Implementasi Pohon berupa Pohon Tingkah Laku

Dimas Wahyu Langkawi 13518069
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13518069@std.stei.itb.ac.id

***Abstrak*—Pohon Tingkah Laku (*Behavior Tree)* adalah model matematika yang cukup sering digunakan pada video game sebagai NPC (*Non-Player Character*). Selain pada video game, pohon tingkah laku cukup populer di suatu kalangan tertentu dan selain di video game, implementasi pohon tingkah laku juga sering ditemukan pada robotik, dengan penerapan berupa aksi apa yang akan dilakukan selanjutnya oleh robot.**

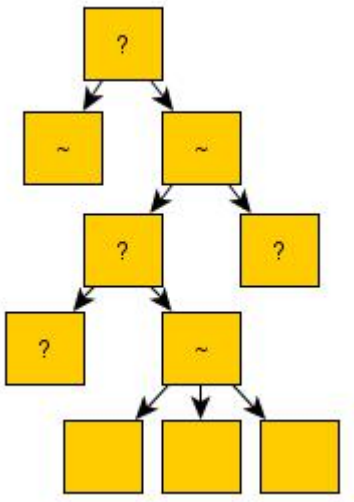
***Kata Kunci*—Pohon, *Behavior Tree, AI,* Video Games.**

I. PENDAHULUAN

Pada dunia video games yang interaktif sering terlihat karakter-karakter yang dapat ber-interaksi dengan seorang pemain, baik berupa pedagang yang dapat menjual banyak barang berguna bagi pemain hingga musuh yang dapat berkeliaran dan menyerang pemain jika pemain terlalu dekat.

Konsep pohon tingkah laku (*Behavior Tree*) pertama kali di publikasi pada tahun 2001 istilah awal yang digunakan contohnya "*genetic softwarre engineering*" atau "*genetic design*".

Pohon tingkah laku ini menjelaskan perpindahan antara suatu aksi yang terbatas secara modular. Salah satu kelebihan dari pohon tingkah laku ini yaitu kemampuannya untuk membuat suatu aksi yang kompleks berdasarkan aksi-aksi yang tidak terlalu kompleks tanpa menghiraukan bagaimana aksi-aksi yang tidak terlalu kompleks itu diimplementasikan.

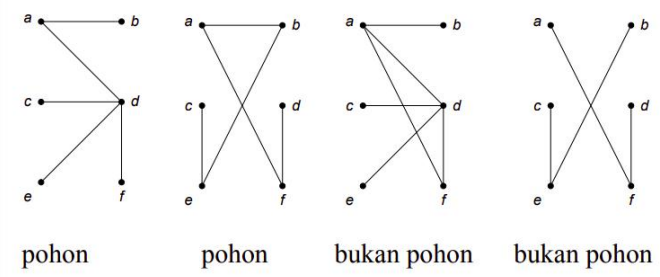


Gambar 1 Contoh abstrak dari pohon tingkah laku
Sumber :
https://www.gamasutra.com/blogs/ChrisSimpson/20140717/221339/Behavior_trees_for_AI_How_they_work.php

II. TEORI DASAR

A. Pohon

Pohon merupakan struktur data yang ada di ilmu informatika. Setiap pohon memiliki akar yang memiliki simpul-simpul anak yang terhubung dengan akar, pohon juga memiliki daun sebagai simpul yang tidak memiliki anak. Pohon adalah sebuah bentuk lain dari graf tak-berarah dan juga tidak mengandung sirkuit



Gambar 2 Definisi pohon dan bukan pohon
Sumber :
http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2013-2014/Pohon%20(2013).pdf

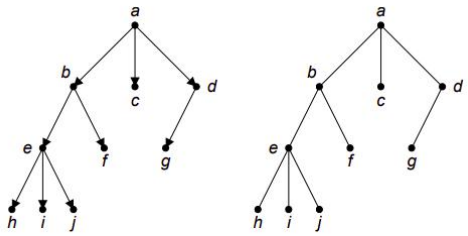
B. Sifat-sifat (properti) pohon

Ada pula cara mendefinisikan suatu pohon dan propertinya dengan cara lain. Misal G = (V,E) adalah graf tak-berarah sederhana dan jumlah simpulnya n. Maka, semua pernyataan di bawah ini adalah ekivalen:

1. G adalah pohon.
2. Setiap pasang simpul di dalam G terhubung dengan lintasan tunggal.
3. G terhubung dan memiliki m = n-1 buah sisi.
4. G tidak mengandung sirkuit dan memiliki m = n-1 buah susu.
5. G tidak mengandung sirkuit dan penambahan satu sisi pada graf akan membuat hanya satu sirkuit.
6. G terhubung dan semua sisinya adalah jembatan.

