

Aplikasi Behavior Tree untuk kepintaran buatan karakter monster pada permainan berbasis kisi

Candra Ramsi 13514090¹

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia

¹candra_ramsi@s.itb.ac.id

Abstract—Kepintaran buatan untuk permainan komputer berbasis kisi memiliki banyak teknik diantaranya finite state machine, adhoc, decision tree, dan behavior tree. Teknik-teknik ini memiliki masalahnya masing-masing. Masalah-masalah ini bersifat trivial dengan solusinya masing-masing. Decision Tree mencoba untuk menghilangkan beberapa masalah yang terdapat pada teknik-teknik lainnya. Penggunaan Decision Tree dapat diimplementasikan dengan mudah dan menghasilkan sebuah puwarupa. Puwarupa ini memberikan gambaran tentang bagaimana Decision Tree menyelesaikan sebuah masalah sederhana.

Keywords—Pohon, Behavior Tree, Kisi, Kepintaran buatan

1. INTRODUKSI

Permainan pada komputer pada umumnya membutuhkan kepintaran buatan. Kepintaran buatan pada permainan komputer sangat dibatasi oleh waktu komputasi yang minimum. Komputer harus dapat membuat keputusan dalam hitungan milidetik jika tidak permainan akan terasa lambat.

Tingkat pertumbuhan komputasi sudah sangatlah tinggi tetapi masih ada *platform* baru yang dibatasi oleh kemampuan komputasinya. Salah satu *platform* ini adalah *platform* mobile. Kapabilitas komputasi *smartphone* masih sangat terbatas dibandingkan dengan *platform* lainnya. Namun, keterbatasan ini akan dapat diatas dalam waktu yang relatif singkat. [2]

Pada platform mobile seperti *smartphone* masalah yang lebih besar adalah penggunaan baterai. Walaupun *smartphone* dapat melakukan komputasi yang sangat besar, *smartphone* masih dibatasi dengan baterai yang relatif minim.

Pada Umumnya permainan berbasis kisi (*grid*) mempunyai kepintaran buatan yang bersifat ad-hoc, hanya berlaku pada kasus yang sangat khusus tersebut. Kepintaran buatan yang bersifat ad-hoc sangat sulit untuk diaplikasikan untuk sifat-sifat yang mirip namun sedikit berbeda.

Selain cara ad-hoc terdapat juga cara yang lebih baik menggunakan *Finite State Machine*. Finite State Machine adalah kepintaran buatan yang memiliki sebuah *state* dimana sebuah aksi dilakukan terus menerus hingga

kondisi untuk berpindah *state* terpenuhi. F.S.M. memiliki masalah dengan penangan kesalahan. F.S.M. juga memiliki masalah jumlah *state* yang dapat bertumbuh sangat banyak jumlah *state*-nya.

Teknik lainnya yang dapat digunakan adalah *Decision Tree*. Decision Tree mengevaluasi sebuah keadaan dan mencari solusi sesuai keputusan yang dibuat pada tiap nodal. Decision Tree hanya memiliki suatu masalah yaitu sulitnya mengkode *Decision Tree* untuk sebuah keputusan yang bersifat prioritas atau serial. Contoh dari keputusan tersebut adalah saat kepintaran buatan ingin masuk kedalam rumah ada beberapa langkah-langkah yang harus diikuti seperti jalan kedepan pintu, membuka pintu, berjalan melewati pintu, dan menutup pintu. Perilaku seperti ini cukup sulit untuk diimplementasikan dengan benar pada Decision Tree.

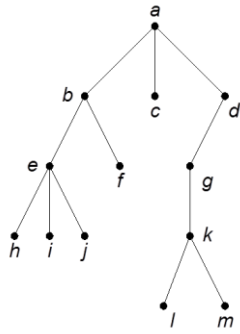
Ketiga teknik diatas, *Finite State Machine*, Ad hoc, dan *Decision Tree* mempunyai kelemahannya masing-masing. Kelemahan-kelemahan tersebut dapat saja ditanggulangi dengan berbagai implementasi yang bersifat ad-hoc. *Behavior Tree* yang akan digunakan pada aplikasi program makalah ini mencoba untuk membuat implementasi-implementasi yang bersifat ad-hoc tersebut menjadi suatu yang lebih terasa alami.

