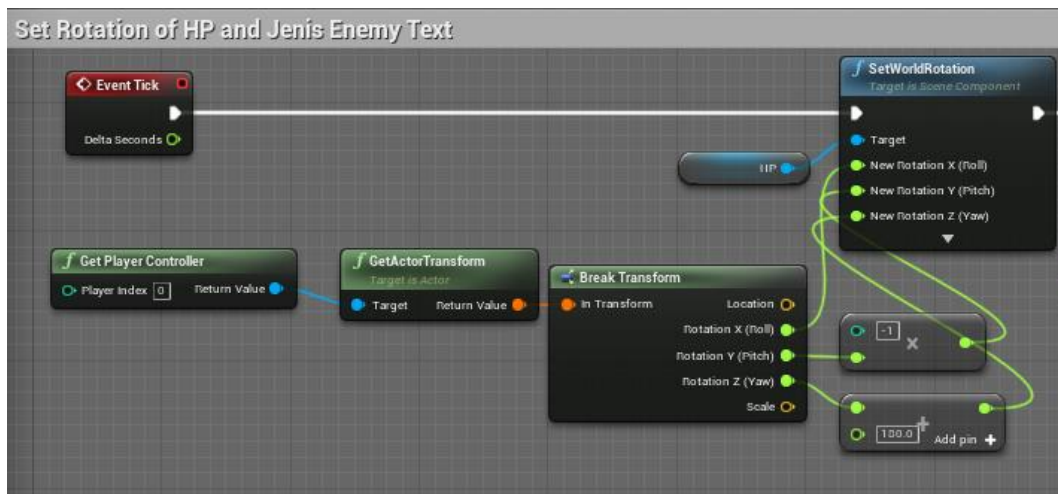
Gambar 4.19 Set Gun Instance



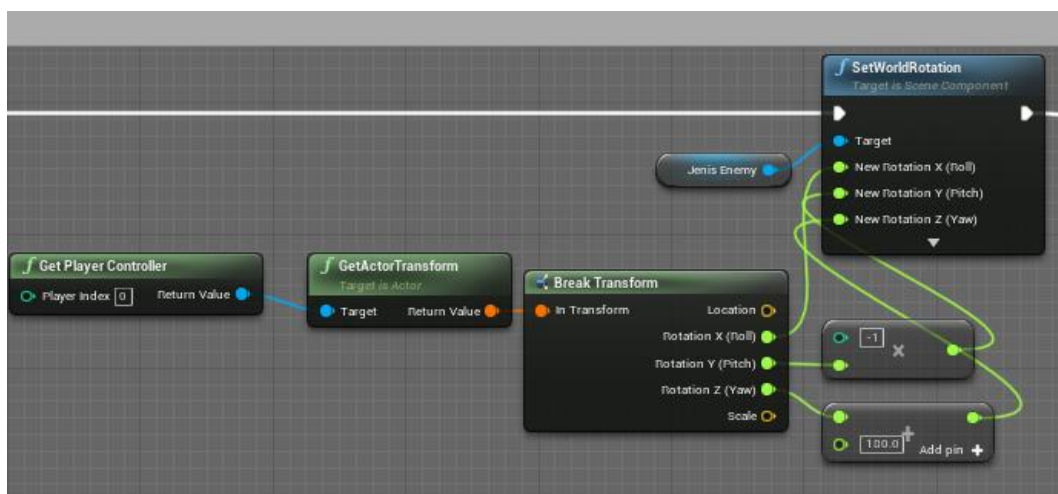
Gambar 4.20 Select State

#### 4.8 Set What to Do Every Time

Melakukan rotasi *text* HP dan *text* jenis *enemy* terhadap layar pemain setiap 2 detik, kemudian melakukan proses *Markov Chain* selama *NPC* masih hidup. Hal ini terlihat pada Gambar 4.21 hingga Gambar 4.23.

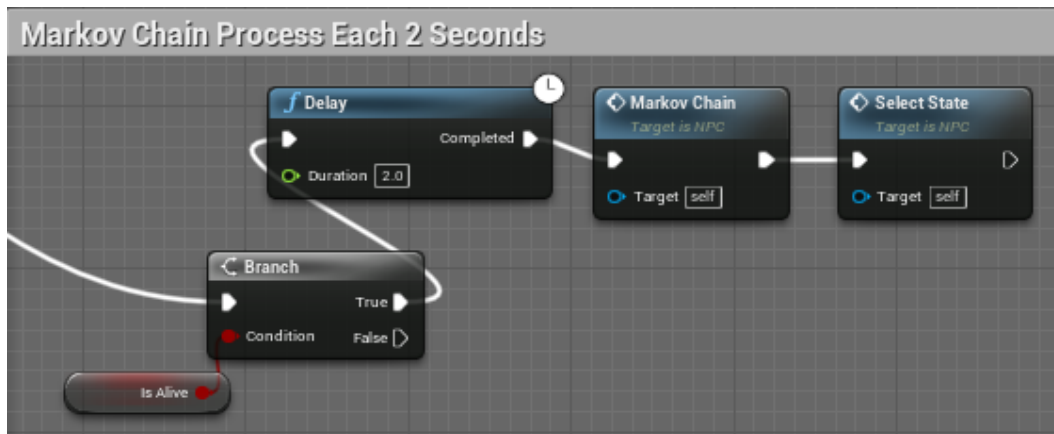


Gambar 4.21 Set Rotation of HP Text



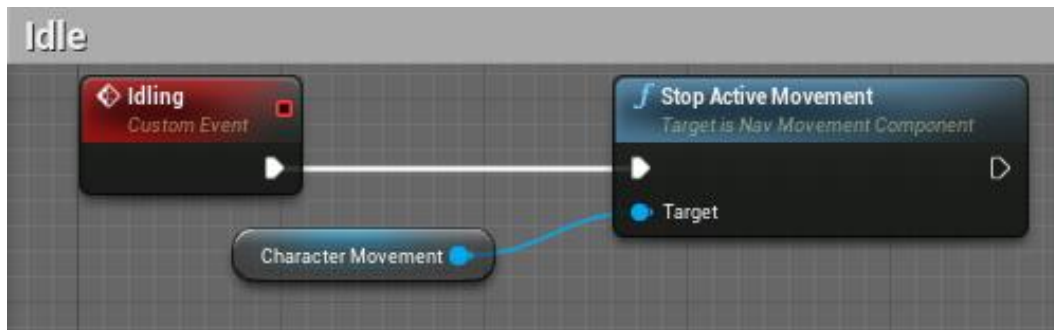
Gambar 4.22 Set Rotation of Jenis Enemy Text





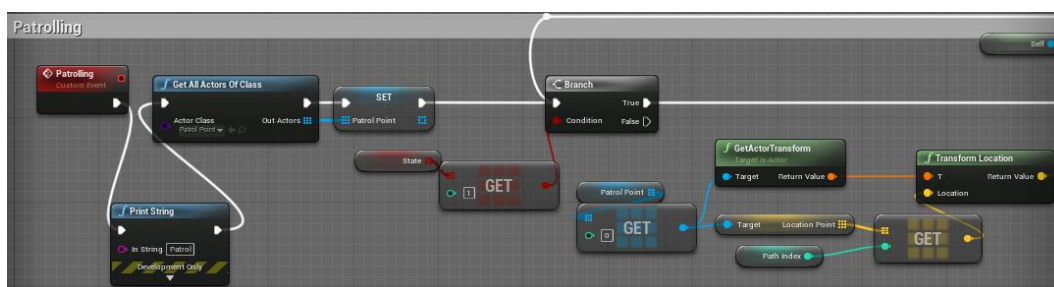
Gambar 4.23 Markov Chain Process

Proses *idle* dengan menghentikan movement seperti pada Gambar 4.24.

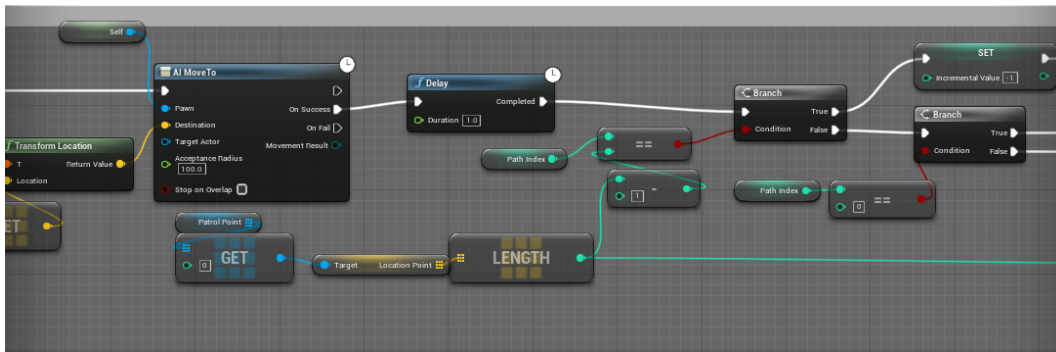


Gambar 4.24 Idle State

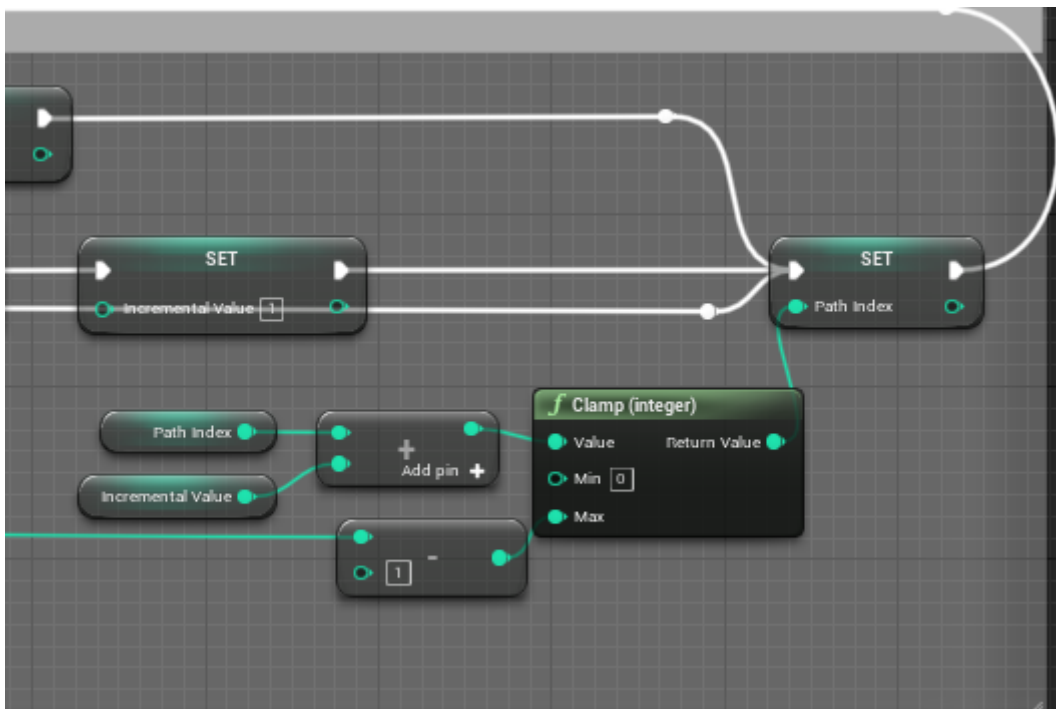
Proses *patrol* dengan mencari titik lokasi *patrol*, kemudian pergi menuju titik tersebut seperti pada Gambar 4.25 hingga Gambar 4.27.



