BAB II

TINJAUAN PUSTAKA

2.1. Teori Dasar yang Digunakan

2.1.1. *Game*

Game berasal dari bahasa inggris yang artinya permainan. Permainan adalah sesuatu yang digunakan untuk bermain yang dimainkan tergantung dari aturan yang telah ditentukan. Permainan yang menggunakan media elektorik merupakan suatu hiburan dalam bentuk multimedia yang dibuat semenarik mungkin secara visual dan audio sehingga mendapatkan kepuasan batin .

Berdasarkan Jenisnya, *game* dibagi menjadi beberapa jenis salah satunya yaitu *Stealth Game*. Dalam jenis game ini pemain akan melakukan misinya dengan cara sembunyi sembunyi dan sebisa mungkin tidak menarik perhatian kawanan musuh disekitarnya.

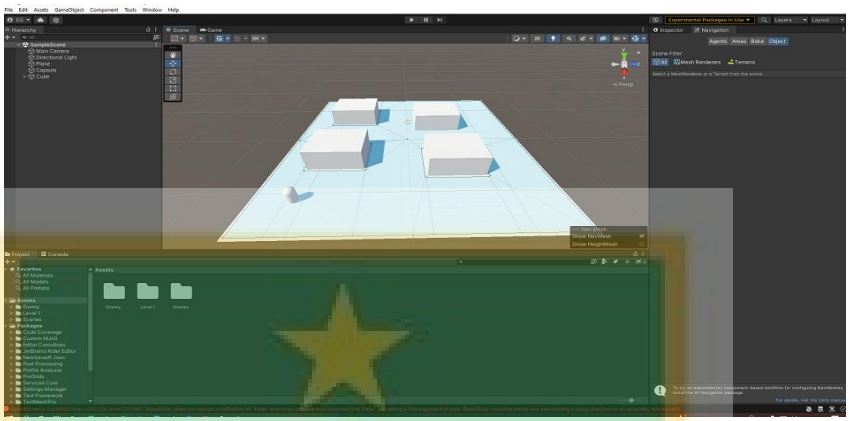
2.1.2. *Game Design Document*

Game Design adalah Tindakan memutuskan seperti apa seharusnya sebuah *game*. Dalam *game design* biasanya untuk memutuskan *game* apa yang akan dibuat dan menghasilkan banyaknya keputusan. Keputusan yang dimaksud antara lain adalah seperti keputusan tentang rules, look and feel, timing, pacing, risk-taking, rewards, punishment dan segala suatu yang dialami *player* adalah tanggung jawab dari *game designer*. Hal tersebut memudahkan *game developer* dalam pembuatan sebuah *game* (Schell, n.d.).

2.1.3. *Unity Engine*

Unity engine merupakan sebuah *software game engine* yang digunakan untuk mengembangkan *game* berbasis 3D atau 2D. Meskipun mempunyai bahasa pemograman dasar C#. *Unity Engine* dapat mendukung berbagai macam bahasa pemograman, mulai dari C++, *javascript*, hingga *Phyton*. Peneliti akan menggunakan Bahasa pemograman C#. *Unity Engine*

juga mendukung berbagai macam *platform* mulai dari Android, IOS, PS3, PS4, PS5, XBOX, dan komputer berbasis Windows (Goldstone, 2009). Gambar 2.1 merupakan tampilan dasar dari *unity engine*.

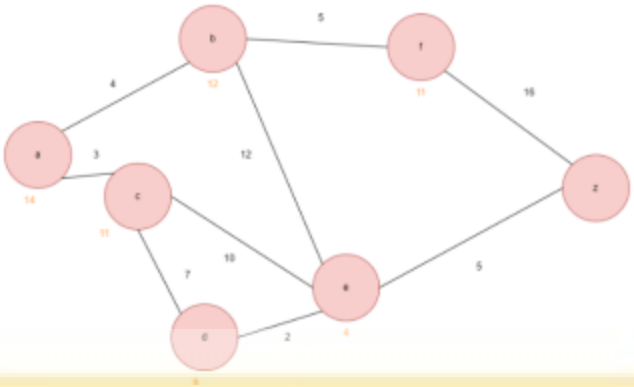


Gambar 2.1 merupakan tampilan dari software unity engine

Gambar 2.1 merupakan tampilan Unity versi 2021.3.12f1 menampilkan *Scene*, *Hierarchy*, *Inspector*, *Game*, dan *Project*. *Scene* pada *unity* untuk menampilkan *game* yang akan dibuat. *Hierarchy* adalah *list* dari *game object* pada tampilan *scene*. *Game* untuk jendela pratinjau yang akan aktif ketika dalam *play mode*. Dan terakhir pada *project* adalah daftar *project asset* yang berfungsi sebagai *library* pada suatu *project*(Goldstone, 2009).

