Pergerakan Otonom Pasukan Berbasis Algoritma Boids Menggunakan Metode *Particle Swarm Optimization*

Syahri Mu'min [1], Mochammad Hariadi [2] dan Supeno Mardi Susiki Nugroho [3]

Jurusan Teknik Elekto, Fakultas Teknologi Industri,
Institut Teknologi Sepuluh Nopember Surabaya
email: syahri88@gmail.com [1], mochar@ee.its.ac.id [2], smardi88@gmail.com [3]

Abstrak

Permainan *real-time strategy* (RTS) merupakan sebuah game yang menarik, layar dipisahkan menjadi peta area, unit, dan bangunan. Bermain game RTS umumnya terdiri dari pemain yang diposisikan di suatu tempat di peta dengan beberapa unit atau bangunan, pemain bergerak dari satu tempat ke tempat yang lain. Kelompok karakter atau pasukan yang bergerak dalam permainan RTS memiliki pendekatan luas, untuk masalah ini menggunakan model boids. Kemampuan *Particle Swarm Optimization* (PSO) untuk mencapai posisi optimum menciptakan kemungkinan untuk secara otomatis menghasilkan jalan non deterministic kerumunan pasukan dari satu posisi tertentu. Dalam kasus ini, kami fokus pada pembuatan pola bergerak yang halus dan fleksibel realistis bagi pasukan virtual dengan memanfaatkan fasilitas komputasi yang ditawarkan oleh PSO. Fungsi tersebut digunakan untuk menggambarkan semua jenis objek dalam sistem simulasi , termasuk target statis , hambatan statis , serta pasukan yang dianggap sebagai partikel dalam mencari cara untuk mencapai solusi terbaik.

Kata Kunci : agen otonom, *boids, particle swarm optimization, real-time game strategy*

Abstract

Real-Time Strategy game (RTS) is an interesting game, the screen is divided into the area map, units, and buildings. Play RTS games are generally made up of players who are positioned in a place on the map with multiple units or buildings, the player moves from one place to another. Group of characters or troops engaged in RTS games have broad approach, for this problem boids model. Ability Particle Swarm Optimization (PSO) to achieve optimum position creates the possibility to automatically generate non deterministic way a crowd of troops from certain positions. In this case, we focus on making the pattern moves are smooth and flexible realistic for virtual troops to take advantage of the computing facilities offered by PSO. That function is used to describe all types of objects in the system simulation, including static targets, static obstacles, and the troops which are considered as particles in finding ways to achieve the best solution.

Keywords: autonomous agents, boids, particle swarm optimization, real-time strategy game.

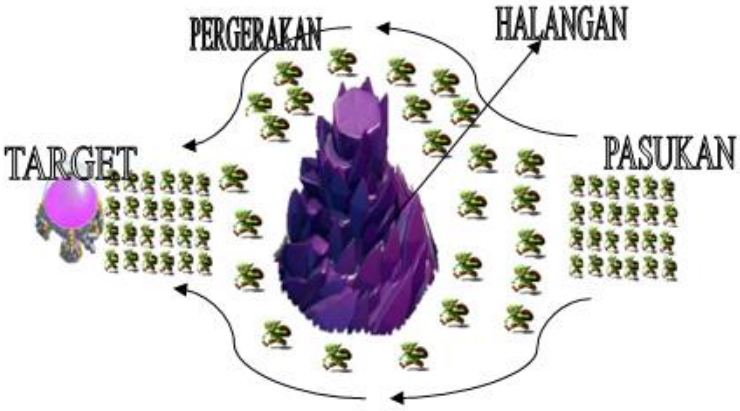
Pendahuluan

Dalam permainan strategi pemain harus mengendalikan pasukan untuk melakukan pertempuran melawan satu atau lebih lawan. Untuk jenis permainan ini dalam memenangkan permainan, pemain harus menggunakan sumber daya yang dikumpulkan seefisien mungkin dan membangun unit dengan optimal. Beberapa strategi permainan sekarang memungkinkan pasukan untuk bergerak dalam formasi sebagai kelompok yang terorganisir, melalui perencanaan penggunaan jalan tingkat lanjut dan kerja tim. Teknik berkelompok juga dapat digunakan untuk memungkinkan pasukan untuk bergerak bersama-sama dalam formasi terorganisir.

