Implementasi Algoritma *Collision Detection* dan *Markov Chain* untuk Menentukan *Behaviour* NPC dan Karakter *Player* pada *Game Higeia*

Sulistyanto Laili R [1], Dodik Arwin Dermawan [2]
[12]Jurusan Teknik Informatika Fakultas Teknik Universitas Negeri Surabaya
[1]sulistyanto.17051204026@mhs.unesa.ac.id,
[2]dodikdermawan@unesa.ac.id

Abstrak— ***Adventure Game* menjadi salah satu *genre game* terbanyak yang dimainkan sejak video *game* dikembangkan. Di dalam *adventure game player* dilatih untuk berfikir menguraikan tempat secara visual, menyelesaikan permasalahan, dan juga merangkai kesimpulan dari peristiwa berdasarkan percakapan maupun benda-benda yang terdapat di dalam *scene game*. Penelitian ini bertujuan untuk membuat sebuah *game* dalam bentuk *adventure game* bertema kesehatan menggunakan Unity dengan nama *game "Higeia". Game* ini dibuat menggunakan gabungan metode dari *Markov Chain* dan *Collision Detection* dimana metode *Markov Chain* digunakan sebagai klasifikasi perilaku penyerangan karakter NPC sedangkan metode *Collision Detection* digunakan untuk memberikan keputusan perilaku ketika karakter *player* menabrak *obstacle* di dalam *gameplay*. Dimana perilaku NPC akan dibagi menjadi tiga perilaku yaitu perilaku patrol, kejar dan serang. Variabel yang digunakan ialah kecepatan (Kc), jarak (Jr), dan nyawa (Ny) dari kondisi karakter *player*. Dari hasil pengujian menggunakan perhitungan metode *markov chain* dengan masukan 15 data nilai awal kondisi *player* didapatkan hasil perilaku NPC 40% NPC melakukan patrol, 13,33% NPC melakukan kejar dan 46,67% NPC melakukan aksi tembak dengan *event collision detection* yang berfungsi sesuai dengan yang diharapkan.**

Kata Kunci*— *Game, Collision Detection, Markov Chain, NPC, Artificial Intelligence

I. PENDAHULUAN

Game (permainan) merupakan sebuah terobosan teknologi yang bisa dimainkan oleh *player* dan dibuat sebagai sarana hiburan dalam bentuk multimedia yang banyak diminati oleh anak-anak hingga orang dewasa. Seiring banyaknya peminat *game*, muncul berbagai macam genre *game* seperti *Survival Game*, RPG (*Role Player Game*), FPS *(First Person Shooter)*, TPS (*Third Person Shooter*) dan masih banyak lainnya [1]. Penggunaan kecerdasan buatan atau AI (*Artificial Intelligence*) sangat diperlukan dalam proses pembuatan *game*, kecerdasan buatan umumnya digunakan untuk mengatur pergerakan karakter *player*, karakter NPC, penetapan *reward* dan sebagainya [2].

Dalam pembuatan *gameplay* aspek terpenting yang harus diperhatikan adalah kualitas dari mekanik *game* dan juga alur *game.* Karena masih banyak *game* yang memiliki keterbatasan seperti karakter NPC yang hanya diatur untuk melakukan

direct feedback ketika karakter *player* berada di dalam jangkauannya, sehingga karakter NPC tidak kembali ke *base* dan melakukan patrol ketika kehilangan jejak karakter *player* saat pengejaran [1]. NPC (*Non Playable Character*) adalah semua karakter atau obyek yang berada di dalam *gameplay* yang pergerakannya dijalankan oleh komputer sehingga dapat berinteraksi dengan *player* agar dapat menambah tantangan bermain, karakter NPC harus menggabungkan antara karakter robot dengan karakter manusia yang di kembangkan sehingga ketika karakter NPC dipicu oleh suatu gerakan maka karakter NPC yang telah diatur akan menjadi aktif [3].

[4] pada penelitiannya yang berjudul Desain *Non Playable Character* Sebagai Musuh Pada *Game* Sepeda Menggunakan Metode *Markov State Machine* menyatakan bahwa metode *markov chain* dapat diimplementasikan untuk mengatur pergerakan dari karakter musuh (*Non Playable Character*) dengan baik, sehingga karakter musuh dapat melakukan perhitungan aksi berdasarkan dari kondisi karakter *player* saat bertemu karakter musuh. Penelitian ini menciptakan sebuah *game* simulasi *gowes* yang digambarkan seseorang mengendarai sepeda berkeliling hutan untuk mencari *obstacle* (rintangan) yang memiliki poin. Meninjau dari penelitian [4] terdapat kekurangan yaitu metode *markov state machine* hanya diterapkan pada NPC berupa seorang pekerja, sehingga ketika *collider* karakter player bersentuhan atau bertabrakan dengan *obstacle non-point* maka tidak terjadi *event* apapun, hal tersebut menjadikan *game* kurang menarik dan menantang.