2.1.4. *Algoritma A**

Algoritma A* merupakan teknik algoritma yang paling popular untuk digunakan karena akurasi serta performanya. Algoritma ini digunakan untuk mencari tahu jalan diantara kedua node yang terpendek. Algoritma ini telah digunakan pada banyak genre video game seperti RTS (Real Time Strategy) RPG(Role-Playing Games), Racing game, serta turn based strategy games. Algoritma ini di kembangkan pertama Kali oleh Hart, Nilsson dan Raphael pada 1967 untuk menyelesaikan banyak masalah. Dalam context Video game. Algoritma ini kerap digunakan dan diimplementasikan pada NPC (Non Player Character) AI untuk menuntun mencari jalan atau untuk menangkap player (Darmawan & Rubhasy, n.d.). Visualisasi dari algoritma A* dapat dilihat pada gambar 2.2 sebagai berikut:



Gambar 2.2 merupakan visualisasi pathfinding A* dari node a ke node z.

Angka yang dinyatakan dengan warna oranye merupakan angka heuristic dan angka yang ditandai dengan warna hitam merupakan angka bobot. Algoritma A* termasuk kedalam algoritma heuristis karena karakteristiknya untuk terus mencari nilai paling kecil untuk memberikan solusi yang paling optimal. A* menggunakan fungsi heuristic f(n) untuk menentukan node, dan nilai dari f(n) didapatkan dengan rumus (Safira et al., 2021).

f(n) = g(n) + h(n) …(1.)

Keterangan :

f(n) : fungsi dari evaluasi

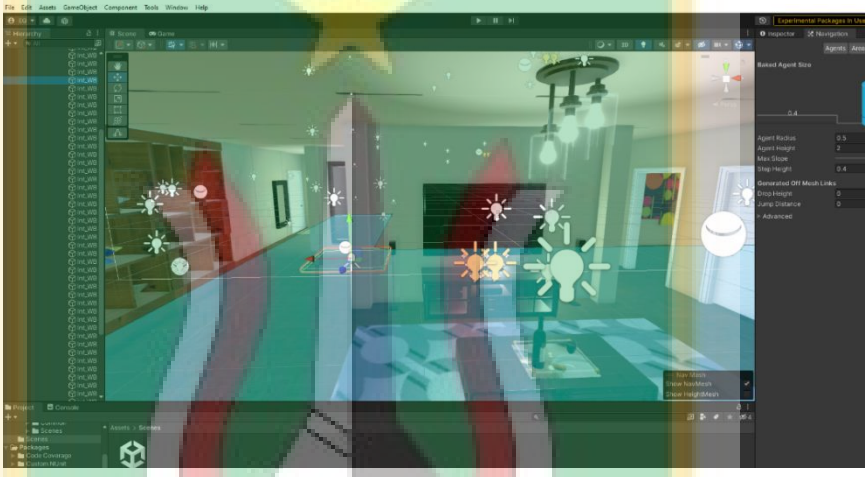
g(n) : biaya yang telah dikeluarkan dari keadaan awal hingga node (n)

h(n) : estimasi dari biaya yang dikeluarkan dari keadaan node (n) hingga sampai tujuan

Ketika grid mempunyai hambatan f(n) akan mengambil estimasi dan mengambil harga paling rendah untuk mendapatkan hasil yang optimal(Febriyanto Riski et al., 2020).

2.1.5. *Navigational Mesh*

Navmesh (*Navigation Mesh*) merupakan algoritma *pathfinding* yang sudah terintegrasikan oleh *Unity Engine*. *Navmesh* terdiri dari beberapa komponen yang berbeda, komponen tersebut terdiri dari *navmesh surface*, dan *navmesh agent*. *Navmesh surface* merupakan jalur yang akan di hasilkan secara automatis oleh *navmesh* ketika pengguna melakukan bake pada area yang ingin di jadikan sebagai jalur (Farrel et al., 2022). Berikut visualisasi implementasi dari *Navmesh* dapat dilihat pada gambar 2.3 berikut :

