PENERAPAN ALGORITMA *COLLISION DETECTION* DAN *BOIDS* PADA *GAME DOKKAEBI SHOOTER*

Lia Musfiroh*, Ahmad Jazuli, Anastasya Latubessy
Program Studi Teknik Informatika, Fakultas Teknik, Universitas Muria Kudus
Gondangmanis, PO Box 53, Bae, Kudus 59352
*Email: liafira72@gmail.com

Abstrak

Game berbasis android telah berkembang pesat, dengan semakin berkembangnya pengguna smartphone maupun tablet di dunia ini. Industri pembuat game pun mempunyai banyak peluang untuk memproduksi game. Banyak keuntungan dari memproduksi game karena banyaknya peminat dari game. Pembuatan game merupakan bidang yang menarik untuk dipelajari karena besarnya potensi baik dari segi ilmu pengetahuan maupun dari segi komersil. Dalam penelitian ini penulis merancang dan membangun aplikasi game Dokkaebi Shooter berbasis android dengan penerapan algoritma collision detection untuk deteksi serangan tabrakan dan algoritma boids untuk pergerakan musuhnya. Metode yang digunakan dalam penelitian ini adalah metode prototype, yaitu metode yang memberikan ide bagi system analyst atau pemrogram dalam menyajikan gambaran lengkap sistem dari identifikasi kebutuhan user (pemakai), mengembangkan prototype, menentukan prototype apakah prototype dapat diterima user dan penggunaan dari prototype. Hasil dari penelitian ini yaitu menghasilkan game mobile android berupa game single player tembakan vertikal monster dokkaebi yang dapat melatih ketangkasan pemain dengan tampilan 2D dan terdiri dari 3 level, yang dapat dijalankan pada smartphone maupun tablet android.

***Kata kunci** : android, boids, game, collision, shooting.*

1. PENDAHULUAN

Seiring berkembangnya kemajuan teknologi. *Game-game* menjadi sangat mudah didapatkan melalui internet, *game* menjadi pilihan utama untuk mengisi waktu senggang atau untuk sekedar melepas ketegangan setelah bekerja.

Salah satu *game* yang paling diminati adalah *game shooting*. *Game shooting* cara bermainnya adalah dengan menembak musuh (karakter *monster*) yang melewati layar yang biasanya menembak secara horizontal. *Game* ini banyak dipilih karena hanya mengandalkan ketangkasan untuk menyelesaikan *game* ini. Apalagi *game* yang sudah berbasis android yang memang di tujukan untuk *smartphone* maupun tablet yang sekarang banyak digunakan.

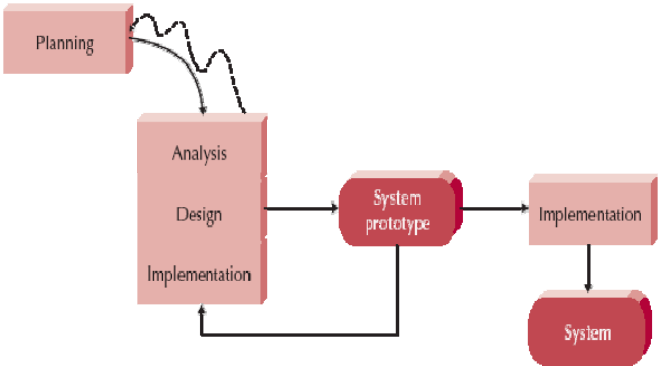
Android merupakan OS *Mobile* yang tumbuh di tengah OS lainnya yang berkembang dewasa ini.OS lainnya seperti *Windows Mobile, i-Phone OS, Symbian* dan masih banyak lagi juga menawarkan kekayaan isi dan keoptimalan berjalan di atas perangkat *hardware* yang ada (Purwanto, 2013).

Masih banyak *game* yang mempunyai keterbatasan seperti musuh muncul ditempat yang sama sehingga pemain dengan mudah dapat membunuh musuh dengan mengarahkan tembakan ketempat musuh tersebut, musuh hanya dapat menyerang pemain, tidak mempunyai kemampuan untuk menentukan pilihan berupa menyerang, bertahan atau melarikan diri ataupun efek suara yang tidak sesuai.

