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

Hendra Winata, Liliana, Hans Juwiantho
Program Studi Informatika Fakultas Teknologi Industri Universitas Kristen Petra
Jl. Siwalankerto 121-131 Surabaya 60236
Telp. (031) – 2983455, Fax. (031) - 8417658
E-Mail: hendrawinata13@gmail.com, lilian@petra.ac.id, hans.juwiantho@petra.ac.id

ABSTRAK

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 buatannya 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 survival, artificial intelligence*

ABSTRACT

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 survival, artificial intelligence*

1. PENDAHULUAN

Game Survival Shooter adalah *game* bertema *open-world survival*. Pada *game survival*, pemain akan bertahan hidup dengan cara mengalahkan semua musuh yang ada agar mencapai suatu tujuan dari sebuah misi [2]. Dalam sebuah *game*, terdapat *NPC* yaitu karakter yang dikendalikan oleh komputer dan biasanya berupa lawan dari pemain [6]. Perilaku *NPC* dipengaruhi oleh *AI* yang sangat menentukan tingkat kesulitan karena semakin variatif *NPC* semakin susah juga untuk ditebak perilaku yang dihasilkan [8].

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 [14]. 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) [4]. *NPC* yang menggunakan *FSM* dinyatakan layak untuk digunakan dengan interpretasi baik. *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. *Markov Chain* adalah model universal untuk memprediksi state dari sistem diskrit [1]. *Markov Chain* umumnya digunakan untuk pemodelan state transition dari sistem stokastik yang kompleks [13]. *Markov Chain* dibuat dengan menggunakan matrik transisi probabilitas [1]. 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 [11]. 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 bervariatif sehingga jika bisa diterapkan pada *game* yang memiliki *fast pace* seperti *game survival*, maka akan membuat game lebih unik dan membuat pemain ingin mencoba

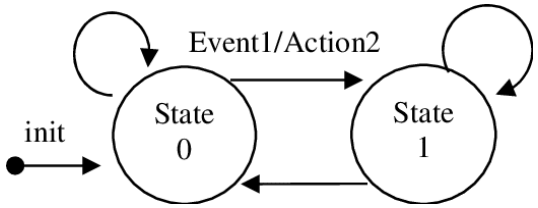
memenangkan permainan dengan mengalahkan *NPC* yang tidak terduga. Pada penelitian ini, pengembangan yang akan dilakukan adalah menggabungkan *Markov Chain* yang menggunakan probabilitas ke dalam *game survival* dengan menggunakan 3 variabel yaitu kesehatan, jarak, dan kecepatan yang bersifat universal sebagai variabel dasar untuk *game 3D survival* dan akan dibandingkan dengan *AI* dari *NPC* yang hanya menggunakan *FSM*.

2. DASAR TEORI

2.1 *Finite-State Machine*

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) [3][4]. *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* [10][12]. 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 [4].

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. Hal ini bisa dilihat dari contoh Gambar 1.



Gambar 1. Gambar *State* Sederhana

2.2 *Markov Chain*

Markov Chain adalah suatu metode *universal* untuk memprediksi *state* dari sistem diskrit [1]. *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 [15]. Dalam analisis *Markov* yang dihasilkan adalah suatu informasi probabilistik 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 [7]. 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 [5]. *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 mxn yang akan dikalikan dengan sebuah variabel [9]. Rumus dari *Markov Chain* dapat dituliskan sebagai berikut.

$$Pr(Xn+1 = x \mid X1 = x1, X2 = x2, \ldots, Xn = xn) = Pr(Xn+1 = x \mid Xn = xn) \quad (1)$$

Dimana:

Pr : Probabilitas
Xn : *Random Variabel*
x : Kondisi saat ini

Rumus tersebut berlaku jika kondisi probabilitas telah ditentukan dan memenuhi syarat

$$Pr(X1 = x1, \ldots, Xn = xn) > 0. \quad (2)$$

Angka yang mungkin dari X1 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 matriks yang disebut *transition matrix* dengan setiap elemen yang dapat didefinisikan dengan rumus berikut.

$$Pij = Pr(Xn+1 = j \mid Xn = i) \quad (3)$$

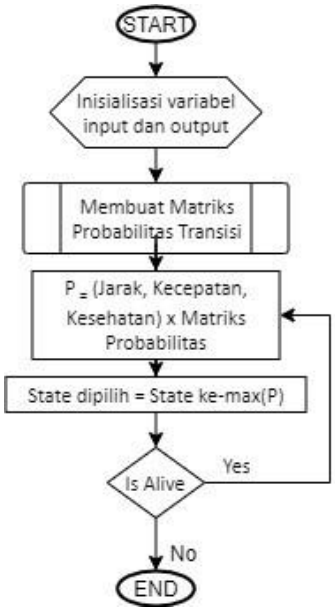
Dimana :

Pij : Probabilitas pada element (i,j)
j : kolom matriks
i : baris matriks

3. DESAIN SISTEM

3.1 Flowchart

Flowchart *Markov Chain* sebagai berikut.



