IMPLEMENTASI ALGORITMA *PARTICLE SWARM OPTIMIZATION* DAN *FINITE STATE MACHINE* UNTUK PERILAKU NPC SECARA BERKELOMPOK DAN PENGAMBILAN KEPUTUSAN PADA GAME MOBA

Mochammad Akbar Fadhlillah Mi'raj, Galih Hermawan

Universitas Komputer Indonesia
Jl Dipatiukur No. 112-114-116
E-mail : makbarfm@gmail.com, galih.hermawan@email.unikom.ac.id

ABSTRAK

*Game Multiplayer Online Battle Arena (*MOBA) merupakan salah satu jenis *genre* permainan yang merupakan pengembangan dari jeins *game Real Time Strategi* (RTS). Tujuan dari *game* ini adalah untuk membuat *base* lawan hancur terlebih dahulu sebelum *base* kita dihancurkan lawan. *Game* MOBA merupakan tipe permainan *multiplayer* yang dimainkan oleh 5 pemain di masing-masing timnya. Setiap pemain akan menggunakan karakter untuk memainkan *game* ini. Dalam *game* MOBA akan muncul NPC yang akan menghalangi setiap tim yang akan bermain. NPC ini akan bergerak secara berkelompok dan akan menyerang pemain. Setiap tim mempunyai NPC-nya masing-masing. Permainan dari MOBA akan berakhir ketika *base* salah satu tim sudah dihancurkan. NPC akan membantu pemain dalam melawan tim lawan, masing-masing tim akan mempunyai NPC nya sendiri ketika berada di dalam permainan.

NPC yang berada di *game* ini harus dapat berkorelasi ketika bergerak secara berkelompok. Oleh karena itu untuk membuat NPC dapat dengat optimal bergerak maka akan digunakan algoritma *Particle Swarm Optimizatio* (PSO) sebagai algoritma untuk pergerakan berkelompoknya. NPC juga harus bisa untuk menentukan bagaimana mereka harus memutuskan keputusan ketika mendapat masukan dari luar. Untuk pengambilan keputusannya akan digunakan *Finite State Machine* (FSM). PSO akan membuat NPC bergerak secara berkelompok dan memilih pemimpinnya untuk diikuti. Algoritma FSM akan membuat NPC dapat mengambil keputusan ketika terjadi masukan. FSM terdiri dari beberapa *state* yang akan berpindah ke *state* lainnya ketika mendapatkan masukan dari luar. *State* akan berpindah ketika suatu *event* terjadi.

Dari hasil pengujian untuk kedua metode ini maka didapatkan hasil keakurasian sebesar 80% untuk metode FSM ketika NPC harus mengambil keputusan ketika mendapatkan masukan dari pemain. Dan juga untuk PSO dapat menentukan pemimpin dari setiap NPC ketika berada di dekat pemain.

Kata Kunci : *Particle swarm optimization*, *finite state machine*, kecerdasan buatan, pengambilan keputusan, pergerakan kelompok

1. PENDAHULUAN

Multiplayer Online Battle Arena (MOBA*)* merupakan jenis video *game* dimana seorang atau lebih *player* akan memainkan satu tipe karakter tertentu dimana tugasnya adalah untuk menghancurkan *tower* dan struktur bangunan lawan. Dalam permainan akan dibagi ke dalam dua tim. Setiap tim terdiri dari 5 pemain dan kelompok NPC. Tim ini akan saling menyerang dalam arena yang di dalamnya terdapat *tower* di setiap jalurnya. Permainan MOBA adalam salah satu *video game multyplater* dimana terdapat satu atau lebih pemain yang bermain dalam permainan ini. Permainan akan berakhir jika bangunan utama milik lawan atau *player* hancur. Untuk mencapai tujuannya player akan dibantu oleh sekumpulan NPC *(Non Playable Character)*. NPC ini bisa disebut *creeps*. NPC ini hanya akan bergerak menelusuri jalur yang sudah ada. NPC ini akan bertemu pemain lawan dan NPC lawan dalam perjalananya kemudian NPC ini juga bertugas untuk menghancurkan *tower*. Setiap kelompok *creeps* ini akan melawan pemain untuk menghambat pemain sehingga pemain tidak dapat dengan mudah menyelesaikan misinya yang yaitu menghancurkan markas musuh.

Dalam menyerang pemain NPC harus bergerak dalam berkelompok. NPC tidak dapat berpisah ketika menyerang pemain, pemain dapat mengekploitasi hal tersebut dan membuat NPC lebih mudah untuk dikalahkan oleh pemain. Jika NPC saling terpisah dengan NPC lainnya di dalam kelompoknya juga akan membuat NPC tidak dapat melakukan serangan secara optimal kepada pemain. NPC juga harus dapat menentukan apa yang NPC harus serang. NPC yang hanya menyerang *tower* dan tidak menyerang selain itu akan membuat permainan berlangsung secara mudah. Karena NPC menjadi

tidak dapat merespons ketika diserang oleh tim lain. NPC harus selalu menyerang pemain agar permainan harus tetap berjalan menantang bagi pemain.