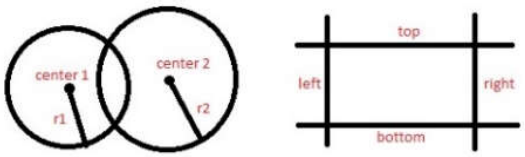
Berdasarkan masalah dari penelitian sebelumnya maka penelitian ini akan membahas perancangan sebuah aplikasi *game* yang berjudul "Implementasi Algoritma *Collison Detection* dan *Markov Chain* Untuk Menentukan *Behaviour* NPC dan Karakter *Player* Pada *Game Higeia*". Di dalam penelitian ini peneliti mencoba membuat *game* yang

bertujuan sebagai sarana hiburan dan juga edukasi dengan menerapkan algoritma *markov chain* untuk menentukan pergerakan dari karakter musuhnya dan algoritma *collison detection* untuk mendeteksi tabrakan antara karakter *player* dengan karakter musuh maupun dengan elemen pendukung di dalam *gameplay*. Penerapan algoritma *collision detection* akan memungkinkan karakter dalam *game* untuk mendeteksi objek di sekitarnya dan membuat aksi semakin mendekati nyata [5]. *Game Higeia* ini merupakan *game adventure* (petualangan) yang mengambil tema kesehatan atau *higeia* dalam bahasa Yunani, berfokus pada pengumpulan skor dan melawan karakter musuh. Penelitian ini akan menjabarkan gabungan metode *collision detection* dan juga *markov chain* yang diharapkan dapat berguna untuk pengembangan penelitian sejenis.

A. *NPC (Non Playable Character)*
Non Playable Character adalah semua karakter yang berada di *gameplay* yang tidak dapat dimainkan oleh *player*. NPC merupakan jenis *autonomous agent* yang cara kerjanya ialah dijalankan oleh komputer sehingga dapat berinteraksi dengan *player* di dalam *game*. Dikategorikan dalam jenis *autonomous agent* karena NPC memiliki kemampuan untuk improvisasi dalam tindakan dan perilakunya [3]

B. *Algoritma Markov Chain*
Markov Chain (Rantai Markov) merupakan metode yang membahas sifat dari suatu variabel pada kondisi saat ini berdasarkan sifat-nya terdahulu agar kemudian digunakan untuk menaksir sifat-sifatnya di masa depan. *Markov chain* memiliki *expositions stokastik* dimana kondisi terdahulu tidak memiliki pengaruh di masa depan apabila kondisi saat ini diketahui. Dengan menggunakan metode *markov chain* hasil yang didapat adalah suatu informasi berupa kemunkinan yang berfungsi untuk mendukung menentukan aksi. Sehingga dapat disimpulkan bahwa metode tersebut bukan merupakan metode optimisasi melainkan metode deskriptif [1].

C. *Algoritma Collision Detection*
Collision Detection adalah suatu proses pemeriksaan antara dua atau lebih objek spasial saling bertabrakan atau tidak dalam bidang koordinat tertentu. Pada dasarnya untuk mempermudah pengenalan proses *collision detection* biasanya objek-objek yang berkaitan direpresentasikan secara visual melalui bentuk segiempat dan lingkaran (untuk koordinat dua dimensi) atau kubus dan bola (untuk koordinat tiga dimensi). Bentuk-bentuk yang mwakili objek-objek ini sering disebut *Bounding Box* atau *Bounding Circle* [6].



