# Implementasi Algoritma Collision Detection Pada Perangkat Lunak Permainan Free Run

**Ari Bagus[*1], Listra Firgia[2]**
[1,2]Jurusan Teknik Informatika; STMIK Pontianak. Jl. Merdeka No.372 Pontianak, 0561-735555
E-mail: [*1]aribgs46@gmail.com, [2]Listra.firgia@stmikpontianak.ac.id

***Abstrak***

*Free run adalah sebuah aplikasi game yang di tunjukan untuk sistem operasi windows dengan genre action game, yaitu game dengan berfokus mengejar point, game free run ini menggunakan karakter robot sebagai karakter utama didalamnya pemain akan dibawa untuk mengendalikan robot untuk menghindari musuh seperti batu yang secara random menghadang. Dalam perjalanan permainan peneliti membangun dengan aplikasi unreal engine 4, game maker membangun game free run dengan sistem drop down dan dengan bahasa pemrograman game maker itu sendiri yaitu C++, dengan aplikasi ini juga dapat mengimplementasikan metode collision detection. Penelitian ini mengunakan metode Collision detection adalah metode yang digunakan dalam menentukan pergerakan dalam game seperti tubrukan suatu objek. Dengan menggunakan algoritma ini dapat menetukan arah serta tujuan dalam membuat game agar dapat mudah dinikmati, metode ini juga membuat dunia game semakin mirip dengan kenyataan di dunia nyata, dengan metode collision dapat membuat game free run menjadi nyata.*

*Kata Kunci - collision detection, game, berbasis komputer, free run, Game maker.*

***Abstract***

*Free run is a game application that is shown for the Windows operating system with the action game genre, which is a game that focuses on chasing points, this free run game uses robot characters as the main character in which players will be taken to control the robot to avoid enemies like random stones block it. In the course of the game, researchers build with the application of Unreal Engine 4, game makers build free run games with the drop down system and with the game maker programming language itself, namely C ++, this application can also implement the collision detection method. This research uses the method of Collision detection is a method used in determining movement in a game such as collision of an object. Using this algorithm can determine the direction and purpose in making the game so that it can be easily enjoyed, this method also makes the game world more similar to reality in the real world, with the collision method can make free run games become real.*

*Keywords - collision detection, game, computer based, free run, Game maker.*

## 1. PENDAHULUAN

Game komputer merupakan salah satu aplikasi software yang saat ini banyak dikembangkan. Dengan jenis yang bermacam-macam dan tampilan yang menarik, game komputer termasuk software yang diminati oleh berbagai kalangan. Selain karena tampilan dan aplikasinya relatif menarik, game komputer juga disinyalir dapat menjadi salah satu sarana refreshing yang cukup menyenangkan terutama bagi orang yang telah terbiasa menggunakan komputer.

Permainan-permainan komputer juga bermacam-macam. Salah satu kelebihannya adalah kita tidak harus mencari orang untuk menjadi lawan tanding jika ingin bermain karena permainan berbasis komputer ini sudah mendukung *single-player mode* dimana kita dapat bermain sendiri melawan komputer yang dirancang untuk dapat berlaku seperti pemain manusia atau yang sering dikenal dengan *artificial inteligince* (AI)[1].

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[2].

implementasi algoritma *collision detection* pada *game free run*, *game* yang berbasis system oprasi windows ini di buat dengan aplikasi *unreal engine 4*, dalam game *free run* ini pemain menggunakan karakter *robot* didalamnya pemain akan dibawa untuk mengendalikan *robot* untuk menghindari musuh seperti batu, setiap pemain mengambil koin tersebut pemain mendapatkan score. Permaianan ini menggunakan system gugur pada saat *robot* menabrak penghalang atau batu. Musuh akan di berposisi secara acak dan akan memberikan celah yang bisa di lewati *robot*, sedangkan permainan kembali ke awal saat menabrak batu, pemaian akan kembali ke start game dengan score yang telah di capai saat gugur dan terdapat penampilan score yang dicapai pemain. Metode *collision detection* pada *game* ini saat *robot* tersebut menabrak penghalang untuk medeteksi si *robot* dan batu tubrukan.