Pada penelitian sebelumnya oleh Victor do Nascimento Silva dan Luiz Chaimowicz [1] mengatakan jika NPC pada *game* MOBA sangat berpengaruh pada jalannya permainan. *Creeps* pada MOBA berguna agar karakter *player* dapat mendapatkan *experience points* yang digunakan untuk karakter pemain bertambah kuat. Pada tipe permainan MOBA kecerdasan buatan pada NPC digunakan dalam bagaimana mereka bergerak dalam kelompok dan bagaimana mereka menyerang dan menentukan apa yang harus diserang oleh NPC. Dalam bergerak NPC tidak dapat bergerak secara masing-masing. NPC akan terus bergerak dalam kelompok. Dari tempat mereka muncul sampai ke tujuan NPC. Dalam pengambilan keputusannya NPC juga harus dapat menentukan mana yang harus mereka pilih terlebih dahulu dalam melakukan tindakan. Setiap tindakan yang diambil harus berpengaruh langsung terhadap pemain yang akan melawan NPC. Sehingga masalah yang muncul adalah NPC harus dapat bergerak secara berkolompok dan menentukan keputusan yang diambil ketika berada dalam permainan.

Algoritma *Particle Swarm Optimization* (PSO) adalah suatu algoritma yang dapat dipakai untuk menentukan perilaku suatu NPC secara berkelompok. Algoritma PSO merupakan algoritma yang terinspirasi dari kawanan burung yang bergerak secara Bersama dan berselarasan dalam kelompoknya. Dalam kelompok NPC akan ditunjuk satu pemimpin yang akan menentukan perilaku yang akan dilakukan kelompok NPC tersebut. Setiap NPC diinisialisasikan ke dalam bentuk partikel. NPC akan bergerak ke arah yang sesuai dengan arah keselarasam dam konvergen [2]. Nantinya setiap NPC akan merubah sampai ke dalam posisi yang memungkinkan ketika menyerang *tower*, *player* atau NPC musuh.

*Finite State Machine*s (FSM) adalah sebuah metode yang terdiri dari serangkaian *State* yang dapat menentukan pengambilan keputusan. Setiap *State* dapat berpindah dari *State* lainnya jika memenuhi kondisi yang telah ditentukan. NPC dapat melakukan aksi jika mendapatkan input dalam hal ini jika bertemu dengan player. Dalam penelitian oleh Michael Waltham dan Deshen Moodley disebut jika FSM adalah yang paling sedikit memakai *resources*, FSM cocok digunakan di dalam *game* dimana NPC bukanlah inti dari *game* tersebut.

Dalam penelitian ini algoritma PSO akan digunakan pada NPC yang berkelompok sebagai kecerdasan buatan untuk perilaku berkelompok NPC dan FSM sebagai pengambil keputusannya dan akan mengukur performansi dari algoritma tersebut.

1.1 *Video Game*

Video game merupakan salah satu sarana hiburan yang dapat dilakukan manusia secara aktif. Menonton film, mendengar lagu merupakan kegiatan hiburan yang pasif. Sedangkan bermain *video game* merupakan kegiatan dimana seorang pengguna memainkan sesuatu yang terjadi di dalam game tersebut. Terdapat dunia di dalam game tersebut yang memiliki aturan yang harus dipahami pengguna. Aturan tersebut adalah sesuatu yang harus dikerjakan agar tujuan dari game tersebut tercapai.

1.2 *Game MOBA*

MOBA merupakan modifikasi dari game RTS. Jika di dalam game RTS pemain harus membangun banyak unit dan mengontrolnya, dalam game MOBA kita hanya mengontrol satu karakter dan unit pasukan dikendalikan oleh kecerdasan buatan. MOBA juga memiliki unsur RPG dimana karakter yang pemain mainkan bisa bertambah kuat dengan *level up* dan memiliki lebih banyak kemampuan setiap *levelnya*..