Gambar 4.25 Patrol State-Part1

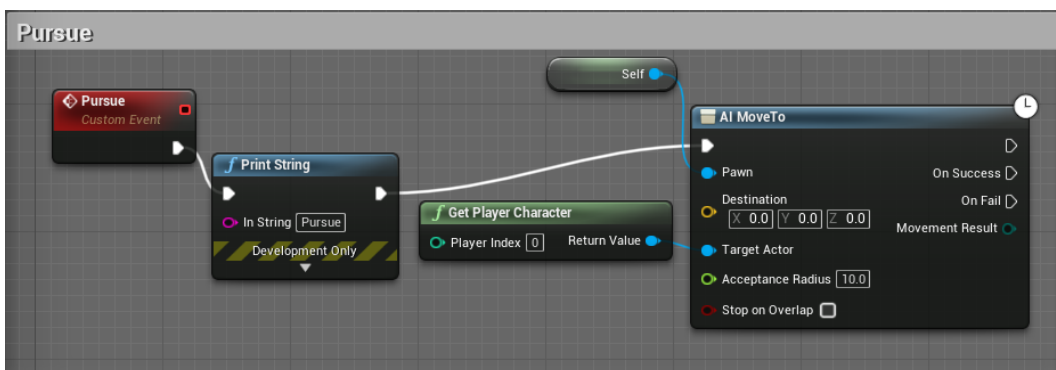


Gambar 4.26 Patrol State-Part2



Gambar 4.27 Patrol State-Part3

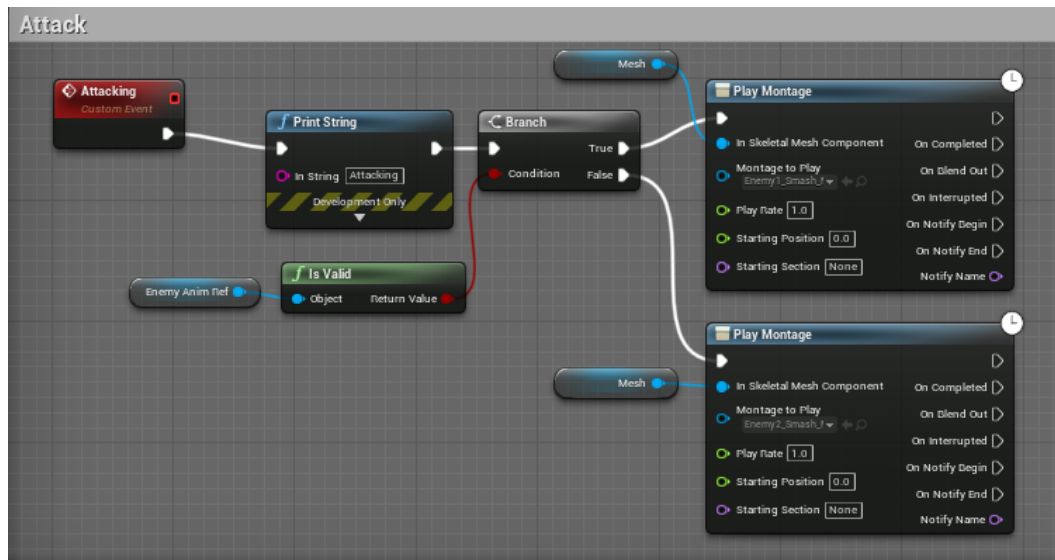
Proses *pursue* atau mengejar pemain dimana AI akan bergerak ke lokasi pemain seperti Gambar 4.28.



Gambar 4.28 Pursue State

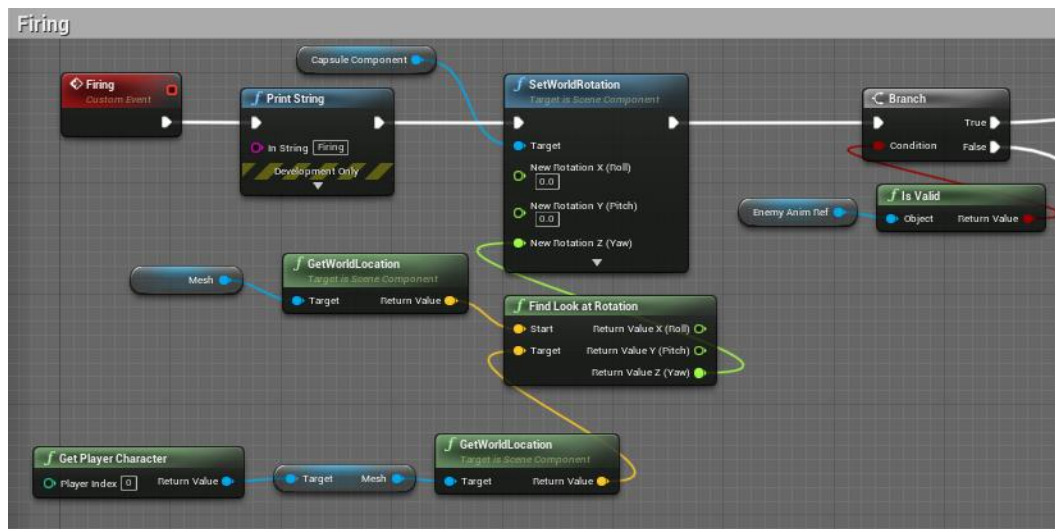


Proses menyerang pemain dengan menggunakan *anim montage* sebagai penggerak animasi seperti pada Gambar 4.29.

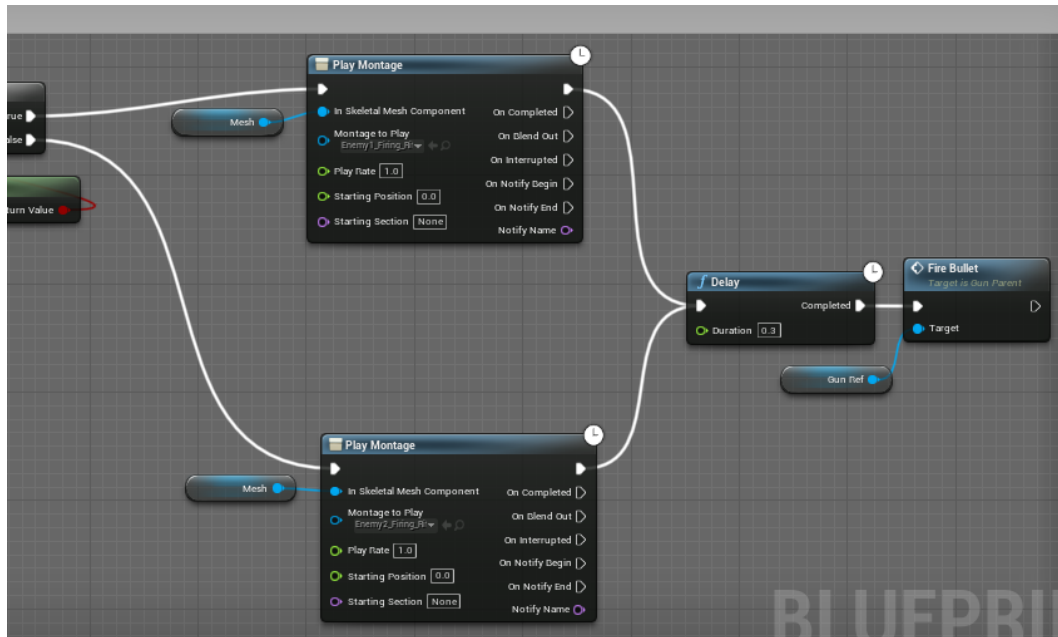


Gambar 4.29 Attack State

Proses menembak pemain dengan memutar arah pandang ke pemain, kemudian melakukan *firing* seperti pada Gambar 4.30 hingga Gambar 4.31.