## 2. METODE PENELITIAN

Bentuk penelitian yang penulis gunakan dalam penelitian ini adalah studi literature, analisis, perancangan, pembangunan, dan pengujian[3]. Studi literatur merupakan strategi penelitian yang berusaha untuk memahami informasi pendukung yang ada dalam permainan komputer. Pada analisis akan diuraikan mengenai analisis permainan *free run* itu sendiri. Analisis yang dilakukan yaitu berupa analisis aturan permainan dan analisis algoritma *collision detection* pada permainan *free run*. Serta analisis kebutuhan sistem yang diperlukan adalah peletakan robot yang akan diletakan dan papan permainan yang digunakan untuk bermaik permainan *free run.* Dalam perancangan yang perlu di desain adalahproses permainan *free run* berupa *use case* dari sistem tersebut, desain skenario permaian *free run* dan desain tampilan permainan *free run*. Pada tahapan pembangunan atau pengembangan ini digunakan tahapan pengembangan rekayasa perangkat lunak, dalam tahapan pembangunan dikembangkan dengan menggunakan *C++*. Untuk menghasilkan suatu perangkat lunak yang bermutu, maka perlu dilakukan pengujian. Aplikasi permainan *free run* dengan menggunakan algoritma *collision detection* ini diuji dengan pengujian *Black-Box Testing* dan *White- Box Testing*.

## 3. HASIL DAN PEMBAHASAN

### 3.1 Business Modeling

Hasil penelitian menyimpulkan agar aplikasi game *free run* dapat berjalan dengan baik sesuai rancangan yang telah dibahas sebelumnya, hasil penelitian juga mengacu semua kegiatan dalam perancangan game *free run* dengan *unreal engine 4*. Pengujian aplikasi game *free run* dibuat untuk perangkat komputer bersistem operasi windows dengan resolusi yang direkomendasikan 1366x768 inc. Pada hasil ini peneliti akan menerangkan pengujian keseluruhan yang telah diimplementasikan sebelumnya, hasil pengujian aplikasi meliputi : pengujian, bentuk pengujian, masukan, keluaran yang diharapkan, hasil yang didapat, dan hasil pengujian

3.1.1 Game maker

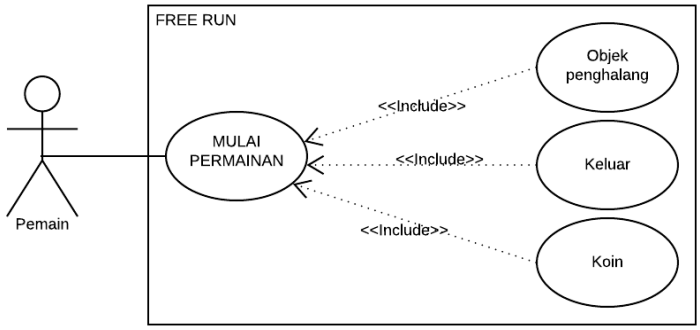
Aplikasi game ini dibangun menggunakan *unreal engine 4* yaitu aplikasi yang mudah membuat dan merancang suatu aplikasi seperti game dengan metode *BluePrint* serta terdapat berberapa language dari game maker itu sendiri, dalam pembuatan *game free run* dari awal.

3.1.2 Skenario dalam game *Free run*

Permainan akan dimulai saat *robot* di *spawn* pada titk yang telah di tentukan penulis dan berjalannya robot mulai menghindari, dalam game ini menggunakan sistem score, semakin score bertambah maka akan bertambah kecepatan robot, apabila robot user mengenai batu dan jatuh sehingga mengenai rumput, maka akan gugur dan mengulang. score akan ditampilkan saat game over sebelum mengulang pada awal game lagi.

