TINJAUAN PUSTAKA

2.1 Kajian Terkait

Berikut merupakan beberapa penelitian sebelumnya yang digunakan sebagai pembanding dalam mengimplementasikan metode Finite State Machine dan Collision Detection .

Firmanu Alamsyah, Wasum, Amak Yunus (2019) melakukan penelitian dengan judul Implementasi Algoritma Collision Detection dan Finite State Machine Untuk Karakter Musuh Pada Game Bertipe Metoridvania,pada penelitian ini bertujuan untuk menerapkan Artificial Intelligence atau AI.Permainan ini berbasis android menggunakan tools Unity 3D, metode yang digunakan menggunakan metode Finite State Machine dan Collision Detection.Aplikasi ini dirancang sebagai hiburan. Pengujian permainan menggunakan LVBC (low Voltage Circuit Breaker) di laboratorium Teknik.

Ita Arfayanti, Salmon, Nursobah, Sugeng Suryani (2020) melakukan penelitian dengan judul Development Zombie Hunter Battleground With Finite State Macin and Collision Detection,pada penelitian bertujuan sebagai hiburan. Metode penelitian menggunakan Finite State Machine dan Collision Detection yang diterapkan kepada player terhadap zombie, permainan ini dibangun dengan menggunakan tahapan pengembangan multimedia.

Habri Kurniawan (2020) melakukan penelitian dengan judul Penerapan Metode Finite state Machine Dan Collision Detection Pada Game Orge Slayer Berbasis Dekstop, metode penelitian menggunakan metode Finite State Machine diterapkan pada karakter terhadap musuh dan pintu. Proses permainan ini dibangun menggunakan MDLC (Multimedia Development Life Cycle), pembuatan game menggunakan Unity 3D.Pengujian penelitian dilakukan menggunakan beta testing dan white box, permainan ini berbasis desktop

Tabel 2.1 Tabel Perbandingan Penelitian

No	Penulis	Judul	Keterangan
1	Firmanu Alamsyah, Wasum Diwa, Amak Yunus	Implementasi Algoritma Collision Detection Dan Finite State Machine Untuk Karakter Musuh Pada Game Bertipe Metroidvania.	• Menggunakan Algoritma Collision Detection dan Finite State Machine antara pemain dengan musuh utama dan regular • Pengujian menggunakan LVBC (Low Voltage Circuit Breaker)
2	Ita Arfyanti, Salmon Salmon, Nursobah Nursobah, Sugeng Suryani	Development Zombie Hunter Battleground With Finite State Machine Dan Collision Detection.	• Menggunakan Algoritma Finite State Machine dan Collision Detection terhadap player zombie 1,zombie 2 dan bos zombie • Menggunakan pengembangan multimedia aplikasi ini dibangun atau MDLC
3	Habri Kurniawan	Penerapan Metode Finite state Machine Dan Collision Detection Pada Game Orge Slayer Berbasis Dekstop	• Menggunakan Metode Finite State Machine dan Collision Detection diterapkan pada player ke musuh dan pintu

			• Melakukan pengujian menggunakan beta testing dan white box • Aplikasi berbasis dekstop

Penelitian yang akan di lakukan berjudul Implementasi Metode Finite Machine Dan Algoritma Collision Detection (CD) Pada Aplikasi Permainan Edukasi Driver Pro, penelitian permainan ini bertujuan memberikan edukasi informasi pembelajaran rambu lalu lintas dijalan raya. Metode penelitian ini menggunakan metode FSM dan Algoritma CD.Pada metode FSM dibagi menjadi dua friendly dan non friendly, untuk penerapan pada friendly diterapkan pada polisi yang berdiri dekat dengan rambu lalu lintas dan non friendly diterapkan pada penjahat (begal). Pada algoritma CD diterapkan pada kendaraan lain seperti mobil, truk, dan lainnya. Metodologi penelitian pengembangan GDLC dan pengujian game di lakukan dengan FSM dan CD.

Adapun penjelasan terkait penelitian yang dilakukan dapat dilihat pada tabel 2.

Tabel 2. 2 Penelitian Yang dilakukan

No	Penulis	Judul	Keterangan
1	M Ridwansyah	Implementasi Metode Finite State Machine Dan Algoritma Collision Detection Pada Aplikasi Permainan Edukasi Driver Pro.	• Permainan bertujuan media informasi pembelajaran rambu lalu lintas pada Kota Pontianak. • Menggunakan Algoritma Collision Detection dan Finite State Machine pada NPC.