Salah satu contoh game RTS adalah *Clash of Clans* (CoC) yang dibuat dan disebarkan oleh Supercell (Clash of Clans Wikia, 2014). Pasukan dalam CoC dapat melakukan serangan berkelompok dengan melakukan pergerakan bersama-sama secara bergerombol seperti pada gambar 1:

Gambar 1 Pergerakan Pasukan Otonom Pada CoC

Pergerakan berkelompok untuk menuju target dalam game CoC terdapat dua tipe yaitu menuju target terdekat dan menuju target tertentu. Pasukan CoC yang menuju target tertentu seperti yang dilakukan oleh *goblin, giant, hog rider* dan *golem* dalam menuju target tertentu dapat menghindari halangan yang ada dengan melakukan pemisahan dan setelah melewati halangan kembali bergerombol seperti pada gambar 2:



Gambar 2 Pergerakan Dengan Halangan Statis

Pergerakan pasukan dengan melakukan pemisahan (*Separation*), menjaga keselarasan kecepatan (*alignmen*) dan menyatu kembali (*Cohesion*) adalah program kehidupan buatan, yang dikembangkan oleh Craig Reynolds pada tahun 1986, yang mensimulasikan perilaku berkelompok burung dan tiga aturan tersebut disebut dengan algoritma Boids (Reynold, 1987). Menurut Meilany Dewi, Algoritma Boids dapat digunakan untuk mengendalikan kerumunan agen dalam menghindari halangan statis dan semakin banyak jumlah agen maka semakin lama untuk sampai ke tujuan (Dewi, Hariadi, & Purnomo, 2011).

Untuk membuat pergerakan pasukan secara otonom yang halus dan fleksibel seperti realita maka diperlukan optimasi dalam pergerakan tersebut. Menurut Ying ping chen untuk mengatur perpindahan kerumunan dalam gafis komputer dapat menggunakan *particle swarm optimization* (PSO) (Ping & Yin, 2009). Setiap agen atau pasukan di inisialisasikan dalam bentuk partikel. Setiap partikel menyesuaikan arah bergerak yang sesuai dengan arah keselarasan dan arah kohesi, serta dalam gerakan konvergen, persamaan pembaruan dari versi standar PSO digunakan dalam menentukan gerakan yang konvergen (Cui & Shi, 2009). Dalam simulasi pergerakan pasukan halangan berupa objek sangat dimungkinkan. Setiap pasukan akan berpindah posisi untuk mencari posisi lain yang dapat menghasilkan fungsi tujuan yang lebih baik dengan memanfaatkan komputasi dalam PSO. Menurut Saleh Alaliyat Algoritma (Alaliyat, Yndestad & Sanfilippo, 2014) *flocking boids* terlihat lebih baik saat menggunakan PSO dan jauh lebih cepat dari pada menggunakan algoritma genetika karena PSO memberikan konvergensi yang lebih cepat dari pada

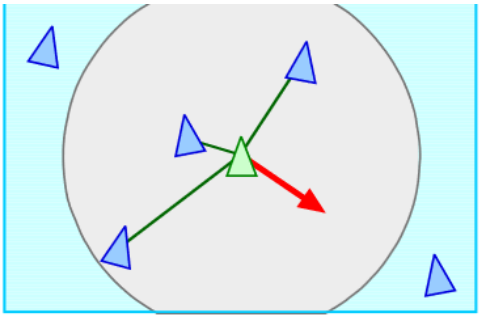
algoritma genetika.

Dalam penelitian ini kami coba mensimulasikan pasukan dengan menggunakan algoritma boids dengan optimasi PSO. Pasukan berpindah untuk mencapai tujuan dengan objek statis atau sumber makanan dengan halangan yang dapat berupa objek statis dan bergerak. Untuk dapat menghindari objek tersebut gerombolan pasukan bergerak menghindarinya dengan merubah arah dalam tujuan, pasukan berpencar dengan mencari posisi yang optimal. Jika halangan bergerak maka pasukan akan berpencar dengan waktu yang lebih lama dalam menemukan posisi optimal. Setelah posisi optimal ditemukan pasukan kembali bergerombol dan menuju tempat tujuan yang telah ditentukan.