Gambar 2. Flowchart Markov Chain

Pada Gambar 2, 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*.

Flowchart Gameplay sebagai berikut.

Gambar 3. *Flowchart Gameplay*

Permainan akan dimulai dengan alur seperti pada Gambar 3 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*.

Pada *Flowchart FSM* pada Gambar 4 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 variabel *Boolean* tesebut. 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.

3.2 Proses *Markov Chain*

Berikut contoh deklarasi variabel *input* berupa jarak *NPC* terhadap pemain, kesehatan *NPC*, dan kecepatan dari *NPC*. Deklarasi variabel *output* berupa *state idle*, *patrol*, *suicide*, mengejar, menyerang, menembak, dan melarikan diri yang dapat terlihat seperti pada Tabel 1. Kemudian mengubah nilai dari variabel *input* dan *output* kedalam bentuk probabilitas dengan cara dibagi dengan total tiap variabel *input* seperti pada Tabel 2.

Tabel 1. Nilai Variabel

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

Tabel 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.

Tabel 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, maka nilai diatas merupakan nilai maksimal pada masing-masing *sub* variabel. Nilai terbesar merupakan kondisi pemain dimana jarak *NPC* terhadap pemain masih jauh, kesehatan *NPC* yang prima, dan kecepatan *NPC* 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*.

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

[Npatrol Nkejar Nserang Nmenembak Nmelarikandiri Nsuicide] =

[Jr Ks Kc] x Matriks Probabilitas Transisi

Keterangan:

Jr : Jarak pemain terhadap *NPC*

Ks : Kesehatan *NPC*

Kc : Kecepatan *NPC*

Contoh dari perhitungan diatas adalah sebagai berikut:

Diketahui pemain dengan kondisi sebagai berikut:

Jarak pemain terhadap *NPC* = 70, Kesehatan *NPC* = 50, Kecepatan *NPC* = 30

Maka akan didapat nilai variabelnya sebagai berikut:

Jr = 70/150=0.467

Ks = 50/150=0.333

Kc = 30/150=0.2

$$[Jr\ Ks\ Kc] = [0.467\ 0.333\ 0.2] \times \begin{bmatrix} 0.3 & 0.25 & 0.15 & 0.2 & 0.05 & 0.05 \\ 0.4 & 0.3 & 0.1 & 0.1 & 0.08 & 0.02 \\ 0.4 & 0.25 & 0.1 & 0.15 & 0.03 & 0.07 \end{bmatrix}$$

$$= [0.3533 \quad 0.26665 \quad 0.12335 \quad 0.1567 \quad 0.05599 \quad 0.04401]$$

Gambar 5. Perhitungan Probabilitas Tiap *State*

Dari hasil perhitungan pada Gambar 5, maka didapatkan nilai probabilitas terbesar ada pada *patrol* dengan nilai 0.3533. Maka *NPC* akan melakukan *patrol*.

4. PENGUJIAN SISTEM

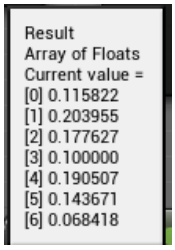
Pada penelitian ini menggunakan *visual scripting* dengan *Unreal Engine* 4.26 yang dapat diunduh melalui https://www.unrealengine.com/en-US/. Hardware yang dipakai saat ini memiliki spesifikasi *processor* Intel Core-i5 dengan tipe 4690, *RAM* sebesar 12 GB serta kartu grafis Nvidia GeForce GTX 750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

4.1 Pengujian Variasi

Pengujian *game* menjelaskan fungsi-fungsi dari proses *Markov Chain* yang diterapkan ke dalam *game* dengan menggunakan 3 *input* variabel. Variabel yang digunakan sebagai input yaitu jarak, kesehatan, dan kecepatan sedangkan untuk variabel *output* yaitu *state idle*, *patrol*, mengejar, menyerang, melarikan diri, *suicide*, dan menembak. 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 6.

Gambar 6. State Patrol

Pada kondisi ini didapatkan nilai dari *cur_distance* atau jarak dari pemain terhadap *NPC* sebesar 8.59, cur_*speed* atau kecepatan dari *NPC* sebesar 0, dan *cur_health* atau darah dari *NPC* sebesar 100.