C. Pohon Berakar (rooted tree)

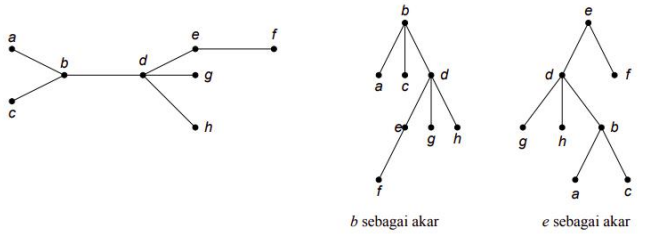
Pohon yang satu buah simpulnya diperlakukan sebagai akar dan sisi-sisinya diberi arah sehingga menjadi graf berarah dinamakan pohon berakar (rooted tree).



(a) Pohon berakar (b) sebagai perjanjian, tanda panah pada sisi dapat dibuang

Gambar 3 Pohon berakar
Sumber :
http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2013-2014/Pohon%20(2013).pdf

Suatu pohon berakar dapat dihasilkan dari pohon yang sama tapi dengan pemilihan akar yang berbeda.

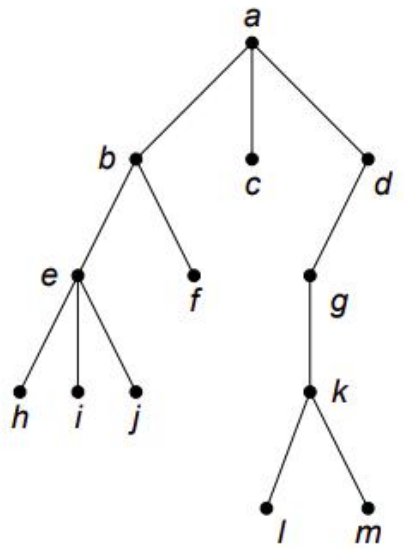


b sebagai akar *e* sebagai akar

Pohon dan dua buah pohon berakar yang dihasilkan dari pemilihan dua simpul berbeda sebagai akar

Gambar 4 Pohon yang sama tapi berbeda akar
Sumber :
http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2013-2014/Pohon%20(2013).pdf

Terminologi-terminologi pada pohon berakar.



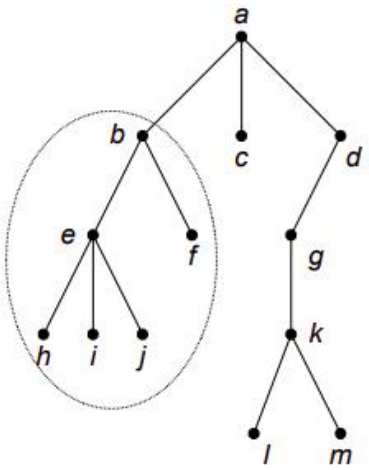
Gambar 5 Pohon berakar 2
Sumber :
http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2013-2014/Pohon%20(2013).pdf

1. Anak (*child* atau *children*) dan Orangtua (*parent*)
Pada gambar 5 simpul-simpul b, c, d adalah anak dari simpul a, dan a adalah orangtua dari simpul-simpul tersebut.

2. Lintasan (*path*)
Pada gambar 5 lintasan dari a ke j adalah a, b, e, j. Panjang lintasan dari a ke j adalah 3.

3. Saudara kandung (*sibling*)
Pada gambar 5 f adalah saudara kandung e, tetapi g bukan saudara kandung e, karena orangtua mereka berbeda.

4. Upapohon (*subtree*)
Upapohon adalah pohon baru yang terbentuk dari memilih salah satu simpul dan memilih anaknya sebagai akar baru dan menghapus sisi antar anak yang dipilih dan orangtuanya dan juga menghapus pohon yang terpisah.



