Aplikasi Graf dalam Perancangan *Game* Digital dengan *Multiple Ending*

Cendhika Imantoro - 13514037
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
cendhikaimantoro@s.itb.ac.id

***Abstract*—Salah satu aspek yang menentukan kualitas dari suatu *game* digital adalah menarik tidaknya *game* untuk dimainkan berulang kali. Memberikan *multiple ending* pada *game* dapat menaikkan nilai *game* dalam aspek ini. Metode ini tentu akan memunculkan banyak percabangan kondisi pada *game* yang tiap cabangnya menuntun *player* ke *ending* yang berbeda. Makalah ini menjelaskan sebuah alternatif cara untuk mendesain percabangan yang ada dalam *game*, yaitu menggunakan graf.**

***Keywords—Action*, *ending*, graf, *state*.**

I. PENDAHULUAN

Kelebihan bidang informatika dibanding bidang lain adalah pelaku informatika dapat berkarya sekreatif mungkin di bidangnya tanpa membutuhkan ruang dan fasilitas yang relatif mahal. Salah satu karya informatika yang cenderung populer di kalangan umum saat ini adalah *game* digital.

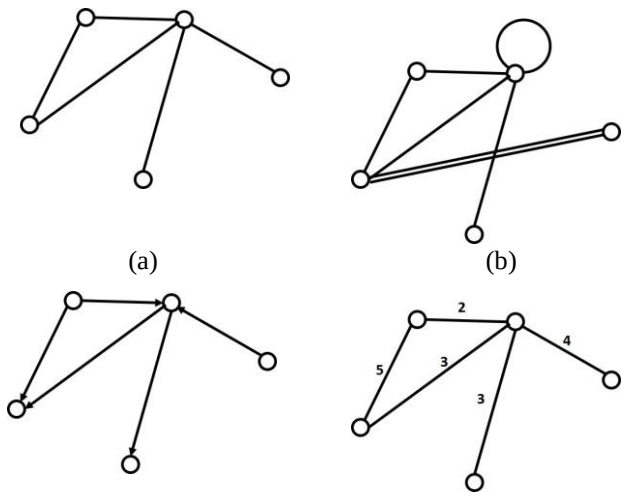
Hingga saat ini, teknologi konsol *game* digital terus berkembang. Munculnya konsol *game* baru membuka ruang baru bagi pelaku di bidang informatika untuk berkarya. Di antara karya-karya tersebut, ada yang berhasil mempertahankan popularitasnya dalam durasi yang lama, namun ada juga yang popularitasnya hanya bertahan dalam waktu singkat.

Diantara sekian banyak *game* digital yang popularitasnya bertahan lama, ada beberapa dapat bertahan karena tidak menimbulkan rasa bosan meskipun dimainkan berulang-ulang. Penyebabnya bisa bermacam-macam. Salah satunya adalah *multiple ending* yang terdapat dalam *game* tersebut. *Player* biasanya tertantang untuk mengetahui semua cabang story dan *ending* dari sebuah *game* yang memiliki *multiple ending*.

Pada *game* dengan *multiple ending*, *action player* yang berbeda akan membawa *player* ke *state game* yang berbeda. Pada *state* yang baru, hal yang sama juga berlaku. *Player* akan melakukan *action* hingga pada akhitnya mencapai suatu *state* yang merupakan *ending* dari *game*. Jika diasumsikan setiap *state* adalah titik dan setiap *action* adalah garis yang menghubungkan dua titik, akan muncul sebuah bentuk yang identik dengan graf. Oleh karena itu, graf dapat digunakan untuk merepresentasikan desain percabangan pada *game* dengan *multiple ending*.

II. GRAF

Graf merupakan bentuk representasi dari relasi beberapa objek. Pada graf, objek direpresentasikan dengan node dan relasi direpresentasikan dengan garis (sisi) yang menghubungkan dua titik. Setiap titik dapat terhubung melalui sisi dengan beberapa titik lain.



Gambar 1 Macam-macam graf (a) graf sederhana, (b) graf tidak sederhana, (c) graf berarah, (d) graf berbobot

Graf bisa dinyatakan sebagai pasangan antara himpunan node yang tidak kosong dan himpunan sisi yang menghubungkan node pada himpunan node (himpunan sisi boleh kosong).

