Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer e-ISSN: 2548-964X
Vol. 1, No. 12, Desember 2017, hlm. 1616-1621 http://j-ptiik.ub.ac.id

Penerapan Algoritma *Hybrid Pathfinding* A* dan Boids untuk *Game* Pesawat Tempur

Firdaus Rahmat Prasetyo[1], Eriq Muh. Adams Jonemaro[2], Muhammad Aminul Akbar[3]

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

Abstrak

Algoritma *hybrid pathfinding* A* dan Boids dapat digunakan sebagai pencari jalur terbang agen pada *game* pesawat tempur. Dalam *hybrid pathfinding*, algoritma A* digunakan sebagai pencari jalur terpendek pesawat dan menghindari halangan statis sedangkan algoritma Boids digunakan sebagai pencari arah terbang pesawat secara berkelompok dan menghindari halangan dinamis. *Grid* yang digunakan untuk pencarian jalur menggunakan algoritma A*digerakan sesuai dengan posisi agen A* sehingga tidak meperbanyak jumlah *node* karena pada *game* pesawat tempur ukuran peta sangat besar. Pencarian jalur dengan menggunakan algoritma A* hanya akan dilakukan saat *grid* menyentuh halangan statis agar pencarian jalur tidak terlalu lama. Hal ini terbukti pada saat agen menggunakan algoritma A* rata-rata kenaikan waktu eksekusi hanya sebesar 4.24% dan rata-rata penurunan FPS sebesar 11.59% dibanding saat tidak menggunakan algoritma A*. Sementara penggunaan algoritma Boids dilakukan secara terus menerus pada pesawat agar pesawat tidak saling bertabrakan. Algoritma ini dipilih karena memiliki kompleksitas waktu yang tidak terlalu tinggi. Hal ini juga terbukti saat terdapat halangan dinamis dalam peta rata-rata kenaikan waktu eksekusi hanya sebesar 6.21%. Rata-rata FPS pada semua peta sebesar 75.91 yang mana diatas batas FPS yang dapat mengelabui mata sebesar 20. Rata-rata waktu eksekusi dari semua peta adalah 102.79 ms. Rata-rata panjang jalur dari semua peta adalah 445.56. Rata-rata jumlah aktor yang hancur dari semua peta adalah 107.

Kata Kunci: *Hybrid Pathfinding*, Algoritma A*, Algoritma *Boids*

Abstract

The A and Boids hybrid pathfinding algorithms can be used as a way of tracking the flying paths of aircraft games. In hybrid pathfinding, A* algorithm is used as the shortest path finding and avoid static obstacles while the Boids algorithm is used as a flyweeker in groups and avoids dynamic obstacles .Grid used for path search using A* algorithm is driven according to A* agent position that it does not increase number of nodes because in aircraft game the map size is so large. Pathfinding using A* algorithm will only be call when grid touch static obstacle, so pathfinding doesn't take a long time. This is proven when the agent uses an A* algorithm, the average of execution time increase by 4.24% and the average of FPS decreases by 11.59% compared with not using the A * algorithm. While the use of Boids algorithm is continuously on the aircraft so that the aircraft does not collide with each other. This algorithm is chosen because it has a time complexity that is not too high. This is also proven when there is dynamic obstacles in map, the average of execution time increase is only 6.21%.The average FPS on all maps is 75.91 which is above of the FPS limit that can trick the eye by 20. The average execution time of all maps is 102.79 ms. The average path length of all maps is 445.56. The average number of actors destroyed from all maps is 107.*

Keywords: *Hybrid Pathfinding, A* Algorithm, Boids Algorithm*

1. PENDAHULUAN

Game pesawat tempur merupakan salah satu *game* yang menarik bagi pemain. Pemain akan mengontrol pesawat untuk saling serang dengan menggunakan senjata di pesawat. Tim yang mampu mengalahkan semua pesawat lawan akan memenangkan *game* (Gaijin Entertaintment, 2016). Dalam *game* pesawat tempur, terdapat beberapa pengambilan

