KAJIAN ALGORITMA CRAIG RAYNOLD PADA KERUMUNAN (FLOCKING)

Lit Malem Ginting[1] , Binsar Siahaan[2] , Bastian Situmorang[3] , Riris Manik[4]

Institut Teknologi Del
Email : litmalem@gmail.com [1], if416033@students.del.ac.id [2], if416034@students.del.ac.id [3], if416035@students.del.ac.id [4]

ABSTRACT

Artificial Intelligence (AI) is a study of making computers that do things that today can be done better by humans. One application in the game is to mimic the behavior of animals in the wild. Set of animal in the wild are often called crowds, herds, schools and swarms. Crowds are animals that gather together in an irregular and unfocused way [4]. In 1987, Craig Reynolds [6] with his book "Flock, Herd, Schools: A Distributed Behavioral Model" invented a technique for simulating animal crowds in the wild. The technique is called flocking which is commonly known as "boids". In this study, an observation about the comparison of boids with the amount of resources used in a simulation game that will be given a device. The number of boid that will be used is 1-500 boid with cohession radius distance = 10 and 15, radius separator = 4 and save radius = 6 with the observation at the angle of 300 and 1800. Based on the results done, then the result is boids neighbor detection radius has a major effect on resource usage.

Keywords: *Artificial Intelligence (AI), Flocking, Craig Raynold, resource device*

ABSTRAK

Artificial Intelligence (AI) merupakan sebuah studi tentang bagaimana membuat komputer melakukan hal-hal yang pada saat ini dapat dilakukan lebih baik oleh manusia. Salah satu penerapan AI dalam game adalah dengan menirukan perilaku hewan di alam bebas. Kerumunan di alam bebas sering disebut dengan crowd, flock, school maupun swarm. Crowd merupakan sejumlah hewan yang berkumpul bersama-sama dengan cara tidak teratur dan tidak terarah [4]. Pada tahun 1987, Craig Reynolds [6] dengan bukunya yang berjudul "Flock, Herd, Schools: A Distributed Behavioral Model" menemukan sebuah teknik untuk mensimulasikan kerumunan hewan di alam bebas. Teknik tersebut dinamakan flocking yang secara umum dikenal dengan istilah "boids". Dalam penelitian ini, dilakukan pengamatan mengenai analisis mengenai perbandingan antara ukuran boids dengan jumlah resource yang digunakan pada sebuah simulasi game yang akan dijalankan pada sebuah device. Adapun jumlah boid yang akan digunakan adalah 1-500 boid dengan jarak cohession radius = 10 dan 15, Separation radius = 4 dan save radius = 6 dengan pengamatan pada sudut yaitu 300 dan 1800. Berdasarkan hasil yang dilakukan, maka diperoleh hasil bahwa radius deteksi neighbor dari boids memiliki pengaruh yang besar terhadap penggunaan resource.

Kata kunci: *Artificial Intelligence (AI), Flocking, Craig Raynold, Resource Device*

PENDAHULUAN

Perkembangan *game* di dunia semakin pesat, khususnya *mobile game*. *Game* merupakan bentuk rekreasi universal, umumnya termasuk aktifitas yang melibatkan permainan ataupun hiburan dan sering membangun situasi yang melibatkan kontes maupun persaingan.[1]

Seorang peneliti bernama Goldeneye berhasil menunjukkan kepada pengguna *game* bahwa dengan menerapkan *Artificial Intelligence* (AI) pada *game* dapat meningkatkan pengalaman bermain yang ditawarkan dalam sebuah *game*. *Artificial Intelligence* (AI) merupakan sebuah studi tentang bagaimana membuat komputer melakukan hal-hal yang pada saat ini dapat dilakukan lebih baik oleh manusia. Sejak pertengahan tahun 1990, AI mulai menjadi salah satu unsur yang diperhatikan dan dikembangkan oleh perusahaan pengembang *game*.[7] Penerapan AI yang pada suatu *game* berbeda-beda tergantung pada *genre* dari *game* itu sendiri. Salah satu penerapan AI dalam *game* adalah dengan menirukan perilaku hewan di alam bebas. Kerumunan di alam bebas sering disebut dengan *crowd*, *flock, school* maupun *swarm*.

Crowd merupakan sejumlah hewan yang berkumpul bersama-sama dengan cara tidak teratur dan tidak terarah[4]. Pada tahun 1987, Craig Reynolds [6] dengan bukunya yang berjudul "Flock, Herd, Schools: A *Distributed Behavioral Model"* menemukan sebuah teknik untuk mensimulasikan kerumunan hewan di alam bebas. Teknik tersebut dinamakan *flocking* yang secara umum dikenal dengan istilah "*boids*".