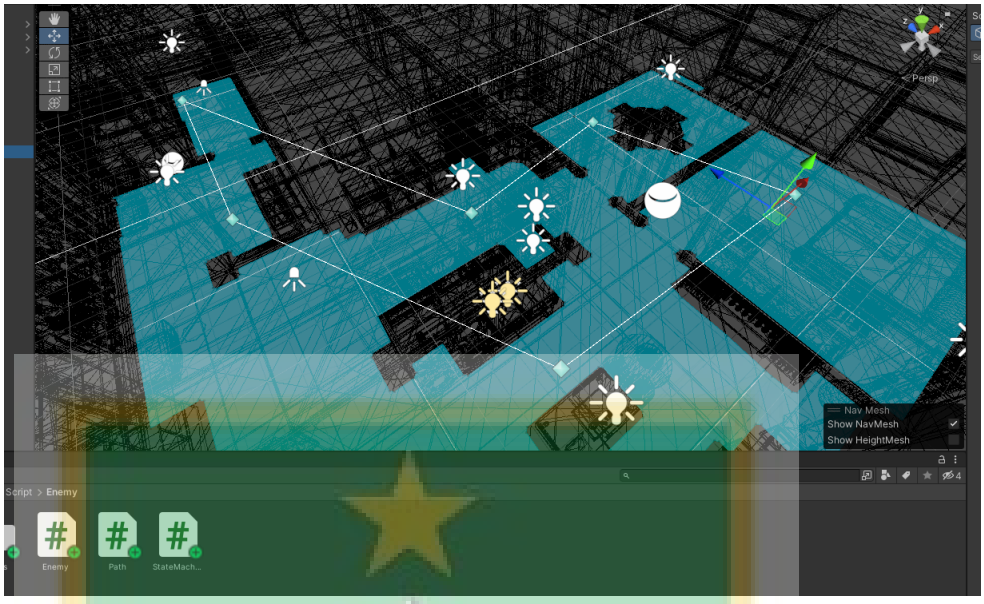


Gambar 2.3 tampilan *polygon map* yang sudah di generate oleh navmesh

Gambar 2.3 adalah Tampilan dari menampilkan *polygon map* yang sudah di *generate* oleh *navmesh* yang berfungsi untuk pijakan yang akan dilalui oleh AI NPC.

2.1.6. *Waypoint pathfinding*

Waypoint pathfinding merupakan salah satu penerapan algoritma A* dengan menempatkan penanda atau waypoint yang sudah ditentukan pada saat proses pengembangan game. Algoritma ini bekerja dengan cara pemberian penanda jalan yang nantinya akan dituju oleh NPC AI. Penanda diletakan manual pada level oleh game designer sebelum akhirnya akan di buat script agar NPC AI dapat menuju pada koordinat penanda tersebut (Alvin et al., 2022).

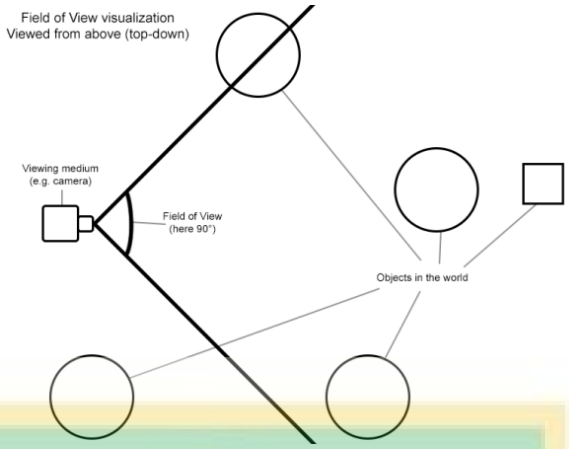


Gambar 2.4 Tampilan implementasi dari waypoint pathfinding

Gambar 2.4 adalah tampilan dari implementasi waypoint pathfinding yang akan dilalui oleh NPC AI. NPC akan melakukan patrol yang sesuai dengan urutan waypoint yang telah ditetapkan oleh game designer. Waypoint tersebut dibuat menggunakan Game Object yang kemudian di letakan pada scene sesuai dengan game design tentukan.

2.1.7. *Field Of View*

Field of view adalah sudut pandang perspektif model kamera yang digunakan untuk menampilkan virtual environment. FoV (Field of view) menunjukan bahwa media tampilan dapat menangkap lingkungan dalam kerucut yang bisa mencapai 90° didepan kita. FoV ini diterapkan ke AI NPC musuh untuk bagian dari mata tersebut (Ljung, 2015). Bayangan dari penggunaan FoV dapat dilihat pada gambar 2.5 dibawah ini.

