Dielektrika – Department of Electrical Engineering University of Mataram
Vol. 10, No. 1, February 2023, pp. 1-13
P-ISSN: 2086-9487, E-ISSN: 2579-650x, (Acredited Sinta – 4, Decree no. 200/M/KPT/2020) 

# Rancang Bangun Prilaku Buatan pada *Non-Player Character* dalam *Game* Pemadam Kebakaran menggunakan *Finite State Machine* dan *Godot Script*

Muhammad Fikriansyah [1], Giri Wahyu Wiriasto [1], A. Sjamsjiar Rachman [1]
[1] Jurusan Teknik Elektro – Universitas Mataram, Jalan Majapahit 62, Mataram, 83115 , Indonesia

**ARTICLE INFO**

**Article history**

Received January 25, 2023
Revised February 05, 2023
Accepted February 22, 2023

**Keywords :**

artificial behaviour;
Non-Player Character;
Fire Extinguisher Game;
Finite state machine;
godot script

**ABSTRACT**

The development of artificial behavior on the object character of a game that is played using a computer device continues to grow. With this artificial behavior, certain characters in the game can move independently according to changes in the flow of the game. Those independent behaviors are known as non-player characters (NPCs). This article describes the development of a role-playing game (RPG) entitled "Fire Extinguisher Game". In this game there is an NPC character "fire monster" as an antagonist whose mission is to defeat human players by burning all objects. The design in developing the behavior design of NPC players uses the Finite State Machine (FSM) method and the implementation uses the godot programming. FSM describes the behavior design of each NPC character in the form of a state diagram. There is a change in state or behavior between players in the scene of the game environment. In its implementation, testing has been carried out to find out how far the game application has been running. This game runs on a desktop and has been tested using the black-box testing method to obtain functional performance tests with parameters including testing the main menu page scene, gameplay page scene and results page scene. The game can run well without any bugs or errors found when the game is played.

**Corresponding Author**:

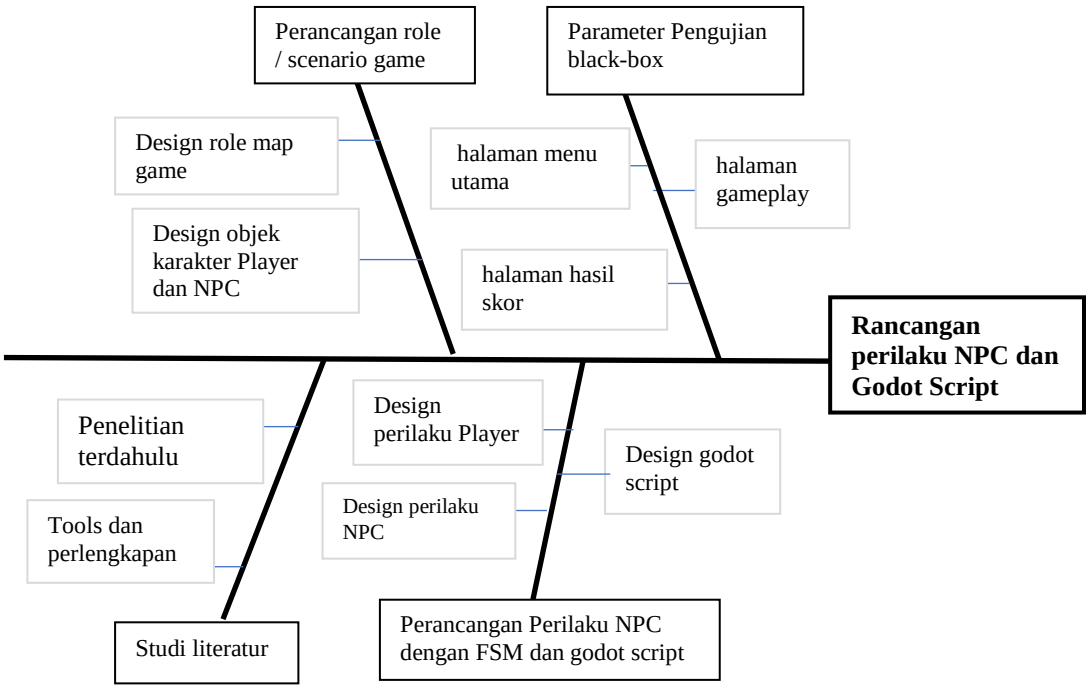
Giri Wahyu Wiriasto, Teknik Elektro Universitas Mataram, Jalan Majapahit 62, Mataram, 83115, Indonesia
Email: giriwahyuwiriasto@unram.ac.id

## 1. PENDAHULUAN

Salah satu unsur penting yang membuat permainan komputer atau game selalu diminati adalah selain role skenario dan design karakternya, unsur penting lainnya yakni perilaku buatan pada komputer player. Perilaku buatan ini dapat membuat karakter pada komputer player bergerak secara autonomus dan terkesan tampak pintar dan atraktif. Karakter komputer player ini dikenal dengan istilah NPC atau non-player charakter. Pergerakan atau perilaku objek karakter NPC dipengaruhi oleh rancangan algoritma yang disematkan padanya oleh pengembang. Semakin tinggi kemampuan adaptasi dari ekosistem role skenarionya maka semakin baik algoritma yang bekerja padanya.

Pada proses perancangannya, digunakan motede *Finite State Machine (FSM).* Dengan metode ini rancangan perilaku buatan yang disematkan pada *NPC* tergambarkan dengan jelas. Metode ini memperlihatkan perilaku sistem dengan berdasarkan tiga hal, yaitu *state* (keadaan), event (kejadian), dan action (aksi) [3]. Pada paper ini dijelaskan tahapan pengembangan permainan dengan kategori *role-playing games* yang diberi nama "*Game Pemadam Kebakaran"*. Dalam permainan ini terdapat karakter *NPC* "monster api" sebagai aktor antagonis yang memiliki misi mengalahkan *human player* yang berupa karakter kartun human 2D sebagai aktor protagonist dengan cara membakar semua objek (rumah, pohon, gedung)

pada *scene* role skenario. Permainan ini berjalan pada desktop dan diuji menggunakan metode *black-box testing* untuk mendapatkan uji kinerjanya secara fungsional. Hasil dari penelitian ini berupa rancangan event perilaku buatan pada *NPC* dengan pendekatan kode *script godot*. Berikut pada Gambar 1 merupakan diagram tahapan rancangan pengembangan game. Beberapa tahapan antara lain studi literatur, perancangan role atau scenario game, perancangan perilaku NPC dengan FSM dan godot script dan tahapan pengujian black-box parameter yang berkaitan dengan keselurahan game.