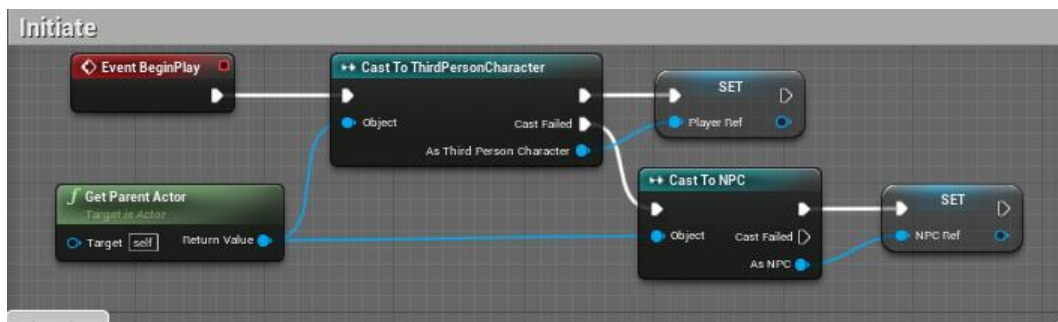


Gambar 4.30 Firing Part1

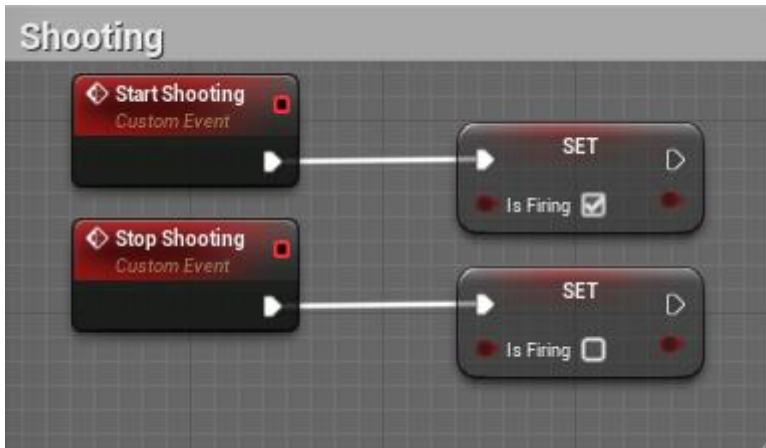


Gambar 4.31 Firing Part2

Buat sebuah *gun blueprint* kemudian buat fungsi untuk konstruktor untuk mengecek apakah parent adalah *NPC* atau *Player*. Buat fungsi untuk start shooting dan stop shooting dengan menambahkan variable Boolean *Is Firing*. Buat fungsi *reload* untuk mengisi ulang peluru. Buat fungsi *fire bullet* untuk mencari lokasi *spawn bullet* hingga menembakkan peluru dan mengatur suara serta peluru yang tersisa. Hal ini terlihat seperti Gambar 4.32 hingga Gambar 4.51.

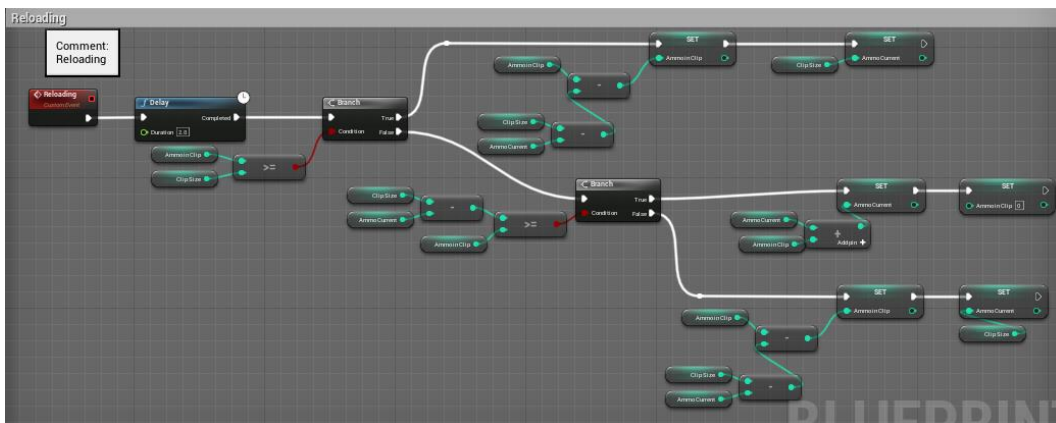


Gambar 4.32 Begin Play atau Constructor untuk Gun Blueprint

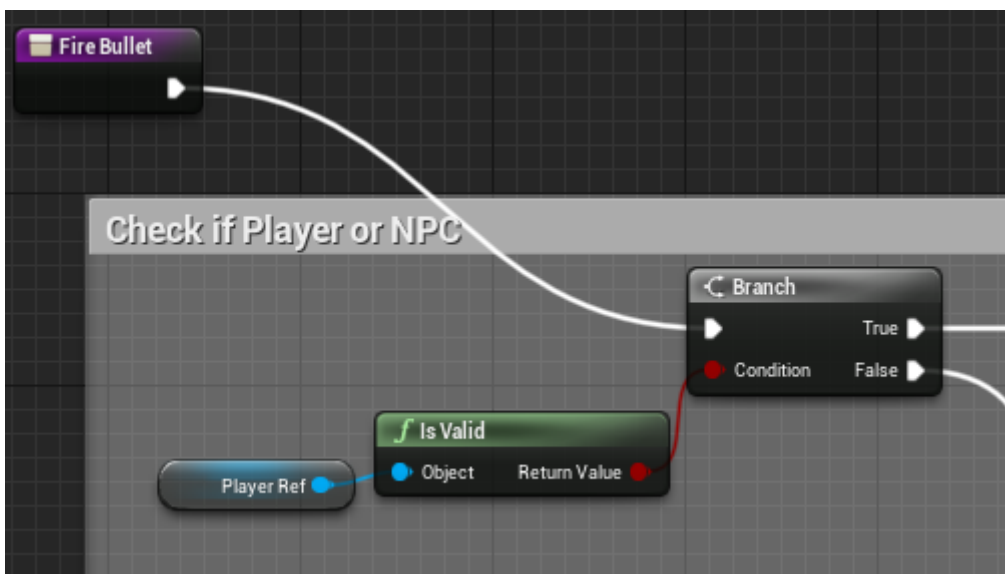


Gambar 4.33 Start and Stop Shooting

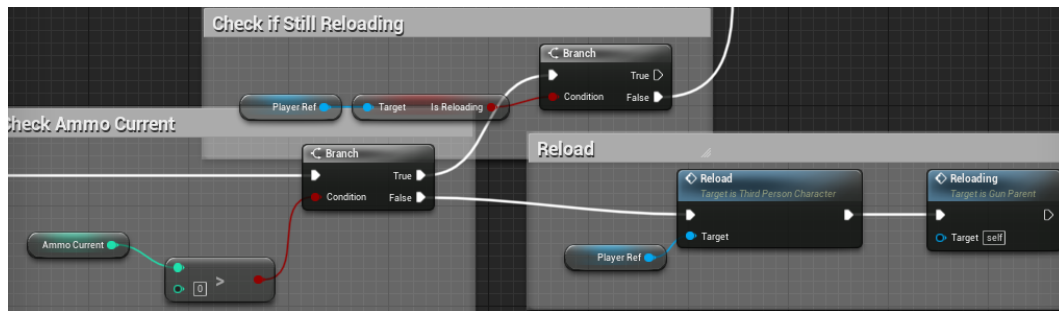
Proses *reload* atau pengisian peluru dilakukan dengan mengecek peluru yang tersisa, kemudian menambahkannya ke dalam jumlah peluru saat ini seperti pada Gambar 4.34.