2. TEORI POHON

Pohon adalah graf tak berarah terhubung yang tidak mengandung sirkuit. [7] Pohon merupakan suatu graf yang memiliki properti-properti khusus. Berikut properti khusus tersebut :

1. Setiap simpul terhubung dengan lintasan tunggal
2. Pohon memiliki tepat n buah simpul dan n-1 buah sisi
3. Pohon tidak mengandung sirkuit
4. Penambahan satu buah sisi akan menghasilkan hanya satu buah sirkuit
5. Pohon terhubung dan semua sisinya adalah jembatan.

Pohon Berakar (*rooted tree*) adalah pohon dengan simpul yang satu sisinya diperlakukan sebagai akar dan sisi-sisinya diberikan arah. Berikut adalah contoh pohon berakar. [7]



Gambar 1. Pohon berakar

(sumber : Presentasi Pohon Bahan Kuliah IF2120 Matematika Diskrit, Rinaldi Munir).

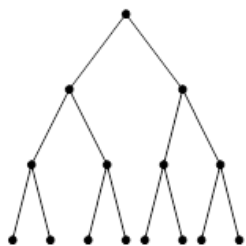
Pada gambar diatas arah tidak digambarkan hanya merupakan persetujuan agar gambar lebih sederhana.

Beberapa terminologi yang akan digunakan adalah

- Anak dan orangtua
Anak adalah simpul-simpul yang berada dibawah suatu simpul.
Contoh : simpul h,i, dan j adalah anak dari simpul e
Orang tua adalah simpul yang berada diatas suatu simpul
- Aras (*level*) atau tingkat
Tingkat dihitung dari akar teratas. Tingkat akar teratas adalah tingkat 0 dan anaknya memiliki tingkat 1 dst.
Contoh : Aras a adalah 0; Aras g adalah 2.
- Daun
Daun adalah simpul yang tidak memiliki anak
Contoh : m adalah daun.
- Saudara kandung
Saudara kandung adalah simpul yang memiliki orangtua yang sama
Contoh : e dan f adalah saudara kandung.
- Akar teratas
Akar teratas adalah nodal denga aras 0
Contoh : a adalah aras teratas pohon berakar tersebut.

Pohon n-ary

Pohon n-ary adalah pohon yang setiap simpulnya mempunyai anak paling banyak sebanyak n-buah. Misalnya pohon yang memiliki anak maksimum dua tiap simpulnya disebut pohon binary. Pohon yang memiliki anak maksimum tiga tiap simpulnya disebut photon trinary, dst. [7]



Gambar 2. Pohon Biner

Decision Tree

Decision Tree adalah pohon yang memiliki nodal-nodal dimana tiap nodal nya dibuat sebuah keputusan hingga mencapai daun dan konklusi didapatkan. *Decision Tree* banyak digunakan untuk aplikasi *data mining*, *machine learning*, dan kepintaran buatan. *Decision Tree* memiliki kelebihan yaitu parameter tiap nodalnya dapat diubah-ubah untuk proses *learning*. [3][4]

Behavior Tree

Behavior Tree adalah pohon yang memiliki nodal-nodal yang memiliki aturan-aturan tersendiri. Ada dua aturan utama yaitu *priority* dan *sequence*. Tiap nodal dapat memberikan tiga jenis jawaban pada umumnya yaitu *running*, *fail*, dan *success*. Aturan *priority* adalah aturan dimana nodal akan mengirimkan *success* jika 1 saja nodal menghasilkan *success* dan akan fail jika semua nodal anak nya menghasilkan fail. Aturan *sequence* memberikan *success* jika semua nodalnya *success*. Ada juga aturan khusus yang tugasnya untuk mengkonversi biasanya membalikan *success* menjadi fail dan sebaliknya.

Proses pengerjaan behavior tree adalah evaluasi pohon dari nodal teratas. Nodal teratas akan melakukan prosesnya. Program akan memproses hingga mencapai nodal daun. Setelah nodal daun tercapai maka sebuah aksi akan dilakukan. Aksi tersebut akan mengembalikan salah satu dari tiga jenis keluaran. Keluaran tersebut akan diproses oleh nodal orang tua dari nodal daun tersebut. Hingga kembali lagi pada akar teratas. Pada umumnya, Akar teratas tidak mempedulikan keluaran dari anak-anak dibawahnya. Akar teratas harus mengulang kembali proses evaluasi dari awal pada setiap iterasi dan tidak mempedulikan keadaan terakhir dari iterasi sebelumnya.