GBR 1. BOUNDING CIRCLE DAN BOUNDING BOX

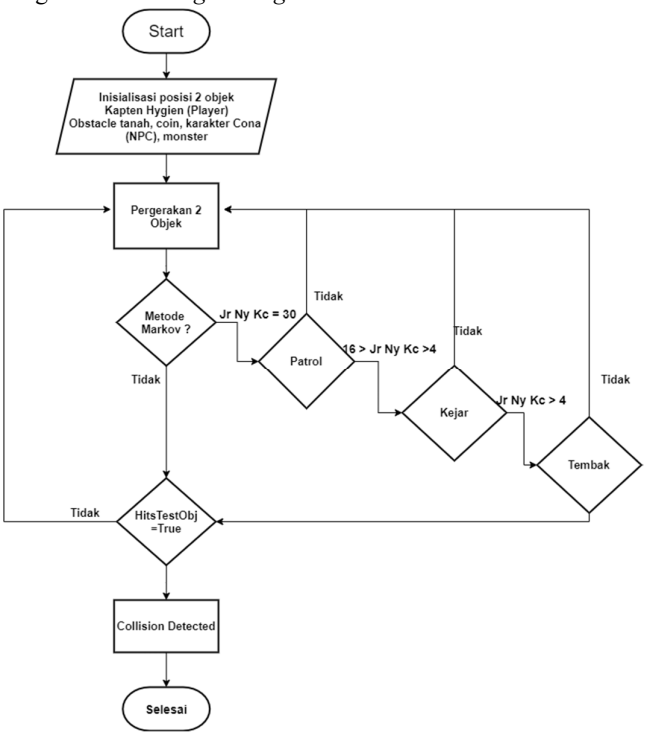
II. METODE PENELITIAN

Dalam pelaksanaan penelitian diperlukan sebuah diagram alur yang digunakan sebagai acuan agar hasil dari penelitian sesuai dengan tujuan awal yang ingin dicapai. Rancangan diagram alur yang diterapkan untuk penelitian ini terdapat pada Gbr 2.

GBR 2. DIAGRAM ALUR METODOLOGI PENELITIAN

Penelitian ini difokuskan pada pengimplementasian metode *markov chain* dan metode *collision detection* pada *game Higeia* untuk mengatur pergerakan dari *NPC* sehingga pergerakan di dalam gameplay menjadi lebih menarik. Metode *markov chain* digunakan untuk menentukan aksi pergerakan dari NPC musuh antara patrol, kejar, dan serang. Sedangkan *collision detection* digunakan untuk membuat

karakter utama *player* menjadi lebih realistis dengan *event collision* yang akan ditentukan pada bab ini. Berikut merupakan *flowchart* dari penggabungan dua metode yang digunakan dalam *game higeia*.



GBR 3. FLOWCHART PROSES MARKOV CHAIN DAN COLLISION DETECTION

A. Analisis Metode Markov Chain

Dalam penelitian ini operasi *markov chain* yang dmanfaatkan adalah matriks probabilitas transisi yang berfungsi untuk memperkirakan peluang kejadian di waktu yang akan datang ketika *player* berada dalam situasi saat ini. Rumus perhitungan probabilitas aksi dari NPC yang digunakan adalah sebagai berikut :

$$[Jr\ Ny\ Kc] = [\ Jr\ Ny\ Kc\ ]\ x\ Matriks\ Probabilitas\ Transisi$$

Keterangan :

Jr = Jarak antara *player* dengan NPC

Ny = Nyawa dari *player*

Kc = Kecepatan pergerakan *player*

Berdasarkan tiga masukan dari variabel Jarak (Jr), Nyawa (Ny), Kecepatan (Kc) nantinya yang akan memutuskan keluaran pergerakan dari variabel perilaku NPC untuk melakukan patrol, pengejaran ataupun penyerangan pada karakter player di dalam *game*.

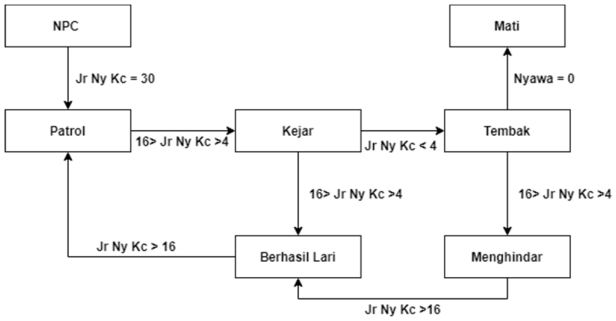
1. Tahap Penentuan Rule Perilaku NPC

Perilaku NPC ditetapkan menggunakan *range* maksimum dan minimum dari nilai suatu parameter pada tabel berikut.

Kondisi	Perilaku
Range 16-30	Patrol
Range 4-15	Kejar
Range 0-3	Serang

TABEL 1. RULE BASE PERILAKU NPC

Pengimplementasian pergerakan dari NPC diilustrasikan seperti dalam *box* diagram dibawah ini.



