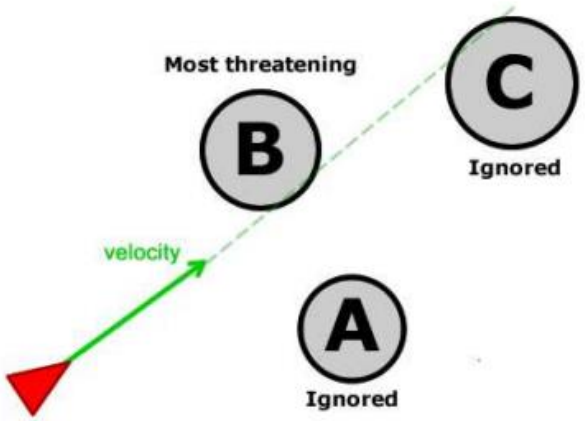
BAB II

LANDASAN TEORI

2.1. Algoritma *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, seperti terlihat pada gambar 2.1 (Winarno & Setiyawan, 2018).



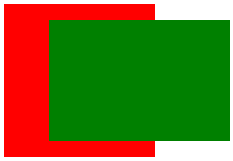
Gambar 2.1 Cara Kerja *Collision Avoidance System* (Bevilacqua, Huang, Figueroa, & Premaratne, 2013)

Pada Gambar 2.1 Cara *Kerja Collision Avoidance System* menjelaskan bahwa musuh yang paling dekat dengan karakter pemain akan melakukan analisa menggunakan *collision detection* yang telah diterapkan sebelumnya terhadap *NPC* untuk melakukan pengecekan keberadaan pemain dan rintangan yang ada

didekatnya. Kunci dari *collision detection* ini ialah kemampuan pengecekan atau deteksi terhadap satu objek yang bertabrakan dengan objek lainnya (Bevilacqua, Huang, Figueroa, & Premaratne, 2013).

1. Pseudocode Collision Detection

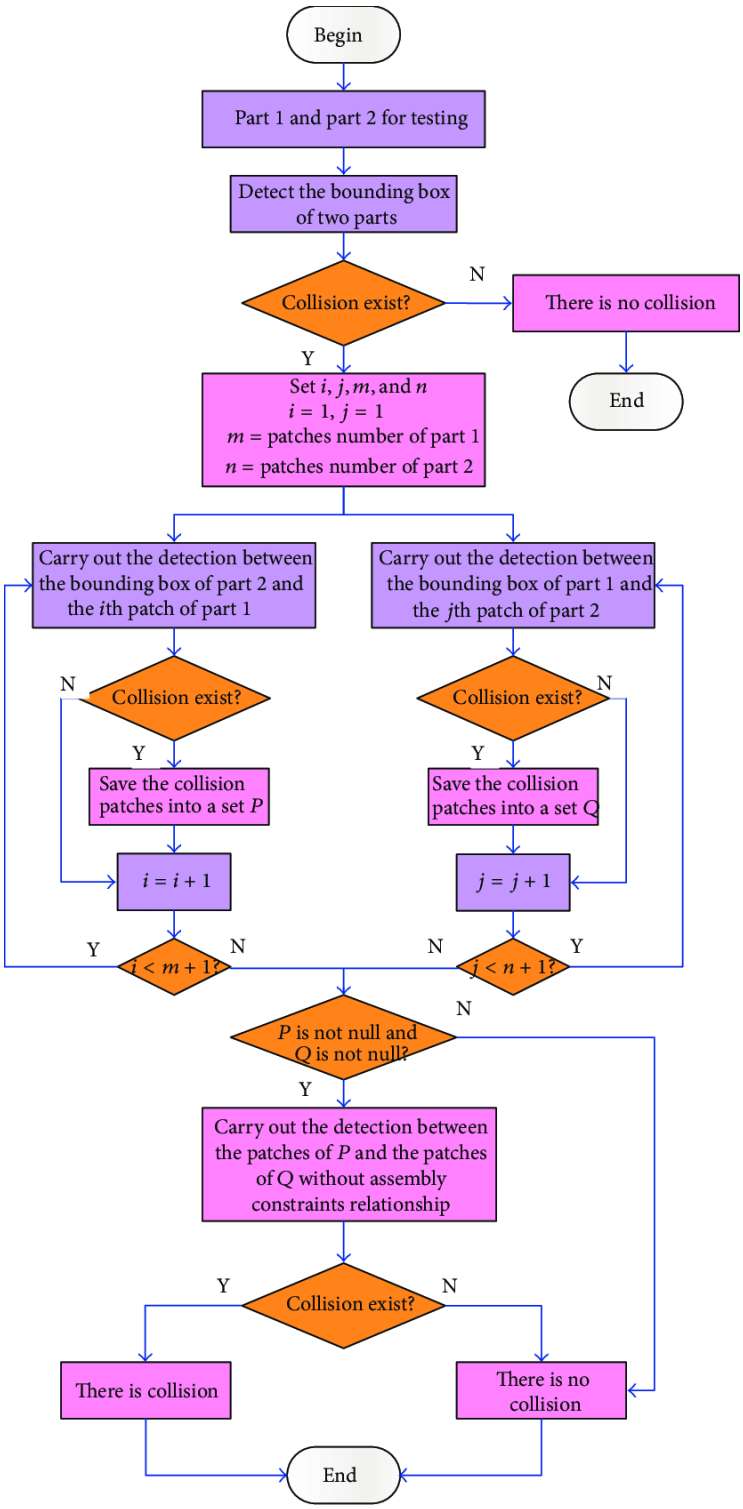
```
var rect1 = {x: 5, y: 5, width: 50, height: 50}
var rect2 = {x: 20, y: 10, width: 10, height: 10}


      if (rect1.x < rect2.x + rect2.width &&
        rect1.x + rect1.width > rect2.x &&
        rect1.y < rect2.y + rect2.height &&
        rect1.y + rect1.height > rect2.y) {
         // collision detected!
      }


      // filling in the values =>


      if (5 < 30 &&
         55 > 20 &&
         5 < 20 &&
         55 > 10) {
         // collision detected!
}
```