1.3 Kecerdasan Buatan

Kecerdasan buatan merupakan salah satu bagian ilmu komputer yang membuat agar mesin komputer agar data melakukan kegiatan atau pekerjaan seperti dan sebaik yang dilakukan manusia. Awal diciptakannya, komputer merupakan hanyalah sebuah alat hitung saja. Semakin berkembangnya manusia, komputer saat ini dipakai dibanyak bidang. Komputer bahkan dituntut semakin bisa untuk melakukan pekerjaan manusia. [3]

Manusia mampu menyelesaikan pekerjaannya karena memiliki kecerdasan dalam otak mereka. Kecerdasan ini didapat manusia melalui proses yang dinamakan belajar. Manusia semakin mampu dalam menyelesaikan pekerjaanya karena melalui proses belajar yang tidak sebentar, bahkan dengan percobaan berkali-kali, sehingga manusia memiliki kecerdasan dalam menyelesaikan pekerjaan dan masalahnya.

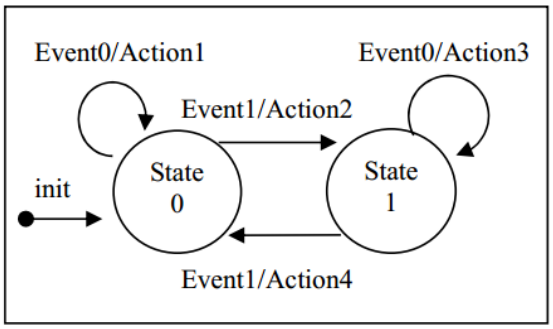
Agar komputer dapat melakukan pekerjaan sebaik ataupun lebih baik dari manusia, diperlukan suatu kecerdasan yang harus ditanam di dalam komputer. Kecerdasan buatan akan memberikan metode-metode bagi komputer agar komputer dapat memiliki nalar seperti manusia dan membuat komputer menjadi semakin pintar.

1.4 *Finite State Machine*

be *Finite State Machine* adalah sebuah metodelogi perancangan sistem control yang menggambarkan tingkah laku dan prinsip kerja sistem dengan menggunakan tiga hal berikut : *State* (keadaan), *event* (kejadian) dan *action* (aksi). Dalam periode waktu tertentu sistem berada pada satu sistem *State* yang sedang dijalankan. Kemudian sistem akan beralih atau berubah menuju *State* lain

jika mendapat *input* dalam kejadian tertentu, baik yang berasal dari luar ataupun komponen sistem tersebut sendiri. Perubahan keadaan ini juga biasanya disertai oleh aksi yang akan dilakukan oleh sistem ketika merespon *input* yang terjadi. Aksi yang dilakukan berikut dapat berupa aksi yang sederhana maupun rangkaian aksi proses yang bersifat kompleks [6].

.



Gambar 4. Diagram Finite State Machine [7]

Gambar 4 memperlihatkan FSM dengan dua buah *State* dan dua buah *input* serta empah buah *outpu* yang berbeda terlihat di dalam gambar, ketika mulai dinyalakan, sistem akan bertransisi menuju *State()*, pada keadaan ini sistem akan memunculkan hasil *action1* jika terjadi *input*an *event*(), sedangkan jika terjadi *event1* maka *action2* akan dieksekusi kemudian sistem akan transisi menuju *State1* dan seterusnya.

Implementasi FSM dapat diterapkan pada pemograman berorientasi objek atau yang sering disebut OOP. Kelebihan OOP dibanding yang lain dalam implementasi FSM adalah kefleksibilitasannya yang tinggi dan pemeliharaannya yang sudah mudah baik pada sistem dalam level manapun. Selain itu juga FSM diuntungkan dengan kelebihan OOP dalam penggunaan kembali kode yang sudah diketik. [6]

1.5. Algoritma *Particle Swarm Optimization*

Algoritma PSO merupakan suatu algoritma yang dikembangkan oleh Kennedy dan R. Eberhart di tahun 1995 yang mendapatkan inspirasi dari perilaku pergerakan kelompok hewan seperti ikan, hewan herbivora, burung dan serangga seperti semut dan lebah. Setiap objek dari hewan tersebut akan disederhanakan menjadi partikel-partikel yang mudah digerakan. Algoritma ini menerapkan sifat masing-masing individu dalam satu kelompok besar dan lalu menggabungkan sifat-sifat tersebut agar dapat menyelasaikan masalah yang ada [2]. Algoritma PSO merupakan metode kecerdasan buatan *swarm intelligence* yang berasaskan perilaku kolektif dalam suat kelompok dan setiap individu dalam kelompok tersebut dapat mengatur dirinya sendiri. Dalam *swarm intelligence* algoritma PSO merupakan metode yang dapat digunakan untuk mengatur posisi partikel terbaik. Konsep pengotimalan partikel individu ini diperkenalkan oleh Kennedy dan R. Ebhert setiap pertikel memiliki nilai kebugaran yang dievaluasi oleh fungsi dioptimalkan, selain itu parameter setiap partikel memiliki beberapa hal berikut ini konstanta kecapatan, koordinat posisi, koordinat posisi terbaik, koordinat posisi terbaik secara global. Dari parameter tersebut yang digunakan di atas dapat membentuk perubahan posisi yang dilakukan secara bersamaan di tiap-tiap partikel. Sebelum mengubah posisi, partikel tersebut terlebih dahulu mengubah kecepatannya. Perubahan kecepatan ini menggunakan persamaan dasar PSO

