Penerapan Finite-State Machines untuk Peningkatan Performa Frame Per Second dalam Game Multiplayer Real Time Strategy

Nicholas Sutikno, Djoni Haryadi Setiabudi, Alvin Nathaniel Tjondrowiguno
Program Studi Teknik Informatika, Fakultas Teknologi Industri, Universitas Kristen Petra
Jln. Siwalankerto 121 – 131 Surabaya 60236
Telp. (031)-2983455, Fax. (031)-8417658
nicoo7tdev@gmail.com, djonihs@petra.ac.id, alvin.nathaniel@petra.ac.id

ABSTRAK

Game banyak diminati mulai dari anak-anak sampai dewasa dan *game* juga dapat dimainkan dari beberapa *platform* yang tersedia seperti *PC*, *mobile*, dan *console*. *Game* memiliki berbagai macam *genre* seperti *Action*, *Adventure*, *RPG*, *Strategy*, dan masih banyak macam *genre* lainnya untuk *game*. Tujuan dalam skripsi ini adalah menerapkan *Finite-State Machines* untuk meningkatkan performa *FPS* pada *game* sehingga *game* menjadi ringan dan nyaman untuk dimainkan. Pada penelitian- penelitian sebelumnya *FSM* digunakan untuk pengujian algoritma *AI* dan untuk membantu dalam pembentukkan cerita mengenalkan sejarah dalam permainan. *Finite-State Machines* adalah sebuah metode atau rancangan yang akan dibuat dan diimplementasikan supaya *AI* dapat melakukan keputusan sendiri berdasarkan kondisi yang telah ditentukan. Menggunakan metode *Finite-State Machines* karena hanya satu tugas yang aktif dibaca sehingga tugas lain tidak dibaca oleh *AI* dalam program dan komputasi yang ringan.

Genre game yang akan diimplementasikan pada game adalah *Real Time Strategy* berjudul "*Attack on Toys*". Pada *Game* tersebut terjadi penurunan performa *FPS* karena banyaknya prajurit dalam permainan. Karena hal tersebut diimplementasikan *FSM* untuk meningkatkan performa *Game* tersebut dan *Game* harus tetap menyenangkan saat dimainkan oleh *player*. Pengujian performa *FPS* dengan membandingkan antar *game* menggunakan *Decision Tree*, menggunakan *FSM* desain pertama, dan menggunakan desain kedua. Pengujian kuesioner juga dilakukan untuk mengetahui apakah *game* menyenangkan dan apakah *AI* dapat bekerja dengan baik saat *game* dimainkan.

Hasil pengujian setelah implementasi *Finite-State Machines* bahwa dari hasil perbandingan pengujian dapat meningkatkan performa *FPS* hingga 90%. Berdasakan hasil kuesioner *Game* tetap menyenangkan untuk dimainkan dan *AI* tetap bekerja sesuai dengan harapan saat dimainkan oleh *player* setelah implementasi *Finite-State Machines* tanpa mengurangi kualitas dari *game* itu sendiri.

Kata Kunci: *Game, Unity, Finite-State Machines*

ABSTRACT

Games are in great demand from children to adults and games can also be played from several available platforms such as PC, mobile and console. Games have various genres such as Action, Adventure, RPG, Strategy, and many other types of genres for games. The purpose of this paper is to apply Finite-State Machines to improve FPS performance in games so that the game becomes light and comfortable to play. In previous studies FSM was used for testing AI algorithms and to assist in story formation introduce history in the game. Finite-State Machines are a method or design that will be created and implemented so that AI can make its own decisions based on predetermined conditions. Using the Finite-State Machines method because only one task is actively read so that other tasks are not read by AI in the program and light computing.

The game genre that will be implemented in the game is the Real Time Strategy entitled "Attack on Toys". In the game there was a decrease in FPS performance because of the many soldiers in the game. Because this was implemented by FSM to improve the performance of the Game and the Game must still be fun when played by the player. FPS performance testing by comparing games without using FSM, using the first FSM design, and using a second design. Questionnaire testing was also conducted to find out whether the game was fun and whether the AI could work well when the game was played.

Test results after the implementation of Finite-State Machines that from the results of comparison testing can increase FPS performance by up to 90%. Based on the results of the Game questionnaire it is still fun to play and AI continues to work as expected when played by players after the implementation of Finite-State Machines without reducing the quality of the game itself.