Gambar 2.2 Hasil *Pseudocode Collision Detection* (Bevilacqua, Huang, Figueroa, & Premaratne, 2013)

2. *Flowchart Collision Detection*

Algoritma *collision detection* adalah proses pengecekan apakah beberapa buah obyek spasial saling bertumpuk atau tidak. Jika ternyata ada paling sedikit dua buah obyek yang bertumpuk dapat di lihat pada *gambar 2.2 Flowchart Collision Detection*, maka kedua obyek tersebut dikatakan saling bertumpukan (Nurdiyanto & Winarno, 2018).



Gambar 2.3 *Flowchart Collision Detection* (Liu, et al., 2013)

2.2. Algoritma *Random Number Generator*

Random Number Generator (*RNG*) adalah sebuah program atau alat untuk menghasilkan urutan angka atau simbol secara tidak teratur. Sistem ini diaplikasikan ke dalam banyak bidang, seperti sampel statistika, simulasi komputer, kriptografi, bahkan untuk desain.

2.2.1 Metode

a) Metode Fisik

Metode paling pertama untuk menghasilkan angka secara acak adalah dengan menggunakan dadu, koin, rolet, dan lain sebagainya. Sampai saat ini, metode ini masih cukup sering digunakan, terutama di dalam game dan perjudian. Karena metode ini dianggap terlalu lambat, pengaplikasiannya untuk statistika dan kriptografi kurang begitu popular saat ini. Dasar dari metode fisik adalah fenomena fisika atomik atau subatomik acak yang tidak bisa diprediksi dapat dilacak dengan menggunakan mekanika kuantum. (Fadila, 2014)

b) Metode Distribusi Probabilitas

Metode ini menggunakan fungsi densitas probabilitas. Metode ini bekerja cukup baik untuk menghasilkan pseudo-random dan true random number. Salah satu metode, yaitu metode inverse, mengintegralkan area lebih dari sama dengan bilangan acak. Metode kedua, acceptance-rejection, memilih antara nilai x dan y, lalu membandingkan apakah fungsi x lebih besar dari nilai y. Apabila fungsi x lebih dari nilai y, maka nilai x akan

diterima. Jika sebaliknya, maka nilai x akan ditolak dan algoritmanya akan mencoba ulang. (Fadila, 2014)

c) Metode Komputasi

Metode ini menggunakan algoritma bernama Pseudo-random number generator yang secara otomatis menghasilkan serangkaian angka acak yang memiliki kualitas baik. Nilai yang dihasilkan oleh algoritma tersebut secara umum ditentukan dengan sebuah konstanta yang disebut seed. Salah satu PRNG yang umum adalah *linear congruential generator*, yang menggunakan rekurens dari persamaan. (Fadila, 2014)

$$X_{n+1} = (aX_n + b) \bmod m$$

untuk menghindari sifat non-acak yang muncul dari linear congruential generator, beberapa random number generator dengan koefisin nilai pengali yang berbeda-beda dapat digunakan secara paralel.