GBR 4. DIAGRAM IMPLEMENTASI RULE NPC

2. Tahap Perancangan Sistem

Game yang dibangun merupakan *game* ber*genre adventure game*, dimana terdapat sebuah karakter utama yang akan melakukan sebuah misi penyelamatan sebuah kota yang ditinggalkan oleh penduduknya karena kota tersebut terdampak virus yang mematikan, untuk mengalahkan virus-virus tersebut Kapten Hygien harus mengumpulkan 7 bahan ajaib agar kemudian dapat mengalahkan Cona. Tantangan pada game ini terdapat pada NPC yang akan diletakkan di beberapa *gamescene* untuk menggagalkan misi Kapten Hygien kemudian pada stage terakhir Kapten Hygien harus merebut 5 bahan ajaib yang tersisa dari tangan Cona yang merupakan NPC utama. Pergerakan karakter Cona inilah yang menjadi objek utama dalam penelitian ini. Untuk mengenal lebih jauh berikut merupakan beberapa karakter yang akan muncul di dalam *game* ini.

No	Nama	Karakter
1.	Kapten Hygien	
2.	Jack	
3.	Penduduk	
4.	Karakter Musuh	
5.	Monster	

TABEL 2. TABEL KARAKTER GAME HIGEIA

3. Tahap Perancangan Tabel
   Tabel ini dirancang untuk menentukan pengoperasian NPC berdasarkan kondisi saat ini. Kondisi yang dimaksudkan berupa inputan nilai awal dari variabel NPC, setelah nilai awal ditetapkan tahap selanjutnya adalah menghitung probabilitas transisi untuk menentukan probabilitas yang akan terjadi di waktu yang akan datang.
   a. Tabel Nilai Awal
      Tabel di bawah merupakan nilai awal yang telah di inputkan. Nilai dari inputan berbeda-beda dipengaruhi oleh kondisi *player* di perjalanan sehingga membuat respon perilaku NPC berubah ketika berhadapan dengan *player*.

No	Input			Jumlah
	Jr	Ny	Kc	
1	30	30	2	62
2	25	20	15	60
3	28	30	0	58
4	10	30	7	47
5	25	15	2	42
6	10	30	0	40
7	9	6	15	30
8	15	8	2	25
9	4	12	7	23
10	13	3	4	20
11	0	15	5	20
12	8	7	5	20
13	12	2	3	17
14	4	10	0	14
15	5	5	2	12

TABEL 3. NILAI AWAL VARIABEL

   b. Tabel Variabel Parameter
      Tabel variabel parameter diperlukan untuk mengubah nilai aksi NPC yang masih berupa *range* menjadi nilai yang mutlak. Dari tabel inilah nantinya bisa dilihat NPC akan melakukan patrol, kejar atau tembak.

Input	Output			
	Patrol	Kejar	Tembak	Jumlah
Jr	25/42	15/42	2/42	42
Ny	15/33	15/33	3/33	33
Kc	20/33	10/33	3/33	33

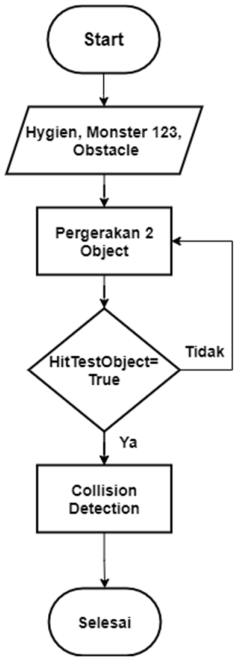
Tabel 4. Tabel Variabel Parameter

Tabel parameter diatas kemudian dijadikan bilangan riil tak negatif sebagai syarat matriks probabilitas transisi, dengan hasil matriks probabilitas transisi sebagai berikut :

0,5952 0,3571 0.0476
0,4545 0,4545 0,0909
0,6060 0,3030 0,0909

Setelah perhitungan selesai tahap menentukan aksi NPC terhadap *player* dilakukan dengan menghitung nilai terdekat dari nilai parameter.

B. Analisis Metode *Collision Detection*
Game Higeia mengimplementasikan *bounding box* sebagai *collider* karakter *player*nya. *Collider* ini berfungsi sebagai *initiator* ketika karakter *player* bertabrakan dengan objek lain didalam *game* sehingga akan muncul respon yang membuat pergerakan *player* semakin *real*. Deteksi *collision* akan dimulai ketika *collider* karakter *player* mulai bergerak dan saling menyentuh dengan *collider* objek lain, kemudian persamaan algoritma akan mengecek apakah dua objek tersebut bertabrakan atau tidak. Apabila dua objek bertabrakan maka *event* yang telah ditetapkan akan muncul, namun apabila tidak bertabrakan maka karakter *player* dapat melanjutkan pergerakannya. Untuk membuat *game* menjadi lebih *attractive behaviour* dari NPC musuh akan dibagi menjadi dua, sehingga alur proses penerapan *collision detection* pada *game Higeia* dapat dilihat pada Gbr 5.