Gambar 6 Contoh Upapohon dari pohon pada gambar 5
Sumber :
http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2013-2014/Pohon%20(2013).pdf
Pada gambar 6 Upapohon yang dilingkari terbentuk setelah memilih b sebagai akar baru.

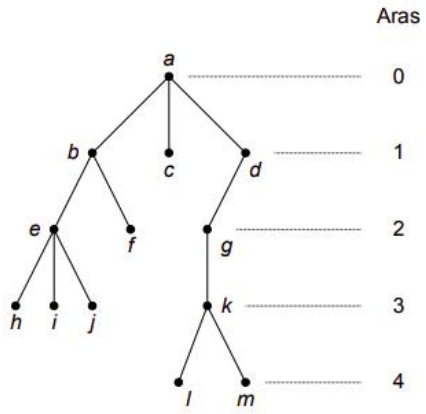
5. Derajat (*degree*)
Derajat sebuah simpul adalah jumlah upapohon (atau jumlah anak) pada simpul tersebut. Pada gambar 5 derajat a adalah 3, derajat c adalah 0. Derajat maksimum dari suatu pohon adalah derajat pohon itu sendiri, dalam kasus ini derajat pohon gambar 5 yaitu 3.

6. Daun (*leaf*)
Simpul yang berderajat nol (atau tidak mempunyai anak) disebut daun. Simpul h, i, j, f, c, l, dan m adalah daun karena simpul-simpul ini berderajat 0 (tidak memiliki anak).

7. Simpul dalam (*internal nodes*)
Simpul yang mempunyai anak disebut simpul dalam. Simpul b, d, e, g, dan k adalah simpul dalam.

8. Aras (*level*) atau Tingkat

Aras atau Tingkat adalah level dari sebuah simpul dihitung dari akar.



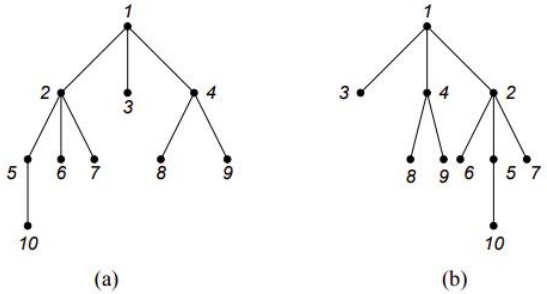
Gambar 7 Tingkat pohon
Sumber :
http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2013-2014/Pohon%20(2013).pdf

9. Tinggi (*height*) atau Kedalaman (*depth*)

Aras maksimum dari suatu pohon disebut tinggi atau kedalaman pohon tersebut. Pohon pada gambar 7 mempunyai tinggi 4.

D. Pohon terurut (*ordered tree*)

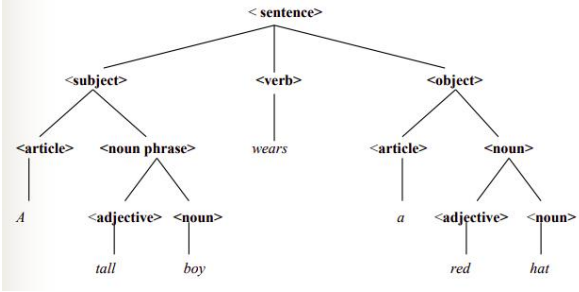
Pohon berakar yang urutan anak-anaknya penting disebut pohon terurut (*ordered tree*).



Gambar 8.a Pohon terurut 8.b Pohon terurut yang berbeda dari 8.a
Sumber :
http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2013-2014/Pohon%20(2013).pdf

E. Pohon n-ary

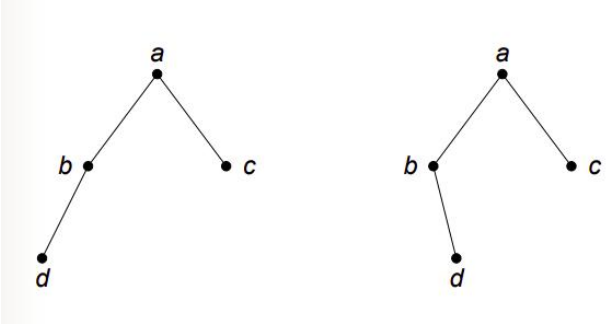
Pohon berakar yang tiap simpul cabangnya mempunyai paling banyak n buah anak disebut pohon n-ary