Model Boids

Boids adalah program kehidupan buatan, yang dikembangkan oleh Craig Reynolds pada tahun 1986, yang mensimulasikan perilaku berkelompok burung (Reynold, 1987). Algoritma ini menunjukkan struktur dari implementasi khas boids. Tiga aturan algoritma boids terdapat perubahan kecepatan dan arah boid. Setelah kecepatan dan jalannya boids telah diperbarui oleh aturan Reynolds kita dapat memperbarui posisi dari boids.

Perilaku pemisahan kemudi memberikan karakter kemampuan untuk menjaga jarak pemisahan tertentu dari karakter lain di dekatnya. Hal ini dapat digunakan untuk mencegah karakter dari berkerumun bersama-sama. Untuk menghitung kemudi dalam pemisahan, pertama pencarian dilakukan untuk menemukan karakter lain dalam lingkungan tertentu. Ini mungkin pencarian yang lengkap dari semua karakter dalam dunia simulasi, atau menggunakan semacam partisi ruang dalam skema *caching* untuk membatasi pencarian ke karakter lokal. Untuk masing-masing karakter di dekatnya, gaya tolak dihitung dengan mengurangi posisi karakter kita dan karakter di dekatnya, normalisasi, dan menerapkan bobot $^1/r$ yang artinya, posisi vektor offset skala oleh $^1/r2$. Perhatikan bahwa $^1/r$ hanya pengaturan yang telah bekerja dengan baik, bukan nilai fundamental. Lihat Gambar 3

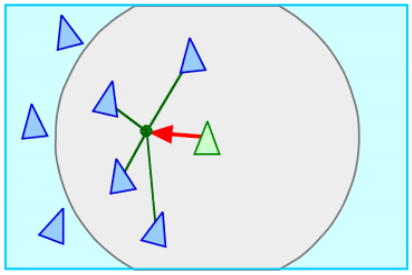


Gambar 3 *Separation*

Separation dapat diformulasikan sebagai berikut (Dewi, Hariadi, & Purnomo, 2011);

$$d(\mathrm{P}_x, \mathrm{P}_b) \leq \mathrm{d}_2 \Rightarrow \vec{V}_{sr} = \sum_{x}^{n} \frac{\vec{V}_x + \vec{V}_x}{d(\mathrm{P}_x, \mathrm{P}_b)}$$

Perilaku kemudi *Cohesion* memberikan karakter kemampuan untuk berpadu dengan karakter lain di dekatnya. Kemudi untuk kohesi dapat dihitung dengan menemukan semua karakter dalam lingkungan lokal seperti dijelaskan di atas untuk pemisahan, menghitung posisi rata-rata atau pusat gravitasi karakter di dekatnya. Kekuatan kemudi dapat diterapkan dalam arah posisi rata-rata dalam mengurangi posisi karakter kita dari posisi rata-rata, seperti dalam boids model asli, atau dapat digunakan sebagai target untuk mencari perilaku kemudi . Lihat Gambar 4.

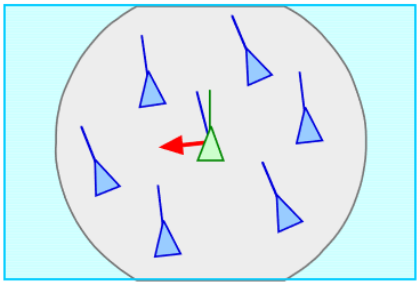


Gambar 4 *Cohesion*

Cohesion dapat diformulasikan sebagai berikut (Dewi, Hariadi, & Purnomo, 2011);

$$d(\mathrm{P}_x, \mathrm{P}_b) \leq d_1 \cap (\mathrm{P}_x, \mathrm{P}_b)\ d_2 \Rightarrow \vec{V}_{cr} = \sum_{x}^{n} (\mathrm{P}_x - \mathrm{P}_b)$$

Perilaku kemudi keselarasan memberikan karakter kemampuan untuk menyesuaikan diri karakter lain di dekatnya, seperti yang ditunjukkan pada gambar 5. Kemudi untuk penyelarasan dapat dihitung dengan menemukan semua karakter dalam lingkungan lokal seperti dijelaskan di atas untuk pemisahan, rata-rata bersama kecepatan atau bergantian, vektor satuan ke depan karakter di dekatnya. Rata-rata ini adalah kecepatan yang diinginkan dan sebagainya vektor kemudi adalah perbedaan antara rata-rata dan kecepatan karakter kita saat ini atau bergantian. Kemudi ini akan cenderung untuk mengubah karakter kita sehingga sejajar dengan kelompok tetangga.