			• Pengujian menggunakan white box Finite state machine dan Collision Detection • Menggunakan metodologi pengembangan game GDLC

2.2 Kecerdasan Buatan

Kecerdasan buatan / Artificial Intelligence adalah bagian dari ilmu komputer yang mempelajari tentang bagaimana sebuah komputer bisa dibuat dengan sedemikian rupa agar dapat melakukan pekerjaan seperti dan sebaik yang dilakukan oleh manusia. Menurut John McCarthy (1956), kecerdasan buatan adalah suatu sistem komputer yang terbentuk untuk mengetahui dan memodelkan proses-proses berpikir manusia dan mendesain mesin agar dapat menirukan perilaku manusia.

Manusia bisa dengan pandai menyelesaikan masalah-masalah yang muncul karena manusia memiliki pengetahuan dan pengalaman yang dapat membantu dalam memecahkan masalah. Agar komputer dapat bertindak seperti dan sebaik manusia maka komputer diberikan pengetahuan dan kemampuan untuk menalar agar dapat mendapatkan pengalaman seperti layaknya manusia.

Ada tiga tujuan kecerdasan buatan, yaitu: membuat komputer lebih cerdas, mengerti tentang kecerdasan, dan membuat mesin lebih berguna. Yang dimaksud kecerdasan adalah kemampuan untuk belajar atau mengerti dari pengalaman, memahami pesan yang kontradiktif dan ambigu, menanggapi dengan cepat dan baik atas situasi yang baru, menggunakan penalaran dalam memecahkan masalah serta menyelesaikannya dengan efektif (Winston dan Prendergast, 1994).

2.3 Game Edukasi

Game edukasi merupakan permainan digital yang dirancang untuk pengayaan Pendidikan (mendukung pengajaran dan pembelajaran), dengan

menggunakan teknologi multimedia interaktif. Menurut Hurd dan Jenuings (2009), perancang yang baik haruslah memenuhi kriteria dari education game itu sendiri (Nugraheny & Destiranti, 2017).

Berikut ini adalah beberapa kriteria dari permainan edukasi, yaitu :

a. Nilai keseluruhan dari suatu game terpusat pada desain dan panjang durasi game.

b. Mudah digunakan dan diakses adalah hal penting bagi pembuat game.

c. Keakuratan diartikan sebagai bagaimana kesuksesan model/gambaran sebuah game dapat dituangkan ke dalam percobaan atau perancangannya.

d. Kesesuaian dapat diartikan bagaimana isi dan desain game dapat diadaptasikan terhadap keperluan user dengan baik.

e. Relevan artinya game sesuai dengan user atau isi game dapat diaplikasikan ke target user. Agar dapat relevan terhadap user, sistem harus membimbing user dalam pencapaian tujuan pembelajaran.

f. Objektifitas menentukan tujuan user dan kriteria dari kesuksesan atau kegagalan dari user dalam bermain.

g. Untuk membantu pemahaman user bahwa permainan atau performa user sesuai dengan objek game atau tidak, feedback harus disediakan.

2.4 Game Petualangan

Menurut David Fox dan Roman Verhosek (2002:9), *game* dapat diklasifikasikan antara lain *Action Game, Combat Game, Adventure Game, Puzzle Game, Strategy Game,* dan *Card Game*. Pada penelitian ini, *game* dibangun dengan jenis *Adventure Game* atau *game* petualangan.

Adventure game sering dianggap sebagai bentuk interaksi fiksi. Interaksi fiksi merupakan sebuah istilah yang mengacu pada media di mana pemain dapat mempengaruhi hasil cerita.

Karakter kunci dari *game* petualangan sebagai berikut:

1. Narasi sebagai daya tarik dimana pergerakan pemain sebagai hasil kemajuan permainan
2. Narasi seringkali diambil dari film, komik dan lain sebagainya.
3. Pemain umumnya mengontrol karakter utama.
4. Permainan sering didasarkan pertanyaan atau teka teki, yang harus diselesaikan dengan berinteraksi dengan lingkungan permainan dan objek objek yang menghasilkan pengalaman bagi pemain.
5. Penekanan dalam *game* petulangan adalah pada eksplorasi, berpikir dan kemampuan pemecahan masalah secara cepat sebagai gaya aksi permainan
6. Unsur – unsur mendasar diantaranya:
    a. *Game*s *Rules*: dijelaskan oleh penulis (pembuat *game*), yang mengatur operasi dan fungsi objek maupun karakter dalam permainan. Peraturan juga bisa didapatkan ketika pemain telah bermain.
    b. *Game World* : objek yang ada dalam permainan gua, pulau, penduduk dan lain sebagainya.
    c. *Plot* : Berisi informasi apa yang terjadi sebelum pemain bermain dan tujuan menyelesaikan permainan dan beberapa cara untuk mnyelesaikan permainan.
    d. *Theme* : Tema moral yang biasanya mendasari permainan. seperti pemaian diharuskan menjawab pertanyaan jika berhasil maka dapat memulihkan keseimbangan lingkungan.
    e. *Characters*: Player maupun NPC cenderung untuk memproses sifat tertentu atau atribut (kekuatan sihir, fitur- fitur khusus, wajah dsb)
    f. *Object / Item* : mempunyai peran penting dan biasanya dikumpulkan dan digunakan oleh pemain untuk memecahkan masalah. Seringkali pemain harus memiliki keahlian tertentu atau pengetahuan untuk menggunakannya.
    g. *Text, Graphics* dan *Sound* : *game* melibatkan kombinasi dari teks, grafis dan suara.
    h. *Animation*: animasi biasanya telah diprogram dan ditanamkan dalam

permainan, yang berfungsi membantu dan memberi kesenangan pada pemain.