keputusan dalam *game* yaitu *macromanagement* dan *micromanagement*. *Macromanagement* adalah strategi pemain dalam mengalahkan lawan. Sedangkan *micromanagement* merupakan pergerakan pemain saat bertempur yaitu menghindar, mengejar, dan menyesuaikan pergerakan dengan pesawat secara berkelompok agar pesawat tidak mudah untuk diserang musuh (Zongxin et al, 2016). Dalam penelitian ini hanya akan berfokus pada *micromanagement* saja yaitu membuat pesawat terbang secara berkelompok dan dapat terbang sampai tujuan dengan jalur terpendek. Dalam mencari jalur tercepat algoritma yang dinilai paling tepat adalah A*, karena algoritma ini mampu menemukan jalur terbaik dari posisi awal menuju posisi akhir. Akan tetapi, ada kemungkinan akan ada pesawat lain yang berada di jalur lintasan yang sudah dicari dengan menggunakan A* pada saat pesawat belum sampai ke tujuan. Hal ini akan menyebabkan tabrakan antar pesawat. Sedangkan jika dilakukan perhitungan ulang pada algoritma A* dengan mempertimbangkan halangan bergerak maka komputasi pada CPU akan lebih berat menurut Harsono (2015). Penghindaran antar pesawat bisa saja dilakukan dengan menggunakan *collision avoidance behavior* hanya saja pada algoritma ini tidak memiliki pengaturan pesawat untuk terbang secara berkelompok dan juga algoritma ini memiliki kompleksitas waktu yang tinggi (Milington dan Funge, 2016).

Algoritma Boids mampu mengatur pesawat agar setiap pesawat dapat terbang secara berkelompok. Akan tetapi, algoritma Boids tidak dapat menemukan jalur akurat pada area yang kompleks (Hagelback, 2016). *Hybrid pathfinding* merupakan penggabungan 2 algoritma dalam menemukan jalur. Terdapat bermacam-macam *hybrid pathfinding* yang ada contohnya *hybrid pathfinding* A* dan Boids, *hybrid pathfinding* A* dan *Potential Field*, *hybrid pathfinding* A* dan Neural Network. Namun menurut Hagelback (2016), jika dibandingkan dengan *hybrid pathfinding* A* dan *Potential Field*, *hybrid pathfinding* A* dan Boids dalam hal waktu eksekusi jauh lebih unggul 7.569 ms .Sedangkan menurut Schnell (2016), *hybrid pathfinding* A* dan Neural Network sangat tidak disarankan untuk *game* karena memiliki performa yang buruk dengan waktu eksekusi sebesar 14761 ms.

Berdasarkan penjelasan sebelumnya, penelitian ini akan melakukan *hybrid pathfinding* alogritma A* dan Boids agar agen dapat menemukan jalur terbaik yang akurat dengan pesawat yang terbang secara berkelompok.

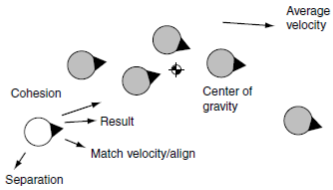
2. *GAME* PESAWAT TEMPUR

Game pesawat tempur merupakan salah satu *game* yang menarik bagi pemain. *Game* ini dimulai dengan pemain dapat memilih pesawat yang ingin dimainkan lalu masuk ke tahap pertarungan baik secara *singleplayer* maupun multiplayer. Dalam *game singleplayer* pemain akan melawan computer sedangkan dalam *game multiplayer* pemain akan melawan satu sama lain. Pemain akan mengontrol pesawat untuk saling serang dengan menggunakan senjata di pesawat. Tim yang mampu mengalahkan semua pesawat lawan akan memenangkan *game* (Gaijin Entertaintment,2016).

Dalam *game* pesawat tempur, rotasi pesawat dari berbagai sumbu memiliki kekuatan yang berbeda. Pesawat dalam dunia nyata memiliki 3 jenis rotasi yaitu *pitch*, *yaw* dan *roll* (Milington dan Funge, 2016). Namun, dalam penelitian ini tidak akan menggunakan rotasi tersebut dan lebih berfokus pada pergerakan pesawat saja.

3. ALGORITMA BOIDS

Algoritma Boids mampu meniru pergerakan hewan seperti ikan dan burung yang bergerak secara berkelompok atau yang disebut juga dengan istilah *flocking*. *Flocking* merupakan salah satu perilaku di dalam steering behaviour yang mensimulasikan pergerakan agen secara berkelompok. *Flocking* biasanya menggunakan algoritma Boids karena Boids hanya melakukan perhitungan sederhana. Di dalam *flocking* terdapat beberapa perilaku yaitu: *separation* agar agen yang berdekatan saling menjauh,*cohesion* yang mana agen akan berusaha untuk mendekati titik tengah grup, *alignment* yaitu agen akan menyesuaikan arah dan kecepatan agar tidak terpisah. Lalu untuk penentuan tujuan digunakan perilaku *arrive*. Penjelasan diatas dapat digambarkan seperti Gambar 1.