Dalam penelitian skripsi ini akan dibahas perancangan sebuah aplikasi *Game Shooting* berbasis *mobile android* yang bertujuan sebagai sarana melatih ketangkasan dengan judul "Penerapan Algoritma *Collision Detection* dan *Boids* pada *Game Dokkaebi Shooter*". Dalam penelitian ini, peneliti mencoba membuat *game* dengan dasar hiburan namun juga bertujuan sebagai sarana melatih ketangkasan dengan penerapan penerapan algoritma *boids* pada pergerakan musuhnya dan algoritma *collision detection* untuk deteksi tabrakan atau serangan musuh . *Game* ini merupakan tembakan secara vertikal dengan berfokus pada pengumpulan skor dan melawan musuh utama yaitu *dokkaebi. Dokkaebi* adalah hantu korea berwujud *monster.*

2. METODOLOGI

Untuk metode penelitian menggunakan metodologi *Prototype*. *Prototype* merupakan metodologi pengembangan *software* yang menitik-beratkan pada pendekatan aspek desain, fungsi dan *user-interface*.Sistem*prototyping*dapat melakukananalisis, desain, dan tahapan pelaksanaan secara bersamaan agar cepat mengembangkan versi sederhana dari sistem yang diusulkan oleh pengguna dan memberikannya kepada pengguna untuk evaluasi dan pembenahan (Dennis dkk., 2013). Diagram metode *prototype* dapat dilihat pada Gambar 1. Diagram Metode *Prototype.*



Gambar 1. Diagram Metode *Prototype* (Dennis dkk., 2013).

2.1 *Planning*

Langkah pertama adalah *planning*, pada tahap ini *developer* dan *client* akan bertemu, dimana *developer* adalah pembuat *game Dokkaebi Shooter* ini. Developer dan *client* bertemu untuk merencanakan tujuan, kebutuhan dan gambaran bagian-bagian yang di perlukan namun tidak secara detail. Selain itu pengumpulan data juga dilakukan pada tahap ini. Untuk memperoleh data tersebut secara lengkap dan akurat dibutuhkan kerjasama dengan pihak-pihak yang terkait, adapun langkah –langkah yang dilakukan untuk pengumpulan data adalah :

2.1.1 Studi Kepustakan

Metode ini dilakukan dengan cara memperoleh data dari buku-buku, literatur, majalah, jurnal tentang hal-hal pokok yang berhubungan dengan game yang akan dibuat atau permasalahan yang dihadapi dan nantinya dapat mendukung kelengkapan informasi yang dibutuhkan.

2.1.2 Wawancara

Metode ini dilakukan dengan cara bertanya langsung kepada pihak-pihak yang terkait sesuai dengan bidang dan ilmunya masing-masing, khususnya dalam bidang *game*. Dalam hal ini penulis mencoba berinteraksi melalui forum-forum *game*.

2.1.3 Observasi

Pengumpulan data dari pengamatan terhadap suatu proses atau objek dari *game* dengan maksud merasakan dan kemudian memahami pengetahuan dari sebuah gagasan yang sudah diketahui sebelumnya, untuk mendapatkan informasi-informasi yang dibutuhkan untuk melanjutkan penelitian.

2.2 *Analysis*

Pada tahap ini *developer* dan client akan bekerjasama untuk menganalisa kebutuhan, konsep, pengumpulan data, dari aplikasi yang akan dibuat secara detail.

2.3 *Design*

Pada tahap ini akan dilakukan perancangan dari *planning* dan *analysis* yang sudah didapatkan. Perancangan dibuat dan dilakukan dengan cepat, karena perancangan ini hanya sementara dan berfokus pada kebutuhan dasar *client* saja, sehingga perancangan ini menjadi dasar pembuatan *prototype*.

2.4 *System Prototype*

Pada tahap ini akan dilakukan proses kerja *prototyping* dari *design* yang telah dibuat.