i. *User Interface* : tampilan pada layar memungkinkan pengguna untuk berkomunikasi dengan mudah melalui pemilihan teks, grafis, suara dan animasi. (Dillon, 2005).

2.5 Storyboard

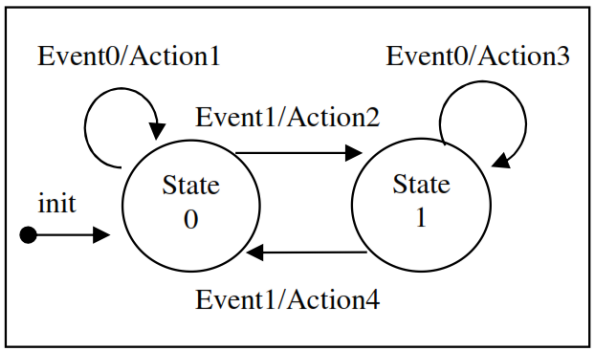
Storyboard merupakan serangkaian sketsa (gambaran kartun) dibuat berbentuk persegi panjang yang menggambarkan suatu urutan (alur cerita) elemen-elemen yang diusulkan untuk aplikasi multimedia. *Storyboard* visualisasi ide dari aplikasi yang akan dibangun, sehingga dapat memberikan gambaran dari aplikasi yang akan dihasilkan. *Storyboard* dapat dikatakan juga visual *script* yang akan dijadikan *outline* dari sebuah proyek, ditampilkan *shot by shot* yang biasa disebut dengan istilah *scene* (Suyanto, 2003).

Sebuah *Storyboard* media interaktif dapat digunakan dalam antarmuka grafik pengguna untuk rancangan rencana desain sebuah *web*site atau proyek interaktif sebagaimana alat visual untuk perencanaan isi. Sebaliknya, sebuah *sitemap* (peta) atau *Storyboard* (diagram alur) dapat lebih bagus digunakan untuk merencanakan arsitektur informasi, navigasi, links, organisasi dan pengalaman pengguna, terutama urutan kejadian yang susah diramalkan atau pertukaran audiovisual kejadian menjadi kepentingan desain yang belum menyeluruh.

Salah satu keuntungan menggunakan *Storyboard* adalah dapat membuat pengguna untuk mengalami perubahan dalam alur cerita untuk memicu reaksi atau ketertarikan yang lebih dalam. Seorang pembuat *Storyboard* harus mampu menampilkan sebuah cerita yang bagus. Pada proyek tertentu, pembuat *Storyboard* memerlukan keterampilan menggambar yang bagus dan kemampuan beradaptasi terhadap desain yang telah dikeluarkan.

2.6 Finite State Machine

Finite state machine (FSM) adalah sebuah metodologi perancangan sistem kontrol yang menggambarkan tingkah laku atau prinsip kerja sistem dengan menggunakan tiga hal berikut: State (Keadaan), Event (kejadian) dan Action (aksi). Pada satu saat dalam periode waktu yang cukup signifikan, sistem akan berada pada salah satu state yang aktif. Sistem dapat beralih atau bertransisi menuju state lain jika mendapatkan masukan atau event tertentu, baik yang berasal dari perangkat luar atau komponen dalam sistemnya itu sendiri (misal interupsi timer). Transisi keadaan ini umumnya juga disertai oleh aksi yang dilakukan oleh sistem ketika menanggapi masukan yang terjadi. Aksi yang dilakukan tersebut dapat berupa aksi yang sederhana atau melibatkan rangkaian proses yang relatif kompleks (Setiawan, 2006).



Gambar 2.1 *Finite State Machine*

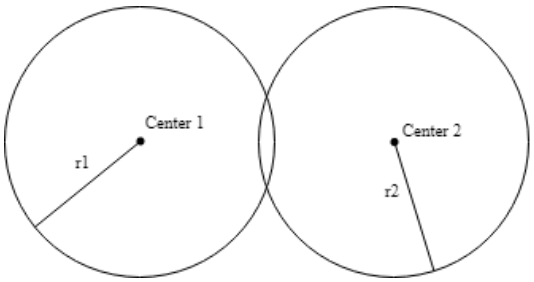
Berdasarkan sifatnya, metode FSM ini sangat cocok digunakan sebagai basis perancangan perangkat lunak pengendalian yang bersifat reaktif dan real time. Salah satu keuntungan nyata penggunaan FSM adalah kemampuannya dalam mendekomposisi aplikasi yang relatif besar dengan hanya menggunakan sejumlah kecil item state. Selain untuk bidang kontrol, Penggunaan metode ini pada kenyataannya juga umum digunakan sebagai basis untuk perancangan protokol-protokol komunikasi, perancangan perangkat lunak game, aplikasi WEB dan sebagainya (Setiawan, 2006).

