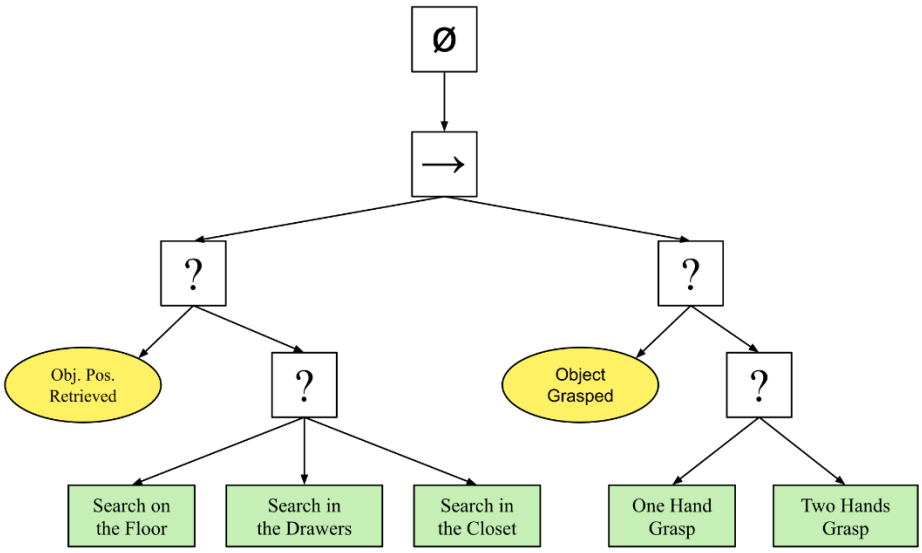
2. LANDASAN TEORI

2.1 Tinjauan Pustaka

2.1.1 Behaviour Tree

Behavior tree adalah suatu model yang berbentuk tree dan isinya terdapat task apa yang akan dilakukan sebuah NPC saat bertemu dengan suatu event. Dalam *Behaviour Tree leaf node* tidak dinamai state lagi melainkan task. Tiap task dalam leaf node bersifat independen sehingga saat mau mengubah atau membuat transisi, syarat yang harus dibetulkan tidak banyak (Florez et al., 2021). Contoh arsitektur *behavior tree* dapat dilihat pada Gambar 2.1.



Gambar 2.1. Arsitektur *Behavior Tree* dalam melakukan pencarian dan juga pengambilan objek.

Sumber : Wikipedia.org (diunduh pada 19 November 2015)

Dalam implementasi *Behavior Tree*, setiap simpul memiliki logika dan aturan tersendiri dan membatasi transisinya untuk tetap berada pada lingkup *state-state* tersebut. Daun pada Behavior Tree mendefinisikan aksi atau perilaku yang digunakan untuk membentuk aksi utama. Berdasarkan fungsinya, simpul pada Behavior Tree dibagi menjadi dua bagian (Faza, 2020):

1. *Leaf*

Simpul terminal yang mempunyai tujuan untuk mengeksekusi bagian paling primitif yang dimiliki oleh *main behavior* dan mengembalikan nilai dari suatu state. Terdapat 2 simpul yang dapat di lihat pada *Behavior Tree*:

a. Simpul Kondisi

Digunakan agar dapat mengecek jika suatu kondisi tertentu telah terpenuhi.

b. SImpul Aksi

Digunakan agar dapat melakukan komputasi terhadap suatu state pada permainan tersebut untuk mengubahnya.

Terdapat 3 nilai yang dapat dikembalikan dari simpul yang dimiliki oleh *Behavior Tree*:

- *Success*

  Syarat-syarat yang diajukan oleh simpul kondisi telah terpenuhi atau eksekusi simpul aksi telah diselesaikan.

- *Failure*

  Syarat-syarat yang diajukan oleh simpul kondisi tidak dapat dipenuhi atau eksekusi simpul aksi tidak dapat diselesaikan.

- *Running*

  Eksekusi simpul aksi sudah dimulai, tetapi program belum dapat memberi kesimpulan untuk hasilnya.

