Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN: 2548-964X
Vol. 3, No. 2, Februari 2019, hlm. 1792-1796 http://j-ptiik.ub.ac.id

Penerapan Algoritme *Finite State Machine* Berbasis *Fragment Shader* untuk Proses Pengambilan Keputusan pada *Non Player Character* (Studi Kasus *Game Battle Tank*)

Muhtadin Ziqi Maulana[1], Eriq Muh. Adams Jonemaro [2], Muhammad Aminul Akbar[3]

Program Studi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Brawijaya
Email: [1]muhtadinzm@gmail.com, [2]eriq.adams@ub.ac.id, [3]muhammad.aminul@ub.ac.id

Abstrak

Non player character (NPC) merupakan karakter dalam sebuah permainan yang tidak dikendalikan oleh pemain, melainkan dikendalikan melalui program komputer yang dibuat oleh manusia. Menurut model *game* AI NPC memiliki kemampuan untuk melakukan pergerakan dan pengambilan keputusan. Permainan *battle tank* yang dikembangkan penulis dalam penelitian kali ini juga memiliki NPC yang dikembangkan. Pada permainan tersebut peneliti menggunakan algoritme *finite state machine* (FSM) dalam proses pengambilan keputusan dari NPC. Namun terdapat sebuah ide tentang penerapan algoritme FSM yaitu dengan menggunakan *fragment shader*. Dengan penerapan algoritme FSM berbasis *fragment shader* diharapkan mendapatkan performa yang lebih baik. Dikarenakan proses dari *fragment shader* dilakukan dalam *graphics processing unit* (GPU). Sehingga proses yang dilakukan dapat dijalankan secara paralel antara proses pengambilan keputusan dan proses yang lain. Dalam penerapan algoritme FSM berbasis *fragment shader* membutuhkan tiga buah *map* yaitu *world map, agent map* dan *fsm map.* Setelah melakukan pengujian pengaruh jumlah NPC menggunakan 1, 5, 10 dan 15 buah NPC secara berurutan mendapatkan hasil rata-rata 147, 69, 24 dan 1 FPS. Sedangkan untuk pengujian pengaruh ukuran peta permainan menggunakan ukuran peta 20x20, 30x30 dan 40x40 secara berurutan menghasilkan nilai rata-rata 66, 61, dan 60 FPS.

Kata kunci: *non player character, finite state machine, fragment shader, battle tank, pengambilan keputusan.*

Abstract

Non player character (NPC) is a character in a game that is not controlled by players, but is controlled through computer programs made by humans. According to the gameAI model the NPC has the ability to make movements and decision making. The battle tank game that was developed by the author in this study also has an NPC developed. there is the game the researcher uses the finite state machine (FSM) algorithm in the decision making process of the NPC. But there is an idea about the application of the FSM algorithm that is by using a shader fragment. With the implementation of the FSM algorithm based on shader fragments, it is expected to get better performance. Because the process of the shader fragment is done in the graphics processing unit (GPU). So that the process carried out can be carried out in parallel between the decision making process and other processes. In applying FSM algorithms based on shader fragment requires three maps, namely world map, agent map and fsm map. After testing the effect of the number of NPCs using 1, 5, 10 and 15 NPCs, respectively, obtained an average yield of 147, 69, 24 and 1 FPS. Whereas for testing the effect of game map size using map sizes of 20x20, 30x30 and 40x40 in succession yielding an average value of 66, 61 and 60 FPS.

Keywords: *non player character, finite state machine, fragment shader, battle tank, decission making*

1. PENDAHULUAN

Kecerdasan buatan dalam *game* atau yang biasa disebut dengan *game* AI kebanyakan digunakan untuk menghasilkan perilaku cerdas pada *non player character* (NPC). Yaitu karakter dalam *game* yang tidak dikendalikan oleh pemain melainkan dikendalikan oleh komputasi yang telah dirancang untuk membuat sebuah karakter yang terlihat lebih cerdas.(Milllington, & Funge, 2009).