Gambar 1. Diagram rancangan pengembangan

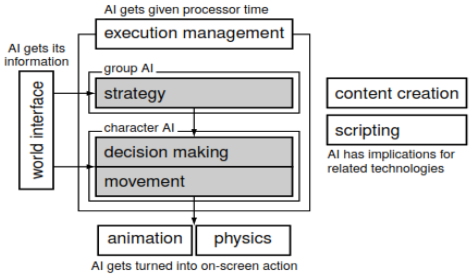
## 2. METODE

Ada beberapa elemen penting dalam pembuatan game yaitu narasi role skenario, karakter, background, aspek multimedia dan algoritma NPC. Secara umum untuk dapat memenuhi elemen-elemen tersebut maka akan diperlukan proses dalam pembuatannya sebagai berikut [3] antara lain, pertama menentukan tema permainan/game. Kedua membuat Gameplay atau aturan yang diperlukan yang kemudian akan dibuat dalam game agar pemain dapat berinteraksi, serta membuat game memiliki ciri khas atau keunikannya sendiri. Ketiga membuat storyboard yang mana storyboard merupaka panduan atau alur dari perjalanan game untuk menuntun pemain dari awal permainan hingga akhir. Storyboard juga dapat berupa susunan sistem level. Keempat, membuat grafis atau animasi untuk keperluan visual game seperti desain karakter di dalam game untuk karakter player maupun NPC *(Non-Player Character)*, desain backgournd map (setting lingkungan game), animasi pergerakan player dan masih banyak lagi. Grafis dan animasi pada game merupakan elemen yang sangat penting bagi game. Kelima membuat kode Program dan mengimplementasi algoritma untuk perilaku buatan pada NPC. Keenam, menambahkan efek multimedia seperti suara latar dan pemberian efek suara pada game untuk menambah kesan dalam permainan sehingga ketika player bermain akan merasa game menjadi lebih hidup. Terakhir adalah tahap ujicoba, dilakukan testing atau pengecekkan pada game menemukan bug, crash atau error yang kemungkinan akan terjadi pada game berdasarkan parameter pengujian yang terdapat pada Tabel 1,2,3 dan Tabel 4.

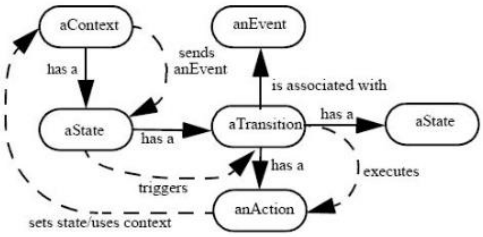
### 2.1. Non Player Character (NPC)

*Non-Player Character* atau disingkat NPC merupakan rancangan karakter otonom yang biasa digunakan dalam permainan komputer. NPC dalam sebuah *game* berupa karakter selain dari *human player* yang dapat mengimprovisasikan tindakannya sendiri berdasarkan algoritmanya [4].

Perilaku buatan pada permainan atau game berupa algoritma komputer yang ditanamkan pada karakter MPC sehingga dapat memerintahkannya berperilaku secara autonomous. Pada proses pengembangan *game* terdapat struktur model yang dapat digunakan dalam menerapkan perilaku buatan pada *game* [5] :




Gambar 2. Model Perilaku buatan untuk NPC [5]




Gambar 3. Alur kerja FSM [3]

Pada Gambar 2 terdapat diagram blok yang menggambarkan bagaimana perilaku buatan dapat mengimplementasi algoritma diantaranya berupa algoritma kecerdasan buatan sebagai salah satu metode yang dapat digunakan. Terbagi menjadi tiga bagian yaitu *strategi*, *decision making* dan *action*. *Strategi* berhubungan tentang pengendalian sifat keseluruhan karakter. *Decision making* atau pengambilan keputusan berhubungan tentang pemilihan tindakan yang akan dilakukan karakter. *Action* atau pergerakan berhubungan dengan perubahan algoritma menjadi semacam gerakan bagi karakter dalam *game*.

### 2.2. Finite State Machine (FSM)

*Finite State Machine* (FSM) merupakan metode yang digunakan dalam merancang perilaku (*behavior*) berbasis *event* [2]. *Finite State Machine* menentukan urutan dari berbagi *behaviour* dan disajikan dalam bentuk *state* dan *transition* [15]. Tampak pada gambar 3 diagram proses FSM dimana terdiri atas tiga hal yaitu *state* (keadaan), *event* (kejadian), dan *action* (aksi). Alur proses dari FSM yaitu ketika sistem sedang berada di *state* yang sedang aktif, sistem akan berpindah ke *state* lain jika mendapatkan input atau *event* tertentu dan saat transisi perpindahan *state* ini akan disertai dengan *action* atau aksi yang akan dilakukan oleh sistem, baik berupa aksi yang sederhana maupun aksi yang kompleks [3]. Sistem akan tetap melakukan aksi yang sama pada suatu state sampai sistem menerima event tertentu baik yang berasal dari perangkat luar atau komponen dari sistem itu sendiri [14].

### 2.3. Pengujian dengan Metode Black-box Testing

B*lack-box Testing* merupakan metode pengujian berdasarkan persyaratan dan spesifikasi [13]. Dengan metode ini dapat diketahui jika fungsionalitas masih dapat menerima masukan data yang tidak diharapkan maka menyebabkan data yang disimpan kurang valid [6].

### 2.4. Pemrograman dengan Godot Engine atau GDscript

*Game engine* adalah sebuah IDE-*framework software* yang di desain untuk pembuatan *game*, yang dilengkapi *library* yang dibutuhkan dan juga tambahan yang disebut *add-on* agar mendukung lingkungan perangkat lunak lain [7]. IDE ini menyediakan tools dalam mendukung pengembangan game yang bersifat *opensource* [8][9]. Bahasa pemrograman yang digunakan disebut *GDscript* [10].