Penelitian yang dilakukan oleh Meilany Dewi dkk [9] mensimulasikan pergerakan dari kerumunan manusia yang bergerak keluar dari suatu pintu. Pada Penelitian ini, penulis akan melakukan penelitian mengenai performa algoritma *flocking* Craig Reynold dengan menggunakan jumlah data yang lebih besar serta penerapan *obstacle avoidance algorithm* dalam penelitian selanjutnya. Peneliti akan melakukan pengkajian terhadap masalah penggunaan *resource* dalam implementasi *boids*. Peneliti akan melakukan analisis mengenai perbandingan antara ukuran *boids* dengan jumlah *resource* yang digunakan pada sebuah simulasi game yang akan dijalankan pada sebuah *device.*

Tujuan

Penelitian ini bertujuan untuk mengimplementasikan Algoritma Craig Reynold pada simulasi game, Mengamati pola transisi antar boid dan pengaruh dari pola transisi terhadap penggunaan resource dan melakukan kajian, untuk mengetahui pengaruh penggunaan *resource* pada *device* pada saat menjalankan simulasi *game* yang menerapkan algoritma Craig Raynold tersebut.

Dengan adanya penelitian ini, diharapkan pengembang yang ingin menggunakan Algoritma Craig Raynold pada penerapan simulasi game, dapat menggunakan hasil analisis ini sebagai dasar pengembangannya untuk menentukan jumlah *boid* yang akan digunakan dengan alasan yang lebih jelas dan lebih terukur.

Ruang Lingkup

Lingkup kajian dalam pelaksanaan penelitian ini adalah:

1 Tahapan yang dilakukan dalam penerapan algoritma Craig Raynold adalah *Alignment, Cohession* dan *Separation.*

2 Untuk Jumlah Boid yang diuji adalah antara 1-500 boids dengan radius yang berbeda.

3 Radius Angel yang digunakan dalam penelitian ini adalah 30^0, 45^0, 180^0 dan 360^0.

4 Pembangunan Simulator dibangun dengan menggunakan Unity Engine dan akan memanfaatkan beberapa fitur yang disediakan oleh Unity untuk membantu penulis dalam melihat penggunaan RAM dan CPU dari *device* yang digunakan.

5 Jenis simulasi *game* yang akan dibuat pada Penelitian ini adalah simulasi *game* 2D.

Swarm Inteligence

Swarm Intelligence adalah kecerdasan kolektif yang muncul dari sekelompok agen sederhana.[15] '*Swarm'* dapat diartikan sebagai sejumlah besar agen sederhana yang homogen. Agen sederhana ini akan berinterak.si secara langsung dalam lingkungan mereka sendiri tanpa adanya kontrol terpusat yang memungkinkan munculnya perilaku baru.[16]

Contoh interaksi langsung adalah melalui kontak visual atau suara seperti gerakan mengibas dari lebah madu, jalan feromon semut untuk mencari sumber makanan. Jenis komunikasi tidak langsung berarti komunikasi melalui lingkungan[18]. Swarm Intelligence Model adalah jenis kecerdasan buatan yang terinspirasi dari berbagi macam kumpulan hewan di alam bebas. Sampai saat ini berbagai jenis swarm intelligence yang menirukan perilaku di alam bebas telah berhasil diterapkan. Contoh dari swarm intelligence model seperti: Ant Colony Optimization[19][15], Particle Swarm Optimization[20], Artificial Immune System dan Glowworm Swarm Optimization. Swarm Intelligence model yang paling popular adalah Ant Colony Optimization dan Particle Swarm Optimization

Algoritma Craig Raynold

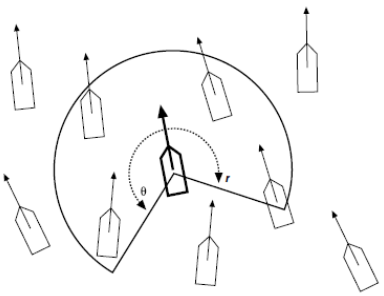
Dalam alam bebas, beberapa jenis hewan bergerak dengan berkelompok dan dapat menyesuaikan dirinya dengan pergerakan sekitarnya. Pergerakan hewan tersebut terlihat acak namun sebetulnya bergerak dalam sebuah sinkronisasi.

Salah satu langkah untuk dapat mempelajari perilaku hewan tersebut adalah dengan membuat simulasi. Perilaku berkelompok hewan di alam bebas disimulasikan pertama kali oleh Craig Reynold pada tahun 1986 dan menyebut program simulasi tersebut dengan "*Boids*". Sampai saat ini *boids* masih digunakan dalam simulasi perilaku kelompok hewan.[1] Setiap boids dalam kerumunan harus mengetahui posisi dan kecepatan boids disekitanya (neighbour).

Dalam melakukan simulasi terdapat 3 aturan yang didefinisikan oleh Craig Reynold.