Update partikel dapat digunakan dengan persamaan dasar PSO : [5]

$$Vi(t+1) = Vi(t) + C1\ r1\ (Pbest\ i - Xi(t)) + C2\ r2(Gbesti - Xi(t)) \quad (2.1)$$

Update posisi partikel dapat dilakukan dengan menggunakan persamaan dasar PSO : [5]

$$Xi(t+1) = Xi(t) + Vi(t+1) \quad (2.2)$$

Mencari nilai *fitness*

$$F(X) = \sqrt{((player.\ x - NPC.\ x)^2 + (player.\ y - NPC.\ y)^2)} \quad (2.3)$$

Di mana :

Vi (t) : kecepatan partikel ke (i) pada iterasi ke (t)
Xi (t) : Posisi partikel ke (i) pada iterasi ke (t)
V : *Velocity* (kecepatan partikel)
X : Koordinat partikel (posisi partikel)
i : Partikel ke (1,2,3 ..... N)
C1 dan C2 : Faktor *cognitive* dan sosial nilai ini adalah nilai konstanta bernilai diantara (0 dan 1)
r1 dan r2 : Bilangan acak antara 0 - 1
Pbest : Posisi terbaik yang dimiliki setiap partikel
Gbest : posisi partikel yang memiliki posisi paling baik dari partikel lainnya.

2. ISI PENELITIAN

2.1. Analisis PSO

2.1.1. Penyelesaian Kasus dengan PSO

Adapun dalam menyelasaikan masalah hal dalam PSO dibutuhkan langkah-langkah berikut ini :

1. Inisialisasi Partikel
   Langkah awal dimana jumlah dan koordinat partikel akan ditentukan terlebih dahulu
2. Melakukan perhitungan nilai fitnees dari setiap partikel
   Untuk mendapatkan nilai fitness dari setiap partikel dibutuhkan nilai koordinat untuk dievaluasi di dalam fungsi *fitness*.
3. Menentukan Pbest
   Pbest adalah posisi terbaik terbaik setiap partikel. Merupakan posisi yang ditempati oleh partikel saat ini.
4. Menentukan Gbest
   Nilai Gbest merupakan nilai partikel terbaik yang sudah melalui fungsi *fitness* dan yang paling optimal.
5. Menghituang Nilai *Velocity*

Nilai ini didapat dari persamaan nilai *velocity* untuk memperbarui kecepatan.

6. Menghitung nilai koordinat baru
   Dengan memasukan nilai V ke dalam persamaan koordinat baru maka nilai koordinat baru akan ditemukan.

2.1.2. Analisis Masukan

Dalam algoritma PSO untuk mencapai suatu titik tujuan setiap partikel akan menentukan partikel mana yang memiliki posisi terbaik terlebih dahulu kemudian partikel lain yang tidak memiliki posisi terbaik akan memperbaharui kecepatan dan posisinya untuk mencapai posisi partikel yang memiliki posisi paling baik tersebut .untuk memperbaharui kecepatan dan posisi dari tiap partikel tersebut dibutuhkan masukan, diantaranya :

1. Nilai koordinat posisi setiap partikel
   Nilai ini didapat dari titik nilai koordinat setiap NPC yang sudah ditentukan terlebih dahulu.
2. Nilai Fitness
   Nilai *fitness* didapat dari mengevaluasi nilai koordinat setiap NPC kedalam fungsi *fitness* dimana fungsi tersebut merupakan fungsi yang akan dioptimasi.
3. Nilai konstanta Ci dan C_2
   Nilai ini merupakan konstanta faktor cognitive dan social. Nilai C_1 dan C_2 berkisar di antara 0 sampai dengan 1. Adapun nilai ini akan ditentukan terlebih dahulu dalam tahao analisis yaitu 1.
4. Bilangan acak r_1 dan r_2
   Nilai ini merupakan nilai random dan akan diperoleh nilai dari 0 sampai dengan 1.
5. Nilai Pbest
   Nilai Pbest atau *personal best* adalah nilai dari posisi terbaik yang dimiliki oleh NPC nilai ini akan diambil dari setiap nilai koordinat NPC nya.
6. Nilai Gbest
   Nilai Gbest atau *global best* adalah nilai dari posisi NPC yang memiliki posisi paling baik dari partikel lainnya. Nilai Gbest didapat dari nilai koordinat NPC yang paling baik dimana NPC yang memiliki nilai Gbest adalah NPC yang meimiliki *fitness* paling optimal. Nilai *fitness* optimal dinilai dari kasus yang dimiliki yaitu maksimasi atau minimasi.