Keywords: *Game, Unity, Finite-State Machines*

1. PENDAHULUAN

Game saat ini menjadi industri yang sangat besar. *Game* termasuk dalam golongan *entertainment* industry sebagai media interaktif. *Game* sering digunakan untuk mengisi waktu luang, mendapatkan hiburan, atau kepuasan tertenu saat *player* memainkan sebuah *game* untuk mencapai target tertentu saat bermain. Industri *game* juga sekarang sudah berkembang sangat pesat, terutama pada masa ini dimana *game* banyak dimainkan, diminati, dan mudah didapatkan. *Game* banyak diminati oleh anak-anak sampai dewasa dan memiliki banyak *genre* seperti *Action Shooter*, *Action*, *Adventure*, *Role-Playing*, *Simulation*, sampai pada *game Strategy* [3].

Finite-State Machines adalah metode untuk menjalankan alur-alur *state* yang telah dibuat dan *Finite-State Machines* memiliki satu *state* atau lebih. Tetapi hanya satu *state* yang aktif dalam waktu yang sama. *Finite State Machines* membentuk pemikiran sebuah pasukan atau *AI* yang akan dibuat dengan kondisi dan aksi yang

akan dilakukan. Karena itu *Finite-State Machines* sangat cocok digunakan untuk *AI* dalam *game* [2].

Beberapa penelitian-penelitian sebelumnya menerapkan *Finite-State Machines* dengan tambahan penerapan metode/algoritma lainya lebih berfokus ke bagian *AI*, cerita, dan pengembangan sebuah *game*. Penelitian ini akan lebih fokus ke performa kerja sebuah *game* dan pengembangan *game* dalam *scale* yang cukup besar, karena performa kerja *game* saat dimainkan sangat penting untuk diperhatikan terutama saat publikasi *game* pada hari pertama rilis.

2. LANDASAN TEORI

2.1 Tinjauan Studi

Dari penelitian Zikrillah [6] telah berhasil diimplementasikan *Finite-State Machines* kedalam *Game* berjudul *Survive From Death*. *Survive From Death* merupakan *Game 3D First person shooter* dimana *player* harus bertahan hidup melawan beberapa musuh *NPC*, lalu akan lanjut ke rintangan berikutnya. *NPC* musuh menggunakan metode *Finite-State Machines* dan juga ditambah dengan algoritma *Pathfinding A** untuk menentukkan jalur terpendek yang dilalui oleh musuh.

Dari penelitian Arippa [1] telah berhasil diimplementasikan *Finite-State Machines* ke dalam *Game* yang telah dibuat. Membuat *virtual environment* yaitu replika dari suatu skenario yang pernah terjadi dalam kehidupan nyata. Mengenalkan sejarah Sumatera Utara melalui interaksi, *environment* pada *map*, dan *NPC* yang menggunakan *Finite-State Machines* pada *game 3D* berjudul *A War To Save Stabat*.

Maka pada penelitian ini *game* akan memiliki animasi pada semua prajurit dan meningkatkan *AI* dengan menambahkan jumlah state yang diperlukan pada *behavior AI* agar tampak lebih hidup, *map* yang lebih banyak, optimisasi pemodelan *3D*, dan berbagai macam jenis *NPC*.

2.2 *Finite-State Machines*

Finite-State Machines adalah sebuah metode atau rancangan yang akan dibuat dan diimplementasikan dalam konteks ini adalah pada *AI* untuk *game* yang dijalankan supaya *AI* atau musuh dalam sebuah *game* dapat menentukan sebuah kondisi, apa yang akan dilakukan setelah memenuhi kondisi tersebut sehingga *AI* memiliki pemikiran, dan membuat keputusan sendiri apa yang akan dilakukan berikutnya. Dalam *Finite-State Machines* hanya ada satu *state* sajah yang aktif dalam waktu yang sama dimana dalam suatu *state AI* akan melakukan fungsi tertentu. Untuk berpindah dari satu *state* ke *state* yang lain menggunakan *action*. *State* adalah suatu keadaan dimana *AI* melakukan suatu fungsi terus menerus sesuai dengan fungsi *state* yang sedang aktif. *Action* adalah sebuah aksi yang memiliki kriteria atau kondisi dimana jika kriteria tersebut terpenuhi maka *state* akan bertransisi ke *state* lain melalui *action* tersebut. *Finite-State Machines* sangat cocok untuk *AI* dalam *game*, memberikan hasil yang baik dan pembuatan koding yang sederhana [2].