Dalam gambar di atas memperlihatkan FSM dengan dua buah state dua buah input serta empat buah output yang berbeda seperti terlihat pada gambar,

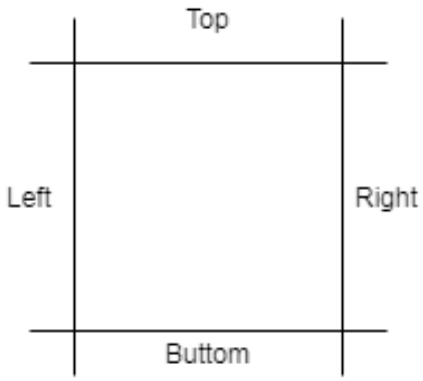
ketika sistem mulai di hidupkan, sistem akan bertransisi menuju State0, jika dimasukkan Event0 akan menghasilkan Action1 sedangkan Event1 maka akan menghasilkan Action2 akan dieksekusi ke sistem selanjutnya bertransisi ke keadaan State1 dan seterusnya.

2.7 Collision Detection

Collision detection merupakan teknik deteksi tabrakan untuk mengetahui obyek-obyek apa saja yang bersentuhan dalam bidang koordinat tertentu. Obyek-obyek ini bisa saja memiliki bentuk yang sangat bervariasi. Obyek-obyek pada game memiliki bentuk yang bervariasi, ada yang berbentuk Kotak, segi-n, sampai bentuk pesawat pemain yang sangat mendetail. Untuk mempercepat proses pada *Collision Detection*, umumnya obyek - obyek ini direpresentasikan secara logic dengan bentuk primitif seperti segiempat dan lingkaran (jika pada koordinat dua dimensi), atau kubus dan bola (jika pada koordinat tiga dimensi). Bentuk primitif yang merepresentasikan obyek ini biasa disebut sebagai Bounding Box atau Bounding Circle.(Nurdiyanto & Winarno, 2018).



Gambar 2. 2 *Bounding Circle*



Gambar 2.3 *Bounding Box*

2.8 Pengertian Rambu Lalu Lintas

Rambu lalu lintas menurut Keputusan Menteri Perhubungan Nomor: 61 tahun 1993 diartikan: "salah satu dari perlengkapan jalan, berupa lambang, huruf, angka, kalimat dan/atau perpaduan di antaranya sebagai peringatan, larangan, perintah atau petunjuk bagi pemakai jalan" (Affandi, 2009).

Dalam keputusan Direktur Jendral Perhubungan Darat (Nomor : SK.116/AJ.404/DRJD/97) tentang penyelenggaraan rambu lalu lintas meliputi:

1. Inventarisasi tingkat pertumbuhan rambu lalu lintas.
2. Survei untuk menentukan kebutuhan rambu termasuk penentuan lokasi penempatan atau pemasangannya.
3. Perkiraan kebutuhan pasang untuk 5 tahun.
4. Penyusunan program dan pengadaan rambu.

Jenis rambu-rambu lalu lintas yang tercantum di undang-undang pemerintah terbagi menjadi 4 yaitu :

1. Rambu Peringatan
2. Rambu Larangan
3. Rambu Perintah
4. Rambu Petunjuk

Secara fisik rambu-rambu dibagi menjadi 2 bagian :

