BAB II

LANDASAN TEORI

2.1 *Game*

Game menurut Kamus Besar Bahasa Indonesia (KBBI) berarti sesuatu yang digunakan untuk bermain. Brathwaite dan Schreiber mengartikan *game* sebagai aktivitas yang melibatkan konflik dengan sesama pemain, dengan sistem itu sendiri, maupun dengan keburuntungan. *Video Game* adalah *game* yang menggunakan layar digital untuk menampilkan video atau animasi. (Wahyudi, 2017).

Schell berpendapat, *game* adalah kegiatan penyelesaian masalah, didekati dengan sikap yang menyenangkan, *game* juga sesuatu yang membuat pemain menemukan kesenangan dalam memainkannya. *Game* yang bagus adalah *game* yang dapat membuat penggunanya berpartisipasi secara aktif dan mempunyai jumlah tantangan yang tepat, tidak terlalu sedikit atau terlalu banyak. Sikap orang ketika sedang bermain *game*, bisa saja berbeda ketika orang itu sedang tidak bermain *game*, karena ketika orang tersebut sedang bermain *game* maka dia akan merasa sedang berada di "dunia" yang *game* tersebut ciptakan. (Singkoh, 2016).

2.1.1 Jenis-Jenis *Game*

Game terbagi dalam beberapa gaya permainan yang lebih dikenal dengan nama *genre*. Perbedaan *genre* itu dilakukan untuk menemukan *genre* baru yang

lebih menarik daripada biasanya. Berikut beberapa jenis genre yang biasa dimainkan yaitu, (Ardi, 2012):

a. Action

Tipe *game* dengan fitur utama berupa banyaknya aksi di mana pemain harus memiliki keterampilan reaksi yang cepat untuk menghindari musuh atau menghindari rintangan. Pengembang *game* tipe ini perlu memastikan *game* yang dibuat dioptimasi sehingga pemain memiliki pengalaman bermain yang baik, yang tidak terganggu oleh *delay* proses yang lama.

b. Adventure

Tipe *game* yang umumnya membuat pemain harus berjalan mengelilingi suatu tempat yang terkondisi, seperti sebuah istana, gua yang berkelok, dan planet yang jauh. Pemain melakukan navigasi suatu area, mencari pesan-pesan rahasia, memperoleh objek yang memiliki kemampuan yang bervariasi, bertempur dengan musuh, dan lain-lain. Perancangan yang akurat dibutuhkan untuk membuat *game* ini, sehingga memiliki alur cerita yang menarik bagi pemain.

c. Sport

Tipe *game* yang berupa kompetisi antara dua pemain atau lebih, di mana pemain dapat berupa individual atau tim. Contoh *game* tipe ini antara lain sepakbola, bola basket, tenis, dan bilyard. Tergantung seberapa cepat permainan yang terjadi, aplikasi *game* perlu dioptimalkan.

d. Role Playing Game (RPG)

Tipe *game* yang seringkali berupa *multi-player game* di mana setiap pemain memiliki karakter dengan kemampuan, kekuatan, dan kelemahan yang spesifik. Pemain saling berkompetisi, berinteraksi, dan bertempur satu sama lain. Tampilan grafis yang khas untuk setiap karakter pemain ditambah dengan *storyline* yang

mendebarkan akan sangat menarik dan memberikan pengalaman yang berbeda di dalam bermain.

e. Platform

Tipe *game* yang mengharuskan pemain mengarahkan suatu objek dengan melalui berbagai tahap atau tingkatan area untuk menyerang musuh dan menghindar terhadap serangan. Tipe *game* ini sedikit serupa dengan *action game*, tetapi aksinya tidak secepat *action game*. Teknik *collision detection* sangat sering dimanfaatkan pada tipe *game* ini.

f. Puzzle

Tipe *game* yang umumnya membuat pemain menggunakan kemampuan berpikirnya sebagai pengganti keterampilan reaksi yang cepat karena terdapat rahasia yang perlu dipecahkan. *Game* ini lebih bersifat statis dibanding *action game*. Pembuatan *game* tipe ini seringkali ditunjang dengan algoritma *Artificial Intelligence (AI).*

g. Sandbox

Tipe *game* yang umumnya ditujukan untuk menjelajahi suatu kota atau tempat dan bebas berinteraksi dengan objek disekitarnya. Mungkin *game* ini bisa di bilang mirip *RPG* tapi *Sandbox* sangat berbeda dengan *RPG game*, perbedaannya adalah tidak adanya peningkatan level karakternya.

h. First Person Shooter (FPS)

Tipe *game* yang mengunakan sudut pandang orang pertama untuk membidik atau membunuh musuh, sehingga hanya bagian tangannya saja yang terlihat dan tidak melihat tubuh karakter yang dimainkan.

i. Third Person Shooter (THS)

Tipe *game* yang mirip dengan *FPS*, tapi bedanya tipe *game* ini mengunakan sudut pandang orang ke-3, sehingga kita bisa melihat seluruh tubuh karakter yang dimainkan.

j. Music

Tipe *game* yang menuntut pemainnya untuk menekan tombol sesuai dengan tombol yang ada di layar dengan diiringi musik.

k. Fighting

Tipe *game* yang intinya harus menjatuhkan lawan tandingnya, entah itu dengan pukulan, tendangan, combo, maupun dengan jurus spesial.

l. Strategy

Tipe *game* yang mengharuskan pemainnya menggunakan taktik dan strategi untuk jeli dalam melihat setiap peluang, kelemahan musuh dan bijaksana dalam menggunakan sumber daya yang ada.

m. Simulation

Tipe *game* yang memberikan pengalaman atau interaksi sedekat mungkin dengan kendaraan yang aslinya, meskipun terkadang kendaraan tersebut masih eksperimen atau bahkan fiktif, tapi ada penekanan khusus pada *detail* dan pengalaman *realistic* menggunakan kendaraan tersebut.

n. Racing

Tipe *game* yang tujuannya adalah mencapai garis *finish* dari suatu *race*, dalam *game* ini biasanya pemain dapat memilih & membeli kendaraan, mendandani, meng-*upgrade* mesin dsb.

2.2 *Artificial Intelligence (AI)*

Kecerdasan Buatan atau *Artificial Intelligence (AI)* adalah teknik yang digunakan untuk meniru kecerdasan yang dimiliki oleh makhluk hidup maupun benda mati untuk menyelesaikan sebuah persoalan, untuk melakukan hal ini, setidaknya ada tiga metode yang dikembangkan.

1. *Fuzzy Logic (FL)*

Teknik ini digunakan oleh mesin untuk mengadaptasi bagaimana makhluk hidup menyesuaikan kondisi dengan memberikan keputusan yang tidak kaku 0 atau 1, sehingga dimunculkan sistem logika fuzzy yang tidak kaku. Penerapan logika fuzzy ini salah satunya adalah untuk sistem pengereman kereta api di Jepang.