2.3 *Unity*

Unity adalah sebuah *Game Engine* untuk mengembangkan sebuah *game* dengan cepat dan efisien dengan fitur-fitur yang sudah tersedia di *Unity* [5]. *Game* yang dibuat di *Unity* dapat dimainkan dalam beberapa macam platform yang disediakan oleh *Unity* seperti *PC*, *Consoles*, dan bahkan *Mobile*. *Unity* dapat membuat *game* secara *3D* maupun *2D*. *Unity* memiliki versi gratis, maupun berbayar *Unity* yang akan digunakan adalah versi gratis *Unity3D*

2.4 *Game*

Game merupakan kata dalam bahasa inggris yang berarti permainan. Permainan adalah sesuatu yang dapat dimainkan dengan aturan tertentu sehingga ada yang menang dan ada yang kalah. *Game* biasanya bersifat tidak serius atau dengan tujuan refreshing untuk memuaskan dan menyenangkan pemain yang sedang memainkan *game* tersebut [6].

3. DESAIN SISTEM

3.1 Desain *Game*

Game yang akan didesain diberi judul "*Attack on Toys*". Dimana dalam *Game* ada sejumlah prajurit mainan dan melawan prajurit mainan dari *nation* lain. *Nation* adalah sebuah bangsa atau bagian grup untuk setiap prajurit mainan. Tujuan dari *Game* ini adalah bertahan dari serangan musuh melindungi *HQ*. *HQ* adalah rumah induk sebuah *Nation* yang harus dilindungi. Jika *HQ* mati maka *player* kalah dalam permainan. Prajurit-prajurit musuh akan keluar dari portal dan menyerang *HQ*.

3.2 *Gameplay*

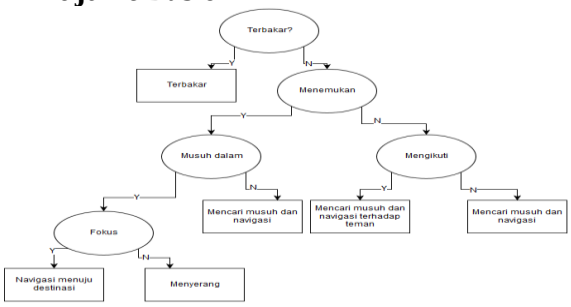
Saat memulai *game* ini *player* akan *spawn* di daerah *HQ* beserta beberapa prajurit-prajurit yang sudah siap dalam bertempur. *Player* mengatur posisi prajurit secara strategis untuk siap menghadapi musuh yang akan datang. *Player* juga dapat meningkatkan pertahanan dari serangan musuh dengan cara membangun prajurit *tower* pada daerah *tower spawn point* yang ada dalam *map*. *Player* juga dapat mengeluarkan beberapa jenis prajurit sesuai yang dipilih oleh *player*. *Player* juga dapat memanggil bantuan berupa prajurit atau senjata dari jarak jauh menggunakan radio. Setelah *player* sudah siap maka prajurit-prajurit musuh akan datang dari portal dan menyerang *HQ* dengan jumlah prajurit tertentu. Jika *HQ* masih hidup maka *game* akan berlanjut ke rintangan berikutnya dengan prajurit musuh yang lebih banyak dan lebih susah.

3.3 Jenis Prajurit

Game akan memiliki berbagai macam-macam jenis prajurit dengan tugasnya masing-masing untuk melawan musuh. Karena setiap jenis prajurit memiliki tugas yang berbeda-beda untuk melawan musuh maka dibutuhkan *Finite-State Machines* untuk setiap jenis prajurit untuk melaksanakan tugasnya.

Pada artikel ini hanya akan dipakai satu jenis prajurit yaitu jenis prajurit *basic* sebagai contoh untuk jurnal.