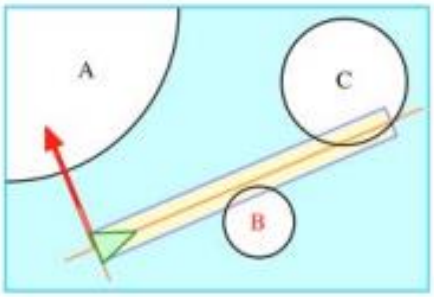
Gambar 5 Aligment

Alignment dapat diformulasikan sebagai berikut (Dewi, Hariadi, & Purnomo, 2011);

$$d(\mathrm{P}_x, \mathrm{P}_b) \leq \mathrm{d}_1(\mathrm{P}_x, \mathrm{P}_b) \quad d_2 \Rightarrow \vec{V}_{ar} = \frac{1}{n}\sum_{x}^{n} \vec{V}_x$$

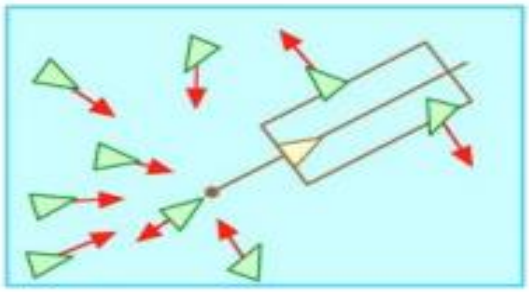
Reynolds (1999) telah menambah model boids termasuk aturan individu berbasis lebih dari kemudi perilaku, untuk memiliki individu yang lebih canggih yang mampu untuk menyelesaikan tugas tertentu atau beradaptasi dengan lingkungan yang kompleks. Beberapa perilaku ini adalah (Reynolds, 1999):

- *Obstacle avoidance* pada gambar 6: Perilaku penghindaran rintangan yang memungkinkan agen bergerak di lingkungan yang ada halangan dengan menghindari di sekitar rintangan.



Gambar 6 *Obstacle Avoidance*

- *Leader Following* pada gambar 7 : perilaku ini menyebabkan satu atau lebih agen untuk mengikuti agen yang terpilih sebagai pemimpin dalam pergerakan berkelompok.



Gambar 7 *Leader Following*

Berdasarkan model Reynolds, kami telah mengimplementasikan Model generik swarm model boids di Unity3D (Unity3d.com 2015), program ini ditulis dalam MonoDevelop Unity C# (Docs.unity3d.com 2015). Tujuannya adalah untuk mengembangkan Model generik yang dapat digunakan dalam simulasi pergerakan pasukan dengan 5 aturan tersebut.

Particle Swarm Optimization

Particle Swarm Optimization adalah salah satu metode optimasi yang terinspirasi dari perilaku model boids yang kemudian tiap objek disederhanakan menjadi sebuah partikel. Suatu partikel dalam ruang memiliki posisi yang dikodekan sebagai vektor koordinat. Vektor posisi ini dianggap sebagai keadaan yang sedang ditempati oleh suatu partikel di ruang pencarian. Setiap posisi dalam ruang pencarian merupakan alternatif solusi yang dapat dievaluasi menggunakan fungsi objektif. Setiap partikel bergerak dengan kecepatan *v*.

Particle Swarm Optimization pertama kali dimunculkan pada tahun 1995 diciptakan oleh James Kennedy dan R.C Eberhart, sejak saat itulah para peneliti banyak menurunkan dan mengembangkan metode PSO.