### 2.5. Perancangan

*Tema Game* yang akan dibuat ini yakni *"Game Pemadam Kebakaran"* termasuk jenis RPG (*Role Playing Game*). Pada game ini terdapat 1 karakter utama yang berperan sebagai *'pemadam kebakaran'* (lihat gambar 14) yang akan dimainkan oleh human *player*. Kemudian akan ada beberapa karakter musuh dengan karakter *'monster api'* (lihat gambar 15) di mana karakter ini merupakan NPC dalam *game* dan merupakan objek dari penelitian ini untuk ditanamkan perilaku buatan menggunakan algoritma.

#### 2.5.1. *Pengembangan Storyline*

*Game* ini berlatar belakang di suatu wilayah yang terkena dampak dari suatu aktifitas bangunan laboratorium (lihat gambar 20) yang letaknya ditengah hutan. Simulasi yang direncanakan adalah dengan terjadinya bencana kebakaran pada gedung laboratorium dan menyambar pohon dan bangunan lain (lihat gambar 17) disekitarnya. Disini sambaran sebaran api diilustrasikan sebagai karakter antagonist disebut 'monster-api' dan akan menguasai wilayah tersebut dengan lahapan api. Karena ditengah hutan tentu penyebaran titik api menjadi lebih cepat. Disisi lain untuk mengatasi dan mencegah kebakaran hutan yang

masif, di siapkan karakter protagonist berupa petugas pemadam kebakaran yang akan melindungi wilayah tersebut. Misi selain memadamkan sebaran dari monster-api yakni menjagai sumber air yang disimbolkan dengn 'Tower utama'(lihat gambar 18) dan kantor stasion pemadam (lihat gambar 19) dari serangan api, membasmi para monster-api yang menyerang dan memadamkan bangunan laboratorium yang merupakan sumber api penghasil monster-api [11].

2.5.2. *Pengembangan Storyboard*

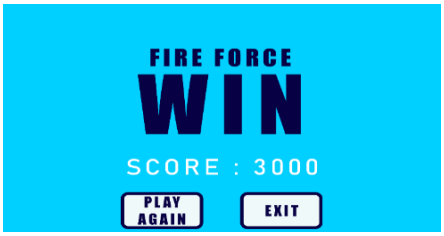
Berdasarkan *storyline*, berikut adalah rancangan *storyboard* dalam bentuk scene dalam *IDE-Godot* untuk menjelaskan tentang rancangan antarmuka halaman dan objek karakter pada *game* antara lain :


Gambar 4. Rancangan *scene* manu utama


Gambar 5. Rancangan *scene* narasi

Saat *game* dijalankan pada komputer desktop maka akan muncul tampilan seperti Gambar 4 yaitu *scene* 'menu utama' dimana human *player* dapat memilih untuk mulai memainkan *game* atau keluar dari *game*. Ketika human *player* memilih untuk memulai permainan maka *game* akan diarahkan menuju *scene 'role atau* narasi'. Pada *scene* ini akan ditampilkan latar cerita dari *game* dalam bentuk dialog karakter-karakternya seperti pada Gambar 5. Setelah latar cerita dari *game* telah selesai ditampilkan, selanjutnya human *player* akan diarahkan ke *scene gameplay* yang dapat dilihat pada Gambar 6. Pada *scene* inilah *player* akan mulai memainkan *game* dengan tujuan untuk menyelesaikan misi yang diberikan. Jika human *player* berhasil menyelesaikan misi dalam permainan, maka *game* akan berakhir dengan status *player* telah memenangkan permainan dan selanjutnya akan diarahkan menuju *scene* hasil pada Gambar 7 untuk menunjukkan total skor yang didapatkan dan human *player* dapat memilih untuk memulai permainan kembali atau kembali ke menu utama.


Gambar 6. Rancangan *scene* gameplay




Gambar 7. Rancangan *scene* hasil ketika menang


Gambar 8. Rancangan *scene* hasil ketika kalah

Sebaliknya, jika human *player* gagal menyelesaikan misi dalam permainan atau karakter mati ditengah permainan, maka *game* akan berakhir dengan status *player* telah kalah dalam permainan dan selanjutnya akan diarahkan menuju *scene* 'result' atau hasil akhir pada Gambar 8, di mana *player* dapat memilih untuk memulai permainan kembali atau kembali ke menu utama.

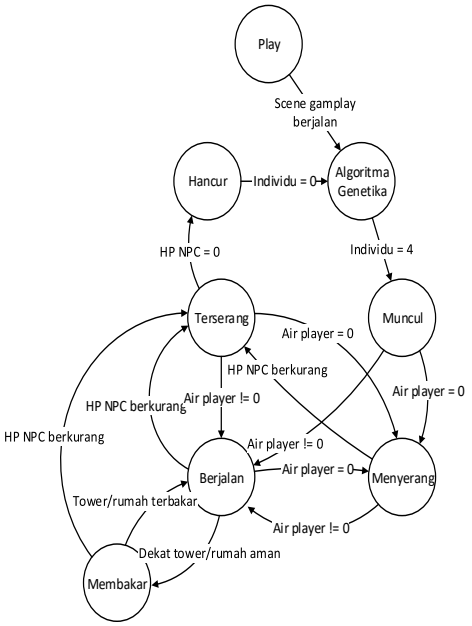
# 3. HASIL DAN PEMBAHASAN

## 3.1. Hasil Rancangan FSM pada karakter NPC player

Terdapat tiga diagram FSM yang akan menggambarkan behavior atau perilaku dari karakter-karakter yang ada di dalam *game* yaitu karakter utama (pemain), NPC musuh, dan bangunan. Diagram FSM dari masing-masing karakter dapat dilihat pada Gambar 9, 10 dan 11 berikut.

Pada Gambar 9 dapat dilihat bahwa perilaku yang dimiliki oleh NPC terbagi atas beberapa *state* dan akan berpindah berdasarkan perubahan *event* yang diberikan. Beberapa contoh *state* yang mengendalikan perilaku yang menjadi perilaku NPC selama berada diwilayah permainan yaitu *state 'berjalan'*, *'menyerang'* dan *'membakar'*. Pada Gambar 10 merupakan potongan *GDscript* fungsi utama dari NPC. Pada fungsi ini akan dideklariskan beberapa variabel yang akan digunakan ada fungsi utama antara lain : Variabel '*raycasts'* yang berupa *dictionary* sebagai objek yang akan mendeteksi objek lainnya pada garis lurus di empat arah. Berikutnya '*velocity'* sebagai kecepatan dan arah pergerakan dari NPC yang disimpan dalam bentuk vektor. Berikutnya lagi '*state'* sebagai parameter perilaku yang dilakukan oleh NPC dalam bentuk String. Selanjutnya *'cooldown_tembak'* sebagai parameter *delay* pada saat NPC menembak dalam bentuk *boolean*. Terakhir variable *'pos_target'* sebagai penentu arah objek yang ditargetkan oleh NPC dalam bentuk String.