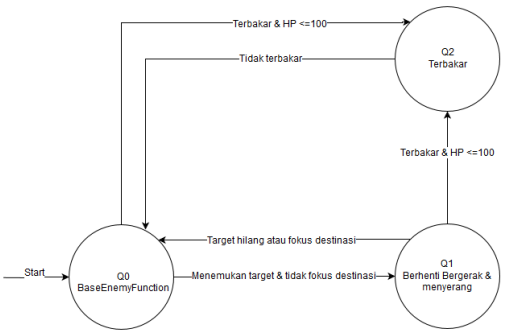
3.4 Prajurit *Basic*



Gambar 1. Desain *Decision Tree* untuk prajurit *basic*.

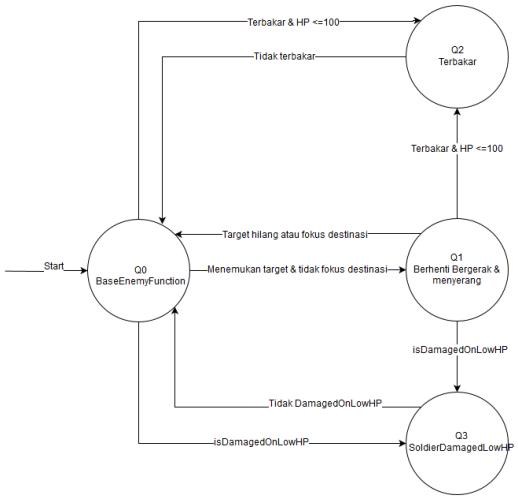
Pada Gambar 1 merupakan desain *Decision Tree* untuk jenis prajurit *basic*. *AI* memiliki *behavior* yang hampir sama dengan

desain pertama *Finite-State Machines* untuk prajurit *basic*



Gambar 2. Desain pertama *Finite-State Machines* untuk prajurit *basic*.

Pada Gambar 2 merupakan desain pertama *Finite-State Machines* untuk jenis prajurit *basic.* Prajurit *basic* dimulai dengan *state* Q0 yaitu prajurit melakukan perhitungan navigasi gerakan dan mencari target dalam *radius* yang telah ditentukan. Ketika menemukan target maka akan masuk ke *state* Q1 yaitu berhenti bergerak dan menyerang target yang telah ditemukan. Jika target telah hilang atau prajurit focus ke destinasi maka akan kembali ke *state* Q0. Prajurit bisa terbakar terkena api maka akan masuk ke *state* Q2 dimana prajurit lari-lari sampai tidak terbakar lagi.



Gambar 3. Desain kedua *Finite-State Machines* untuk prajurit *basic*

Pada Gambar 3 merupakan desain *FSM* kedua untuk prajurit *basic* adanya penambahan satu *state* dimana dapat memperlambat serangan senjata dan dapat menghentikan pergerakkan prajurit dapat meningkatkan sedikit performa rata-rata pada *FPS.*

Gambar 4. Inspirasi untuk prajurit *basic* [4].

Pada Gambar 4 merupakan inspirasi terbentuknya jenis prajurit *basic* yang terdapat dalam permainan.

4. PENGUJIAN SISTEM

Pengujian dilakukan dengan jumlah prajurit yang sama pada jenis skenario yang sama dan pengujian akan dilakukan dalam tiga macam yaitu sebagai berikut:

- Pengujian *Average Frame Per Second* menggunakan *Decision Tree* dalam dua skenario yang berbeda.
- Pengujian *Average Frame Per Second* dengan *FSM* desain pertama dalam dua skenario yang berbeda.
- Pengujian *Average Frame Per Second* dengan *FSM* desain kedua dalam dua skenario yang berbeda.

4.1 Metode Pengujian Performa *FPS*

Mendapatkan performa *Frame Per Second* saat *game* berjalan akan di hitung secara rata-rata dari awal mulai perang sampai selesai perang dan juga rata-rata *Frame Per Second* terendah untuk mengetahui kondisi rata-rata *Frame Per Second* terburuk saat *game* berjalan. Berikut ini adalah spesifikasi PC yang digunakan untuk menguji performa *Frame Per Second* dalam *game* adalah sebagai berikut:

- *Operating System: Windows 7 Ultimate 64-bit*
- *Processor: Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz (8 CPU), ~3.4GHz*
- *Memory: 8192MB RAM.*
- *GPU: Nvidia GeForce GTX 960*