Beberapa bahasa pemrograman memiliki fungsi yang bersifat *random number generator*. Fungsi-fungsi ini biasanya digunkan untuk menghasilkan angka, kata, atau bilangan real yang tersebar diantara 0 dan 1. Fungsi-fungsi tersebut biasanya memiliki sifat statistika yang buruk. Biasanya fungsi-fungsi tersebut diinisialisasi menggunakan real time clock sebagai seed menyebabkan perhitungan yang dilakukan di dalam millisecond dan sangat jauh jika dibandingkan dengan presisi manusia. (Fadila, 2014)

Fungsi-fungsi tersebut memberikan hasil yang cukup untuk beberapa tugas (contohnya video game), tetapi tidak cocok digunakan saat tingkat acak yang dibutuhkan sangat tinggi, seperti aplikasi untuk kriptografi dan analisis numerik dalam statistik.

Salah satu contoh sederhana *pseudo-random number generator* adalah metode *Multiplywith-carry* yang ditemukan oleh *George Marsaglia*. Program ini memiliki kecepatan dan sifat acak yang cukup baik. (Fadila, 2014)

```
m_w = <choose-initializer>;    /* must not be zero */ m_z = <choose-initializer>;    /* must not be zero */    uint get_random() {      m_z = 36969 * (m_z & 65535) + (m_z >> 16);  m_w = 18000 * (m_w & 65535) + (m_w >> 16);      return (m_z << 16) + m_w;  /* 32bit result */ }
```

Contoh fungsi sederhana dalam Bahasa C :

```
int rand()  { random_seed = random_seed * 1103515245 +12345;      return (unsigned int)(random_seed / 65536) % 32768;}
```

2.3. *Artificial Intelliigence*

Ada beberapa konsep yang harus dipahami dalam kecerdasan buatan, diantaranya (Kusrini, 2006) :

1. *Turing Test* – Metode Pengujian Kecerdasan

   Turing Test merupakan sebuah metode pengujian kecerdasan yang dibuat oleh Alan Turing. Proses uji ini melibatkan seorang penanya (manusia) dan dua obyek yang ditanyai. Yang satu adalah seorang manusia dan yang satunya adalah sebuah mesin yang akan diuji. Penanya tidak dapat melihat langsung

kepada obyek yang ditanyai. Penanya diminta untuk membedakan mana jawaban komputer dan mana jawaban manusia berdasarkan jawaban kedua obyek tersebut. Jika penanya tidak dapat membedakan mana jawaban mesin dan mana jawaban manusia maka Turing berpendapat bahwa mesin yang diuji tersebut dapat diasumsikan cerdas.

2. Pemrosesan Simbolik

   Komputer semula didesain untuk memproses bilangan atau angkaangka (pemrosesan numerik). Sementara manusia dalam berfikir dan menyelesaikan masalah lebih bersifat simbolik, tidak didasarkan pada sejumlah rumus atau melakukan komputasi matematika. Sifat penting dari AI adalah bahwa AI merupakan bagian dari ilmu komputer yang melakukan proses secara simbolik dan non-algoritmik dalam penyelesaian masalah.

3. Heuristic

   Istilah Heuristic diambil dari bahasa Yunani yang berarti menemukan. Heuristic merupakan suatu strategi untuk melakukan proses pencarian (search) ruang problem secara selektif, yang memandu proses pencarian yang dilakukan disepanjang jalur yang memiliki kemungkinan sukses paling besar.

4. Penarikan Kesimpulan (*Inferencing*)

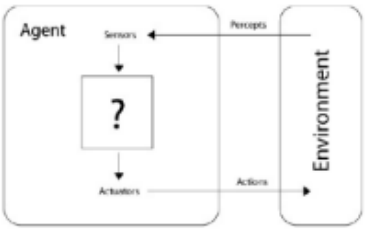
   AI mencoba membuat mesin memiliki kemampuan berfikir atau mempertimbangkan (reasoning). Kemampuan berfikir (reasoning) termasuk didalamnya proses penarikan kesimpulan (inferencing) berdasarkan fakta-fakta dan aturan dengan menggunakan metode heuristic atau pencarian lainnya.