Game battle tank merupakan permainan yang sedang dikembangkan oleh penulis. Yang mana *game* ini memiliki tujuan untuk

menghancurkan lawan. Dengan *game play* pertarungan saling menembak anatara dua atau lebih *tank* atau kelompok *tank*. Pada saat ini penulis fokus pada pengembangan proses pengambilan keputusan dari NPC pada *game* tersebut.

Pengambilan keputusan pada penelitian yang akan dilakukan kali ini adalah menggunakan algoritme *finite state machine* (FSM). Metode FSM ini juga diterapkan pada penelitian implementasikan FSM pada *game save the animal* (Rizaldi, 2017). Dan hasil pengujian yang dilakukan menunjukkan bahwa metode FSM dapat berhasil diterapkan pada *game* tersebut dan berjalan sesuai dengan yang diharapkan.

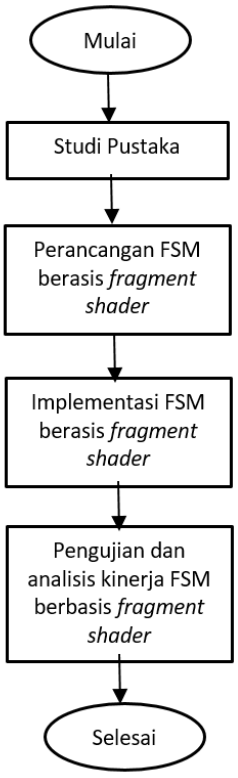
Finite State Machine (FSM) merupakan salah satu algoritme yang dapat diterapkan untuk pengambilan keputusan dalam pengembangan kecerdasan buatan dalam agen permainan. FSM sendiri memiliki beberapa *state* yang saling terhubung oleh transisi di dalam *graf*. Dalam FSM agen menempati suatu keadaan. Yang mana keadaan tersebut akan dikaitkan ke dalam masing-masing *state*. Sehingga, selama agen masih tetap dalam keadaan tersebut ia akan terus melaksanakan tindakan yang sama. Kelebihan dari metode FSM ini antara lain mudah diterapkan, membutuhkan komputasi yang ringan, dan mudah ditransfer dari abstrak menjadi kode program.

Graphics processing unit (GPU) adalah salah satu bagian komputer yang melakukan perhitungan matematis yang cepat, terutama untuk tujuan me*render* gambar. Dikarenakan saat ini banyak program grafis seperti *AutoCAD* dikembangkan. Oleh karena itu GPU muncul sebagai cara untuk mengurangi beban dari *Central Processing Unit*(CPU) dalam melakukan proses komputasi. GPU sendiri bekerja secara *pipeline* sehingga semakin cepat melakukan proses komputasi. Dalam menampilkan gambar secara umum GPU melakukan empat proses antara lain *vertex processing*, *clipping and primitive assembly*, *resterization* dan *fragment processing* (Angel, & Shreiner, 2012). *Fragmen shader* sendiri berfungsi untuk menyimpan warna yang nantinya akan ditampilkan.

Terdapat ide pengambangan yang menarik tentang *texture* dalam pengembangan *Finite State Machine* (Rudomin, Milan, & Herandez, 2005). Dimana *texture* disini berfungsi untuk menentukan *state* dari NPC itu sendiri. Dalam penelitian yang dilakukan oleh Isak, Erik dan Benjamin pada tahun 2005. Mereka menemukan ide untuk melakukan pendekatan yang lebih cepat, yaitu menggunakan pemrosesan pada GPU. Mereka menerapkan algoritme *Finite State Machine* berbasis *fragment shader* dengan tiga macam gambar atau tekstur. Antara lain *world space map, agent space map,* dan FSM *map*. Dalam penelitiannya merekan menerapkan metode ini dalam *predator-prey example.* Dan dengan metode ini menghasilkan kinerja yang lebih baik. Sehingga peneliti disini akan melakukan penelitian tentang penerapan algoritme *finite state machine* berbasis *fragment shader* untuk proses pengambilan keputusan pada *non player character* (studi kasus *game battle tank*).