Gambar 9 Pohon parsing dari kalimat "*A tall boy wears a red hat*"
Sumber :
http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2013-2014/Pohon%20(2013).pdf

F. Pohon Biner (*binary tree*)

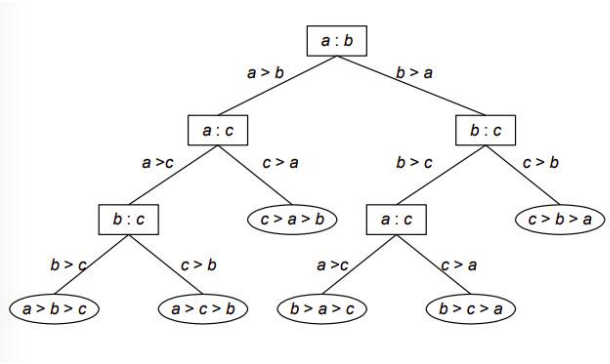
Pohon biner adalah pohon n-ary dengan n = 2, simpul anak pada pohon biner dibedakan menjadi dua yaitu anak kiri (*left child*) dan anak kanan (*right child*). Karena ada perbedaan urutan anak maka pohon biner adalah pohon terurut



Gambar 9 Pohon biner yang berbeda
Sumber :
http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2013-2014/Pohon%20(2013).pdf

G. Pohon keputusan (*Decision tree*)

Pohon keputusan adalah salah satu bentuk aplikasi dari pohon biner. Tiap simpul internal pada pohon keputusan adalah suatu ekspresi yang perlu dipenuhi untuk mengetahui ke arah mana pohon akan ditelusuri



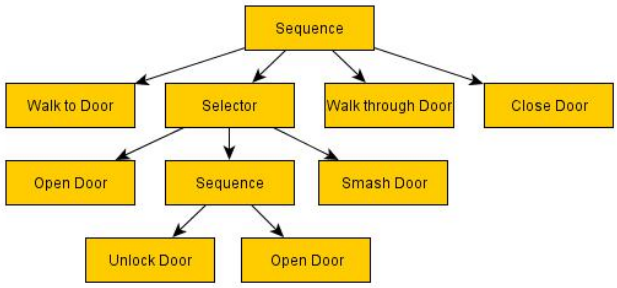
Gambar 10 Pohon keputusan untuk mencari angka terbesar diantara a, b dan c.
Sumber :
http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2013-2014/Pohon%20(2013).pdf

H. Pohon tingkah laku (*behavior tree*)

Pohon tingkah laku / pohon perilaku (*Behavior Tree*) merupakan salah satu penerapan pohon berakar. Pohon perilaku adalah pohon dengan simpul hirarkis yang mengatur aliran keputusan yang diambil oleh entitas AI (*Artificial Intelligence*). Pada pohonnya sendiri daun lah yang merupakan aksi yang perlu dilakukan. Simpul yang membentuk cabang

adalah berbagai jenis simpul utilitas yang mengatur jalannya alur AI menuruni pohon tersebut untuk mendapatkan aksi yang cocok berdasarkan situasi yang ada (Contoh pada gambar 1).

Pohon perilaku bisa menjadi sangat dalam dan kompleks, dengan simpul memanggil upapohon yang melakukan suatu fungsi tertentu. Design pohon perilaku yang mudah dimengerti oleh manusia dan kemungkinan yang bisa dilakukan oleh pembuat membuatnya sangat populer, pohon perilaku bisa menjadi sangat kompleks pada game yang menyerahkan *gameplay* pada AI karena menarik atau tidaknya suatu game bisa ditentukan dari situ. Selain pada game pohon perilaku juga seringkali digunakan pada bidang robotik, dikarenakan pengembangan yang mudah dilakukan.



Gambar 11 Contoh behavior tree
Sumber :
https://www.gamasutra.com/blogs/ChrisSimpson/20140717/221339/Behavior_trees_for_AI_How_they_work.php

Pada Gambar 11 ditampilkan AI sederhana yang dapat membuka pintu bahkan jika terkunci. Perhatikan bahwa pohon perilaku tidak memperhatikan bagaimana aksi-aksi pada daunnya diimplementasikan, pohon perilaku hanya peduli terhadap alur cara pemilihan aksi saja.