Gambar 1. *Flocking*

Kompleksitas waktu dari algoritma Boids adalah *O(n)* yang mana n adalah total banyaknya *behaviour* pada dalam satu grup. Prioritas dari ketiga perilaku tersebut akan lebih baik jika *separation* lebih tinggi dari *cohesion* dan *cohesion* lebih tinggi dari *alignment* (Milington dan Funge, 2016).

4. ALGORITMA A*

Algoritma A* dinilai sangat mudah untuk diterapkan, efisien dan memiliki banyak optimasi. Algoritma A* merupakan algoritma yang berfungsi sebagai pencari jalan terpendek dari posisi awal ke posisi tujuan. Dalam pomrosesan setiap *node* dilakukan secara berulang dengan menelusuri *node* ke *node*. Kompleksitas waktu dari algoritma ini adalah *O(lm)*, dengan *l* adalah jumlah *node* pada *open list* dan *m* adalah rata-rata jumlah tetangga dari setiap *node*. (Milington dan Funge, 2016).

4.1. Langkah Algoritma

.Dalam setiap *node* terdapat nilai berupa *cost* yang dibutuhkan untuk menempuh dari *node* sebelumnya yang disebut juga dengan *cost-so-far* atau *gcost*/*g(n)*.Di dalam setiap *node* juga terdapat nilai jarak antara *node current node* dengan *node* tujuan yang disebut juga dengan *heuristic* atau *hcost*/*h(n)*.Hasil penjumlahan dari kedua nilai tersebut akan dijadikan sebagai dasar *pathfinding* atau fcost/*f(n)*.Penghitungan fcost ditunjukkan pada Persamaaan 1.

$$f(n) = g(n) + h(n) \qquad (1)$$

4.2. Perhitungan jarak

Penghintugan jarak antar *node* dapat dilakukan dengan menggunakan berbagai cara contohnya Heuristik ataupun Euclidean. Penghitungan jarak dengan menggunakan metode heuristik dapat menyebabkan penghitungan jarak yang kurang akurat sehingga jalur yang ditemukan bukan jalur yang terbaik. Penghitungan jarak dengan menggunakan metode Euclidean dinilai lebih akurat dibanding Manhattan sehingga dalam pencarian jalur akan lebih akurat. Penghitungan jarak dengan menggunakan cara Euclidean ditunjukkan pada Persamaan 2.

$$D = \sqrt{x^2 + y^2} \qquad (2)$$

5. *HYBRID PATHFINDING*

Hybrid pathfinding adalah pencarian jalur dengan mengkombinasikan dua algoritma. *Hybrid pathfinding* dinilai lebih efektif dalam mencari jalur karena, saling mengkombinasikan kelebihan dan kekurangan masing-masing algoritma. Biasanya *hybrid pathfinding* digunakan untuk *crowd simulation* yaitu agen yang dipakai dalam jumlah yang besar sehingga membutuhkan cara agar antara agen tidak bertabrakan sekaligus dapat sampai ke tujuan dengan melewati jalur yang ditemukan. Pada algoritma ini terdapat beberapa penelitian yang telah dilakukan diantaranya adalah *hybrid pathfinding* A* dan Boids, *hybrid pathfinding* A* dan *Potential Field*, *hybrid pathfinding* A* dan Neural Network. Dalam penelitiannya, Hagelback (2016) membandingkan 2 *hybrid pathfinding* A* dan Boids dengan *hybrid pathfinding* A* dan *Potential Field* pada *game* Starcraft. Data yang didapatkan dari penelitian tersebut ditunjukkan pada Tabel 1.