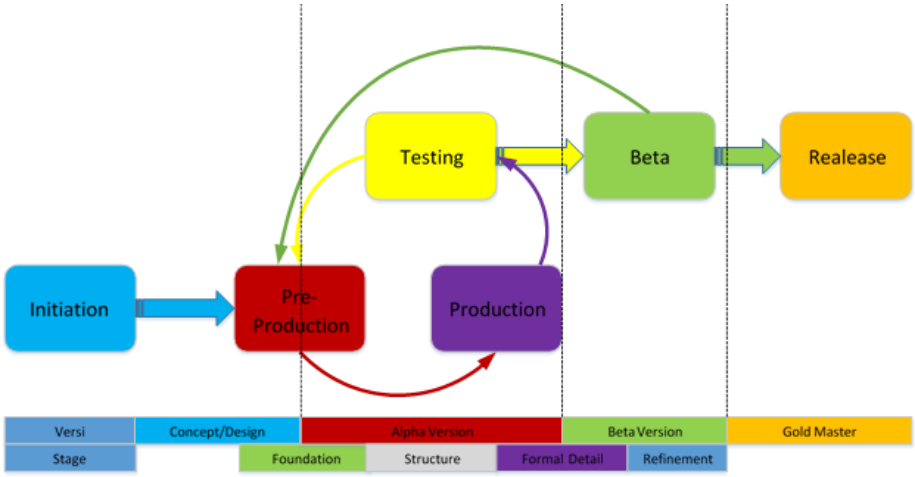
1. Daun Rambu

2. Tiang Rambu

2.9 Game Development Life Cyle (GDLC)

Game Development Life Cyle (GDLC) merupakan sebuah metode yang menangani pengembangan game di mulai dari titik awal hingga paling akhir. Di mulai dari tahap pembuatan ide dan konsep mengenai *game* yang akan dibuat, sedangkan tahap akhir dari game development adalah saat game di rilis (Putra & Kesuma, 2021).

Game Developmeent Life Cycle (GDLC) terdiri beberapa fase untuk membangun sebuah permainan sebagai berikut:



Gambar 2.4 *Game Development Life Cycle*

2.9.1 Initiation (Inisiasi)

Tahap inisiasi adalah proses awal yang berupa pembuatan konsep kasar dari game, mulai dari menentukan game seperti apa yang akan dibuat, mulai dari indentifikasi dari trending, topik, target user dari game yang akan dibuat.

2.9.2 Pre- Production (Pra- Produksi)

Tahap pra-produksi melibatkan penciptaan dan revisi desain game dan pembuatan prototipe permainan. Desain game berfokus pada mendefinisikan genre

permainan, gameplay, game mekanik/konfensional, alur cerita, karakter, tantangan, faktor kesenangan, aspek teknis, dan dokumentasi elemennya dalam Dokumen Desain Game (GDD).

2.9.3 Production (Produksi)

Produksi adalah proses inti yang berputar sekitar penciptaan aset pembuatan (karakter pemain, karakter NPC, obyek pendukung, latar belakang), kode sumber, integrasi kedua elemen.

2.9.4 Testing (Pengujian)

Pengujian dalam konteks ini berarti pengujian internal dilakukan untuk menguji kegunaan permainan dan menjalankan di target platform. Aksesibilitas dapat diuji melalui pengamatan perilaku penguji. Jika tester merasa sulit untuk bermain dan memahami permainan, itu berarti bahwa *game* tersebut tidak cukup dapat diakses. *Output* dari pengujian adalah laporan *bug*, permintaan perubahan, dan keputusan pengembangan. Hasilnya akan memutuskan apakah sudah waktunya untuk maju ke fase berikutnya (Beta) atau mengulangi siklus produksi.

2.9.5 Beta (Beta)

Beta adalah fase untuk melakukan pengujian pihak ketiga atau eksternal. Pengujian beta masih menggunakan metode pengujian yang sama dengan metode pengujian sebelumnya. Pengujian beta adalah perincian dan penyempurnaan formal.

2.9.6 Release (Rilis)

Tahap akhir dari semua proses pengembangan game yaitu game dirilis ke publik. Rilis melibatkan peluncuran produk, dokumentasi proyek, berbagi pengetahuan, dan perencanaan untuk pemeliharaan dan ekspansi permainan.

2.10 Android

Menurut Safaat (2015) Android adalah sebuah sistem operasi untuk perangkat mobile berbasis linux yang mencakup sistem operasi, middleware, dan aplikasi (Fauziah, 2018).

Perkembangan versi android Menurut Istiyanto (2014) adalah sebagai berikut: Android versi 1.1, Android versi 1.5 (Cupcake), Android versi 1.6 (Donut), Android versi 2.0/2.1 (Éclair), Android versi 2.2 (Froyo), Android versi 2.3 (Gingerbread), Android versi 3.0/3.1 (Honeycomb), Android versi 4.0 (Ice Cream Sandwich), Android versi 4.1 (Jelly Bean), Android versi 4.4 (KitKat), Android versi 5.0 (Lollipop), Android versi 6.0 (Marshmallow), Android versi 7.0 (Nougat), Android versi 8.0 (Oreo), Android versi 9.0 (Pie) (Fauziah, 2018).

2.11 Construct 2

Construct 2 adalah tools pembuat game berbasis HTML5 yang dikhususkan untuk platform 2D yang dikembangkan oleh Scirra. Pada software Construct 2 kita tidak menggunakan bahasa pemrograman khusus, karena semua perintah yang digunakan pada game diatur dalam EvenSheet yang terdiri dari *Event* dan *Action* (Ridoi, 2018).