5. Pencocokan Pola (*Pattern Matching*)

   AI bekerja dengan metode pencocokan pola (pattern matching) yang berusaha untuk menjelaskan objek, kejadian (Event) atau proses, dalam hubungan logika atau komputasional.

2.4. *Intelligent Agent* / Agen Cerdas

Permainan atau aplikasi yang dirancang menggunakan agen cerdas sebagai otak untuk melawan player. Agen adalah sesuatu yang dapat mengesankan lingungannya melalui sensors dan mengambil tindakan terhadap lingunganya melalui actuators. (Putri & Prathivi, 2016) dapat di lihat pada Gambar 2.4 Agen yang berinteraksi dengan lingunganya melalui sensor dan actuator.



Gambar 2.4 Agen yang berinteraksi dengan lingunganya melalui sensor dan actuator (Putri & Prathivi, 2016)

Definisi agen rasional adalah untuk setiap deretan persepsi yang mungkin, sebuah agen rasional hendaklah memilih satu tindakan yang diharapkan memaksimalkan ukuran *performance*-nya dengan adanya bukti yang diberikan oleh deretan presepsi apapun pengetahuan terspasang yang dimiliki agen itu. Empat agen dasar yaitu *simple reflex agents, model based refle agent, goal-based agents dan unity-based agents.* (Stuart & Norvig, 2005)

2.5. *Non-player Character*

Autonomous character adalah jenis otonomous agent yang ditujukan untuk penggunaan computer animasi dan media interaktif seperti games dan *virtual reality*. Agen ini mewakili tokoh dalam cerita atau permainan dan memiliki kemampuan untuk improvisasi tindakan mereka. Ini adalah kebalikan dari seorang tokoh dalam sebuah film animasi, yang tindakannya ditulis di muka, dan untuk "*Avatar*" dalam sebuah permainan atau *virtual reality*. Tindakah yang diarahkan secara *realtime* oleh pemain. Dalam permainan karakter otonom biasanya disebut dengan *NPC (Non Player Character).* (Arif, Wicaksono, & Fachrul, 2012)

Perilaku karakter yang otonom dapat lebih baik dipahami dengan membaginya menjadi beberapa lapisan. Lapisan ini dimaksud untuk kejelasan dan kekhusussan dalam diskusi. *NPC* terbagi menjadi 3 divisi gerak prilaku otonom hirarki karakter menjadi tiga lapisan : Seleksi tindakan, steering dan penggerak. (Arif, Wicaksono, & Fachrul, 2012)

2.6. *Game*

Definisi *game* menurut John C Beck dan Mitcell Wade (2014) adalah penarik perhatian yang telah terbukti. *Game* adalah lingkungan pelatihan yang baik bagi dunia nyata dalam organisasi yang menuntut pemecahan masalah secara kolaborasi. Menurut *Kimpraswill* (dalam As'adi Muhammad, 2009: 26) mengatakan bahwa definisi permainan atau *game* adalah usaha diri (olah pikiran dan olah fisik) yang sangat bermanfaat bagi peningkatan dan pengembanganmotivasi, kinerja, dan prestasi dalam melaksanakan tugas dan kepentingan organisasi yang lebih baik. *Mobile game* merupakan permainan yang menggunakan media elektronik yang

dapat menjadi sebuah sarana hiburan berbetuk multimedia. Berikut jenis – jenis game diantaranya:

a) Shooter

Shooter adalah jenis game yang di mainkan menggunakan sbuah senjata, biasanya pistol, senapan, atau senjata jarak jauh lainnya. Biasanya tujuan dari permainan ini adalah menebak lawan dan memenangkan misi tanpa harus gugur di medan perang.

b) *Strategy Games*

Strategy adalah jenis game yang mengharuskan pemianya menggunakan taktik dan strategi untuk jeli dalam melihat setiap peluang, kelemahan musuh dan kebijakan dalam menggunakan sumber daya yang ada untuk mengatur suatu unit atau pasukan untuk menyerang markas musuh dalam rangka memenangkan permainan. Biasanya di dalam game Strategy dituntut untuk mencari uang, emas, poin atau semua yang berfungsi untuk membiayai masukan.

c) *Racing Games*

Racing adalah game yang di mainkan dengan mengendalikan sebuah kendaraam untuk memenangkan sebuah balapan atau garis finish dari suatu *race*, dalam game ini biasanya pemain dapat memiliki & membeli serta dapat mengupgrade kendaraan.