Tabel 1. Perbandingan *Potential Field* dan Boids

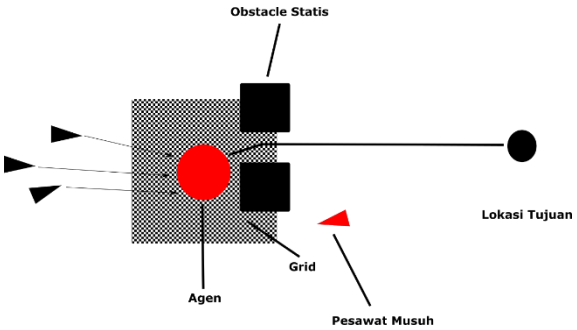
Jenis	Rata-Rata Waktu Eksekusi (ms)	Rata-Rata Maks. Waktu Eksekusi (ms)
A*-PF	7.632	20.337
A*-Boids	0.063	0.625

Data pada Tabel 1 menunjukkan waktu eksekusi *hybrid pathfinding* A* dan Boids lebih cepat 7.569 ms daripada *hybrid pathfinding* A* dan *Potential Field*. Sedangkan menurut Schnell (2016), *hybrid pathfinding* A* dan Neural Network sangat tidak disarankan untuk *game* karena memiliki performa yang buruk dengan waktu eksekusi sebesar 14761 ms.

6. PENERAPAN *HYBRID PATHFINDING*

Pada penerapan *hybrid pathfinding grid* akan digerakkan sesuai dengan posisi agen. Ketika *grid* tersebut tidak mendeteksi adanya penghalang statis maka agen akan berjalan lurus ke lokasi tujuan. Sedangkan saat agen A* mendeteksi adanya halangan statis maka akan

dilakukan pencarian jalur menggunakan A*.Agen yang menggunakan algoritma A* memiliki *grid* sebesar 10x10x10 sebagai pendeteksi halangan sekaligus sebagai *node* dalam mencari jalur terdekat ke target. Agen algoritma Boids menjadikan agen A* sebagai tujuan dan pusat untuk *flocking*. Target dari pencarian jalur bukan merupakan posisi sebenarnya dari target melainkan posisi terdekat dari *grid* ke target. Pada Gambar 2 adalah metode penerapan *hybrid pathfinding* yang telah dijelaskan diatas.



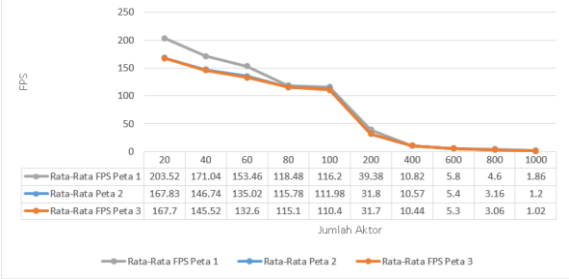
Gambar 2. Penerapan Algoritma A*

7. PENGUJIAN

Pada setiap peta diuji dengan parameter yang sama yaitu FPS, waktu eksekusi, panjang jalur dan jumlah aktor. Percobaan dilakukan 10 kali pada setiap peta. Data FPS,waktu eksekusi dan panjang jalur yang didapatkan adalah rata-rata dari semua agen dalam *flock*. Peta 1 tidak terdapat, peta 2 hanya ada halangan statis dan peta 3 terdapat halangan statis dan dinamis.

7.1. Pengujian FPS

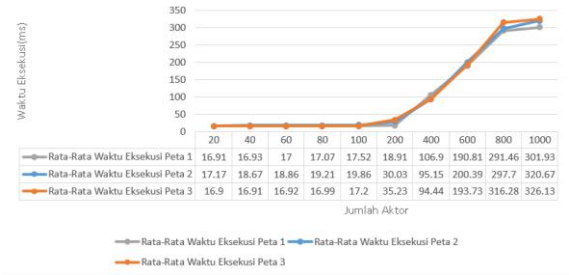
Pengujian FPS dilakukan pada semua peta dengan interval 20 aktor pada percobaan 1 sampai percobaan 5 dan interval 200 aktor pada percobaan 6 sampai percobaan 10.Dapat dilihat bahwa grafik semakin menurun. Rata-rata penurunan FPS pada peta 1 dengan pengambilan sampel uji coba ke-1 sampai ke-5 adalah 12.82%. Rata-rata penurunan FPS pada peta 2 dengan pengambilan sampel uji coba ke-1 sampai ke-5 adalah 9.64%. Rata-rata penurunan FPS pada peta 3 dengan pengambilan sampel uji coba ke-1 sampai ke-5 adalah 9.74% yang dapat dilihat pada Gambar 3.