4.2 Hasil Pengujian Performa *FPS*

Dapat dilihat bahwa dari Tabel 1, Tabel 2, dan Tabel 3 bahwa performa *FPS* tetap meningkat dari hasil pengujian *game* menggunakan *Decision Tree* sampai menggunakan *FSM* desain kedua. Saat pengujian simulasi perang terkadang dapat memberikan hasil yang yang berbeda walaupun jumlah dan jenis prajurit yang sama. Sehingga dapat memberikan nilai *Average Frame Per Second* yang berbeda dalam setiap pengujian. Karena itu dibutuhkan data berupa *Lowest Average Frame Per Second* yang berguna sebagai nilai pengukuran kedua yang memberikan nilai yang lebih pasti dan untuk mengetahui rata-rata performa *FPS* terendah yang pernah terjadi saat melakukan simulasi perang. Pada Tabel 1 berisi data statistik dari hasil pengujian *Average Frame Per Second*. Pada Tabel 2 berisi data statistik dari hasil pengujian *Lowest Average Frame Per Second*. Pada Tabel 3 berisikan peningkatan *FPS* dari semua hasil pengujian.

Tabel 1. Hasil semua data statistic *Average Frame Per Second* pengujian dikelompokkan dalam tabel

Jenis Pengujian		***Average Frame Per Second***			
		MAX	**MIN**	**AVG**	**STDEV**
Decision Tree	Skenario Pertama	161	10	95	45
	Skenario Kedua	167	24	90	38
FSM Desain Pertama	Skenario Pertama	190	19	108	51
	Skenario Kedua	224	26	120	44
FSM Desain Kedua	Skenario Pertama	245	19	122	65
	Skenario Kedua	323	30	144	66

Tabel 2. Hasil semua data statistik *Lowest Average Frame Per Second* pengujian dikelompokkan dalam tabel

Jenis Pengujian		*Lowest Average Frame Per Second*			
		MAX	MIN	AVG	STDEV
Decision Tree	Skenario Pertama	40	10	17	9
	Skenario Kedua	53	10	16	9
FSM Desain Pertama	Skenario Pertama	67	12	28	10
	Skenario Kedua	51	12	27	9
FSM Desain Kedua	Skenario Pertama	65	12	28	10
	Skenario Kedua	56	10	28	9

Tabel 3. Hasil semua peningkatan *FPS* dikelompokkan dalam tabel

Hasil Perbandingan Antar Pengujian		*Average Frame Per Second*	*Lowest Average Frame Per Second*
		Rata-Rata Peningkatan Dalam %	Rata-Rata Peningkatan Dalam %
Decision Tree Dengan *FSM* Desain Pertama	Skenario Pertama	23%	78%
	Skenario Kedua	46%	90%
FSM Desain Pertama Dengan *FSM* Desain Kedua	Skenario Pertama	13%	1%
	Skenario Kedua	19%	6%

4.3 Pengujian Fitur Pada *Game*

Pengujian fitur adalah dengan mencoba fitur-fitur yang terdapat dalam *game*. Pengujian fitur dari menu *game* sampai player bisa memainkan *game.*

Gambar 5. Menu *more missions*

Pada Gambar 5 terdapat dua tombol untuk navigasi pemilihan bagian menu yaitu tombol ke kiri dan tombol ke kanan. Tombol bertuliskan "*Tan Army Invasion*" berfungsi untuk *quick play* agar *player* langsung dengan cepat memainkan *game*. Tombol *more missions* di atas tombol berfungsi untuk menampilkan semua misi yang dapat dipilih oleh *player*. Terdapat juga tombol *credits* dan tombol *close* untuk menutup aplikasi.

Gambar 6. Menu *Multiplayer 'LAN'*

Pada Gambar 6 player akan mengisikan nickname. Selanjutnya memilih misi yang akan dimainkan berupa *dropdown* dan *create room* setelah memilih misi yang mau dimainkan. Jika *player* mau bergabung dengan *player* lain maka memilih tombol *refresh* dan *join room* dengan *player* lain yang sedang bermain.