2.1.3. Analisis Kasus

Diketahui

Tiga buah NPC berada dalam arena permainan. NPC ini berada di awal mula permainan. Karakter pemain berada di sisi lain arena permainan juga. NPC dan pemain memiliki nilai titik koordinat yang menentukan titik lokasi dalam permainan. NPC akan bergerak menuju pemain dengan mengikuti salah satu NPC yang akan dijadikan pemimpin. Maka perhitungan dari kasus ini adalah sebagai berikut.

Kasus minimasi

Persamaan fungsi fitness

F (X) = √ ((player. x— NPC. x)^2+ (player. y— NPC. y)^2

Titik koordinat player = (400,280)

C1 = C2 = 1 // Nilai C1 dan C2 sudah ditentukan (nilai antara 0 sampai dengan 1)

r1 , r 2 = nilai random // nilai berkisar antara 0 sampai dengan 1

Banyak partikel = 3

Nilai titik Koordinat awal NPC :

Koordinat NPC1 : X1= (170,175)

Koordinat NPC2 : X2 = (330,190)

Koordinat NPC 3 : X3 = (260,150)

Nilai Velocity setiap partikel saat ini

Velocity NPC 1 : V1= 0

Velocity NPC 2 : V2= 0

Velocity NPC 3 : V3= 0

Penyelesaian :

Iterasi 0

Nilai random: r1 = 0,1 r2 = 0,4

Langkah Pertama

Nilai koordinat NPC saat ini :

Koordinat NPC1 : X1= (170,175)

Koordinat NPC2 : X2 = (230,190)

Koordinat NPC3 : X3 = (260,150)

Langkah kedua : masukan nilai koordinat NPC ke dalam fungsi fitness

F (X) = √ ((player. x— NPC. x)^2+ (player. y— NPC. y)^2

Fitness NPC1 : F(X) 1 = √ (400-170)^2+ (280-175)^2 = 152

Fitness NPC2 : F(X) 2 = √ (400-330)^2 +(280-190))^2 = 14

Fitness NPC3 : F(X) 3 = √ (400-260)^2 + (280-150)^2 = 91

Langkah Ketiga

Nilai Pbest = nilai koordinat partikel saat ini

Pbest NPC 1 : Pbest1 = (170,175)

Pbest NPC 2 : Pbest2= (330,190)

Pbest NPC 3 : Pbest3 = (260,150)

Langkah Keempat

Nilai Gbest merupakan nilai titik koordinat NPC yang memiliki nilai fitness paling optimal dimana dalam kasus ini akan meminimalkan jarak antara NPC dengan player. NPC dengan jarak paling sedikit merupakan NPC yang paling optimal dan dijadikan pemimpin. Adapun nilai fitness yang di dapat :

Fitness NPC1 : F(X) 1 = 152

Fitness NPC2 : F(X) 2 = 14

Fitness NPC3 : F(X) 3 = 91

Dengan melihat nilai fitness setiap NPC diatas, NPC yang memiliki nilai fitness paling rendah adalah NPC ke 2. Maka didapat nilai Gbest = koordinat NPC ke 2 = (330,190).

Langkah Kelima

NPC lain yang bukan pemimpin memperbaharui kecepatan dengan persamaan sebagai berikut :

Vi (t+1) = Vi(t)+C1.r1(Pbest i – Xi(t)) + C2.r2(Gbest i –Xi(t))

Perhitungan nilai perbaharui kecepatan tiap NPC :

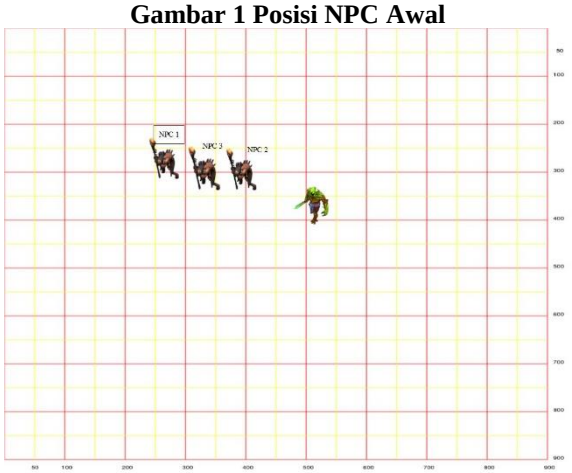
NPC1 : Vx1 (1) = 0 + 0.1(170– 170) + 0.4(330 – 170) = 80
Vy1 (1) = 0 + 0.1(175-175) + 0.4(190-175) = 7.5
NPC3 : Vx3 (1) = 0 + 0.1(360- 360) + 0.4(430 - 360) = 35
Vy3 (1) = 0 + 0.1(150-150) + 0.4(190-150) = 20
Langkah Keenam
NPC lain yang bukan pemimpin memperbaharui posisi dengan persamaan sebagai berikut :
Xi (t+1) = Xi (t) +Vi (t+1)
Perhitungan perbaharui kecepatan tiap NPC :
NPC 1 : X1(1) = 170 + 80 = 150
Y1(1) = 175 + 7.5 = 182.5
NPC3 : X3(1) = 260 + 35 = 295
Y3(1) = 150 +20 =170
Nilai Koordinat NPC awal
X1(0) = (170,175)
X3(0) = (260,150)
Nilai koordinat setelah perbaharuan kecepatan
X1(1) = (150,182.5)
X3(1) = (295,170)
Langkah Ketujuh