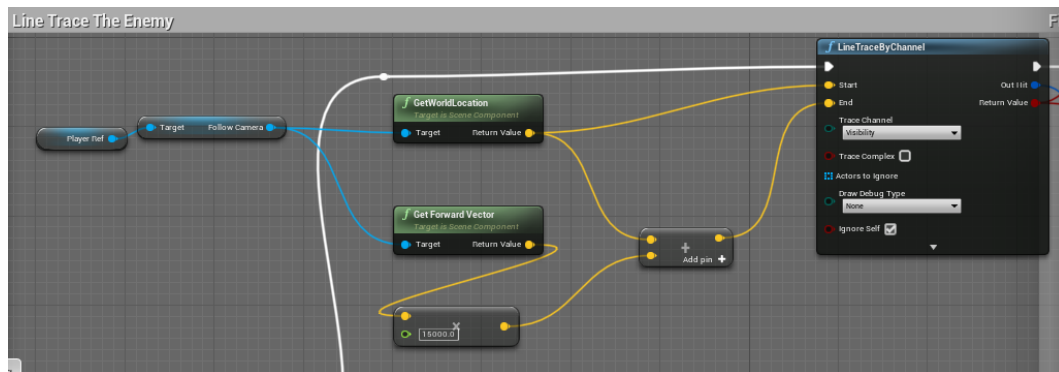
Gambar 4.34 Reloading



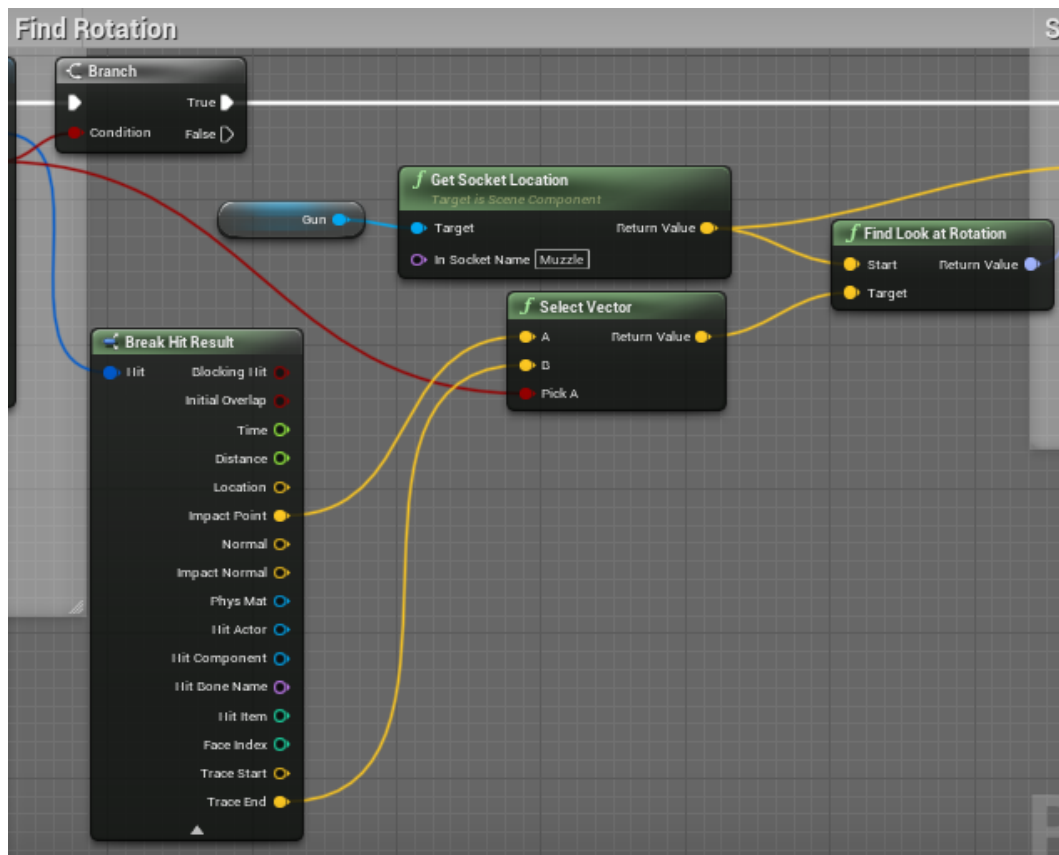
Gambar 4.35 Fire Bullet Part 1



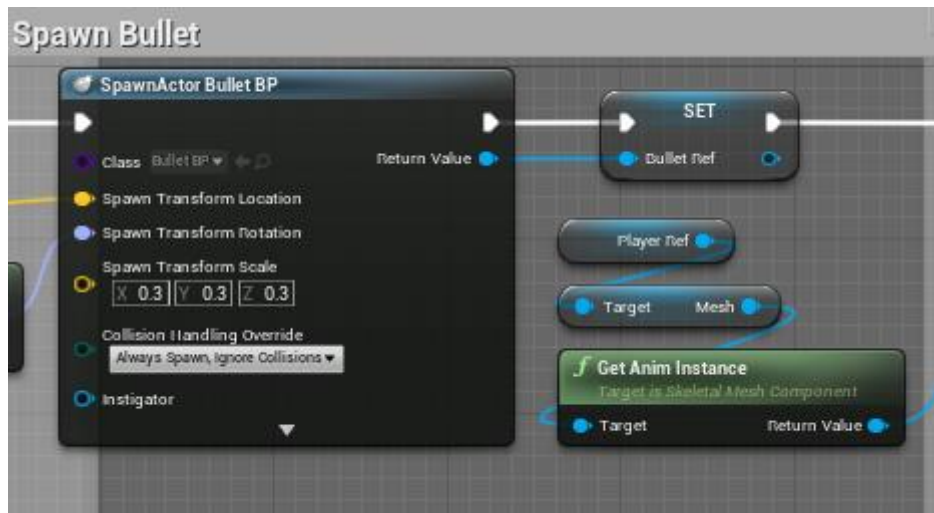
Gambar 4.36 Fire Bullet Part 2



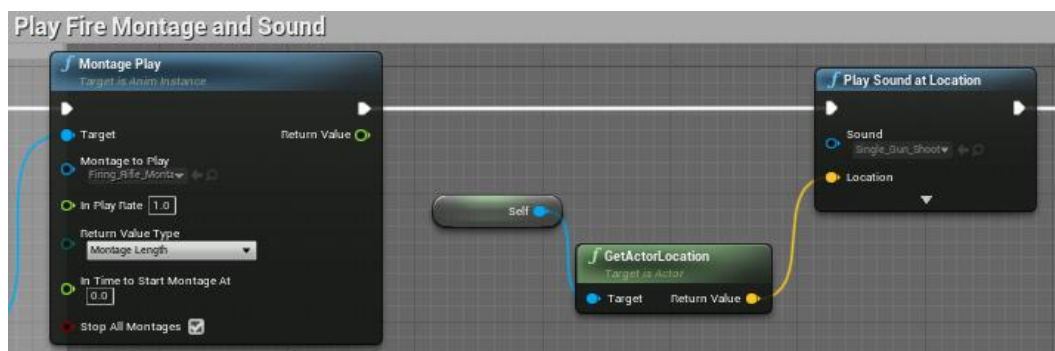
Gambar 4.37 Fire Bullet Part 3



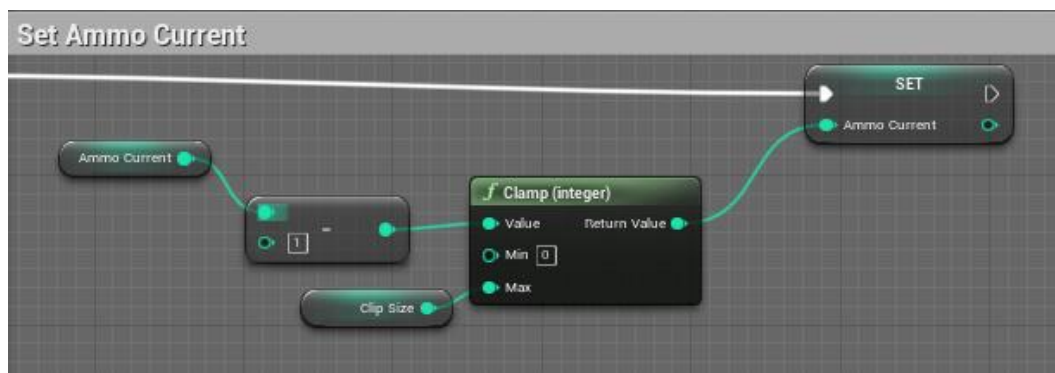
Gambar 4.38 Fire bullet Part 4



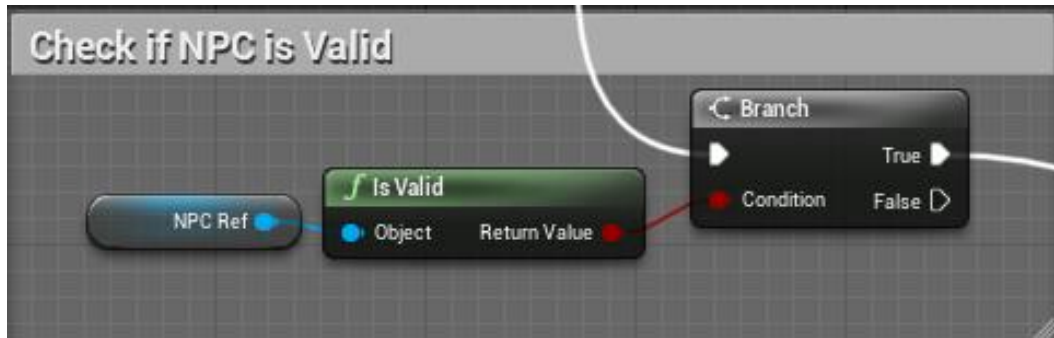
Gambar 4.39 Fire Bullet Part 5



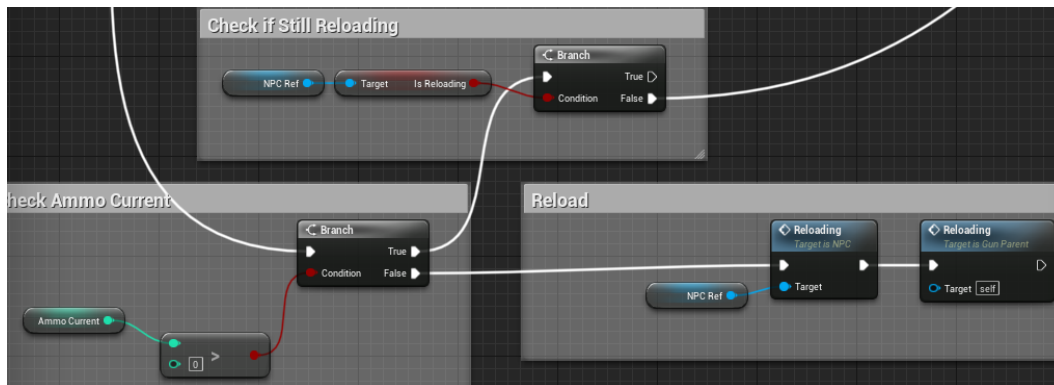
Gambar 4.40 Fire Bullet Part 6



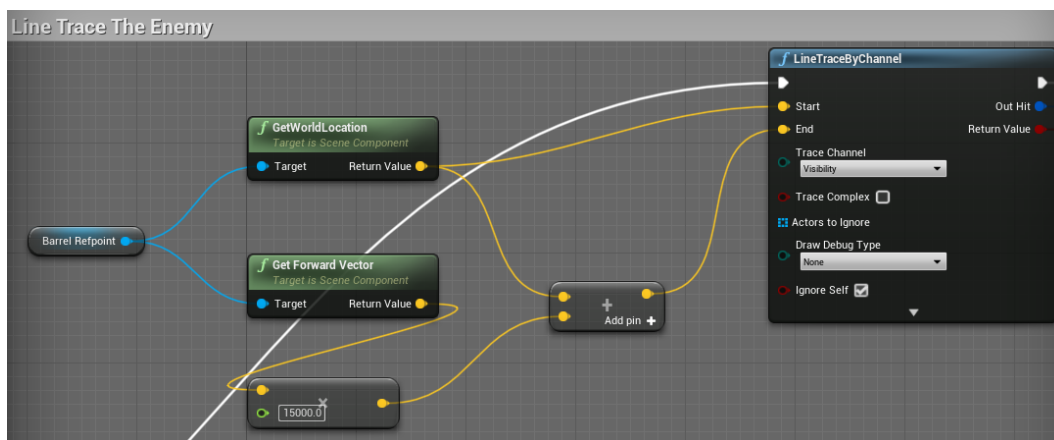
Gambar 4.41 Fire Bullet Part 7



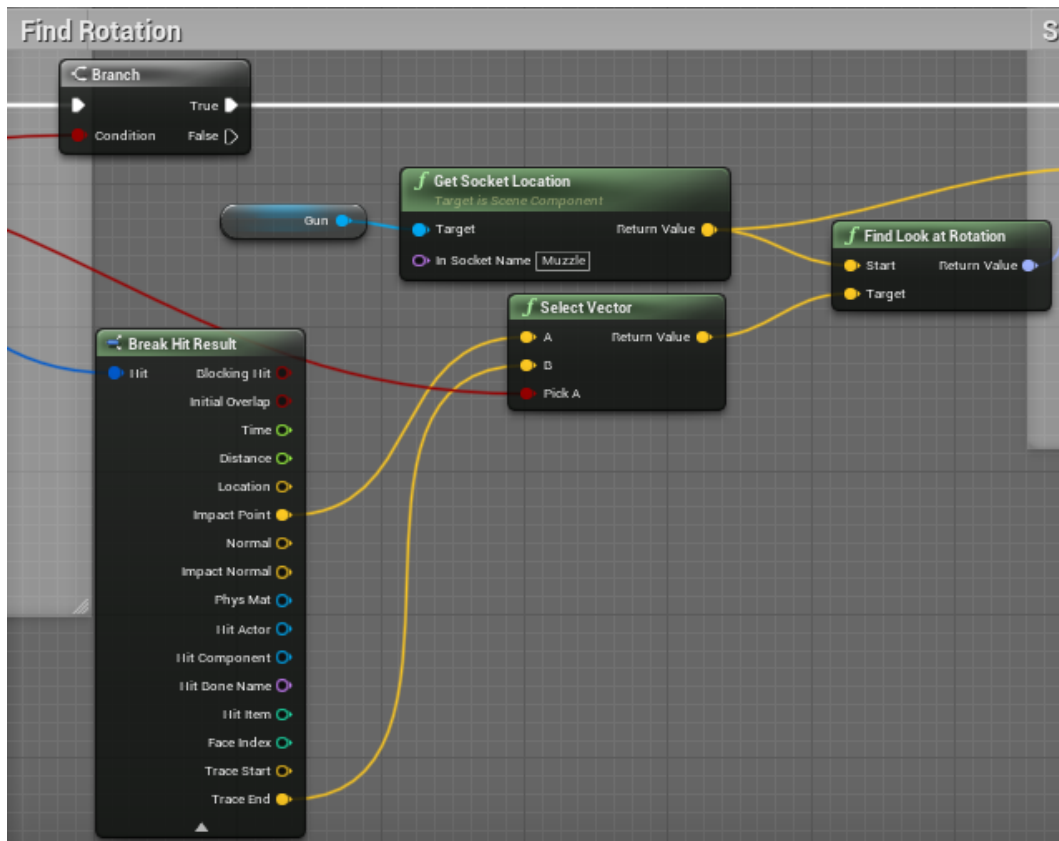
Gambar 4.42 Fire Bullet Part 8



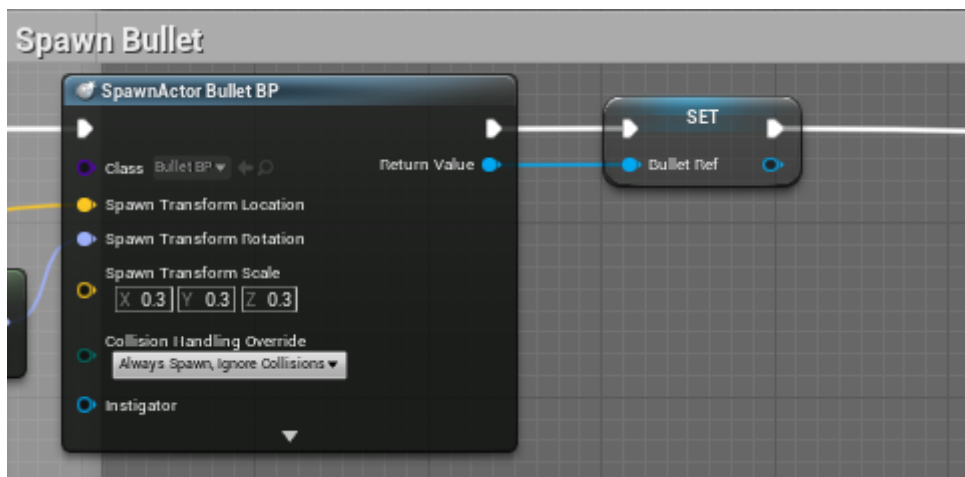