Behavior Tree banyak digunakan dalam industri permainan komputer karena kemudahan dalam penanganan masalah. Selain penanganan masalah, *Behavior Tree* juga banyak digunakan karena aturan *sequence* dan prioritas terasa alami dalam mendesain sebuah kepintaran buatan permainan komputer. Sebaliknya Sifat-sifat *sequence* dan prioritas lebih sulit diimplementasikan dalam *Decision Tree*. Tetapi *Behavior Tree* jarang digunakan pada riset kepintaran buatan karena properti *Behavior Tree* sulit untuk dilatih. [1][3][5]

3. APA YANG SALAH DENGAN FINITE STATE MACHINE ?

Finite state machine sangat baik dalam penanganan kepintaran buatan yang sederhana. Namun, dengan jumlah state yang sangat banyak, Finite State Machine menjadi sulit untuk dipelihara. F.S.M. memiliki implementasi yang menyebabkan banyaknya duplikasi kode jika 2 state berbeda tetapi melakukan hal yang mirip. Masalah-masalah F.S.M. dapat diselesaikan dengan design pattern. Tetapi masih sangat mudah untuk melakukan hal-hal yang menyebabkan sulitnya pemeliharaan F.S.M. [8]

4. APA YANG SALAH DENGAN DECISION TREE ?

Decision Tree tidak umum digunakan pada kepintaran buatan permainan komputer, karena sulitnya mengimplementasikan proses sequence dan priority. Sequence adalah proses yang berurut namun mungkin tidak dapat dilaksanakan dalam satu iterasi. Proses prioritas dimana pemrosesan hal tertentu diprioritaskan dan hanya berpindah ke proses berikutnya jika proses pertama gagal. Proses tersebut sulit untuk diimplementasikan menggunakan decision tree. [3]

5. ATURAN PERMAINAN

Permainan kali bukan merupakan permainan sebenarnya dan hanya merupakan sebuah puwarupa untuk menunjukkan konsep penggunaan Behavior Tree untuk kepintaran buatan monster pada permainan berbasis kisi.

Pada permainan berbasis kisi terdapat sebuah peta dua dimensi dan terdapat kisi-kisi yang membagi-bagi peta tersebut. Pembagian tersebut menghasilkan jumlah sel-sel yang dapat ditempati oleh objek-objek pada permainan tersebut.

Pada permainan ini terdapat 4 buah objek pada peta yaitu batu(rock), rumput(grass), kelinci(rabbit), dan rubah(fox). Tiap sel pada peta hanya dapat ditempatkan oleh tepat satu objek. Tiap objek hanya dapat berpindah sel sebelahnya dan tidak dapat berjalan menyerong.

Tujuan dari permainan ini adalah agar kelinci dapat memakan rumput sebanyak-banyaknya sebelum tertangkap oleh rubah dan dimakan.

Permainan kali ini tidak terdapat pemain yang bersifat aktif. Pada permainan ini pemain hanya menentukan sebuah input dimasukan untuk menentukan posisi awal 2 jenis ras yaitu kelinci dan rubah. Lalu permainan akan disimulasikan dan akan terdapat pesan-pesan lokasi kelinci dimakan rubah atau memakan rumput.

Kondisi berakhirnya permainan adalah saat semua rumput termakan atau semua kelinci termakan. Sebelum kondisi ini tercapai permainan akan berlangsung terus menerus.

6. Implementasi *Behavior Tree* pada permainan berbasis kisi

Program diimplementasikan menggunakan Bahasa pemrograman JavaScript. *JavaScript* digunakan karena fleksibilitas tipe data yang dapat digunakan. Kefleksibilitas *JavaScript* membuat *JavaScript* menjadi Bahasa yang ideal untuk membuat sebuah puwarupa. Selain itu, *JavaScript* adalah Bahasa yang familiar bagi saya. *JavaScript* kali ini menggunakan perangkat lunak *Nodejs* untuk menjalankannya.

Kepintaran buatan yang akan diimplementasikan adalah kepintaran buatan untuk dua buah monster saja. Dua monster ini direpresentasikan dalam bentuk dua binatang yaitu rubah dan kelinci. Rubah dan kelinci dapat meninggal karena kekurangan makanan.