2.5 *Implementation* pertama

Pada tahap ini ini dilakukan untuk mengetahui *prototype* yang dibagun oleh developer sudah sesuai dengan keinginan *client* atau belum. Jika *prototype* belum memenuhi keinginan *client* maka *prototype* akan direvisi dengan mengulang langkah 2, 3, dan 4. Dan jika *prototype* sudah sesuai dengan keinginan *client* maka dilanjutkan dengan langkah ke 5.

2.6 *Implementation* kedua

Pada tahap ini *prototype* yang sudah direvisi dan sudah lengkap maka dilakukan implementasi kedua untuk mengetahui kesempurnaan sistem tersebut.

2.7 *System*

Perangkat lunak yang telah melalui beberapa revisi dan diterima *client* akhirnya menjadi sebuah sistem dan siap untuk digunakan.

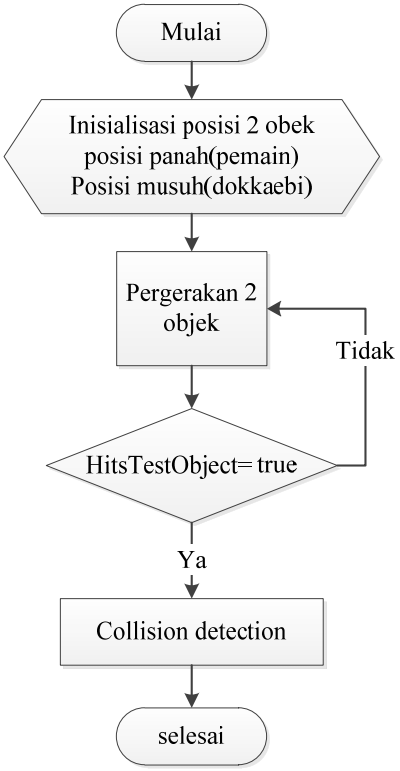
3. HASIL DAN PEMBAHASAN

3.1 Algoritma

3.1.1 *Collision Detection*

Algoritma *collision detection* adalah proses pengecekan apakah beberapa buah objek spasial saling bertumpuk atau tidak. Jika ternyata ada paling sedikit dua buah objek yang bertumpuk, maka kedua objek tersebut dikatakan saling bertumpukkan. Pada ruang spasial dua dimensi.Objek yang bertumpuk berarti objek spasialnya beririsan (Nugraha, 2013).

Penerapan algoritma *collision detection* pada *game dokkaebi shooter*, berikut Gambar 2 *Flowchart Collision Detection* Antara Panah dengan Musuh *(Dokkaebi)* yang menggambarkan alur dari algoritma *collision detection,* kemudian pengimplementasiannya ke dalam *source code* seperti yang terlihat pada Gambar 3. *Source Code* Implementasi Algoritma *Collision Detection* Antara *Dokkaebi* dan Panah *Player* dan implementasinya dalam antarmuka seperti Gambar 4. Implementasi Algoritma *Collision Detection* dalam Antarmuka Aplikasi.



Gambar 2. *Flowchart Collision Detection* Antara Panah dengan Musuh *(Dokkaebi).*

Implementasi ke dalam *source code:*

```
var flock:Flock = new Flock();
addChild(flock);

function shootArrow():void {
    // inisialisasi movieclip anak
    var arrows: arrow = new arrow();
    // posisi x panah di set sama seperti posisi x player/shaman.
    arrows.x = shaman_mc.x;
    arrows.y = shaman_mc.y -80;
    // display panah di stage
    addChild(arrows);
    //  listener untuk panah untuk memanggil moveArrow/pergerakan panah
    arrows.addEventListener(Event.ENTER_FRAME, moveArrow);
}
function moveArrow(e:Event):void {
    // event "target" akan menjadi movieclip panah
    e.target.y -= speed;
    // detect collision dengan hitTestObject
    if (e.target.hitTestObject(flock)) {
        dokkaebiIsHurt = true;
        removeChild(flock);
        score += 3;
        updateTextFields();
        trace(score);
        e.target.removeEventListener(Event.ENTER_FRAME, moveArrow);
        removeChild(MovieClip(e.target));
        }
```

Gambar 3. *Source Code* Implementasi Algoritma *Collision Detection* Antara *Dokkaebi* dan Panah *Player.*

Implementasi dalam antarmuka aplikasi :

Gambar 4. Implementasi Algoritma *Collision Detection* Dalam Antarmuka Aplikasi.