2. METODOLOGI PENELITIAN

Penelitian yang dilakukan bersifat implementatif. Langkah-langkah metode yang digunakan pada penelitian ini dapat dilihat pada Gambar 1.



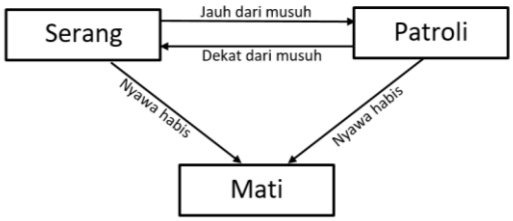
Gambar 1. Diagram alir metode penelitian

Pada tahap pertama dalam metodologi penelitian yaitu studi pustaka. Pada tahap ini penulis melakukan pengumpulan teori, informasi dan konsep yang diperlukan dengan sumber yang dapat dipercaya. Pada tahap selanjutnya penulis melakukan perancangan FSM berbasis *fragment shader*. Yang mana pada

tahap tersebut meliputi tentang perancangan FSM, perancangan *agent map, world map* dan FSM *map*. Tahap selanjutnya dilakukan implementasi FSM berbasis *fragment shader*. Dimana pada tahap ini dilakukan implementasi hasil dari tahap perancangan. Pada tahap terakhir adalah dilakukan pengujian dan analisi kinerja FSM berbasis *fragment shader*. Pada tahap ini penulis melakukan pengujian pengaruh jumlah NPC dan pengaruh ukuran peta permainan. Pengujian tersebut dilakukan untuk mendapatkan nilai *frame per second* (FPS) sebagai representasi performa algoritme FSM berbasis *fragment shader*. Kemudian penulis melakukan analisis terhadap hasil yang didapatkan.

3. PERANCANGAN DAN IMPLEMENTASI

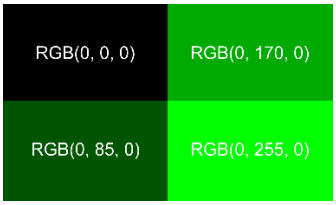
Pada tahap perancangan dilakukan perancangan FSM pada NPC bertujuan untuk merancang skenario yang dilakukan oleh agen NPC. Diagram FSM untuk agen *tank* dapat dilihat pada Gambar 2. Dalam perancangan FSM yang dibuat oleh penulis terdapat tiga buah *state* yaitu serang, patroli dan mati. *State* serang dijalankan ketika NPC berada dekat dengan musuh. Untuk *state* patroli sendiri dijalankan ketika NPC berada jauh dari musuh. Sedangkan *state* mati terjadi saat jumlah nyawa dari NPC adalah nol.



Gambar 2. Diagram FSM agen *tank*

Dalam melakukan implementasi algoritme FSM berbasis *fragment shader* diperlkan tiga buah *map*. Diantaranya *agent map, world map* dan FSM *map*.

World map digunakan sebagai menyimpan nilai dari *event* seperti jarak dengan musuh dan jumlah nyawa. *World map* sendiri digunakan untuk merubah nilai *event* supaya dapat dijalankan sebagai *transition rule. World map* bisa dilihat pada Gamber 3.



Gambar 3. *World map*

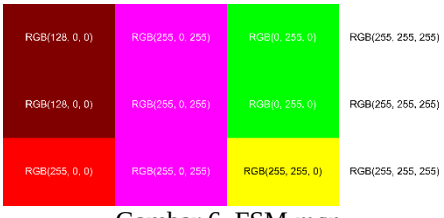
Agent map berisi informasi tentang agen NPC. Dalam penerapan pada permainan *battle tank* ini *agent map* menyimpan nilai dari *state* yang sedang sedang dijalankan oleh NPC. Untuk *agent map* dapat dilihat pada Gambar 4.