Berdasarkan titik yang dihubungkan oleh sisi pada graf, ada dua jenis graf. Jenis graf yang pertama adalah graf sederhana. Pada jenis graf ini, tiap sisi menghubungkan dua node berbeda dan tidak ada dua sisi yang menghubungkan pasangan node yang sama. Jenis lainnya adalah graf tidak sederhana. Pada graf ini, ada sisi yang kedua ujungnya berada pada node yang sama atau dua sisi yang menghubungkan pasangan node yang sama. Sisi yang kedua ujungnya berada pada node yang sama disebut loop. Dua sisi yang menghubungkan pasangan node yang sama disebut sisi rangkap.

Berdasarkan arah sisinya, graf dibagi menjadi dua. Yang pertama adalah graf tidak berarah. Setiap sisi pada jenis graf ini merepresenttasikan relasi yang bersifat timbal balik. Pada graf ini, untuk setiap relasi R yang menyatakan m R n, relasi n R m juga dinyatakan dengan sisi yang sama. Graf lainnya disebut graf berarah. Pada graf ini, ada sisi yang merepresentasikan hubungan searah. Jika suatu sisi berarah menyatakan relasi m R n, maka relasi n R m hanya dapat direpresentasikan oleh sisi lain.

Berdasarkan bobot sisinya, graf dibagi menjadi dua, yaitu graf tidak berbobot dan graf berbobot. Pada graf tidak berbobot, setiap sisi memiliki tingkatan yang setara, tidak ada hubungan lebih-kurang antara dua sisi. Pada graf berbobot, tiap sisi selain mewakilkan relasi antara dua node, juga mewakilkan sebuah angka yang merupakan bobot sisi tersebut. Karena bobot tiap sisi tidak selalu sama, maka ada kemungkinan dua buah sisi memiliki hubungan lebih-kurang.

Sebuah node dan sebuah sisi disebut bersisian jika node tersebut terletak disalah satu ujung sisi yang bersangkutan. Dua buah node disebut bertetangga jika ada sisi yang bersisian dengan kedua node.

Rangkaian sisi yang menghubungkan dua node disebut sebagai lintasan. Dua node dikatakan terhubung jika ada lintasan yang menghubungkan kedua node tersebut.

Suatu graf disebut merentang jika untuk tiap pasangan node pada graf tersebut, selalu ada lintasan yang menghubungkan kedua node.

Jika tidak ada lintasan yang menghubungkan suatu nide dengan tiap node lain,maka node tersebut disebut node terisolasi.

Ada beberapa macam graf khusus yang cukup sering digunakan dalam aplikasi graf. *Tree* adalah salah satu jenis graf yang memiliki sifat merentang namun tidak memiliki sirkuit. Graf lengkap adalah graf sederhana yang setiap nodenya bertetangga dengan semua node lain yang ada pada graf tersebut.

III. PENERAPAN GRAF DALAM PERANCANGAN *GAME* DENGAN *MULTIPLE ENDING*

A. Representasi Komponen Dasar Penentu Ending dalam Graf

Komponen dasar *game* untuk menentukan *ending* ada dua, yaitu *state* dan *action*. *State* merupakan kondisi *game* yang mungkin. *Action* merupakan tindakan yang dilakukan user yang menyebabkan perubahan *state*.

State pada *game* memiliki beberapa sifat dasar. Untuk setiap *state* pada *game*, ada beberapa *action* yang mungkin dilakukan *player*. Untuk setiap *action* yang dilakukan *player*, *state game* akan berubah ke *state* lain. Ada kemungkinan setiap *action* yang berbeda akan membawa *game* ke *state* yang berbeda pula.

Node graf memiliki beberapa sifat yang mirip. Untuk tiap node pada graf, ada beberapa sisi yang dapat terhubung dengan node tersebut. Tiap sisi menghubungkan satu node dengan node lain. Karena kemiripan tersebut, setiap *state* pada *game* dapat direpresentasikan dengan node pada graf.