Mememeriksa posisi tempat NPC apakah sudah konveregen atau belum Dengan melihat hasil perhitungan pada iterasi ke 0 diatas. Jika belum konvergen maka perhitungan akan diulang di iterasi berikutnya.

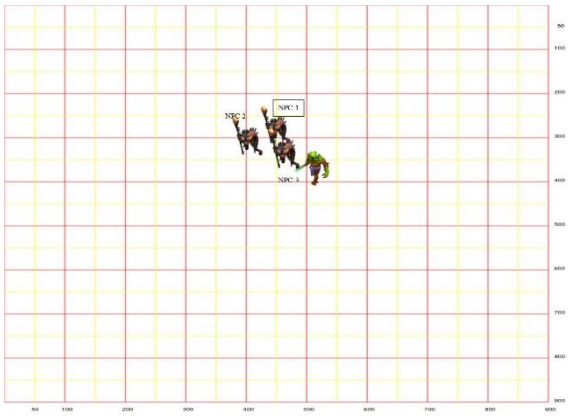
2.2.4. Perubahan Posisi NPC



Gambar 1 Posisi NPC Awal



Gambar 2 Posisi NPC di Iterasi 0



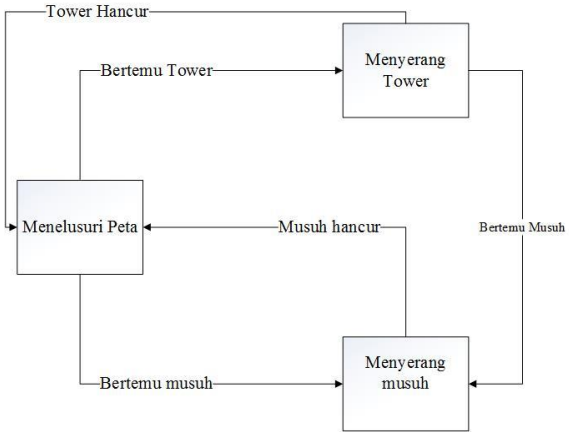
Gambar 3 Posisi NPC di Iterasi 1

2.2. Analisis FSM

2.2.1. *Finite State Machine* pada NPC

Dalam penilitian ini FSM akan digunakan NPC untuk pengambilan keputusan dalam permainan ini. *State* yang aktif dalam satu waktu hanya bisa aktif satu saja. Setiap *state* akan merepresantasikan suatu aksi seperti menyerang dan memilih target. *State* pertama yang aktif pada awal mula permainan adalah mencari *tower* dalam jalur yang sudah disediakan.

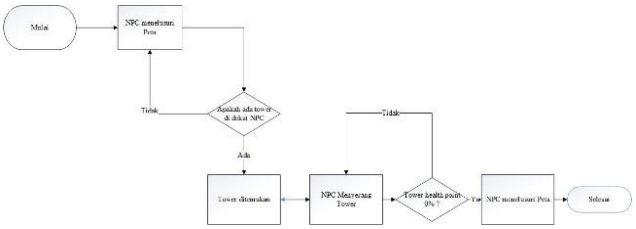
State pertama ketika permainan dimulai adalah untuk mencapai tower pertama dalam hal ini NPC akan mencari jalur dengan algoritma *path finding*. *State* awal mencari tower ini akan terus aktif sampai NPC bertemu tower. Jika sudah menemukan tower maka *state* NPC akan bertransisi menuju menghancurkan tower tersebut. *State* tersebut akan terus aktif sampai tower hancur. Jika dalam *state* tersebut bertemu NPC musuh maka *state* akan bertransisi menuju menghancurkan musuh. NPC hanya akan mengeksekusi satu *state* dalam satu waktu tertenu saja. Gambara *state* awal ini bisa dilihat pada gambar di bawah ini :