Pada Gambar 11 terlihat bahwa ada beberapa simpul bercabang yang memiliki nama, mereka adalah simpul utilitas. Simpul-simpul utilitas inilah yang akan mengatur bagaimana program menelusuri pohon dan memilih aksi-aksi yang ada pada daunnya. Sering kali upapohon di pohon perilaku memiliki fungisnya sendiri sebagai contoh, upapohon yang dihasilkan setelah membuat simpul "*selector*" pada Gambar 11 menyajikan fungsi berupa membuka pintu baik yang terkunci ataupun tidak.

Fungsionalitas inti umum untuk semua simpul pada pohon perilaku yaitu mereka dapat mengembalikan satu dari tiga status (bergantung pada cara pohon perilaku ini diimplementasikan angka yang ditunjukkan disini sangat mungkin untuk berubah-ubah) yaitu *Success, Failure, Running*. Sesuai nama mereka 2 status pertama akan mengembalikan nilai tersebut jika aksi yang dilakukan berhasil atau gagal, sedangkan status ke tiga (*Running*) akan dikembalikan jika simpul yang ada masih berjalan dan memiliki potensi menjadi *Success* atau *Failure*, simpul akan dijalankan hingga akan dicek lagi nanti.

Fungsionalitas ini adalah salah satu komponen utama pada pohon perilaku, karena dengan ini program mampu membedakan mana yang berhasil dan gagal, dan bahkan mana yang sedang berjalan dan akan dicek kemudian. Contohnya jika pada Gambar 11 daun "*Unlock Door*" sedang berjalan maka upapohon yang sedang membuka pintu sedang berjalan dan program harus menunggu.

Dengan adanya status maka kita dapat pula mengimplementasikan berbagai macam simpul, yang cukup sering digunakan yaitu *Sequence, Selector, Leaf* (ketiganya ada pada Gambar 11). *Leaf* sesuai yang sudah dijelaskan sebelumnya adalah simpul yang mengeksekusi suatu aksi tertentu, sedangkan *Sequence* dan *Selector* adalah suatu *control node* atau simpul kontrol.

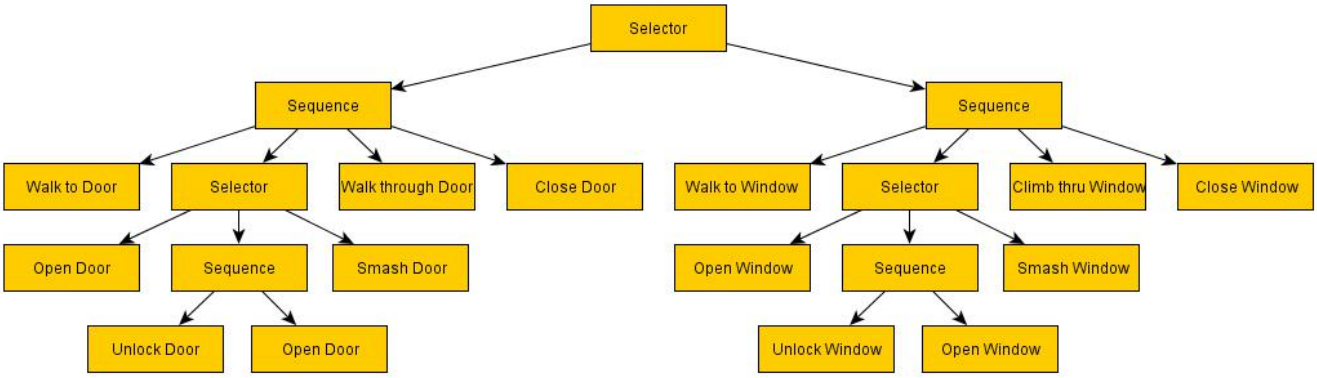
Simpul *Sequence* biasanya memiliki 2 atau lebih anak dan simpul ini memproses anak-anaknya satu persatu dari kiri dan akan menunggu hingga simpul yang dijalankan sudah mengembalikan nilai *Success*. Jika ada 1 saja simpul anak *Sequence* yang mengembalikan gagal maka Sequence juga akan mengembalikan nilai gagal.