Pada fungsi utama akan dilakukan penentuan *state* dari NPC menggunakan *method match* yaitu fungsi *statement control* pada bahasa GDscript untuk memilih *state*ment yang sesuai dengan nilai variabel kondisi. Dalam program di atas, variabel *state* akan menjadi variabel kondisi yang digunakan. Jika *state* bernilai "jalan" maka hal ini membuat NPC berada pada *state* berjalan. begitupun jika *state* bernilai "tembak" maka hal ini membuat NPC berada pada *state* menyerang atau membakar. Untuk program pada *state* "jalan" akan menggunakan fungsi jalan() sebagai fungsi utama untuk melakukan pergerakan. Tahapan pada *state* ini dimulai dari pengecekan pada status pergerakan NPC yang dapat dilihat pada Gambar 10.




Gambar 9. FSM NPC Karakter

```
onready var raycasts = {'kanan': $raycast_kanan,
                        'kiri': $raycast_kiri,
                        'atas': $raycast_atas,
                        'bawah': $raycast_bawah}

var velocity = Vektor2.ZERO
var state : String
var cooldown_tembak : bool = false
var pos_target

func _physics_process(_delta):
match state:
  "jalan" :
      if not move_and_slide(velocity):
          ubah_arah()
      jalan()

  "tembak" :
  var target = raycasts[pos_target].get_collider()
      if target == null or !target.is_in_group('tile'):
          if !cooldown_tembak:
              shoot()
      else :
          state = "jalan"
```

Gambar 10. GDscript fungsi utama karakter player NPC

Jika NPC secara fisik tidak sedang bergerak maka NPC akan melakukan perubahan arah dengan menggunakan fungsi ubah_arah() dan kemudian akan melakukan pergerakan menggunakan fungsi jalan(). Untuk fungsi jalan() dapat dilihat pada gambar berikut. Pada Gambar 11 tampak GDscript fungsi untuk membuat NPC bergerak. Terdapat beberapa variabel tambahan yang digunakan pada fungsi ini yaitu : Pertama, variable 'kejar' sebagai status mengejar dari NPC dalam bentuk Boolean. Kedua, 'arah' sebagai

arah NPC bergerak lurus dalam bentuk string atau kalimat. Ketiga, ‘speed’ merupakan kecepatan NPC bergerak. Keempat, variable ‘macam_arah’ yang berupa dictionary untuk mendefinisikan setiap arah dari pergerakan NPC dimana ia akan memiliki nilai vektor. Tahapan pertama dari fungsi ini akan dicek status pergerakan mengejar dari NPC menggunakan statement if. Jika status mengejar adalah true (benar) maka akan dijalankan program pengejaran. Sedangkan jika tidak dalam status mengejar maka nilai kecepatan dan arahnya akan diatur dengan formula nilai vektor dari arahnya akan dikalikan dengan nilai ‘speed’ dari NPC. Selanjutnya NPC akan digerakan menggunakan method move_and_slide. Pada GDscript method ini berfungsi untuk menggerakan objek berdasarkan nilai vektor yang diberikan dimana pada program menggunakan nilai vektor pada variabel ‘velocity’.

```
var kejar
var arah
var speed
var macam_arah = {'kanan': Vektor2(1, 0),
                  'kiri': Vektor2(-1, 0),
                  'atas': Vektor2(0, -1),
                  'bawah': Vektor2(0, 1)}
func jalan():
if kejar:
  if path.size() > 0:
  _kejar()
else:
  velocity = macam_arah[arah].normalized()
* speed
  velocity = move_and_slide(velocity)
```

Gambar 11. GDscript fungsi jalan()

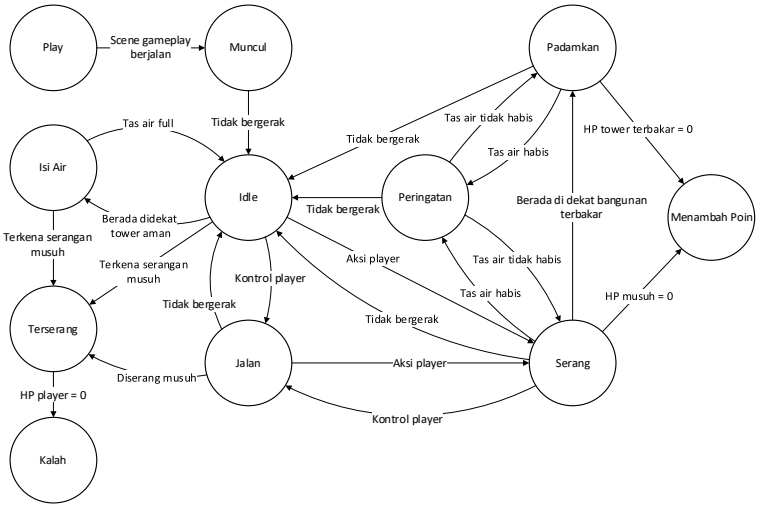
```
var bidikan

func shoot():
var peluruBaru = PeluruApi.instance()
peluruBaru.power = ATK
peluruBaru.npcID = ID

if pos_target == 'kanan':
  bidikan.position = Vektor2(1.323, -5.598)
if pos_target == 'kiri':
  bidikan.position = Vektor2(-1.323, -5.598)
if pos_target == 'atas' or pos_target == 'bawah':
  bidikan.position = Vektor2(0, -4.233)

peluruBaru.global_position = position + bidikan.position
peluruBaru._tembak(macam_arah[pos_target])
get_tree().current_scene.get_node("Map").add_child(pelur
uBaru)

velocity = Vektor2.ZERO
cooldown_tembak = true
yield(get_tree().create_timer(float(1)/ATK_spd),
"timeout")

cooldown_tembak = false
if target["tipe"] == "kosong":
        state = "jalan"
```

Gambar 12. GDscript fungsi shoot()