Action pada *game* memiliki sifat dapat mengubah *state game* ke *state* lain. Pada tiap *state*, ada kemungkinan banyak *action* yang dapat dilakukan berbeda. Sifat ini mirip dengan sifat yang dimiliki sisi pada graf. Sisi pada graf menghubungkan satu node dengan node lain. Banyak sisi yang bersisian dengan suatu simpul bisa berbeda dengan banyak sisi yang bersisian dengan simpul lain.

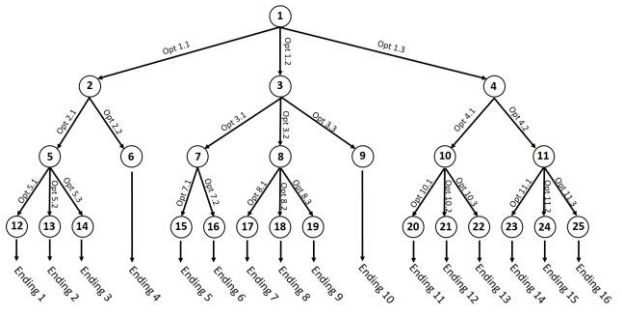
Karena kemiripan tersebut, *Action* dapat direpresentasikan sebagai sisi dalam graf. Namun, sisi graf bermacam-macam. Oleh karena itu beberapa sifat lain dari *action* harus dipertimbangkan.

Pada setiap *state*, tidak semua *action* dapat memindahkan *player* ke *state* yang sama.Oleh karena itu, tiap *action* yang bersisian dengan suatu *state* harus dapat dibedakan. Berdasarkan sifat ini, sisi yang bersisian dengan node harus dapat dibedakan. Oleh karena itu tiap sisi harus memiliki identitas. Identitas yang dimaksud bisa berupa penamaan maupun bobot sisi.

Pada *state ending*, *state* tidak akan dapat kembali ke *state* sebelumnya karena sudah mencapai akhir *game*. Dengan kata lain tidak ada *action* yang dapat dilakukan pada *state ending*. Pada beberapa *state*, jika anda berpindah dari *state* tersebut ke *state* lain, belum tentu ada *action* yang dapat mengembalikan *game* ke *state* sebelumnya. Dari sifat tersebut, dapat diketahui *action* tidak bersifat dua arah. Sifat yang harus dimiliki sisi untuk memenuhi sifat *action* tersebut adalah sisi harus berarah.

B. Penentuan Ending Dengan Tree

Contoh dari *game* jenis ini adalah *game* berjenis visual novel. Dalam *game* jenis ini, hampir tidak ada *action* yang sama pada setiap *state*. *Action* yang dilakukan *player* sebagian besar hanyalah memilih option yang telah disiapkan oleh *game*. Struktur dasar dari *game* jenis ini berupa percabangan yang disebabkan option yang ada. Karena setiap *state* memiliki *action* yang sangat berbeda dengan *state* lainnya, jenis graf yang paling sederhana untuk merepresentasikan *game* jenis ini adalah *tree* dengan setiap *state* direpresentasikan dengan satu node. Dalam perancangan *game* jenis ini menggunakan *tree*, akar dari *tree* merupakan *state* awal mulainya *game*. Daun dari *tree* merupakan *state ending game*.

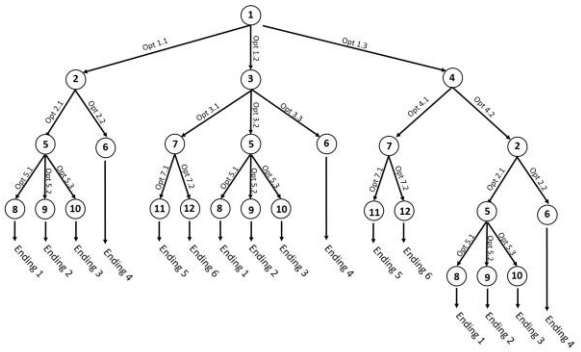


Gambar 2 Contoh sederhana *tree* rancangan *game*

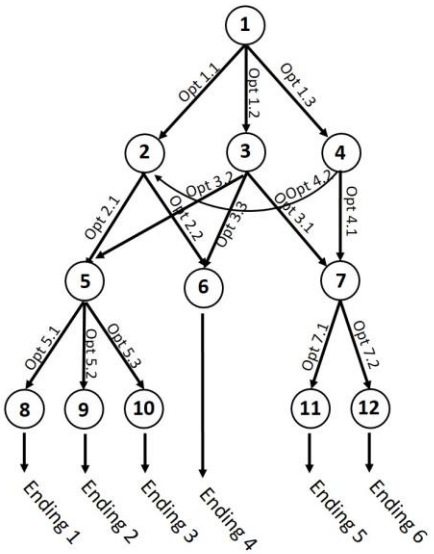
Kelebihan dari metode ini adalah memungkinkan menghasilkan banyak *ending*. Namun kelemahannya