3.1.2 Boids

Algoritma *boids* adalah sebuah algoritma yang merepresentasikan gerak dari sebuah kawanan.Perilaku yang dihasilkan sangat mirip dengan kumpulan ikan atau kawanan burung (Saleh dkk., 2013).

Tiap *boids* memiliki set dari atribut berikut: posisi , kecepatan , up-vector , dan tiga *steering forces* (*separation, cohesion,* dan *alignment*).

3.1.2.1 Separation

Tiap anggota kelompok cenderung menghindari tubrukan dengan tetangganya.Tujuan ini disebut pemisahan (*separation)* atau *collision avoidance*.Ini mengarahkan *boids* untuk menghindari *overcrowding* pada *flockmate* lokal. Jika gaya untuk mengarahkan/mengemudikan ini diterapkan, maka *flock* akan menghilang. Solusi sederhana untuk gaya ini adalah berbelok pada saat bertemu dengan agen yang lainnya.

3.1.2.2 Cohesion

Cohesion atau pemusatan *flock*, merupakan gaya kemudi yang memindahkan *boids* menuju pusat *flock* yang terlihat. Tindakan ini melengkapi pemisahan / jika kemudi ini diaplikasikan, *flockmate*akan bersatu ke dalam posisi tunggal. *Cohesion* dari *boids* dinotasikan oleh dan dihitung dengan dua langkah.

Pertama, dihitung terlebih dahulu pusat dari set yang terlihat secara langsung. Pusat ini dinotasikan dengan dan terhubung ke pusat kepadatan dari semua *boids* yang terlihat

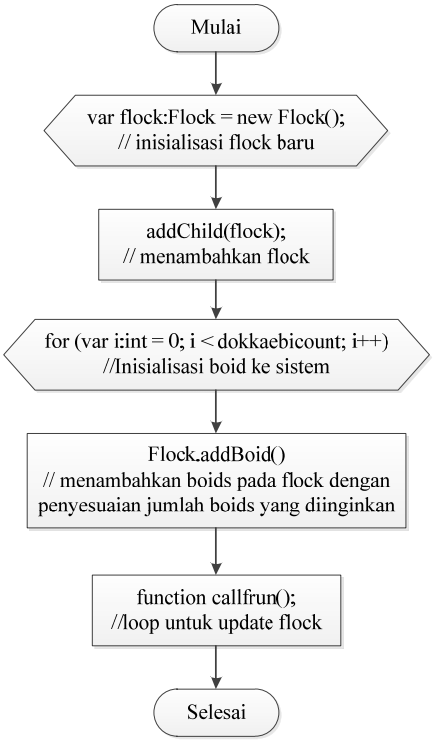
Boid dikemudikan ke arah tengah dari *visible flock* ( ) dengan tujuan untuk dihitung sebagai pemindahankohesi vektor .

3.1.2.3 Alignment

Steering force terakhir, *alignment* atau penyesuaian kecepatan (*velocity*), penting bagi *boids* untuk mengikuti gaya dorong. *Boids* menjaga agar tetap sejajar dengan kecepatan *flockmate*-nya.

Laju kecepatan adalah ukuran dari vektor. Oleh karena itu, *boids* akan mempercepat atau memperlambat secara otomatis tergantung *flockmate*-nya. Jika percepatan *boids* terlalu banyak maka akan memindahkan *visibility flockmate* dan keluar secepatnya (Nugroho dkk., 2010).