2. *Composite*

Composite dipakai untuk mendefinisikan dan juga mengatur hubungan antara simpul-simpul lainnya, seperti bagaimana atau kapan simpul tersebut harus diproses.

a. *Selector*

Memproses setiap anaknya secara bergantian. Ketika salah satu dari anaknya mengembalikan nilai *success*, maka *selector* akan segera mengembalikan nilai *success* tersebut. Tetapi, apabila anak yang sedang diproses mengembalikan nilai *failure*, maka *selector* akan memproses anak berikutnya hingga tidak ada anak lagi yang tersisa dan mengembalikan nilai *failure*.

b. *Sequence*

Memproses setiap anaknya secara bergantian. Ketika salah satu dari anaknya mengembalikan nilai *failure*, maka *selector* akan segera mengembalikan nilai *failure* tersebut. Apabila anak yang sedang diproses mengembalikan nilai *success*, maka *selector* akan memproses anak berikutnya hingga tidak ada anak lagi yang tersisa dan mengembalikan nilai *success*.

c. *Decorator*

Setiap *decorator* hanya memiliki anak tunggal dan berfungsi untuk memodifikasi *behavior* anak tersebut (*wrapped task*). Tipe-tipe *decorator* diantaranya adalah :

- *Always Fail*

  Selalu mengembalikan nilai *failure*.

- *Always Succeed*

  Selalu mengembalikan nilai *success*.

- *Include*

  Mencantumkan sebuah pohon eksternal.

- *Invert* (*negation*)

  Mengembalikan nilai *success* apabila *wrapped task* gagal dan jika *wrapped task* berhasil akan mengembalikan nilai *failure*.

- *Limit*

  Membatasi berapa kali *wrapped task* dapat diproses.

- *Repeat*

  Mengulangi pemrosesan *wrapped task* sebanyak n kali.

- *UntilFail*

  Mengulangi pemrosesan *wrapped task* hingga *wrapped task* bernilai *failure.*

- *UntilSuccess*

  Mengulangi pemrosesan *wrapped task* hingga *wrapped task* bernilai *success*.

2.1.2 Unreal Engine 5.0.3

Unreal Engine adalah aplikasi pengembangan game (Game Engine) yang dikembangkan oleh Epic Games yang memulai debutnya pada tahun 1998 dengan game bertema first-person shooter (FPS). Unreal Engine sudah menyediakan banyak sekali library yang dapat kita langsung pakai, seperti kata (Torres-Ferreyros, et.al, 2016) manfaat utama menggunakan mesin game seperti Unreal Engine adalah code yang dapat digunakan kembali dengan library, konsep objek pemrograman berorientasi dan proses yang dihasilkan komputer grafis.

Unreal Engine digunakan dalam berbagai genre game tiga dimensi (3D) dan digunakan di industri lain, terutama industri film dan televisi. Ditulis dalam C ++, Unreal Engine menawarkan portabilitas tingkat tinggi dan mendukung berbagai platform desktop, seluler, konsol, dan realitas virtual. *Unreal Engine 5* generasi terbaru diluncurkan pada April 2022.

Dengan kode pemrograman yang ditulis dalam C++, aplikasi ini mampu membawakan kemudahan portabilitas yang tinggi serta merupakan aplikasi yang kerap digunakan berbagai pengembang permainan video kini. Contoh permainan yang dibuat menggunakan *Unreal Engine* meliputi *PUBG, BioShock, Gears of War 3, Borderlands 2*, dan *Batman: Arkham City.*

2.1.3 Survival Game

Survival Games adalah permainan di mana Anda berhadapan dengan lingkungan yang tidak bersahabat di dunia yang umumnya terbuka, seringkali dimulai dengan hanya sedikit peralatan, persediaan, dan inventaris terbatas. Sementara banyak genre video game lainnya menggabungkan elemen *survival*, genre *survival* secara keseluruhan adalah salah satu yang menempatkan elemen *combat* sebagai sekunder dan sering kali dapat dihindari. Permainan bertahan hidup sering kali terbuka, dengan player ditugaskan untuk bertahan hidup selama jangka waktu tertentu atau selama mereka bisa dalam permainan individu itu (Plarium, 2022).