Gambar 4. *Agent map*

FSM *map* merupakan tempat dari proses pengambilan keputusan dari NPC. Dengan memanfaatkan posisi *state* sebelumnya dan *event* yang sedang terjadi. Untuk proses perancangan fsm *map* dapat dilihap pada Gambar 5. Yang tedapat tiga buah *state*. Untuk hasil dari fsm *map* sendiri terdapat pada Gambar 6.

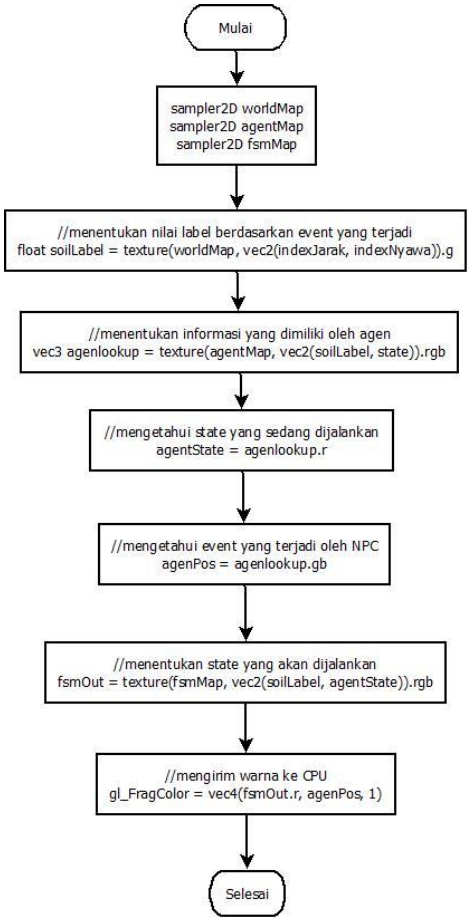
R/GB	Dekat musuh dan nyawa > 0	Dekat musuh dan nyawa = 0	Jauh musuh dan nyawa > 0	Jauh musuh dan nyawa = 0
Patroli (State=1)	2, 1, 1 RGB(128, 0, 0)	3, 1, 2 RGB(255, 0, 255)	1, 2, 1 RGB(0, 255, 0)	3, 2, 2 RGB(255, 255, 255)
Serang (State=2)	2, 1, 1 RGB(128, 0, 0)	3, 1, 2 RGB(255, 0, 255)	1, 2, 1 RGB(0, 255, 0)	3, 2, 2 RGB(255, 255, 255)
Mati (State=3)	3, 1, 1 RGB(255, 0, 0)	3, 1, 2 RGB(255, 0, 255)	3, 2, 1 RGB(255, 255, 0)	3, 2, 2 RGB(255, 255, 255)

Gambar 5. Perancangan fsm *map*



Gambar 6. FSM *map*

Setelah membuat tiga buah *map* untuk mengimplementasikan algoritme FSM berbasis *fragment shader* masing-masing *map* akan diproses pada program *fragment shader* yang akan diproses pada *graphics processing unit* (GPU). Pada gambar 7. Merupakan diagram alir untuk mengimplementasikan algoritme FSM berbasis *fragment shader*.