Untuk GDscript pada *state* ‘tembak’ akan menggunakan fungsi *shoot*() sebagai fungsi utama untuk menembakan peluru api kepada targetnya. Pada Gambar 10, jika target NPC adalah karakter petugas pemadam yang dimainkan human player maka NPC akan berada pada *state* menembak. Sedangkan jika target NPC adalah objek bangunan maka NPC akan berada pada *state* ‘membakar’. Tahapan pada program ini dimulai dari penentuan dan pengecekkan target dengan menggunakan metode *raycasting*. Raycasting adalah sebuah teknik untuk menampilkan perspektif tiga dimensi pada layar dua dimensi, dengan menggunakan perhitungan geometri terutama perhitungan titik potong antara sinar garis dan bidang [12]. Pada karakter NPC dipasang 4 buah garis *raycast* dan disimpan pada variabel *raycasts* untuk mendeteksi target pada 4 arah yaitu depan, belakang, kiri dan kanan. Selanjutnya akan dilakukan pengecekkan untuk objek yang dideteksi oleh *raycast*. Jika yang dideteksi adalah dinding map maka *state* akan kembali menjadi ‘jalan’. Sedangkan jika yang dideteksi bukanlah dinding map maka maka fungsi *shoot*() akan dijalankan ketika *cooldown*_tembak telah berhenti. Untuk fungsi *shoot*() dapat diliat pada program berikut.

Pada program di Gambar 12 merupakan fungsi untuk membuat dan menembakkan peluru api dari NPC. Terdapat beberapa variabel tambahan yang digunakan pada fungsi ini yaitu : Pertama, *‘peluruBaru’* untuk menyimpan objek dan GDscript dari peluru api yang akan dibuat dan kedua, *‘bidikan’* untuk titik awal peluru

api akan muncul dalam vector. Untuk tahapan dari fungsi ini dimulai dari membuat peluru api dalam bentuk objek pada variabel *peluruBaru* dengan menurunkan beberapa nilai dari NPC yaitu instansiasi objek ditulis sebagai 'ATK' untuk jumlah kekuatan peluru dan ID untuk menentukan identitas penembak pada peluru. Selanjutnya mengatur posisi *bidikan* dari NPC berdasarkan dari variabel *pos_target*. Ketika posisi bidikan sudah pas, maka posisi peluru akan diset sesuai dengan posisi bidikan dan akan ditembakkan dengan menggunakan *fungsi_tembak()* pada objek *peluruBaru* dengan parameter posisi target dalam vektor. Ketika semuanya sudah di set, maka objek ini akan dimunculkan dengan menambahkannnya sebagai node baru pada *scene* utama game. Ketika peluru telah berhasil ditembakkan, NPC tidak akan bisa bergerak dan *cooldown_tembak* akan aktif selama beberapa saat. Ketika *cooldown_tembak* telah berhenti, akan dilakukan pengecekkan target dari NPC. Jika NPC sudah tidak menargetkan objek apapun, maka *state* akan kembali menjadi *'jalan'*. Untuk diagram FSM dari karakter lain pada game ini dapat dilihat pada Gambar 13 dan Gambar 14.



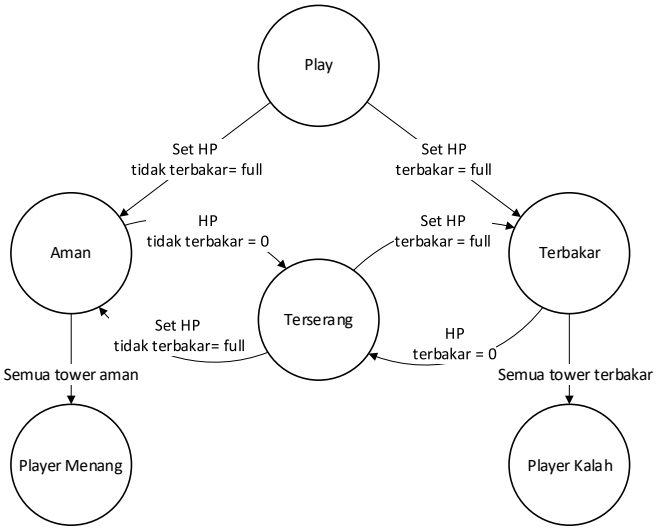

Gambar 13. FSM karakter human-player

Pada Gambar 13 menggambarkan perilaku dari karakter *human player* dalam permainan dalam bentuk *state* yang akan berubah berdasarkan input atau event yang diberikan. Adapun perilaku yang dimiliki oleh karakter *human player* berdasarkan dari diagram FSM di atas antara lain : a). Dapat berjalan sesuai arah navigasi ; b). Dapat menyerang (tembak bola air); c). Dapat memadamkan bangunan terbakar; d). Tidak dapar menyerang saat air habis; e).Dapar mengisi ulang air; f).HP akan berkurang saat terkena serangan; h).Menambah poin saat mengalahkan NPC atau padamkan bangunan; i).Akan kalah saat HP habis. Pada Gambar 14 dapat dilihat perilaku yang dimiliki oleh bangunan pada wilayah permainan antara lain : a).Bangunan akan aman ketika *player* memadamkan bangunan saat terbakar; b).Bangunan akan terbakar ketika NPC membakar bangunan saat aman; c).HP akan berkurang saat terkena serangan; d).*Human Player* akan menang jika semua bangunan aman dan semua musuh hancur; e).*Human Player* akan kalah jika semua bangunan terbakar.

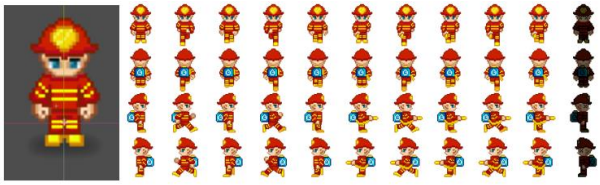
### 3.2. Hasil Rancangan Karakter pada Sitemap game dan Halaman Antarmuka

Ketika *game* telah berhasil diselesaikan, *game* perlu dilakukan pengujian terlebih dahulu untuk mengetahui bagaimana kinerja *game* ketika dimainkan. Dalam hal ini peneliti akan melakukan pengujian fungsional untuk mengetahui kinerja *game*. Metode pengujian yang digunakan adalah *Blackbox Testing* atau pengujian *Blackbox*. Pada game ini dihasilkan beberapa desain karakter yang digunakan yaitu karakter player, NPC dan bangunan.