2.2 Tinjauan Studi

2.2.1 Membuat Game 3d Survival Horror "Suanggi Survival Papua" Berbasis Desktop Menggunakan Unity (Aris et al., 2020)

- Masalah yang diangkat adalah membuat *game* dengan *genre horror survival* dari cerita rakyat Papua berdasarkan *urband legend* yang dimiliki oleh cerita rakyat tersebut, yaitu Suanggi.
- Metode yang diusulkan dari penelitian ini adalah menggunakan tahapan metode MDLC untuk pembuatan *game*, dengan menggunakan *game engine* Unity.
- Hasil dari penelitian ini membuktikan kalau menggunakan *game engine* Unity pada penelitian sangatlah efektif. Tetapi masih ada kekurangan yang dimiliki oleh penelitian ini, seperti hanya *support* sistem operasi Windows, tidak ada *multiplayer mode, Third person shooter, online,* dan mungkin ada bug yang tidak diketahui karena dalam pengujian hanya berfokus pada fungsi *program* saja.
- Perbedaan penelitian yang dilakukan dengan skripsi ini dapat dilihat pada Tabel 2.1.

Tabel 2.1

Perbandingan Penelitian 1

Penelitian Sebelumnya	Skripsi ini
Menggunakan metode yang berbeda di mana penelitian ini menggunakan metode MDLC. Selain itu *game engine* yang digunakan dalam pembuatan *game* ini adalah *unity*.	Menggunakan metode *behaviour tree* dalam pembuatan AI NPC *enemy*. *Game engine* yang akan digunakan dalam pembuatan *game* ini adalah *unreal engine 5.*

2.2.2 Implementasi Dynamic Difficulty Adjustment pada Racing Game Menggunakan Metode Behaviour Tree (Isthofi et al., 2019)

- Masalah yang diangkat adalah membuat *game racing* yang di mana akan menerapkan implementasi *Dynamic Difficulty Adjustment* dalam metode *Behaviour Tree*.
- Metode yang diusulkan dari penelitian ini adalah *Behaviour Tree* dan penggunaan sistem *Dynamic Difficulty Adjustment*.
- Hasil dari penelitian ini membuktikan kalau penerapan dari kedua metode tersebut membuat *game* lebih menyenangkan, di mana AI statis akan bertingkah laku seperti *monotone* dan tidak bisa menyesuaikan dengan kemampuan *player*. Sedangkan AI dinamis yang ada dalam *game* ini akan bertingkah laku secara dinamis dan dapat mengimbangi kemampuan *player* dengan baik.
- Perbedaan penelitian yang dilakukan dengan skripsi ini dapat dilihat pada Tabel 2.2.

Tabel 2.2

Perbandingan Penelitian 2

Penelitian Sebelumnya	Skripsi ini
Menggunakan 2 metode yaitu *behaviour tree* dan *fuzzy waypoint tactic*. Selain itu penelitian ini menggunakan *unity* sebagai *physic game engine* dan *visual studio* sebagai *code editor.*	Hanya menggunakan 1 metode saja dalam pembuatan *game* ini yaitu *behaviour tree* dalam pembuatan AI NPC *enemy*. *Game engine* yang akan digunakan dalam pembuatan *game* ini adalah *unreal engine 5.*

2.2.3 Pemanfaatan Behavior Tree dan Fuzzy Waypoint Tactic pada Game Strategi "War of Zombies" (Christie et al., 2020)

- Masalah yang diangkat adalah membuat *game* yang di mana akan diterapkan 2 macam AI pada *game* yang dibuat.
- Metode yang diusulkan dari penelitian ini adalah *Behaviour Tree* dan *Fuzzy Waypoint Tactic*.
- Hasil dari penelitian ini membuktikan kalau AI yang dipakai dalam penelitian ini sangat efektif atau baik dalam mengatur AI yang ada pada *game.*
- Perbedaan penelitian yang dilakukan dengan skripsi ini dapat dilihat pada Tabel 2.3.

Tabel 2.3

Perbandingan Penelitian 3

Penelitian Sebelumnya	Skripsi ini
Menggunakan 2 metode yaitu *behaviour tree* dan *fuzzy waypoint tactic*. Selain itu penelitian ini menggunakan *unity* sebagai *physic game engine* dan *visual studio* sebagai *code editor.*	Hanya menggunakan 1 metode saja dalam pembuatan *game* ini yaitu *behaviour tree* dalam pembuatan AI NPC *enemy*. *Game engine* yang akan digunakan dalam pembuatan *game* ini adalah *unreal engine 5.*