2. *Evolutionary Computing (EC)*

Teknik Pendekatan ini menggunakan skema evolusi yang menggunakan jumlah individu yang banyak dan memberikan sebuah ujian untuk menyeleksi individu terbaik untuk membangkitkan generasi selanjutnya. Seleksi tersebut digunakan untuk mencari solusi dari suatu permasalahan. Contoh dari pendekatan ini adalah Algoritma Genetika yang menggunakan ide mutasi dan kawin silang, *Particle Swarm Optimization (PSO)* yang meniru kumpulan binatang seperti burung dan ikan dalam mencari mangsa, *Simulated Annealing* yang menirukan bagaimana logam ditempa, dan masih banyak lagi.

3. *Machine Learning (ML)*

Teknik *Machine Learning (ML)* atau pembelajaran mesin merupakan teknik yang paling populer karena banyak digunakan untuk menggantikan atau menirukan perilaku manusia ntuk menyelesaikan masalah. *ML* mencoba menirukan

bagaimana proses manusia atau makhluk cerdas belajar dan menggeneralisasi (Hania, 2017).

2.3 *Artificial Intelligence (AI)* pada *Game*

Kecerdasan buatan adalah penciptaan program komputer yang meniru cara bertindak dan berpikir manusia, suatu proses meniru cara bertindak dan berpikir secara rasional. Definisi cerdas ini mencakup kecerdasan kognitif dan kecerdasan berperilaku (suatu bentuk emulsi dari tindakan dan berpikir), ini juga termasuk pada penerapan sikap rasionalitas dan "kemanusiaan", karena menjadi manusia kadang-kadang jauh dari rasional, tetapi masih dianggap cerdas. Kecerdasan buatan pada *game* tidak membutuhkan penerapan seluas itu, *game* berbasis kecerdasan buatan tidak mengharapkan segala gagasan mengenai kecerdasan buatan diterapkan padanya. *Game* berbasis kecerdasan buatan dikhususkan pada kemampuan komputer mengendalikan unsur-unsur dalam *game* membuat keputusan cerdas ketika suatu kondisi memiliki beberapa pilihan dengan hasil akhir yang berbeda, sehingga menghasilkan perilaku yang relevan, efektif, dan berguna. Kecerdasan buatan dalam permainan sangat berorientasi pada hasil, dan dengan demikian, kita dapat mengatakan bahwa dunia *game* sangat berkaitan dengan perkembangan ilmu kecerdasan buatan (Troy, 2015).

2.4 Algoritma *A**

Konsep algoritma *A** (*A Star*) dalam sains komputer adalah algoritma komputer yang digunakan secara luas dalam mencari jalur (*path finding*) dan grafik melintang *(graph traversal)*, proses *plotting* sebuah jalur melintang secara efisian antara titik-titik, disebut *node*. Terkenal karena penampilan dan akurasinya, algoritma ini diperluas untuk berbagai bidang. *A Star* mencapai penampilan yang lebih baik dengan menggunakan heuristik. *A Star* menggunakan *Best First Search (BFS)* dan menemukan jalur dengan biaya terkecil (least-cost path) dari *node* awal *(initial node)* yang diberikan ke *node* tujuan (*goal node*). Algoritma ini menggunakan fungsi heuristik jarak ditambah biaya (biasa dinotasikan dengan f(x)) untuk menentukan urutan di mana *search*-nya melalui *node-node* yang ada di pohon *(tree)*.

Notasi yang dipakai oleh algoritma *A Star* adalah sebagai berikut:

f(n) = g(n) + h(n)

f(n) = biaya estimasi terendah

g(n) = biaya dari *node* awal ke *node* n

h(n) = perkiraan biaya dari node n ke *node* akhir.

Penerapan algoritma *A Star* memiliki beberapa terminologi dasar diantaranya *starting point*, simpul (*nodes*), A, *open list, closed list*, harga (*cost*), halangan (*unwalkable*). (Dalem, 2018).

Tabel 2.1 *pseudocode* algoritma *A Star*

```
function A*(start,goal)
closedset := the empty set // The set of nodes already
evaluated. openset := {start} // The set of tentative nodes to
be evaluated, initially containing the start node
came_from := the empty map // The map of navigated nodes.
```

Tabel 2.1 *pseudocode* algoritma *A Star* (Lanjutan I)

```
g_score[start] := 0 // Cost from start along best known path.
// Estimated total cost from start to goal through y.
f_score[start] := g_score[start] + heuristic_cost_estimate(start, goal)

while openset is not empty
    current := the node in openset having the lowest f_score[] value
    if current = goal
    return reconstruct_path(came_from, goal)

        remove current from openset
        add current to closedset
    for each neighbor in neighbor_nodes(current)
        if neighbor in closedset
            continue
        tentative_g_score := g_score[current] +
        dist_between(current,neighbor)

        if neighbor not in openset or tentative_g_score <
        g_score[neighbor]
        came_from[neighbor] := current
        g_score[neighbor] := tentative_g_score
        f_score[neighbor] := g_score[neighbor] +
        heuristic_cost_estimate(neighbor, goal)
        if neighbor not in openset
        add neighbor to openset
            return failure
function reconstruct_path(came_from,current)
total_path := [current]
while current in came_from:

current := came_from[current]
total_path.append(current)
return total_path
```

2.5 *Navigation Mesh (Navmesh)*

Navigation Mesh adalah *Pathfinding* yang dilakukan diantara *polygon* dalam *mesh* bisa diselesaikan dengan algoritma pencarian salah satu grafik dalam jumlah besar, misalnya *A Star*. (Snook, 2000).

Navigation mesh (Navmesh) telah menjadi konsep populer yang digunakan dalam masalah penelusuran jalan terpendek dari game 3D, karena lingkungan 3D sebagian besar menggunakan struktur poligon. *Navmesh* properti dari objek poligon atau medan dapat menjamin berjalan bebas untuk karakter permainan selama karakter yang berada di dalam poligon yang sama adalah seperangkat poligon kompleks.

Unity adalah mesin permainan lintas platform yang dikembangkan dan didirikan oleh Unity Technologies pada tahun 2005. Unity telah merilis beberapa versi, salah satunya adalah versi terbaru yang memiliki *fiture* baru yang disebut jalur *Navmesh Library*. Terdapat beberapa komponen yang menerapkan sistem *Navmesh* di Unity, seperti: *NavMesh, NavMesh Obstacle, Navmesh Agent,* dan *Off Mesh Link*. *Navmesh* adalah struktur poligon yang menggambarkan permukaan *walkable* dari dunia game dan memungkinkan untuk menemukan jalur dari satu lokasi *walkable* ke lokasi lain di dunia *game*. *Navmesh Agent* adalah komponen yang dapat bergerak ke arah tujuan di permukaan *Navmesh*, sehingga agen dapat menghindari rintangan. *Navmesh Obstacle* adalah objek yang harus dihindari oleh gerakan agen. *Off Mesh Link* adalah titik koneksi yang memungkinkan untuk menggabungkan pintasan navigasi yang tidak dapat direpresentasikan saat menggunakan permukaan *walkable*. (Zikky, 2016).