Gambar 4.43 Fire Bullet Part 9



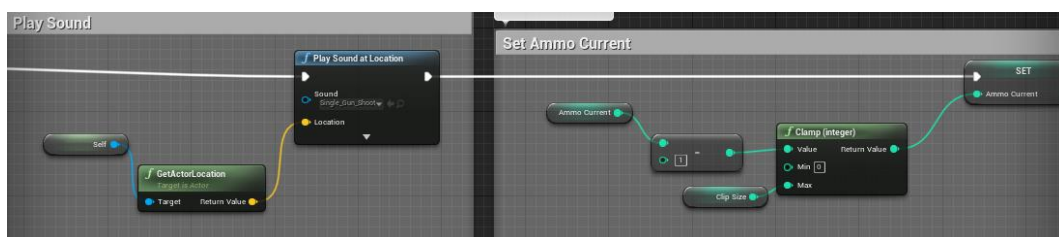
Gambar 4.44 Fire Bullet Part 10



Gambar 4.45 Fire Bullet Part 11

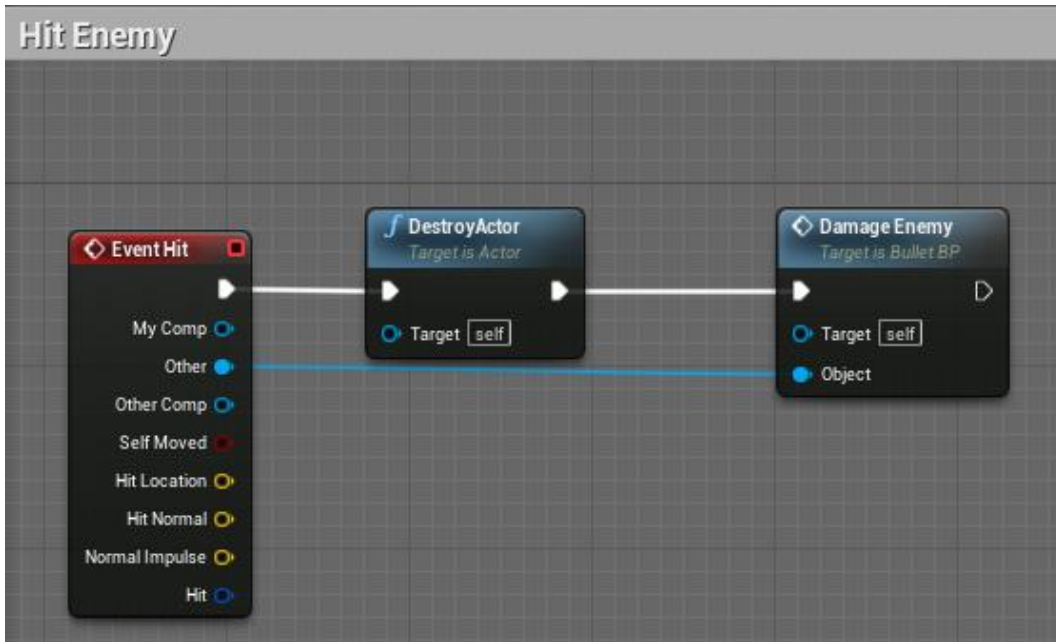


Gambar 4.46 Fire Bullet Part 12

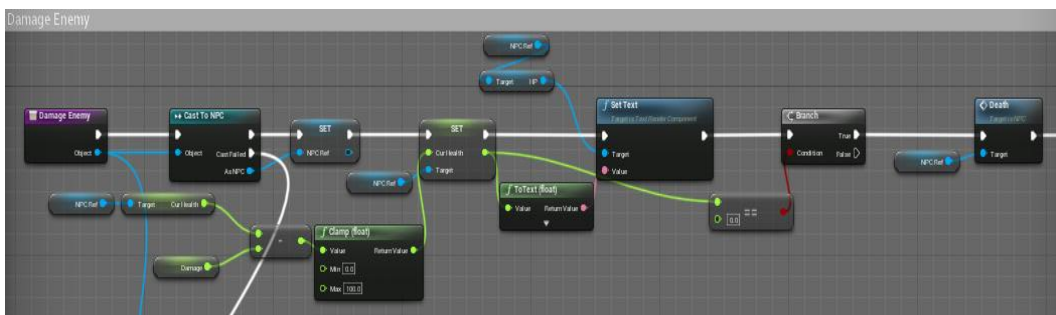


Gambar 4.47 Fire Bullet Part 13

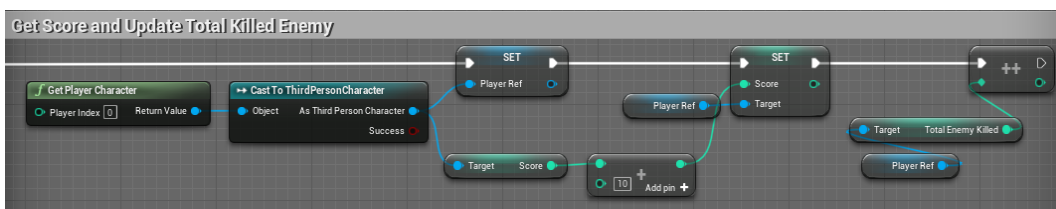




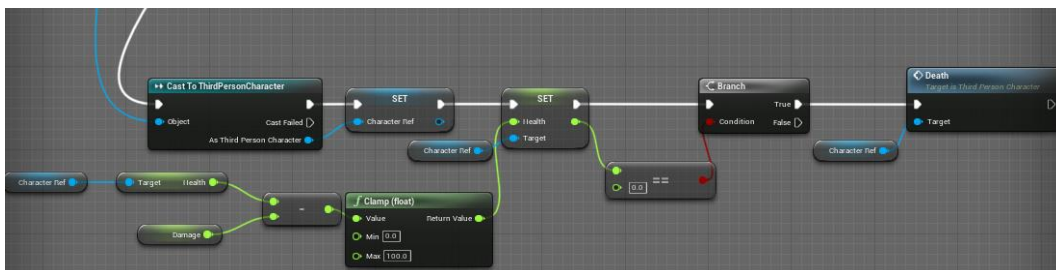
Gambar 4.48 Hit Enemy Event With Bullet BP



Gambar 4.49 Damage Enemy Event



Gambar 4.50 Get Score and Update Total Killed Enemy



Gambar 4.51 Damage Player Event



Proses *suicide* dengan memanggil fungsi buatan *Death* untuk memainkan animasi *death* dan *destroy NPC* seperti pada Gambar 4.53 dan Gambar 4.54.