Gambar 3. Perbandingan FPS Setiap Peta

7.2. Pengujian Waktu Eksekusi

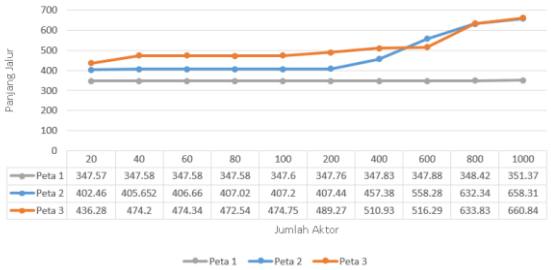
Pengujian waktu eksekusi dilakukan pada semua peta dengan interval 20 aktor pada percobaan 1 sampai percobaan 5 dan interval 200 aktor pada percobaan 6 sampai percobaan 10.Dapat dilihat bahwa grafik semakin meningkat. Rata-rata kenaikan waktu eksekusi pada peta 1 dengan pengambilan sampel uji coba ke-1 sampai ke-5 adalah 0.89%. Rata-rata peningkatan waktu eksekusi pada peta 2 dengan pengambilan sampel uji coba ke-1 sampai ke-5 adalah 3.74%. Rata-rata peningkatan waktu eksekusi pada peta 3 dengan pengambilan sampel uji coba ke-1 sampai ke-5 adalah 0.69% yang dapat dilihat pada Gambar 4.



Gambar 4. Perbandingan Waktu Eksekusi

7.3. Pengujian Panjang Jalur

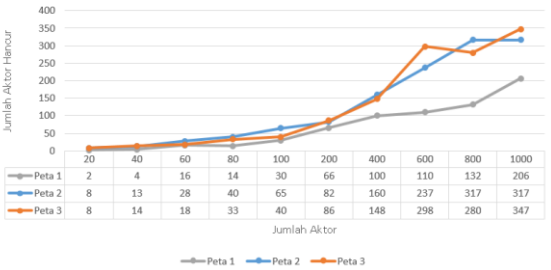
Pengujian panjang jalur dilakukan pada semua peta dengan interval 20 aktor pada percobaan 1 sampai percobaan 5 dan interval 200 aktor pada percobaan 6 sampai percobaan 10.Dapat dilihat bahwa grafik semakin meningkat. Rata-rata kenaikan panjang jalur pada peta 1 dengan pengambilan sampel uji coba ke-1 sampai ke-5 adalah 0.001%. Rata-rata kenaikan panjang jalur pada peta 2 dengan pengambilan sampel uji coba ke-1 sampai ke-5 adalah 0.24%. Rata-rata kenaikan panjang jalur pada peta 3 dengan pengambilan sampel uji coba ke-1 sampai ke-5 adalah 2.31% yang dapat dilihat pada Gambar 5.



Gambar 5. Perbandingan Panjang Jalur.

7.4. Pengujian Jumlah Aktor Hancur

Pengujian panjang jalur dilakukan pada semua peta dengan interval 20 aktor pada percobaan 1 sampai percobaan 5 dan interval 200 aktor pada percobaan 6 sampai percobaan 10. Dapat dilihat bahwa grafik semakin meningkat. Rata-rata kenaikan panjang jalur pada peta 1 dengan pengambilan sampel uji coba ke-1 sampai ke-5 adalah 154.7%. Rata-rata kenaikan panjang jalur pada peta 2 dengan pengambilan sampel uji coba ke-1 sampai ke-5 adalah 70.77%. Rata-rata kenaikan panjang jalur pada peta 3 dengan pengambilan sampel uji coba ke-1 sampai ke-5 adalah 52.02% yang dapat dilihat pada Gambar 6.



Gambar 6. Perbandingan Jumlah Aktor Hancur

Data peningkatan dan penurunan setiap 20 aktor yang telah dijelaskan diatas, akan disajikan dalam bentuk Tabel 2.

Tabel 2. Rata-Rata Peningkatan dan Penurunan Parameter Pengujian(%).

Peta	Rata-Rata Penurunan FPS	Rata-Rata Peningkatan Waktu Eksekusi	Rata-Rata Peningkatan Waktu Eksekusi	Rata-Rata Peningkatan Jumlah Aktor Hancur
1	12.82	0.89	0.001	154.7
2	9.64	3.74	0.28	70.77
3	9.74	0.69	2.31	52.02
Rata-Rata	10.73	1.77	0.86	92.49

Data keseluruhan yang didapatkan dari pengujian yang telah dijelaskan diatas akan disajikan dalam bentuk Tabel 3.