Gambar 7. *User interface player* saat bermain

Pada Gambar 7 adalah *user interface* yang dimiliki oleh *player*. Memberitahu kepada player waktu yang masih tersedia untuk membuat pertahanan dan jumlah musuh yang akan datang pada *wave* / rintangan saat ini. Menunjukkan jumlah *HP* yang dimiliki *HQ*, *player*, dan juga jumlah koin yang dimiliki *player*. *Player* dapat menggunakan *build mode* untuk membangun atau membuat prajurit dan juga *map mode* untuk melihat situasi *map* secara keseluruhan.

4.4 Hasil Kuesioner

Kuesioner dibagikan kepada teman mahasiswa Universitas Kristen Petra dan melalui media sosial. Mendapatkan total sebanyak 19 kuesioner, setiap kuesioner terbagi menjadi tiga bagian yaitu *game* dengan *Decision Tree*, *game* dengan *FSM* desain pertama, dan *game* dengan *FSM* desain kedua

Tabel 4. Hasil semua jenis pengujian kuesioner dikelompokkan dalam tabel

Jenis Kuesioner	*Game* Menyenangkan	*AI* Bekerja Dengan Baik	Wave Menang
Decision Tree	3,42	3,26	2,42
FSM Desain Pertama	3,68	3,63	2,84
FSM Desain Kedua	3,78	3,78	2,89

Pada Tabel 4 kesimuplan yang didapat setelah melakukan kuesioner untuk pengujian *game* bahwa *game* cukup menyenangkan dan *AI* juga dapat cukup bekerja dengan baik. Terjadi peningkatan nilai hasil pengujian kuesioner dari *game* menggunakan *Decision Tree* sampai *game* menggunakan *FSM* desain kedua.

5. KESIMPULAN DAN SARAN

Implementasi *Finite-State Machines* untuk *game* berjudul "*Attack on Toys*" dapat meningkatkan performa *Frame Per Second* saat *game* dimainkan.

Tugas yang terdapat dalam *state* pada desain *Finite-State Machines* terhadap *AI* dapat mempengaruhi performa *FPS*. Semakin rumit tugas dalam *state* maka performa *FPS* juga akan ikut menurun. Walaupun desain *Finite-State Machines* semakin rumit tetapi jika terdapat tugas *state* yang dapat meringankan *AI Finite-State Machines* dapat meringankan performa *FPS*.

Berdasarkan hasil kuesioner dengan implementasi *Finite-State Machines game* berhasil untuk membuat pemain tetap senang memainkan *game* dan *AI* tetap bekerja dengan baik

Berdasarkan hasil penelitian untuk *Game* "*Attack on Toys*", saran yang dapat digunakan untuk pengembangan game lebih lanjut adalah sebagai berikut:

- Membuat misi berupa *campaign* atau *story* untuk *game* sehingga dapat mengikat pemain dengan permainan.
- Meningkatkan performa *FPS* pada *game* jauh lebih baik lagi dengan desain *FSM* yang lebih baik maupun dengan metode yang berbeda.
- Membuat fitur *multiplayer* yang lebih dalam seperti pembuatan *lobby*, *text chat*, dan interaksi dengan sesama *player*

6. DAFTAR REFERENSI

[1] Arippa, A. 2016. *Behaviour NPC Game Historical War 3D Using Finite State Machine*. Tersedia di: http://repository.usu.ac.id/handle/123456789/62120?show=full

[2] Bevilacqua, F. 2013. *Finite-State Machines: Theory and Implementation*. Tersedia di: http://gamedevelopment.tutsplus.com/tutorials/finite-statemachines-theory-and-implementation--gamedev-11867

[3] Fauzi, M.H., Rodiah. 2013. *First-Person Shooter 3D "GamaShoot"* dengan *Blender* dan *Unity 3D*. Tersedia di: https://www.researchgate.net/publication/301348645_First-PErson_Shooter_3D_GamaShoot_dengan_Blender_dan_Unity_3D

[4] Firehawk894. 2018. *Army Men RTS: Unit guide*. Tersedia di: https://steamcommunity.com/sharedfiles/filedetails/?id=1454172075

[5] Unity. 2018a. *Game engines—how do they work?*. Tersedia di: https://unity3d.com/what-is-a-game-engine

[6] Zikrillah, F. 2013. Aplikasi game *3D First Person Shooter (FPS) Survive From Death*. Tersedia di: http://elib.unikom.ac.id/gdl.php?mod=browse&op=read&id=jbptunikompp-gdl-fajarzikri-31059&q=survive%20from%20death