[illegible]

Universitas Kristen Petra

## 5. PENGUJIAN SISTEM

Pada bab ini akan dibahas mengenai pengujian program yang telah dibuat. Tahap ini meliputi pengujian terhadap inisialisasi *variable input*, pembuatan matriks probabilitas, dan hasil *output state*.

### 5.1 Perangkat Lunak dan Spesifikasi yang dipakai

Pada penelitian ini menggunakan *visual scripting* dengan Unreal Engine 4.26 yang dapat diunduh melalui <https://www.unrealengine.com/en-US/>. *Hardware* yang dipakai saat ini memiliki spesifikasi *processor* Intel Core-i5 dengan tipe 4690, RAM sebesar 12 GB serta kartu grafis Nvidia GeForce GTX 750 Ti sebesar 2GB.

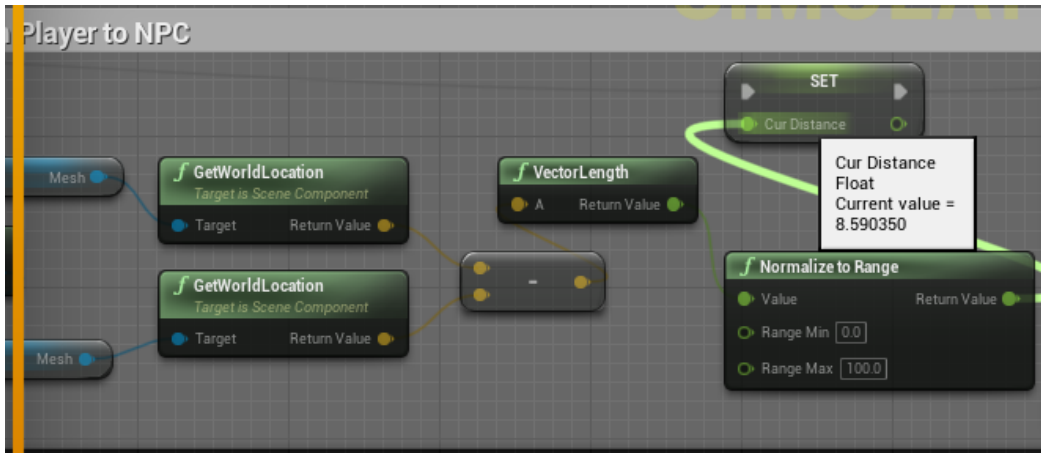
### 5.2 Pengujian Variasi

Pengujian *game* menjelaskan fungsi-fungsi dari proses Markov Chain yang diterapkan ke dalam *game* dengan menggunakan 3 input *variable*. Penjelasan akan dilakukan menggunakan beberapa gambar yang diambil secara berurutan dan sebagai petunjuk state akan di print di sebelah kiri atas layar. Di awal permainan, *NPC* menunjukkan bahwa dia memilih *state patrol* seperti pada Gambar 5.1.

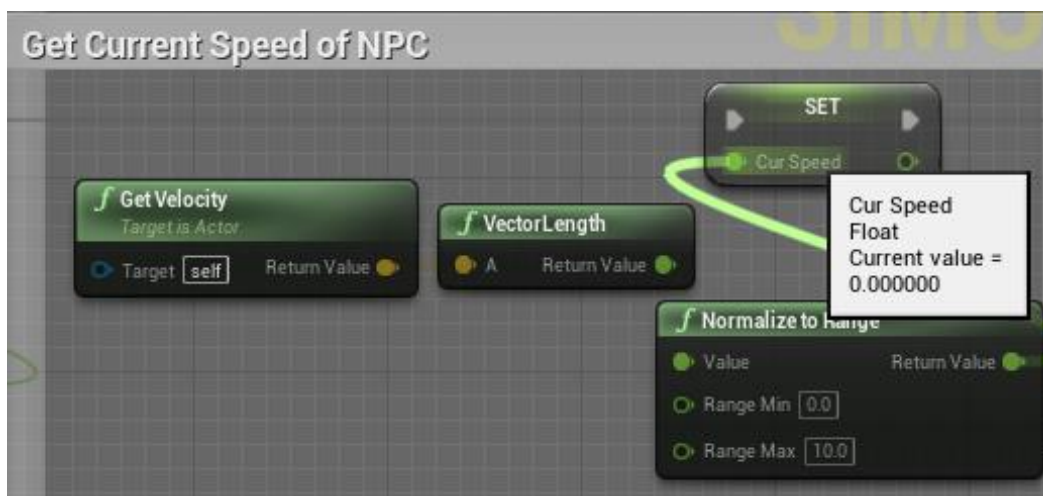


Gambar 5.1 Awal Game Berjalan Memilih State Patrol

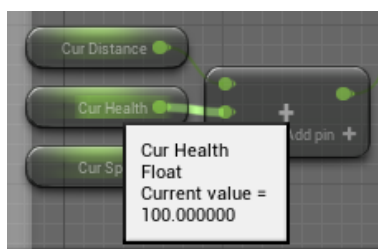
Pada kondisi ini didapatkan nilai dari *cur\_distance* sebesar 8.59, *cur\_speed* sebesar 0, dan *cur\_health* sebesar 100 seperti pada Gambar 5.2 hingga Gambar 5.4.



Gambar 5.2 Current Distance from Player



Gambar 5.3 Current Speed of NPC



Gambar 5.4 Current Health of NPC

Dari hasil *current* variabel dikalikan dengan matriks probabilitas sehingga didapatkan masing-masing probabilitas untuk variabel output, kemudian dipilih probabilitas terbesar pada indeks ke-1 yaitu *patrol* seperti pada Gambar 5.5.

```

Result
Array of Floats
Current value =
[0] 0.115822
[1] 0.203955
[2] 0.177627
[3] 0.100000
[4] 0.190507
[5] 0.143671
[6] 0.068418

```

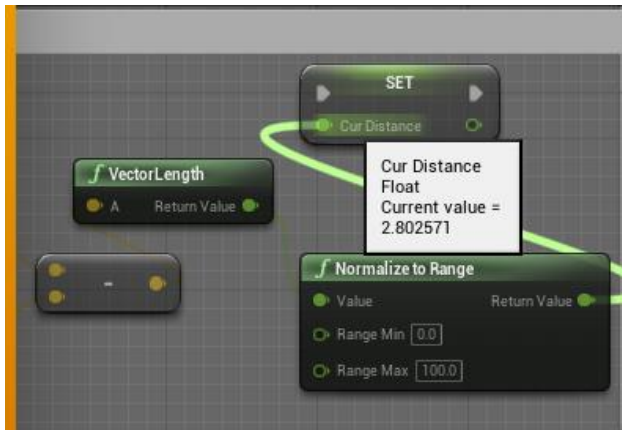
Gambar 5.5 Hasil perkalian Current Variabel dengan Matriks Probabilitas

Dari hasil yang pertama, lalu dilakukan perubahan pada jarak, kecepatan, dan kesehatan dari pemain sehingga didapatkan *state idle* seperti pada Gambar 5.6.

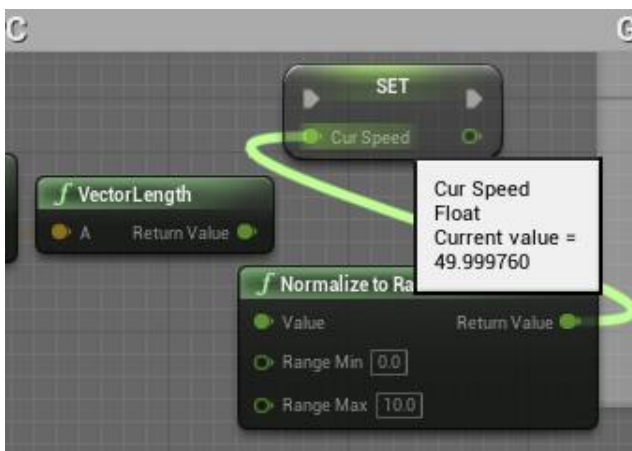


Gambar 5.6 *State Idle*

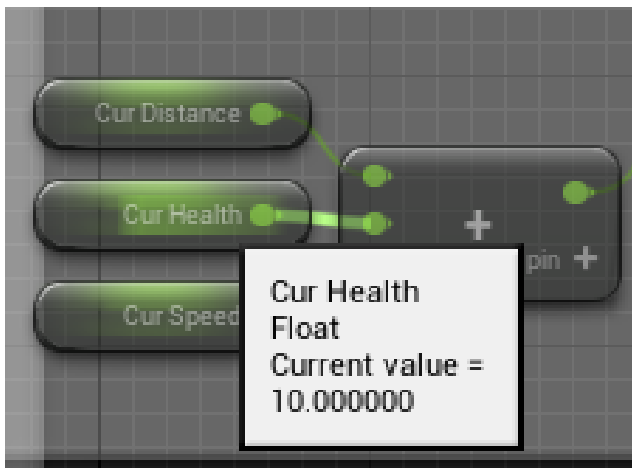
Pada kondisi ini didapatkan nilai dari *cur\_distance* sebesar 2.80, *cur\_speed* sebesar 49.99, dan *cur\_health* sebesar 10 seperti pada Gambar 5.7 hingga Gambar 5.9.