1. ***Separation*** yaitu pemisahan dengan kelompok lokal
2. ***Alignment*** yaitu mengarahkan sesuai dengan tujuan kelompok
3. ***Cohesion*** yaitu mengarahkan agar pergerakan sesuai dengan posisi kelompok

Setiap boids dalam kerumunan harus mengetahui posisi dan kecepatan boids disekitanya (*neighbour*). Setiap kali kerumunan bergerak, posisi setiap *boids* akan berubah secara konstan sehingga setiap *boids* harus memperbaharui informasi *neighbour* selama *game* berjalan. Jumlah *neighbour* suatu *boid* akan menentukan perilaku dari kerumunan (*flock*).



Gambar 1. Neighbour Detection

Separation

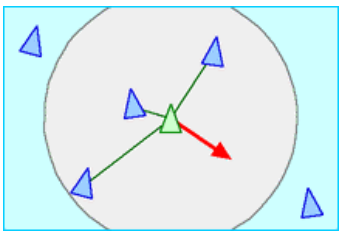
Setiap boids akan menjaga jarak dan tidak tertabrakan dengan boids yang berada disekitarnya. Sangat penting untuk dicatat bahwa jarak dari yang boids mulai menghindari satu sama lain harus kurang dari jarak dari mana boids menarik satu sama lain (karena aturan kohesi). Jika tidak ada kumpulan akan dibentuk. Aturan ini akan mengarahkan (steering) dalam pemindahan boids dan menghindari kerumunan yang berdekatan. Hal ini memungkinkan boids untuk:

1. Menghindari tabrakan

2. Menjaga boids tetap terpisah dalam jarak tertentu (tidak terlalu dekat)

Adapun persamaan untu menjaga agar boid mengalami separasi adalah sebagai berikut:

$$d(P_x,P_b) \le d_2 \Rightarrow \vec{v}_{sr} = \sum_x^n \frac{\overrightarrow{\vec{v}_n + \vec{v}_b}}{d(P_x,P_b)} \quad \ldots\ldots\ldots\ldots\ldots(1)$$



Gambar 2. Separation Image

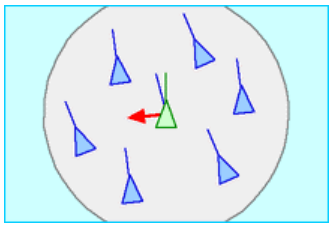
Alignment

Mengarahkan *boids* sesaui dengan tujuan (arah) kelompok. Aturan ini akan membuat *boids* menirukan haluan dan kecepatan *boids* lain disekitarnya. Jika aturan ini tidak diterapkan *boids* akan melompat dan tidak membentuk pola kelompok yang sesuai dengan di alam bebas.

Hal ini akan memungkinkan *boids* untuk:

1. Menyeimbangkan pemisahan (*separation*)
2. Membuat *boids* bergerak dalam satu arah dan tujuan, dengan fungsi yang sama

$$d(P_x,P_b) \le d_1 \cap (P_x,P_b) \ge d_2 \Rightarrow \vec{v}_{ar} = \frac{1}{n}\sum_x^n \vec{v}_x \quad \ldots\ldots (2)$$



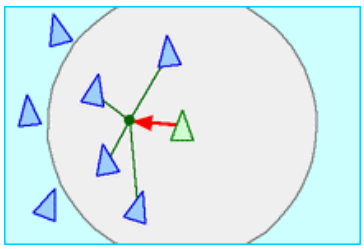
Gambar 3. Alignment Image

Cohession

Boids akan menjaga jarak agar dekat dengan kelompok. *Cohesion* adalah aturan yang menjaga kerumunan tetap bersama, tanpa *cohesion* tidak akan ada kerumunan sama sekali. Ketika diimplementasikan algoritma akan mencai posisi dari boids tetangga dan bergerak kearahnya. Dengan aturan ini *boids* dimungkinkan untuk:

1. Tetap bersama dengan kerumunannya
2. Menyatukan kumpulan yang terpisah kedalam kerumunan

$$d(P_x,P_b) \le d_1 \cap (P_x,P_b) \ge d_2 \Rightarrow \vec{v}_{cr} = \sum_x^n \frac{}{P_x - P_b} \quad \ldots\ldots\ldots\ldots(3)$$



Gambar 4. Cohession Role

Unity

Unity merupakan sebuah game engine yang membantu para *developer* dalam mengembangkan *game* berbasis 2d / 3d dan multi platform. Dengan menggunakan Unity Software, developer hanya membuat 1 kode program untuk berbagai platform seperti Android, iOS, Windows Phone, PC, Mac, Linux, PS3 atau XBOX 360.

Pada penelitian ini, akan menggunakan service *Unity Performance Reporting*. *Unity Performance Reporting* akan otomatis meengumpulkan *error* yang terjadi pada aplikasi dan *platform*, sehingga *error* dapat ditangani dengan segera.