Simpul *Selector* bisa dibilang cara bekerjanya berbanding terbalik dengan *Sequence,* jika *Sequence* mengembalikan berhasil jika semua simpulnya mengembalikan berhasil maka *Selector* hanya butuh satu simpul anak yang berhasil untuk mengembalikan nilai *Success* ke orangtua-nya. *Selector* ibaratkan *if else statement* dalam pemrograman, jika ada satu yang berhasil maka sisanya tidak akan perlu dicek lagi. Pada contoh Gambar 11 *Selector* digunakan untuk membuka pintu, perhatikan bahwa ada 3 kemungkinan yaitu : pintu tidak terkunci, pintu terkunci dan memegang kunci, pintu terkunci namun tidak memegang kunci. Jika ada satu dari tiga simpul ini yang menghasilkan berhasil maka *Selector* akan mengembalikan nilai *Success* pada orangtua-nya yaitu dalm kasus ini simpul akar *Sequence*.

III. Implementasi Pohon Tingkah Laku

Salah satu topik yang sering dibahas pada komunitas pengembang game adalah AI, dan kita dapat mengimplementasikan pohon tingkah laku ini untuk menjadi AI di dalam suatu game. Mari ambil contoh yaitu lanjutan dari Gambar 11. Pada pohon ini program akan berusaha untuk memasuki suatu ruangan atau bangunan.

Pertama sebuah *Sequence* akan dijalankan dengan 4 anak yang 3 diantaranya merupakan daun, jika simpul daun"*Walk to Door*" berhasil dijalankan maka program akan melanjutkan ke simpul *Selector.* Pada simpul ini dapat dilihat bahwa program



Gambar 12 Contoh behavior tree lanjutan
Sumber : https://www.gamasutra.com/blogs/ChrisSimpson/20140717/221339/Behavior_trees_for_AI_How_they_work.php

akan membuka pintu dan memiliki 3 cara yang akan dicoba secara berurut mulai dari kiri karena makin di kiri simpul maka makin diiginkan untuk terjadi terlebih dahulu (prioritas), jika simpul daun "*Open Door*" berhasil maka tidak perlu mengecek simpul lainnya di *Selector* tersebut dan kembali ke akar dan melakukan 2 aksi simpul daun sisanya hingga program masuk ke dalam suatu bangunan tersebut.

Namun jika "*Open Door*" gagal (misal pintu dikunci) maka program akan mencoba simpul berikutnya yaitu yang ada dikanannya berdasarkan prioitas yakni simpul *Sequence* baru lagi. Di simpul ini dapat dilihat bahwa program akan mencoba untuk membuka kunci lalu membuka pintu (mungkin program memegang kuncinya, atau punya skill yang dapat membuka pintu terkunci tanpa kunci pintu itu sendiri dll), jika simpul daun "*Unlock Door*" gagal (misal program tidak memiliki kunci, mungkin program tidak memiliki skill untuk membuka kunci tanpa kunci yang dibutuhkan) maka program akan mengevaluasi simpul daun terakhir pada simpul *Selector* ini.

Simpul terkahir pada *Selector* berupa sebuah simpul daun berupa aksi "*Smash Door*" dimana itu akan menjadi satu-satunya pilihan bagi program jika ingin memasuki ruangan/bangunan tersebut.

Jika Salah satu dari *Selector* itu berhasil maka akan dapat kembali ke akar dimana simpul *Sequence* akan melanjutkan menjalankan simpul lain setelah *Selector* yaitu dua buah simpul daun lain berupa aksi yang akan menghasilkan program sudah berada di dalam ruangan atau bangunan yang diinginkan pemain.

Jika dari ketiga anak *Selector* semuanya mengembalikan nilai gagal maka *Selector* akan mengembalikan gagal ke akar dan simpul akar tidak akan melanjutkan ke simpul daun

lainnnya sedangkan akan mengembalikan nilai gagal. Dan karena simpul akar mengembalikan nilai gagal maka kita dapat simpulkan bahwa pohon tingkah laku ini gagal dilakukan.

Dengan contoh itu dapat terlihat jelas *readablity* dari pohon tingkah laku dan membuat pohon tingkah laku ini menjadi pilihan yang sering disebut ketika sedang memilih cara mengimpelemntasikan AI pada program.