GBR 5. DIAGRAM ALUR COLLISION DETECTION

Setelah mengetahui diagram alur dari metode *collision detection* maka berikut *event collision detection* yang diterapkan pada *game Higeia*.

1. Ketika karakter *player* bertabrakan dengan ikon poin, maka variabel poin akan bertambah dan memainkan *sound coin.wav*
2. Ketika karakter *player* bertabrakan dengan objek *surprise box*, maka akan muncul *item* herbal1 dan memainkan *sound* kejutan1.*wav*
3. Ketika karakter *player* bertabrakan dengan monster1, monster2, monster3, maka variabel nyawa akan berkurang dan memainkan sound kurangnyawa.*wav*
4. Ketika karakter *player* bertabrakan dengan item nyawa, maka variabel nyawa akan bertambah dan memainkan sound tambahnyawa.*wav*

III. HASIL DAN PEMBAHASAN

Pada penelitian ini diterapkan dua metode yaitu metode *markov chain* dan *collision detection*. Bab ini akan membahas mengenai implementasi dari perancangan yang sudah dibuat pada metode penelitian.

A. Implementasi Game

Pada tahap implementasi semua *asset* yang telah dibuat dan disiapkan akan diimplementasikan menjadi *game Higeia*. Selain pengimplementasian *interface* dari *gameplay* pada tahap ini juga dilakukan pengimplementasian dari metode *markov chain* dan juga *collision detection*. Berikut merupakan implementasi tampilan dari *game Higeia.*

GBR 6. TAMPILAN AWAL GAME HIGEIA

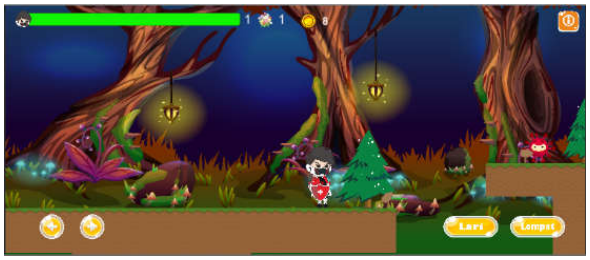
Pada Gambar 6 terlihat tampilan awal dari *game Higeia*. *Game Higeia* ini merupakan *game adventure* bertema kesehatan dimana karakter utama yang dimainkan oleh *player* memiliki misi untuk membebaskan sebuah kota dari serangan virus yang dibuat oleh Cona (karakter npc).

GBR 7. TAMPILAN STAGE 1-1 GAME HIGEIA

Di dalam *stage* 1-1 ini metode yang digunakan adalah metode *collision detection* dimana ketika *collider* dari dua karakter bersentuhan maka akan memicu *event* sebuah percakapan. Dimana isi dari percakapan tersebut berfungsi sebagai pembangun alur cerita dari *game Higeia* ini.

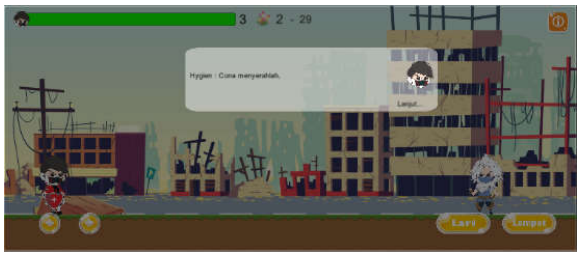
GBR 8. TAMPILAN STAGE 1-2 GAME HIGEIA

Di dalam *stage* 1-2 metode yang digunakan adalah metode *collision detection* dimana metode ini diterapkan pada tiga objek yaitu, karakter *player*, koin, dan juga NPC musuh1. *Event* yang terjadi pada *stage* ini penambahan poin ketika karakter *player* bertabrakan dengan *object* koin, pengurangan nyawa ketika karakter *player* bertabrakan dengan NPC musuh1.



GBR 9. TAMPILAN STAGE 2 GAME HIGEIA

Di dalam *stage* 2 metode yang digunakan adalah metode *collision detection* dimana metode ini diterapkan pada karakter *player*, koin, objek bunga, dan juga NPC penduduk. Apabila *collider* dari karakter *player* bersentuhan dengan *collider* NPC penduduk maka akan terjadi sebuah *event* percakapan.