adalah node yang terbentuk terlalu banyak. Memberikan perlakuan khusus untuk tiap node sangat sulit. Pembuat *game* juga harus membuat *ending* untuk tiap daun *tree*, dan jumlahnya sangat banyak.

Karena kelemahan tersebut, biasanya ada beberapa *state* yang digunakan beberapa kali. Beberapa cabang juga dibuat agar berakhir di *ending* yang sama. Hal ini menyebabkan *tree* yang terbentuk dapat disederhanakan dengan menggabungkan beberapa *state* yang muncul lebih dari sekali. Banyak *ending* juga dapat dibatasi dengan cara ini. Sisi yang terbentuk dari hasil penyederhanaan mungkin akan terlihat rumit dan saling berpotongan, namun dalam implementasinya pada program akan lebih mudah karena node-nya lebih sedikit.



(a)



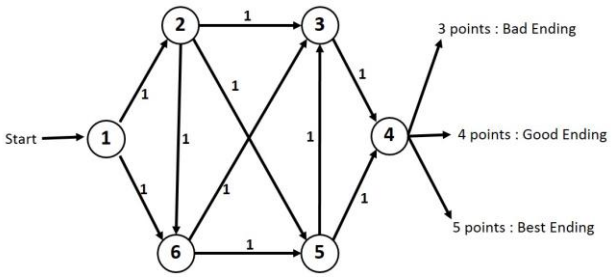
(b)

Gambar 3 (a) *Tree* yang bisa disederhanakan menjadi graf biasa dan (b) hasil penyederhanaannya

Meskipun hasil penyederhanaan *tree* bisa menjadi graf yang cukup rumit, *ending* yang perlu dibuat dapat dikurangi tanpa mengurangi variasi *action* yang mungkin dilakukan *player*. Agar penyederhanaan ini dapat dilakukan, dalam perancangan *tree*, sebaiknya tidak sering membuat node baru.

C. Penentuan Ending Dengan Bobot Action

Untuk *game* jenis ini, graf yang digunakan adalah graf berbobot. Untuk setiap *action* yang dilakukan *player*, bobot pada sisi graf yang bersesuaian dengan *action* diakumulasi. *Ending* dari *game* ditentukan berdasarkan total bobot yang telah didapatkan *player* dari *action* yang dilakukan. Kelebihan dari penentuan *ending* dengan metode ini adalah bobot dapat diatur sedemikian rupa sehingga *ending* tiap alternatif *action* sesuai dengan keingingan pembuat *game*. Dengan metode ini, node *ending* hanya sedikit, namun penentuan *ending* dilakukan setelah mencapai node *ending*.



Gambar 4 Contoh sederhana graf penentuan *ending game* dengan akumulasi bobot

Kelemahan dari metode perancangan *game* ini adalah ketika satu bagian graf diubah, semua *ending* akan terpengaruhi, sehingga harus dilakukan penyesuaian terhadap bagian graf yang lain. Selain itu, penentuan bobot sendiri mengharuskan perancang *game* untuk berpikir agar *ending* tiap rangkaian *action* sesuai dengan yang diinginkan.