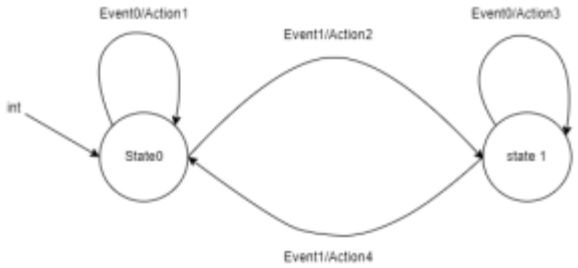


Gambar 2.5 Visualisasi dari *Field of View*

Gambar 2.5 adalah visualisasi dari *Field of View*. Segala sesuatu yang berada didalam kerucut *field of view* tersebut akan di*render* dan terlihat pada final gambar (terlihat kotak kecil yang terhalang oleh lingkaran) yang didalam lingkungan virtual tersebut hanya akan terlihat lingkaran, kotak yang berada dibelakang lingkaran tersebut tidak akan terlihat. Hal tersebut akan di implementasikan pada AI NPC sebagai pengubah *State* pada *Finite State Machine*.

2.1.8. *Finite State Machine*

Finate State Machines (FSM) adalah sebuah metodologi perancangan sistem kontrol yang menggambarkan tingkah laku atau prinsip kerja sistem dengan menggunakan state (Keadaan), event (kejadian) dan action (aksi). Dengan kata lain, system akan memasuki salah satu state atau keadaan ketika mendapatkan input action atau event tertentu (Alvin et al., 2022). Seperti yang tertera pada gambar 2.6 sebagai berikut:



Gambar 2.6 merupakan dasar dari finite state machine

Gambar 2.6 memperlihatkan 2 buah *state* dan dua buah *input* serta empat buah *output* yang berbeda. Ketika *game* akan dijalankan akan bertransisi menuju *State0*, pada keadaan sistem akan menghasilkan *action1* jika terjadi masukan *event0*, sedangkan jika terjadi *event1* maka *action2* akan dieksekusi kemudian sistem akan bertransisi ke keadaan *State1* dan seterusnya.

2.2. Tabel Studi Literatur

Judul	KOMBINASI ALGORITMA A STAR DAN FINITE STATE MACHINE PADA AI ENEMY GAME HORROR 3D ESCAPED
Penulis	Alvin Prasetyo Darmawan, Fauziah dan Albaar Rubhasy
Tahun Penerbitan	2022
Metode atau Algoritma yang digunakan	Menggunakan algoritma A Star untuk Pathfinding waypoint serta menggunakan metode Finite State Machine sebagai otak pada AI tersebut.
Kesimpulan	Keduanya algoritma tersebut dikombinasikan dengan hasil yang baik. Respons time algoritma menunjukan angka paling besar dibawah 5ms dengan besar grid 10000 serta penggunaan resource hanya menggunakan 0,6% power. Dengan kombinasi algoritma pathfinding A* dengan State machine. NPC dapat menjelajahi level yang telah dibuat tanpa bertabrakan dengan asset 3D lainnya yang dianggap sebagai rintangan. NPC juga dapat mengejar pemain dengan secara dynamic ketika pemain bergerak tanpa masalah.
Kesamaan	Menggunakan Algoritma A* sebagai waypoint NPC AI serta menggunakan Finite State Machine
Perbedaan	Menggunakan Algoritma A* sebagai waypoint dengan menambahkan Navigational Mesh sebagai pathfinding dan menambahkan Finite State Machine sebagai otak pada AI NPC.

Judul	IMPLEMENTASI ALGORITMA A*(A-STAR) UNTUK MENENTUKAN RUTE TERPENDEK PADA NPC GAME HEALTHY FOOD
Penulis	NUR KARTIKA OKTAFIANI, Muhammad. Faisal, Hani Nurhayati,
Tahun Penerbitan	2020