2.12 HTML 5

HTML5 adalah sebuah bahasa markah untuk menstrukturkan dan menampilkan isi dari World Wide Web, sebuah teknologi inti dari Internet. HTML5 adalah revisi kelima dari HTML (yang pertama kali diciptakan pada tahun 1990 dan versi keempatnya, HTML4, pada tahun 1997 dan hingga bulan Juni 2011 masih dalam pengembangan. Tujuan utama pengembangan HTML5 adalah untuk memperbaiki teknologi HTML agar mendukung teknologi multimedia terbaru, mudah dibaca oleh manusia dan juga mudah dimengerti oleh mesin .

HTML5 merupakan salah satu karya Konsortium World Wide Web (*World Wide Web Consortium,* W3C) untuk mendefinisikan sebuah bahasa markah tunggal yang dapat ditulis dengan cara HTML ataupun XHTML. HTML5

merupakan jawaban atas pengembangan HTML 4.01 dan XHTML 1.1 yang selama ini berjalan terpisah, dan diimplementasikan secara berbeda-beda oleh banyak perangkat lunak pembuat web (Wikipedia, 2022).

2.13 Netlify

Netlifiy adalah perusahaan *cloud computing* yang menawarkan layanan *hosting* dan beckend tanpa server untuk aplikasi web dan web statis. Perusahaan ini menyediakan *hosting* untuk situs web yang berkas sumbernya disimpan dalam sistem pengendalian versi git yang kemudian dibuat menjadi konten web statis (Sarfaraz Reydhan, 2020).

2.14 MIT App Inventor

App Inventor adalah aplikasi web *open source* yang awalnya dikembangkan oleh Google, dan saat ini dikelola oleh MIT. Situ ini memungkinkan pengguna baru untuk memprogram komputer untuk menciptakan aplikasi perangkat lunak bagi sistem operasi Android. App Inventor menggunakan antarmuka grafis, serupa dengan antarmuka pengguna pada Scratch dan Star Logo TNG, yang memungkin pengguna untuk men-*drag*-dan *drop* obyek visual untuk aplikasi yang bisa dijalankan pada perangkat Android (Hardesty, 2010).

2.15 Pengujian Finite State Machine

Pengujian menggunakan transisi level 1

Tabel 2.3 Transisi Level 1 FSM

State	Next State	
	0	1
P1	P1	P2
P2	P2	P3
P3	P3	F
F	F	Ø

Penjelasan :

a. Jika skor pada level 1 tidak terpenuhi, maka sistem akan kembali ke *stage* 1 dan jika skor terpenuhi maka sistem akan lanjut ke *stage* 2

b. Jika skor pada level 2 tidak terpenuhi, maka sistem akan kembali ke *stage* 2 dan jika skor terpenuhi maka sistem akan lanjut ke *stage* 3.

c. Jika skor pada level 3 tidak terpenuhi, maka sistem akan kembali ke *stage* 3 dan jika skor terpenuhi maka sistem akan lanjut ke *stage* akhir (F) atau misi selesai.

2.16 Pengujian Collision Detection

Pada pengujian Collision Detection di terapkan pada pengujian Bounding Box terhadap regional- regional Collision Detection yang saling bertabrakan atau tidak, hal tersebut dilakukan sebuah pengujian dengan membandingkan nilai maksimum dan nilai minimum di area x.y, kordinat dua regional akan saling bertabrakan jika keadaan berikut (Nurdiyanto & Winarno, 2018):

Untuk menentukan bounding box pada obyek ditentukan rumus berikut :

Regional | $R=\{(x,y)\}$ | $\min X \leq x \leq \max X$

$\min Y \leq y \leq \max Y$

Regional R = Regional bounding box collision

X,y = titik koordinat x,y

minX,minY = Nilai minimum koordinat x,y

maxX,maxY =nilai maximum koordinat x,y

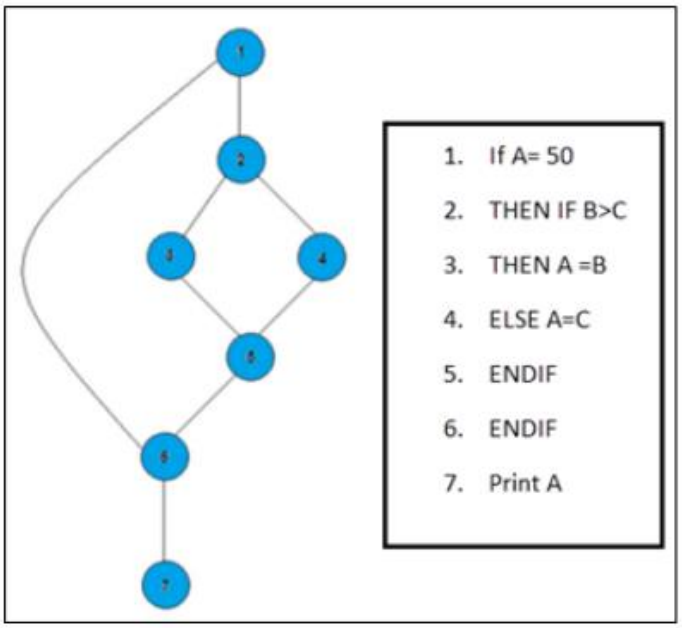
2.17 Pengujian White Box

Pengujian *white box* adalah pengujian yang dilakukan untuk menguji perangkat lunak dengan cara analisa dan meneliti struktur internal dan kode dari perangkat lunak, pengujian ini berfokus pada aliran *input* dan *output* dari perangkat lunak. Untuk melakukan pengujian ini, penguji perlu memiliki kemampuan dalam kode dari suatu program sehingga pengujian ini tidak bisa dilakukan oleh