Pencarian *A Star* tidak dapat memberikan jalur nyata, proses lebih lanjut diperlukan untuk menemukan jalan nyata, algoritma saluran bodoh sederhana, digunakan untuk mengatasi masalah tersebut. (Xiao Cui, 2012).

2.6 Sensor *Gyroscope*

Salah satu *fiture* yang tersedia dalam beberapa *smartphone* yang dapat melakukan eksplorasi adalah sensor. Terdapat sekitar 13 sensor yang terdapat di *smartphone* Android. Eksplorasi data-data sensor salah satunya dapat digunakan untuk mendeteksi pergerakan manusia. Sensor *accelerometer* akan mengambil data secara *realtime* dari *smartphone* berbasis Android, dimana sensor ini

mengambil data percepatan linier dari *smartphone* berbasis Android. Sensor *gyroscope* juga akan mengambil data-data secara realtime dari *smartphone* berbasis Android, namun data yang diambil adalah data kecepatan sudut, percepatan sudut, serta perubahan sudut dari pergerakan tubuh manusia. Penggunaan sensor *accelerometer* dan sensor *gyroscope* dapat mendeteksi aktivitas fisik *user* pada saat terjatuh dan dapat membedakan dengan aktivitas fisik *user* yang menyerupai aktivitas jatuh seperi duduk cepat melompat ke atas, berbaring, dan lain-lain. (Zakiyan, 2017).

2.7 *Virtual Reality*

Virtual reality merupakan sebuah konsep dimana seseorang diberikan interferensi dengan menggunakan stimulus sensorik buatan yang menyebabkan seseorang tersebut mengalami perilaku tertentu. Era modern *VR* disebabkan oleh kemajuan dari segi tampilan, pengindraan, teknologi komputer dan industri *smartphone.* Perangkat *VR* diproduksi secara besar-besaran yang menyebabkan tren *VR* ini sejalan dengan revolusi komputer rumahan dan juga *web browser*. (LaValle, 2017).

Sherman dan Craig mendefinisikan *VR* sebagai sebuah media yang terdiri dari simulasi komputer interaktif yang merasakan posisi dan tindakan peserta, memberikan umpan balik sintetis untuk satu atau lebih indera, memberikan perasaan tenggelam atau hadir dalam simulasi. Perhatikan bahwa definisi menyatakan bahwa pengalaman virtual reality memberikan rangsangan sintetis untuk satu atau lebih dari indra pengguna. Sistem *VR* akan menggantikan

setidaknya rangsangan visual, dengan rangsangan aural juga sering disediakan. Dua istilah lain yang berhubungan dengan realitas *virtual* dan satu sama lain adalah "*telepresence*" dan "*augmented reality*" *(AR). Telepresence* mirip dengan *VR*, dalam hal ini adalah sarana untuk menempatkan pengguna di lokasi lain di mana mereka tidak hadir secara fisik. Perbedaan dari *VR* adalah bahwa lokasi ini sebenarnya adalah sebuah tempat yang nyata bahwa untuk satu atau alasan lain terlalu berbahaya atau tidak nyaman bagi orang untuk mengunjungi secara pribadi, seperti *telepresence, augmented*. (Zakiyan, 2017).

2.8 Unity

Unity adalah sebuah *game engine,* yaitu *software* pengolah gambar, grafik, suara, input, dan lain-lain yang ditunjukan untuk membuat *game*. Unity 3D merupakan *game engine multiplatform*, yang mampu di publish secara *standalone* (*.exe), berbasis *web*, Android, IoS, XBOX, maupun PS3, dengan catatan mendapatkan lisensi. Versi gratis hanya dapat dipublish ke dalam bentuk *standalone* dan *web*.

Unity merupakan sebuah *game engine* yang dibuat oleh Unity Technology. Kelebihan Unity dibandingkan dengan *game engine* lainnya adalah kemampuan membuat *game cross platform. Game* yang dibuat di dalam Unity 3D dapat dimainkan diberbagai perangkat, seperti smartphone dan *game console*. Unity sendiri dapat membuat berbagai macam *game*, seperti *RPG (Role Playing Game*), *Shooter, Racing*, dan lain sebagainya.

Unity 3D dibagi menjadi dua versi, yaitu versi berbayar dan versi gratis. Versi gratis memiliki beberapa *fiture* yang tidak dapat digunakan, seperti tidak dapat melakukan konversi *game* ke *console*. Meskipun demikian, dengan Unity 3D versi gratis, *game* yang dibuat masih dapat dimainkan. (Andi, 2014).

2.8.1 *Fiture* pada Unity 3D

Unity 3D memiliki banyak *fiture* yaitu, (Andi, 2014):

a. *Rendering* adalah Unity telah mendukung penggunaan *graphic engine*, seperti *Direct*3D (Windows, Xbox 360), *Open GL* (Mac, Windows, Linux, PS3), Open GL ES (Android, iOS), dan *proprietary APIs* (Wii). *Unity* 3D juga mendukung penggunaan *bump mapping*, *reflection mapping*, *parallax mapping*, *screen space ambient occlusion (SSAO)*, *dynamic shadows using shadow maps*, *render-to-texture and full-screen post-processing effects*. *Unity* 3D mendukung penggunaan *software* pengolahan gambar lain untuk meningkatkan kualitas pemetaan atau tokoh dalam *game*,, seperti 3ds max, Maya, Softimage, Blender, Modo, Zbrush, Cinema 4D, Cheetah 3D, Adobe Photoshop, Adobe Fireworks, dan Allegorithmic Substance.

b. *Scripting* yang digunakan pada Unity dibangun menggunakan Mono 2.6. Mono 2.6 merupakan implementasi *open-source* dari *.NET Framework*. Bahasa pemrograman yang didukung oleh Unity 3D, antara lain *JavaScript*, *C#*, dan Boo (menggunakan sintaks *phyton*). Mulai dari Unity 3D versi 3.0, Unity digunakan Mono Develop untuk *debugging script*.

c. *Asset Store* merupakan aspek dari permainan yang akan direferensikan oleh beberapa komponen, *asset* itu sendiri, atau kelengkapan penunjang pembuatan *game*. *Asset store* merupakan tempat untuk mendapatkan *asset* yang digunakan untuk menunjang pembuatan *game*. *Asset* yang ada pada Unity 3D dibagi menjadi dua, yaitu eksternal dan internal. *Asset* eksternal merupakan asset yang ditambahkan dari sumber di luar Unity 3D, seperti 3D *Model, Texture*, dan *Sound Effect*. *Asset* Internal merupakan *asset* yang sudah ada dalam Unity, seperti *Materials*, *Shaders*, *Cube Maps*, *Physics Materials*, dan *Prefabs*.

d. *Platforms* yang didukung antara lain Xbox One, BlackBerry 10, Windows 8, Windows Phone 8, Windows, Mac, Linux, Android, iOS, Unity Web Player, Adobe Flash, PlayStation 3, Xbox 360, Wii U and Wii. Selain itu, Unity 3D juga mendukung PlayStation Vita, meskipun belum ada informasi resmi. Rencananya, Unity 3D versi berikutnya akan mendukung PlayStation 4.