d) *Adventure Games*

Adventure adalah jenis game yang umumnya membuat pemain harus berjalan mengelilingi suatu tempat yang telah di desain sedemikian rupa, seperti

sebuah istana, gua yang berkelok, dan planet yang jauh. Pemain melakukan navigasi suatu area, menncari pesan – pesan rahasia, memperoleh objek yang memiliki kemampuan yang bervariasi, bertempuh dengan msuh, dan lain-lain.

e) Simulation

Simulation adalah jenis game yang memberikan pengalaman atau interaksi semirip mungkin dengan kendaraan yang aslinya, meskipun terkadang kendaraan tesebut masih eksperimen realistic menggunakan kendaraan tersebut.

f) Platform

Platform adalah jenis game yang mengharuskan pemain mengarahkan suatu objek dengan melalui berbagai tahap atau tingkatan area untuk menerang musuh dan menghindari terhadap serangan. Jenis game ini sedikit serupa dengan action game, tetapi aksinya tidak secepat *action game*. Teknik *collision detection* sangat sering dimanfaatkan pada jenis game ini.

g) *Action Game*

Action adalah jenis game dengan fitur utama berupa banyaknya aksi dimana pemain harus memiliki keterampilan reaksi yang cepat untuk menghindari musuh atau menghindari rintangan.

2.7. Animasi

Animasi berasal dari kata Yunani yaitu anima yang berarti jiwa, hidup, nyawa, semangat. Animasi juga dapat didefinisikan sebagai proses perubahan bentuk property objek yang ditampilkan dalam suatu pergerakan tansisi dalam suatu kurun waktu, sedangkan menurut *Tay Vaughan* (2011) animasi adalah perubahan

visual sepanjang waktu dan memberikan kekuatan besar pada proyek multimedia atau halaman web.

2.8. *Unity 3D Game Engine*

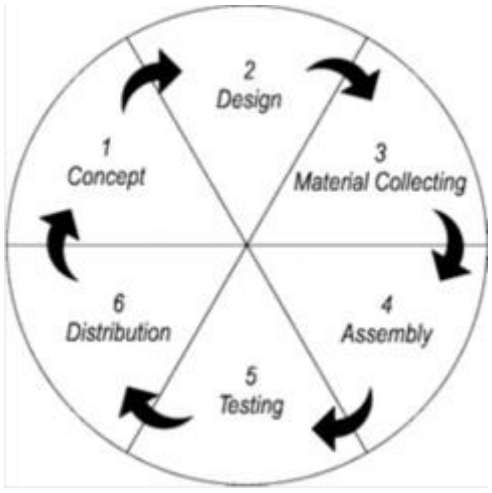
Unity Game Engine adalah *software* atau *Game engine* yang digunakan untuk membuat video Game berbasis dua atau tiga dimensi dan dapat digunakan secara gratis, selain untuk membuat Game, Unity 3D juga dapat digunakan untuk membuat konten yang interaktif lainnya seperti, *visual arsitektur* dan *real-time 3D* animasi, selain sebagai Game engine Unity 3D juga dapat digunakan sebagai sebuah editor bagi Game yang sudah ada. (Ekasari, 2012)

Unity 3D dibuat dengan menggunakan bahasa perogram C++, Unity 3D mendukung bahasa program lain seperti *JavaScript*, C#, dan Boo, Unity memiliki kemiripan dengan Game engine lainnya seperti, Blender Game engine, Virtools, Gamestudio, adapaun kelebihan dari Unity 3D, Unity dapat dioperasikan pada platform Windows dan Mac Os dan dapat menghasilkan Game untuk Windows, Mac, Linux, Wii, iPad, iPhone, google Android dan juga browser. Game Unity 3D juga mendukung dalam pembuatan Game untuk *console Game Xbox 360* dan *PlayStation* (Ekasari, 2012)

2.9. Metode MDLC (*Multimedia Development Life Cycle*)

Menurut Riyanto & Singgih, (2015), MDLC (*Multimedia Development Life Cycle*) merupakan metode pengembangan system yang cocok untuk pengembangan system berbasis multimedia. *Multimedia Development Life Cycle* terdiri dari enam tahap, yaitu tahap pengonsepan (*concept*), perancangan (*design*), pengumpulan

bahan (*material collecting*), pembuatan (*assembly*), pengujian (*testing*) dan pendistribusian (*distribution*) seperti terlihat pada gambar 2.5. Langkah-langkah pokok penelitian dan pengembangan *MDLC* yang membedakannya dengan pendekatan penelitian lain adalah :