1. *Collect error:* Mengumpulkan error dari seluruh perangkat dan platform
2. *React in real-time:* Secara otomatis mengumpulkan dan menunjukkan error segera setelah ditemukan pada aplikasi
3. *CPU Usage Profiler* menampilkan alokasi penggunaan waktu dalam game. Ketika dipilih, jendela bagian bawah menampilkan tingkatan alokasi waktu pada setiap frame

4. Terdapat dua model yang dapat digunakan dalam memory profiler untuk memeriksa penggunaan memory dalam aplikasi.

PEMBAHASAN

Secara garis besar algoritma *boids* dilakukan 3 tahapan yaitu *Separation*, *Alignment* dan *Cohesion*. Pada bagian ini dijelaskan mengenai analisis dari masing-masing tahapan tersebut.

Dalam penerapan aturan Craig Reynold tidak diperbolehkan salah satu *rule* lebih dominan dibandingkan yang lainnya. Jika *cohesion* lebih dominan, *boids* akan berkerumun namun kemungkinan akan bertabrakan atau bergerak melewati *boids* lain. Kekuatan setiap *rule* harus dipastikan seimbang.

Avoidance rule akan memisahkan sebuah *boids* dengan *boids* lainnya dan dalam waktu yang sama juga akan mempertahankan jarak untuk tetap dekat dengan *boids* lainnya dengan menggunakan *Cohession rule*. Dengan begitu ketika *boids* terpisah jauh, *Cohession rule* akan bekerja menyatukan *boids* dan ketika *boids* terlalu dekat, *Avoidance rule* akan memisahkan *boids*. Dalam hal ini jarak pemisahan *boids* sangat berpengaruh.

Variabel

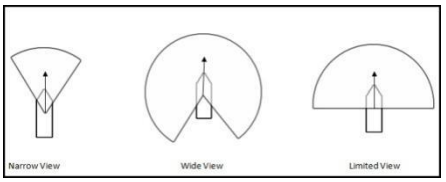
Dengan menggunakan algoritma Craig Reynold, terdapat beberapa variabel yang mempengaruhi penggunaan *resource* pada saat Algoritma Craig Reynold diimplementasikan pada sebuah *game*. Adapun variabel yang mempengaruhi penggunaan *resource* tersebut adalah :

1. Jumlah *boids*
2. Ukuran jari jari dari *neighbour*

Boids

Sebuah *boids* memiliki jari jari dan sudut. Semakin besar jari jari sebuah *boids* maka semakin besar jumlah *neighbour* yang dapat dilihat atau dijangkau oleh *boids*. Semakin besar jari jari sebuah *boids* akan menghasilkan kerumunan yang lebih kompak. Namun besarnya jari jari sebuah *boid* berbanding lurus terhadap besarnya penggunaan *resource* dikarenakan *looping neighbour* yang semakin besar pula.

Dalam implementasi, pandangan yang paling luas adalah 360^o. Beberapa implementasi *flocking* menggunakan pandangan 360^o karena lebih gampang untuk diimplementasikan. Namun hasilnya dalah kerumunan yang tidak realistis. Biasanya dalam implementasinya sebuah boids menggunakan paling tidak pandangan 270^o untuk mendapatkan kerumunan yang baik.



Gambar 5. Analisis Bentuk Boids

Perancangan Algoritma Craig Raynold

Pada perancangan simulasi yang dilakukan dilakukan beberapa perencanaan yaitu :

1. Satu boid bertemu dengan satu boid
2. Bagaimana sebuah boid akan bertemu dengan satu boid yang lain. Kemudian dilakukan analisis bagaimana kedua boids tersebut akan bertemu dan menghitung resource dari device yang akan menjalankan simulasi tersebut.
3. Satu boid bertemu dengan satu kerumunan
4. Bagaimana sebuah boid akan bertemu dengan satu kerumunan boid lain. Kemudian dianalisis bagaimana kedua boids tersebut akan bertemu dan menghitung resource dari Device yang akan menjalankan simulasi tersebut.
5. Satu kerumunan boids bertemu dengan satu kerumunan boids

Bagaimana satu kerumunan *boids* akan bertemu dengan satu kerumunan *boids* yang lain. Kemudian dianalisis bagaimana kedua *boids* tersebut akan bertemu dan menghitung *resource* dari *device* yang akan menjalankan simulasi tersebut. Kerumunan

boids yang akan diamati sesuai dengan jumlah *boids* yang ditetapkan pada rancangan jumlah *boids* tersebut.

Implementasi Simulator

Pada bagian ini akan dijelaskan mengenai implementasi simulator yang digunakan pada penelitian ini. Tahapan yang dilakukan antaralain :

1. Membuat *role* dari Algoritma Craig Raynold yang terdiri dari *separation, Allignment, Cohession, Neighbour*. Adapun *pseudocode* yang digunakan adalah sebagai berikut :

```
Data : A Boid
Result : The course of the boid is
updated.
goal ← (0,0);
neighbours ← getNeighbours(boid);
foreach nBoid in neighbours do
        goal ← goal +
positionOf(boid)-
positionOf(nBoid);
end
goal ← goal / neighbours.size();
steerThoward(goal,boid);
```