3.1.3 Use Case Diagram

*Use case diagram* adalah diagram yang menyajikan interaksi antara *use case* dan *actor*. Dimana *actor* dapat berupa orang, peralatan atau system lain yang berinteraksi dengan system yang sedang dibangun. *Use case diagram* menggambarkan fungsionalitas system atau persyaratan-persyaratan yang harus dipenuhi system dari pandangan pemakai. Dapat dilihat pada gambar 4.2.




Gambar 3.2 Use Case Diagram

Tabel 3.1
Daftar Use Case Mulai Permainan

| | | |
|---|---|---|
| Use Case Name: | Menampilkan permainan baru | |
| Scenario: | Menampilkan medan pertempuran sebagai memulai permainan baru | |
| Brief Description: | Pengguna memulai level permainan | |
| Actor: | Pengguna | |
| Relate Usercase: | Tipe Permainan | |
| Stakeholder: | Pengguna | |
| Precondition: | - | |
| Postcondition: | Kata yang dipilih Memulai Permainan | |
| Flow of Events: | Actor | Sistem |
| | User memulai permainan baru | Memulai Permainan Baru |

Tabel 3.2
Berlari

| | |
|---|---|
| Use Case Name: | Berlari |
| Scenario: | User berlari sambal menghindari batu yang ada |

| | | |
|---|---|---|
| Brief Description: | Semua Koin dapat di ambil | |
| Actor: | User | |
| Relate Usercase: | - | |
| Stakeholder: | Pengguna | |
| Precondition: | - | |
| Postcondition: | Mulai permainan | |
| Flow of Events: | Actor | Sistem |
| | User dapat berlari | koordinat yang telah ditetapkan user |

Tabel 3.3
Tanding Ulang

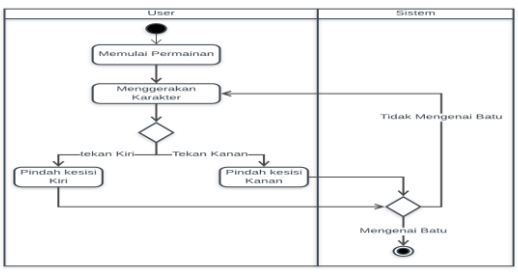
| | | |
|---|---|---|
| Use Case Name: | - | |
| Scenario: | User dapat bermain berulang-ulang kali tanpa harus keluar terlebih dahulu | |
| Brief Description: | - | |
| Actor: | Pengguna | |
| Relate Usercase: | - | |
| Stakeholder: | Pengguna | |
| Precondition: | - | |
| Postcondition: | Pemain menabrak batu | |
| Flow of Events: | Actor | Sistem |
| | User dapat mengulangi pertandingan | Mengulangi pertandingan tanpa harus menghapus skor |

Tabel 3.4
Keluar

| | | |
|---|---|---|
| Use Case Name: | Keluar | |
| Scenario: | Pemain dapat keluar dari permainan pada saat sedang bermain maupun kalah | |
| Brief Description: | User dapat keluar dari permainan | |
| Actor: | Pengguna | |
| Relate Usercase: | - | |
| Stakeholder: | Pengguna | |
| Precondition: | - | |
| Postcondition: | Saat bermain maupun mati | |
| Flow of Events: | Actor | Sistem |
| | User keluar dari permainan setelah menekaan tombol alt+f4 | Menutup aplikasi |