D. Penentuan Ending Dengan Pembatasan Action

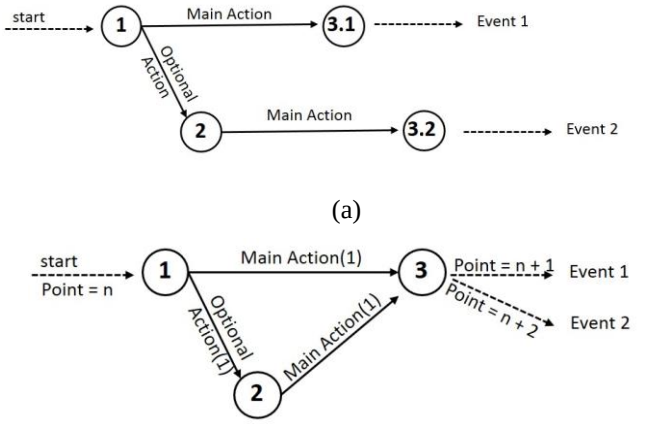
Pada penentuan ending jenis ini setiap state bisa menjadi state ending. Di awal *game*, pemain mendapat batasan untuk banyak action yang dapat dilakukan. Batasan dapat berupa jumlah action maupun jumlah bobot action. Ketika pemain sudah mencapai batasan dan tidak bisa melakukan action lagi, state yang ditempati saat itu yang menentukan ending yang diperoleh player. Banyak node yang dibutuhkan untuk *game* jenis ini bisa dijadikan seminimal mungkin, namun agar story dan event *game* tidak rusak, tiap node harus menyimpan semua event sesuai banyak kemungkinan urutan action.

IV. PENGEMBANGAN RANCANGAN LEBIH LANJUT

A. Penyimpanan History Action Dengan Menggunakan Bobot

Terkadang pada *state* yang sama, event *game* yang terjadi bisa berbeda tergantung pada *action* apa saja yang telah dilakukan pada *state* sebelumnya. Penanganan untuk hal ini biasanya mengharuskan munculnya percabangan di *state* sebelumnya agar perlakuan khusus dapat dilakukan untuk history *action* tertentu. Namun, jika menggunakan

graf berbobot, kita dapat meminimalkan percabangan *state* dengan menyimpan history *action* tersebut sebagai bobot graf.



Gambar 5 (a) Penyimpanan history *action* dengan cabang *state* dan (b) penyimpanan history *action* dengan bobot

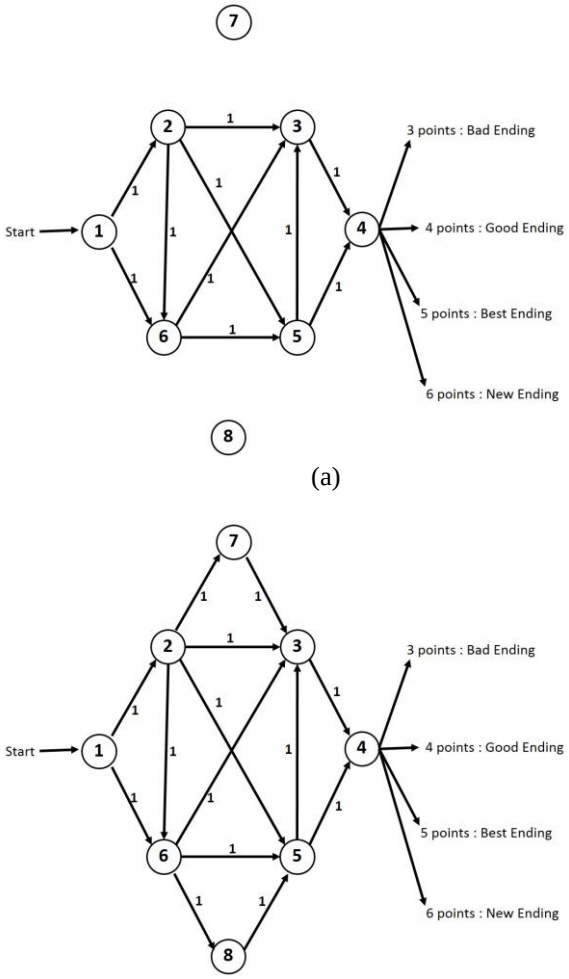
Jika penyimpanan history dilakukan dengan menggunakan percabangan *state*, node yang dibutuhkan akan menjadi lebih banyak dan hal ini akan menyebabkan mplementasi graf pada program lebih sulit. Jika penyimpanan history dilakukan dengan menggunakan bobot graf, setelah terjadi pencabangan, kedua cabang tersebut akan segera bergabung kembali ke *state* yang sama. Lalu pada *state* tersebut, event dieksekusi tergantung pada bobot graf yang baru dilewati.