Kepintaran buatan kelinci akan mencoba mencari

rumput dan selamat selama-lamanya. Rubah akan mencari kelinci mengejanya dan mencoba untuk selamat dari serangan rubah.

Peta berkisi pada program diimplementasikan menggunakan array satu dimensi yang diabstraksi. Array yang diabstraksi dibuat seolah-oleh memiliki aksis dan ordinat. Proses abstraksi ini menggunakan rumus

$$M_{x,y} = A_{y*\text{width}+x}$$

Dimana x dan y adalah koordinat pada peta kisi, dan width adalah lebar peta.

Proses pencarian rute pada kelinci dan rubah menggunakan algoritma path finding Breath First Search. B.F.S. dipilih dengan tujuan agar proses sederhana. Proses pencarian jalur menggunakan B.F.S. dilakukan dengan cara menyebar ketitik-titik hingga persebaran sampai kondisi yang dituju yaitu tepat disebelah makanan berada.

Implementasi B.F.S. adalah seperti berikut

misal ada sebuah antrian bernama Q
masukan posisi awal pada q nyatakan q sudah dicapai

selama Q belum habis

cek apakah kepala Q adalah kondisi berhenti

jika ya maka keluarakan jawaban

jika tidak

masukan seluruh tetangga kepala Q ke Q

hapus kepala Q

Implementasi algoritma diatas merupakan gambaran umum dari bagaimana B.F.S. bekerja. [6]

Proses memasukan tetangga kepala Q pada program puwarupa mengikuti aturan permainan. Pada proses tersebut dilakukan pemeriksaan ke atas, bawah, kiri, dan kanan dari player. Dari setiap lokasi tersebut harus diperiksa apakah lokasi tersebut sudah ada di Q, apakah lokasi tersebut terdapat objek yang menempatnya, dan apakah lokasi tersebut diluar peta. Lokasi-lokasi yang lulus pemeriksaan baru dimasukan kepada Q.

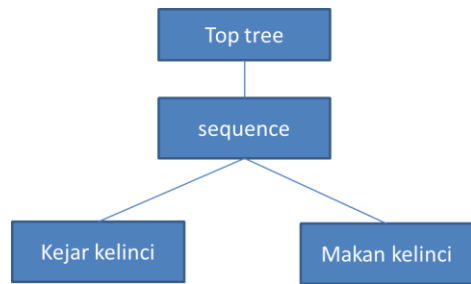
Kondisi berhenti dari program puwarupa ini adalah saat kepala Q berada di sebelah dari target yang ingin dicapai. Bersebelahan pada kasus ini adalah saat jumlah perbedaan ordinat dan absis kedua objek tidak lebih dari satu. Target yang ingin dicapai diperiksa melalui tipe objek tersebut.

Jawaban yang diinginkan dari proses pathfinding pada program puwarupa ini adalah jalan terdekat untuk mencapai objek yang ingin dicapai. Proses yang dilakukan untuk mendapatkan jawaban tersebut adalah *backtracking*. Pada proses *backtracking* nodal kepala dicari sumber datangnya. Proses ini dilakukan secara terus menerus hingga mencapai titik awal. Akumulasi dari semua nodal yang ditemukan akan memberikan sebuah jawaban mengenai jalur yang harus dicapai untuk mencapai objek tujuan.

Behavior Tree harus melakukan sebuah proses tiap iterasinya yaitu menghapus kondisi pada iterasi sebelumnya. Pada implementasi puwarupa ini digunakan sebuah fungsi bernama *top tree* yang dapat melakukan

aksi penghapusan tersebut. Fungsi top tree ini ditambahkan pada setiap akar teratas dari *Behavior Tree*. Tetapi top tree penggunaannya tidak terbatas pada akar teratas dari sebuah *Behavior Tree*.