Boids untuk mengakumulasi percepatan berdasarkan 3 aturan *boids* yaitu, *separate, align* dan *cohesion* mulai dari inisialisasi *flock* baru, menambahkan *flock,* inisialisasi boids ke sistem, penambahan *boids* dengan penyesuaian jumlah *boids* yang diinginkan, *loop* untuk *update flock* sesuai inisialisasi *boids* ke sistem. Gambar 5. Flowchart Boids merupakan flowchart algoritma boids, pengimplementasiannya ke dalam source code seperti Gambar 6. Source Code Implentasi Algoritma Boids dan pengimplementasiannya dalam antarmuka aplikasi seperti pada Gambar 7. Implementasi Algoritma *Boids* Dalam Antarmuka Aplikasi.



Gambar 5. *Flowchart Boids*

Implementasi ke dalam *source code :*

```
62      public function flock(boids:Array)
63      {
64          var sep:Vector3D = separate(boids);// Separation
65          var ali:Vector3D = align(boids);// Alignment
66          var coh:Vector3D = cohesion(boids);// Cohesion
67          // Arbitrarily weight these forces
68          sep.ScalarMultiplication1(2.0);
69          ali.ScalarMultiplication1(1.0);
70          coh.ScalarMultiplication1(1.0);
71          // Add the force vectors to acceleration
72          acc.VectorAddition1(sep);
73          acc.VectorAddition1(ali);
74          acc.VectorAddition1(coh);
75
76      }
77      // Method to update location
78      public function update():void
79      {
80          // Update velocity
81          vel.VectorAddition1(acc);
82          // Limit speed
83          vel.Limit(maxspeed);
84          loc.VectorAddition1(vel);
85          // Reset accelertion to 0 each cycle
86          acc.setXYZ(0,0,0);
87      }
```

```
40  var flock:Flock = new Flock();
41  addChild(flock);
```

```
55  for (var i:int = 0; i < dokkaebicount; i++)
56  {
57      flock.addBoid(new Boid(new Vector3D(700/2,400/2,0.0),2.0,0.006));
58  }
59  function callfrun(e:Event)
60  {
61      flock.frun();
62  }
63  this.addEventListener(Event.ENTER_FRAME, callfrun);
64
```

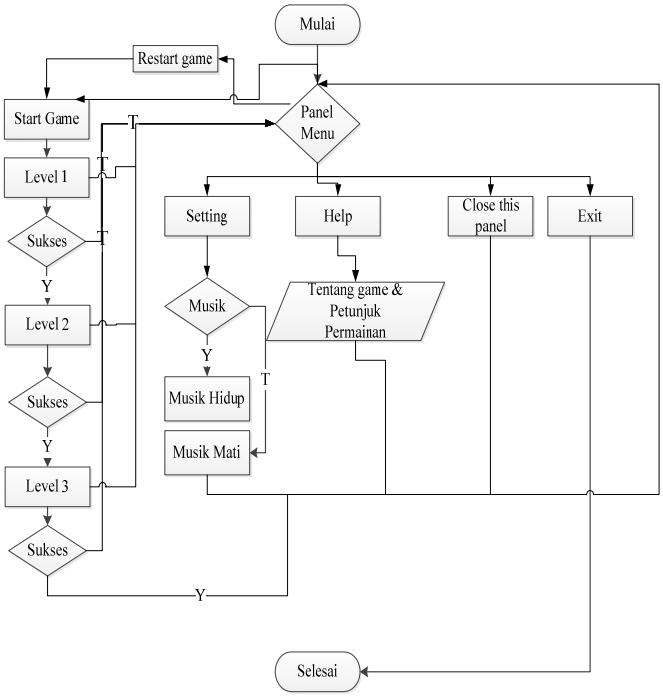
Gambar 6. *Source Code* Implentasi Algoritma *Boids.*

Implementasi ke dalam antarmuka:

Gambar 7. Implementasi Algoritma *Boids* Dalam Antarmuka Aplikasi

3.2 *Gameplay*

Gameplay disajikan untuk menggambarkan alur dalam permainan *game "Dokkaebi Shooter"* ini. Alur permainan *game* ini dapat dilihat pada gambar Gambar 8. *Flowchart Gameplay* pada *Game "Dokkaebi Shooter*