GBR 10. TAMPILAN FINAL STAGE GAME HIGEIA

Di dalam *stage* 3 metode yang digunakan merupakan gabungan dari metode *markov chain* dan *collision detection*. Dimana metode ini akan di terapkan pada Cona (Karakter NPC) sebagai musuh utama di dalam *game Higeia* ini. *Even*t yang terjadi pada *stage* 3 apabila karakter *player* berada pada jarak jangkauan dari NPC maka NPC akan menentukan tindakan berdasarkan metode *markov chain* yang telah diimplementasikan. Tindakan yang dapat dilakukan oleh NPC diantaranya patrol apabila karakter *player* masih berada diluar jangkauan NPC, kejar apabila karakter *player* terdeteksi berada didalam jangkauan NPC, tembak apabila karakter *player* terdeteksi berada pada jarak sangat dekat dari NPC. Apabila karakter *player* gagal menghindari tembakan dari NPC maka nyawa dari karakter *player* akan berkurang.

B. Pengujian Metode *Markov Chain*

Pada tahap ini uji coba skenario digunakan untuk melihat apakah metode *markov chain* yang diimplementasikan sudah berfungsi dengan semestinya. Perhitungan *markov chain* pada *game Higeia* disesuaikan dengan algoritma yang sudah dibahas pada bab metode penelitian yaitu dengan menentukan terlebih dahulu probabilitas transisinya.

Dengan cara perhitungan diatas didapatkan hasil dari beberapa input kondisi awal *player* yang telah disebutkan pada bab metode penelitian. Hasil kali dari nilai awal dengan probabilitas transisi dapat dilihat sebagai berikut.

Output		
Jr	Ny	Kc
0,5274	0,4024	0,0698
0,551	0,37605	0,0728
0,5225	0,4075	0,0699
0,5069	0,41119	0,0816
0,5454	0,3892	0,0651
0,4897	0,4302	0,0801
0,5726	0,3495	0,0779
0,5510	0,3839	0,0594
0,525	0,3913	0,0833
0,5763	0,3609	0,0628
0,4923	0,4165	0,0909
0,5487	0,3776	0,0735
0,5803	0,3659	0,0543
0,4946	0,4266	0,0785
0,5384	0,3887	0,2091

TABEL 5. HASIL KALI MATRIKS PROBABILITAS TRANSISI

Hasil diatas didapat dari nilai awal yang terdapat pada Tabel 3 dikalikan dengan matriks probabilitas transisi yang didapat setelah mengubah nilai parameter menjadi bilangan riil tak negatif sebagaimana terlihat seperti pada Gambar 3 dan Tabel 4. Karena syarat dari probabilitas transisi adalah angka tidak boleh merupakan bilangan negatif dan lebih dari satu maka sebelum nilai awal dikalikan dengan matriks probabilitas transisi nilai awal perlu dibagi dengan keseluruhan total nilai awal *player*, atau bisa dituliskan sebagai berikut.

$$\text{Jr} = \frac{Nilai\ awal\ Jarak}{Total\ Nilai\ Awal}$$

$$\text{Ny} = \frac{Nilai\ awal\ Nyawa}{Total\ Nilai\ Awal}$$

$$\text{Kc} = \frac{Nilai\ awal\ Kecepatan}{Total\ Nilai\ Awal}$$

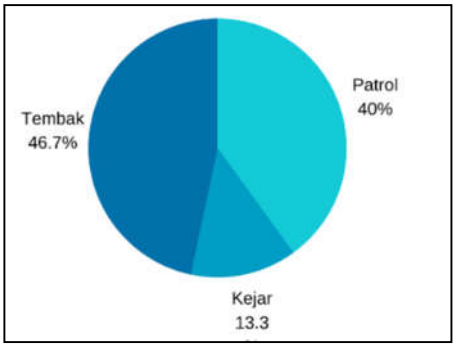
Tahap selanjutnya output dari perkalian matriks pada Tabel 5 akan dikalikan dengan total nilai awal untuk mendapatkan parameter yang nantinya akan menentukan perilaku dari NPC.

Output			Status
Jr	Ny	Kc	
0,5274	0,4024	0,0698	Patrol
0,551	0,37605	0,0728	Patrol
0,5225	0,4075	0,0699	Patrol
0,5069	0,41119	0,0816	Patrol
0,5454	0,3892	0,0651	Patrol
0,4897	0,4302	0,0801	Patrol
0,5726	0,3495	0,0779	Kejar
0,5510	0,3839	0,0594	Kejar
0,525	0,3913	0,0833	Tembak
0,5763	0,3609	0,0628	Tembak
0,4923	0,4165	0,0909	Tembak
0,5487	0,3776	0,0735	Tembak
0,5803	0,3659	0,0543	Tembak
0,4946	0,4266	0,0785	Tembak
0,5384	0,3887	0,2091	Tembak

TABEL 6. OUTPUT AKSI NPC

Tabel 6 merupakan hasil akhir dari perhitungan metode *markov chain*. Hasil tersebut didapat dari perhitungan nilai terdekat dari parameter.