Untuk memahami lebih lanjut pohon tingkah laku maka mari ambil contoh berupa pohon tingkah laku yang lebih kompleks. Untuk kemudahan mari ambil tujuan yang sama yaitu memasuki suatu ruangan atau bangunan namun dengan suatu tambahan berupa selain dari pintu program juga memiliki pilihan untuk memasuki suatu ruangan atau ruangan tersebut melalui jendela.

Perhatikan Gambar 12, mungkin gambar tersebut membuat pohon tingkah laku terlihat kompleks namun kenyataannya jika diperhatikan baik baik maka dapat dilihat bahwa pohon tingkah laku ini berupa Gambar 11 yang ditambah suatu orangtua berupa *Selector* yang memiliki anak berupa upapohon yang memiliki fungsi yaitu memasuki ruangan menggunakan cara lain selain pintu yaitu melalui jendela.

Perhatikan bahwa jika program dapat memasuki ruangan melalui pintu maka program tidak akan perlu mencoba masuk lewat jendela karena akar yang berupa *Selector* jadi simpul anak berikutnya tidak akan dicek jika salah satu anaknya ada yang berhasil maka dari itu akar yang tepat adalah akar berupa simpul *Selector*.

Pada upapohon kanan cara kerjanya tidak berbeda begitu jauh dengan upapohon kiri, yang membedakan hanya jalur masuk yang dilewati. Pada upapohon kanan *Sequence* memiliki 4 anak juga yang 3 diantaranya adalah simpul anak, dan 1 simpul *Selector*.

Di simpul *Selector* ini program akan mencoba membuka kunci jendala dibanding upapohon kiri yang mencoba membuka pintu. Skenario terbaik adalah jendela tidak terkunci dan program bisa langsung masuk. Jika terkunci maka program akan terlebih dahulu membuka kunci jendele, jika gagal maka program akan menghancurkan jendela.

Walaupun Gambar 12 terlihat jauh lebih kompleks dari Gambar 11 pada kenyataannya kedua pohon tingkah laku tidak berbeda jauh, pohon tingkah laku pada Gambar 12 hanyalah perluasan dari pohon tingkah laku pada Gambar 11.

Perhatikan bahwa penambahan fitur baru dan perluasan tingkah laku yang dapat dilakukan oleh program sangat mudah dilakukan tanpa perlu mengubah-ubah pohon yang sudah ada, yang juga berarti lebih sedikit masalah dalam proses implementasi pohon ini.

IV. KESIMPULAN

Pohon tingkah laku ini jelas masih banyak jenis implementasinya bahkan dalam bidang robotik, supaya robot dapat menentukan aksi maan yang akan dilakukan selanjutnya berdasarkan situasi dunia luar yang dapat diobservasi dan input-input lainnya. Contoh yang penulis bahas pada makalah ini hanya sebagian kecil dari pemanfaatan pohon tingkah laku, masih banyak jenis implementasi di luar yang penulis bahas pada makalah ini.

Di dunia *Video Games* pun banyak penerapan dari pohon tingkah laku ini selain dari yang dibahas pada makalah ini.

Mulai dari cara pedagang dalam game menjual barang pada pemain. Bahkan hingga bos yang harus dikalahkan pemain memiliki sifat yang berbeda bergantung pada konsdisi saat ini bos itu sendiri.

V. Ucapan Terima Kasih

Ucapan terima kasih sebesar besarnya penulis berikan kepada Allah SWT yang telah memberikan nikmat sehat baik jasmani maupun rohani serta rezeki untuk menyelesaikan makalah ini. Dan tentu tak lupa pula ucapan terima kasih penulis ucapkan untuk semua pihak yang telah membantu dan mendukung dibuatnya makalah ini. Mudah-mudahan makalah ini dapat menambah ilmu bagi pembaca.

References

[1] *https://www.gamasutra.com/blogs/ChrisSimpson/20140717/221339/Behavior_trees_for_AI_How_they_work.php*. Diakses pada 4 Desember 2019

[2] Isla D. 2005. Handling complexity in the Halo 2 AI. In Game Developers Conference (Vol. 12). Diakses pada 6 Desember 2019

[3] *http://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2013-2014/Pohon%20(2013).pdf. Diakses pada 5 Desember 2019*

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 6 Desember 2017

Ttd (scan atau foto ttd)

Dimas Wahyu Langkawi
13518069