Gambar 2.5 *Multimedia Development Life Cycle*

1.) Concept

Tahap untuk menentukan tujuan dan siapa pengguna program. Tahap pengonsepan (*Concept*) adalah tahap untuk menentukan tujuan dan kepada siapa *multimedia* di tujukan (audiens identification). Selain itu menentukan jenis aplikasi (presentasi, interaktif, dan lain-lain) dan tujuan aplikasi (hiburan, pembelajaran, dan lain-lain). Dasar aturan untuk perancangan juga ditentukan pada tahap ini misalnya ukuran, target. Output dari tahap ini biasanya berupa dokumen yang bersifat naratif untuk mengungkapkan tujuan proyek yang ingin di capai.

2.) Design

Tahap pembuatan spesifikasi mengenai arsitektur proyek, gaya, tampilan, dan kebutuhan material/bahan untuk proyek. Perancangan (design) adalah tahap pembuatan spesifikasi meliputi arsitektur proyek, gaya, tampilan dan kebutuhan material atau bahan untuk program. Spesifikasi dibuat serinci mungkin sehingga pada tahap berikutnya yaitu material collecting dan assembly, pengambilan keputusan baru tidak diperlukan lagi, cukup ini biasanya menggunakan storyboard untuk menggambarkan deskripsi tiap scene dengan mencantumkan semua obyek multimedia.

3.) *Material Collecting*

Tahap pengumpulan bahan yang sesuai dengan kebutuhan yang dikerjakan. Pengumpulan materi adalah tahap pengumpulan bahan yang sesuai dengan kebutuhan yang dikerjakan. Bahan-bahan tersebut antara lain seperti clip-art, graphic, animasi, video, audito. Tahap ini dapat dikerjakan secara parallel dengan tahap assembly. Namun dapat juga tahap material collecting dan tahap assembly akan dikerjakan secara linear dan tidak parallel.

4.) *Assembly*

Tahap assembly adalah tahap pembuatan semua obyek atau bahan multimedia dibuat. Pembuatan proyek didasarkan pada tahap design. seperti storyboard, bagan alir atau struktur navigasi.

5.) *Testing*

Tahap Pengujian dilakukan setelah selesai tahap pembuatan (*assembly*) dengan menjalankan proyek apakah ada kesalahan atau tidak. Tahap ini disebut sebagai tahap pengujian alpha (alpha test) dimana pengujian

dilakukan oleh pembuat, Fungsi dari tahap ini adalah melihat hasil pembuatan proyek apakah sesuai dengan yang diharapkan atau tidak, maka akan dibuat tabel pengujian untuk menguji kriteria proyek tersebut (Irawan, Laurin, & Suherman, 2015).

6.) Distribution

Proyek akan disimpan dalam sebuah media penyimpanan. Pada tahap ini proyek akan disimpan dalam suatu media penyimpanan. Jika media penyimpanan tidak cukup menampung proyeknya makan kompresi terhadap proyek itu akan dilakukan. Tahap ini juga dapat disebut sebagai tahap evaluasi untuk pengembangan produk yang sudah jadi supaya menjadi lebih baik. Hasil evaluasi ini dapat digunakan sebagai masukan untuk tahap concept pada produk selanjutnya

2.10. *Black-Box Testing*

Menurut Muhamad Irwan (2013) *Black Box Testing* berfokus pada spesifikasi fungsional dari perangkat lunak. *Tester* dapat mendefinisikan kumpulan kondisi input dan melakukan pengetesan pada spesifikasi fungsional program. *Black Box* Testing bukanlah solusi alternatif dari *White Box Testing* tapi lebih merupakan pelengkap untuk menguji hal-hal yang tidak dicakup oleh *White Box Testing*. *Black Box Testing* cenderung untuk menemukan hal-hal berikut:

1. Fungsi yang tidak benar atau tidak ada.
2. Kesalahan antarmuka (*interface errors*).
3. Kesalahan pada struktur data dan akses basis data.

4. Kesalahan performansi (*performance errors*).
5. Kesalahan inisialisasi dan terminasi.

Pengujian didesain untuk menjawab pertanyaanpertanyaan berikut:

1. Bagaimana fungsi-fungsi diuji agar dapat dinyatakan valid?
2. Input seperti apa yang dapat menjadi bahan kasus uji yang baik?
3. Apakah sistem sensitif pada input-input tertentu?
4. Bagaimana sekumpulan data dapat diisolasi?
5. Berapa banyak rata-rata data dan jumlah data yang dapat ditangani sistem?
6. Efek apa yang dapat membuat kombinasi data ditangani spesifik pada operasi sistem?