Metode atau Algoritma yang digunakan	Membuat NPC atau AI menggunakan Algoritma A* untuk menentukan rute terpendek dari npc menuju posisi player.
Kesimpulan	percobaan waktu tempuh dengan kondisi map normal hasil dimana untuk pergi ke posisi pemain NPC memerlukan waktu sebanyak 32 detik.percobaan waktu tempuh dengan kondisi map diperbesar 1.25 kali pengujian kali ini diperoleh hasil dimana untuk pergi ke posisi pemain NPC memerlukan waktu sebanyak 41 detik.Penerapan algoritma A* mempengaruhi waktu tempuh yang dibutuhkan NPC ke posisi pemain pada game Healthy Food. Hal ini ditinjau dari sepuluh percobaan yang dilakukan dimana rata-rata waktu tempuh yang dibutuhkan menggunakan algoritma A* sebesar 48.4 detik sedangkan waktu tempuh yang dibutuhkan tanpa menggunakan algoritma A* sebesar 97.6 detik, dapat ditarik kesimpulan bahwa penggunaan algoritma A* dapat menjadi solusi untuk memangkas waktu tempuh yang diperlukan NPC untuk mencari rute menuju ke posisi pemain menjadi lebih cepat.
Kesamaan	Menggunakan Algoritma A* untuk waypoint serta pada AI NPC.
Perbedaan	Penggunaan Algoritma yang diterapkan serta penggunaan metodenya.

Judul	KOMBINASI ALGORITMA A* PATHFINDING NAVIGATIONAL MESH DENGAN WAYPOINT PATHFINDING PADA GAME TOWER DEFENSE RISE OF MACHINE
Penulis	Yans Farrel , Fauziah dan Aris Gunaryati
Tahun Penerbitan	2022
Metode atau Algoritma yang digunakan	Membuat algoritma A* dan Navigational Mesh sebagai waypoint dan pathfinding.
Kesimpulan	Hasil yang didapat adalah dari 300 frame data yang terekam, algoritma waypoint memiliki spike pada frame 21 dan memiliki nilai total resource sebesar 0,6% sedangkan pada algoritma kombinasi dari 300 frame yang terekam waypoint pathfinding memiliki spike pada frame 5 yang memiliki nilai total resource sebesar 1,3% dan algoritma navmeh memiliki spike pada frame 158 yang memiliki nilai total resource sebesar 2,8%.Hasil monitoring menunjukan bahwa metode algoritma kombinasi menggunakan resource CPU yang lebihbesar dikarenakan pada proses

	kombinasi penggunaan resource akan di jalankan sebanyak 2 kali karena penggunaan 2 algoritma yang berbeda berjalan secara bersamaan. Hasil tersebut mendapatkan nilai total pemakaian resource pada algoritma kombinasi sebesar 1,3% pada waypoint dan 2,8% pada navmesh. Sedangkan pada aplikasi yang hanya menggunakan algortima metode waypoint saja memiliki total penggunaan resource sebesar 0,6% saja.
Persamaan	Menggunakan Algoritma A* dan Navigational Mesh untuk waypoint serta pathfinding.
Perbedaan	Algoritma yang diterapkan tanpa menggunakan Finite State Machine.

Judul	PERANCANGAN DAN PEMBUATAN SERIOUS GAME SEBAGAI SIMULASI AKTIVITAS BISNIS DAN AKUNTANSI MENGGUNAKAN PENDEKATAN AGENT-BASED MODELLING
Penulis	Angga Ari Wijaya, Saiful Bukhori dan Nelly Oktavia
Tahun Penerbitan	2017
Metode atau Algoritma yang digunakan	Membuat algoritma A* sebagai pathfinding serta menggunakan Finite State Machine
Kesimpulan	Serious game dapat menjadi media alternatif untuk proses belajar dan latihan yang melibatkan berbagai topik, salah satunya adalah bisnis dan akuntansi. Hasil pada penelitian ini dijadikan suatu konsep dasar game bisnis yang dapat menghasilkan transaksi untuk menunjukan siklus akuntansi.
Persamaan	Menggunakan Algoritma A* serta menggunakan metode Finite State Machine
Perbedaan	Algoritma yang diterapkan tanpa menggunakan Navigational Mesh.

2.3. Aplikasi

Dalam hal Aplikasi Game memiliki setiap platform ada perbedaan antara game seluler dan game pc dalam cara penggunaannya, operator, saluran utama, desain game, dan bahkan cara kerja game. Kita tahu bahwa semua game yang ada di platform console, PC maupun mobile, memiliki ciri khas masing-masing yang memiliki kelebihan dan kekurangan. Semua platform bersaing memperebutkan rating demi keberlangsungan platform mereka. Untuk rating, semua platform game mengembangkan game yang lebih mumpuni.(P.S, 2015)