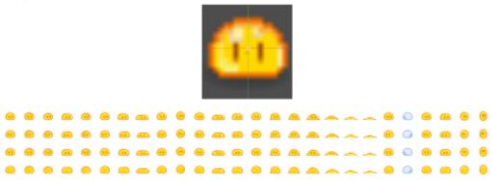
Pada Gambar 15 merupakan desain karakter human player berupa karakter *pemadam kebakaran* dengan pakaian anti api yang lengkap dengan tas penyimpanan airnya. Pada Gambar 16 merupakan desain karakter NPC dimana karakter ini memiliki peran sebagai makhluk berupa *monster api* yang dapat menembakkan api. Untuk desain bangunan pada game akan bertindak sebagai objek lingkungan yang mampu terbakar saat diserang player NPC dan dapat dipadamkan oleh *human player*.




Gambar 14. FSM bangunan



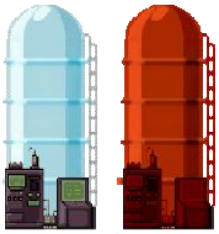
Gambar 15. Desain karakter human player



Gambar 16. Desain komputer karakter sebagai NPC

Pada game, bangunan memiliki beberapa jenis yaitu sebagai berikut. Gambar 17 merupakan desain bangunan yang berupa rumah biasa sebagai rumah penduduk yang terserbar di wilayah permainan. Gambar 18 merupakan desain bangunan yang berupa tower sebagai tempat player mendapatkan pasokan airnya kembali. Namun tower tidak akan dapat digunakan ketika bangunan sedang terbakar.

Gambar 17. Desain bangunan jenis rumah



Gambar 18. Desain bangunan jenis tower

Gambar 19 merupakan desain bangunan yang berupa *fire station* sebagai markas utama karakter human player dan tempat utama mendapatkan pasokan air. Ketika bangunan ini terbakar, hal itu menandakan bahwa sudah tidak adanya tempat aman diwilayah tersebut.


Gambar 19. Desain bangunan jenis fire station

Gambar 20. Desain bangunan jenis laboratorium

Gambar 20 merupakan desain bangunan yang berupa *bangunan Laboratorium* sebagai sumber utama munculnya titik Api sebagai player NPC dan tempat sebaran titik api muncul secara terus menerus atau sebagai markas *monster-api*. Ketika bangunan ini sudah dipadamkan, hal itu akan membuat NPC akan berhenti diproduksi dan menandakan bahwa sudah tidak adanya tempat yang terbakar diwilayah tersebut. Untuk desain *environment* atau lingkungan permainan yang digunakan di dalam *game* dapat dilihat pada gambar 21 berikut. *Environment* atau lingkungan wilayah permainan pada game yang berupa sebuah hutan yang membentuk *maze* atau labirin, di mana akan tersebar beberapa bangunan di beberapa titik pada lingkungan game. Untuk status dari bangunan akan di set-up sebanyak 2 Tower di sebelah kiri dari map akan diset dalam keadaan aman, sedangkan untuk 2 Tower lainnya di arah berlawanan akan di set dalam keadaan terbakar.

Gambar 21. Desain sitemap / lingkungan permainan


Gambar 22. Halaman scene menu utama

Untuk *Fire Station* akan diset dalam keadaan aman, sedangkan untuk *bangunan lab* akan diset dalam keadaan terbakar. Untuk semua *rumah* akan diset dalam keadaan aman. Untuk implementasi desain antarmuka pada *game* dapat dilihat pada Gambar 18. Pada Gambar 23 merupakan tampilan halaman *scene* menu utama dengan dua tombol ; '*Play'* untuk memulai permaian, dan '*Quit'* untuk keluar dari *game*. Pada gambar 24 *Scene* role narasi akan menampilkan cerita pengantar dari *game* sebelum mulai bermain sesuai dengan rancangan awal. Pada *scene* ini akan ditampilkan gambaran karakter sesuai dengan ceritanya ditambah dengan adanya hal berikut:

a. Dialog box yang akan menampilkan percakapan dari karakter,
b. Tombol *Next* untuk melanjutkan percakapan karakter,
c. Tombol *Skip* untuk melewatkan seluruh narasi dan langsung menuju *scene gameplay*.


Gambar 13. Desain scene scenario narasi alur permainan


Gambar 24. Tampilan scene gameplay

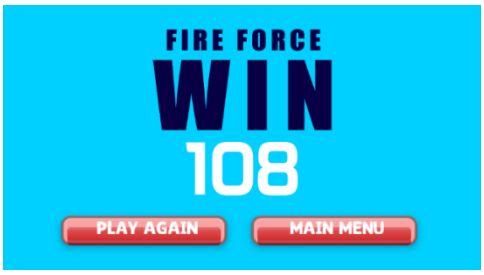
Berdasarkan Gambar 24 *Scene Gameplay* akan menampilkan *environment* atau wilayah permainan di dalam *game* saat permainan dimulai, di mana terdapat karakter pemain yang akan menjadi pusat dari layar permainan sehingga layar akan mengikuti gerakan dari karakter utama. Kemudian terdepat beberapa GUI yang digunakan pada permainan yaitu :

a) Pada bagian kiri atas layar terdapat HP dan *Water* bar untuk indikator nilai ketahanan pemain dan banyaknya air yang ada ditas pemain. Kemudian dibawahnya terdapat panel skor untuk memperlihatkan skor yang diperoleh oleh pemain selama bermain dan terdapat beberapa Icon yang menunjukkan situasi pada lingkungan permainan.
b) Pada bagian kanan atas layar terdapat tombol pause yang digunakan untuk menghentikan permainan untuk sementara.
c) Pada bagian kiri bawah layar terdapat 4 tombol navigasi untuk menggerakan karakter utama sesuai arahnya.

d) Pada bagian kanan bawah layar terdapat 2 tombol aksi. Tombol pertama yang paling kanan adalah tombol serangan (*Splash*) yang digunakan untuk *player* menembakkan peluru airnya. Kemudian tombol di sebelah kirinya digunakan untuk mengisi ulang tas air *player* (*Refill*) jika tas airnya kosong.