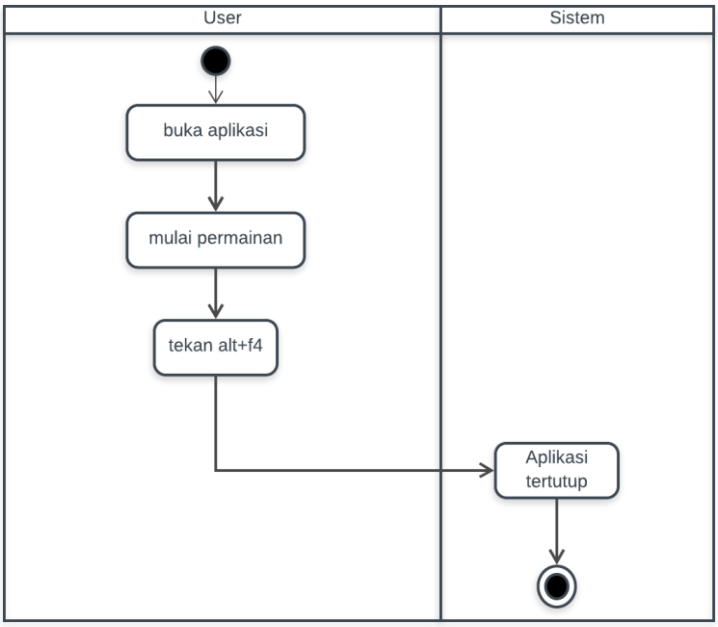
3.1.4 Activity Diagram

Activity Diagram menggambarkan alur kerja dari perancangan permainan *free run.* Gambar 4.2 Merupakan diagram aktivitas yang dilakukan *user* untuk memulai permainan.



Gambar 3.3 Activity Diagram Memulai Permainan Baru

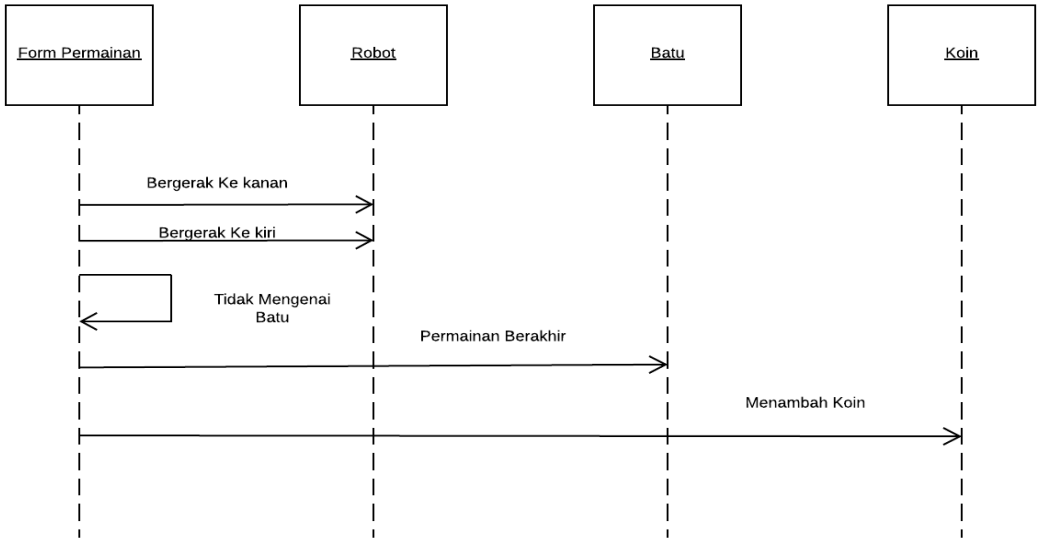
Gambar 3.3 Merupakan diagram aktivitas yang dilakukan user setelah menggunakan langkahnya.




Gambar 3.4 Activity Diagram Keluar

3.1.5 *Sequance Diagram*

*Sequance Diagram* menunjukan interaksi yang terjadi antara objek di dalam dan di sekitar (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. Interaksi *user* dan aplikasi pembentukan permainan *battleship* ini dapat diilustrasikan sebagai berikut



Gambar 3.5
Sequance diagram permainan *free run*

Pada gambar 3.5 terlihat ketika *user* menjalankan aplikasi ini, pertama *user* akan menentukan terlebih dahulu. Lalu *user* harus menentukan langkah yang telah pemain tentukan agar dapat memenangkan permainan. *Looping* akan terus terjadi sampai pemain menabrak batu.

## 3.2 Data Modeling

Penulis menggunakan informasi yang didapat dalam tahap diatas untuk menentukan banyaknya modul dan *form* yang akan digunakan dalam program tersebut. Jumlah komponen yang akan terdapat dalam setiap modul dan form akan ditentukan juga.