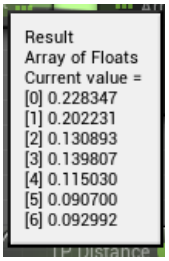
Gambar 7. Hasil Perkalian Markov Chain dengan Cur Variabel

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 7.

Dari hasil *Markov Chain* yang pertama, lalu dilakukan perubahan pada jarakpemain terhadap *NPC*, kecepatan *NPC*, dan kesehatan dari *NPC* shingga didapatkan *state idle* seperti pada Gambar 8.

Gambar 8. State Idle

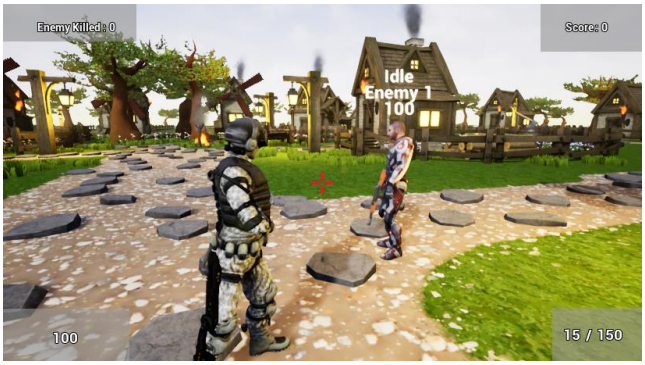
Pada kondisi ini didapatkan nilai dari *cur_distance* atau jarak dari pemain terhadap *NPC* sebesar 2.80, *cur_speed* atau kecepatan sebesar 49.99, dan *cur_health* sebesar 10.



Gambar 9. Hasil Perkalian Markov Chain dengan Cur Variabel

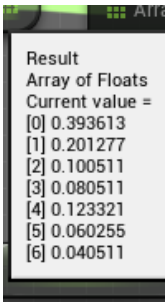
Dari hasil *current* variabel dikalikan dengan matriks probabilitas sehingga didapatkan masing-masing probabilitas untuk variabel *output*, kemudian dipilih probabilitas terbesar pada indeks ke-0 yaitu *idle* pada Gambar 9.

Pada percobaan kedua, digunakan matriks probabilitas yang berbeda dari sebelumnya, di awal permainan, *NPC* akan memilih *state idle* seperti pada Gambar 10.



Gambar 10. State Idle

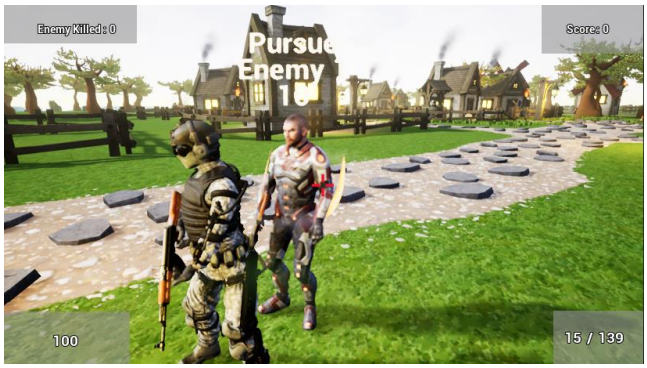
Pada kondisi ini didapatkan nilai dari *cur_distance* atau jarak *NPC* terhadap pemain sebesar 2.62, *cur*_speed atau kecepatan sebesar 0, dan *cur_health* atau darah dari *NPC* sebesar 100.



Gambar 11. Hasil Perkalian Markov Chain dengan Cur Variabel

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* pada Gambar 11.

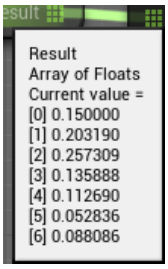
Dari hasil yang pertama, lalu dilakukan perubahan pada jarak, kecepatan, dan kesehatan dari pemain sehingga didapatkan *state pursue* seperti pada Gambar 12.



Gambar 12. State Pursue

Pada kondisi ini didapatkan nilai dari *cur_distance* atau jarak pemain terhadap *NPC* sebesar 4.08, *cur_speed* atau kecepatan sebesar 49.99, dan *cur_health* atau darah *NPC* sebesar 10.

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 13.