Gambar 5.7 *Current Distance from Player*



Gambar 5.8 *Current Speed of NPC*



Gambar 5.9 *Current Health of NPC*

Dari hasil *current* variabel dikalikan dengan matriks probabilitas sehingga didapatkan masing-masing probabilitas untuk variabel output, kemudian dipilih probabilitas terbesar pada indeks ke-0 yaitu *idle* seperti pada Gambar 5.10.



```
Result
Array of Floats
Current value =
[0] 0.228347
[1] 0.202231
[2] 0.130893
[3] 0.139807
[4] 0.115030
[5] 0.090700
[6] 0.092992
```

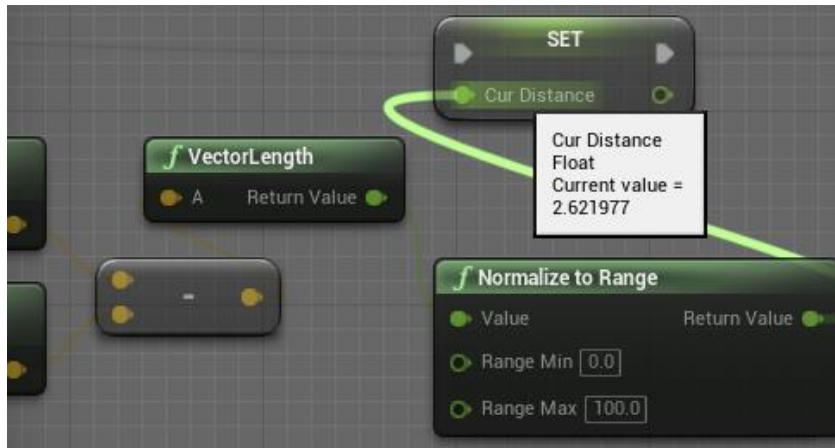
Gambar 5.10 Hasil perkalian Current Variabel dengan Matriks Probabilitas

Pada percobaan kedua, digunakan matriks probabilitas yang berbeda, di awal permainan, NPC akan memilih *state idle* seperti pada Gambar 5.11.

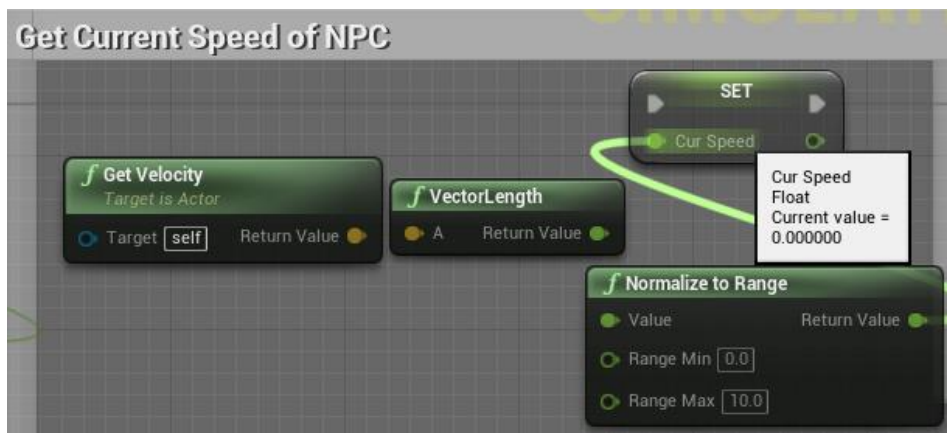


Gambar 5.11 Awal Permainan, NPC Memilih State Idle

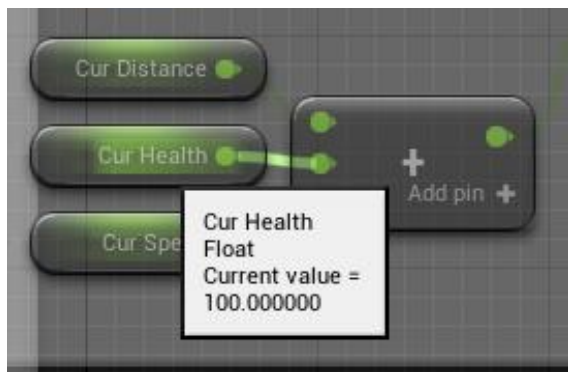
Pada kondisi ini didapatkan nilai dari *cur\_distance* sebesar 2.62, *cur\_speed* sebesar 0, dan *cur\_health* sebesar 100 seperti pada Gambar 5.12 hingga Gambar 5.14.



Gambar 5.12 Current Distance from Player



Gambar 5.13 Current Speed of NPC



Gambar 5.14 Current Health of NPC

Dari hasil *current* variabel dikalikan dengan matriks probabilitas sehingga didapatkan masing-masing probabilitas untuk variabel output, kemudian dipilih probabilitas terbesar pada indeks ke-0 yaitu *idle* seperti pada Gambar 5.15.

```

Result
Array of Floats
Current value =
[0] 0.393613
[1] 0.201277
[2] 0.100511
[3] 0.080511
[4] 0.123321
[5] 0.060255
[6] 0.040511

```

Gambar 5.15 Hasil perkalian Current Variabel dengan Matriks Probabilitas

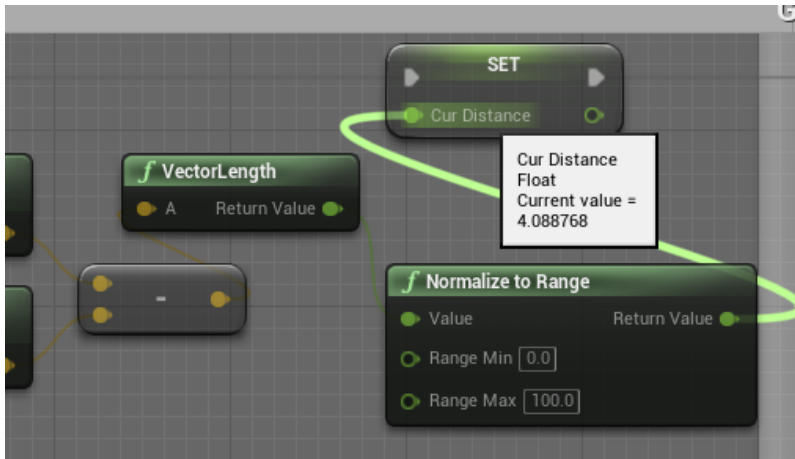
Dari hasil yang pertama, lalu dilakukan perubahan pada jarak, kecepatan, dan kesehatan dari pemain sehingga didapatkan *state pursue* seperti pada Gambar 5.16.



Gambar 5.16 State Pursue

Pada kondisi ini didapatkan nilai dari *cur\_distance* sebesar 4.08, *cur\_speed* sebesar 49.99, dan *cur\_health* sebesar 10 seperti pada Gambar 5.17 hingga Gambar 5.19.

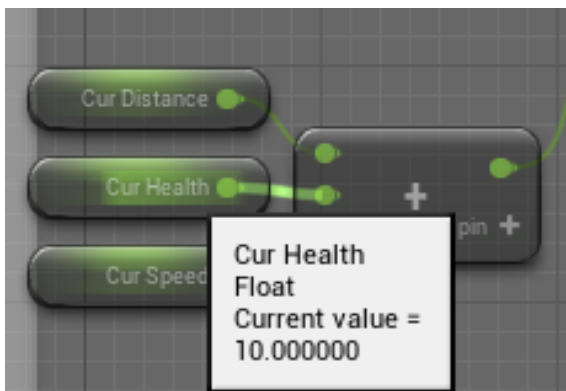




Gambar 5.17 *Current Distance from Player*



Gambar 5.18 *Current Speed of NPC*



Gambar 5.19 *Current Health of NPC*

Dari hasil *current* variabel dikalikan dengan matriks probabilitas sehingga didapatkan masing-masing probabilitas untuk variabel output, kemudian dipilih probabilitas terbesar pada indeks ke-2 yaitu *pursue* seperti pada Gambar 5.20.

```

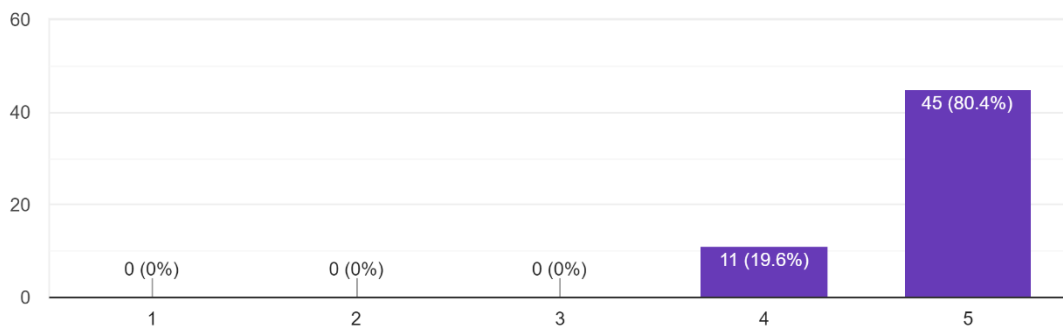
Result
Array of Floats
Current value =
[0] 0.150000
[1] 0.203190
[2] 0.257309
[3] 0.135888
[4] 0.112690
[5] 0.052836
[6] 0.088086

```

Gambar 5.20 Hasil perkalian Current Variabel dengan Matriks Probabilitas

Seberapa variatif NPC yang dihadapi?

56 responses

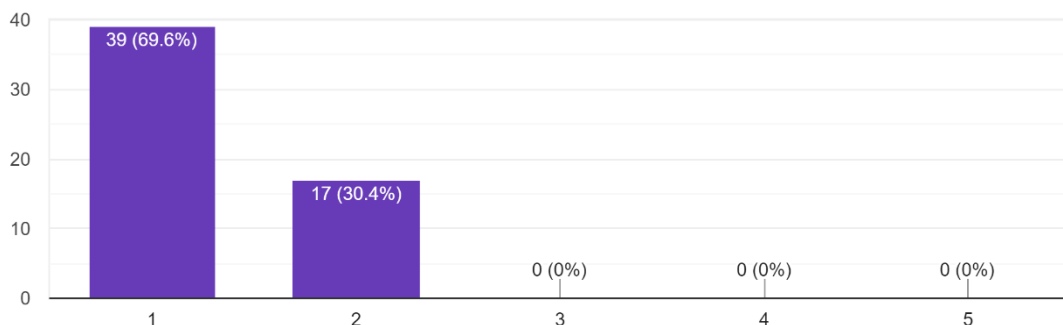


Gambar 5.21 Hasil Responden Terhadap Variasi NPC

Berdasar hasil survey pada Gambar 5.21, maka didapatkan sebanyak 80.4 % cenderung mengatakan bahwa NPC sangat bervariasi, dan 19.6% mengatakan bahwa NPC bervariasi.