Perangkat lunak penerapan algoritma *collision detection* pada permainan *free run* dirancang dengan menggunakan bahasa pemrograman *C++* dan *bluePrint*. Perangkat lunak permainan ini memiliki beberapa form, seperti:

a. Form Main
b. Form Lorong

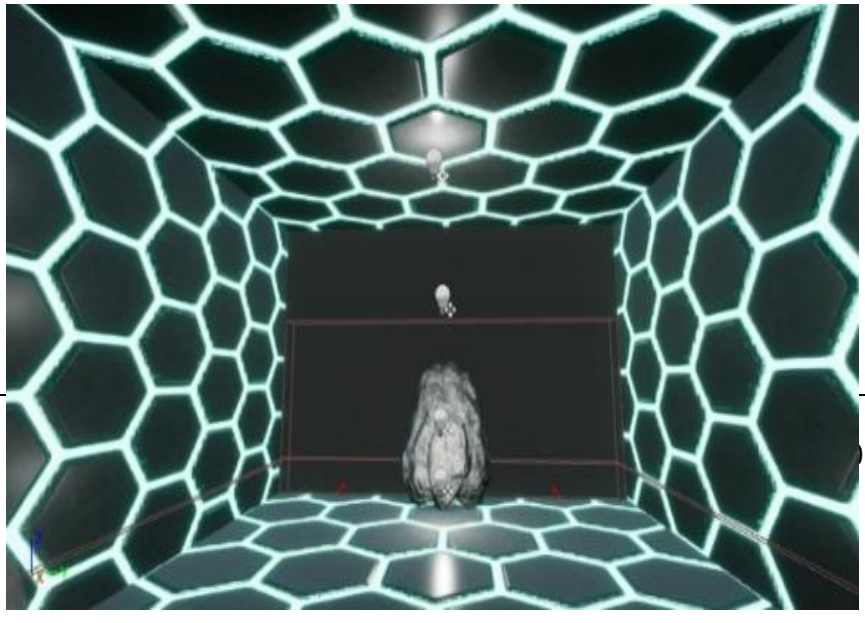
### 3.2.1 Form Splash Screen

Form Splash screen berfungsi sebagai form pembuka perangkat lunak Pada Form ini aplikasi dapat langsung dijalankan. Gambar 3.6 berikut merupakan desain form Splash screen.


Gambar 3.6 Rancangan Form Spash Screen

### 3.2.2 Form Lorong

Pada form ini pengguna perangkat lunak dapat bermain dalam permainan *free run*. karena permainan *free run* ini hanya memiliki 1 level saja,



Gambar 3.7 Rancangan Form Lorong

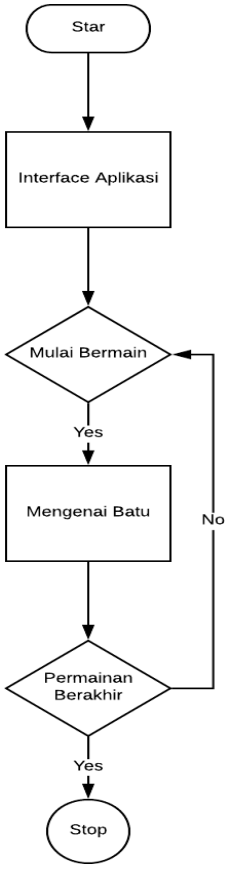
## 3.3 Process Modelling

*Form* dan *modul* yang sudah didefinisikan sebelumnya beserta komponennya disatukan untuk membentuk suatu program utuh. Hubungan antara modul dengan form juga didefinisikan oleh penulis.

Seperti yang telah dipaparkan pada bab sebelumnya, algoritma yang digunakan untuk membangun permainan ini adalah *collision detection* dengan menggunakan *tools unreal engine 4. Collision detection* adalah algoritma yang berbasis pada *bounding-box* untuk menetapkan jarak antara dengan *player* secara lebih optimal. Algoritma ini secara sistematis mencari solusi persoalan di antara semua kemungkinan solusi yang ada.