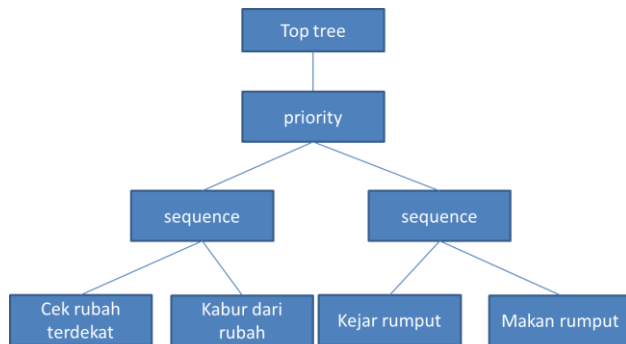
Logika Behavior Tree yang digunakan pada binatang rubah adalah seperti berikut



Gambar 3. Logika Behavior Tree pada rubah

Proses logika rubah adalah mengejar kelinci hingga suatu saat kelinci tercapai dan nodal kejar kelinci mengembalikan success. Lalu akan sequence dilanjutkan ke proses makan kelinci. Pada tiap iterasi top tree akan melakukan reset pada sequence. Hal ini perlu dilakukan karena proses iterasi harus dimulai kembali pada akar paling atas.

Logika Behavior Tree yang digunakan pada binatang kelinci adalah seperti berikut



Gambar 4. Logika Behavior Tree pada kelinci

Berbeda dengan rubah kelinci harus memperhatikan apakah dirinya berada didekat rubah atau tidak. Jika ia berada didekat rubah maka ia harus kabur secepat-cepatnya. Jika tidak ada rubah disekitarnya maka ia akan mencoba untuk melakukan sequence lainnya yaitu mencari makanan. Jika makanan sudah tercapai maka kelinci akan melanjutkan ke proses makan rumput.

7. HASIL KELUARAN PUWARUPA

Puwarupa dapat dilihat pada halaman web <https://github.com/rmxhaha/BehaviorTree>.

Program dimulai dengan peta sepuluh baris dan sepuluh kolom didalamnya terdapat tiga kelinci, tiga rumput, dan satu rubah.

Berikut tampilan program pada awalnya

```

=====
F
R GR
R
GG
=====
grass has been eaten
=====
  
```

```

F R
R
R
GG
=====
  
```

Lalu kelinci merasa aman dan mencoba untuk mencari rumput terdekat. Salah satu kelinci lalu memakan sebuah rumput.

```

rabbit has been eaten
=====
  
```

```

R
F
R GG
=====
  
```

Rubah mengejar salah satu kelinci dan berhasil memakannya. Kelinci lainnya juga berada pada jangkauan rubah. Maka kelinci-kelinci lainnya juga melakukan proses pelarian dari rubah tersebut.

```

=====
F R
RGG
=====
rabbit has been eaten
=====
  
```

```

F
R GG
=====
  
```

```

F
R GG
=====
rabbit has been eaten
  
```

=====

F
GG
=====

Proses kabur dan mencari makanan terus dilakukan oleh kelinci hingga kondisi selesai terpenuhi. Pada simulasi kali ini rubah menang karena rubah berhasil memakan seluruh kelinci yang berada pada peta.

Hasil puwarupa ini merupakan sebuah bukti konsep bahwa *Behavior Tree* dapat digunakan pada aplikasi pembuatan kepintaran buatan untuk permainan komputer berbasis kisi. *Behavior Tree* pada puwarupa ini sangat sederhana namun dapat memberikan gambaran mengenai penggunaan *Behavior Tree*.

8. KONKLUSI

Aplikasi Behavior Tree dalam permainan komputer berbasis kisi mudah untuk dilakukan. Proses pembentukan puwarupa ini memberikan sebuah gambaran terhadap cara pembuatan kepintaran buatan pada permainan komputer berbasis kisi.

REFERENSI

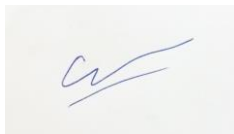
- [1] <http://www.gamasutra.com/blogs/ChrisSimpson/20140717/221339/>
- [2] <http://www.digitaltrends.com/computing/why-your-smartphone-wont-be-your-next-pc/>
- [3] <https://prezi.com/ez0617qeja6s/copy-of-ai-decision-tree-behaviour-tree/>
- [4] <http://whatis.techtarget.com/definition/decision-tree>
- [5] <http://gamedev.stackexchange.com/questions/51738/behavior-trees-actions-that-take-longer-than-one-tick>
- [6] <http://www.cs.nott.ac.uk/~psznza/G5BADS04/graphs2.pdf>
- [7] Presentasi Pohon Bahan kuliah IF2120 Matematika Diskrit – Rinaldi Munir
- [8] Game Programming algorithms and techniques : a platform-agnostic approach – Madhav, Sanjay

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, 10 Desember 2015

ttd



Candra Ramsi dan 13514090