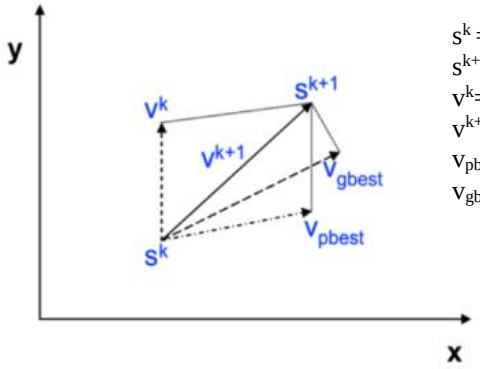
Particle Swarm Optimization memiliki kesamaan sifat dengan teknik komputasi seperti Algoritma Genetika (*Genetic Algorithm*). Sistem PSO diinisialisasi oleh sebuah populasi solusi secara acak dan selanjutnya mencari titik optimum dengan cara meng-update tiap hasil pembangkitan. Metode optimasi yang didasarkan pada swarm intelligence ini disebut algoritma *behaviorally inspired* sebagai alternatif dari algoritma genetika, yang sering disebut evolution-based procedures. Dalam konteks optimasi multi variabel, kawanan diasumsikan mempunyai ukuran tertentu atau tetap dengan setiap partikel posisi awalnya terletak di suatu lokasi yang acak dalam ruang multidimensi. Setiap partikel diasumsikan memiliki dua karakteristik yaitu posisi dan kecepatan. Setiap partikel bergerak dalam ruang tertentu dan mengingat posisi terbaik yang pernah dilalui atau ditemukan terhadap sumber makanan atau nilai fungsi objektif. Setiap partikel menyampaikan informasi atau posisi bagusnya kepada partikel yang lain dan menyesuaikan posisi dan kecepatan masing-masing berdasarkan informasi yang diterima mengenai posisi yang bagus tersebut.

Dengan demikian perilaku kawanan burung akan didasarkan pada kombinasi dari 3 faktor simpel berikut:

1. Kohesi
2. Separasi
3. Penyesuaian (*alignment*)

Metodologi PSO beroperasi dengan menempatkan kelompok pada partikel individu dalam mencapai suatu ruang terus menerus, dimana setiap partikel dijalankan dengan posisi acak dan kecepatan awal acak dalam pencarian ruang kosong. Di lain kata, setiap partikel melacak koordinat dalam ruang solusi yang berkaitan dengan solusi terbaik yang telah dicapai sejauh ini oleh agen. Nilai ini disebut terbaik (*pbest*), nilai personal lain terbaik yang dilacak oleh PSO adalah nilai terbaik diperoleh sejauh oleh partikel di lingkungan partikel itu. Nilai ini disebut (*best*). Jadi konsep dasar PSO terletak pada mempercepat setiap partikel menuju lokasi *pbest* dan *gbest*, dengan akselerasi bobot pada masing-masing langkah pada

waktu seperti yang ditunjukkan gambar 8.



s^k =titik pencarian saat ini
s^{k+1}= titik pencarian modifikasi
v^k=kecepatan arus
v^{k+1}=kecepatan modifikasi
v_{pbest}=kecepatan berdasar pbest
v_{gbest}=kecepatan berdasar gbest

Gambar 8 konsep posisi partikel pada PSO

Partikel-partikel PSO bergerak melalui penelusuran ruang menuju suatu solusi potensial. Pergerakan PSO dipengaruhi oleh nilai *velocity* yang dinamis. Nilai *velocity* ini dipengaruhi oleh tiga faktor, yaitu nilai *velocity* partikel sebelumnya, posisi partikel terbaik dalam populasi (gbest) dan posisi terbaik yang pernah diraih oleh partikel itu sendiri (Pbest). Pada awalnya persamaan velocity PSO adalah seperti berikut ini:

$$v = v + c1 * rand() * (pbest - x) + c2 * rand() * (gbest - x)$$

Keterangan:

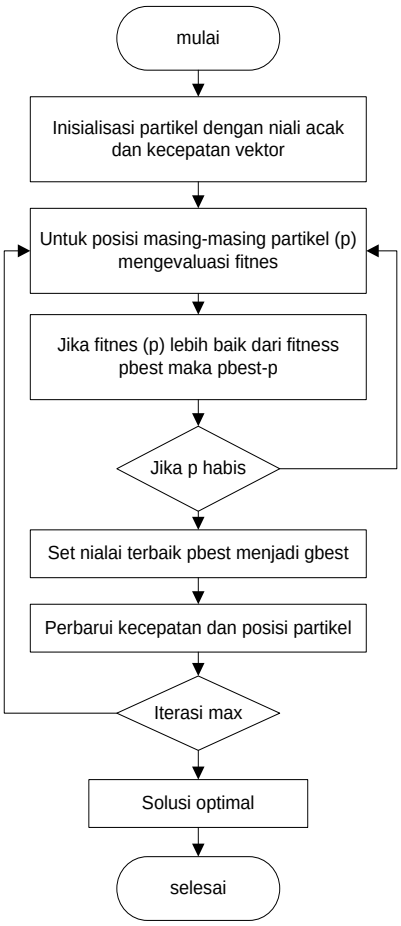
v = velocity sebuah partikel.

c1 dan c2 = correction factor.

pbest = posisi terbaik partikel.

gbest = posisi partikel terbaik dari seluruh partikel yang ada di populasi.