Gambar 13. Hasil Perkalian Markov Chain dengan Cur Variabel

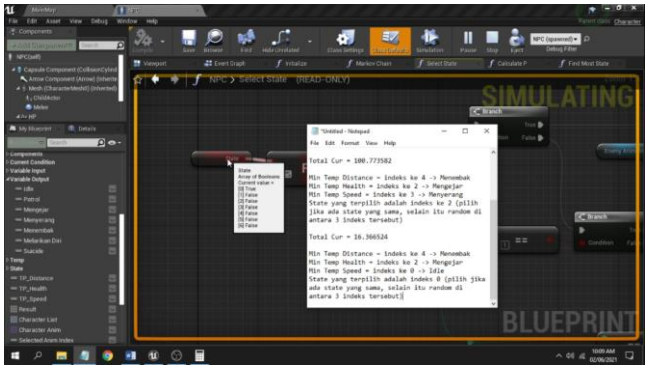
4.2 Pengujian *Main Menu*

Terdapat dua *button* utama pada bagian *main menu*, yaitu *campaign button* dan *exit button*. 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*. Hal ini bisa dilihat pada Gambar 14.

Gambar 14. Pengujian Main Menu

4.3 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* pada Gambar 15.

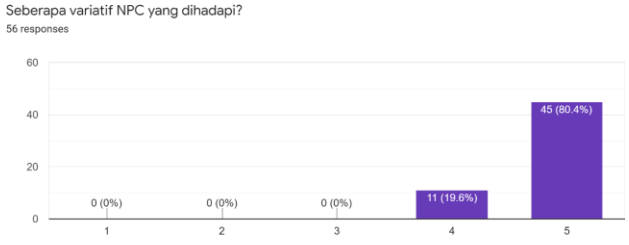


Gambar 15. Pengujian Gameplay

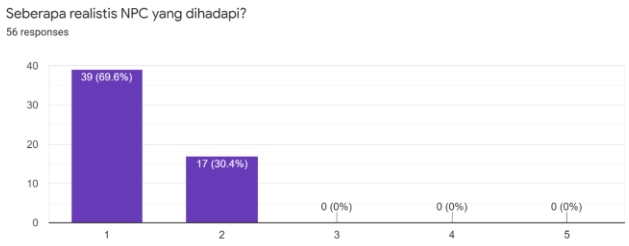
Pada proses tersebut, nilai dari variable kesehatan, kecepatan, dan jarak mempengaruhi *AI* dalam memilih *state*. Tanpa adanya variable tersebut maka hasilnya selalu statis pada satu state saja. Nilai dari variable kesehatan, kecepatan, dan jarak selalu berubah setiap saat dan membuat hasil perhitungan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

4.4 Pengujian Terhadap User

Dilakukan pengujian terhadap user menggunakan survey dan mendapatkan hasil seperti pada Gambar 16 tentang banyaknya variatif tindakan dari *NPC* dan Gambar 17 tentang seberapa realistis tindakan yang dihasilkan *NPC* berdasar kondisi yang dihadapi.



Gambar 16. Survey Terhadap Banyaknya Variatif Tindakan *NPC*



Gambar 17. Survey Terhadap Seberapa Realistis Tindakan yang Dihasilkan *NPC* Terhadap Kondisi yang Dihadapi

5. KESIMPULAN DAN SARAN

5.1 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 bervariatif.

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.

5.2 Saran

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

a) Penambahan variabel *output* sehingga lebih banyak state yang bervariasi.

b) Penggunaan *Markov Chain* disesuaikan kembali agar tetap bervariatif namun lebih realistis.

6. REFERENSI

[1] 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.

[2] 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.

[3] Hernawan, S. R. 2018. Penerapan Metode Finite State Machine Pada Game " The Mahasiswa " Guna Membangun Perilaku Non Playable Character Halaman Pengesahan Dosen Penguji Penerapan Metode Finite State Machine Pada Game " The Mahasiswa " Guna Membangun Perilaku Non Playable Char. *Jurnal Edukasi Dan Penelitian Informatika (JEPIN)*, Retrieved from https://jurnal.untan.ac.id/.

[4] 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

[5] Kasse, I., Didiharyono, 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

[6] 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

[7] Sri, U. , Wayan, M. , Purnaba, I. G. P. (2018). Evaluasi Numerik Penduga Fungsi Nilai Harapan Dan Fungsi Ragam Proses Poisson Majemuk Dengan Intensitas Eksponensial Fungsi Linear. *Journal of Mathematics and Its Applications*, 17(2), 157-169. https://doi.org/10.29244/jmap.17.2.157-169.