2.8.2 *Fiture* Penting Pada Unity 3D

Fiture penting pada Unity 3D yang perlu diperhatikan pada saat merancang dan membangun, (Andi, 2014):

a. *Scenes* secara sederhana diartikan sebagai level pada *game*. Pada setiap level dapat diletakkan berbagai macam objek. Semakin banyak level yang dibuat maka semakin banyak pula *scenes* yang dibutuhkan. *Scenes* yang sudah dibuat akan ditampilkan pada jendela *Project* bagian *Assets*.

b. *Packages* merupakan kumpulan *asset* yang sudah dijadikan satu. Melalui *packages* ini dapat berbagi *asset* dengan pengguna *Unity* 3D lain. Beragam *packages* dapat diunduh di situs Unity.

c. *Prefabs* merupakan sebuah kontainer atau sebagai salah satu cara untuk membuat group *asset* sehingga dapat digunakan berkali-kali di dalam sebuah *project*. *Prefabs* juga dianggap sebagai simbol, tidak hanya materi level design saja yang dapat dijadikan *prefabs*, tetapi gabungan *script* juga dapat dijadikan *prefabs*.

d. *Game object* merupakan tempat membuat level *game* berisi berbagai macam script yang digunakan untuk membuat karakter atau peta. Melalui *game object* ini dapat menentukan cara kerja dari *game* yang dibuat. Ciri-ciri *game object*:

   1) Objek dapat dipindahkan (*Move*), diputar (*Rotated*), dan mengubah ukuran (*Scale*).
   2) Objek dapat diberikan nama.
   3) Mempunyai sifat *Hierarchy* atau *Link.*
   4) Objek dapat didefinisikan melalui *component*.

e. *Component* adalah grup dari suatu fungsi yang berisikan parameter-parameter yang didefinisikan seperti apa bentuk ataupun sifat dari *game object*, dengan kata lain, *component* akan mendefinisikan *game object*.

f. *Asset* merupakan bagian-bagian yang akan membentuk *game.* Manfaat dari *fiture asset* yaitu, dapat membuat lingkungan, tokoh, atau pengendalian *game* (*player control*).

g. *Script* merupakan bagian yang dapat digunakan untuk membuat kecerdasan buatan (*artificial intelligence*) yang mengatur bagaimana *game* berjalan. *Script* dapat melengkapi *asset* yang tidak dapat ditemukan saat membuat *game.*

2.9 Blender

Blender adalah *software* untuk membuat *modelling, rendering,* dan *animasi 3D bersifat* gratis *open source.* Blender mendukung seluruh alur kerja 3D seperti *modeling*, *rigging*, animasi, simulasi, *rendering*, *compositing* dan *motion tracking*, bahkan pengeditan video dan pembuatan *game*. Digunakan untuk dikembangkan secara komersial, tetapi sekarang dirilis di bawah *GPL* (*GNU General Public License*). *Blender* sangat cocok digunakan oleh perseorangan maupun oleh studio kecil yang bermanfaat dalam proyek 3D. Berikut kelebihannya, (Hendratman, 2015):

1. *Open Source* pada *blender* merupakan salah satu software *open source*, dimana kita bisa bebas memodifikasi *source code*-nya untuk keperluan pribadi maupun komersial, asal tidak melanggar GNU (*General Public License)* yang digunakan Blender.
2. *Multi Platform* karena sifatnya yang *open source*, Blender tersedia untuk berbagai macam operasi sistem seperti Linux, Mac dan Windows. *File* yang dibuat menggunakan Blender versi Linux tak akan berubah ketika dibuka di Blender versi Mac maupun Windows.

3. *Update* pada Blender dengan status yang *open source*, Blender bisa dikembangkan oleh siapapun. *Update software* ini jauh lebih cepat dibandingkan *software* sejenis lainnya, bahkan dalam hitungan jam, terkadang *software* ini sudah memiliki *update*. *Update*-an tersebut tak tersedia di situs resmi *blender.org* melainkan di *graphicall.org*
4. *Free*, Blender merupakan sebuah *software* yang gratis, Blender gratis bukan karena tidak laku, melainkan karena luar biasanya fitur yang mungkin tak dapat dibeli dengan uang, selain itu dengan digratiskannya software ini, siapapun bisa berpartisipasi dalam mengembangkannya untuk menjadi lebih baik.
5. Lengkap, Blender memiliki fitur yang lebih lengkap dari *software* 3D lainnya. *Software* 3D Blender di dalamnya sudah dilengkapi dengan fitur Video *editing*, *Game Engine*, *Node Compositing*, dan *Sculpting*.
6. Ringan, *Blender* relatif ringan jika dibandingkan *software* sejenis. Hal ini terbuti dengan sistem minimal untuk menjalankan *Blender*.
7. Komunitas Terbuka Tidak perlu membayar untuk bergabung dengan komunitas Blender yang sudah tersebar di dunia. Pengguna *newbie* sampai yang sudah *advance* terbuka untuk menerima masukan dari siapapun, selain itu mereka juga saling berbagi tutorial dan file secara terbuka. Salah satu contoh nyatanya adalah *open movie* garapan *Blender Institute*.

2.10 Android

Android adalah sistem operasi untuk telepon seluler yang berbasiskan Linux. Android menyediakan *platform* terbuka (*open source*) bagi para pengembang untuk menciptakan aplikasi mereka sendiri sehingga dapat digunakan oleh peranti penggerak. (Firdan, 2011). Awalnya Google Inc. membeli Android Inc. pendatang baru yang membuat *software* (perangkat lunak) untuk telepon genggam. Kemudian untuk mengembangkan Android di bentuklah Open Handset Alliance yang merupakan gabungan dari 34 perusahaan peranti keras, peranti lunak dan telekomunikasi termasuk Google, HTC, Intel, Motorola, Qualcomm, TMobile, dan Nvidia. (Firdan, 2011).

Perilisan perdana Android pada tanggal 5 november 2007, Android bersama Open Handset Alliance menyatakan mendukung pengembangan standar terbuka pada perangkat seluler. Google merilis kode-kode Android dibawah lisensi Apache, sebuah lisensi perangkat lunak dan standar terbuka perangkat seluler. Terdapat dua jenis distributor sistem operasi Android. Pertama yang dapat dukungan penuh dari Google atau *Google Mail Service* (*GMS*) dan kedua adalah yang benar-benar bebas distribusinya tanpa dukungan langsung dari Google atau dikenal sebagai *Open Handset Distribution (OHD)*. (Firdan, 2011).