Konsep dasar PSO terletak dalam mempercepat setiap partikel ke arah pbest dan lokasi gbest, dengan bobot akselerasi acak pada setiap langkah. Berikut flow chart PSO:



Gambar 9 *Flow Chart* PSO

Pada tahun 1998 Eberhart dan Shi menambahkan inertia weight (*w*) sebagai faktor yang mengendalikan besarnya pengaruh velocity sebelumnya terhadap velocity saat ini (Y.Shiand & R.C. Eberhart, 1998). Inertia weight juga digunakan sebagai kontrol yang dapat menyeimbangkan antara explorasi global dan exploitasi lokal. Sehingga persamaan velocity menjadi:

$$v = w*v + c1*rand()*(pbest - x) + c2*rand()*(gbest - x)$$

Model ini akan disimulasikan dalam ruang dengan dimensi tertentu dengan sejumlah iterasi sehingga di setiap iterasi, posisi partikel akan semakin mengarah ke target yang dituju (minimasi atau maksimasi fungsi). Ini dilakukan hingga maksimum iterasi dicapai atau bisa juga digunakan kriteria penghentian yang lain.

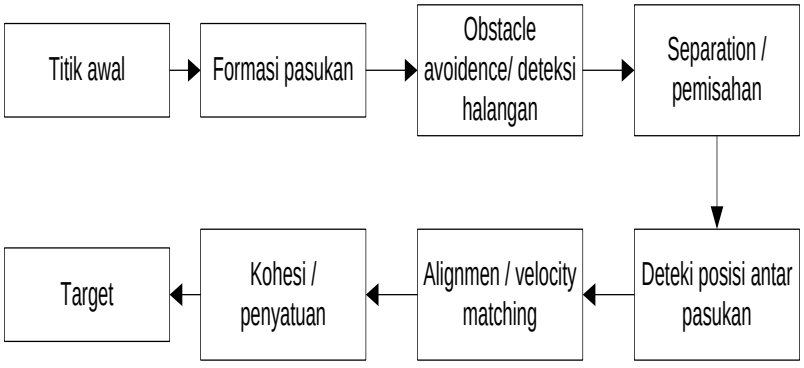
Simulasi BOIDS Menggunakan PSO

Pemodelan dan simulasi gerakan kerumunan ini akan diimplementasikan dengan menggunakan bahasa C# di *game engine* unity3d seperti pada gambar 12 dengan parameter pada tabel 1. Terdapat beberapa halangan pada lingkungan sekitar pasukan. pasukan bergerak menuju target yang telah ditentukan dengan menghindari halangan yang berupa blok dan pepohonan.

Tabel 1 PSO parameter

Populasi	1
Dimensi	4
Iterasi maksimal	100
C1	2
C2	2
Inertia	1
Batas atas variabel	1
Batas bawah variabel	0

Pengujian dilakukan dengan cara mensimulasikan hasil rancangan sistem. Pengujian *collision avoidance* dilakukan agar agen dapat berinteraksi dengan lingkungan tanpa adanya tabrakan terhadap lingkungan yang dilaluinya. Skenario pada percobaan ini terdapat pada gambar gambar 11.