Gambar 5. *State* pada NPC

2.2.2. *State* NPC Menyerang *Tower*

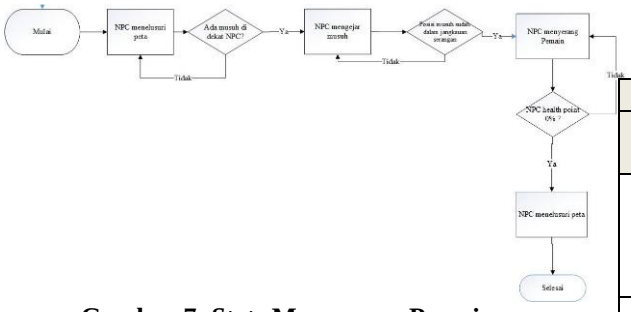
State ini merupakan transisi langsung dari *state* mencari *tower* ketika NPC sudah menemukan *tower*. Yang dilakukan NPC adalah pertama adalah mengecek apakah di dekat *tower* terdapat NPC musuh ataupun pemain lawan. Jika terdapat musuh, maka NPC akan menjalankan *state* menyerang musuh terlebih dahulu. *State* ini akan bertransisi kelmbali menjadi *state* menyerang *tower* ketika musuh sudah tidak ada lagi dalam jangkauan NPC. *state* ini akan membuat NPC menyerang *tower* sampai *tower* tersebut hancur. *state* akan bertransisi kembali ke mencari *tower* jika *tower* sudah hancur dan tidak ada musuh di sekitar NPC.



Gambar 6. Diagram *State* Menyerang *Tower*

2.2.3. *State* Menyerang Pemain

State ini akan dilakukan ketika NPC berada di dekat dengan pemain. Ketika mengejar pemain NPC akan memakai metode PSO ketika menyerang secara berkelompok. NPC yang akan mengecek beberapa hal seperti jarak dengan pemain, dan *health point* musuh. Ketika NPC sudah mengecek hal tersebut maka NPC dapat melakukan serangan kepada musuh. NPC dapat menyerang secara berkelompok ataupun individu. Setelah *health point* musuh mencapai 0% maka NPC akan bertransisi ke *state* lain. NPC akan memprioritas kan menyerang karakter pemain terlebih dahulu daripada karakter NPC.



Gambar 7. *State* Menyerang Pemain

2.5 Pengujian Sistem

Pengujian metode pada permainan *miniMOBA* untuk metode PSO dan FSM yaitu pengujian terhadap ketepatan pengambilan keputusan untuk NPC ketika mendapatkan masukan dari luar.

2.5.2. Pengujian Finite State Machine

Pada pengujian *finite state machine* NPC akan disimpan di dalam sebuah kondisi dan akan dilihat bagaimana NPC tersebut akan mengambil keputusan yang sudah ditetapkan sebelumnya. Untuk *state* NPC bergerak pada arena permainan, NPC akan ditetapkan di dalam kelompok dan akan dicoba untuk diberikan kondisi. Pada *state* ini algoritma PSO akan digunakan.

Tabel 1. Pengujian FSM NPC bergerak

Kesimpulan dan Hasil Uji			
Kondisi	Hasil yang Diharapkan	Hasil Pengamatan	Kesimpulan
NPC bergerak pada *map* menuju *tower*	NPC bergerak ke *tower* pertama	NPC bergerak ke *tower*	Berhasil
Pemain berada di dekat NPC	NPC mengejar pemain	NPC mengejar pemain	Berhasil
NPC mengejar pemain	NPC mengikuti pemimpin	NPC bergerak mengikitu pemimpin	Berhasil

Pada pengujian untuk *state* menyerang pemain, NPC dan *tower* akan disatukan. NPC pada dasarnya akan menyerang *target* yang memiliki posisi lebih dekat dengan NPC, akan tetapi untuk pengujian kali ini, *healthpoint* akan dijadikan *parameter* baru untuk pengujian dan akan dilihat apakah NPC dapat mengambil keputusan dengan benar dan memilih *target* yang seharusnya dipilih.