Gambar 7. Diagram alir proses program *fragment*

4. PENGUJIAN DAN ANALISIS

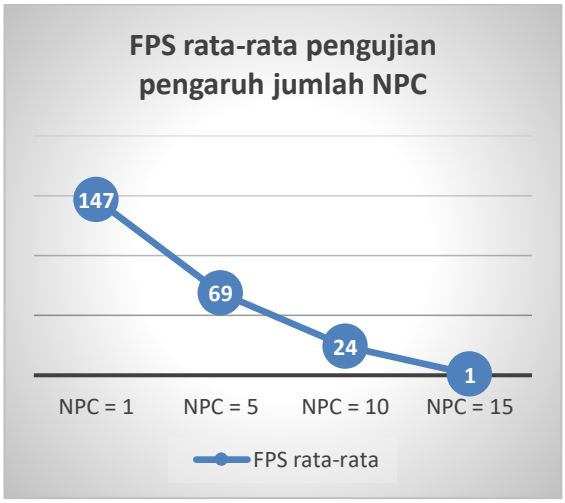
Dalam pengujian algoritme FSM berbasis *fragment shader* penulis menggunakan dua skenario pengujian. Yang pertama penulis menguji pengarus jumlah NPC. Kedua penulis melakukan pengujian pengaruh ukuran peta permainan. Skenario pengujian dilakukan untuk mengetahui performa dari algoritme FSM berbasis *fragment shader* untuk proses pengambilan keputusan pada permainan *battle tank*. Dalam pengujian tersebut didapatkan hasil *frame per second* (FPS) yang dapat merepresentasikan dari performa algortime tersebut.

4.1. Spesifikasi Perangkat

Digunakan beberapa perangkat untuk melakukan pengujian algoritme FSM berbasis *fragment shader*. Perangkat yang diperlukan berupa perangkat keras dan perangkat lunak. Spesifikasi perangkat keras yang digunakan dalam melakukan pengujian adalah *processor intel core* i5-7200U 2.5GHz, *Harddisk* 1000GB, RAM 4GB, *VGA Intel HD Graphics* 620, Monitor 15 inci resolusi *Full* HD dan *Keyboard*. Untuk spesifikasi perangkat lunak yang digunakan adalah Windows 10 64 bit dan Visual Studio 2017.

4.2. Pengujian Pengaruh Jumlah NPC

Dalam pengujian pengaruh jumlah NPC penulis melakukan empat buah pengujian dengan jumlah NPC yang berbeda. Penulis menggunakan 1, 5, 10 dan 15 NPC dalam melakukan pengujian pengaruh jumlah NPC.

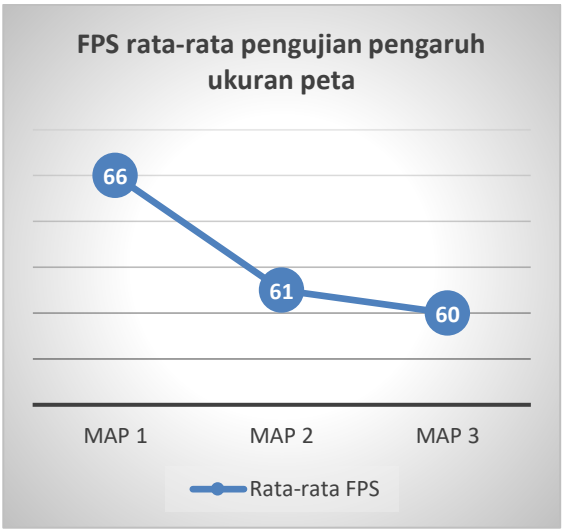


Gambar 8. Hasil pengujian pengaruh jumlah NPC

Pada Gambar 8. merupakan hasil dari pengujian pegaruh jumlah NPC. Dalam diagram tersebut menunjukkan bahwa pengujian menggunakan satu buah NPC mendapatkan hasil rata-rata 147 FPS. Dan untuk pengujian menggunakan lima buah NPC mendapatkan hasil rata-rata 69 FPS. Untuk pengujian menggunakan sepuluh buah NPC mendapatkan hasil rata-rata 24 FPS. Dan untuk pengujian menggunakan 15 buah NPC mendapatkan hasil rata-rata 1 FPS.