Pseudocode 1 Separation Pseudocode

```
Data : A Boid
Result : The course of the boid is
        updated
goal ← (0,0);
neighbours ← getNeighbours(boid);
foreach nBoid in neighbours (boid);
        goal←          goal          +
        positionOf(nBoid);
end
goal ← goal/neighbours.size();
     steerThoward(goal,boid);
```

Pseudocode 2 Cohession Pseudocode

```
Result : The course and velocity
        of the boid is updated.
dCourse ←0;
dVelocity ← 0;
neighbours                          ←
        getNeighbours(boid);
foreach nBoid in neighbours do
dCourse←          dCourse          +
        getCourse(nBoid)           -
        getCourse(boid);
dVelocity←        dVelocity        +
        getVelocity(nBoid)         -
        getVelocity(boid);
end
dCourse                             ←
        dCourse/neighbours.size()
        ;
dVelocity←
        dVelocity/neighbours.size
        ();
boid.addCourse(dCourse);
boid.addVelocity (dVelocity);
```

Pseudocode 3 Alignment Pseudocode

2. Membuat simulator dengan Unity

Pada bagian ini akan dijelaskan proses implementasi pembangunan simulator yang dibangun. Tahap yang dilakukan dalam pembuatan simulator adalah:

a) Membuat kode program sesuai dengan algoritma Craig Reynold. Untuk menjalankan *flocking* sesuai dengan algoritma Craig Raynold, boid tersebut harus sesuai dengan 3 aturan yang terlah ditetapkan yaitu :

Cohession

Pada tahapan *cohession*, *boid* harus selalu bersama-sama dalam jarak dan radius yang sama. Untuk menjaga agar *boids* selalu berada pada jarak yang sama digunakan kode program seperti berikut ini:

```
Void          CohesionSystem(float
deltaTime){
var goal = Vector2.zero;
var max = AllBoids.Count;

for (int i = 0; i < max; i++) {
var selectedBoid = AllBoids [i];
goal = Vector2.zero;

var          maxNeighbour          =
selectedBoid.CohesionNeighbour.C
ount;
for    (int    j    =    0;    j    <
maxNeighbour; j++) {
goal         =         goal         +
selectedBoid.CohesionNeighbour
[j].Position;
}
goal         =         goal         /
selectedBoid.CohesionNeighbour.C
ount;
selectedBoid.CohessionVelocity  =
goal.normalized;
                }
        }
```

Pseudocode 4 Cohesion Pseudocode

Separation

Pada tahapan *separation*, setiap *boids* akan menjaga jarak dan tidak tertabrakan dengan *boids* yang berada disekitarnya. Sangat penting untuk dicatat bahwa jarak dari yang *boids* mulai menghindari satu sama lain harus kurang dari jarak dari mana boids menarik satu sama lain (karena aturan kohesi).

Kode program untuk membuat masing-masing boids melakukan *separation* dapat dilihat seperti berikut:

```
void SeparationSystem(float
deltaTime){
var goal = Vector2.zero;
var max = AllBoids.Count;
for (int i = 0; i < max; i++) {
var selectedBoid = AllBoids [i];
goal = Vector2.zero;

var maxNeigbour =
selectedBoid.SeparationNeighbour.C
ount;
for (int j = 0; j < maxNeigbour;
j++) {
goal = goal +
(selectedBoid.Position -
selectedBoid.SeparationNeighbour[j
].Position);
        }
goal = goal /
selectedBoid.SeparationNeighbour.C
ount;
selectedBoid.SeparationVelocity =
goal.normalized;
                }

        }
```

Pseudocode 5 Seperation Pseudocode

Alignment

Pada tahapan *alginment*, mengarahkan agar kumpulan *boids* mengarah ke tujuan yang sama, sehingga tidak terjadi tabrakan antar *boids* tersebut. Kode program untuk membuat masing-masing *boids* melakukan separation dapat dilihat seperti berikut:

```
VoidAligmentSystem(float
deltaTime)
{
  var goal = Vector2.zero;
  var max = AllBoids.Count;
  for (int i = 0; i < max; i++) {
    var  selectedBoid  =  AllBoids
[i];
    goal = Vector2.zero;
var maxNeighbour =

selectedBoid.CohesionNeighbour.Cou
nt;
for (int j = 0; j < maxNeighbour;
j++)
     {
 goal = goal +
 selectedBoid.CohesionNeighbour
[j].Velocity;
        }

maxNeighbour=selectedBoid.Separati
onNeighbour.Count;
for (int j = 0; j < maxNeighbour;
j++) {
goal = goal +
 selectedBoid.SeparationNeighbour
[j].Velocity;
                        }

Var totalNeighbour =

selectedBoid.SeparationNeighbour.C
ount +

selectedBoid.CohesionNeighbour.Cou
nt;

goal = goal / totalNeighbour;
    selectedBoid.AligmentVelocity
    =goal.normalized;
                        }


        }
```