sembarangan. Berikut teknik yang dapat ada pengujian *White Box* pada perangkat lunak.

2.17.1 Basis Path Testing

Teknik pengujian ini merupakan metode pengujian struktural yang melibatkan penggunaan kode sumber suatu program untuk menemukan setiap basis jalur yang dapat dieksekusi dalam waktu tertentu. Teknik ini secara mendasar dapat membantu pengembang untuk menentukan semua kesalahan yang terletak di dalam sepotong kode.



Gambar 2. 5 Grafis Basis *Path* Testing (guru99.com)

2.17.2 *Condition Coverage*

Condition coverage merupakan cakupan ekspresi untuk menguji dan mengevaluasi variabel atau sub-ekspresi dalam pernyataan kondisional. Tujuan dari teknik white box testing jenis ini adalah untuk memeriksa hasil individu pada setiap kondisi logis tertentu. Teknik ini menawarkan sensitivitas yang lebih baik terhadap aliran kontrol dalam suatu program. Meski begitu, teknik condition coverage tidak memberikan jaminan tentang cakupan keputusan yang penuh. Berikut ini adalah

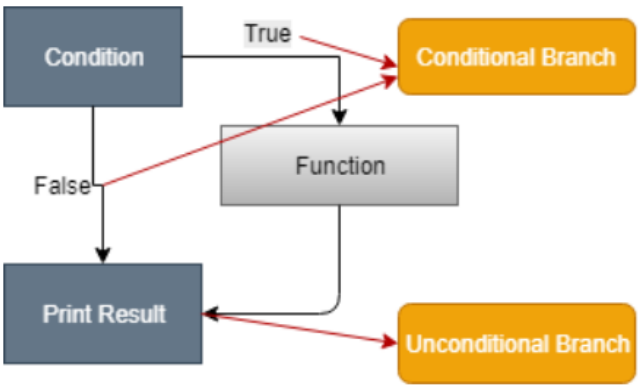
rumus formula yang dapat digunakan untuk menghitung condition coverage (Dewi, 2021):

$$Condition\ Coverage = \frac{Number\ of\ Executed\ Operands}{Total\ Number\ of\ Operands}$$

Gambar 2. 6 Rumus *Condition Coverage* (guru99.com)

2.17.3 *Branch Coverage*

Teknik *branch coverage* dalam *white box* testing digunakan untuk menutupi semua cabang dari grafik aliran kontrol (*control flow*). Aksi ini mencakup semua hasil, baik benar maupun salah, dari setiap kondisi titik keputusan sistem, setidaknya sekali dari setiap titik. Teknik *white box* testing ini adalah teknik pengujian yang memastikan bahwa setiap cabang dari setiap titik keputusan sistem harus dieksekusi. Teknik *branch coverage* ini umumnya akan mengikuti titik keputusan (*node*) dan tepi cakupan (edges) untuk menemukan persentase program dan jalur eksekusi selama dilakukan eksekusi program.



Gambar 2. 7 *Brach Coverage* (javatpoint.com)

2.17.4 . *Statement Coverage*

Statement coverage merupakan teknik *white box* testing yang melibatkan eksekusi semua pernyataan; setidaknya sekali; dalam kode sumber. Teknik *white*

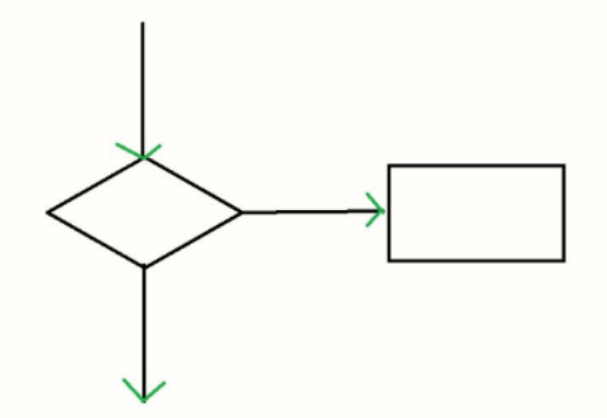
box testing ini berbentuk matriks yang digunakan untuk menghitung dan mengukur jumlah pernyataan dalam kode sumber yang telah dieksekusi,

$$\textit{Statement Coverage} = \frac{\textit{Number of executed statements}}{\textit{Total number of statements}} \textit{ x } 100$$