Gambar 8. *Flowchart Gameplay* pada *Game Dokkaebi Shooter.*

3.3 Hasil Pengujian Game Dokkaebi Shooter pada Operasi Sistem Android Gingerbread - KitKat.

Pengujian dilakukan pada 4 *smartphone* dengan versi *android* yang berbeda, dari versi *Gingerbread, Ice Cream Sandwich, Jelly Bean* hingga *KitKat*. Hasil pengujian dapat dilihat pada

Tabel 1. Hasil Pengujian pada Operasi Sistem *Android Gingerbread, Ice Cream Sandwich, Jelly Bean* dan *KitKat.*

Tabel 1. Hasil Pengujian pada Operasi Sistem *Android Gingerbread, Ice Cream Sandwich, Jelly Bean dan KitKat.*

No	Tipe *Smartphone*	Operasi *Sistem*	*RAM*	*Hasil*
1.	Samsung Galaxy Mini 2 S6500	*Android v2.3.6 Gingerbread*	512 MB	*Game* agak lambat jalannya ketika input nama dan menekan *key* menu pada halaman *high score* pada *game,* selain itu *game* berjalan dengan normal.
2.	Samsung Galaxy Chat GT-B5330	Android v4.0 Ice Cream Sandwich	512 MB	Game agak lambat jalannya ketika menekan key menu pada halaman high score pada game, selain itu game berjalan dengan normal.
3.	Smartfren Andromax C	Android v4.1.2 Jelly Bean	512 MB	Game berjalan dengan normal sepenuhnya.
4.	LG L40	Android v4.4.2 KitKat	512 MB	Game berjalan dengan normal sepenuhnya.

4. KESIMPULAN

Dari hasil analisa terhadap *game Dokkaebi Shooter* dapat disimpulkan bahwa kelebihan dari *game* ini yaitu *game* ini dapat mengimplementasikan *algoritma collision detection* dan *boids* dengan baik, *game* ini tidak hanya bergantung pada matinya musuh namun juga tergantung *timer* untuk memenangkan *game* ini, musuh dapat menyerang pemain, dan dapat menyimpan *score*. Kekurangan dari *game* ini yaitu tidak bisa dimainkan multi *player*, tidak bisa dimainkan dengan jaringan, kecepatan permainan *game* ini tergantung pada kapasitas RAM *smartphone* yang digunakan. Pengujian *game* dengan resolusi *smartphone* terbaik terdapat pada resolusi 600 x 1024 piksel dan dari pengujian versi *android* yang telah dilakukan pada *smarthphone* dengan versi *gingerbread, ice cream sandwich, jelly bean dan KitKat game* ini terbaik dimainkan pada *smartphone* yang mempunyai versi *android KitKat*.

DAFTAR PUSTAKA

Purwanto, (2013), *Analisis dan Pembuatan Game Tictac MatchBerbasis Android,* Tesis, Program Studi Teknik Informasi, STMIK AMIKOM YOGYAKARTA, Yogyakarta.

Dennis, A. dkk., (2013), Systems Analysis and Design with UML, 4th Edition,John Wiley and Sons, New York.

Nugraha, R.M., (2011), *Penggunaan Struktur Data Quad-Tree dalam Algoritma Collision Detection pa da Vertical Shooter Game,* Makalah IF3051 Strategi Algoritma, Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Bandung.

Saleh, A., dkk., (2011), *Model Penghindaran Tabrakan Multi Obyek Menggunakan Repulsive Field,* Jurnal Industrial Electronics Seminar, Surabaya, ISBN: 978-979-8689-14-7.

Nugroho, dkk., (2010), *Perilaku Agen Cerdas Berbasis BOIDS Untuk Simulasi Kerumunan Pada Keadaan Bahaya,* JAVA Journal of Electrical and Electronics Engineering, Vol. 8, No.1, ISSN: 1412-8306.