Saat ini terdapat banyak metoda atau Teknik untuk melaksanakan Black Box Testing, antara lain:

a. *Equivalence Partitioning*
b. Boundary Value Analysis/Limit Testing
c. Comparison Testing
d. Sample Testing
e. *Robustness Testing*
f. *Behavior Testing*
g. Requirement Testing
h. Performance Testing
i. Uji Ketahanan (*Endurance Testing*)
j. Uji Sebab-Akibat (*Cause-Effect Relationship Testing*)

2.11. *White-Box Testing*

Pengujian *white box (glass box)* adalah pengujian yang didasarkan pada pengecekan terhadap detil perancangan, menggunakan struktur kontrol dari desain program secara procedural untuk membagi pengujian ke dalam beberapa kasus pengujian. Penentuan kasus uji disesuaikan dengan struktur system, pengetahuan mengenai program digunakan untuk mengidentifikasikan kasus uji tambahan. (Liana, 2015)

Tujuan penggunaan *white box* untuk menguji semua statement program. Penggunaan metode pengujian *white box* dilakukan untuk :

1. Memberikan jaminan bahwa semua jalur independen suatu modul digunakan minimal satu kali.
2. Menggunakan semua keputusan logis untuk semua kondisi *true* atau *false*.
3. Mengeksekusi semua perulangan pada batasan nilai dan operasional pada setiap kondisi.
4. Menggunakan struktur data internal untuk menjamin validitas jalur keputusan. (Liana, 2015)

2.12. Penelitian Terdahulu dan Matriks Penelitian

A. Penelitian Terdahulu

Penelitian terdekat ini menjadi salah satu acuan penulis dalam melakukan penelitian sehingga penulis dapat memperkaya teori yang digunakan dalam mengkaji penelitian yang dilakukan. Perbedaan penelitian yang dilakukan dengan penelitian terdahulu ialah penerapan algoritma *collision detection* pada *non player character* sehingga mampu menghilangkan jeda dan membuat non payer character

lebih responsive dan membuat game menjadi lebih hidup. Penambahan algoritma *random number generator* pada proses kemunculan *drop item* menjadikan suasana *game* lebih bervariasi dan lebih menantang untuk dimainkan. Tidak ditemukannya penelitian dengan judul yang sama seperti judul penelitian pada penelitian terdahulu, hanya mengangkat beberapa penelitian sebagai referensi dalam memperkaya bahan kajian pada penelitian ini. Pada tabel 2.1 merupakan penelitian terdahulu berupa jurnal terkait dengan penelitian ini

Tabel 2.1 Penelitian Terdahulu

No	Peneliti/Tahun	Judul	Algoritma	State Of The Art
1	Harso Kurniadi, Kusrini, (2017)	Implementasi Algoritma *A Stars, Tilebase Collision Dan Fuzzy Logic* Pada Game Strategy	*A Stars, Tilebase Collision Dan Fuzzy Logic*	Pada penelitian Harso Kurniadi, Algoritma *Tilebase Collision* dapat digunakan untuk mendeteksi tabrakan antara semua objek dan menjalankan algoritma *Fuzzy Logic* untuk mengantar karakter keluar dari *collision* dan kembali menuju target sehingga diperlukan 2 algoritma untuk melakukan deteksi terhadap *collision* dan keluar dari kondisi *collision*. Sedangkan pada penelitian yang penulis lakukan, cukup menggunakan 1 algoritma yaitu algoritma *collision detection* untuk masuk dan keluar dari kondisi *collision.*
2.	Arif Nurdiyanto, Edy Winarno, (2018)	Penerapan Metode *Collision Detection* Pada Game Petualangan	*Collision Detection*	Pada penelitian Arif Nurdiyanto, algoritma *Collision Detection* diterapkan hanya pada objek yang bersentuhan langsung dengan karakter utama. Pada penelitian yang