Pseudocode 6 Seperation Pseudocode

b) Proses penambahan boid kedalam simulator untuk menghitung penggunaan resource
c) Penghitungan Penggunaan Resource
d) Untuk menghitung penggunaan resource, digunakan fitur yang terdapat pada unity secara langsung untuk mempermudah pengambilan data. Digunakan fitur Profiler untuk melihat secara langsung penggunaan resource pada saat menjalankan simulasi game tersebut.
e) Selesai

Eksperimen

Pada sub bab ini, akan dijelaskan tentang hasil eksperimen, dimana eksperimen dilakukan berulang dengan membedakan jumlah boids, ukuran radius boids dan sudut boids. Sesuai dengan tujuan yang pernah dijelakan sebelumnya bahwa bagaimana jumlah boids akan mempengaruhi kenaikan penggunaan resource.

Hasil Eksperimen Kumpulan Boids

Pada eksperimen yang dilakukan terhadap kelompok boids, dibutuhkan kelompok yang berbeda yaitu 1, 10, 50, 100, 200, 300, 400, 500 dengan cohession radius = 10 save radius = 4 dan separation radius = 3. Boids akan dijalankan dalam simulasi dalam environment tak terbatas.

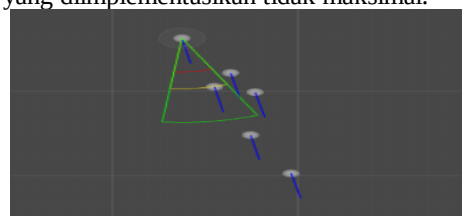
Hasil Eksperimen Range Boids

Pada eksperimen yang dilakukan terhadap kelompok boids, dibutuhkan kelompok yang berbeda yaitu range kecil dan luas. Boids dengan range kecil = 10, boids medium = 15 Boids akan dijalankan dalam simulasi dalam environment tak terbatas

Hasil Eksperimen Sudut Boids

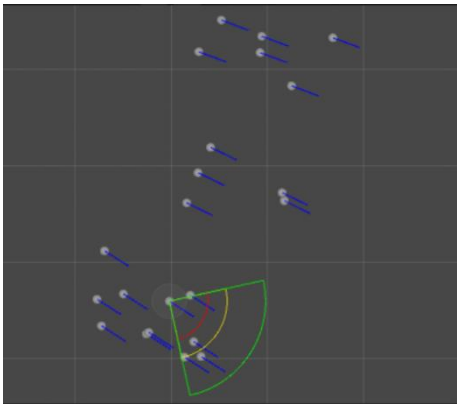
Pada eksperimen yang dilakukan terhadap kelompok boids, dibutuhkan kelompok yang berbeda yaitu dengan sudut 30^0 dan 180^0. *Boids* akan dijalankan dalam simulasi dalam *environment* tak terbatas.

Sudut radius *boids* menentukan bentuk dan perilaku flock. Pada experiment yang menggunakan sudut 30^0 , kelompok boid akan bergerak sejajar dan dengan kelompok yang ramping. Hal ini dikarenakan *vision boid* yang kecil sehingga *role* Craig Raynol yang diimplementasikan tidak maksimal.



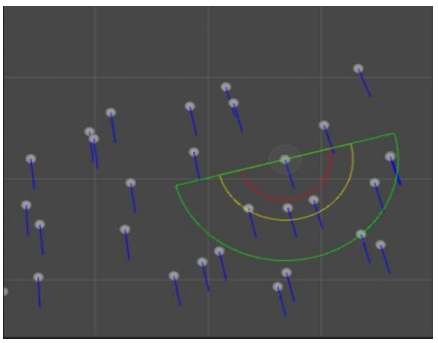
Gambar 6. Eksperimen Sudut Boids 30^0

Pada experiment yang menggunakan sudut 90^0, kelompok *boid* akan bergerak sejajar dan dengan kelompok yang ramping. Ketika mengimplementasikan sudut 90^0 penentuan neighbour tidak maksimal sehingga boid tidak separate dengan sempurna.



Gambar 7. Eksperimen Sudut Boids 90^0

Pada experiment yang menggunakan sudut 180^0, boids terpisah merata namun tidak mengenali neighbour yang tidak tepat berada di belakang sudut radius.

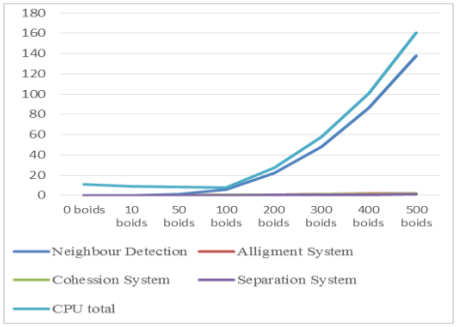


Gambar 8. Eksperimen Sudut Boids 180^0

Pada experiment yang menggunakan sudut 360^0, kelompok boid akan bergerak dengan jarak yang sempurna. Sudut 360^0 merupakan implementasi paling baik pada boid namun dengan penggunaan resource paling banyak pula