4.3. Pengujian Pengaruh Ukuran Peta Permainan

Dalam pengujian pengaruh ukuran peta permainan. Penulis menggunakan tiga buah ukuran peta yang berbeda. *Map* 1 penulis menggunakan ukuran peta 20x20. Untuk *map* 2 penulis menggunakan ukuran 30x30. Dan untuk *map* 3 penulis mengunakan ukuran 40x40. Pengujian ini dilakukan menggunakan 5 buah NPC.



Gambar 9. Hasil pengujian pengaruh ukuran peta permainan

Pada Gambar 9. merupakan hasil dari pengujian pengaruh ukuran peta permainan. Dari pengujian pengaruh ukuran peta permaninan mendapatkan beberapah hasil sebagai berikut. Pada pengujian *map* 1 (20x20) mendapatkan hasil rata-rata 66 FPS. Untuk pengujian *map* 2 (30x30) mendapatkan hasil rata-rata 61 FPS. Dan untuk pengujian *map* 3 mendapatkan hasil rata-rata 60 FPS.

4.2. Analisis Hasil Pengujian

Setelah dilakukan pengaruh pengujian jumlah NPC dan pengaruh ukuran peta permainan didapatkan hasil seperti pada sub bab sebelumnya.

Dalam pengujian pengaruh jumlah NPC dapat dianalisis bahwa jumlah NPC berpengaruh terhadap performa algoritme FSM berbasis *fragment shader*. Semakin banyak jumlah NPC maka semakin sedikit jumlah FPS yang didapatkan. Hal tersebut dikarenakan semakin banyak jumlah NPC semakin banyak juka proses pengambilan keputusan FSM yang dijalankan.

Sedangkan untuk pengujian pengaruh ukuran peta permainan menunjukkan bahwa semakin besar ukuran peta permainan semakin sedikit nilai FPS yang didapatkan. Hal tersebut dikarenakan semakin banyak rintangan yang dibuat sehingga semakin banyak pula proses yang dilakukan.

5. KESIMPULAN

Dalam penerapan algoritma FSM berbasis *fragment shader* untuk proses pengambilan keputusan pada *non player character* pada permainan *battle tank*, didapatkan kesimpulan sebagi berikut:

1. Penerapan algoritme FSM berbasis *fragment shader* dapat diproses dengan cara paralel. Yang mana proses pengambilan keputusan dilakukan pada GPU dan untuk proses yang lain dijalankan pada CPU. Hal tersebut terjadi karena proses pengambilan keputusan dijalankan pada *fragment shader* yang pada penelitian ini terdapat pada program *shader*.
2. Proses penyimpanan *state* pada algoritme FSM berbasis *fragment shader* disimpan pada *agenMap* yang terdapat pada program *fragment shader*.
3. Jumlah NPC berpengaruh terhadap performa algoritme FSM berbasis *fragment shader* karana proses FSM yang diakukan bertambah sesuai dengan jumlah NPC yang dibuat.
4. Ukuran peta permainan berpengaruh terhadap performa algoritme FSM berbasis *fragment shader* karena semakin besar ukuran peta permainan semakin banyak pula proses yang dilakukan.

6. DAFTAR PUSTAKA

Angel, E. & Shreiner, D., 2012. *Interactive Computer Graphics Top Down Approach with Shader-Based OpenGL*. Lodon: Pearson.

Millington, I. & Funge, J., 2009. *Artificial Intelligence for Games Second Edition*. Burlington: Morgan Kaufmann Publishers.

Rizaldi, I. M., 2017. Implementasi finite state machine pada game “Save the Animals”. *Seminar Informatika Aplikatif Polinema*, [online] Tersedia di: < http://jurnalti.polinema.ac.id/index.php/SIAP2016/article/view/160> [Diakses 30 Agustus 2018]

Rudomin, I., Millan, E. & Hernandez, B., 2005. Fragment shader for agent animation using finite state machine. *Simulation Modelling Practice and Theory*, [online] tersedia di: <https://www.researchgate.net/publication/222560844_Fragment_shaders_for_agent_animation_using_finite_state_machines> [Diakses 20 Agustus 2018]