[8] Saprudin, S., Liliasari, 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

[9] 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

[10] Tirtana, A., & Pumpungan, M. (2020). Perancangan Game Visual Novel " Coconut Kids " Sebagai Sarana Edukasi Pelestarian Alam. *Universitas 17 Agustus 1945 Surabaya*. Retrieved from http://repository.untag-sby.ac.id/id/eprint/5224.

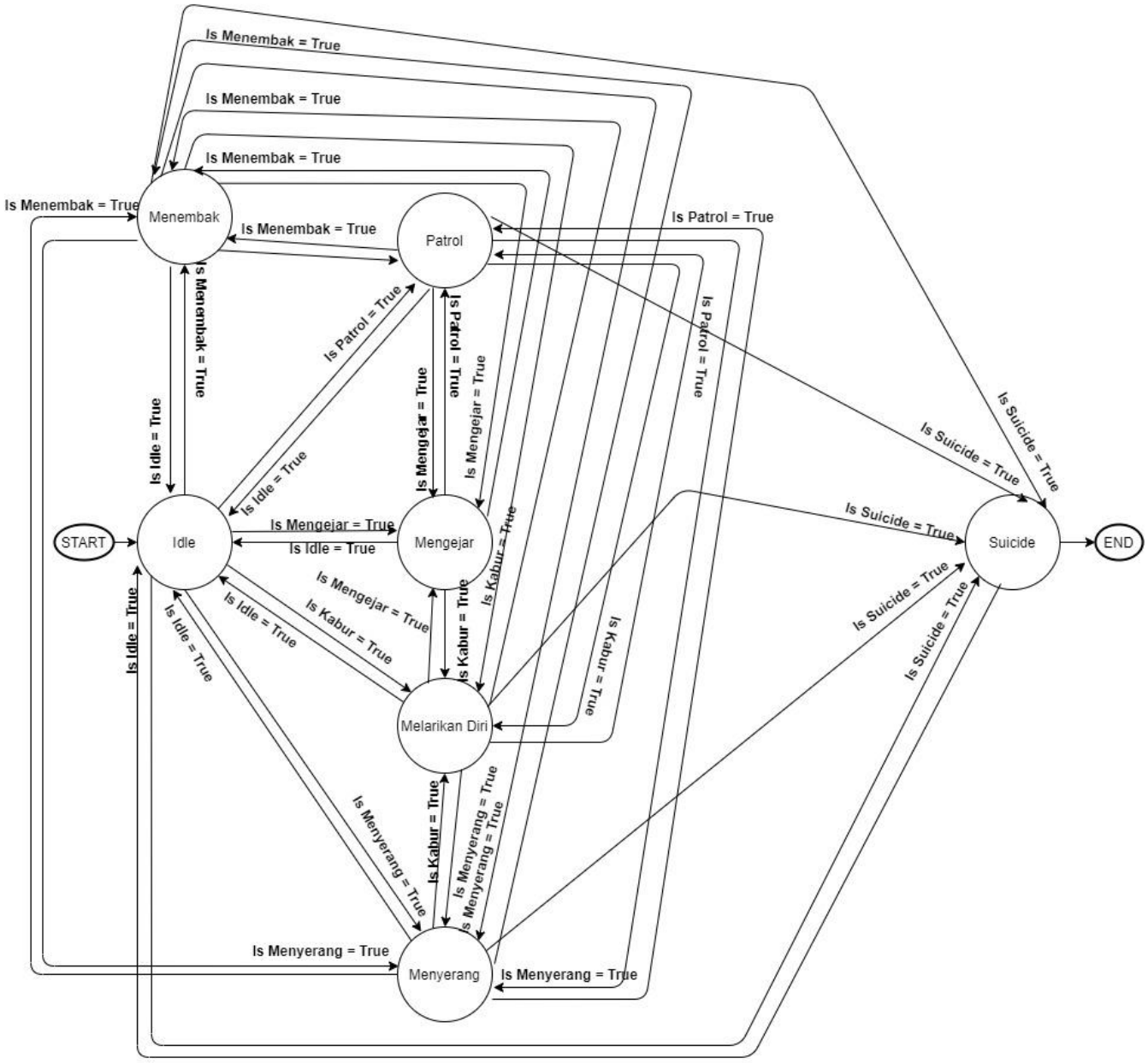
[11] Wang, L., Huang, W., Li, Y., Evans, J., & He, S. (2020). Multi-AI competing and winning against humans in iterated rock-paper-scissors game. *Sci Rep* 10, 13873 (2020). https://doi.org/10.1038/s41598-020-70544-7.

[12] 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

[13] 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

[14] 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

[15] 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



Gambar 4. Flowchart FSM