KESIMPULAN

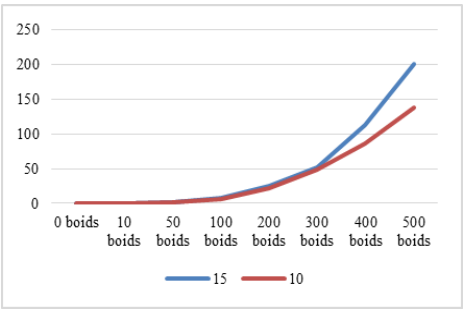
Berdasarkan eksperimen yng telah dilakukan terhadap perngujian perbedaan jumlah, luas radius dan sudut boids, diperoleh hasil sebagai berikut: Semakin besar jumlah boids yang digunakan maka semakin besar pula resource yang dibutuhkan dan pertumbuhan jumlah boids tidak linear dengan pertumbuhan penggunaan resource. Pada keseluruhan grafik pada jurnal ini, sumbu horizontal merupakan jumlah *Boids* dan sumbu vertikal merupakan penggunaan *resource*.



Gambar 9. Grafik Penggunaan Resource Berdasarkan Roles

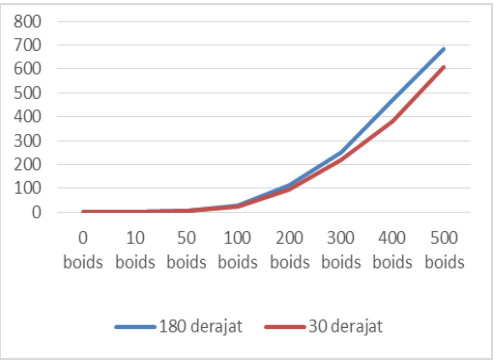
Dalam penerapan algoritma Craig Reynolds, tahapan yang paling banyak menggunakan resources adalah neighbour detection. Lebih dari 80% resource yang digunakan dalam implementasi algoritma dibebankan pada neighbour detection dan sisanya digunakan untuk menghitung cohession, separation dan alligment

Luas radius boids mempengaruhi peningkatan penggunaan resource saat menjalankan algoritma Craig Reynold. Dengan menggunakan kelompok boids yang sama dan sudut radius yang sama, pertumbuhan kebutuhan resource bertambah dalam jumlah yang signifikan. Berikut adalah grafik perbandingan penggunaan resource antara boids dengan luas radius neighbour 10^0 dan 15^0.

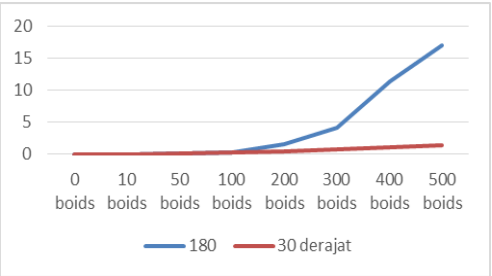


Gambar 10. Grafik Penggunaan Resource Berdasarkan Range Radius

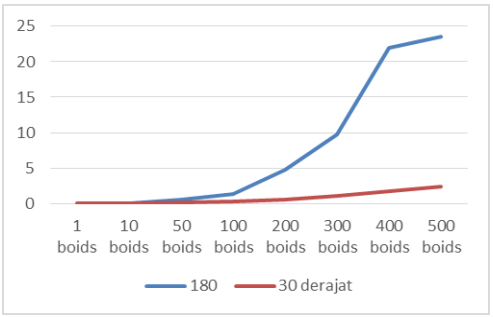
Dengan kelompok boids dan range boids yang sama namun dengan menggunakan sudut yang berbeda membuktikan bahwa sudut pada range boids memiliki pengaruh terhadap penggunaan resource. Pengaruh signifikan tidak pada neighbour detection namun pada rule alligment, cohession dan separation. Berikut hasil pengujian terhadap kelompok boid dengan sudut 180^0 dan 30^0.



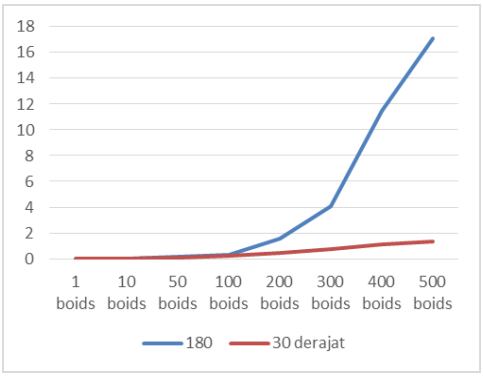
Gambar 11. Grafik Penggunaan Resource Pada Neigbour Detector



Gambar 12. Grafik Penggunaan Resource Pada Separation System



Gambar 13. Grafik Penggunaan Resource Pada Alignment



Gambar 14. Grafik Penggunaan Resource Pada Separation

PENUTUP

Adapun kesimpulan yang diperoleh selama pengerjaan Penelitian antara lain :