Tabel 3. Rata-Rata Hasil Pengujian.

Peta	Rata-Rata Penurunan FPS	Rata-Rata Peningkatan Waktu Eksekusi (ms)	Rata-Rata Peningkatan Waktu Eksekusi	Rata-Rata Peningkatan Jumlah Aktor Hancur
1	82.51	99.54	348.11	68
2	72.94	103.77	474.27	126
3	72.28	103.073	514.32	127
Rata-Rata	75.91	102.79	445.56	107

8. KESIMPULAN

Data Berdasarkan hasil pengujian dan analisis yang telah dilakukan terhadap penerapan *hybrid pathfinding* A* dan Boids pada *game* pesawat tempur didapatkan kesimpulan sebagai berikut:

1. Pada penerapan *hybrid pathfinding* algoritma A* digunakan untuk menghindari halangan statis dan mencari jalur terpendek, sedangkan algoritma Boids digunakan untuk menghindari halangan dinamis dan arah terbang pesawat secara berkelompok.
2. Semakin banyak halangan pada peta akan menurunkan FPS. Rata-rata penurunan FPS saat terdapat halangan statis pada peta sebesar 11.59 % sedangkan saat terdapat halangan statis dan dinamis pada peta sebesar 12.39 %.
3. Semakin banyak halangan pada peta akan meningkatkan waktu eksekusi. Rata-rata peningkatan waktu eksekusi saat terdapat halangan statis pada peta sebesar 4.24 % sedangkan saat terdapat halangan statis dan dinamis pada peta sebesar 6.21 %.
4. Semakin banyak aktor maka akan menurunkan FPS. Rata-rata penurunan FPS setiap 20 aktor dari semua peta sebesar 10.73 %.
5. Semakin banyak aktor maka akan menaikkan waktu eksekusi. Rata-rata kenaikan waktu eksekusi setiap 20 aktor dari semua peta sebesar 1.77 %.
6. Semakin banyak halangan pada peta akan meningkatkan panjang jalur. Rata-rata peningkatan panjang jalur saat terdapat halangan statis pada peta sebesar 36.24 %

sedangkan saat terdapat halangan statis dan dinamis pada peta sebesar 47.74 %.

7. Semakin banyak aktor maka akan memperbesar panjang jalur aktor. Rata-rata peningkatan panjang jalur setiap 20 aktor dari semua peta sebesar 0.86%.
8. Semakin banyak aktor maka akan memperbanyak jumlah pesawat yang hancur. Rata-rata peningkatan jumlah aktor yang hancur setiap 20 aktor dari semua peta sebesar 92.49%.
9. Instansiasi pesawat dilakukan secara random sehingga tabrakan bisa saja terjadi pada antar pesawat karena posisi yang sangat berdekatan.
10. Rata-rata FPS dari semua peta adalah 75.91 yang mana diatas batas FPS yang dapat mengelabui mata yaitu sebesar 20. Rata-rata waktu eksekusi dari semua peta adalah 102.79 ms. Rata-rata panjang jalur dari semua peta adalah 445.56. Rata-rata jumlah aktor yang hancur dari semua peta adalah 107 dari 3300 aktor.

DAFTAR PUSTAKA

GAIJIN ENTERTAINTMENT, 2016. Official Wiki of War Thunder .[online] Gaijin Entertaintment. Tersedia di: < https://wiki. warthunder. com/>[Diakses 10 Maret 2017]

HAGELBACK,J. , 2016. *Hybrid pathfinding in Starcraft.* IEEE Transactions On Computational Intelligence And AI In *Game*s.

HARSONO, I. A. , 2015. *Implemtasi Real-Time Pathfinding Menggunakan A* dan Reynolds Steering Obstacle Avoidance pada Permainan Komputer.* S1. Universitas Malang Raya.

SCHNELL,B. , 2016 .*A Hybrid Architecture for Pathfnding Indynamic Environments.* Hamburg: Institute for Software, Technology, Systems & Hamburg University of Technology.

ZONGXIN, YAO. ET AL. , 2016. *Mission Decision-making Method of Multi-aircraft Cooperatively Attacking Multi-target based on Game Theoritic Framework.* Shenyang: Chinese Society of Aeronautics and Astronautics & Beihang University.