2.10.1 Arsitektur Android

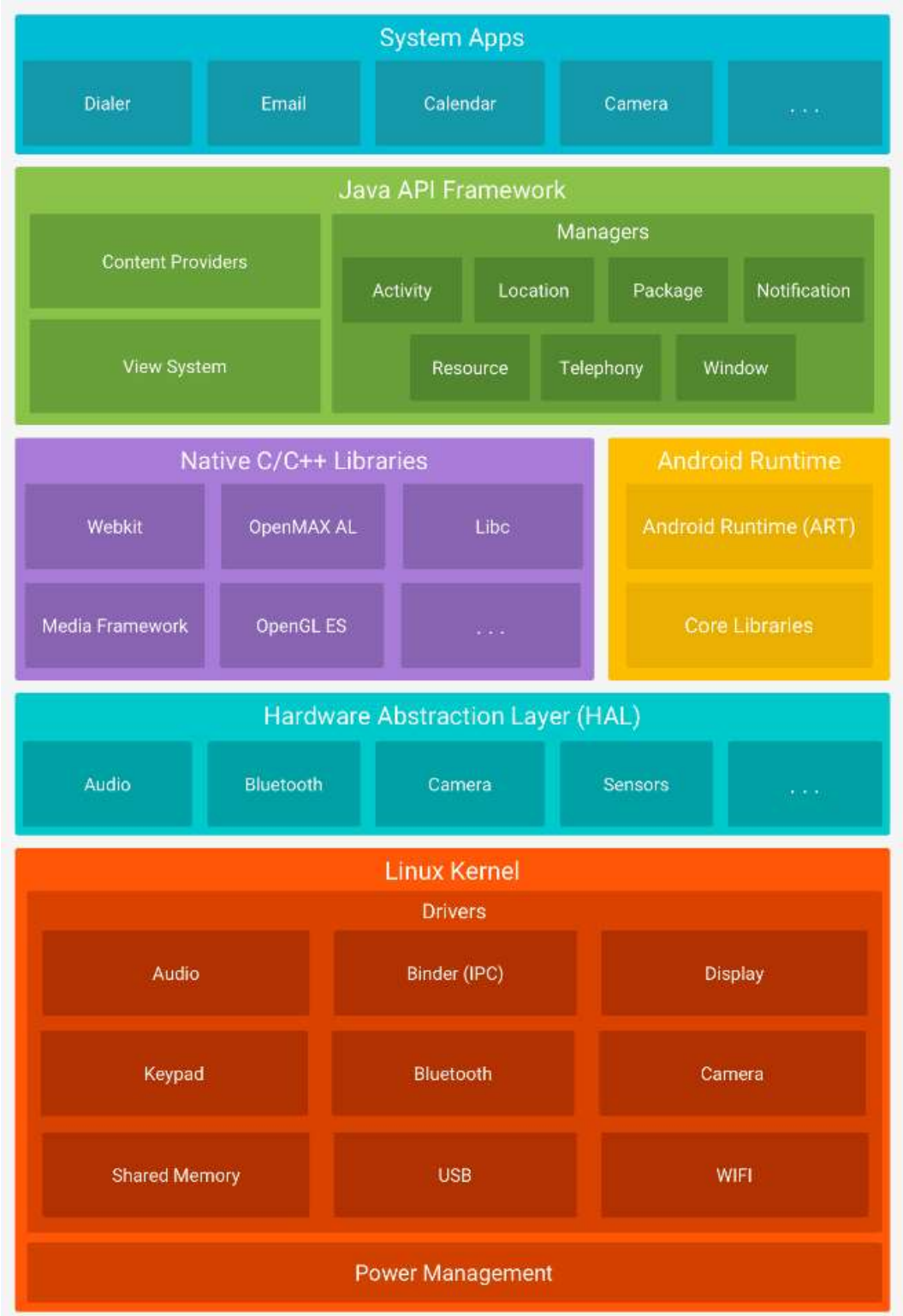
Android adalah tumpukan perangkat lunak berbasis Linux sumber terbuka yang dibuat untuk berbagai perangkat dan faktor bentuk. Diagram berikut menunjukkan komponen besar dari platform Android.

a. Linux *Kernel*

Pondasi *platform* Android adalah *kernel* Linux. Android *Runtime* (ART) bergantung pada *kernel* Linux untuk fungsionalitas dasar seperti *threading* dan manajemen memori tingkat rendah. Menggunakan *kernel* Linux memungkinkan Android untuk memanfaatkan fitur keamanan inti dan memungkinkan produsen perangkat untuk mengembangkan *driver* perangkat keras untuk kernel yang cukup dikenal.

b. *Hardware Abstraction Layer (HAL)*

Hardware Abstraction Layer (HAL) menyediakan antarmuka standar yang mengekspos kemampuan perangkat keras di perangkat ke kerangka kerja Java API yang lebih tinggi. *HAL* terdiri atas beberapa modul pustaka, masing-masing mengimplementasikan antarmuka untuk komponen perangkat keras tertentu, seperti modul kamera atau bluetooth. Bila API kerangka kerja melakukan panggilan untuk mengakses perangkat keras, sistem Android memuat modul pustaka untuk komponen perangkat keras tersebut.



Gambar 2.1 Tumpukan perangkat lunak Android.

c. Android *Runtime*

Perangkat yang menjalankan Android versi 5.0 (API level 21) atau yang lebih tinggi, setiap aplikasi menjalankan proses masing-masing dengan tahap Android *Runtime* (ART). ART ditulis guna menjalankan beberapa mesin *virtual* pada perangkat bermemori rendah dengan mengeksekusi *file* DEX, format *bytecode* yang didesain khusus untuk Android yang dioptimalkan untuk *footprint* memori minimal. Jack akan mengumpulkan sumber Java ke bytecode DEX, yang dapat berjalan pada platform Android.

Beberapa fitur utama ART mencakup:

1) Kompilasi mendahului waktu (AOT) dan tepat waktu (JIT)
2) Pengumpulan sampah (GC) yang dioptimalkan
3) Dukungan *debug* yang lebih baik, mencakup *profiler* sampling terpisah, pengecualian diagnostik mendetail dan laporan kerusakan dan kemampuan untuk mengatur titik pantau guna memantau bidang tertentu.

Sebelum Android versi 5.0 (API level 21), *Dalvik* adalah waktu proses Android. Jika aplikasi berjalan baik pada ART, semestinya berfungsi baik juga pada *Dalvik*, tetapi mungkin tidak sebaliknya.