Dari hasil 15 kali percobaan yang telah dilakukan dapat diketahui bahwa kemungkinan pergerakan aksi dari NPC adalah 40% NPC melakukan patrol, 13,33% NPC melakukan kejar dan 46,67% NPC melakukan tembak. Untuk memudahkan melihat hasil *output* aksi dapat digambarkan melalui diagram pada Gbr 11. di bawah ini



GBR 11 . DIAGRAM LINGKARAN OUTPUT AKSI NPC

C. Pengujian Metode *Collision Detection*

Pengujian *Collision Detection* ini dilakukan untuk mengetahui apakah algoritma yang diimplementasikan sudah berjalan dengan semestinya. Berikut merupakan algoritma yang digunakan untuk mengetahui apakah objek di dalam *gameplay* mengalami *collision detection* atau tidak.

```
void Start(){
    rb    = GetComponent<Rigidbody2D>();
    anim  = GetComponent<Animator>();
}

void FixedUpdate()
{
    if(moveInput != 0){
        anim.SetBool("isWalking", true);
    }

    if(moveInput < 0){
        transform.eulerAngles = new Vector3(0, 0, 0);
    }else if(moveInput > 0){
        transform.eulerAngles = new Vector3(0, 180, 0);
    }
    rb.velocity = new Vector2(moveInput * 3, rb.velocity.y);
}

void OnCollisionEnter2D(Collision2D col) {
    if (col.gameObject.tag.Equals("Player"))
    {
        moveInput = 0;
        anim.SetBool("isWalking", false);
```

GBR 12. ALGORITMA COLLISION DETECTION

Pada algoritma ini menggunakan komponen *rigidbody* pada objek yang berfungsi untuk membuat objek terdeteksi pada *engine,* setelah objek terdeteksi maka objek ini secara otomatis akan terpengaruh oleh *gravity* sehingga dapat dikendalikan melalui *script.* Kemudian apabila ditambahkan dengan *collider* yang sesuai maka objek akan merespon apabila terjadi tabrakan dengan objek lainnya. Untuk mengetahui apakah algoritma tersebut berjalan dengan baik maka algoritma tersebut akan diuji dengan data uji dibawah ini.

No	Event	Keluaran Harapan	Hasil
1	Karakter tabrak coin	Poin bertambah, sound coin.wav	Sesuai
2	Karakter tabrak surprise box	Muncul item herbal, sound surprise1.wav	Sesuai
3	Karakter tabrak NPC musuh123	Nyawa berkurang, sound kurangnyawa.wav	Sesuai
4	Karakter tabrak item nyawa	nyawa bertambah	Sesuai

TABEL 7. TABEL UJI COBA COLLISION DETECTION

Berdasarkan data hasil uji coba diatas penerapan algoritma *collision detection* dapat disimpulkan sudah berjalan dengan semestinya.

IV. KESIMPULAN

Dilihat dari hasil dan pembahasan penelitian ini dapat disimpulkan bahwa penggabungan dua metode *markov chain* dan metode *collision detection* dapat memperbaiki hasil dari penelitian sebelumnya yaitu NPC didalam *game* yang hanya diterapkan pada satu NPC dengan perilaku kejar dan serang sehingga apabila karakter *player* menemui *obstacle non-point* maka tidak terjadi *event* apapun. Hasil penerapan penggabungan metode *markov chain* dan *collision detection* ini membuktikan bahwa algoritma *markov chain* dan *collision detection* membuat pergerakan dari NPC menjadi lebih *real* dan *gameplay* menjadi lebih *attractive*. Dari hasil pengujian menggunakan perhitungan metode *markov chain* dengan masukan 15 data nilai awal kondisi *player* didapatkan hasil perilaku NPC 40% NPC melakukan patrol, 13,33% NPC melakukan kejar dan 46,67% NPC melakukan aksi tembak dengan *event collision detection* yang juga berfungsi sesuai dengan yang diharapkan.

V. SARAN

Berdasarkan perancangan dan implementasi program, *game* yang telah dibuat perlu pengembangan lebih lanjut, oleh sebab itu penulis menyarankan penelitian lanjutan untuk penerapan metode *markov chain* dan *collision detection* agar dapat diimplementasikan secara maksimal. Selain itu juga disarankan untuk menambahkan tantangan pada *gameplay* berupa NPC dengan kemampuan berbeda sehingga *game* menjadi lebih menarik.