Seberapa realistis NPC yang dihadapi?

56 responses



Gambar 5.22 Hasil Responden Terhadap Tingkat Realistis NPC

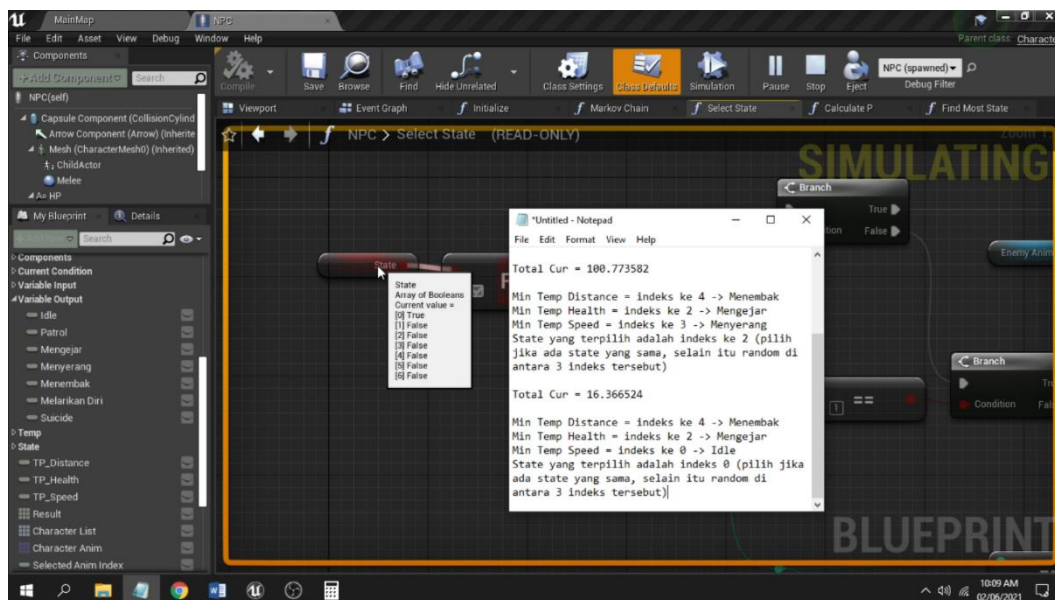
Berdasar hasil survey pada Gambar 5.22, maka didapatkan sebanyak 69.6 % cenderung mengatakan bahwa *NPC* sangat tidak realistis, dan 30.4% mengatakan bahwa *NPC* tidak realistis.

### 5.3 Pengujian Main Menu

Fungsi *campaign button* adalah untuk menjalankan permainan dan fungsi *exit button* adalah untuk keluar dari permainan. Pemain perlu memilih salah satu tombol tersebut saat berada di *main menu*.

### 5.4 Pengujian Gameplay

Pengujian ini dilakukan saat *game* dijalankan, *NPC* akan di spawn di sebuah titik. *NPC* yang di spawn akan langsung menjalankan proses *Markov Chain* dan menghasilkan state yang akan di jalankan oleh *NPC*. Contohnya pada percobaan berikut menghasilkan state *Idle* seperti pada Gambar 5.23.



Gambar 5.23 State Yang Terpilih dari Proses Markov Chain

Pada proses tersebut, nilai dari *variable* kesehatan, kecepatan, dan jarak mempengaruhi *AI* dalam memilih *state*. Tanpa adanya *variable* tersebut maka hasilnya selalu statis pada satu *state* saja. Nilai dari *variable* kesehatan, kecepatan, dan jarak selalu berubah setiap saat dan membuat hasil perhitungan berubah, lalu pengambilan keputusan juga ikut berubah sehingga bervariasi namun kurang realistis.

## 6. KESIMPULAN DAN SARAN

Bab ini akan menjelaskan mengenai kesimpulan yang didapat dari penerapan metode *Markov Chain* ke dalam *NPC* sehingga *NPC* menghasilkan *state* yang bervariasi.

### 6.1 Kesimpulan

Setelah dilakukan pengujian dengan menggunakan metode *Markov Chain* dalam membuat *state NPC* menjadi bervariasi, maka dapat disimpulkan sebagai berikut:

- a) Sebanyak 80.4% menyatakan sangat setuju dan 19.6% setuju bahwa *NPC* menghasilkan variasi tindakan yang berbeda saat dimainkan sehingga *NPC* menjadi lebih bervariasi.
- b) Sebanyak 69.6% menyatakan sangat tidak realistis dan 30.4% menyatakan tidak realistis pada tindakan yang dipilih *NPC* pada kondisi yang dihadapi sehingga pergerakan *NPC* menjadi tidak nyata.

### 6.2 Saran

Berdasarkan hasil sistem berupa permainan yang telah dibuat beserta dengan hasil *state*-nya, maka disarankan bahwa:

- a) Penambahan variable output sehingga lebih banyak *state* yang bervariasi.
- b) Penggunaan *Markov Chain* disesuaikan kembali agar tetap bervariasi namun lebih realistis.

## DAFTAR PUSTAKA

- Ge, M., Hu, J., Liu, M., & Zhang, Y. (2018). Reassembly classification selection method based on the Markov Chain. *Assembly Automation*, 38(4), 476–486. <https://doi.org/10.1108/AA-03-2017-043>
- Hassan, I., Faisal, M., & Arif, Y. M. (2018). *Implementation of Artificial Bee Colony Algorithm to Generate NPC Behavior in Survival Horror Game " Left Alone " As A Media Introduction to House of Cut Nyak Dhien*. 7(1), 16–24.
- Hernawan, S. R. (2018). *PENERAPAN METODE FINITE STATE MACHINE PADA GAME “ The Mahasiswa ” G UNA MEMBANGUN PERILAKU NON PLAYABLE CHARACTER HALAMAN PENGESAHAN DOSEN PENGUJI PENERAPAN METODE FINITE STATE MACHINE PADA GAME “ The Mahasiswa ” G UNA MEMBANGUN PERILAKU NON PLAYABLE CHAR*. 14523032.
- Hidayat, E. W., Rachman, A. N., & Azim, M. F. (2019). Penerapan Finite State Machine pada Battle Game Berbasis Augmented Reality. *Jurnal Edukasi Dan Penelitian Informatika (JEPIN)*, 5(1), 54. <https://doi.org/10.26418/jp.v5i1.29848>
- Kasse, I., Didiaryono, D., & Maulidina, M. (2020). Metode Markov Chain untuk Menghitung Premi Asuransi pada Pasien Penderita Penyakit Demam Berdarah Dengue. *Al-Khwarizmi: Jurnal Pendidikan Matematika Dan Ilmu Pengetahuan Alam*, 7(2), 151–160. <https://doi.org/10.24256/jpmipa.v7i2.1251>
- Kopel, M., & Hajas, T. (2018). Implementing AI for Non-player Characters in 3D Video Games. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10751 LNAI, 610–619. [https://doi.org/10.1007/978-3-319-75417-8\\_57](https://doi.org/10.1007/978-3-319-75417-8_57)
- S. Utami 1 , I W. Mangku 2, I. G. P. P. 2. (2018). *EVALUASI NUMERIK PENDUGA FUNGSI NILAI HARAPAN DAN FUNGSI RAGAM PROSES POISSON MAJEMUK DENGAN INTENSITAS EKSPONENSIAL FUNGSI LINEAR* S. Utami 1 , I W. Mangku 2 , I G. P. Purnaba 2.
- Saprudin, S., Liliyasi, L., Setiawan, A., & Prihatmanto, A. S. (2019). The effectiveness of using digital game towards students’ academic achievement in small and large classes: A comparative research. *International Journal of Learning, Teaching and Educational Research*, 18(12), 196–210. <https://doi.org/10.26803/ijlter.18.12.12>
- Tadayon, M., & Pottie, G. J. (2020). Predicting Student Performance in an Educational Game Using a Hidden Markov Model. *IEEE Transactions on Education*, 63(4), 299–304.

- <https://doi.org/10.1109/TE.2020.2984900>
- Tirtana, A., & Pumpungan, M. (2020). *Perancangan Game Visual Novel “Coconut Kids” Sebagai Sarana Edukasi Pelestarian Alam*. 45.
- Wang, L., Huang, W., Li, Y., Evans, J., & He, S. (2020). Multi-AI competing and winning against humans in iterated rock-paper-scissors game. *ArXiv*.
- Yulsilviana, E., & Ekawati, H. (2019). Penerapan Metode Finite State Machine (Fsm) Pada Game Agent Legenda Anak Borneo. *Sebatik*, 23(1), 116–123.  
<https://doi.org/10.46984/sebatik.v23i1.453>
- Zhou, Y., Wang, L., Zhong, R., & Tan, Y. (2018). A Markov Chain Based Demand Prediction Model for Stations in Bike Sharing Systems. *Mathematical Problems in Engineering*, 2018.  
<https://doi.org/10.1155/2018/8028714>
- Zhu, X. (2019). Behavior tree design of intelligent behavior of non-player character (NPC) based on Unity3D. *Journal of Intelligent and Fuzzy Systems*, 37(5), 6071–6079.  
<https://doi.org/10.3233/JIFS-179190>
- Zou, Q., Li, Q., Guo, H., & Shi, J. (2018). A discrete-time and finite-state Markov Chain model for association football matches. *Communications in Statistics: Simulation and Computation*, 47(8), 2476–2485. <https://doi.org/10.1080/03610918.2017.1348518>

## LAMPIRAN

### Lampiran 1 Kuesioner Penelitian

#### Pertanyaan Kuesioner:

Pada bagian ini responden dapat mengisi sesuai dengan apa yang didapatkan oleh responden pada setiap pertanyaan.

Keterangan:

1 = Sangat Tidak Setuju

2 = Tidak Setuju

3 = Netral

4 = Setuju

5 = Sangat Setuju

No.	Pertanyaan	Pilihan Jawaban				
		1	2	3	4	5
1	Seberapa variatif NPC yang dihadapi?					
2	Seberapa realistis NPC yang dihadapi?					