1. Hal yang mempengaruhi penggunaan *resource* pada *device* adalah :
   a. Jumlah boids yang digunakan. Pertumbuhan penggunaan resource pada pertambahan jumlah boids tidak linear
   b. Radius boids. Perubahan radius boids sangat berpengaruh pada neighbour detection serta pada rule separation, cohession dan alligment. Pertumbuhan penggunaan resource karena perubahan radius tidak linear.
   c. Sudut radius boids. Sudut radius boids sangat berpengaruh pada rule alligment, separation dan cohession namun tidak memberi pengaruh besar pada neighbour deetection.
2. Ukuran radius boids sangat berpengaruh pada perilaku kerumunan boids. Semakin besar radius boids, perilaku kerumunan lebih menyerupai kerumunan di alam bebas. Radius yang luas juga menjadikan flock lebih solid. Namun, penggunaan radius boids yang luas tidak disarankan apabila mempertimbangkan efisiensi penggunaan resource.
3. Sudut radius boids berpengaruh pada bentuk flock. Sudut radius tertentu akan membentuk flock tertentu.

Berikut adalah saran yang dapat diperhatikan untuk pengembangan Penelitian ini dikemudian hari :

1. Pada pengembangan selanjutnya, diharapkan mensimulasikan object 3D dan menggunakan animation, agar dapat dilihat pengaruh dan perilaku flock dalam menjalankan algoritma Craig Reynold
2. Percobaan dilakukan tanpa algoritma collusion avoidance. Diharapkan pada penelitian selanjutnya menerapkan algoritma collusion avoidance dan pengaruhnya pada penggunaan resource dan perilaku flock.

DAFTAR PUSTAKA

[1] E. Britannica, "Encyclopedia Britannica Online," Game, 2008. [Online]. Available: Http://Serch.Eb.Com/Eb/Article?Eu=36648. [Accessed 28 September 2016].

[2]"Video Game," Oxford Living Dictionary, [Online]. Available: Http://En.Oxforddictionaries.Com/Definition/Video_Game. [Accessed 28 September 2016].

[3]A. Elmenthaler, "Hagenuk Mt -2000 With Tetris," 17 June 2013. [Online]. Available: Handy-

Sammler.De. [Accessed 28 September 2016].

[4]"Oxford Living Dictionary," [Online]. Available: Http://En.Oxforddictionaries.Com /Definition/Crowd. [Accessed 28 September 2016].

[5] G. Pentheny, "Advanced Technique for Robust, Efficient Crowds," Game AI Pro 2 Collected Wisdom of Game AI Professionals, pp. 173-181, 2015

[6]C. W. Reynold, "Flocks, Herds and School: A Distributed Behavioral Model!," 1999.

[7] I. Millington, Artificial Intelligence For Games Seceond Edition, India: Elsevier, 2009.

[8]"Prime Megazine," 2009. [Online]. Available: http://www.primermagazine.com. [Accessed 29 September 2016].

[9] M. Dewi, "Simulation the Movement of the crowd in an environment using flocking," International Confrence on Instrumentation, comunication, information technology and Biomedical Engineering, no. 6108638, 2011.

[10]Williams and Sawyer. (2010). Using Information Techonolgy 9th Edition. McGraw-Hil.

[11]Samsung. (n.d.). Samsung. Retrieved October 02, 2016, from Samsung: http://www.samsung.com/id/smart phones/finder/

[12] A. Ismain, Education Games : Menjadi Cerdas dan Ceria dengan Permainan Edukatif, 2006: Pilar Media, Yogyakarta.

[13]D.O,"Swarmintelligence Fromnatural To Artifiaial System : Ant Colony Optimization," International Journal on Applications of Graph Theory in Wireless Ad hoc Networks and Sensor , vol. 8, no. 1, pp. 9-17, 2016.

[14]B. K. Panigrahi, Y. Shi, and M.-H. Lim (eds.): Handbook of Swarm Intelligence. Series: Adaptation, Learning, and Optimization

[15]M. Dorigo, E. Bonabeau, and G. Theraulaz, Ant algorithms and stigmergy, Future Gener. Comput. Syst., Vol. 16, No. 8, pp. 851–871, 2000

[16]J. Kennedy and R. C. Eberhart. Particle Swarm Optimization. In Proceedings of IEEE International Conference on Neural Networks, Perth, Australia, pp. 1942– 1948, 1995

[17] R. C. Eberhart and Y. Shi, Tracking and optimizing dynamic systems with particle swarms, Proc. Congress on Evolutionary Computation 2001, Seoul, Korea, 2001

[18]H. Ahmed, "Swarm Intelligence: Concepts, Models," School of Computing, Canada, 2012.

[19]A new Optimizer Using Particle Swarm Theory, Russell Eberhart

[20]Unity – Game Engine. [Online] Available at : http://www.unity3d.com [Accessed 5 Ocktober 2016].