Android juga menyertakan serangkaian pustaka waktu proses inti yang menyediakan sebagian besar fungsionalitas bahasa pemrograman Java, termasuk beberapa fitur bahasa Java 8, yang digunakan kerangka kerja Java API.

d. *Native C/C++ Libraries*

Banyak komponen dan layanan sistem Android inti seperti ART dan *HAL* dibuat dari kode asli yang memerlukan pustaka asli yang tertulis dalam C dan C++.

Platform Android memungkinkan kerangka kerja Java API mengekspos fungsionalitas beberapa pustaka asli pada aplikasi. Misalnya, mengakses OpenGL ES melalui kerangka kerja Java OpenGL API Android guna menambahkan dukungan untuk menggambar dan memanipulasi grafik 2D dan 3D pada aplikasi Anda. Pengembangan aplikasi yang memerlukan kode C atau C++, bisa menggunakan Android NDK untuk mengakses beberapa pustaka *platform* asli langsung dari kode asli.

e. Kerangka Kerja Java API

Keseluruhan rangkaian fitur pada Android OS tersedia untuk Anda melalui API yang ditulis dalam bahasa Java. API ini membentuk elemen dasar yang Anda perlukan untuk membuat aplikasi Android dengan menyederhanakan penggunaan kembali inti, komponen dan layanan sistem modular, yang menyertakan berikut ini:

1) Tampilan Sistem yang kaya dan luas bisa Anda gunakan untuk membuat UI aplikasi, termasuk daftar, kisi, kotak teks, tombol, dan bahkan browser web yang dapat disematkan
2) Pengelola Sumber Daya, memberikan akses ke sumber daya bukan kode seperti string yang dilokalkan, grafik, dan file layout
3) Pengelola Notifikasi yang mengaktifkan semua aplikasi guna menampilkan lansiran khusus pada bilah status
4) Pengelola Aktivitas yang mengelola daur hidup aplikasi dan memberikan back-stack navigasi yang umum

5) Penyedia Materi yang memungkinkan aplikasi mengakses data dari aplikasi lainnya, seperti aplikasi Kontak, atau untuk berbagi data milik sendiri

Developer memiliki akses penuh ke API kerangka kerja yang sama dengan yang digunakan oleh aplikasi sistem Android.

f. Aplikasi Sistem

Android dilengkapi dengan serangkaian aplikasi inti untuk *email*, perpesanan SMS, kalender, menjelajahi internet, kontak, dan lain sebagainya. Aplikasi yang disertakan bersama platform tidak memiliki status khusus pada aplikasi yang ingin dipasang pengguna. Aplikasi pihak ketiga dapat menjadi browser web utama, pengolah pesan SMS atau bahkan keyboard utama (beberapa pengecualian berlaku, seperti aplikasi *Settings* sistem).

Aplikasi sistem berfungsi sebagai aplikasi untuk pengguna dan memberikan kemampuan kunci yang dapat diakses oleh *developer* dari aplikasi mereka sendiri. (Developer, 2018).

2.10.2 Android *Software Development Kit (SDK)*

Android *SDK* adalah *Software Development Tools* yang berisi dari sekumpulan *API libraries* serta *tools-tools* yang dibutuhkan untuk membuat aplikasi *Android*. *Android SDK* berbasiskan bahasa pemrograman java, sehingga untuk menggunakan *Android SDK*, *java* harus sudah terinstall pada *PC*. Perlu diingat *Android SDK* bukan *tools* untuk membuat aplikasi, melainkan hanya sekumpulan *API libraries*, maka dari itu dibutuhkan *Integrated Development*

Environment (IDE) yang dapat terintegrasi dengan Android *SDK* untuk membuat sebuah aplikasi. (Sulihati, 2016).

2.10.3 *Java Development Kit (JDK)*

Java Development Kit (JDK) adalah sekumpulan perangkat lunak yang dapat digunakan untuk mengembangkan perangkat lunak yang berbasis *Java*, sedangkan JRE adalah sebuah implementasi dari *Java Virtual Machine* yang benar-benar digunakan untuk menjalankan program *java*. Biasanya, setiap *JDK* berisi satu atau lebih JRE dan berbagai alat pengembangan lain seperti sumber compiler *java*, *bundling*, *debuggers*, *development libraries* dan lain sebagainya. (Nyura, 2010).

2.11 *State of The Art*

Jurnal yang berjudul "Penentuan Pergerakan *Non-Player Character* Menggunakan Algoritma *A Star* Pada *Game Action- Role-Playing Game*" yang ditulis oleh Atthariq berisi tentang penelitian yang melakukan percobaan pengujian terhadap algoritma *A Star* pada lingkungan sebuah *game.* Game adalah salah satu bentuk dari animasi interaktif dimana player dapat berinteraksi dengan dunia game. Unsur yang dapat dianggap penting untuk mendukung jalannya game dan realitas dari dunia game adalah *NPC (Non-Player Character). NPC* dapat membuat sebuah game menjadi lebih nyata dari segi cara Perpindahannya, maka dibutuhkan suatu algoritma pathfinding yang mampu membuat NPC tersebut melakukan perpindahan layaknya suatu makhluk hidup berpindah di dunia nyata.

A (A Star)* adalah algoritma pencarian yang dapat digunakan untuk melakukan pathfinding, dalam hal ini *A Star* akan digunakan untuk mencari suatu jarak terpendek antara NPC dan karakter player. Berdasarkan hasil dari implementasi dan pengujian yang dilakukan peneliti, maka dapat ditarik kesimpulan bahwa penggunaan algoritma *A Star* berhasil di terapkan pada game Action-RPG sebagai pembangkit perilaku pencarian pada NPC, dan NPC mampu berjalan dan melewati halangan yang ada, dan berhasil menemukan keberadaan karakter utama atau player. (Atthariq, 2017).

Penelitian yang dilakukan oleh Ida Bagus Gede Wahyu Antara Dalem yang dijabarkan dalam jurnal yang berjudul "Penerapan Algoritma A* (*A Star*) Menggunakan *Graph* Untuk Menghitung Jarak Terpendek" menyatakan bahwa Algoritma *A Star* merupakan salah satu algoritma yang termasuk dalam kategori metode pencarian yang memiliki informasi (*informed search method*). Algoritma ini sangat baik sebagai solusi proses *pathfinding* (pencari jalan). Algoritma ini mencari jarak rute terpendek yang akan ditempuh suatu *point* awal (*starting point*) sampai ke objek tujuan. Teknik pencarian yang digunakan dalam simulasi ini adalah menggunakan Algoritma *A Star* dengan fungsi heuristic. Tujuan utama penelitian ini mempelajari cara kerja algoritma *A Star* dalam mencari jarak tercepat, yang disimulasikan seperti kondisi ketika seorang mencari rute dalam keadaan jalanan macet. Simulasi ini memberikan gambaran yang lebih realistis terhadap perilaku algoritma *A Star* dalam pencarian jarak rute terpendek. Kesimpulan yang didapat dari penelitian ini

adalah penggunaan algoritma *A Star* dapat dijalankan dengan baik dimana *A Star* mampu mencari rute tercepat. (Dalem, 2018).