Di dalam algoritma *collision detection*, terdapat beberapa fungsi dan variabel yang digunakan di dalam pemrosesan solusi, yaitu ruang solusi, fungsi pembangkit, serta fungsi pembatas. Ruang solusi persoalan dinyatakan sebagai vektor dengan n-tuple. Fungsi pembangkit untuk membangkitkan nilai untuk nilai elemen dengan indeks tertentu yang merupakan komponen vektor solusi. Fungsi pembatas digunakan untuk menentukan apakah suatu simpul dapat mengarahkan ke solusi atau tidak.

Solusi dicari dengan cara Metode *Bounding-Box Collision Detection* hanya melakukan proses pendeteksian tabrakan Simpul-simpul yang sudah dibentuk dinamakan simpul hidup. Simpul hidup yang sedang dikembangkan dinamakan simpul-E. Setiap kali simpul-E dikembangkan, lintasan yang dibangun pun bertambah Panjang.



Gambar 3.8 Flowchart Aplikasi *Free run*

Untuk lebih jelasnya, proses aplikasi *battleship* akan dipaparkan sebagai berikut:

a. User menjalankan aplikasi permainan.
b. Selama permainan berlangsung komputer akan membangkitkan fungsi terminal test untuk mengecek apakah permainan telah berakhir atau belum. Apabila belum permainan akan terus berlanjut. Jika perrnainan telah berakhir maka user otomatis memulai permainan baru atau user dapat keluar dari permainan.

Setiap kali user ingin keluar dari permainan *free run*, user dapat menekan tombol alt+f4 untuk keluar dari permainan.

3.4 *Testing* dan *Turnover*

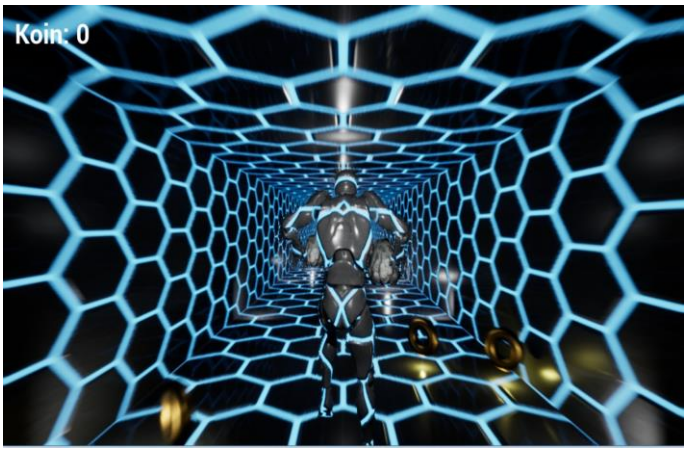
Setelah modul dirancang ke dalam program tersebut, penulis melakukan testing pada form yang membuat modul tersebut. Setelah setiap modul dan form terbentuk dan diuji, semua modul dan form tersebut kemudian disatukan dan dilakukan pengujian kembali akan integritasnya, termasuk didalamnya pengujian validitas input tiap form.

Setelah penulis melakukan perancangan untuk tampilan antar muka maka selanjutnya penulis melakukan pengujian permainan *free run* dengan algoritma *collision detection*. Tampilan awal program dimulai dengan tampilan Splash screen, setelah itu program akan menampilkan form Main. Pada pengujian kali ini, pemain mencoba bermain. Berikut ini merupakan gambar dari tampilan awal. Berikut adalah spesifikasi perangkat keras yang digunakan oleh penulis dalam melakukan tes pada *game free run.*

*a. CPU: AMD FX-6300*
*b. GPU: ZOTAC GeForce GTX 750 TI 2GB*
*c. RAM: 8GB*
*d. MOTHERBOARD: ASUS M5A78L*
*e. PSU: EVGA 500B*