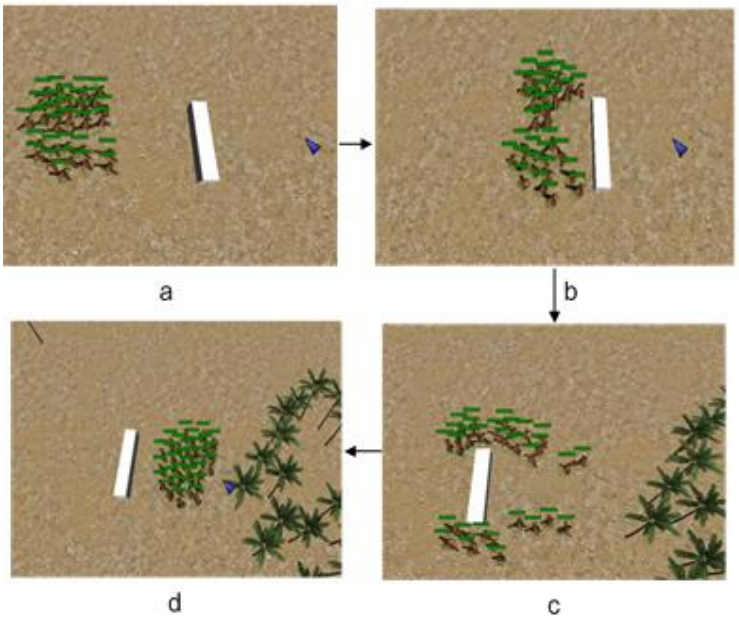


Gambar 10 Skenario Percobaan

Simulasi BOIDS Menggunakan PSO

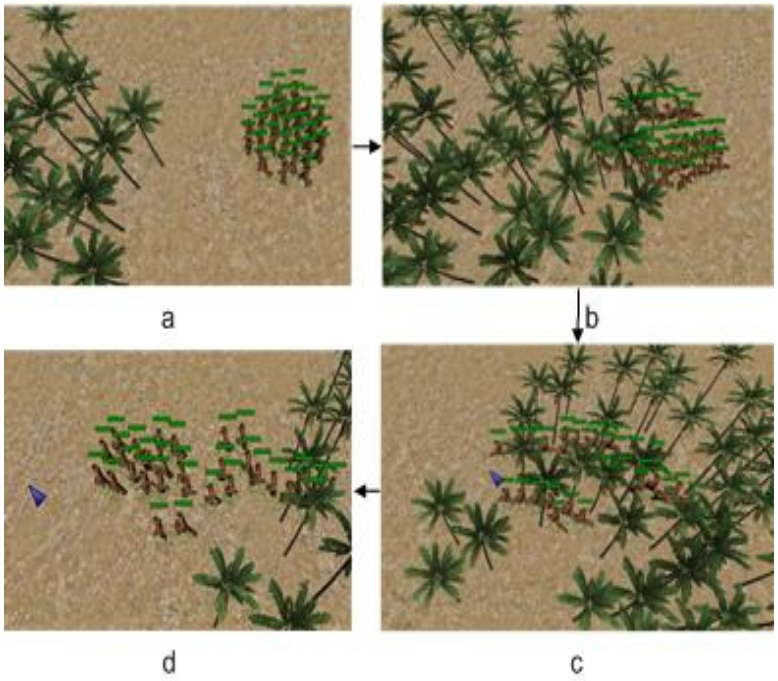
Percobaan dengan skenario pada gambar 10 adalah untuk melihat apakah algoritma PSO dapat digunakan untuk melakukan simulasi melewati halangan di

mana agen akan menggunakan karakteristik kohesi, pemisahan dan keselarasan yang sesuai dengan model boids. Target yang digunakan akan dipilih dari beberapa target yang tersedia. Skenario membuktikan bahwa simulasi ini dapat melewati halangan berupa sebuah balok sedangkan dalam simulasi ini semua agen berada di kecepatan yang sama seperti pada gambar 11.



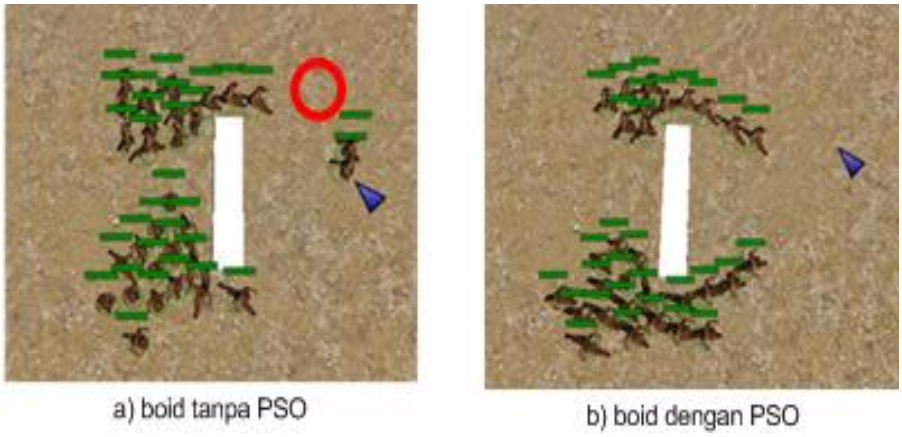
Gambar 11 Pasukan Melewati Sebuah Halangan

Halangan yang terdapat pada gambar 11 hanya ada 1 dan untuk pengujian selanjutnya kami menggunakan halangan lebih dari 1 dengan halangan berupa pepohonan. Simulasi ini berjalan sesuai dengan yang diharapakan karena semua pasukan atau agen dapat melewati halangan berupa pepohonan dan sampai ke target yang telah ditentukan seperti pada gambar 12.