Berdasarkan Gambar 25 dan Gambar 26 *Scene* hasil ketika *player meme*nangkan atau kalah dalam permainan. Pada layar terdiri tulisan '*Fire Force Win'* atau '*Fire Force Lose* '. Dibawahnya ditampilkan score akhir. Kemudian ada dua tombol berwarna merah yang sejajar yaitu

a. *Play Again* untuk memulai permaian kembali,
b. *Main Menu* untuk kembali ke *scene* menu utama.




Gambar 26. Halaman scene hasil ketika human player berhasil mengalahkan NPC


Gambar 26. Halaman scene hasil ketika NPC berhasil mengalahkan human player

### 3.3. Hasil pengujian fungsional

Pada pengujian ini peniliti akan melihat apakah *game* yang telah dibuat dapat dimainkan dan dapat berjalan dengan lancar sesuai dengan rancangan *game* yang telah dibuat. Hasil dari pengujian fungsional menggunakan black box testing yang dapat dilihat pada Tabel 1 - 2 berikut.

Tabel 1. Parameter dan hasil pengujian pada *scene halaman* menu utama

| Aktivitas pengujian | Hasil pengujian |
|---|---|
| *Scene* Menu Utama | |
| *Player membuka game* | Muncul tampilkan *scene* Menu Utama di mana terdapat logo permainan *FireForce Tower*, tombol *Start* dan *Quit* |
| *Player menekan* tombol *start* | *Player akan* diarahkan ke *scene* narasi yang menjadi prolog / pengantar *game* di mana akan ada gambaran cerita, tombol *Next* dan *Skip* |
| *Player menekan* tombol *quit* | *Player akan* keluar dari *game* |

Berdasarkan Tabel 1, pengujian pada *scene* menu utama berhasil dijalankan tanpa adanya error baik ketika *game* dibuka maupun pada saat menekan button pada *scene.*

Tabel 2. Parameter dan hasil pengujian pada scene halaman menu utama

| Aktivitas pengujian | Hasil pengujian |
|---|---|
| *Scene* Narasi | |
| *Player menekan* tombol *Next* | Percakapan pada *scene* Narasi akan dilanjutkan ke text selanjutnya |
| *Player menekan* tombol *Skip* | Muncul *scene Gameplay* di mana karakter *player* telah muncul di wilayah permainan (*environment game*) |

Berdasarkan Tabel 2, pengujian pada *scene* narasi berhasil dijalankan dimana button pada *scene* ini bekerja dengan semestinya.

Tabel 3. Hasil pengujian pada scene halaman gameplay

| Aktivitas pengujian | Hasil pengujian |
|---|---|
| *Scene Gameplay* | |
| *Player menek*an tombol navigasi (atas, kiri, kanan, bawah) | Karakter *player* pada *game* bergerak sesuai arah tombol navigasi yang diklik |
| Menekan tombol serang saat *WATER* bar *player full* / masih tersisa | Muncul aksi karakter *player* yang menembakan peluru air sehingga *Water* bar akan berkurang |
| Menekan tombol serang saat *WATER* bar *player habi*s | Tidak terjadi apa - apa |
| Menekan tombol refill saat *WATER* bar *player habi*s dan sedang berada didekat *Tower* yang aman | Muncul aksi *player* mengisi *Water* bar *player* hingga full kembali atau terhenti karena *Tower* berubah menjadi terbakar |
| Menekan tombol refill saat *WATER* bar *player full* / masih tersisa dan berada di sekitar *Tower* aman atau terbakar | Tidak terjadi apa - apa |
| Menekan tombol refill saat tidak berada didekat *Tower* | Tidak terjadi apa - apa |
| Serangan *player* mengenai NPC | *Healt Point* (HP) bar NPC berkurang |
| Saat HP NPC mencapai ‘0’ | NPC menghilang dari wilayah permainan, nilai pin pada *Icon* NPC berkurang sesuai jumlah NPC yang bertahan dan nilai pada panel skor bertambah |
| Saat berhasil memadamkan *Tower* terbakar | Status *Tower* berubah menjadi aman, nilai pin pada *Icon Tower* akan berkurang sesuai jumlah *Tower* yang masih terbakar dan nilai pada panel skor akan bertambah |
| Serangan NPC mengenai *player* | *Healt Point* (HP) bar *player* berkurang |
| Serangan NPC mengenai bangunan (*Tower* atau Rumah) yang dalam status aman | *Healt Point* (HP) bar bangunan berkurang |
| Serangan NPC mengenai bangunan (*Tower* atau Rumah) yang dalam status terbakar | Tidak terjadi apa - apa |
| Saat HP bangunan (*Tower* atau Rumah) yang dalam status aman mencapai 0 | Status bangunan berubah menjadi terbakar dan *Healt Point* (HP) bar bangunan akan tereset menjadi full kembali |
| Saat HP bangunan (*Tower* atau Rumah) yang dalam status terbakar mencapai 0 | Status bangunan berubah menjadi aman dan *Healt Point* (HP) bar bangunan akan tereset menjadi full kembali |
| *Player* menekan tombol *Pause* | *Scene Gamplay* menjadi *freeze* (berhenti) |
| Pada mode *Pause player* menekan tombol *Resume* | *Game* kembali berjalan |
| Pada mode *pause player menek*an tombol *Main Menu* | *Player* akan diarahkan ke *scene* menu utama |
| Saat *player* mengamankan semua bangunan dan mengalahkan semua NPC | *Icon Tower* dan NPC telah menghilang dan muncul tulisan *WIN*, total score yang didapatkan selama bermain dan muncul dua tombol (*Play Again* dan *Main Menu*) |
| Saat HP *player menc*apai 0 atau semua *Tower* dan Rumah terbakar | Muncul tulisan *LOSE*, total score yang didapatkan selama bermain dan muncul dua tombol (*Play Again* dan *Main Menu*) |

Berdasarkan Tabel 3, pengujian pada *scene gameplay* berhasil dijalankan dimana tombol-tombol yang tersedia pada *scene* ini bekerja dengan semestinya dan setiap karakter di dalam *game* dapat berperilaku sesuai dengan rancangan FSM karakternya.

Tabel 4. Hasil pengujian pada *scene* halaman hasil

| Aktivitas pengujian | Hasil pengujian |
|---|---|
| *Scene* Hasil | |
| *Player* menekan tombol *Play Again* | Muncul *scene Gameplay* kembali dan permainan akan dimulai kembali dari awal |
| *Player* menekan tombol *Main Menu* | Muncul *scene* Menu Utama. |

Berdasarkan Tabel 4, pengujian pada *scene* hasil, baik saat *player* memenangkan permainan maupun saat *player* kalah dalam permainan berhasil dijalankan dimana tombol-tombol pada *scene* ini bekerja dengan semestinya.