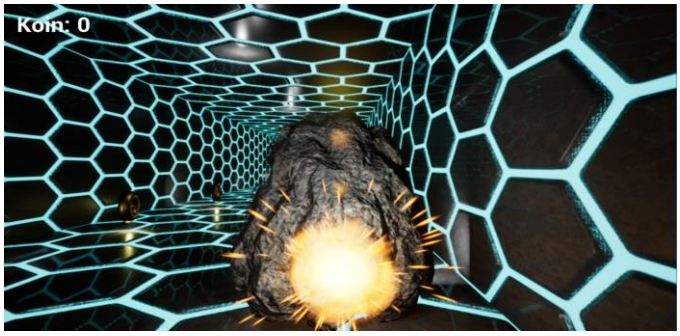

Gambar 3.9 Tampilan Awal Permainan

Gambar berikut menampilkan tampilan memulai permainan *free run* setelah memilih *play*.



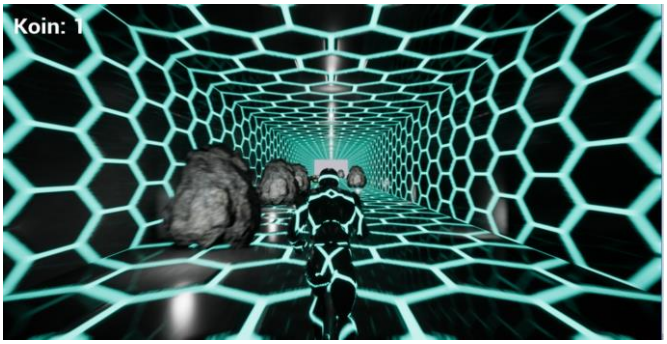

Gambar 3.10 Proses Permainan Sedang Berlangsung

Gambar berikut menampilkan tampilan pada saat player mulai mengumpulkan koin. Pada tahap ini karakter akan di munculkan di bagian belakang dari map/level, semua *collision* telah tergabung dengan material yang solid, sehingga objek-objek akan secara acak bermunculan di dalam arena.




Gambar 3.11 Kondisi Permainan Selesai

Gambar di atas menampilkan tampilan pada saat player menabrak batu dan mengenai *collision detection* pada batu, dan player akan otomatis mengulang permainan dari awal lagi, c*ollision detection* akan terjadi bila karakter yang di gunakan membentur objek batu, bila itu terjadi maka akan memunculkan suatu kondisi dimana karakter akan di hapus dari permainan dan akan kembali pada titik pertama permainan.




Gambar 3.12 Proses *Player* Mengumpulkan Koin

Pada saat mulai bermain player di haruskan mengumpulkan koin sebanyak banyaknya dan menghindari rintatangan berupa batu, bila karakter pemain mengenai objek berupa koin maka akan memasuki kondisi dimana karakter akan mendapatkan satu poin dari hasil tubrukan

antara karakter dan objek koin, bila tubrukan terjadi secara beruntun terhadap koin maka poin yang terletak pada bagian atas layer akan selalu bertambah jumlahnya seiring banyaknya koin yang di tabrak, dan karakter tidak akan memasuki mode dimana karakter akan mati.

## 4. KESIMPULAN

Setelah menyelesaikan perancangan perangkat lunak permainan *free run* dengan menggunakan algoritma *collision detection*, penulis menarik kesimpulan aplikasi ini sangat bermanfaat bagi user karena dapat mengasah otak user untuk menentukan jalur kiri atau kana yang tepat dan tidak sia-sia, algoritma *collision detection* dapat diterapkan di dalam aplikasi permainan *free run*, perangkat lunak hanya dapat dimainkan sendiri dengan satu computer dan dengan menggunakan algoritma *collision detection* untuk kecerdasan buatan dalam permainan *free run*, user (manusia) akan kesulitan untuk menang melawan kecerdasan buatan tersebut.