Paulus Harsadi menuliskan dalam jurnalnya yang berjudul "Pathfinding Pada Lingkungan Statis Berdasarkan Artificial Potential Field Dengan Flocking Behavioruntuk Non-Player Character Follower Pada Game" tentang Artificial Intelligence (AI) di dalam video game merupakan hal penting untuk memberikan tantangan kepada pemain, salah satunya membuat karakter didalam video game seperti perilaku manusia atau hewan sesungguhnya. Beberapa masalah yang muncul dalam penerapan AI pada sebuah game meliputi learning, planning, natural language processing, dan pathfinding.Pathfinding merupakan salah satu yang paling banyak dan umum diteliti dari pada teknik yang lain dikarenakan pathfinding merupakan teknik dasar yang wajib dan paling banyak diaplikasikan dalam sebuah video game. Penelitian ini dilakukan untuk mengurangi waktu komputasi dalam permasalahan pathfinding dengan penerapan Artificial Potential Field (APF) yaitu fungsi attractive potential field saja disebut Improvedattractive potential field untuk bergerak ketujuan dan memanfaatkan pergerakan kawanan boid untuk menghindari halangan dalam lingkungan statis, sehingga dapat mempercepat waktu untuk menuju ketujuan. b. Simulasi diterapkan dalam lingkungan game dan hasil dari pengujian didapat bahwa algoritma pathfinding yang diusulkan lebih unggul dibanding dengan algoritma *A Star pathfinding*, unggul selisih 1.4 detik. (Harsadi, 2015).

Ryan Mahendra Kusuma Putra dalam jurnalnya yang berjudul "Perancangan Game First Person Shooter 3D "Zombie Hunter" dengan Menggunakan Metode

A Star" berisi tentang perancangan *game* 3D ber-*genre FPS* dengan memanfaatkan algoritma *A Star* sebagai pergerakan musuh dalam *game. Video Game* adalah sebuah permainan dengan tampilan sebuah gambar atau visual yang dapat memberikan respon balik jika diberikan perintah-perintah tertentu menggunakan alat kontrol pada seperangkat sistem elektronik. *Game First Person Shooter (FPS)* adalah salah satu jenis game *(genre)* yang digemari menggunakan pandangan orang pertama di mana pemain seolah-olah menjadi karakter utama dalam game yang berpusat pada permainan di sekitar senjata-senjata dan peluru tempur. Selanjutnya penerapan *game FPS* yang menceritakan tentang seorang pemburu zombie ini menggunakan *Game Engine* Unity 3D baik pada saat kondisi lingkungan dan karakter. Linngkungan dimana sebuah hotel serta karakter-karakter di dalam *game* yang diujicobakan pada *Game Engine Unity 3D* menandakan tingkat keberhasilan sesuai dengan keadaan dalam bentuk modeling. Aplikasi hanya dapat berjalan pada perangkat desktop dengan sistem operasi Windows, dengan penerapan algoritma *A Star* didalam permainan, dapat membuat karakter musuh menemukan posisi karakter utama atau *player* secara cepat melalui jalur terpendek tanpa menabrak penghalang. (Putra R. M., 2015).

I Putu Agus Edy Saputra menuliskan dalam jurnalnya yang berjudul "Optimasi Lintasan Game Mekepung 3d Pada Engine Unity3d" berisi tentang media promosi budaya, peranan game sangatlah penting, dikarenakan banyak produsen game dari Asia bahkan Eropa yang terkenal secara tidak langsung mempromosikan budaya mereka dalam game yang mereka buat, maka dari itu tidak heran bahwa anak Indonesia khususnya Bali banyak mengenal budaya luar

ketimbang budaya lokal, karena pengaruh game yang di mainkan anak-anak tersebut bernuansa budaya dari produsen game yang membuatnya. Penelitian ini merancang game yang memberikan informasi bagaimana dan apa sebenarnya permainan tradisional bali "Makepung". *Game* ini dirancang berbasis mobile dan desktop dengan bahasa pemrograman C# yang menggunakan unity3D untuk menggabungkan bahasa pemrograman dan model 3D supaya menjadi sebuah game. *Game* "Makepung" ini diimplementasikan menggunakan algoritma *A* (A Star)* serta menerapkan konsep looping (perulangan) pada lintasan agar sistem pada game lebih sedikit merender object dibandingkan harus menaruh lintasan full di dalamnya. Lintasan ini di bagi menjadi beberapa bagian lalu ketika player melewati trigger yang ada di lintasan maka bagian dari lintasan lainnya diciptakan. Dari hasil penelitian yang telah dilakukan bahwa penerapan algoritma pencarian A*(star) dalam game "Makepung" ini mampu memecahkan permasalahan bagaimana lawan menghindar melewati rintangan. Optimasi pada lintasan dengan cara membagi lintasan menjadi beberapa bagian mampu membuat game berjalan bagus dibandingkan pada saat lintasan tidak dibagi, dikarenakan game sistem merender sedikit object ketika lintasan tersebut dibagi. (Saputra, 2015).

Penelitian yang dilakukan oleh Yoppi Sazaki dalam jurnalnya yang berjudul "Analisa Perbandingan Algoritma *A Star* Dan Dynamic Pathfinding Algorithm Dengan Dynamic Pathfinding Algorithm Untuk Npc Pada Car Racing Game" berisi tentang permainan mobil balap adalah salah satu permainan simulasi yang membutuhkan *Non-Playable Character (NPC)* sebagai pilihan lawan bermain

ketika pemain ingin bermain sendiri. Permainan mobil balap, *NPC* membutuhkan pathfinding untuk bisa berjalan di lintasan dan menghindari hambatan untuk mencapai garis finish. Metode pathfinding yang digunakan oleh *NPC* dalam game ini adalah *Dynamic Pathfinding Algorithm (DPA)* untuk menghindari hambatan statis dan dinamis di lintasan dan Algoritma *A Star* yang digunakan untuk mencari rute terpendek pada lintasan. Hasil percobaan menunjukkan bahwa NPC yang menggunakan gabungan DPA dan Algoritma *A Star* mendapatkan hasil yang lebih baik dari NPC yang hanya menggunakan Algoritma DPA saja, sedangkan posisi rintangan dan bentuk lintasan memiliki pengaruh yang besar terhadap *DPA*. (Sazaki, 2018).