Tabel 2. Pengujian FSM Menyerang NPC

Kesimpulan dan Hasil Uji			
Kondisi	Hasil yang Diharapkan	Hasil Pengamatan	Kesimpulan
NPC berada di dekat NPC lawan	NPC menyerang NPC lawan	NPC menyerang NPC lawan	Berhasil
NPC bertemu pemain ketika sedang menyerang NPC lain	NPC menyerang pemain terlebih dahulu	NPC menyerang pemain	Berhasil

2.5.1. Pengujian PSO

Pengujian kebenaran posisi pemilihan NPC pemimpin ini merupakan pengujian yang dilakukan terhadap *game* untuk menyesuaikan nilai *fitness* yang optimal dengan fungsi minimasi dengan NPC

yang terpilih menjadi pemimpin.

$$F(X) = j\,(player.x - NPC.x)^2 + (player.y - NPC.y)^2$$

Dimana :
player.x adalah koordinat player X
player.y adalah koordinat player Y
NPC.x adalah koordinat player X
NPC.y adalah koordinat player Y

Tabel 3. Tabel Pengujian Pertama

ɔmor	Hasil Uji				
		Koordinat NPC		Koordinat Pematin	
		Titik X	Titik Y	Titik X	Titik Y
1	NPC 1	310	116		
2	NPC 2	383	322.99		
3	NPC 3	437	152.87		

Nilai-nilai tersebut akan dimasukan ke dalam fungsi *fitness* untuk menentukan NPC yang paling cocok untuk dijadikan pemimpin.

Tabel 4. Hasil Uji

Nomor	NPC	Hasil Pengujian	
		Nilai *fitness*	Keterangan
1	NPC 1	329	Pengikut
2	NPC 2	119	Pemimpin
3	NPC 3	255	Pengikut

Untuk pengujian kondisi kedua :

Tabel 5. Pengujian Kondisi Kedua

ɔmor	Hasil Uji				
		Koordinat NPC		Koordinat Pematin	
		Titik X	Titik Y	Titik X	Titik Y
1	NPC 1	579	294		
2	NPC 2	437	151		
3	NPC 3	381	196		

Nilai-nilai tersebut akan dimasukan ke dalam fungsi *fitness* untuk menentukan NPC yang paling cocok untuk dijadikan pemimpin.

Tabel 6. Hasil Uji kondisi Kedua

Nomor	NPC	Hasil Pengujian	
		Nilai *fitness*	Keterangan
1	NPC 1	125	Pemimpin
2	NPC 2	192	Pengikut
3	NPC 3	267	Pengikut

3. PENUTUP

Pada penelitian ini telah menghasilkan sebuah kecerdasan buatan pada NPC di permainan *miniMOBA*, yang digunakan sebagai perilaku berkelompok dan juga sebagai pengambil keputusan. NPC dapat bergerak secara berkelompok dan menyerang pemain di dalam arena permainan. NPC juga dapat mengambil keputusan ketika terdapat masukan dari pemain ataupun NPC lainnya juga.

Algoritma FSM yang digunakan memberikan hasil sebesar 80% untuk NPC bagaimana mereka harus bertindak ketika terjadi suatu kondisi. Untuk pengujian untuk menentukan pemimpin dapat pada kasus uji, berhasil menetapkan pemimpin dengan nilai *fitness* 125.

Sistem yang saat ini dibangun masih dapat dikembangkan lagi sesuai dengan kebutuhan pengguna dan juga dapat mengembangkan *gameplay* yang lebih baik. Saran untuk pengembangan selanjutnya pada sistem yang dapat dijadikan pertimbangan yaitu :

a. Menambahkan jalur baru di dalam arena permainan.
b. Menambahkan *State* baru untuk NPC agar NPC mempunyai keputusan yang lebih baik.
c. Menambah jumlah NPC yang akan digunakan untuk penelitian
d. Tambah jumlah pemain pada penilitian.
e. Memberikan *map* yang mengharuskan NPC untuk mencari jalur terbaik.
f. Membuat jenis NPC yang berbeda-beda.

DAFTAR PUSTAKA

[1] V. do Nascimento Silva dan L. Chaimowicz, "On the Development of Intelligent Agents for MOBA Games," *Brazilian Symposium on Computer Games and Digital Entertainment*, vol. 14, hal. 143-152, 2014.

[2] S. Mu'min, M. Hariadi dan S. M. Susiki Nugroho, "Pergerakan Otonom Pasukan Berbasis Algoritma Boids Menggunakan Metode Particle Swarm Optimization," *Journal of Animation and Games Studies*, vol. 1, no. 1, hal. 1-16, 2015.

[3] S. Kusumadewi, Artificial Intelligence (Teknik dan Aplikasinya), Yogyakarta: Graha Ilmu, 2003.

[4] R. Roedavan, Unity Game Engine, Bandung: Informatika, 2016.

[5] R. E. dan Kennedy, "Particle Swarm Optimization," dalam *Conference on Neural Networks*, Perth, Australia, 1948.

[6] M. Fauzan Rahadian, A. Suyatno dan S. Maharani, "Penerapan Metode Finite State Machine pada Game "The Relationship"," *Jurnal Informatika Mulawarman*, vol. 11, no. 1, hal. 14-22, 2016.