## 5. SARAN

Saran-saran yang dapat Penulis berikan dalam pengembangan program aplikasi game memberikan beberapa saran perangkat lunak dapat dikembangkan untuk *user* yang lebih banyak (lebih dari 1 orang), perangkat lunak dapat ditambahkan konsep *Artificial Intelligence* (AI) sehingga dapat dimainkan dengan computer, untuk menambahkan fitur-fitur yang lebih interaktif untuk pemain sehingga pemain tidak merasa kesulitan untuk memainkan permainan tersebut, memperindah tampilan grafik agar pemain tidak merasa bosan dengan tampilan permainan, sebaiknya aspek multimedia perlu lebih ditambahkan, misalnya suara untuk mendukung tampilan aplikasi dan menambahkan algoritma selain *collision detection* agar kemampuan kecerdasan buatanya lebih bertambah.

## UCAPAN TERIMA KASIH

Penulis ingin mengucapkan terima kasih kepada:
1. Bapak Sandy Kosasi, SE.,MM.,M.Kom. selaku Ketua Sekolah Tinggi Manajemen Informatika dan Komputer Pontianak.
2. Bapak Gat, S.Kom., M.Kom Selau ketua jurusan Teknik Informatika dan Pembimbing Jurnal.
3. Kedua Orang Tua, Keluarga dan adik-adik tercinta yang telah mendukung dan memberikan doa yang tulus dan dorongan semangat kepada penulis selama melakukan penulisan penelitian.

## DAFTAR PUSTAKA

[1]. Arsandi, A., SN, S. M., & Hariadi, M. (2012). VISUALISASI GERAKAN OBJEK 3D PADA AUGMENTED REALITY DENGAN DETEKSI TUMBUKAN BERBASIS BOUNDING BOX. Pasca Sarjana Jaringan Cerdas Multimedia (Game Teknologi) Teknik Elektro,Teknologi Industri ITS, 1-10.

[2]. Ganni, E., & Tendean, H. (2007). PERANCANGAN GAME LIVE FOR GYM. JURNAL ILMU KOMPUTER DAN SISTEM INFORMASI, 1-11.

[3] David, David. "Perancangan Game Mobile Android Bergenre Horror." *Cogito Smart Journal*, vol. 2, no. 2, 2016, pp. 167-179, doi:10.31154/cogito.v2i2.27.167-179.

[4]. Handoyo, E. D. (2010). Pembuatan Permainan Super Noseman. 1-11.

[5]. Haryono, A. N., Hendro, S., & Setiawan. (2010). penggunaan struktur data Quad Tree dalam Algoritma Collision Detection pada Vertical Shooter Game penggunaan struktur

data Quad-Tree dalam Algoritma Collision Detection pada Vertical Shooter Game penggunaan struktur data Quad-Tree dalam Algoritma Collisio. JURNAL INFORMATIKA , VOLUME 6 NO 1 TAHUN 2010, 1-10.

[6]. Arsandi, A., Mardi, S., & Hariadi, M. (t.t). Visualisasi Gerakan Objek 3D pada Augmented Reality dengan Deteksi Tumbukan Berbasis Bounding Box.

[7] Musfiroh, L., Jazuli, A., & Latubessy, A. (2014). PENERAPAN ALGORITMA COLLISION DETECTION DAN BOIDS PADA GAME DOKKAEBI SHOOTER. Prosiding SNATIF Ke-1 Tahun 2014 ISBN: 978-602-1180 04-4, 1-8.

[8] Putrady, E. (2011). Optimasi Collision Detection Menggunakan Quadtree. Makalah IF3051 Strategi Algoritma – Sem. I Tahun 2010/2011, 1-5.

[9] Sibero, I. C. (2010). Membuat Game 2D Menggunakan Game Maker. Yogyakarta: MediaKom.

[10] Susilawati. (2014). PERANCANGAN GAME SPACE SHIP DENGAN METODE QUAD TREE. Pelita Informatika Budi Darma, Volume : ViI, Nomor: 3, Agustus 2014 ISSN : 2301-9425, 1-5.