Gambar 2. 8 Rumus *Statement Coverage*

2.17.5 *Loop Testing*

Loop testing merupakan teknik pengujian *white box* testing yang dilakukan untuk memvalidasi loop dalam struktur kontrol. *Loop* sendiri diartikan sebagai struktur atau rangkaian proses yang awal hingga ujungnya saling terhubung



Gambar 2. 9 Grafis *Loop Testing* (geeksforgeeks)

2.17.6 *Multiple Condition Coverage*

Teknik *Multiple Condition Coverage* dalam *white box* testing merupakan matriks yang digunakan untuk mengeksekusi pernyataan dalam sistem suatu program. Dalam matriks ini semua pernyataan harus dieksekusi dan semua kombinasi nilai kebenaran dalam tiap keputusan harus terjadi setidaknya sekali untuk mencapai cakupan penuh.

2.18 Pengujian Alpha

Pengujian *Alpha* adalah salah satu strategi pengujian perangkat lunak yang

paling umum digunakan dalam pengembangan perangkat lunak, hal ini khusus digunakan oleh organisasi pengembangan produk dengan tujuan agar *system* yang dikembangkan terhindar dari cacat atau kegagalan penggunaan .

Pengujian *Alpha* biasanya dilakukan oleh kelompok yang independen dari tim desain, tim pengembang tapi masih dalam perusahaan, misalnya di rumah insinyur pengujian perangkat lunak, atau insinyur perangkat lunak QA Pengujian *alpha* berlangsung di situs pengembang oleh tim internal, sebelum rilis kepada pelanggan eksternal. Agar nantinya ketika pelanggan menggunakan sistem ini tidak kecewa karena masalah cacat atau kegagalan aplikasi.

Pengujian *alpha* secara umum memiliki dua fase:

1. Pada tahap pertama dari pengujian *alpha*, perangkat lunak diuji oleh pengembang di lingkungan internal developer. Tujuannya adalah untuk menangkap bug dengan cepat.
2. Pada tahap kedua pengujian alpha, *software* ini diserahkan kepada staf QA, untuk pengujian tambahan dalam lingkungan yang mirip dengan penggunaan yang dimaksudkan. Hal ini untuk simulasi suasana atau lingkungan pengujian yang sebenarnya sehingga ketika sistem tersebut dipasang, sudah tidak terjadi kegagalan maupun cacat sistem secara real.

2.19 Pengujian Beta

Pengujian Beta adalah aplikasi “hidup” dari perangkat lunak dalam sebuah lingkungan yang tidak dapat dikendalikan oleh pengembang (Pressman, 2010:572). Pengujian beta dilakukan diluar lingkungan yang tidak dapat dikendalikan oleh pengembang. Pengujian Beta suatu produk dilakukan oleh "pengguna nyata" dari aplikasi perangkat lunak dalam "lingkungan nyata". Versi Beta dari perangkat lunak ini dirilis ke sejumlah terbatas pengguna akhir produk untuk mendapatkan umpan balik mengenai kualitas produk. Pengujian Beta mengurangi risiko kegagalan produk dan memberikan peningkatan kualitas produk melalui validasi pelanggan. Pengujian Beta adalah tes terakhir sebelum mengirim produk ke pelanggan.

Keunggulan utama pengujian Beta yaitu dapat menerima umpan balik dari pengguna akhir.

2.20 Pengujian Validitas

Pengujian adalah uji yang digunakan untuk menunjukkan sejauh mana alat ukur yang digunakan dalam suatu mengukur apa yang diukur dengan tujuan untuk menilai apakah seperangkat alat ukur sudah tepat mengukur apa yang seharusnya di ukur. Pengujian validitas dilakukan dengan menghitung korelasi antara masing-masing pernyataan / indikator dengan skor total menggunakan korelasi *Product Moment* (r). Rumus korelasi *Product Moment (Perarson)* yang dilambangkan dengan *r*, dapat di tuliskan sebagai berikut (Gregory, 2021) :

$$rxy=\sum xy(\sum x2)(\sum y2)\surd$$

x = X – X = dan

y = Y – Y = sehingga

$$rxy=n(\sum xy)-(\sum x\sum y)[n\sum x2-(\sum x)2][n\sum y2-(\sum y)2]\surd$$

di mana :

rxy =Koefisien Korelasi

n = Jumlah Sampel

x = Skor total item pertanyaan

y = Skor Total item pertanyaan

$\sum x$ = Jumlah skor item pertanyaan

$\sum y$ = Jumlah skor total item pertanyaan

$\sum xy$ = Jumlah perkalian x dan y