		Menggunakan Aksara Jawa		penulis lakukan, *Collision Detection* juga diterapkan pada jalur yang akan dilalui oleh karakter utama untuk memicu *enemy* atau *event*.
3.	Lia Musfiroh*, Ahmad Jazuli, Anastasya Latubessy (2014)	Penerapan Algoritma *Collision Detection* dan BOIDS pada Game Dokkaebi Shooter	*Collision Detection* dan *BOIDS*	Metode yang digunakan dalam penelitian ini adalah metode 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. Perbedaan penelitian yang dilakukan peneliti dengan penelitian Lia Musfiroh ialah dalam penggunaan metode serta penerapan algoritma *Collision Detection* yang hanya dilakukan pada 3 objek saja. Belum adanya animasi pada game Dokkaebi

				merupakan salah satu perbedaan antara kedua penelitian ini.
4.	Siti Asmiatun (2016)	Penerapan Algoritma Collision Detection Dan Bayesian Untuk Strategi Menyerang Jarak Dekat Pada Npc (Non Player Character) Menggunakan Unity 3d	*Collision Detection* dan *Bayesian*	Penelitian ini menggabungkan dua metode Collision Detection dengan bayesian untuk strategi menyerang jarak dekat. Dalam penelitian ini adalah membagi beberapa perilaku penyerangan NPC ketika berada pada posisi paling dekat dengan musuh. Penelitian ini menerapkan algoritma bayesian untuk klasifikasi perilaku penyerangan NPC dan algoritma collision detection untuk pengambilan keputusan perilaku ketika NPC menabrak player. Perbedaan penelitian yang dilakukan dengan penulis ialah collision detection yang bukan hanya diterapkan pada NPC dengan jarak dekat melainkan pada setiap objek environment dalam game dan juga pada

				enemy yang berada di jarak lebih jauh dari karakter untuk memberitahu NPC posisi karakter utama.
5.	Lui Haekal Fasha, Fauziah, M. Gufroni (2018)	Implementasi Algoritma *Collision Detection* pada *game simulator driving car*	Collision Detection	Penelitian ini metode ini untuk perancangan sebuah game Simulasi berkendara, ketika player menabrak sebuah objek obstacle / penghalang yang sudah disusun pada sebuah lintasan agar mendapatkan sebuah reaksi pada salah satu objek yang ditentukan. Perbedaan dengan penelitian yang dilakukan penulis ialah platform yang digunakan, penerapan collision detection pada objek obstacle akan membuat game reset sedangkan pada penelitian yang dilakukan peneliti dapat menimbulkan kondisi-kondisi yang berbeda dan posisi obstacle yang tidak berubah menyebabkan game menjadi monoton sedangkan pada penlitian penulis drop item di buat acak menggunakan random number generator yang akan

				memberikan tatangan yang berbeda untuk setiap player atau setiap stage game yang dimainkan.

B. Matrik Penelitian

Penyusunan matrik diantaranya meliputi judul, permasalahan, variabel, indikator, data yang hendak di gali maupun teknik untuk pengumpulan data yang dilakukan. Beberapa variabel biasanya di muat di dalamnya, selain itu indikator-indikator yang berpengaruh juga menjadi bahan masukan penelitian, biasanya data yang digali ada beberapa hal termasuk di antaranya hal-hal yang hendak digali lebih jauh dengan dilakukannya penelitian tersebut.

Penelitian ini dibuat kedalam bentuk Matrik Penelitian yang dapat dilihat pada tabel 2.2

Tabel 2.2 Matrik Penelitian

No	Peneliti	Judul	Lingkup Penelitian				
			Algoritma	Variabel	Indikator	Sumber Data	Metode Penelitian
1.	Arif Nurdiyanto, Edy Winarno, (2018)	Penerapan Metode Collision Detection Pada Petualangan Aksara Jawa	✔	✔	✔	✔	✔
2.	Siti Asmiatun (2016)	Penerapan Algoritma *Collision Detection* dan *Bayesian* untuk Strategi menyerang jarak dekat pada *NPC* menggunakan Unity3D	✔	✔	✔	✔	✔
3.	Lui Haekal Fasha, Fauziah, M. Gufroni (2018)	Implementasi Algoritma *Collision Detection* pada *game simulator driving car*	✔	✔	✔	✔	✔

4	Andhi Indra Lestya Wicaksono, Eriq Muhammad Adams Jonemaro, Muhammad Aminul Akbar	Optimasi *Collision Detection* Pada 2D *Spaceship Game* Menggunakan Metode *Quadtree*	✔	✔	✔	✔	✔
5.	Imran Hasbi Zulfahmi (2020)	Implementasi Algoritma *Collision Detection* Pada *Enemy* Dan *Algoritma Random Number Generator* Dalam Game Pencarian Harta Karun	✔	✔	✔	✔	✔