Pada alternatif pertama setiap node menyimpan satu event, namun ada kemungkinan banyak *state* mirip yang haris diduplikat. Pada alternative kedua, satu *state* bisa menyimpan lebih dari satu event, namun cabang yang muncul segera bergabung lagi dengan cabang utama.

Meskipun perancangan graf berbobot cenderung lebih sulit, pengelolaan alternatif kedua cenderung lebih mudah karena pemetaan *state* dan pemetaan event dikelola secara terpisah dan cenderung tidak mempengaruhi satu sama lain.

B. Pemasangan Ending Baru

Biasanya setelah mencapai *ending* dari suatu *game*, ada fitur yang disebut New *Game* +. Fitur ini memungkinkan *player* untuk mengulangi permainan dengan beberapa tambahan didalamnya. Bentuk tambahan tersebut bisa bermacam-macam. Salah satu bentuknya adalah munculnya *ending* baru. Untuk representasi *game* dalam graf, hal ini tentunya mengharuskan adanya penambahan node dan sisi.



Gambar 6 (a) kondisi peta *state* sebelum New *Game* + dan (b) kondisi peta *state* setelah New *Game* +

Untuk mempermudah hal ini, node baru dapat disiapkan sejak awal sebagai node terpencil, lalu ketika *player* menggunakan fitur New *Game* +, yang perlu ditambahkan hanyalah sisi. Event *ending* baru juga bisa diperlakukan demikian. Sejak awal *ending* tersebut bisa dibuat ada, namun hanya bisa tercapai jika node baru sudah muncul. Dengan begitu akan tampak seolah-olah *ending* tersebut merupakan *ending* baru.

Pada Gambar 6 (a), node 7 dan 8 sudah ada, namun belum bisa diakses. New *ending* juga sudah ada, namun tidak mungkin tercapai karena node 8 belum bisa diakses. Pada Gambar 6 (b), muncul sisi baru yang menghubungkan graf lama dengan node 7 dan 8. Graf ini adalah graf setelah *player* masuk ke New *Game* +. Dengan munculnya sisi ini, New *Ending* pun dapat dicapai oleh *player*.

C. Penghapusan Node

Pada *game* dimana player benar-benar memiliki kebebasan untuk memilih urutan state, untuk mencegah pengulangan event pada state yang sama, penghapusan state terkadang diperlukan. Namun penghapusan state

berarti juga harus menghapus semua action yang berhubungan dengan state tersebut. Untuk mempermudah penghapusan, node tidak perlu benar-benar dihapus. Yang diahpus cukup sisi yang menuju ke node tersebut. Dengan penghapusan ini, node yang bersangkutan tidak akan bisa dilalui player lagi. Namun dalam penghapusan node ada hal yang perlu diperhatikan. Terhapusnya node tidak boleh menghalangi pemain untuk mencapai ke state ending. Jika hal ini terjadi, pemain tidak akan bisa mencapai ending dari *game*. Hal ini bisa dimasukkan ke kategori bug.

V. SIMPULAN

Salah satu cara untuk merancang *game* dengan *multiple ending* adalah dengan menggunakan graf. Perancangan menggunakan gaf tak berbobot cenderung lebih mudah namun node graf menjadi lebih banyak. Perancangan dengan menggunakan graf berbobot lebih sulit, namun node yang terbentuk lebih sedikit.

VI. TERIMA KASIH

Puji Syukur penulis ucapkan kepada Tuhan Yang Maha Esa karena hanay atas rahmat-Nya makalah ini dapat terselesaikan. Ucapan terima kasih juga penulis ucapkan kepada Bapak Dr. Ir. Rinaldi Munir, M.T. dan ibu Dra.Harlili S., M.Sc. selaku dosen pembimbing mata kuliah IF2120 Matematika Diskrit yang telah memberikan ilmu kepada penulis yang merupakan dasar dari penulisan makalah ini.

REFERENSI

[1] Munir, Rinaldi. 2006. *Diktat Kuliah IF2120 Matematika Diskrit*. Bandung.

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, 11 Desember 2015

Cendhika Imantoro - 13514037