UCAPAN TERIMAKASIH

Puji syukur atas rahmat dan ridha Allah SWT yang telah diberikan kepada penulis, sehingga penulis dapat menyelesaikan penelitian ini. Terimakasih juga penulis ucapkan kepada kedua orangtua dan orang-orang disamping penulis yang selalu memberikan dukungan dan semangat kepada penulis untuk menuntut ilmu. Serta terimakasih kepada dosen pembimbing yang sudah membimbing penelitian ini dari awal hingga akhir.

DAFTAR PUSTAKA

[1] D. Gunawan, A. Atthariq, and A. Aswandi, "Meningkatkan Behaviour Npc Pada Game 3d Survival Menggunakan Metode Markov," *J. Infomedia*, vol. 2, no. 1, pp. 7–12, 2017, doi: 10.30811/.v2i1.477.

[2] F. Alamsyah, Wasum, and A. Yunus, "Implementasi Algoritma Collision Detection Dan Finite State Machine Untuk Karakter Musuh Pada," *Rainstek*, vol. 1, no. 2, pp. 8–13, 2019.

[3] Q. Galvane, M. Christie, R. Ronfard, C. K. Lim, and M. P. Cani, "Steering behaviors for autonomous cameras," *Proc. - Motion Games 2013, MIG 2013*, no. June 2002, pp. 71–79, 2013, doi: 10.1145/2522628.2522899.

[4] N. P. Wildan, "Desain Non Playable Character Sebagai Musuh Pada Game Sepeda Menggunakan Metode Markov State Machine," 2016.

[5] Y. Abbas, E. Winarno, P. Studi, T. Informatika, F. T. Informasi, and U. Stikubank, "Perancangan Game Edukasi Pengenalan Angka Dalam Bahasa," no. 2012, pp. 347–352, 2018.

[6] A. Nurdiyanto and E. Winarno, "Penerapan Metode Collision Detection Pada Game Petualangan Menggunakan Aksara Jawa," *Univ. Stikubank*, pp. 978–979, 2018, [Online]. Available: https://www.unisbank.ac.id/ojs/index.php/sendi_u/article/view/5983.

[7] D. A. Dermawan, D. F. Suyatno, and M. S. Hawari, "Simulasi Standard Operational Procedure Laboratorium Komputer Dengan Pemodelan Finite State Machine Pada Perilaku Teknisi," *SCAN - J. Teknol. Inf. dan Komun.*, vol. 14, no. 1, 2019, doi: 10.33005/scan.v14i1.1448.

[8] A. P. Yani, M. Y. Charnelia, and K. Kunci, "SYSTEMIC : Information System and Informatics Journal Rancang Bangun Game Edukasi Media Pembelajaran Bahasa Inggris Menggunakan Pemodelan Finite State Machine," vol. 5, no. 2, pp. 37–43, 2019.

[9] D. V. Apriloka, S. Suyadi, and N. Na'imah, "The Use of Games Virus Hunter in Pandemic COVID-19 Against Development of Early Childhood," *Indones. J. Early Child. Educ. Stud.*, vol. 9, no. 1, pp. 19–23, 2020, doi: 10.15294/ijeces.v9i1.39153.

[10] I. Borovikov, J. Harder, M. Sadovsky, and A. Beirami, "Towards interactive training of non-player characters in video games," *arXiv*, no. Hill, 2019.

[11] L. H. Fasha, F. Fauziah, and M. Gufroni, "Implementasi Algoritma Collision Detection pada Game Simulator Driving Car," *STRING (Satuan Tulisan Ris. dan Inov. Teknol.*, vol. 3, no. 1, p. 58, 2018, doi: 10.30998/string.v3i1.2586.

[12] R. Huang, "Optimizing collision detection in 3D games with model attribute and bounding boxes," *Proc. - 2012 IEEE Symp. Electr. Electron. Eng. EEESYM 2012*, pp. 589–591, 2012, doi: 10.1109/EEESym.2012.6258726.

[13] A. B. G. Sang, P. W. Buana, and I. K. A. Purnawan, "Permainan Edukasi Labirin Virtual Reality Dengan Metode Collision Detection Dan Stereoscopic," *Lontar Komput. J. Ilm. Teknol. Inf.*, vol. 8, no. 2, p. 65, 2017, doi: 10.24843/lkjiti.2017.v08.i02.p01.