Penelitian yang dilakukan oleh Wandah Wibawanto dalam jurnalnya yang berjudul "Metode *Trigger Detection* Untuk Gerakan Kendaraan NPC Dalam Game" membahas tentang penggunaan metode *trigger detection* untuk membuat simulasi lalulintas yang melibatkan kendaraan *NPC (Non Player Character)* untuk keperluan *game*. Sebuah *game* bertipe *racing* atau *open world* seperti *Need For Speed, Grand Theft Auto, Watch Dog* dan sejenisnya, simulasi kendaraan *NPC* diperlukan untuk menghasilkan kesan realistis. Membuat alogaritma gerakan kendaraan *NPC* pada umumnya digunakan metode *pathfinding* dan metode *waypoint.* Alternatif yang dapat digunakan adalah metode *trigger detection*, yaitu dengan menempatkan sejumlah sensor di sekeliling kendaraan. Selanjutnya sensor akan mendeteksi tumbukan antara sensor dengan objek yang berada di lingkungan virtual dalam game. Hasil dari tumbukan tersebut diolah lebih lanjut melalui proses posisi, deteksi, reaksi dan negosiasi agar menghasilkan gerakan yang

dinamis. Metode ini selanjutnya diujicoba untuk mengetahui keefektifannya. Metode tersebut telah diujicobakan dalam 2 bentuk simulasi : (1) simulasi dengan metode artbased dengan melibatkan 12 kendaraan *NPC*, dimana simulasi menghasilkan sebuah sistem lalulintas sederhana, kendaraan *NPC* bergerak secara dinamis dan tidak bertabrakan sepanjang proses ujicoba, (2) simulasi lalu lintas dengan metode tilebased dalam game *Ace Gangster 2* dengan melibatkan 40 kendaraan *NPC* pada lingkungan virtual 100 x 100 grid. Kedua uji coba tersebut menunjukkan bahwa pemakaian metode *trigger detection* dapat dilakukan dengan sederhana, menggunakan sedikit *memory* dan menghasilkan simulasi lalulintas yang dinamis/realistis. (Wibawanto, 2017).

Jurnal yang berjudul "Penerapan Algoritma *A Star Pathfinding* dalam Pencarian Solusi *Game Peanut* Labirin dengan Makromedia Flash" yang ditulis oleh Hernika Ari Wibowo berisi tentang penggunaan algoritma *A Star* dalam pencarian jalur terpendek yang dibutuhkan untuk menuju ke titik tujuan dengan map yang dibuat dalam bentuk labirin. Kesimpulan dari penelitian yang dilakukan dalam jurnal ini adalah penggunaan algoritma *A Star* dapat menyelesaikan permasalahan dalam pencarian jalur pada map labirin. (Wibowo, 2015).

Moh Zikky menjelaskan dalam jurnalnya yang berjudul "*Review of A Star (A Star) Navigation Mesh Pathfinding as the Alternative of Artificial Intelligent for Ghosts Agent on the Pacman Game*" bahwa masalah pencarian jalan terpendek telah menjadi masalah yang populer di *Game's Artificial Intelligence (AI)*. Beberapa teori telah disampaikan untuk menyelesaikan masalah *pathfinding*, seperti *Finite State Machine (FSM), Graph, Dijkstra*, dan *A Star*. *A Star* adalah

algoritma pencarian generik yang dapat digunakan untuk menemukan solusi untuk banyak masalah, dan pathfinding adalah salah satunya. Algoritma *A Star* berulang kali memeriksa lokasi yang belum dijelajahi yang paling menjanjikan yang pernah dilihatnya. Ketika lokasi yang dieksplorasi adalah tujuannya, algoritma itu selesai. Jika tidak, ia telah mencatat semua lokasi di sekitar untuk eksplorasi lebih lanjut. *Navigation mesh (Navmesh)* telah menjadi konsep populer yang digunakan dalam masalah penelusuran jalan terpendek dari game 3D, karena lingkungan 3D sebagian besar menggunakan struktur poligon. *Navmesh* properti dari objek poligon atau medan dapat menjamin berjalan bebas untuk karakter permainan selama karakter yang berada di dalam poligon yang sama adalah seperangkat poligon kompleks. (Zikky, 2016).

2.12 Matrik Penelitian

Tabel 2.2 Matrik Penelitian

Peneliti	Judul	Metode			Pathfinding					Genre				
		Deskriptif	Eksperimental	D&C	A Star	IAPF	Trigger Detection	DPA	Navmesh	FPS 3D	Puzzle	Racing	RPG	Simulasi
Attariq, 2018	Penentuan Pergerakan *Non-Player Character* Menggunakan Algoritma *A Star* Pada *Game Action-Role-Playing Game*	✓			✓			✓					✓	
Dalem, 2018	Penerapan Algoritma A* (*A Star*) Menggunakan Graph Untuk Menghitung Jarak Terpendek	✓			✓									✓
Harsadi, 2015	Pathfinding Pada Lingkungan Statis Berdasarkan Artificial Potential Field Dengan Flocking Behavioruntuk Non-Player Character Follower Pada Game		✓			✓				✓				
Putra, 2015	Perancangan Game First Person Shooter 3D "Zombie Hunter" dengan Menggunakan Metode *A Star*	✓			✓					✓				
Saputra, 2015	Optimasi Lintasan Game Mekepung 3d Pada Engine Unity3d	✓			✓				✓			✓		
Sazaki, 2017	Analisa Perbandingan Algoritma *A Star* Dan Dynamic Pathfinding Algorithm Dengan Dynamic Pathfinding Algorithm Untuk Npc Pada Car Racing Game	✓			✓			✓				✓		

Tabel 2.2 Matrik Penelitian (Lanjutan 1)

Wibawaryo, 2016	Metode *Trigger Detection* Untuk Gerakan Kendaraan NPC Dalam Game	✓					✓							✓
Wibowo, 2015	Penerapan Algoritma *A Star Pathfinding* dalam Pencarian Solusi *Game Peanut* Labirin dengan Makromedia Flash	✓			✓						✓			
Zikky, 2016	*Review of A Star (A Star) Navigation Mesh Pathfinding as the Alternative of Artificial Intelligent for Ghosts Agent on the Pacman Game*	✓			✓				✓					✓
Penelitian yang dilakukan				✓					✓	✓				

Tabel 2.2 menunjukkan matrik penelitian yang telah dilakukan terkait mengenai *game* yang menerapkan *pathfinding* di dalamnya. Keterbaruan dari penelitian dengan judul "**Implementasi *Artificial Intellegence* menggunakan *Pathfinding* pada *Game FPS* 3D *Robots Invasion***" adalah melakukan perbandingan penggunaan *pathfinding* antara algoritma *A Star* dan *Navmesh* pada karakter *NPC* dalam *game* berbasis Android yang mengusung *genre FPS 3D*. Penggunaan metode penelitian *design and creation* didasari pada proses penelitian yang dilakukan, yaitu proses pengembangan produk dan keilmuan dibidang *IT*. Metode penelitian ini meliputi penganalisaan, perancangan dan pengembangan produk berbasis komputer seperti website, animasi computer. Penelitian harus mempunyai kualitas akademik seperti analisis, keterangan, argumen, justifikasi dan evaluasi, serta berkontribusi pada ilmu pengetahuan.