## 4. KESIMPULAN

Game permainan dengan implementasi rancangan perilaku buatan pada NPC player dengan *GDscript* mengacu pada rancangan motede FSM telah dikembangan. Karakter di dalam *game* akan bergerak dan berperilaku sesuai dengan perancangan FSM yang telah dibuat. Hasil pengujian *black-box testing* ini didapatkan parameter pengujian antara lain parameter pengujian pada *scene halaman* menu utama, scene halaman gameplay dan *scene* halaman hasil didapati *game* dapat berjalan dengan baik tanpa ditemukan lagi kesalahan atau *bug* pada setiap *scene* dan karakter. *Game* dapat berjalan dengan semestinya tanpa adanya kesalahan atau error pada saat *game* dimainkan.

## REFERENCES

[1] P. Warman, T. Wijman, O. Meehan, and B. De Heij, “Global *game*s market report,” 2022.
[2] M. F. Rahadian, A. Suyatno, and S. Maharani, “Penerapan Metode *Finite State Machine* Pada *Game* ‘The Relationship,’” *Inform. Mulawarman J. Ilm. Ilmu Komput.*, vol. 11, no. 1, p. 14, 2016, https://doi:10.30872/jim.v11i1.198 .
[3] R. Bimantika, “Pengembangan *Game* the Galaxy Menggunakan Metode Fsm ( *Finite State Machine* ),” J. Mhs. Tek. Inform., vol. 1, no. 1, pp. 180–187, 2017.
[4] Y. M. Arif, A. Wicaksono, & F. Kurniawan, Pergantian Senjata NPC pada *Game* FPS Menggunakan Fuzzy Sugeno. Prosiding Seminas Competitive Advantage, Vol 1, No 2, 2012.
[5] I. Millington and J. Funge, *Artificial Intelligence* for *Game*s. 2018. https://doi:10.1201/9781315375229
[6] W. N. Cholifah, Y. Yulianingsih, and S. M. Sagita. “Pengujian Black Box Testing pada Aplikasi Action & Strategy Berbasis Android dengan Teknologi Phonegap,” STRING (Satuan Tulisan Ris. dan Inov. Teknol., vol. 3, no. 2, p. 206, 2018. https://doi:10.30998/string.v3i2.3048 ,CAN - J. Teknol. Inf. dan Komun., vol. 11, no. 2, JITTER- Jurnal Ilmiah Teknologi dan Komputer Vol. 2, No. 3.
[7] Griya Media Nusantara. (2021). Mengenal Game Engine [Online]. Available : https://www.griyawebsite.com/mengenal-game-engine/
[8] Ferdi and S. A. Arnomo, “Perancangan Game Platformer Pemburu Koin Menggunakan Godot Engine,” Comasie, vol. 3, no. 3, pp. 21–30, 2020, [Online]. Available: https://forum.upbatam.ac.id/index.php/comasiejournal/article/view/5341/2495
[9] C. A. R. Tobi, R. Andrea, and A. Yusika, “Membangun Side Scrolling Game Flying Enggang Berbasis Android Dengan Godot Engine,” J. Inform. Wicida, vol. 8, no. 1, pp. 1–8, 2019, https://doi:10.46984/inf-wcd.1234 .
[10] J. Linietsky, A. Manzur dan Godot community. *GDScript: An introduction to dynamic languages* [Online]. Available : https://docs.godotengine.org/en/stable/tutorials/scripting/gdscript/gdscript_advanced.html
[11] M. Fikriansyah. “Non-Player Character Pada Game Pemadam Kebakaran Dengan Algoritma Genetika”. unpublished.
[12] Wikipedia. (2019, Oct. 1). Raycasting [Online]. Available : https://id.wikipedia.org/wiki/Raycasting
[13] R. Dev, A. Jääskeläinen, M.Katara,Model-Based GUI Testing: Case Smartphone Camera and Messaging Development, Advances in Computers, Elsevier, Volume 85, 2012, Pages 65-122,ISSN 0065-2458, ISBN 9780123965264, https://doi.org/10.1016/B978-0-12-396526-4.00002-3 .

[14] W. Safitra, A. Faisol dan S. A. Wibowo, " Finite State Machine Method to Non Player Character (NPC) Action Strategy Game 'Ouroboros' ,Vol. 4 No. 2 (2020): JATI Vol. 4 No. 2, https://doi.org/10.36040/jati.v4i2.2828

[15] A. F. Naharu, E. M. A. Jonomaro dan M. A. Akbar, "Penerapan *Hierarchical Finite State Machine* untuk Pengambilan Keputusan *Non-Player Character* (Studi Kasus: Gim *Hack and Slash*)", Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN: 2548-964X Vol. 5, No. 3, Maret 2021, hlm. 1136-1141 http://j-ptiik.ub.ac.id

## BIOGRAPHY OF AUTHORS

**Muhammad Fikriansyah,** Completed Bachelor degree program in Electrical Engineering Department at University of Mataram, 2022. Have a penchant for game programming. mailto : mfikriansyah1011@gmail.com

**Giri Wahyu Wiriasto ,** Completed Bachelor degree program in Electrical Engineering Department at Institut Teknologi Nasional of Malang, 2006. Completing a Master's Degree Program in Electrical Engineering and Multimedia Smart Computer Network System at Institut Teknologi Sepuluh Nopember of Surabaya, 2010. As a lecturer at the Electrical and Computer Engineering Department - University of Mataram since 2010. Interest with computer and reading. Research interest : Pattern Recognition, Deep Neural Network, Serious Game and Digital Image Processing. mailto : giriwahyuwiriasto@unram.ac.id

**A. Sjamsjiar Rachman,** Completed Bachelor degree program in Electrical Engineering Department at Brawijaya University, Malang, 1998. Completing a Master's Degree Program in Electrical Engineering at Gajah Mada University, Yogyakarta, 2003. Since the year 1999 actively teach as a lecturer at the Electrical and Computer Engineering Department - University of Mataram. Have Competence and teaching course include: Criptography, Wireless sensors network, dan Sistem mailto : sjamsjiar@unram.ac.id