Gambar 12 Pasukan Melewati Pepohonan

Percobaan simulasi pada gambar 11 dan gambar 12 menggunakan model boids tanpa optimasi PSO. Simulasi dapat berjalan menuju target secara berkelompok dengan dengan melewati halangan yang ada. Pada gambar 11c dan gambar 12d terlihat jarak beberapa pasukan tidak saling berdekatan seperti berpisah dari kelompok. Hal tersebut karena pada saat pasukan mempunyai kecepatan yang sama dan untuk menuju target terdapat hambatan berupa halangan. pasukan yang terhambat halangan terpisah dengan pasukan yang sudah melewati halangan. Pasukan mempunyai kecepatan yang sama sehinga tidak dapat melakukan perlambatan dan percapatan untuk melakukan penyelarasan atau *velocity matching* dengan baik. Pada simulasi selanjutnya model boids menggunakan optimasi PSO sehingga pasukan dapat melakukan penyelarasan lebih baik seperti pada gambar 13.



Gambar 13 Penyelarasan Kecepatan

Kesimpulan Dan Saran

Model boids ini sering digunakan dalam komputer grafis untuk mewujudkan representasi dari kehidupan buatan secara realistis dari pergerakan. Misalnya, banyak animasi memerlukan perilaku yang terlihat alami dari sejumlah besar karakter (boids). Pergerakan kelompok merupakan hasil dari interaksi individu, jadi setiap individu menghasilkan gerak sendiri, ini lebih mudah dan menghasilkan gerakan alami. pergerakan pasukan dihitung dengan menggabungkan semua vektor *steering behaviour*. Untuk memiliki perilaku alam di lingkungan yang berbeda, gerakan boids itu harus dioptimalkan dan disesuaikan. Pasukan melakukan perlambatan dan percepatan untuk menyelaraskan posisi antar agen atau pasukan. Algoritma PSO digunakan untuk mengoptimalkan model boids dengan mengoptimalkan koefisien dari vektor bergerak untuk meminimalkan fungsi *cost* dan *flocking* terlihat lebih bagus.

Optimasi penyelarasan dapat menggunakan metode yang lain semisal genetic algoritma, artificial bee colony dan metode yang berkaitan dengan pengendalian pergerakan berkelompok. Pada penelitian selajutnya mungkin dapat membandingkan kecepatan konvergensi PSO dengan algoritma yang lain dalam mengoptimalkan model boids.

Referensi

Alaliyat S., Yndestad H. , Sanfilippo F., Optimisation of Boids Swarm Model Based on Genetic Algorithm and Particle Swarm Optimisation Algorithm (Comparative Study), *Proceedings of the 28th European Conference on Modelling and Simulation (ECMS)*, Brescia, Italy; 2014.

Chen, Ying-Piing., Lin. Ying-Yin., "Controlling the movement of crowds in computer graphics by using the mechanism of particle swarm optimization", *Aplied Soft Computing Elsevier*. 2009.

Cui Z., Shi Z. "Boids Particle Swarm Optimization", *International Journal of Innovative Computing and Applications*, 2009.

http://clashofclans.wikia.com/wiki, diakses maret 2015.

M. Dewi, M. Hariadi, and M.H. Purnomo, "Simulating The Movement Of The Crowd In An Environment Using Flocking", *IEEE International conference on Instrumentaion, Communication, Information Technology and Biomedical Engineeering*. 2011.

J. Kennedy dan R. C. Eberhart. Particle Swarm Optimization. *In Proceedings of the 1995 IEEE International Conference on Neural Networks*. IEEE Service Center, Piscataway, 1995.

Y.Shiand dan R.C. Eberhart. Parameter selection in particle swarm optimization. In V. W. Porto, N. Saravanan, D. Waagen, and A. Eibe, editors, *Proceedings of the Seventh Annual Conference on Evolutionary Programming*, Springer-Verlag, 1999

[halaman ini sengaja dikosongkan]